# Conception et étude d'une architecture numérique de haute performance pour le calcul de la fonction exponentielle modulaire

A. Bernal

**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

|_|_|_|_|_|_|_|_|_|_|

*THÈSE*

pour obtenir le grade de **DOCTEUR**

de **l'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

***Discipline***: **Microélectronique**

présentée par

**Álvaro BERNAL NOREÑA**

le 22 octobre 1999

*Titre*

# Conception et Etude d'une Architecture Numérique de Haute Performance pour le Calcul de la Fonction Exponentielle Modulaire.

***Directeur de thèse***

**M. Alain Guyot**

**JURY**

| | |
|---|---|
| M. Jean-Claude Bajard | Président |
| M. W.A.M. Van Noije | Rapporteur |
| M. Habib Mehrez | Rapporteur |
| M. Bertrand Hochet | Examinateur |
| M. Alain Guyot | Examinateur |

Thèse préparée au sein du Laboratoire TIMA-INPG à Grenoble

À Margarita.

*L'homme, un dieu quand il rêve,*
*juste un mendiant quand il pense.*

*Hölderlin Hyperion.*

# Remerciements

Ce travail a été réalisé dans le laboratoire TIMA de l'Institut National Polytechnique de Grenoble dirigé par M. Bernard Courtois, que je tiens à remercier pour son accueil au sein de ce laboratoire.

Je veux également remercier plus particulièrement M. Alain Guyot, Maître de Conférence à l'ENSIMAG et directeur de cette thèse, pour ses conseils et sa coopération.

M. Jean-Claude Bajard pour avoir accepté la présidence de mon jury de thèse, et également Messieurs W.A.M. Van Noije et Habib Mehrez pour avoir accepté d'être rapporteurs. Par la même occasion je remercie M. Bertrand Hochet. D'une manière générale, je remercie l'ensemble des personnes que m'ont fait l'honneur d'être membres du jury.

Je tiens à remercier tous mes collègues de travail et tout l'ensemble du personnel de TIMA pour leur accueil mais aussi pour l'ambiance chaleureuse qui règne au sein du laboratoire.

Egalement Messieurs Kholdoun Torki, Jean-François Paillotin, Alejandro Chagoya, Robin Roland pour leur aide dans la réalisation des circuits, et pour l'assistance logicielle et test.

J'adresse mes remerciements à l'Université del Valle et COLCIENCIAS, pour avoir rendu possible la réalisation de ce travail et notamment pour ce qui concerne la partie administrative. Je tiens particulièrement à remercier Mlle. Blanca Stella Barona et Mme. Corinne Durand-Viel pour leur aide logistique et pour le soutien émotionnel qu'elles m'ont aussi apporté.

Evidemment je ne pourrais pas terminer ces remerciements sans citer mes amis et ma famille, et tout particulièrement Margarita pour sa présence constante.

# Résumé

Au cours de ces dernières années, on assiste à un développement des communications régies par des réseaux d'ordinateurs et systèmes électroniques; l'extension importante de ces types de moyens électroniques de communication et d'échange d'information met en évidence les besoins de sécurité des canaux de transmission.

Les processus de sécurisation d'information privée reposent principalement sur des protocoles qui utilisent le concept de fonction à sens unique ou fonction très difficilement inversible. Ce type de brouillage mathématique est fréquemment réalisé par la fonction exponentielle modulaire. A l'heure actuelle une partie importante de ces protocoles est orientée à l'utilisation des algorithmes programmés. Dans ce travail se présente la conception d'une architecture performante qui satisfait aux caractéristiques les plus importantes afin de garantir la viabilité d'un circuit intégré pour le calcul de la fonction exponentielle modulaire. L'analyse de l'architecture permet d'évaluer les gains en vitesse qu'une réalisation matériel pourrait permettre par rapport aux algorithmes programmés.

L'architecture calcule la fonction exponentielle modulaire des numéros représentés en notation modulaire en combinant les avantages de l'algorithme de Montgomery pour la multiplication et ceux de la méthode généralise de multiplications répétées, pour l'exponentielle. A partir de ces résultats deux prototypes ont été dessinés, fabriqués et vérifiés en utilisant une technologie AMS-CMOS de 0,6 µm. L'architecture présente une bonne performance et une modularité qui permet d'élargir le nombre de bits des chiffres a être calculés.

D'autre part, la nécessité de performances élevées inhérentes aux applications, soit en cartes à puce soit en communication par satellite, a fait considérer le AsGa comme une technologie appropriée pour l'implémentation de ce type de système. La conception de deux des principaux blocs de l'architecture envisageant la basse consommation a été aussi réalisée.

Mots clés: Cryptographie, Architectures d'Arithmétique Modulaire, Exponentiel Modulaire, Arséniure de Gallium.

**Présentation Etendue de la thèse**.

---

Conception et Etude d'une Architecture Numérique de Haute Performance pour le Calcul de la Fonction Exponentielle Modulaire.

---

Table de Matières.

# Abstract

During these last years, the development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and excursions with telecommunications. The significant extension of these types of electronic means of communication and exchange of information highlights the needs for security of the data communication channels. Secret digital writing is being used to avoid message transformations. Techniques to avoid eavesdroppers actions are known as cryptography.

Currently, cryptosystems are more frequently required in applications as remote cash dispensers, high speed computers terminals, authentication, digital signatures and private communication between others. The cryptography processes rest mainly on protocols which use the concept of one-way function. At present a significant part of these protocols is directed to be used as programmed algorithms.

The type of mathematical jamming is frequently carried out by the modular exponentiation function. In this work, the design of an alternative architecture which satisfies the most significant characteristics in order to guarantee the viability of an integrated circuit for calculating the modular exponentiation function is presented. In the architecture, the main advantages of both, generalised square-multiply binary for exponentiation function and the Montgomery's algorithm for modular multiplication are mixed. The architecture is oriented to compute the modular exponentiation of large integer numbers. Two prototypes were designed, fabricated and tested to validate the architecture, which presents a good performance and a modularity being easily expandable to larger bit-widths.

In addition, as several of cryptography applications use satellite communication where high performances but principally radiation tolerant integrated circuits are needed, AsGa become as a suitable technology for the implementation of this type of system. The design of two of the principal blocks of the proposed architecture considering low power strategies consumption are also presented.



Keywords: Modular Arithmetic architectures, Modular Exponentiation, Gallium Arsenide.

## I. Table of contents

## II. List of figures

**Chapter 1.**



**Chapter 2.**



**Chapter 3.**



**Chapter 4.**

**Chapter 5**

**Chapter 6.**



**Chapter 7.**

## III. List of tables

### Chapter 1.



### Chapter 2.



### Chapter 3.



### Chapter 4.

**Chapter 5.**



**Chapter 6.**



**Chapter 7.**

# 1. Introduction.

## 1.1. Cryptography.

In the electronic age, information exchange that could benefit persons or groups can also be used against such groups or individual persons. Industrial espionage among highly competitive businesses often requires that extensive security measures be put into place. And, those who wish to exercise their personal freedom may also wish to encrypt certain information. Cryptography is the art or science of secret writing, or more exactly, of storing information, is the art of encoding data in a way that only the intended recipient can decode it, and know that the message is authentic and unchanged.

There are many reasons for using encryption techniques. Different applications that require privacy, trust and access control, like electronic money, secure communications, passwords, and many others, should all use strong encryption methods when possible.

A cryptosystem is a method to accomplish that. The ideal cryptosystem would be an applied specific system for one particular purpose, which would satisfy the requirements of security, reliability and ease-of-use : reliability means that the cryptosystem, when used as its designer intended it to be used, will always reveal exactly the information hidden when it is needed. Security means that the cryptosystem will in fact keep the information hidden for all those persons intended to crack the system. Cryptanalysis is the practice of defeating such attempts to hide information. Cryptology includes both cryptography and Cryptanalysis.

The security of a cryptosystem is always relative to the difficulty of breaking a secret message and the conditions under which it will be used.
In general, the security of a cryptosystem can only be measured by its resistance to actual attempts to break it in practice. Those that have resisted the attentions of many cryptanalysts for many years may be deemed secure, at least until better methods of Cryptanalysis are invented.

### 1.1.1. Practical Cryptosystems.

Still, the methods of data encryption and decryption are relatively straightforward, and easily mastered. A cryptosystem is designed considering that decryption can be accomplished only under certain conditions, which generally means, only by persons in possession of both a decryption engine and a particular piece of information called the decryption key, which is supplied to the decryption engine in the process of decryption. All modern algorithms use a key to control the encryption and decryption. The message can only be decrypted if the key matches with the key used to encrypt it. The key used for decryption can be different from the key used in encryption, and this divides the algorithms in *symmetric* (or *secret-key*) and *asymmetric* (or *public-key*) classes.

*Symmetric* algorithms, also called *secret-key* algorithms, use the same key for both encryption and decryption. The key is not to be leaked to outside enemies, should be changed often and be sufficiently random. Different symmetric algorithms use different length keys, usually a longer key means higher security. Symmetric algorithms are generally faster than asymmetric ones and use a much shorter key.

*Public key* systems were developed in the 1970s to solve the problem of secure key exchange. In this system the decryption key is not the same as the encryption key. Such public key systems can, if used properly, go a long way toward solving the problem of secure key exchange because the encryption key can be given out to the world without compromising the security of communication, provided that the decryption key is kept secret.

Although public key cryptography in theory solves the problem of secure key exchange, it does in general have a couple of disadvantages compared to symmetric (or secret) key systems. The first is speed. Generally public key systems, such as PGP, are much slower than secret key systems, and so may be suitable for encrypting small amounts of data. The second disadvantage of public key systems is that there is a problem of key validation.
There are numerous public key cryptosystems, the most well known being the one based on the RSA. Messages ciphering and digital signature are two of the most extended cryptography applications.

The idea behind public key encryption *messages ciphering*, is that it is computationally infeasible to calculate the secret key from the public key and that no information can be obtained about the secret key from any message by knowing the public key.

*Digital signatures* are a way of signing data in the same way that we sign documents today. Digital signatures cannot be forged by someone else; the signature is proof that the signer

signed the message; the signature is an integral part of the message and cannot be transferred to another message; the signed message cannot be changed in any way without being detected; the signer cannot deny signing a message after doing so.

In some cases it is possible to show that cracking a cryptosystem is equivalent to solving some particular mathematical problem. Most implementations of public key cryptography rely on the hard problems of factoring large numbers, it means numbers with several hundred decimal. Whereas it is relatively easy to multiply two large primes, it is currently very difficult to factor the result back to the two original primes. In the case of RSA algorithm, it is widely believed that these are secure if and only if the problem of factoring large numbers is insoluble, that is, computationally infeasible in real time.

There are few operations in mathematics that are truly 'irreversible'. In the case of the RSA encryption algorithm, it uses very large prime numbers to generate the public key and the private key. Although it would be possible to factor out the public key to get the private key, the numbers are so large as to make it very impractical to do.

Modern cryptographic algorithms are meant to be executed by computers or specialised hardware devices for which there are several different cryptographic algorithms and methods which rely for its security on the difficulty of factoring large numbers. Traditionally, several methods can be used to encrypt data streams, all of which can easily be implemented through software, but not so easily decrypted when either the original or its encrypted data stream are unavailable.

Although several methods are developed to be implemented through software, some algorithms and protocols have been oriented to hardware implementation, more specifically, to smart cards applications. As known, smart cards market is each time more important. The U.S. market has lagged behind Europe and Asia in using smart cards. The companies concluded that several factors must be combined to push smart cards successfully into the U.S. market. They included greater interoperability and the ability to use the chip cards for unattended needs such as telephones, parking meters, and transit systems. The smart card is currently used mostly in Europe where it is used as a pay phone calling card or for vending machines. The United States is beginning to see smart cards in use for GSM phones, laundry, and vending applications.

Smart cards are plastic cards with a credit card size, that have an embedded computer chip. The card companies are eager to move to smart cards from today's standard magnetic stripes on the back of cards so that more data or applications can be loaded onto cards. Multifunction cards include more than one use, for example, loyalty or frequent-user programs on the same card with e-cash, credit, and debit uses. The magnetic stripe works by encoding an identification on

a magnetic tape similar to how a computer writes information onto a floppy disk. This method, though powerful, has proved to be insecure in many instances.

Magnetic cards are easy to reproduce and many use no form of encryption on their identifier. The smart card achieves this because the card has a small yet powerful computer built into it. This computer allows the card to interact with the card reader, not just pass information to it. Just as there are many different uses for a smart card there are many different flavours of smart cards not to be confused with optical memory cards.

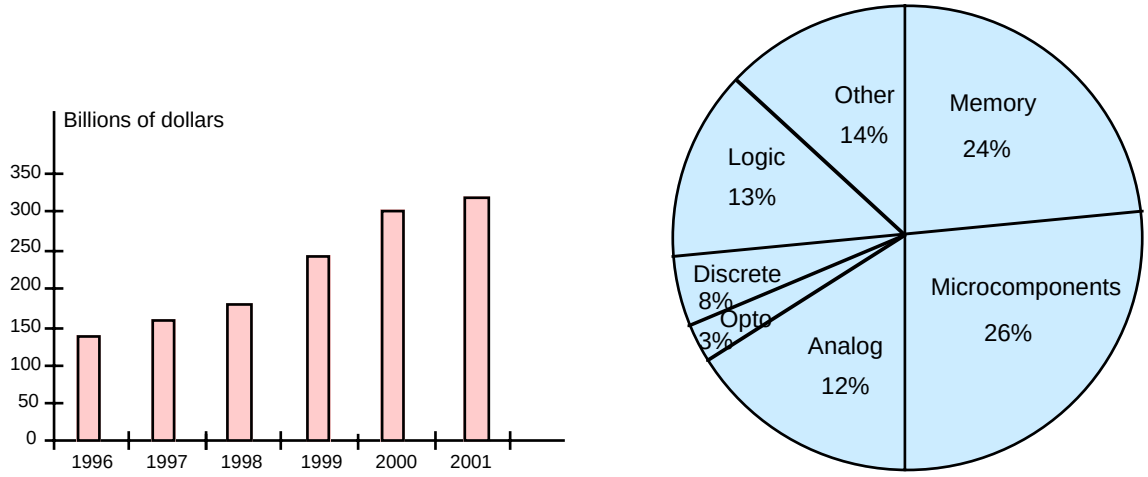
One of the key benefits of smart cards is the ability for some cards to support on board cryptography. Cryptographic smart cards open up a whole new realm in information security because it now allows a secure place for storing of keys and key rings. By doing the actual cryptography on the card, the keys never have to leave their storage place. This gives the card holder a secure way of storing keys especially if the key pair was generated on the card. Smart cards performing cryptographic functions can be utilised in applications such as key and certificate verification, encryption, and random number generation between others.

Although the uses for smart cards are numerous, there is still the cost issue. Magnetic stripe cards cost as little as 6 - 8 cents to be made, whereas a smart card can cost up to 10 to 15 times that cost. For this reason, small size architectures without degradation of performance is very attractive for this specific application. Modular exponentiation is the operation most widely used in many protocols and algorithms. The design of the chip for performing modular exponentiation based on regular and small architectures would allow to implement this function on smart cards.

Considering that several cryptographic applications require high performance systems, low power strategies applied to high speed technologies as GaAs must be considered.

### 1.2. Gallium Arsenide Technology.

Nevertheless, the world-wide semiconductor market in 1996 down 6.2 percent from 1995's, this market is still growing and will surpass the $ 300 billion of dollars point in the year 2001. Semiconductor market analyst explain that the drop was due in large part to the bottom falling out of the DRAM market in late 1995 and throughout 1996. In 1996, the world-wide semiconductor market achieved at $ 141.7 billion of dollars. European semiconductor market consumption revenue achieved at $ 28.5 billion and is expected to grow to $ 62.1 billions by the year 2001. As can be seen from the Figure 1.1, where both revenue forecast and revenue by product are presented, the expectations of growth are attractive.




(a) Revenues forecast. (b) Revenue by product.

Figure 1.1. World-wide semiconductor market. [1]

Silicon MOS technology has been the main medium for computer and system applications for a long time and this technology will continue to fill this role. However, in Silicon MOS technology several limitations are already becoming apparent in state-of-the-art fast digital systems [2], due to the fact that, system level requirements quickly surpassed the performance that silicon was able to deliver. Since a few years, in order to overcome these limitations some developments in silicon technology have been achieved.

However, with the development of communications and more specifically portable telecommunication and multimedia systems which require high clock frequency, logic families as BiCMOS, GaAs and SiGe are becoming more attractive to those types of applications. For that reason, some parallel significant advances are also beginning to take place with Gallium Arsenide technologies.

Gallium Arsenide (GaAs) MESFETs became an enabling technology that allows overcome the silicon limitations in ultra high speed applications. This technology has evolved and changed over the last 30 years and has finally found its marketplace into the semiconductor's industry. Nevertheless, some efforts are currently done in order to develop other technologies, GaAs technology continues to play an important role in communication applications, such as: compact cell phones, high frequency wireless base stations and global positioning systems (GPS) [3].

Gallium Arsenide’s resurgence also stems from an ever-expanding profusion of applications, such as medical [4], analogue cellular/PCS handsets, digital cordless handsets, wireless local loop [5], wide band CDMA, automotive [6] and radar communication (IMT 2000 system). Analogue and discrete GaAs semiconductors continue to be a very important segment in digital

communications system because of the peak power, supply voltage and signal distortion requirement.

Since a few years, digital GaAs applications have emerged in the form of one company which has become the market leader: Vitesse Semiconductor. Vitesse has managed important growth in an area in which digital GaAs is again an enabling technology. The company, whose process technology is said to address the high-speed needs of telecom and datacom, reported in the third quarter of 1998 revenue of 46.1 millions of dollars, up 67% from last year and up 15% from second quarter of 1998 [7].
Currently, the use of compilers for digital GaAs IC design using Vitesse technology is available [8]. Another data-path compiler for the public domain ALLIANCE CAD System has been also developed [9].

The market for GaAs beginning 1997, started growing at 50% to 60% a year [10]. Digital GaAs market reached over $1 billion in 1996. On the other hand, the GaAs wafer industry is expected 8 MSI (million square inches) by the year 2000. Revenue is expected to increase from $ 153 million in 1996 to over $ 400 million in 2000 [1].

The perceived European GaAs IC market from 1984 to 1994 is shown in figure 1.2. The leading sector until the late 1980's was analogue MMIC, but that both digital and optoelectronic ICs will be employed increasingly in systems. The market for digital GaAS integrated circuits in Europe increased from US $ 58.8 M in 1989 to US $ 1.088 B in 1994.

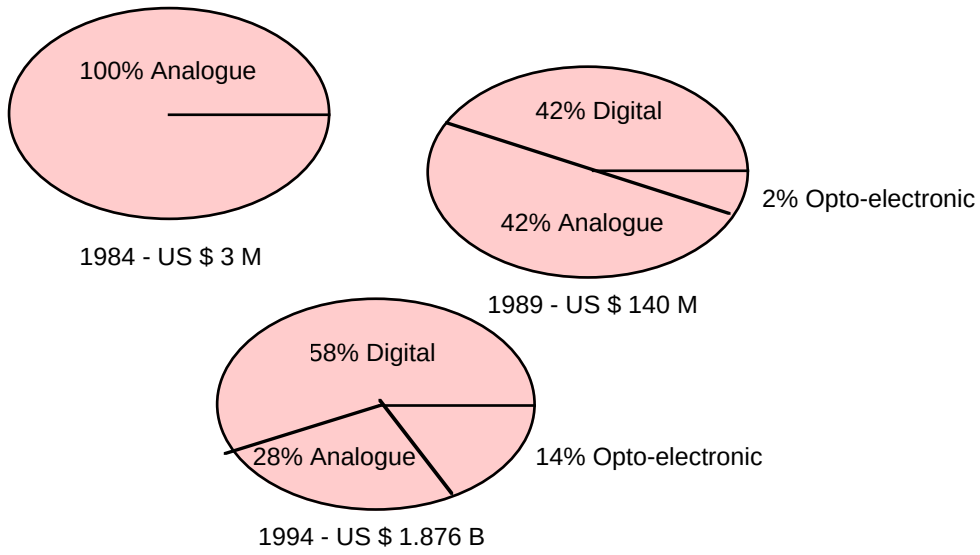



Figure 1.2. GaAs ICs European market sectors [11].

Top four U.S. producers: Anadigics, TriQuint, Vitesse, Motorola and their homologous Japanese are the main GaAs ICs industries, corning the GaAs IC world market. In 1992, the top seven European producers of GaAs devices: Alenia, Alcatel, Daimler-Benz, GEC Marconi Materials Technology, Philips Microwave Limeil, Siemens and Thomsom-CSF created into the Eureka Program the EuroGaAs initiative which was oriented to penetrate the GaAs ICs world market.

In 1995, the European GaAs ICs production represented only 24% of the global market but its sales achieved hardly 10% of world market. The evolution of the captive market for GaAs ICs is shown in table I. As can be seen, the market is growing very fast. Specifically for digital GaAs ICs the market will be increased from 71 millions of dollars in 1991 to 681 millions of dollars in 2000.

Table I. Market of GaAs devices from 1991 to 2000 [12].

| **Millon of dollars** | **1991** | **1996** | **2000** |
|---|---|---|---|
| Digital ICs GaAs | 71 | 324 | 681 |
| MMIC GaAs | 104 | 289 | 643 |
| Discrete GaAs | 176 | 237 | 341 |
| Total | 351 | 850 | 1665 |

However, having discussed the potential of GaAs technology, the performance of GaAs integrated circuits with reasonable complexity must be also considered. GaAs complex systems perform better in terms of propagation delay but not in terms of power dissipation. Power consumption has become a critical concern in both high performance and portable applications.

Over the past years, much effort has been directed towards increasing the speed of digital integrated circuits and decreasing the area size. Only in recent times the power consumption of these circuits has been considered as a third constraint during their development. Currently, the researching in high speed VLSI design has been shifted from high speed to low power emphasis due to the proliferation and rapidly growing range of portable electronic systems containing microelectronic devices [13]. This factor has forced a new definition of priorities and considerations of design.

### 1.2.1. Low Power GaAs Circuits.

Because of GaAs technology requires also very low power dissipation a large effort has been spent in the development of both efficient low-power GaAs technologies [14], [15] and high speed low power GaAs logic styles [16], [17] which should allow the extension of this attractive technology to high speed low power applications.

Several low-power GaAs circuits techniques concerning reductions in charging capacitance, operating voltage, static current and leakage current [18], [19], [20], [21], [22], [23], [24]

have been published obtaining a reduction in active global power dissipation. In each case, the speed-power product has been improved.

Methods for solving the power dissipation drawbacks in GaAs technologies are being strongly studied. The high electronic mobility and low knee voltages of GaAs are ideal for low supply voltage operations [25]. Low voltage operation has already been one of the most important design issues for GaAs circuitry not only to further reduce power dissipation , but also to ensure reliability for devices. In parallel, some important progress in power reduction, performance and temperature tolerance in several GaAs complex systems have been obtained.

So, Low Power GaAs LSI technology is an attractive researching field in which considerable attention is being focused. Because this is an important and growing area of electronics, in 1995, Motorola has developed a self aligned complementary GaAs technology for Low-Power, high speed digital and mixed mode applications. The complementary GaAs (CGaAs) shows a speed power performance of 0.01µW/MHz/gate at 0.9V in digital circuits [14].

All mentioned and recent exploratory achievements in the movement towards low power operation seemingly give promise of future improvements. This new researching field promises to satisfy the speed requirements of present day computers and indeed the super computers. Low Power Gallium Arsenide technology will not displace silicon but may be used in conjunction with silicon to satisfy the need for Ultra High Speed Integrated Circuits (UHSI).

The early years of mobile communication were based on first generation analogue systems, such as NMT/TACS/AMPS, the development of which were regionally based in Europe and the USA. However, worlwide there is a steady migration underway towards second generation digital systems, driven largely by the need for increased capacity.

In Europe the migration to digital technology is based on the GSM standard which was launched in 1992 and was rapidly adopted in Africa, the Middle East and the Asia Pacific Region. DCS 1800 system is based on the same protocol as GSM but at twice the carrier frequency. In the USA the move to second generation digital systems started using one of the two competitive digital standards: TDMA (Time Division Multiple Access) or CDMA (Code Division Multiple Access).
The different services to be offered following introduction of any digital communication system included: short message service, calling line identification, conference calls, high speed data and others. Additionally, there are some extra benefits such as encryption and the ability to provide a portfolio of data related services based on digital technology [26]. VLSI circuits that accelerate the encryption and decryption of messages using the RSA encryption technique and circuits capable of performing long word length modulo multiplication at very high speed attract much

interest for cryptography applications. For that reason, designer who research to accelerate RSA cryptographic processing are looking the high speed advantages of Gallium Arsenide VLSI technology as an interesting alternative.

### 1.3. Objectives

Cryptographic methods such as encryption and decryption process and other secret communication problems require the exponentiation arithmetic function to hidden information. Exponentiation arithmetic function is executed as repetitive multiplication leading to the long word length modulo multiplication operation to be the main and more frequently function to be performed. Arithmetic operators exhibit in general a great activity and dissipate consequently a significant share of the power supplied to a circuit. That is specifically true for a multiplier which dissipates much more power than an adder when activated due to the fact that its design or layout structure is not as regular as an adder.

Considering that, the ultimate performance of an integrated circuit can be substantially improved by using a full customised macro cells library for its design (being that particularly true for GaAs circuits where speed performance is critical, the cost of real estate is high, and design expertise is scarce), the goal of this work is focused on developing an alternative architecture (in CMOS technology) for executing modular exponentiation which must satisfy the requirements of speed, power consumption and size for smart card implementation.

Also, low power design GaAs strategies to be used in arithmetic macro blocks implementation are considered. These structures are conceived to be applied in an eventual Low Power GaAs Cryptosystem using the same architecture. Up to now there is no arithmetic macro blocks to complete the development of Low Power GaAs VLSI cryptosystems. Typical delay, average power consumption and area for each function are the principal features to be characterised.
Compacts and high speed designs combined with new low power strategies which take advantage of superior performance of GaAs are proposed looking predominantly the power reduction constraint as a principal goal. In order to minimise the power of these ICs, different low power methodologies are applied at different abstraction levels of the system design.

Two specific performance parameters which must be improved are consumption and operating voltage. A reduction in current consumption and operating voltage allows to obtain a significant reduction of power dissipation. The significance of improvement in these two parameters is the main motivation.

This work is mainly focused on the reduction of both complexity and power consumption of a cryptography system in CMOS and GaAs technologies. Several chips were designed

considering low power strategies, in order to verify the power reduction that can be achieved. In order to minimise the power of these ICs, different low power methodologies will be applied at different abstraction levels of the system design. So, in table II, are shown the five distinct levels where optimisation techniques must be implemented.

Table II. Low Power Hierarchical Design.

| |
|---|
| System design |
| Algorithms |
| Architecture design |
| Circuit design |
| Process technology |

This work will contain 7 chapters. Chapter 2 presents an introduction of cryptography and modular notation concepts which will be used along this work. Chapter 3 shows an alternative architecture for executing modular multiplication, as well as, simulation and experimental results of the prototypes.

Chapter 4, presents the conceived architecture for calculating modular exponentiation which is based on multiplier proposed architecture; performance and simulations are discussed. Those chapters describe in general the internal architecture used to compute modular exponentiation which is the core operator of the cryptosystem. It could be also used to implement the core of a Low power GaAs cryptosystem.
Chapter 5 discusses some important characteristics of GaAs technology and comparison on silicon, the general concepts which will be used for implementing a low power GaAs functions as well as the low power techniques used in each design.

Cache-memory play an important role in the overall power-efficiency of a cryptographic system, since it can reduce the data traffic between the arithmetic operator and external memory. For this reason, we study in chapter 6 a novel Low Power GaAs memory cell. So, as on-chip memory accesses consume significantly less energy than accesses to off-chip memory, this cell is appropriated to implement cache high speed memories. Chapter 7 analyses two asynchronous structures to design low power GaAs arithmetic circuitry. Both approaches were verified through two full custom eight bits ripple carry adders obtaining significant power consumption reduction.

The low power functions will be designed with Vitesse III technology (0.6 µm - GaAs) using the design kit of Vitesse on Cadence environment. All experimental measures were done at CIME Test Department. The test chips were fabricated through CMP Service. Part of the research was supported by GARDEN (Galium Arsenide Reliable Design Environment) ESPRIT project CT93-0385.

## 1.4. References


[1] E.J. Lum, "GaAs semiconductors: New market opportunities and emerging applications trends.", Invited paper, Proc. IEEE 5th European Gallium Arsenide and related III-V compounds Applications Symposium, Bologna, Italy, Sept., 1997.

[2] K. Eshraghian, "Gallium Arsenide Integrated Circuits Design", Internal Report, Electronics Laboratory of the Swiss Federal Institute of Technology, Lausanne, Switzerland, April, 1989.

[3] E. Fishkill, IBM Eyes Merchant Packaging Services, Hebdo, Electronic Engineering Times, N.Y., July 13, 1998.

[4] K. Carr, “Use of Gallium Arsenide in Medical Applications”, IEEE GaAs IC Symposium, USA, 1995.

[5] M. Mitama, "Mobile Communications Systems Trend in Japan and Device Requirements", IEEE GaAs IC Symposium, USA, 1995.

[6] A. Colquhoun, H. Meinel, “Automotive Applications of GaAs Components”, Proc. IEEE European Gallium Arsenide and related III-V Compounds Applications Symposium, Paris, France, 1996.

[7] Margaret Quan, Motorola, AMD Post Losses, Hebdo, Electronic Engineering Times, Finance, pps. 78-79, July 13, 1998.

[8] R. Oettel, “The use of Compilers for Digital IC Design”, IEEE Gallium Arsenide IC Symposium, USA, 1993.

[9] O. Beaurin, A. Amara, “A GaAs Data-Path Compiler”, Proc. IEEE European Gallium Arsenide and related III-V Compounds Applications Symposium, Paris, France, 1996.

[10] L. Armstrong, O. Port, S. Brull, “GaAs Guzzlers on the Info Highway ?, Science and Technology, Business week , August 19, 1996.

[11] "Critical Design Issues for GaAs VLSI Circuits", Internal Report, Microelectronic Center, Middlesex University, London, UK, June, 1991.

[12] L. Stéphan, "L’Europe devient compétitive en circuits GaAs ", Hyper 97, Electronics International, Hebdo, No. 248, January, 1997.

[13] A.P. Chandrakasan, R.W. Brodersen, "Low Power Digital CMOS Design", Kluwer Academic Publishers, 1995.

[14] B. Bernhardt, M. LaMacchia, J. Abrokwah, J. Hallmark, R. Lucero, B. Mathes, B. Crawforth, D; Foster, K. Clauss, S. Emmert, T. Lien, E. Lopez, V. Mazzota, B. Oh, “Complementary GaAs (CGaAs): A high performance BiCMOS Alternative”, IEEE GaAs IC Symposium, USA, 1995.

[15] H. Fawaz, J.F. Thiery, N. Linh, F. Mollot, J. Pesant, M. Francois, M. Muller, E. Delos, G. Salmer, “III-V Complementary HIGFET technology for low power microwave and high speed/low power digital integrated circuits”, Proc. IEEE European Gallium Arsenide and related III-V Compounds Applications Symposium, Paris, France, 1996.

[16] A. Chandna, R. Brown, D. Putti, C.D. Kibler, "Power Rail Logic: a Low Power Logic Style for Digital GaAs Circuits. IEEE Journal of Solid-State Circuits. Vol.30, No.10, October, 1995.

[17] K.R. Nary, S. Long, "GaAs Two-Phase Dynamic FET Logic: A Low Power Logic Family for VLSI”, IEEE Journal of Solid-State Circuits. Vol.27, October, pp. 1364-71, 1992.

[18] R. Kanan, B. Hochet, M. Declerq, "Pseudo-Complementary FET Logic (PCFL): A Low Power Logic Family in GaAs”, IEEE Journal of Solid-State Circuits. Vol.31, No.7, July, 1996.

[19] P. Lassen, S. Long, K. Nary, "Ultra-Low Power GaAs MESFET MSI Circuits using Two-Phase Dynamic FET Logic”, IEEE Journal of Solid-State Circuits. Vol.28, No. 10, October, pp. 1038-45, 1993.

[20] V. Chandramouli, N. Michell, K. Smith, "A New, Precharged, Low-Power Logic Family for GaAs Circuits”, IEEE Journal of Solid-State Circuits. Vol.30, No. 2, February, pp. 140-43, 1995.

[21] D.H.K. Hoe, A.T. Salama, “Pipelining of GaAs Dynamic Logic Circuits”, Proc. IEEE International Symposium on Circuits and Systems, San Diego, USA, May, 1992.

[22] H. Kawasaki, "A Low Power 128x1-bit GaAs FIFO for ATM Packet Switcher” *IEEE Journal of Solid-State Circuits*. Vol.31, No.10, October, 1996.

[23] R. Kanan, A. Guyot, B. Hochet, M. Declerq, "A Divided Decoder-Matrix (DDM) Structure and its Application to a 8Kb GaAs MESFET ROM", Proc. 30th IEEE ISCAS, Hong Kong, 1997.

[24] D. Abbott, K. Eshraghian, “SiGe versus GaAs - Is there a challenge ?”, Proc. IEEE European Gallium Arsenide and related III-V Compounds Applications Symposium, Paris, France, 1996.

[25] C. Huang, "GaAs ICs for 3 volt Electronic", Proc. of the IEEE 5th European Gallium Arsenide and related compounds Applications Symposium, Bologna, Italy, Sept., 1997.

[26] K.M. Baughan, "The wireless Communication Market - Is there a Place for GaAs", IEEE GaAs IC Symposium, USA, 1995.

# 2. Modular Notation and Cryptography.

## 2.1. Introduction

Most of cryptography applications such as key public cryptosystems incorporate a exponentiation unit to implement algorithms for executing modular operations. The most widely used operations are addition, multiplication and exponentiation. The numbers to be operated are usually represented in modular representation. In this chapter we will briefly introduce the basic concepts of modular arithmetic as well as some general concepts of cryptography including some known protocols in order to familiarise oneself with a language.

The development of computer controlled communication networks promises effortless and inexpensive contact between people or computers on opposite sides of the world, replacing most mail and excursions with telecommunications. For many applications these contacts must be secure against both eavesdropping and the injection of illegitimate messages. Secret digital writing is being used to avoid message transformations. Techniques to avoid eavesdroppers actions are known as **cryptography**. The word comes from the Greek words kryptos ("hidden") and graph ("writing"). The history of cryptography dates far back. The Spartans used the "scytale" method as early as 400 BC. Secret writing has been used by many ancient societies to protect information beyond typical methods.

Currently, cryptosystems are more frequently required in applications as remote cash dispensers, high speed computer terminals, authentication, digital signatures and private communication between others. In Europe, the new Smart Card Microcomputer Center will orient their principal applications to bank and telecommunications security, developing a cryptoprocessors family [1].

Due to the fact that, several of cryptography applications use satellite communication where principally radiation tolerant integrated circuits are needed, cryptography systems are also included into the market behaviour of GaAs digital integrated circuits which will be doubled in the next four year [2].

## 2.2 Concepts

Several concepts are used in cryptography. First of all, we will define the terms more frequently used, in order to manipulate a common language:

- *Cryptography* may be considered as the art and science of both keeping messages secure and reading messages meant to be secure. In other words, is the study of secret writing and is used to protect the exchange information between people or computers.

- *Encryption* (encode) and *decryption* (decode) are two inverse procedures always used in cryptography operations. The procedures allow to cipher or decipher a message to be transmitted through public channels.

- *Cryptology* is the study of encryption and decryption methods. Is the branch of the mathematics embodying the art and science of both keeping messages secure and reading messages to be secure.

- *Cryptanalisys* consist in breaking a single secret message. To recognise patterns in order to develop decryption algorithms, find general weakness in encryption algorithm.

In terms of security, two concepts are frequently used: *unconditionally secure* is a system which can resist any cryptanalytic attack and is based on the existence of meaningful solutions to a cryptogram. The another term is *computationally secure* which denotes a secure system due to the computational cost of cryptanalysis.

## 2.3 Applications

In cryptography, mathematical systems are studied in order to solve two security problems:

**i. Privacy** or **secrecy** requires that an intruder should not be able to determine the plain text corresponding to given cipher text and should not be able to reconstruct the key by examining cipher text for known plain text. In other words, to prevent the extraction of information by unauthorised parts (ciphering messages). The message must not be vulnerable to eavesdropping or alteration.

**ii. Authentication** requires that the sender can validate the source of message, that means that it was transmitted by a properly identified sender and is not a reply of a previously transmitted message. We can identify two aspects:

- **Message authentication** or **integrity** requires the ability to insure that a message was not modified accidentally or deliberately in transit, by replacement, insertion or deletion. It is used to prevent the unauthorised injection of messages into a public channel.

- **User authentication** service is used to verify that an individual is who he claims to be. Also is a protection against a sender of message later denying transmission. It is also known as **non repudiation** service or **digital signature**.

The security problems which must be solved by cryptography systems are the insecurity of the publics channels (eavesdropping, injection of illegitimate messages) and authentication (illegitimate messages, digital signatures).

Strong security levels are required in applications as: remote cash dispensers, computers terminals, image compression, access control, authentication, confidentiality protection, key exchange, digital signatures, distributed network security management, private communication and hybrid systems.

## 2.4. Mathematical basis.

The mathematical fundament of the modern cryptography are functions of difficult inversion like as one-way functions, trap-door one-way functions, hash functions and one-way hash functions.

### 2.4.1 One-way functions

The one-way functions are easy to compute but difficult to invert. So, given some variable **x** and a one-way function **f**, is easy to compute **f(x)**, but given **f** and **f(x)** is difficult to compute **x**. However, there is no proof that one-way functions exist. Mathematical discoveries are showing that more and more functions considered initially as one-way, are no longer so.

### 2.4.2 Trap-door one-way functions

Trap-door one-way functions are a subset of one-way functions. For these functions, giving a secret piece of information makes easy to compute the inverse of the function.

### 2.4.3. Hash functions.

Hash functions are usually many-to-one functions. They are used to characterise a larger piece of data. A Hash function accepts a variable-size message **X** as input and outputs a fixed-size representation **H(X)** of **X**, sometimes called a message digest. In general **H(X)** will be much smaller than **X**. **H(X)** might be 64 or 128 bits, whereas **X** might be a megabyte or more.

### 2.4.4. One-way hash functions.

These type of functions are both one-way and hash functions as explained below. They are also known as compression function, cryptographic checksum, manipulation detection code, message authentication code, data integrity check or contraction functions.

The one-way hash function has the additional property that given a hash value **y,** it is difficult to find a value **x** such that **f(x) = y.** There are also hash functions that require a key. Given **k** and **x**, you can compute **y**, but having any other combination of data does not provide enough information to easily compute any other data. Additionally, one-way hash functions used in cryptography are random. Each change in any bit of the input, changes in average half of the bits in the output. For this reason, one-way hash functions can serve to detect modification of a message, that is to say, it can serve as a cryptographic checksum.

One-way hash function presents as input a string of arbitrary length and its output is a unique fixed length number. Example MD5 produces 128-bits hash value. One way property consists in making computationally infeasible to find two documents with same hash. The properties of one-way hash functions can be summarised as:

- **f** can be applied to an argument of any size.
- **f** produces a fixed-size output.
- **f(x)** is relatively easy to compute for any given **x**.
- For any given **y**, it is computationally infeasible to find **x** with **f(x) = y**
- For any fixed **x**, it is computationally infeasible to find **x'≠x** with **f(x)' = f(x).**

The security of public key systems depends on the fact that the public transformations are trapdoor one-way functions. Trapdoors permit decoding by recipients. The modular arithmetic facilitates the wrapping concept. In next section the principal terminology and basically theory of modular arithmetic will be discussed.

## 2.5. Modular Arithmetic

### 2.5.1. Introduction

In normal arithmetic, operations like adding or multiplication present usually an important increasing output natures for each increasing input pattern set. This is not necessarily true in modular arithmetic. Modular arithmetic is an interesting and viable alternative for doing arithmetic on large integer numbers.

This "other" arithmetic is based on some simple principles of number theory. It is possible to represent any integer number **X**, using several module $\mathbf{m_1}$, $\mathbf{m_2}$, $\mathbf{m_3}$, ....., $\mathbf{m_h}$, that contain not common factors:

$$\mathbf{x_1 = X \bmod m_1,}$$
$$\mathbf{x_2 = X \bmod m_2,}$$
$$\vdots$$
$$\mathbf{x_h = X \bmod m_h}$$

Now, having the complete modular representation, it is possible to operate indirectly with "residues" obtained: X mod $m_1$, X mod $m_2$, .....,X mod $m_h$, instead of directly with the number **X.** It is easy to compute ($\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$, ....., $\mathbf{x_h}$ ) from an integer number **X,** without lost of information in this process. If $\mathbf{x_i}$ equals the remainder of **X** divided by $\mathbf{m_i}$, then modular arithmetic expresses this as: $X \bmod m_i = \mathbf{x_i}$ and is read "**X** modulo $\mathbf{m_i}$ equals $\mathbf{x_i}$" [3] or can also be expresses as $\mathbf{X} \equiv x_i \bmod m_i$, and is read "X is congruent to $x_i$ modulo $m_i$" Additionally the notation X≡Y mod M means that M divides X - Y that is X and Y lie in the same residue class modulo. *ModM* denotes a number Y such that X≡Y mod M.

### 2.5.2. Modular arithmetic operations.

Modular arithmetic can be added, subtracted, multiplied and exponentiated, the equivalent of repeated multiplication. Modular arithmetic satisfies the following properties for all residues α obtained from a division by an integer **M** expressed in radix **r**, that remainders form what is called the ring of residues modulo M.

| | | | |
|---|---|---|---|
| i. | $(\alpha + 0) = (0 + \alpha) = 0$ | | (3) |
| ii. | $\alpha + (M - \alpha) = 0$ | (M - α is the additive inverse of α ) | (4) |
| iii. | $\alpha.1 = 1.\ \alpha = \alpha$ | | (5) |
| iv. | $\alpha.\alpha^{-1} = \alpha^{-1}.\ \alpha = 1$ | ($\alpha^{-1}$ is the multiplicative inverse of α) | (6) |
| v. | $(M - 1) \bmod M = -1$ | | (7) |

vi. $(x + y) \bmod M = ((x \bmod M) + (y \bmod M)) \bmod M$ (8)

vii. $(x - y) \bmod M = ((x \bmod M) - (y \bmod M)) \bmod M$ (9)

viii. $(x \cdot y) \bmod M = ((x \bmod M) \cdot (y \bmod M)) \bmod M$ (10)

ix. $(x \exp y) \bmod M = ((x \exp (y-r) \bmod M) \cdot (x \exp r \bmod M)) \bmod M$ (11)

From mentioned properties we have:

- If $XY \bmod M = R$ then $(XS)Y \bmod (MS) \operatorname{div} S = R$ (12)
- $X R^{-1} + Y R^{-1} \equiv S R^{-1} \bmod M$ if and only if $X + Y \equiv S \bmod M$ (13)

One can not divide congruencies in all cases. Another interesting and remarkable property consists in that for any pair of relatively prime integers, multiples of each can always be found such that their difference is unity. In other words, there always exists some multiple of an integer *p* which leaves a remainder of one when divided by another integer prime to it. The multiplier of *p* is always a smaller number than the divisor of the product.

x. $(a \cdot x) - (b \cdot y) = 1$ (14)

This property brings us to the inverse modulo function which is equivalent to finding a number such that:

xi. $(y \cdot x) \bmod M = 1$ or $y \cdot x \equiv 1 \bmod M$ (15)

For cryptographic applications, we want **M** to be as large as possible, it is easiest to let $\mathbf{m_1}$ be the largest odd number, and to let $\mathbf{m_2}$ be the largest odd number minor than $\mathbf{m_1}$ that is relatively prime to $\mathbf{m_1}$.

### 2.5.3. Underlying functions

#### 2.5.3.1. Euler and Fermat totient function

Also known as only Euler totient function or Euler Φ function. Euler and Fermat identity are considered as a way to choose the large random numbers $\mathbf{m_1}$ and $\mathbf{m_2}$. For any integer message X which is relatively prime to M, it means, gcd(X, M) = 1 then $X^{\Phi(M)} \bmod M = 1$. Let Φ(M) be the Euler totient function giving the number of positive integers smaller than M which are relatively prime to M. Unlike Fermat's little theorem, M does not have to be prime. For this reason Euler totient function is refereed to as Euler's generalisation of Fermat's little theorem. The Euler totient function presents the following properties for any integer (message) X:

i. Euler totient function is multiplicative, so, if $\Phi(m_1)=x$ and $\Phi(m_2)=y$, then $\Phi(m_1m_2) = xy$.
ii. If $m_1$ is prime, then $\Phi(m_1) = m_1-1$, since all numbers smaller than a prime are not divisors of prime.
iii. Then, if $M = m_1m_2$, where $m_1$ and $m_2$ are primes we can write $\Phi(M) = \Phi(m_1)\Phi(m_2)$ and $\Phi(M) = \Phi(m_1-1)\Phi(m_2-1)$, so, $\Phi(M) = M - (m_1 + m_2) + 1$.

So, if we choose a given number *d* which is relatively prime to $\Phi(M)$, it has a multiplicative inverse *e* in the ring of integers modulo $\Phi(M)$. It is denoted as $e.d \equiv 1 mod \Phi(M)$. Euclid's algorithm allows to calculate that.

### 2.5.3.2. Euclid's Algorithm.

The Euclid's algorithm [4], is very useful in modular arithmetic and basically is used to calculate the greatest common divider (gcd) of two integer numbers $r_0$ and $r_1$, allowing also to compute the inverse multiplicative of a number. The algorithm is shown below:

$r_0 = q_1r_1 + r_2 \qquad 0 < r_2 < r_1$
$r_1 = q_2r_2 + r_3 \qquad 0 < r_3 < r_2$
$r_2 = q_3r_3 + r_4 \qquad 0 < r_4 < r_3$
. .
$r_{m-2} = q_{m-1}r_{m-1} + r_m \qquad 0 < r_m < r_{m-1}$
$r_{m-1} = q_mr_m$

$\gcd(r_0, r_1) = \gcd(r_1, r_2) = \gcd(r_2, r_3) = \ldots\ldots\ldots = \gcd(r_{m-2}, r_{m-1}) = \gcd(r_{m-1}, r_m) = r_m$
Then $\gcd(r_0, r_1) = r_m$

Euclid's algorithm can be used to determine if a positive integer $b < M$, has its module M inverse multiplicative, that is $\exists\tau$ such that $\tau.b \bmod M = 1$. Replacing $r_0 = M$ and $r_1 = b$ it is possible to know the existence of the inverse multiplicative.

**Theorem:** Let $t_0, t_1, \ldots\ldots\ldots t_m$ .be a sequence of recurrences:

$t_0 = 0$
$t_1 = 1$
.
.
$t_j = t_{j-2} - q_{j-1}t_{j-1} \bmod r_0 \qquad$ if $j \geq 2$.

For each j, such that $0 \leq j \leq m$, we have a $r_j \equiv t_j\, r_1 \bmod r_0$ where $r_j$ and $t_j$ are defined as below:

By mathematical induction procedure we assume that is true for j = 0 and j = 1 and then we proof that is true for j = i - 1 and j = i - 2, i ≥ 2.

$r_{i-2} \equiv t_{i-2}\, r_1 \bmod r_0$

$r_{i-1} \equiv t_{i-1}\, r_1 \bmod r_0$

Then we have:

$r_i = r_{i-2} - q_{i-1} r_{i-1}$

$r_i \equiv t_{i-2}\, r_1 - q_{i-1} t_{i-1} r_1 \pmod{r_0}$

$r_i \equiv (t_{i-2} - q_{i-1} t_{i-1})\, r_1 \bmod r_0$

$r_i \equiv t_i\, r_1 \bmod r_0$

As a corollary of the last theorem we have that: if $\gcd(r_0, r_1) = 1$, then $t_m = r_1^{-1} \bmod r_0$

The inverse modulo function is that number which multiplied by the original number gives one as the remainder $x \equiv y^{-1}$ modn. If y and n are relatively primes then $x = y^{-1}$modn has a unique solution, on the contrary if y and n are not relatively primes the equation has not solution. If n is a prime number, then every number from 1 to n-1 is relatively prime to n and has exactly one inverse in that range. The Euclid's algorithm which is used to find the inverse multiplicative in modular representation can be expressed as:

$n_0$ <----- n
$b_0$ <----- b
$t_0$ <----- 0
t <----- 1
q <----- $n_0 / b_0$
r <----- $n_0 - q \times b_0$
While r > 0 do
    temp <---- $t_0 - q \times t$
    If temp ≥ 0 do temp <---- temp mod n
    If temp ≤ 0 do temp <---- n - ((-temp) mod n)
    $t_0$ <----- t
    t <----- temp
    $n_0$ <----- $b_0$
    $b_0$ <----- r
    q <----- $n_0 / b_0$
    r <----- $n_0 - q \times b_0$
If $b_0 \neq 1$ then the inverse multiplicative of b does not exist
If $b_0 = 1$ then $b^{-1} \bmod n = t$

### 2.5.3.3. Fermat theorem

For any number x, which is not divisible by its exponent p, which is a prime number, in general we have that: if $x^p - x = y$ then **y** is divisible by **p.**

Now, factoring: $x^p - x = x(x^{p-1} - 1) = y$, being the entire expression divisible by p, but x is not divisible by p, so $(x^{p-1} - 1)$ is divisible by p. The Fermat theorem is: if x is any integer not divisible by the prime p, then $(x^{p-1} - 1)$ is divisible by p. In other terms: $x^{p-1} \equiv 1 \bmod n$. It is also known as a Fermat's little theorem. Can be also expressed as: if for any prime *p* and any element $a$ , $a < p$ :

i. $a^p \bmod p = a$ or $a^{p-1} \bmod p = 1$
ii. $ax \bmod p = 1$ Combining $ax \bmod p = 1 = a^{p-1} \bmod p$ or
iii. $x = a^{p-2} \bmod p$

This type of representation, is also known as **residue arithmetic** or **modular arithmetic**, which uses the **residue number system** representation.

### 2.5.4. The residue number system.

The residue number system is an integer number system whose most important property is that additions, subtractions and multiplication are inherently carry-free, allowing add, subtract or multiply numbers in one step regardless of the length of the number involved [5].

A residue number system is characterised by a base that is not a single radix but an h-tuple of integers ( $m_1, m_2, m_3, ....., m_h$ ) where each of these $m_i$ ( i = 1, 2, 3, ..... h ) is called a **modulus** or **module.** An integer X is represented in the residue number system by an h-tuple $(x_1, x_2, x_3, ....., x_h)$ where $x_i$ is a nonnegative integer satisfying:

$$\mathbf{X = m_i\ q_i + x_i} \qquad (16)$$

where $x_i$ is the largest integer such that: $0 \le x_i < m_i$, $x_i$ is known as the residue of $Xmodulom_i$. Next two notations: $\boldsymbol{Xmodm_i}$ and $|\mathbf{X}|_{\mathbf{m_i}}$ are commonly used. As can be seen the number X does not have to be a positive integer but can be negative as well.

It is easy to compute ( $x_1, x_2, x_3, ....., x_h$ ) from an integer number X by means of divisions and it is possible to recompute X from ( $x_1, x_2, x_3, ....., x_h$ ) provided that $0 \le X < M$, where M is the least common multiple (lcm) of the basis $x_i$. Satisfying the last condition we can say that the residue representation is unique, however the converse is not true. Residue representation is periodic and for that reason the range must be limited to include the numbers wanted. The number of the elements in the useful range is the least common multiple of the module. To get the largest range we must select a module which factors are relatively prime.

In cryptography applications, residue number system can be used to compute multiplication and exponentiations module a very large M, where M requires hundreds or a few thousands of bits**.** It is the appropriate and more frequently number system used to implement higher speed performed cryptosystems.

### 2.5.4.1. Advantages of RNS

Residue arithmetic or modular arithmetic allows to implement a simple and fast realisation of addition, subtraction and multiplication. These operations must be performed on relatively short operands. Residue number system shows an easy range extension increasing the number of modules or their magnitude. Another advantage consists that in hardware implementation some modularity can be achieved. The system is also inherently parallel.

### 2.5.4.2. Disadvantages of RNS.

The residue number system is not weighted and for this reason some disadvantages are present in this notation system. The disadvantages of a modular representation are that it is comparatively difficult to test whether a number is positive or negative (sign detection) or to test whether or not $(\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}.....,\mathbf{u_r})$ is greater than $(\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3},.....,\mathbf{v_r})$ (magnitude comparison).

Additionally, intermodular operations as scaling and division are difficult to implement. It is also difficult to detect an overflow when operations as addition, subtraction or multiplication have been executed. Algorithms using sign detection must be avoided.

From the point of view of hardware implementation, extra converters are needed. Binary - Residue converters for inputs and Residue - Binary converters for outputs. Also extra bits are needed to represent numbers, that means extra input and output lines.

The amount of time required to add, subtract or multiply n-digit numbers using modular arithmetic is essentially proportional to **n** (not counting the time to convert in and out of modular representation). This is a disadvantage if let consider add and subtract operations, but it is a remarkable advantage with respect to multiplication.

Modular representation can be justified only if fast means of conversion between modular and positional notation are available. Due to the periodic representation of residue, we must limit the

range of numbers to include only numbers such that $X < M$. This is a consequence of the Chinese remainder theorem.

### 2.5.5. Chinese Remainder Theorem.

This theorem also known as **theorem C**, was apparently first stated and proved in its generality by Chin Chiu Shao (1247), but a very special case of this theorem had been already stated by the Chinese mathematician Sun Tsü probably between 280 and 473 [6]. The Chinese Remainder Theorem is a method to solve the congruence problems.

**Theorem C:**

Let $\mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \ldots\ldots, \mathbf{m_h}$ be positive integers that are already prime in pairs, that means, $\mathbf{gcd\ (\ m_i,\ m_j\ )\ =\ 1}$ when $j \neq k$, and let $\mathbf{a}, \mathbf{m_1}, \mathbf{m_2}, \mathbf{m_3}, \ldots\ldots, \mathbf{m_h}$ be integers.
Then there is exactly one integer **X** that satisfies the conditions:

$$\mathbf{a \leq X < a + M,} \quad \text{and} \quad \mathbf{X \equiv x_i\ (mod\ m_i\ )} \quad \text{for} \quad \mathbf{1 \leq i \leq h}$$

$$\text{With} \quad \mathbf{M = m_1.m_2.m_3.....m_h}$$

In other words, by Chinese remainder theorem we can express:

$$X = \sum_{j=1}^{h} Q_j X_j \bmod M \quad \text{where } Q_j = (M/m_j)^{m_j - 1} \bmod M \qquad (17)$$

As a consequence of theorem C , it is possible to use modular representation for numbers in any consecutive interval of $\mathbf{M = m_1 m_2 m_3 \cdots\cdots m_h}$ integers. Numbers out of this range would have a repeated representation, that means, the residue representation is periodic. In general, the residue representation is unique, however the converse is not true due to its periodicity.

### 2.5.6. Periodicity properties.

The **period** of the odd module **M** denoted **P(M)**, is the minimum distance between two distinct 1's in the sequence of residues of powers of 2 taken mod M:

$$\mathbf{P(M) = min\ \{\ i\ /\ i > 0\ \ and\ \ |\ 2^i\ |_M\ = 1\ \}} \qquad (18)$$

The **half period** of the module **M**, denoted as **HP(M)** is the minimum distance between a pair of subsequent 1 and M - 1 in the sequence of residues of powers of 2 mod M. P(M) exists for

any M, but HP(M) exists for some M only and then 2HP(M) = P(M). In table I, we can see some periodicity example for some module. In table II, the period and half period for each case [7].

Table I. Periodicity example for some module.

| **i** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $2^i$ | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 |
| $\lvert 2^i \rvert_3$ | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 |
| $\lvert 2^i \rvert_5$ | 1 | 2 | 4 | 3 | 1 | 2 | 4 | 3 | 1 | 2 | 4 | 3 | 1 |
| $\lvert 2^i \rvert_7$ | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 | 2 | 4 | 1 |
| $\lvert 2^i \rvert_9$ | 1 | 2 | 4 | 8 | 7 | 5 | 1 | 2 | 4 | 8 | 7 | 5 | 1 |
| $\lvert 2^i \rvert_{11}$ | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 | 1 | 2 | 4 |

Table II. Periods and half periods

| i | P(M) | HP(M) |
|---|---|---|
| $\lvert 2^i \rvert_3$ | M(3) = 2 | HP(3) = 1 |
| $\lvert 2^i \rvert_5$ | M(5) = 4 | HP(5) = 2 |
| $\lvert 2^i \rvert_7$ | M(7) = 3 | HP(7) = ∝ |
| $\lvert 2^i \rvert_9$ | M(9) = 6 | HP(9) = 3 |
| $\lvert 2^i \rvert_{11}$ | M(11) = 10 | HP(11) = 5 |

HP(M) and P(M) can be calculated by using the following recursive equation.

$$\lvert 2^i \rvert_M = \lvert 2 \lvert 2^{i-1} \rvert_M \rvert_M \tag{19}$$

The global measure of periodicity is:

$$\text{Per}(M) = \min \{\text{HP}(M) , \text{P}(M) \} \tag{20}$$

Due to the fact that, periodicity properties allow an extensive use of some arithmetic functions like carry-save adders (CSAs), and carry-propagate adders (CPAs) with end-around-carry (EAC), we will briefly mention some additional periodicity properties:

**i.** The equations to compute HP(M) and P(M) for some M.

$$\text{HP}\,(2^{i-1} + 1) = (i - 1) \tag{21}$$

$$\text{P}\,(2^{i-1} + 1) = 2(i - 1) \tag{22}$$

If M = kB with k and B odd, then P(M) is a multiple of P(B).

**ii.** For periodicity measures the following inequalities hold:

- $a \leq P(M) \leq M - 1$ (23)
- If HP(A) exits, then $a - 1 \leq HP(M) \leq (M - 1)$ (24)
- $Per(M) \leq (M - 1) / 2$ (25)

**iii.** The concept of the period of M allows to generalise the identity.

from: $\left|2^{ta}\right|_{2^a - 1} = 1$, t nonnegative integer (26)

to: $\left|2^{tP(M) + i}\right|_M = \left|2^i\right|_M$ t is any nonnegative integer (27)

which holds for any M.

**iv.** Similarly, given an odd M with $HP(M) < \infty$, with the concept of the half-period of M, the well known identity:

$$\left|2^{t(a - 1)}\right|_{2^a - 1} + 1 = (-1)^t, \quad (28)$$

is generalised to: $$\left|2^{tHP(M) + i}\right|_M = (-1)^t \left|2^i\right|_M, \quad (29)$$

As mentioned before, the residue representation is periodic and for this reason the range must be limited to include only the needed numbers. The number of the elements in the useful range would be the least common multiple (l.c.m.) of the basis: $M = l.c.m\ (m_1 m_2 m_3 \cdots\cdots m_n)$**.** So, to get the largest range we must select a module which factors are relatively prime, two module **mi** and **mj** are said to be relatively prime if:

$gcd(mi, mj) = 1$ where **gcd** is the greatest common divisor.

So, if all module are relatively prime, the M range will be optimised, and:

$$M = \prod_{i=1}^{n} m_i \quad (30)$$

Writing as **XmodM,** the result produces numbers from **0** to **n - 1.** In table III, can be seen the range for different module and periods.

Table III. Parameters of RNS with small $a_i$ and Per ($M_i$)

| $\max_i \{a_i\}$ | $\text{Max}_i \{Per(M_i)\}$ | RNS | Range |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 4 | 4 | { 5, 7, 9, 16 } | $1.23 \cdot 2^{12}$ |
| 4 | 5 | { 5, 7, 9, 11, 16 } | $1.69 \cdot 2^{15}$ |
| 4 | 6 | { 5, 7, 9, 11, 13, 16 } | $1.37 \cdot 2^{19}$ |
| 5 | 4 | { 5, 7, 9, 17, 32 } | $1.30 \cdot 2^{17}$ |
| 5 | 5 | { 5, 7, 9, 11, 17, 31, 32 } | $1.74 \cdot 2^{25}$ |
| 5 | 6 | { 5, 7, 9, 11, 13, 17, 31, 32 } | $1.41 \cdot 2^{29}$ |
| 6 | 7 | { 5, 7, 9, 11, 13, 17, 31, 43, 64 } | $1.89 \cdot 2^{35}$ |
| 7 | 7 | { 5, 7, 9, 11, 13, 17, 31, 43, 127, 128 } | $1.87 \cdot 2^{43}$ |

As next useful concept, we will define $\theta(\mathbf{r})$ as the minimum number of logic levels on a CSA tree with **r** inputs.

### 2.6. Algorithms

As can be seen from the next summary, all algorithms used in cryptography involve the modular exponentiation procedure. The security of this scheme is based on the complexity:

- Of computing the discrete algorithms,
- Of big numbers factorisation.

The NP (nondeterministic polynomial) complete problems show promise for cryptographic use, but no security has still been proved, additionally are too difficult to implement in hardware.

#### 2.6.1. Discrete logarithm problem.

One of the most important one-way functions in cryptography is based on the discrete logarithm problem in the finite Galois field GF(p). Given a large prime p and a primitive element $\alpha \in \mathbf{GF(p)}$, it is feasible to compute the value of $\mathbf{y} = \alpha^{\mathbf{x}}$, using $\Theta(\mathbf{log x})$ modular multiplication. It is however infeasible to compute the value of **x**, given **y**, $\alpha$ and **p**.

For the solution of this discrete logarithm problem long integer multiplication are needed. Similar discrete logarithm problem can be found in finite fields $GF(p^k)$ of prime characteristic p or in the group of points on an elliptic curve.

#### 2.6.2. Factoring.

For a given number **n**, the difficulty of factoring it is related with the required time to transformate the number and with the performance/costs of the computation machines. In table IV, this situation is illustrated.

Another interesting considerations are shown in table V. Considering a super computer performing a million operations per second, and that a network of a million of such computers is assigned the task, it will take the time shown in table V to execute the factoring for each length of **n**.

Table IV Factoring **n** difficulty.

| n(bits) | $ Machine | Time |
|---|---|---|
| 100 | $ 25.000 | 2 weeks |
| 150 | $10.000.000 | 1 year |
| 200 | $ 10.000.000.000 | 1 year |

Table V. Factoring for each length of n

| n | Digits | # oper. | Years |
|---|---|---|---|
| 512 | 154 | ------- | ------- |
| 664 | 200 | $10^{23}$ | 3700 |
| 1028 | 308 | ------- | $10^{10}$ |

### 2.7 Types of Cryptosystems

A cryptographic system is a single parameter family $\{S_K\}_{K_\varepsilon\{K\}}$ of invertible transformations $S_K$ : {M}-->{C}, from a space {M} of plaintext messages to a space {C} of ciphertext messages. Where M and C are the original and encrypted messages respectively: M = ($m_1$, $m_2$.....$m_n$) and C = ($c_1$, $c_2$, $c_3$, .....$c_n$). The parameter K is called the key and is selected from a finite set {K} called the keyspace [8]. In private communication applications using public channels, two different approaches have been proposed to transmit information without compromising the security of the system: **public key distribution system** or also known as symmetric key system and **public key cryptosystem**, also known as asymmetric-key system.

#### 2.7.1 Public key distribution system

In this approach, two users must exchange a key over a secure channel and then use it for both enciphering and deciphering messages. These users can only use one key in common being computationally infeasible to compute the key from the information overhead. In figure 2.1 the scheme is illustrated.

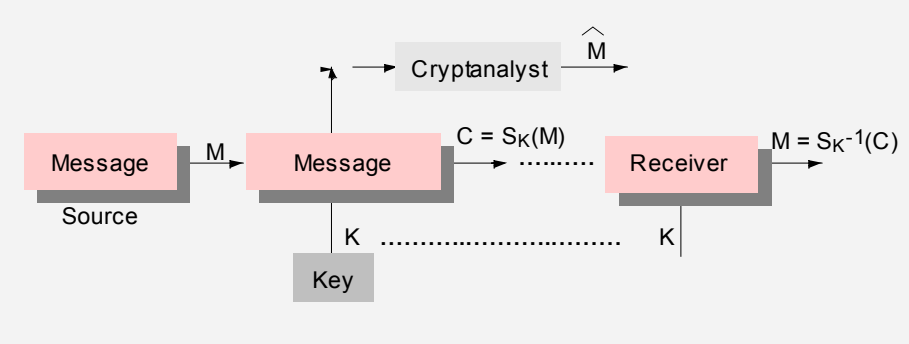



Figure 2.1 Symmetric key cryptosystem

With this system a separate key is needed for every pair of users, that means that, **n** potential keys to be transmitted would be required if exist **n** recipients. Then **n(n - 1)/2** keys would be required for **n** users.

As can be seen from the graph, a transmitter generates a plain text or unciphered message **M** to be communicated over an insecure channel to the legitimate receiver. To prevent the eavesdropper, operates on **M** with an invertible transformation $\mathbf{S_K}$ to produce the ciphertext or cryptogram $\mathbf{C= S_K(M)}$. The key **K**, is transmitted only to the legitimate receiver via a secure channel. The receiver knows **K**, so, he can decipher **C** by operating $\mathbf{S_K^{-1}}$ to obtain the original message **M**. In the figure M^ represents a non authorised decrypted message.

$$\mathbf{S_K^{-1}(C) = S_K^{-1}(S_K(M)) = M} \qquad (1)$$

In this system of cryptography, data are encrypted and decrypted using the same key. So, we have that, for a symmetric system where only one key is used: **M = D(K, E(K, M))**. The strength of this scheme largely depends on size of key. The system is much closer to realisation and is useful in one-way authentication systems. The drawback of the system consists in that a secure channel to send the private key is needed. Classical examples of implementation are Caesar cipher, one time pad, Enigma. Also, several fast and tested algorithms are available, like, DES, RC4, IDEA.

### 2.7.2. Public key cryptosystems

The first Public Key Encryption algorithm was proposed by Whitfield Diffie and Martin Hellman in their seminal paper [8]. Also, Ralph Merkle independently presented the concept in 1976 [9]. Public key transformation is one way encryption with a secret way to decrypt. In this system each user places in a public file an encryption procedure **E.** The directory giving the encryption procedure of each user is a public file. On the other hand, the user keeps secret his

corresponding decryption procedure **D.** So, enciphering and deciphering processes are governed by distinct **E** and **D** procedures which must satisfy:

i. D(E(M)) = M
ii. E(D(M)) = M
iii. Both **E** and **D** must be easy to compute.
iv. Revealing **E**, the user does not reveal an easy way to compute **D**.

So, computing **D** from **E** is computationally infeasible, about $10^{30}$ instructions considering 200-bits number representation. This will be discussed in section 2.7.3.1. This approach is more powerful and eliminates the need for a secure key distribution channel. For an asymmetric system where the keys are different we can write

$$\mathbf{M = D(K_D, E(K_E, M))} \quad (2)$$

In figure 2.2, this approach is represented.

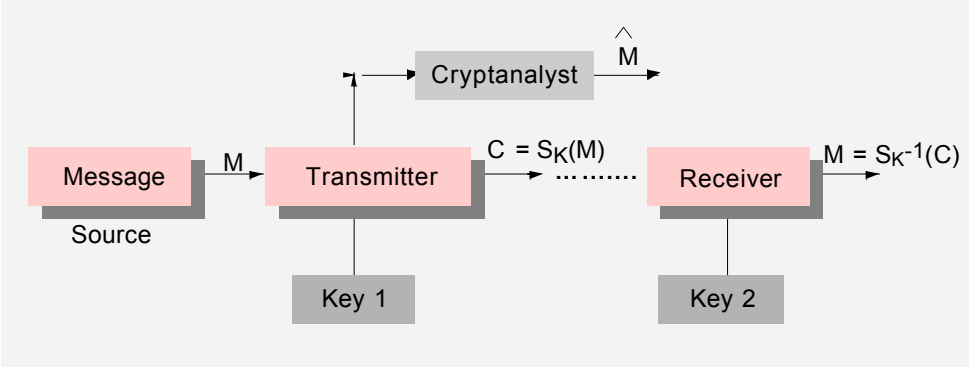



Figure 2.2. Public key cryptosystem

Each user has a key pair, a public key and a private key. Each user's public key is listed in a public directory and must carefully guard his private key against disclosing. Anything encrypted with one key may only be decrypted by the other. To make message readable only by B, encrypt it using B's public key. Public key encryption relies on the difficulty of factoring a very large number. The public and private keys are usually functions of a pair of large prime numbers.

This general scheme allows to implement several protocols oriented to ciphered messages and/or authentication procedures. It is slower than symmetric cryptography. Between others, some of the most popular protocols are RSA, ElGamal, Fiat-Shamir, etc. Some of them will be briefly presented.

### 2.7.2.1. RSA encryption and signature protocol

The RSA protocol is named after its three inventors, Ron Rivest, Adi Shamir and Leonard Adleman [10], who first introduced and patented the algorithm in 1978. RSA system offers high security, but its speed is quiet low, the method does not readily lend itself to efficient implementation in hardware, limiting the range of potential applications. RSA encryption function consists in computing $C = X^e \text{mod} M$**,** where M = pxq being *p* and *q* very large random primes (*p* and *q* remain secret). The RSA algorithm is based on the difficulty of prime factorisation of large integers. The modular exponentiation of the plaintext X, produces the ciphertext C using the encryption key *e*.

The public key is composed by M and a number *e* relatively prime to (p-1)(q-1). An integer *d* is computed from *e*, *p*, and *q*, to be a multiplicative inverse of *e*.mod(p-1)(q-1). It means that a private key is computed as $d = e^{-1} \text{mod}(p-1)(q-1)$. The encryption process consists in computing $c = x^e \text{mod} M$ while the decryption process computes $x = c^d \text{mod} M$.

As can be seen, the decryption process is also a modular exponentiation using the secret key. In general, compute $X^e \text{mod} M$ require both $2\log_2 e$ multiplication and $2\log_2 e$ divisions. The RSA cryptosystem is considered secure if the integers X and M have several hundred decimal digits.

### 2.7.2.2. ElGamal protocol

The ElGamal protocol [11] consists of a *s* secret key and a public key which is composed by: the prime number **p,** an integer number $\alpha \text{mod} \mathbf{p}$ and the P integer number modulo p: $P = \alpha^s$. The scheme is shown in figure 2.3. This digital signature algorithm is 10 to 40 times slower than RSA for signature verification. Additionally, it has been criticised because of its short key length.

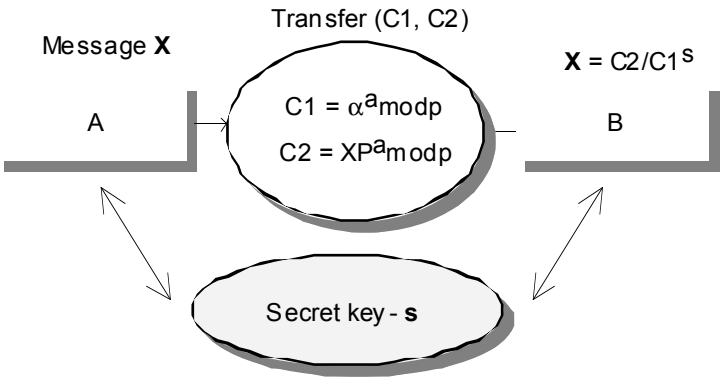



Figure 2.3. ElGamal representation protocol.

To send a message from A to B, the public key is chosen and two number $C_1$ and $C_2$ are calculated. $C_1 = \alpha^a \text{mod} \mathbf{p}$ and $C_2 = xP^a \text{mod} \mathbf{p}$, then the message is composed by $(C_1, C_2)$. To decrypt the message, it is possible to get X from $P^a$ because it is easy for the receiver to find it from s. So, $P^a = \alpha^{sa} = C_1^s$ then $M = C_2/C_1^s$

### 2.7.2.3. RPK Encryption

The RPK system [12] is based on the discrete logarithm problem, it means that, the public key is calculated from a private key using operations mathematically equivalent to exponentiation in finite fields. In this system each user select a private key which is composed by three numbers $D_m, D_t, D_b$. The public key of the user A will consist of the states $E_m, E_t, E_b$ of the three *component generators* after $D_m, D_t, D_b$ clock cycles respectively. The three numbers $E_m, E_t, E_b$ are assumed to be publicly known. When user B, wishes to encrypt a plaintext message M to be only decrypted by user A using A's private key, user B generates a true random initialisation key $R = R_m, R_t, R_b$ to be used during the encryption of M. The random initialisation key is used to exponentiate the base state generating an open key $Q = Q_m, Q_t, Q_b$ which is included within a header, preceding the main body of the ciphertext. R is also used to exponentiate the public key E, generating a final generator initialisation state K. Starting from the state K, the mixture generator must be activated to obtain a keystream output and combine it with the plaintext M to obtain the main body of the cipher text C.

To decrypt users A first uses the state given by the open key Q contained in the message header to compute the generator state. Using the private key, exponentiate the open key Q to compute the final initialisation key K which will give the state of the mixture generator. Further, for each block of the ciphertext body, the mixture generator runs to obtain a part of the keystream output. It will be used to generate a pseudo random permutation table. Running again the mixture generator is possible to obtain additional keystream which combined with the ciphertext allow to generate through some permutations the original plaintext. In polynomials terms:

i. $K_j(x) = [\, E_j(x)]^{R_j} = \text{mod} p(x)$, for j = m, t, b
ii. $E^R = (X^D)^R = K = (x^R)^D = Q^D$

### 2.7.3. Analysis signature using Public Key Cryptosystems.

Physical, hand-written signatures are used in everyday life to solve many problems. The signatures are used to prove that a person was in physical contact with a particular document and usually to certify the reading of the document. Physical hand-written signatures present some inherent traits that can be used to enforce legal implications, as well as to solve identification problems. First one, it can not be denied or forged, anyone can identify the author

seeing and reading it. Second, the signature cannot be transferred to another document and finally, it had to be made by deliberate action. Using the two different cryptography systems is also possible to sign a message. Like as in various protocols of public-key cryptosystems, digital signature would exist as additional digital information. General attributes of the procedure are:

i. **A** uses a digital signature algorithm with his private key to sign the message.
ii. **A** sends the message to **B**.
iii. **B** verifies **A** digital signature using **A** public key.

Anyone who has the public key can verify that he signed the message. The signature is a function of the message, so it cannot be applied to another message. If someone cut the signature out and copied it to another message, the verification algorithm would indicate that they do not match. Nobody but A can make the signature, and he needs to apply the algorithm in order to do so, which shows it was deliberate.

#### 2.7.3.1. Diffie-Hellman key exchange protocol

Like other protocols (ElGamal), the Diffie-Hellman protocol [13] is based on the discrete logarithm problem over finite fields. This protocol is considered as a public key distribution system rather than a true public-key cryptosystem.

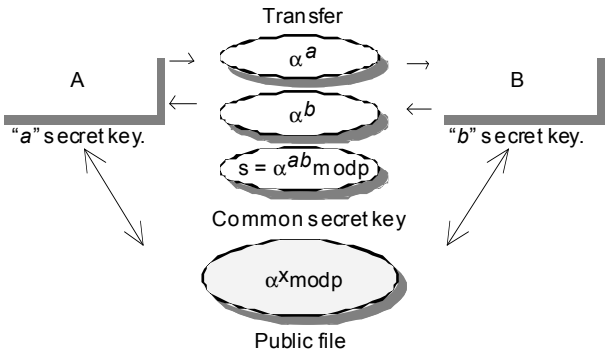



Figure 2.4. Diffie-Hellman scheme.

The system parameters -as is shown in figure 2.4- are composed by a module which is selected to be a big prime number **p**, and an integer number $\alpha$ which is a fixed primitive element of GF(p)**.** Each user generates an independent, secret and random number $1 \leq x \leq p-1$. To exchange messages, each user places $\alpha^x \bmod \mathbf{p}$ in a public file with his name and address**.** So, to send a message from A to B, a secret and random number *a* is chosen for A. The emitter A sends $\alpha^a$ to B. B chooses a second secret and random number *b* and sends $\alpha^b$ to A. The common secret key would be $S = \alpha^{ab} \bmod \mathbf{p}$. So, the first user , can find *s* calculating $(\alpha^b)^a$, the second user , can find *s* calculating $(\alpha^a)^b$. Each one can verify the identification without

knowing the secret key number. The secret key *s* is never sent through the public channel. If p is a prime slightly less than $2^n$ then all quantities are representable as n-bit numbers. Exponentiation then takes at most 2n multiplications modp while taking logs requires $p^{1/2} = 2^{n/2}$. Then the cryptanalytic effort grows exponentially relative to legitimate efforts. If n = 200, then at most 400 multiplications are required to compute $\alpha^a mod p$ from *a*, yet taking logs modp requires $2^{100}$ or aproximately $10^{30}$ operations.

### 2.7.3.2. Guillou-Quisquater protocol

The Quisquater protocol [14] is used for verification of signatures and keys adaptation. The system consists of a secret *s* and a random *r* keys, a small integer number *v* which is known by two users and a big integer number **n** product of two primes numbers **p** and **q**.

$s = r^{-1/v} \text{mod} \mathbf{n}$ where $r < n$.

To send a signed message from A to B:

i A sends $T = r^v \text{mod} \mathbf{n}$ to B and B resends $u = rs^d \text{mod} n$.

ii. The recipient compares $r^d u^v$ with Tmodn authenticating $r^d u^v = T \text{mod} n$.

The Quisquater protocol is used in P83C 852 and P83C855 Philips and SCALPS from Catholic University of Louvain in Belgium.

### 2.7.3.3. Fiat-Shamir user authentication protocol

The Fiat-Shamir identification protocol [15] allow to verify the identification of an user. The module is selected to be a big integer number n, which is a product of two big primes numbers p and q. The protocol consists of two keys. A private key which is a square residue number module n , that means, $x^2 \equiv v \text{mod} \mathbf{n}$. The last expression must have a solution and it must be guaranteed that $v^{-1} \text{mod} \mathbf{n}$ exist. The second key s is a small integer number such that $s = (v^{-1})^{1/2} \text{mod} \mathbf{n}$. To verify the authentication from A to B, A sends $x \equiv r^2 \text{mod} \mathbf{n}$ to B and B resends a value b=0 or b=1. A responds with y=r (if b=0) or y=r.s modn (if b=1).

### 2.7.3.4. Schnorr identification and signatures

The Schnorr identification protocol [16] which is based on discrete logarithms allows to verify the identification of a user. The identification systems are composed by two primes p and q such that $q \geq 2^{140}$, $p \geq 2^{512}$, a number $\alpha \in Z_p$ with order q as $\alpha^q = 1 \text{mod} p$ $\alpha \neq 1$, a one-

way hash function h: $Z_q$ x Z ---> (0,....$2^t$ - 1), and its own private and public key. The system publishes p, q, α, h and its public key.

The user can generate himself his private key *s* which is a random number in (1,2,...q) and the corresponding public key $v = \alpha^{-s} \text{mod} p$. The scheme to verify the identity implies the generating of an identification string I, and a signed pair (I, *v*).

i. A send to B its identification string I and its public key *v*. B checks *v* by verifying the system signature transmitted by A.
ii. A picks a random number $r \in (0,....q - 1)$ and computes $x \equiv \alpha^r \text{mod} p$ sending x to B.
iii. B sends a random number $\in (0,....2^t - 1)$ to A;
iv. A sends to B, $y \equiv r + se \text{ mod} q$
v. B checks that $x = \alpha^y v.^s \text{mod} p$ and accepts A's proof of identity.

### 2.7.3.5. Yen-Laih digital signature verification

The Yen-Laih digital signature verification protocol [17] allows to verify several signed messages sent by a user. The system parameters are composed by a prime modulus $|M| \geq 512$ where |M| means $1 + [\log_2 M]$, q a prime divisor of (M-1) with $|q| \geq 140$, and a number $\alpha \in Z_N$ with order qmodM. The protocol consists of two keys. Each user selects an integer $s \in Z_q$ as his secret key and corresponding to *s*, user computes his public key as $k \equiv \alpha^{-s} \text{mod} M$. Now, for generating the signature for a single message X, the signer A computes:

i. $y \equiv \alpha^r \text{mod} M$ where $r \in Z_q$ is a random integer.
ii. $e = h(y, X) \in Z_q$ where h() is a one-way hash function.
iii. $z \equiv r + s_x e \text{mod} q$

Considering that the computation of $k \equiv \alpha^r \text{mod} M$ is independent of the message to be transferred it can be precomputed in advance. {X, (x,z)} constitutes the message-signature pair signed by A with public key *k* and secret key *s*. This protocol allows to verify the signature in a batch manner of several messages improving efficiency. To check the validity of the signature (y, z) on X signed by A, the verifier B computes:

i. $e = h(y,X)$ ii. $\alpha^z k^e \equiv x \text{mod} M$

As can be seen modular exponentiation is a basic operation widely used in cryptography and constitutes a computational bottleneck in many protocols.

## 2.8. Conclusions.

It is become clear that public-key cryptography is an indispensable tool for simplifying key management and enabling secure communication. What is less clear is which of the available public-key cryptosystem is best. For analyzing this question several criteria must be considered: security, computation speed, key size and the intended application, being the security the most important consideration in choosing among public-key technologies. Wiener in its paper [18] makes it clear that there is no single "best" public key technology, but that the best choice is situation dependent. The three main uses of public-key techniques are digital signatures, encryption and decryption for passing symmetric keys, and on-line key exchange, as can be seen in table VI, all three public-key cryptoystems can be used for each of these purposes.

Table VI. Public-key techniques [18].

| After | Rivest Shamir Adleman | Diffie Helman | | Elliptic curves | |
|---|---|---|---|---|---|
| Main uses. | RSA | DH | DSA | ECDH | ECDSA |
| Digital Signature. | Yes | --- | Yes | --- | Yes |
| Encryption/decryption | Yes | Yes | --- | Yes | --- |
| On-line key exchange | Yes | Yes | --- | Yes | --- |
| Exponent size | 1024 | 160-256 | 160 | 160-200 | 160-200 |

DH = Diffie-Hellman key exchange algorithm. ECDH =Elliptic curve key exchange algorithm

DSA = Diffie-Hellman Digital Signature Algorithm. ECDSA = Elliptic curve Digital Signature Algorithm

There are three main public-key cryptosystems contenders (RSA, DH, Elliptic curves) and to compare their speeds, it is necessary to decide what key sizes give comparable security levels. Each technique has a variable key size that can be increased to achieve higher security at the cost of slower cryptographic operations. So, RSA with a 1024 bit module is used as the basis of the comparison. On the one hand, to perform a discrete logarithm with 1024 bits Diffie-Hellman or DSA modules requires about the same run-time than factoring 1024 bits with RSA. On the other hand to achieve the level of security of 1024 bit-RSA using elliptic curves requires a key size of multipliers in teh range 171-180 bits. In table VII, a summary of different features is shown.

Table VII. Cryptography techniques features.

| Techniques | Size key | Typical | Size exp. | Techniques |
|---|---|---|---|---|
| RSA | Module | 1024 | 1024 | Involves exponentiation module a number M that is the product of two large prime numbers. |
| DH/DSA | Prime p. | 1024 | 160 | Involves exponentiation module a large prime number p |
| Elliptic curves | Prime number | 1024 | 160-200 | Computations with points on an elliptic curve. |

Modular Exponential function is a basic operation widely used in many protocols oriented to cryptography applications. This condition makes that, modular exponentiation becomes a core function of the cryptosystems.

Because of cryptographic applications must insure the communication between people or electronic systems, and considering that the security of many algorithms is based on the difficulty of prime factorisation of large integers, modular representation of numbers is required. Modular arithmetic is appropriated for executing arithmetic on large integer numbers.

Modular exponentiation function is executed as repeated modular multiplications. So, modular multiplier architecture looking high performance and simplicity is needed. It is important to investigate regular configurations with smallest circuit depth in order to improve the performance. Area size is a constraint to be also considered.

## 2.9. References.


[1] Y. A. Les Investissements en cartes à puce se multiplient, Electronic International HEBDO, No. 282, 30 Oct., 1997

[2] L'Europe devient compétitive en circuits GaAs. Hyper 97, Electronic's journal.

[3] D. E. Knuth, Seminumerical algorithms, Second edition, The art of Computer Programming, Addison Wesley Publishing Co., v. 2 , Massachusetts, 1981.

[4] Stinson, Douglas R., Cryptography Theory and Practice, CRC Press Inc. Florida, 1995.

[5] I. Koren, Computer Arithmetic Algorithms, Prentice Hall, New Jersey, 1993

[6] K. Hwang, Computer Arithmetic, John Wiley, New York, 1979

[7] S. Piestrak, Arithmétic des résidues: applications et conception de matériels, TIMA Laboratory conference, Grenoble, December, 1996.

[8] W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, IT-22, No. 6, Nov 1976, pp. 644-654.

[9] R. Merkle, Secure communication over an insecure channel, Communications of the ACM, 1976.

[10] R. Rivest, A. Shamir, L. Adleman , A method for obtaining digital signatures and Public-key cryptosystems, Communication of the ACM, 21, 1978, pp 120 - 126.

[11] T. ElGamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions of Information Theory, v. IT-31, No. 4, Jul. 1985, pp. 469 - 472.

[12] W.M.Raike, The RPK Public Key Cryptographic System, Technical summary July, 1996.

[13] W. Diffie, M. Hellman, Privacy and Authentication: An Introduction to Cryptography, Proc. of the IEEE, v. 67, No. 3, Mar 1979, pp. 397-427.

[14] J.J. Quisquater, C. Couvreur, Fast decipherment algorithm for RSA Public-key Cryptosystem, Electronics Letter, vol.18, pp. 905-907, 1982.

[15] A. Fiat, A. Shamir, How to improve yourself: Practical solutions to identification and signature problems. Advances in Cryptology - Proc. of Crypto'86, pp. 186-194, 1986.

[16] C.P. Schnorr, Efficient signatures generation for smart cards. J. Cryptology, Vol. 4, No. 3, pp. 161-174, 1991.

[17] Sung-Ming Yen, Chi-Sung Laih, Improved digital signature suitable for batch verification, IEEE Transaction on computers, Vol. 44, No. 7, pp. 957-959, July, 1995.

[18] Wiener M.J., Performance comparison of Public-Key Cryptosystems, CryptoBytes, Vol. 4, Number 1, Summer 1998.

# 3. Architecture for Computing the Modular Multiplication.

## 3.1. Introduction.

VLSI circuits that accelerate the encryption and decryption of messages using encryption techniques and circuits capable of performing long wordlength modulo multiplication at very high speed attract much interest for cryptography applications. The modular multiplication problem consists in calculating the number **C** such that: **C=X.YmodM,** where **X**, **Y** and **M** have several hundred decimals digits. This long wordlength modulo multiplication has also applications in other secret communication problems and other cryptographic methods.

Basically, module multiplication can be executed in two ways, the first one consists in mixing into a single operation the multiplication and reduction steps, as multiplication partial products are formed, a decision is taken whether or not to perform a reduction on these partial products. In the second one, multiplication and reduction are separated tasks, with the output of multiplier feeding the input of the reduction unit.

In this chapter the state of the art of modular multiplication algorithms as well as an alternative architecture for computing modular multiplication based on Montgomery's algorithm will be presented. This architecture will be used in next chapter to implement the architecture for executing the algorithm for bit modular exponentiation.

## 3.2 Multiplication algorithms.

Since several algorithms have been developed to compute the multiplication function, there are different possibilities for calculating this function. Existing state-of-the-art dedicated to hardware for calculating XYmodM, makes use of several techniques for speeding up the calculation. Some of those techniques are listed in table I.

More specifically algorithms based on modular multiplication method have been proposed by Brickell, Eldridge, Walter, Baker, Omura, Sedlak, Bucci, Montgomery. Some of them will be briefly reviewed below.

Table I Techniques for speeding up the calculation [1]

| **Technique** | **Implication** |
|---|---|
| Shift M up so that its most significant digit always has the same position in the hardware. | This allows module with different number of bits to be used easily. |
| Interleave modular subtractions with the normal calculation of the product by repeated shift and add. | To save register space. Because the number stay roughly the size of the modulus M rather than becoming as large as the product XxY. |
| Increase the base of the number representation. | Reduce the number of digits in the multiplicand, reducing the number of clock cycles in the algorithm. |
| Use a redundant representation for calculations instead binary representations. | This avoids the unbounded propagation of carries, allowing add all the digit operation in parallel. |
| Shift up the multiplicand A and the modulus M by several places. | Addition of the digit multiple Y* of the multiplier does not affect the topmost bit used to decide the multiple M* of M which must be subtracted. |
| Use the precalcualtion of some or all of the linear combination $Y^* \pm M^*$. | Speed up the calculation but can increase the area of further registers. |
| Choose modulus having a decomposition as a product of pairwise coprime numbers | The arithmetic can be done independently module each $M_i$ allowing use the Chinese Remainder Theorem |
| Decide which multiple of M to subtract early enough. | The adder not to be kept waiting for it. |
| If the multiplicand X is not already in non redundant form, convert it. | To produce the digits in the order they are consumed. |

### 3.2.1. Brickell's algorithm

The Brickell's algorithm [2] uses a delay carry representation which consists of two registers of ***n*** bits each one for the uncarried carries. This approach allows execute the modular multiplication in ***n*** + 10 cycles clock pulses. Up to n clock ticks may eventually be required to assimilate the carries at the end of the computation. Eldridge and Walter [3] have reported several sources of possible error in the hardware implementation and have also done some contributions to remove those difficulties.

One of the ideas tacit in Brickell's algorithm is to calculate *XYmodM* as *XS*YmodMS/S* for a fixed r-power $S = r^E$ causing a shift up by E places. The number E of extra digits needed at the top of the registers is large enough to make the contribution of Y to the partial product R insignificant in a certain precise way. In conclusion, Brickell has shown how to design a compact operation based on a radix 2 without frequent data transmission between processors performing modular multiplication and memories storing data in progress. Brickell's algorithm uses a redundant representation for calculations, this means that numbers may have

digits from a range greater than that required, such as {0,1,2} instead of only {0,1} for binary representations, this avoids the unbounded propagation of carries illustrated by the decimal addition of 1 and 9999..9, allowing execute all the digit operations of an addition in parallel with carries influencing only the digit sums in the next one or two places.

### 3.2.2. Eldridges's algorithm

Due to the fact that iterated addition step of the multiplication algorithm may involve unlimited carry propagation, Eldridge [4] proposes to use a redundant number system in order to avoid carry propagation. It uses also larger bases (radix 4). The base 4 representation have approximately half the number of digits. This feature allows speed up calculations. This algorithm involves both a high radix representation and recoding **Y.**

$$S_{i+1} = \begin{cases} (S_i + x_i y + M) \text{ div } 2 \text{ if odd.} \\ (S_i + x_i y + \delta_i M) \text{ div } 4 \text{ if even.} \end{cases}$$

where $\delta_i$ satisfy $S_i + x_i Y + \delta_i M$

The algorithm compute $XYR^{-1} \bmod M$, where M is odd. R is a power of 4 and $M < R$ has at most n binary digits. So, $R = 4^n$ and X is expressed as redundant base 4. Each step is simple leading that hardware may use a faster clock speed.

### 3.2.3. Walter's algorithm.

Walter in its paper [5], presents several details to generalise Brickell's fast modular multiplication algorithm when the number of representations have a general 2-power radix. The propose consists in using a redundant number system to enable parallel digit operations.

Additionally, the effect of varying the radix, also proposed by Kameyama [6], on the efficiency of hardware implementations is considered showing that lower order terms in general hardware are dominated for small values of the base of the number of representation. So, increasing the base reduces the number of digits in the multiplicand and so reduces the number of clock cycles in the algorithm. However, the depth of hardware that has to be driven in a single clock cycle is increased as well, so that a slower clock must be used [1]. The suggestion consists in taking a small increase above 2 of the radix but shows also that there is not advantage in taking a much larger radix.

The algorithm computes a residue R and an integer quotient Q satisfying X*Y = M*Q + R, where R is either the smallest non-negative residue of X*YmodM or differs by at most M from it. This algorithm requires n+E cycles for executing the modular multiplication, where E represents the power of the radix.

In another paper [7], the same author proposed a new technique for speeding up modular multiplication which consists in scaling the modulus. The technique truncates the least significant digits of both modulus and partial product allowing calculate the quotient easier because it no longer depends on any digits of modulus as they are fixed.

### 3.2.4. Even's algorithm.

Even [8] presents a systolic array for performing modular multiplication of long integers which are represented in binary. They are fed serially, least significant bit first, to the first cell of the n-identical cells array. Consequently the product is supplied serially by the first cell, least significant bit first. This algorithm is based on the a modular reduction system proposed by Montgomery requiring per one *n*-bits modular multiplication 3***n*** clock ticks.

### 3.2.5. Morita's algorithm

In [9], the Brickell's algorithm was extended to be used on a radix higher than two, being the execution time faster than conventional algorithms based on radix 2. Later [10], a method for eliminating the slow restoring in modular multiplication was presented.

### 3.2.6. Massey-Omura's algorithm

The Massey-Omura's algorithm [11] is based on the discrete logarithm problem in the finite Galois field GF(p). For the finite fields $F_{2^n}$, an appropriate selection of the ortonormal base will allow execute the exponentiation of a n-bit number as a sequence of shifts.

So, if $N = \{e_0, e_1, e_2 \ldots\ldots\ldots, e_{n-1}\}$ is a normal base of the vectorial space $F_{q^n}$ over $F_q$, being q the characteristic of F and $e_i = e^{q^i}$, then for each integer k we have $e_i^{q^k} = e_{i+k}$, for all exponent of e reduced module M.

This method converts a modular exponentiation q module M in a cyclic shift of coordinates for a A-vector. Where $A^q = \{a_{n-1}, a_0, a_1, a_2 \ldots a_{n-2}\}$ and $A = \{a_0, a_1, \ldots\ldots a_{n-1}\}$ and k = 1. Omura has proved that a normal base is optimum when there is 2n-1 terms non zero in its product matrix. The circuit CY512i of CYLINK uses the Omura's algorithm

### 3.2.7. Sedlak's algorithm

In this process [12], the exponentiation function is obtained through a sequence of multiplication, each one, executed as a sequence of additions. Of this form, the modular operation is reduced to a sequence of subtractions. The maximum number of additions and subtractions can be predicted by using subalgorithms allowing speed up the multiplication. Both, the multiplication and division must be accelerated in the same proportions. Using this approach the multiplication can be executed three times more quickly than using classic technique.

### 3.2.8. Bucci's algorithm

The algorithm of Bucci-Barret [13] is very close to Montgomery's algorithm, in this approach the integer number to be reduced is replaced by a reduced number. So, the size of the number to be reduced is limited. For exponentiation function, the ideal size of the number to be reduced is the half of the size modulus. That represents the principal drawback of the approach due to the final results are highly degraded. However, the chip RSA512 of ANTEC uses the Bucci' s technique.

### 3.2.9. Montgomery's algorithm.

Montgomery's algorithm [14] is oriented to fast execution of modular multiplication. If **n** is the bit length of the modulus M, then modular multiplication is represented as:
**C = X.YmodM**, where M is an odd integer. X, Y and M are n-bit binary positive integers related by X, Y $\in$ [0, M-1].

The Montgomery's algorithm is a method for multiplying two integers modulo M avoiding division by M. It was proposed to compute $XYR^{-1}$modM, where R is a power of radix (r) used for representation of numbers. A non standard way representation called M-residues is used. Suppose all integer representations are in binary, it means that the radix r = 2. If ***n*** is the bit length of the modulus M, then modular multiplication is represented as XYmodM, where X, Y and M are n-bit binary integers.

**R** = $2^n$ - R is a power of radix. R is prime to M
**M** > 1 is an odd integer.
**n** is a number of bits of M. So, $2^{n-1} \leq M < 2^n$
M < R has at most **n** binary digits.

$0 \le Y \le M$ and an integer number $R^{-1}$ such that $RR^{-1} \bmod M = 1$ exists.

Let $X = \sum_{i=0}^{n-1} X_i 2^i$ where $X_{|i|}$ is the value of the i-th bit.

Let $Y = \sum_{i=0}^{n-1} Y_i 2^i$ where $Y_{|i|}$ is the value of the i-th bit.

$X = ( x_0, x_1, x_2, x_3, ....., x_{n-1} )$ where each $x_i, y_i$ is 0 or 1.
$Y = ( y_0, y_1, y_2, y_3, ....., y_{n-1} )$ such that $0 \le Y < M$

$X \equiv Y \bmod M$ means that M divides X - Y and X and Y lie in the same residue class modulo. A "mod M" expression denotes a number Y such that $X \equiv Y \bmod M$. Additionally, for every X, $Y \in Z$ let $X' \equiv XR \bmod M$ and $Y' \equiv YR \bmod M$ be; Then X' and Y' are called the image of X and Y respectively. In other words:

$X2^n \bmod M$ is a Montgomery's representation of X
$Y2^n \bmod M$ is a Montgomery's representation of Y

A representation of an image has at most n bits, it may exceed M, but is nonnegative. The idea is to do all the modular operations with images, what it is called the residues field. M(X.Y) denotes a Montgemery's multiplication.

If we have $Y' \equiv YR \bmod M$, we can compute Z'
where:

$Z \equiv XY \bmod M$
$T = X'Y'$
$T \equiv XYR^2 \bmod M$
$T \equiv Z'R \bmod M$

The algorithm computes $XYR^{-1} \bmod M$. We can generate a sequence where each Si satisfies the condition: $0 \le M + Y \le 2M$

$S_0 := 0$
For i= 0 to n-1 do
If $(S_i + x_i Y)$ is even.
$S_{i+1} = (S_i + x_i Y) \text{ div } 2$
else
$S_{i+1} = (S_i + x_i Y + M) \text{ div } 2$

By induction:

$2^i \times S_i \equiv \sum_{j=0}^{n-1} (X_j 2^j)\, Y \bmod M$, $\quad S_{i+1}$ is an integer for i = 0, 1, 2..... n-1,

the last expression can also be written as $2^i \times S_i \equiv (x_{i-1} x_{i-2} x_{i-3} \ldots x_0) Y \bmod M$, obtaining that $RS_n \equiv XY \bmod M$. Therefore $XYR^{-1} \bmod M$ is either $S_n$ or $S_n - M$, with $0 \leq S_i < M + Y < 2M$.

A division by $R^{-1} \bmod M$ can be easily executed calculating $((XYR^{-1} \bmod M)(R^2 \bmod M)) \bmod M$. The final result would be $XY \bmod M$. For radix=2 then: $R^2 \bmod M = (2^n)^2 \bmod M$. $R^2 \bmod M = 4^n \bmod M$. If X = Y we will have $X^2 \bmod M$.

### 3.2.9.1. Result Analysis.

Let $S_0$ be the result of the first algorithm iteration, $S_1$ and the following sequences considering the worst case could be expressed as:

$S_1 = (S_0 + M + Y) \text{ div } 2$

$S_2 = (S_1 + M + Y) \text{ div } 2$

$S_2 = ((S_0 + M + Y) \text{ div } 2 + M + Y) \text{ div } 2$

$S_2 = S_0 2^{-2} + (M + Y)(2^{-2} + 2^{-1})$

For the j-th term: $S_j = S_0 2^{-j} + (M + Y)(2^{-j} + 2^{-j+1} + 2^{-j+2} \ldots\ldots\ldots + 2^{-2} + 2^{-1})$

But :

$$\sum_{j=1}^{h} 2^{-j} = (2^{-h} + 2^{-h+1} + 2^{-h+2} \ldots\ldots\ldots + 2^{-2} + 2^{-1}) \approx 1$$

Then we can re-write:

$$S_j < S_0 2^{-j} + (M + Y)$$

As Y < M, we can note that the length word of $S_0$ will be n + 1. Being $S_n$ the n-th the final result would be $XY2^{-n} \bmod M$.

This algorithm presents less area requirements and possibility of using segmentation techniques [15]. Additionally, unlike the previous algorithms, it avoids regular division replacing it by an operation which requires less time. We can mention as principal drawbacks that an additional constant must be computed. Besides two additional modular multiplication must also be executed**.** Montgomery's algorithm is competitive only if the number of modular multiplication to be executed is high, like as in modular exponentiation case.

This algorithm provides significant reduction in the number of multiplication required to compute modular exponentiation when **n** has several hundred bits [16]. Montgomery's algorithm multiplication process takes a time proportional to the number of digits in X. For last reason, some authors [17] have proposed increasing the radix of the representation in order to decrease the number of digits in X.

Using 4-power or 8-power radix representation is possible to speed up the operation. The use of redundant number system in order to avoid the unbounded propagation of carries allows execute all the digits operation of an addition in parallel with carries influencing only the digit sums in the next one or two places [1], this can be seen in [2][18][19]. In table II the principal features of each hardware implementation are shown.

Table II. Hardware implementation of modular multiplication.

| **Algorithm** | **Clock pulses** | **Radix(r)** | **Notation** | **Technique** |
|---|---|---|---|---|
| Montgomery | …….. | r = 2 | Binary | Mod. Reduction |
| Brickell | n+10 | r = 2 | Binary | Delayed carry rep. |
| Eldridges | ……. | r ≥ 2 | RNS | Large base |
| Morita | LogM/logr | r ≥ 4 | Not redund. | Compact operators |
| Even | 3n | r = 2 | Binary | Systolic arrays |
| Walter | n+(1/ρ)logS | r ≥ 4 | RNS | Large base |

### 3.3. Hardware for Computing Modular Multiplication Algorithm.

In this section we will examine the characteristics of an alternative two radix architecture for computing a modular multiplication based on Montgomery's algorithm, useful in performing the RSA Public Key Cryptosystems. Considering that the ultimate performance of an integrated circuit can be substantially improved by using optimised architectures, is important to investigate regular configurations with the smallest circuit depth achievable for executing the required operation.

Currently, all of the chips perform the exponentiation as a series of modular multiplication. For that reason the request for fast and inexpensive modular multipliers for long integers continues. Some efforts have been oriented to achieve that, however, the corresponding circuits tend to be complex.

Arithmetic operators exhibit in general a great activity and dissipate consequently a significant share of the power supplied to a circuit. Specifically, a multiplier dissipates much more power than an adder when activated due to its design or layout structure is not as regular as an adder.

Modular exponentiation can be executed by standard multiplication followed by a modular reduction or combining multiplication and modular reduction tasks [16]. We have shown that one of the most widely used algorithms for modular multiplication is the Montgomery' algorithm [14] which is a method for multiplying two integers modulo M avoiding division by M. This algorithm execute the modular multiplication as a series of additions. Also, it is suitable for hardware implementation allowing mix some ideas to overlap the multiplication and reduction phases.

Several architectures have been reported using residue or redundant numeric representation. In this case, an alternative architecture using binary representation will be discussed. From Montgomery' method, the M-residues are represented in a non-standard way. This method is very useful only if the number of modular computation is high. Our purpose is to examine an alternative architecture to be used in executing of modular multiplication operation oriented to cryptography systems design (exponentiation function).

In order to evaluate the architecture performance, a prototype implementation allowing to check step by step the partial results, each one of the additions, shifting and parity evaluation is required. For that reason a small 12 bits prototype has been designed. Additionally, using long bit numbers for the prototype would require a long row of I/O pads, which would define the total area of the prototype, while the active area would represent a small porcentage of the global area. With long bit numbers the prototype becomes more expensive and it does not give greater additional information,

The experimental 12x12 bits modular multiplier prototype has been designed, fabricated and tested using this architecture. The circuit was fabricated by AMS using 0.6 μm-CMOS technology. The architecture, its operation and some simulation and experimental results are presented. The evaluation is provided according to the functionality, power consumption and performance under the condition of 5V supply voltage. The experimental circuit includes 4100 transistors into an active area of 1.33 x 0.93 $mm^2$ .

### 3.3.1. Carry Save Adders

Carry Save Adders are usually used when three or more operands must be simultaneously added, as shown below. In our case, at most three operands must be added, for this reason the technique which is called *Carry-Save Addition* will be used. The carry save adder is a technique widely used, it accepts three n-bit operands and generates two n-bits results. Is also called a "*three inputs Wallace's tree*". So, as can be seen from figure 3.1, for three inputs x, y, z, we have two outputs s and c where:

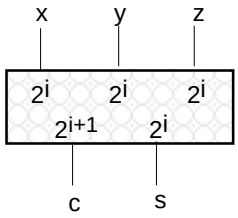



Figure 3.1. CSA unit

$s_i = x_i \oplus y_i \oplus z_i$ and $s_i = p_i \oplus z$, where $p_i = x_i \oplus y_i$

$c_{i+1} = x_i y_i + z_i(x_i \oplus y_i)$ or $c = Maj\,(x_i, y_i, z_i)$

or

$c_{i+1} = g_i + p_i \,.\, z$, where $g_i = x_i y_i$

In general $x_i + y_i + z_i = 2c_{i+1} + s_i$.

As can be seen, the CSA cell functions as a counter of "1"s. In carry-save addition, the carry is propagated only in the last step, while in all the other steps partials sum and sequences of separately carries are generated. Using two operand adders the time consuming carry propagation must be repeated several times, if the number of operands is k, then carries have to propagate (k-1) times.

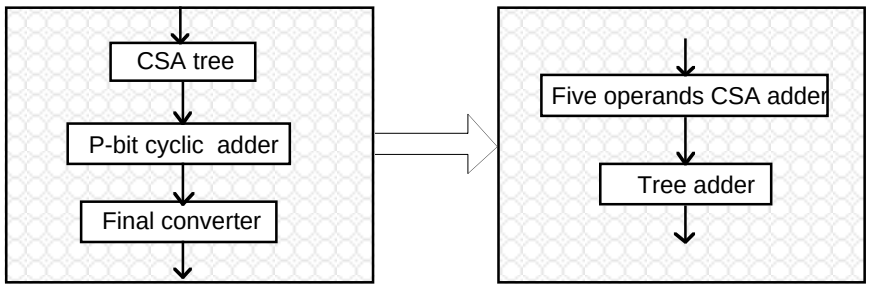



Figure 3.2. Block diagram of the CSA technique.

In carry save addition technique, we let the carry propagate only in the last step, while in all the others steps we generate a partial sum and a sequence of carries separately. So, in general terms, CSA technique require both a carry save addition tree and a carry propagation adder tree as is depicted in figure 3.2.

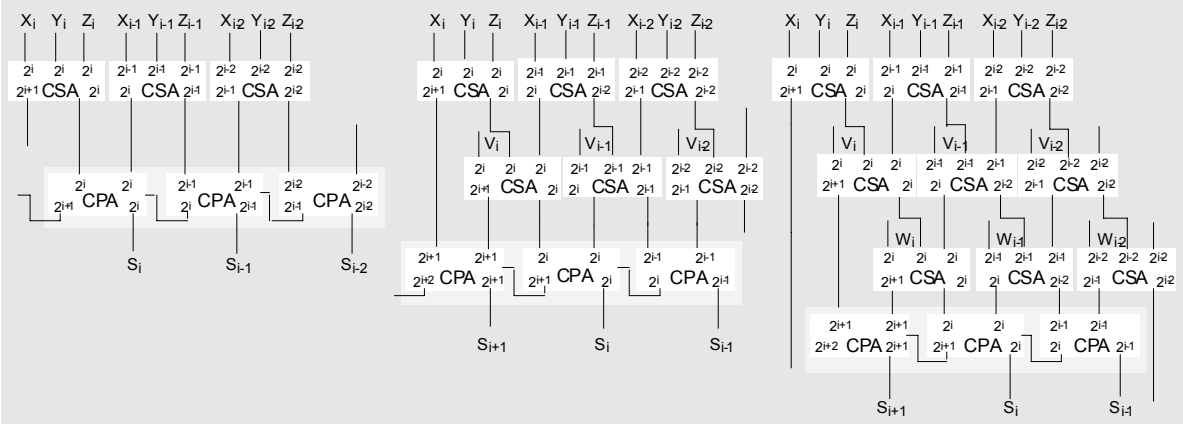


Figure 3.3. CSA trees for 3, 4, and 5 bits operands.

Five operands CSA adder requires three full adder levels. Architecture implementation using three, four or five operands carry-save adders, requires the topological configurations shown in figure 3.3. In table III, the minimum number of levels Θ(r) on a CSA tree with r input operands is shown.

Table III. Number of levels Θ(r) in function of r.

| R | 3 | 4 | 5-6 | 7-9 | 10-13 | 14-19 |
|---|---|---|---|---|---|---|
| Θ(r) | 1 | 2 | 3 | 4 | 5 | 6 |

But some important hardware considerations can be done. Now, we must define the global architecture using this approach.

### 3.3.2. Hardware implementation

In general, the objective consists in obtaining a fast and regular architecture to execute the modular multiplication based on Montgomery's algorithm, as is described in figure 3.4.

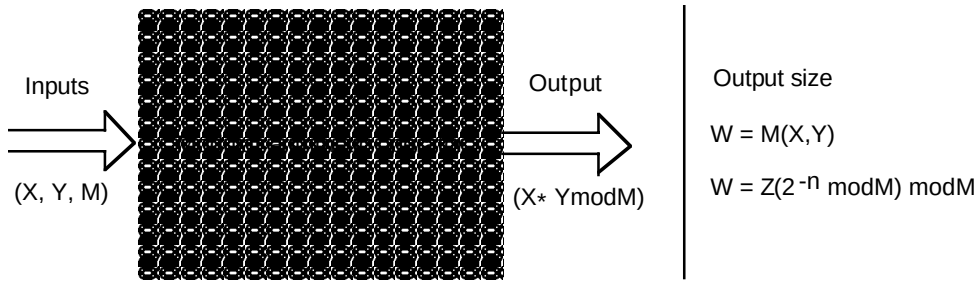



Figure 3.4. Modular exponentiation description

Due to the fact that, Montgomery's algorithm executes the modular multiplication as a series of additions and shifts, a general architecture of a modular multiplier would be composed by an adder which uses an accumulator register to accelerate the needed feedback loops. Additionally, a shifter to execute division by 2 would be also required. A general block diagram is shown in figure 3.5.

Some authors have proposed different hardware implementations of the Montgomery's algorithm [1][8]. Other have introduced several modifications [4][5] and have proposed new algorithms looking for a reduction of the computational complexity in order to speed up public key cryptographic functions [2][9].

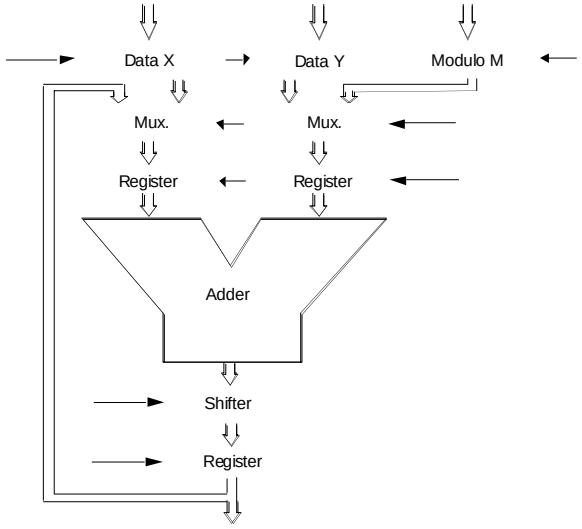



Figure 3.5. Modular Multiplier Datapath Architecture.

### 3.3.3. Architecture.

The new algorithm implementation allows to speed-up modular multiplication by using only selectors and three operands Carry Save Adder, which does not have carry propagagtion. From Montgomery's algorithm, the calculation $S_i + x_iY$ or $S_i + x_iY + M$ is performed for each $x_i$, so, several successive steps are executed.

As known, placing m parallel procesing stages and the interconnection buses require too much silicon area, this disadvantage becomes more critical if long length numbers are considered. For that reason, bit-serial processing stage architectures are attractive. These stages are smaller than their m-bit parallel counterpart by at least a factor of m.

In general, bit-serial processing stages are a factor of 1/m as fast as parallel processing stages, which might negate the gain achieved. However, bit serial multipliers and adders can be designed to eliminate carry propagation delay. This allows a net processing speed advantage when n-bit-serial processing stages replace a single n-bit parallel processing stage.

The proposed hardware implementation of a bit-serial modular multiplier used to execute the operation ($XYR^{-1}$modM) is showed in figure 3.6. It is basically composed by two functional blocks: a master synchroniser control part and a regular datapath. The first one includes the logic gates for generating synchronised control signals. The second one, is constructed by arranging n modular multiplier cells into an array.

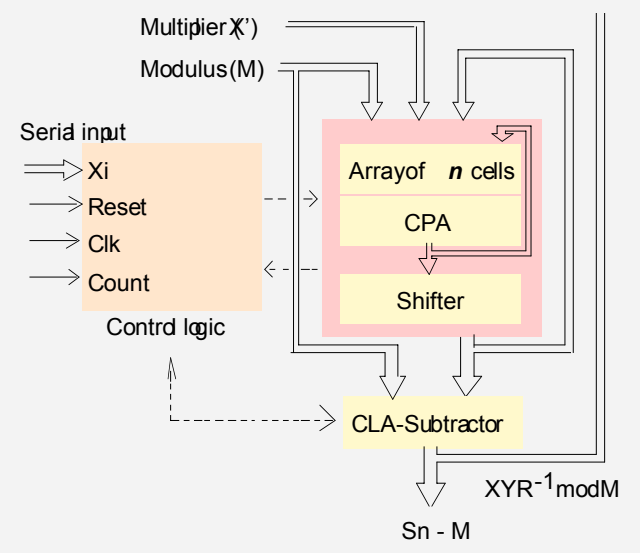



Figure 3.6. Modular Multiplier Hardware System.

The algorithm for this bit-serial multipier requires the multiplicand Y in parallel and the multiplier X in bit-serial form. During the first iteration, the first bit of the multiplier $x_0$ is entered and "*operated*" with the parallel multiplicand Y producing a set of variables which are added by the full adders to compute a set of partial product. Each step, the parity condition of the partial product is verified to decide if modulus must be added or not to previous product. The partial product calculation ends when it is shifted right one bit. This partial product $S_{i-1}$ is fed back in order to calculate the next partial product $S_i$. With this organisation, the system requires a feedback loops for each bit-cell. This operation continues until the multiplier X is exhausted and the entire modular representation final product is obtained.

In figure 3.7, a schematic capture of four bits of the modular multiplier architeture is shown. The inputs to the control part are supplied from the outside, and for every 0 < xi < n-1 the outputs of the control part are the corresponding control inputs to datapath. All control signals are synchronised by an external signal ***Count*** (*Counter* signal) which can also be generated internally. This signal allow count the ***n*** fields of serial input multiplicand X and is used to validate the load signal for processing new data.

The modulus M and the multiplier Y are fed parallely to datapath while the multiplicand X (*Xi* signal) is fed serially being processed only by the control part. A parity bit signal is carried out from the least significative bit of the multiplier output to the control part. In order to generate the appropriate signals to execute Si + xiY or Si + xiY + M, the control part (*ctrl* block) samples in each partial result both the serial input entering $0<x_i<n-1$ and the parity bit signal of $S_{i-1}$. The shifter used to execute division by 2 will be embedded into the array.

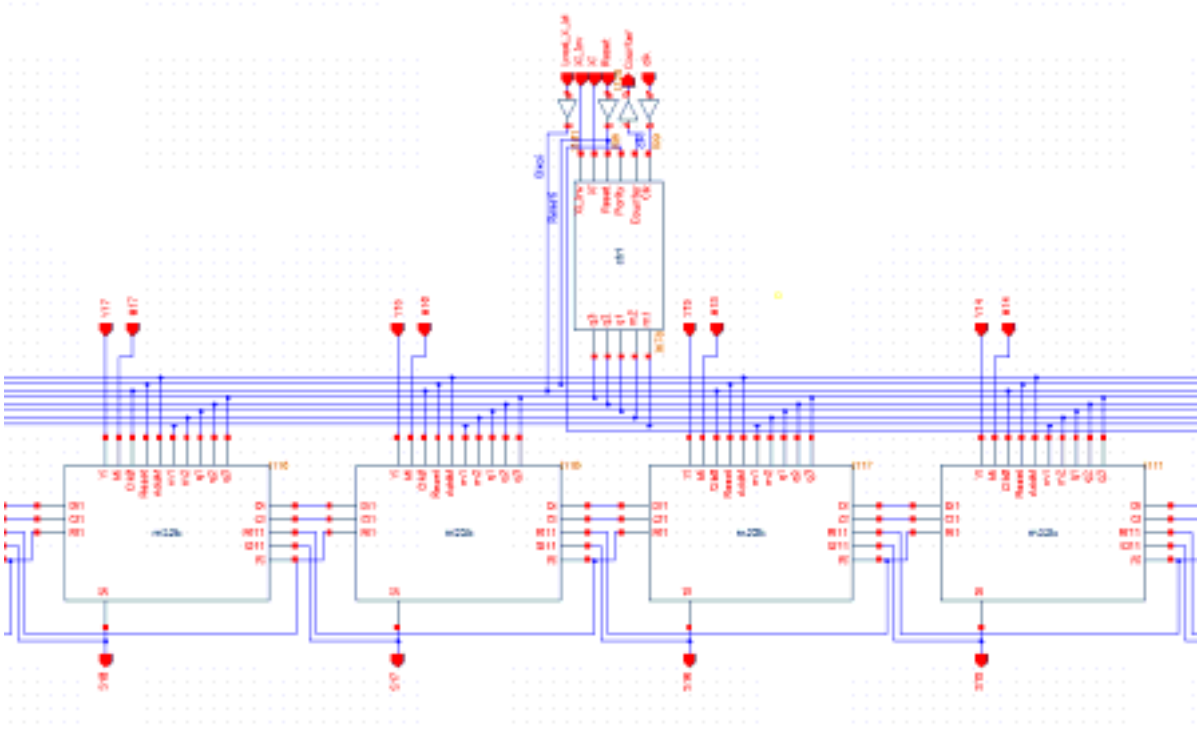


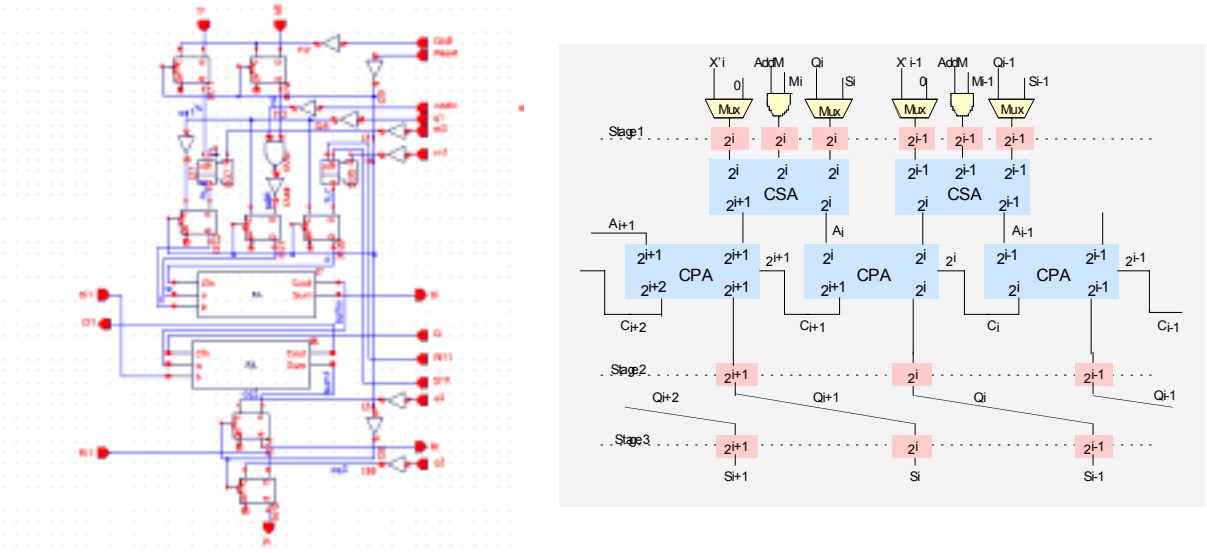
Figure 3.7. Modular architecture of the multiplier.

As can be seen from the graph, processing stages properly designed, allow individual stages to be directly connected as they are placed, thereby eliminating interconnection buses. This technique is known as "*interconnection by default*" [20] and is very useful for saving layout time and silicon area.

### 3.3.4. Modular multiplier cell

The modular multiplier datapath is composed by a regular array of *n* cells. The basic cell is composed by 5 registers, two full adders, and some logic gates containing about 72 equivalent gates as is shown in the schematic capture of the cell in figure 3.8(a). In figure 3.8(b) a connectivity between consecutive basic cells is also depicted.

First of all, we have chosen R to be a power of 2. It means that in this approach both the positive integer to be multiplied and the modulus are represented in binary form. On the one hand, bigger radix system can be used in order to reduce the whole process time, which is proportional to the number of digits in X. But, on the other hand, when bigger radix are used each step of addition becomes more complex.




(a) (b)

Figure 3.8. Basic cell. (a) Schematic capture. (b) Connectivity.

This implementation requires only six gate delays. The operation of the cell is straightforward. All flip-flops are set to reproduce a "zero" output at time t=0, by a reset signal. There is a global clock (*clk* signal) whose pulses synchronise all load signals of the flip-flops. Both rising and falling edge of a system clock are used to generate the load signals.

Three very simple clocked logic levels (stages) can be observed. In the first one, two multiplexers are used to choose the appropriate value to be added by the CSA units. After

each execution cycle (ti), the previous modulo partial result Si is fedback to be added itself by $x_i.y_i$. Depending of parity bit value, the second tick adds $M_i$ to $Q_i$.

The second one, load the addition executed through one row of CSA (Carry Save Adder) which have not carry propagation and whose outputs fed one row of the CPA (Carry Propagate Adder). One second tick would be required if a parity condition is detected. Finally, the flip-flops of the third stage are used to shift the partial result and their output at time t+1 is equal to their input divided by two at time t.

The total time to do a modular multiplication consists, therefore, of ***n*** execution cycles, each one requiring five clock ticks if parity condition is detected. Only three ticks would be required if parity condition is not present. The whole process takes a time proportional to the number of digits in X. Clock speed is bound by the number of gates on the longest or critical paths, which is found by adding the lengths of the critical pads needed to compute S and to perform the addition $Si + x_iY$ or $S_i + x_iY + M$.

Here, the remaining operation of subtracting M, if necessary is not shown. It will be discussed in next chapter. Observe that since M is odd, the addition of M to $Si + x_iY$, causes the new $S_{i+1}$ to be even, in other words the least significant bit of $S_{i+1}$ will be 0, so any information will be lost when division by 2 (shifted away) be executed. The diagram timing of the cell is depicted in figure 3.9.

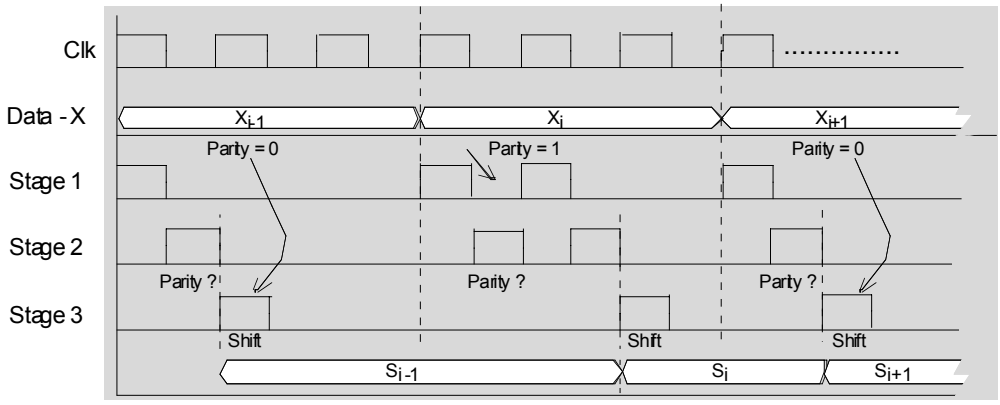



Figure 3.9. Timing diagram

Computing requires n iterations, but to calculate XYmodM will require 2n iterations. Parity evaluation and shift operation can be executed in the same execution cycle using a three

operands CSA. To calculate XYmodM is necessary to compute a modular multiplication of $XYR^{-1}$modM and $4^n$modM. This last quantity can be pre-calculated.

The structure of the proposed hardware is very simple and it does not use long distance interconnections. This simplicity determines the critical path length in the hardware. Now, recall that hardware clock speed is limited by the longest path in the circuit from input to output, so, the clock speed and overall time are also defined by this simplicity.

Generating the new partial product for the next iteration results in a critical path length of six gate delays. The clock cycle is approximately the sum of delay times associated with the gates on such path.

If long word operands must be multiplied, ie, 512 bits or more rather than 32 bits, the CPA array must be implemented using tree structures or Carry Look Ahead adders. If performance is more important than implementation cost, then Carry Look Ahead adders is more attractive. However, the implementation cost can be reduced specially when full custom VLSI is employed. Mentioned schemes are widely used for accelerating carry propagation.

The principal idea behind both schems is an attempt to generate all incoming carries (from CSA units) in parallel and avoid the need to wait until the correct carry propagates from the stage (FA) of the adder where it has been generated. If x', y' represent the input coming from CSA array, the basic cell of the both CLA adder and tree structures could be expressed by the equation: $c_{i+1} = x'_i y'_i + c_i (x'_i + y'_i)$ where:

$g_i = x'_i \wedge y'_i$ denotes the carry generation of the i-th bits.
$p_i = x'_i + y'_i$ denotes the carry propagation of the i-th bits.

There are stages in which a carry-out is generated regardless of the incomig carry, not requiring additional information on previous input digits. Other stages, are only capable of propagating the incoming carry, only the stage in which $x_i y_i = 0$ does not propagate of carries. Substituing $c_i = g_{i-1} + c_{i-1}p_{i-1}$ the equation above yields: $c_{i+1} = g_i + g_{i-1}p_i + c_{i-1} p_{i-1}p_i$ and making more subtitutions we have in general: $c_{i+1} = g_i + g_{i-1}P_i + g_{i-2}p_{i-1}p_i + ……..+ c_0 p_0 p_2 …….. p_i$.

This type of expression allows calculate all the carries in parallel from the original digits of $x_i'$ and $y_i'$. In figure 3.10, the tree structure for calculating $C_{16}$ is shown. This type of architecture is more deeply explained in [21].

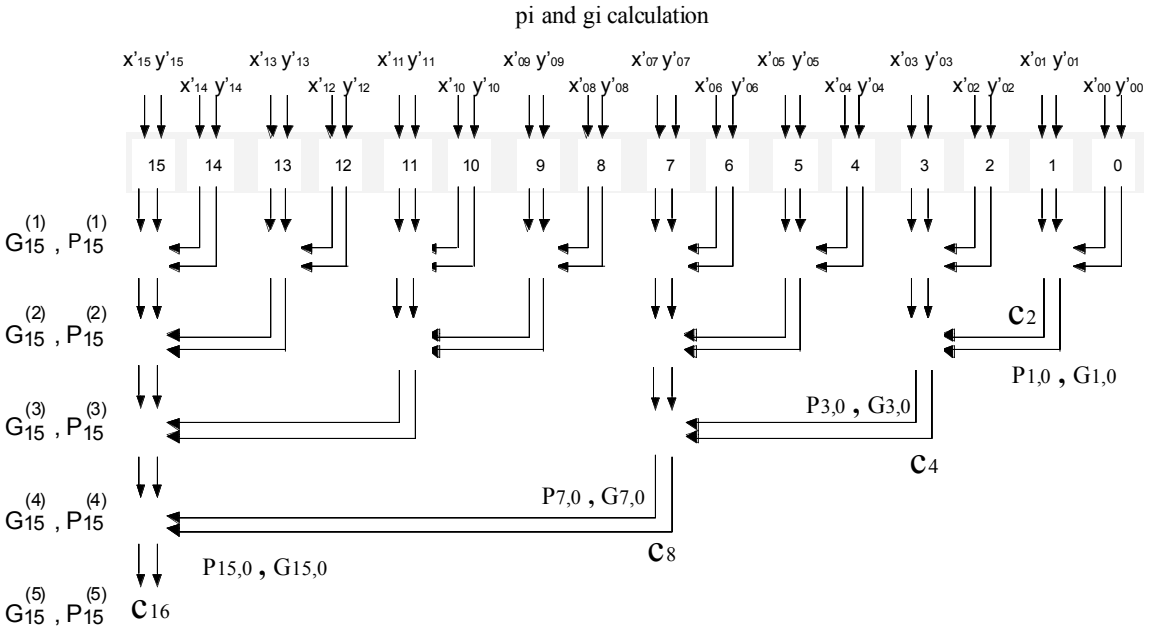



Figure 3.10. A tree structure for calculating $C_{16}$.

In the first level we calculate $P_{i,1}$ and $G_{i,1}$ or also denoted as $G_i^{(1)}$ and $P_i^{(1)}$, in the second level we calculate $P_{i,2}$ and $G_{i,2}$ or also denoted as $G_i^{(2)}$ and $P_i^{(2)}$ and so on. The notation is shown bellow:

| | |
|---|---|
| $G_{i,i} = g_i = x'_i \wedge y'_i$ | Carry generation of the i-th bit. |
| $P_{i,i} = p_i = x'_i \oplus y'_i$ | Carry propagation of the i-th bit. |
| $G_{i,k} = G_{i,j} \vee P_{i,j} \wedge G_{j-1,k}$ | Group carry generation between i and k bits, $n \geq i \geq j \geq k \geq 0$ |
| $P_{i,k} = P_{i,j} \wedge P_{j-1,k}$ | Group carry propagation from k to i bits. |
| $c_{i+1} = G_{i,0} \vee P_{i,0} \wedge c_0$ | Output carry of the i+1-bit |

Note that $P_{i,1}$ calculates $x'_i \oplus y'_i$ instead of $x'_i + y'_i$. All the circuits in the second through the fifth levels are identical allowing to implement a regular layout. We must remember that a regularity of the design and size of the required area determine the implementation cost. So this type of adder allows to take advantage of the architecture modularity.

### 3.3.5.12x12 bits Modular Multiplier Prototype.

In order to validate the architecture and demonstrate its performance, a 12x12-bits modular multiplier prototype was designed and fabricated using 0.6µm AMS-CMOS technology. The layout of the prototype presents a bit-slice structure, two slices are shown in figure 3.11.

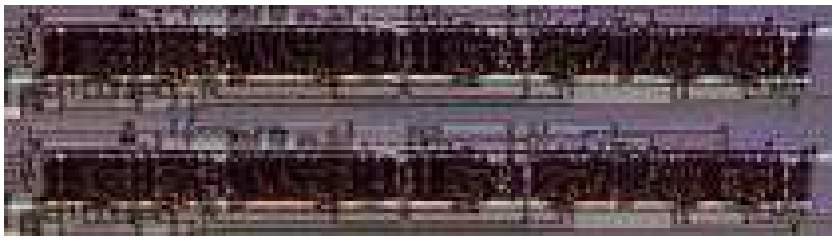
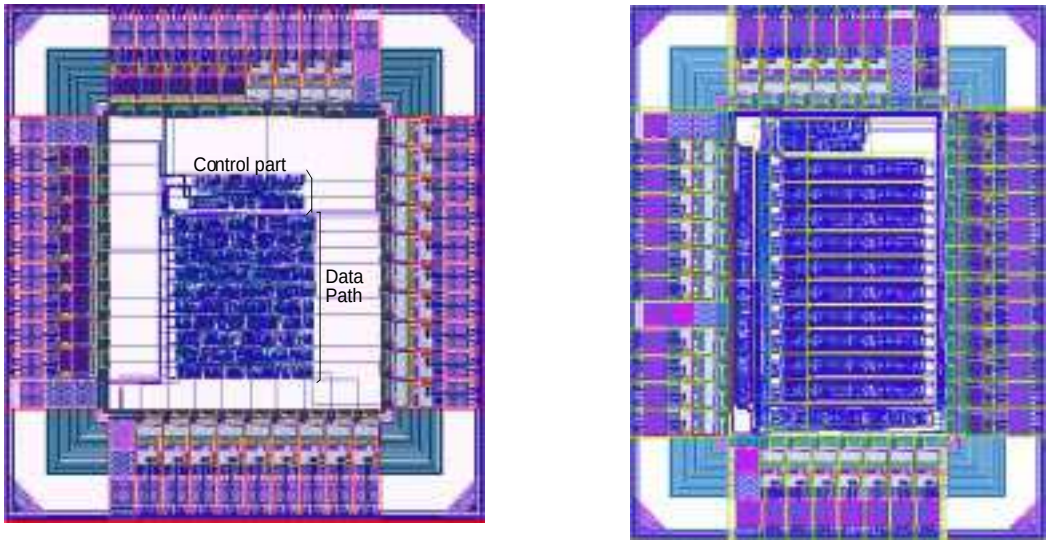


Figure 3.11. Bit slices

Layout automatic edition using placement and routing tools of CADENCE version 4.4.1 were used. In figure 3.12, a 12x12-bits modular multiplier graphic comparison between a standard cell based bit slice guided layout and a plain standard cell layout is shown. Including bonding pads, the bit-slice option, occupies an area of 3.85 $mm^2$ while for the plain standard cell layout the dimensions are 3.78 $mm^2$. For the first case, the core system occupies 1.24 $mm^2$ containing about 1020 equivalent gates. Nevertheless, the second option require less area, the high density bit-slices macro cells were used because they contain critical paths.




(a) (b)

Figure 3.12. 12x12-bits modular multiplier layout. (a) plain standard-cell. (b)standard-cell based bit-slice.

### 3.3.5.1. Control unit.

The control unit of the multiplier is shown in figure 3.13, it represents only 8% of the core size as can be seen in figure 3.12. The control unit contains sequencer, muxes, memory elements and combinational and sequencial logic. This unit generates all required control signal for operation. As can be seen it is not very complex, and does not represent an important hardware overhead.

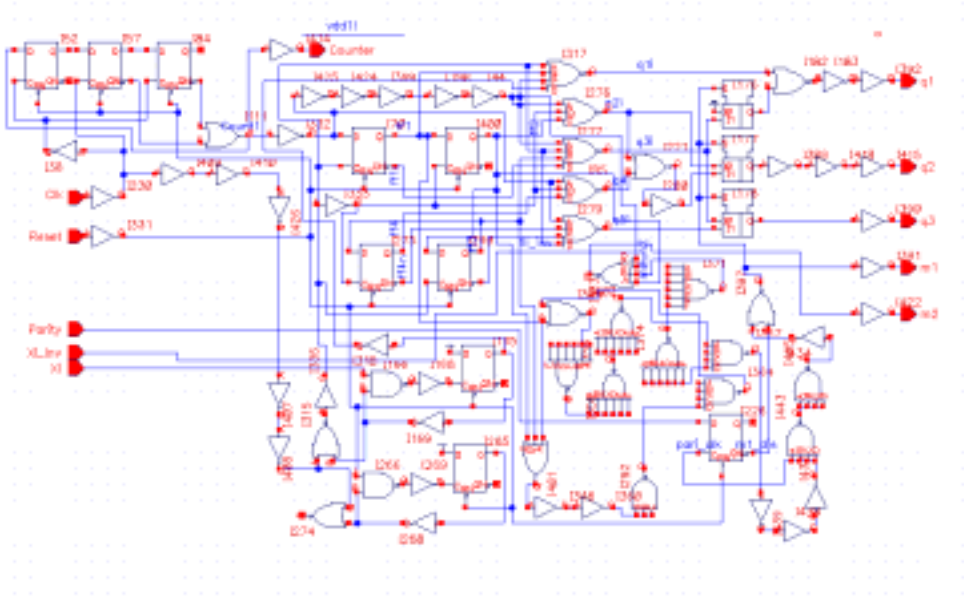


Figure 3.13. Control part

All required control signals are synchronised by *clk* signal. Sequencing signals *q1, q2, q3* and select signals *m1* and *m2* are generated depending of bit $x_i$ (*Xi* signal) and parity condition (*Parity* signal). The control unit is also fed by an asynchronous *Reset* which reachs all flip-flops and set them to reproduce a *zero* output, at time t=0.

*Counter* signal is internally generated. It is used to load a new bit of X, which appear serially , least significant bit first.

### 3.3.6. Simulation results

From behavioural post-simulation results using the 0.6µm CMOS-AMS standard cell library and Verilog behavioural simulator on CADENCE environment, the execution delay time, most of which is the delay time of adders is about 42 ns. As each execution cycle has at most three cycles clock, so a maximum frequency of 72 MHz using typical parameters was obtained. Two different examples of simulation results of the cell are depicted in figures 3.14(a) and 3.14(b).

(a)

(b)

Figure 3.14. Simulation results

Several simulations were done by using a reasonable number of pattern inputs. As shown in figures 3.14, where two examples of multiplication module 2773 are presented, we have for cases (a) and (b) M=2773, $R=2^{12}$ = 4096 and consequently $R^{-1}$mod2773 = 262. So, it is possible to verify that 4096x262mod2773 = 1.

For case (a), we have Y=1198 which is fed in parallel and X = 1134 = [010001101110]
For case (b), we have Y=1169 which is fed in parallel and X = 2070 = [100000010110]

Let remember that X is fed serialy least significant bit first. The binary representation of X can be observed in signal *Xi* which is applied during rising edge of *Count* signal. Note that the final result is $XYR^{-1}$mod M, for each case $R^{-1}$mod2773 = 262.

So, case (a) 1198x1134x262mod2773 we obtain Z = 1423
Case (b) 1169x2070x262mod2773 we obtain Z = 1697

### 3.3.7. Experimental results

In this section we will illustrate the behaviour of the proposed architecture [22]. The 12x12 bits modular multiplier prototype was fabricated using 0.6 µm CMOS-AMS technology. A die photo of this experimental circuit using standard cell based bit slice guided layout is shown in figure 3.15 The active area size is 1.33 x 0.93 $mm^2$ containing a number of transistors about 4100. Thus the density of transistors is 3.3 $k/mm^2$.

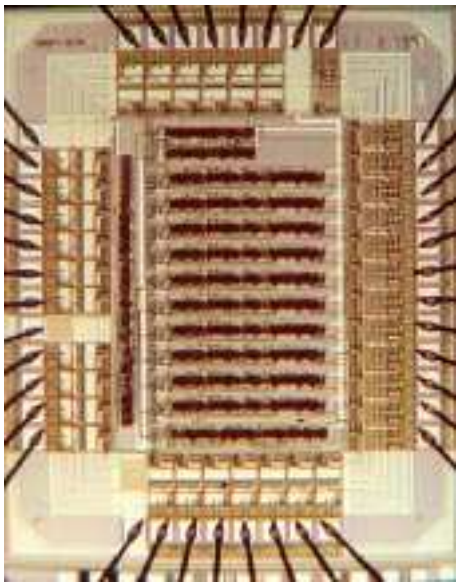


Figure 3.15. Test chip microphoto

Measures in term of functionality, performance and power consumption were done using a Test Station IMS ATS Blazer System. First, simple functional tests at different frequencies were done. In figure 3.16, an oscillograph screen illustrating the functional testing results at both 25 MHz and 50 MHz are presented. The *Clock* and *Count* input signals and the four least significant bits of the modular multiplication result are shown.

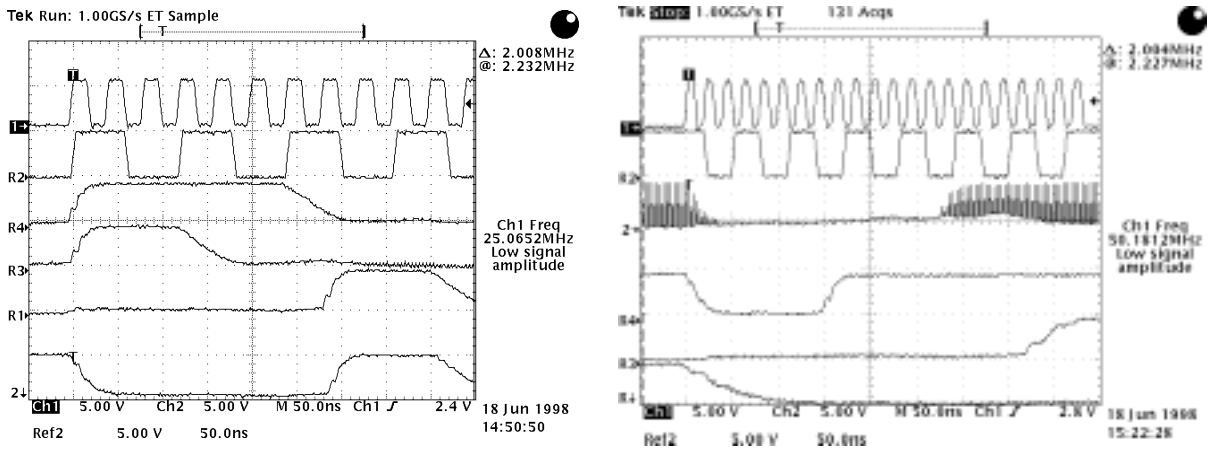



Figure 3.16. Functional testing results.

In order to evaluate the power consumption of the unit cell, the prototype was designed using three separated supply pads for the cell array core, control part and pads. Five prototypes were tested. The current consumption measures for both the control and operative parts at several frequencies are shown in tables IV and V respectively.

Table IV. Power consumption of the control part.

| Fq. | Prot. 1 | Prot. 2 | Prot. 3 | Prot. 4 | Prot. 5 |
|---|---|---|---|---|---|
| Mz | µA | µA | µA | µA | µA |
| | | | | | |
| 1 | 14 | 14.6 | 14.5 | 14.6 | 14.7 |
| 5 | 73 | 73 | 72 | 72 | 72 |
| 10 | 100 | 146 | 145 | 145 | 146 |
| 20 | 300 | 296 | 298 | 290 | 293 |
| 30 | 443 | 440 | 441 | 439 | 442 |
| 40 | 594 | 593 | 590 | 588 | 594 |
| 50 | 741 | 741 | 738 | 735 | 741 |
| 60 | 888 | 887 | 885 | 880 | 888 |
| 70 | 1.00 mA | 1.02 mA | 1.01 mA | 1.01 mA | 1.02 mA |
| 80 | 1.16 mA | 1.16 mA | 1.15 mA | 1.15 mA | 1.16 mA |
| 90 | 1.31 mA | 1.31 mA | 1.30 mA | 1.30 mA | 1.31 mA |
| 100 | 1.51 mA | 1.46 mA | 1.50 mA | 1.45 mA | 1.46 mA |

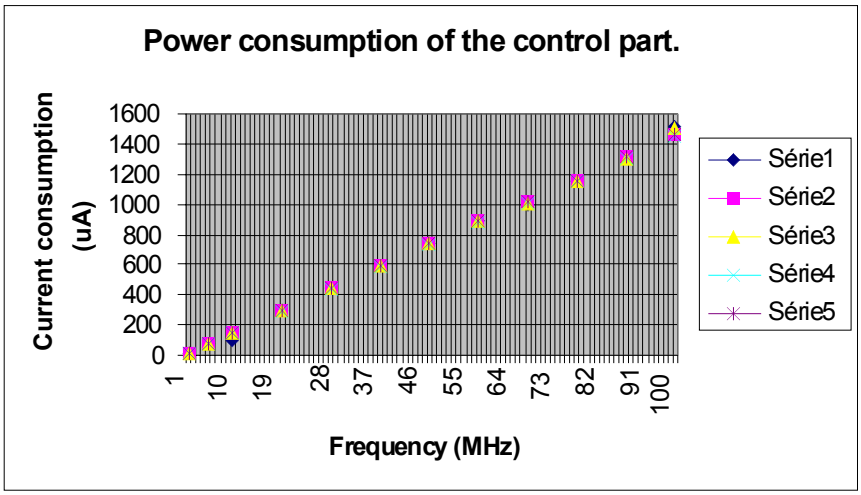



Table V. Power consumption of the operative part.

| Frq. | Prot. 1 | Prot. 2 | Prot. 3 | Prot. 4 | Prot. 5 |
|---|---|---|---|---|---|
| MHz | mA | mA | mA | mA | mA |
| 1 | 0.04 | 0.04 | 0.03 | 0.04 | 0.04 |
| 5 | 0.26 | 0.26 | 0.25 | 0.25 | 0.23 |
| 10 | 0.54 | 0.55 | 0.51 | 0.53 | 0.53 |
| 20 | 1.14 | 1.14 | 1.06 | 1.12 | 1.11 |
| 30 | 1.73 | 1.73 | 1.63 | 1.70 | 1.71 |
| 40 | 2.34 | 2.34 | 2.20 | 2.30 | 2.30 |
| 50 | 2.98 | 3.00 | 2.81 | 2.93 | 2.94 |
| 60 | 3.61 | 3.63 | 3.40 | 3.54 | 3.56 |
| 70 | 4.23 | 4.25 | 3.99 | 4.13 | 4.19 |
| 80 | 4.89 | 4.89 | 4.58 | 4.74 | 4.81 |
| 90 | 5.56 | 5.57 | 5.20 | 5.43 | 5.48 |
| 100 | 6.22 | 6.23 | 5.79 | 6.07 | 6.06 |

In table VI, the average global current consumption of the chip considering several operating frequencies is presented. All measures were done using a power supply voltage of 5V. As can be seen, the current consumption increases linearly in accordance with the increase in the frequency.

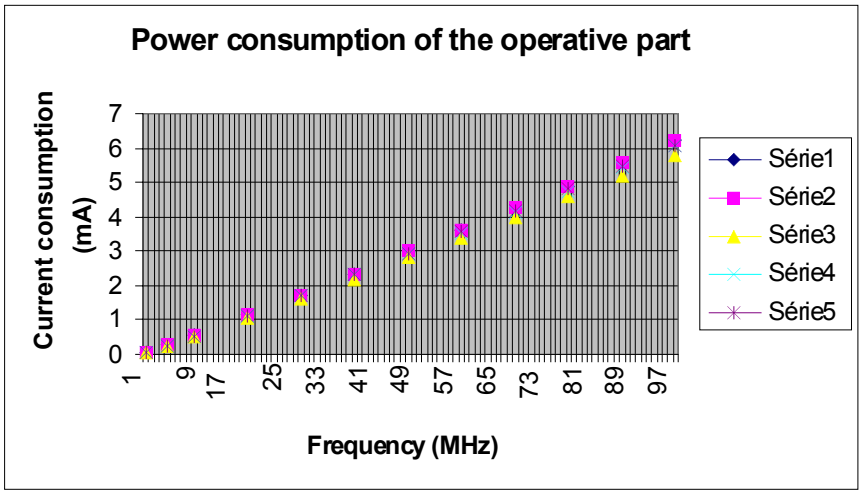
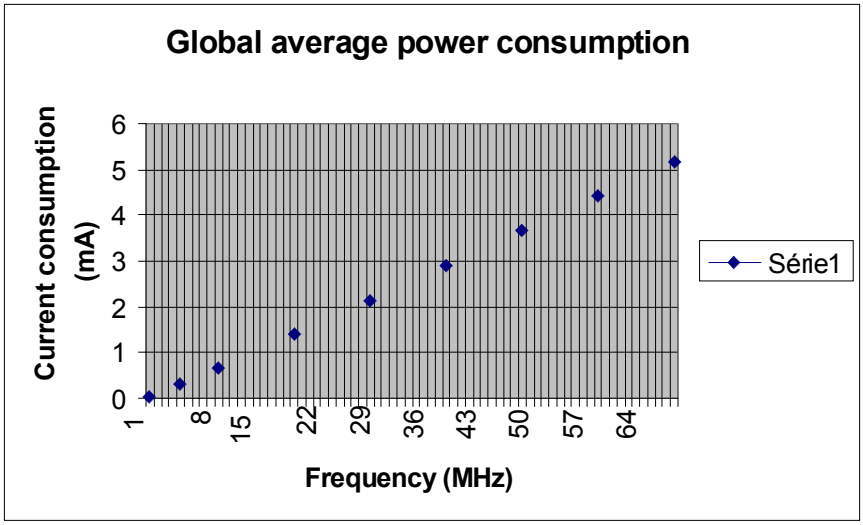


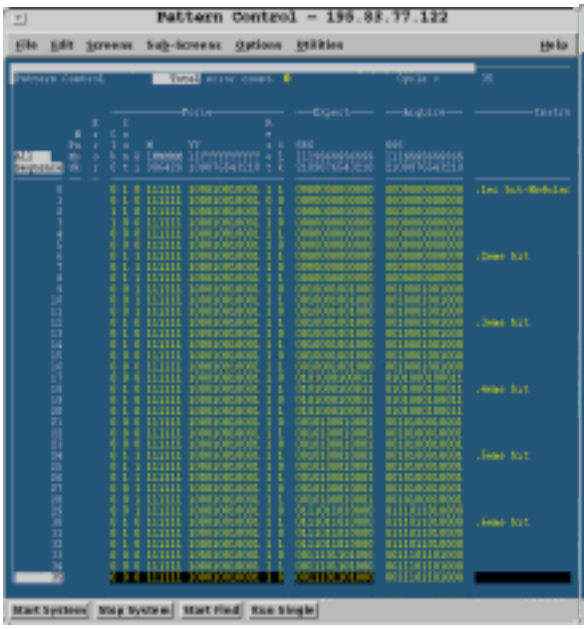

Table VI. Global power consumption

| F | MHz | 1 | 5 | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|---|---|---|---|---|---|---|---|---|---|---|
| P | nA | 0.052 | 0.322 | 0.668 | 1.409 | 2.141 | 2.887 | 3.671 | 4.433 | 5.17 |

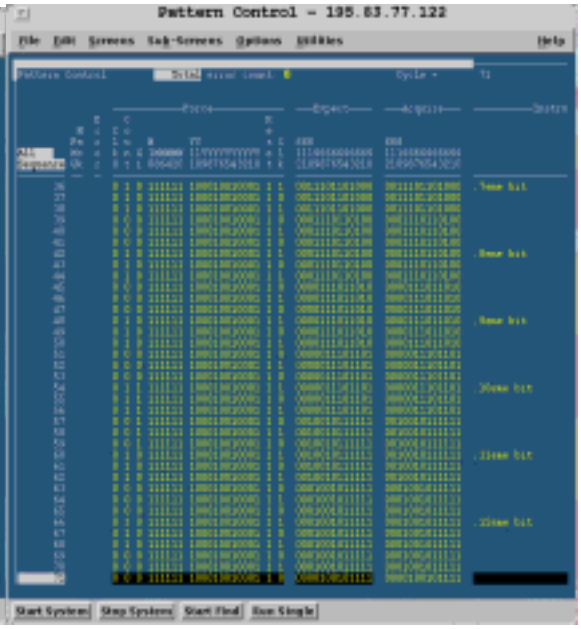



From experimental results and using a reasonable number of input patterns, the circuit was found to be operational at a maximum frequency of 71 MHz. Figures 3.17. (a)(b)(c), show the time scale, a set input patterns used during one of the several testing procedures and their waveform. All figures correspond to print screens of the Test Station System.

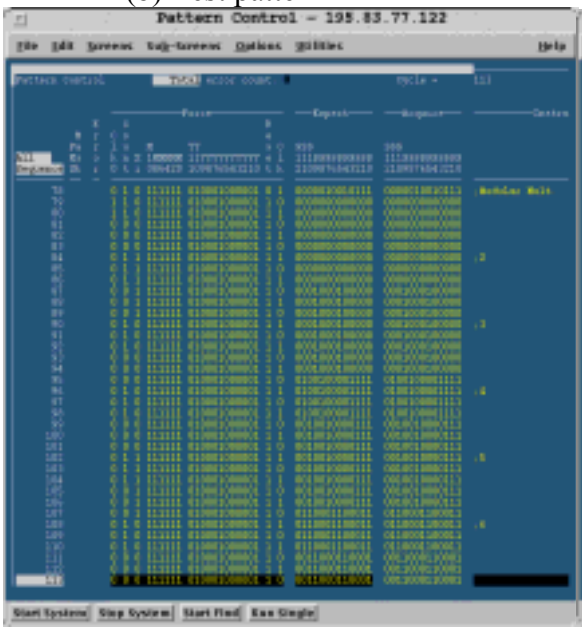

(a)



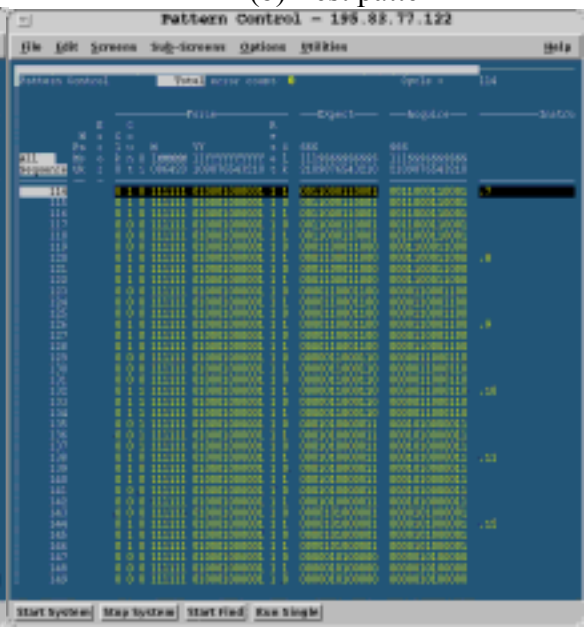

(b) Test pattern1



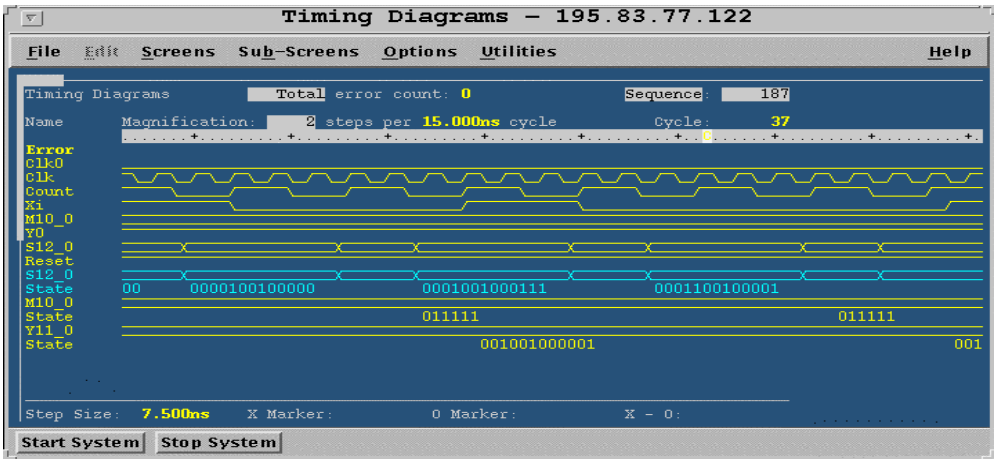

(b) Test pattern2




(b) Test pattern3

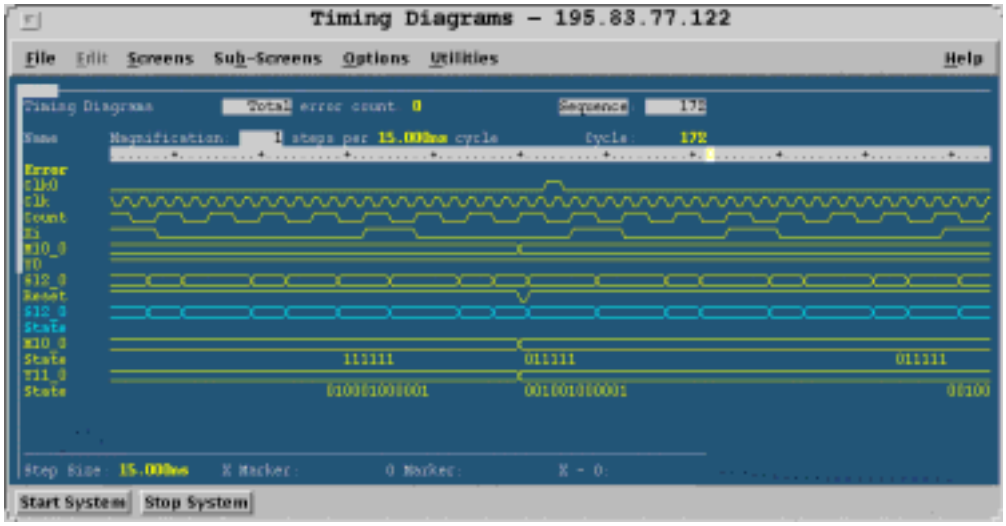



(b) Test pattern4




(c) Test pattern1


(c) Test pattern2




(c) Test pattern3

Figure 3.17. Test system screen plots.

Considering mentioned features a summary of the basic cell performance is given in table VII.

Table VII. Architecture performance [23].

| | |
|---|---|
| Modulus (M) | 12 bits $2^n > M > 2^{n-1}$ |
| Multiplicand (X) | 12 bits (Bin) |
| Multiplier(Y) | 12 bits (Bin) |
| Product | n + 1 bits ($XYR^{-1}$modM) |
| Multiplication time | 540 ns |
| Power dissipation | 93.5 mW (at 70 MHz) |
| Active area size | 1.33 x 0.93 $mm^2$ |
| Transistor count | 4100 |
| Frequency | 71 MHz |
| 128 bit [seg] | $5.76 \times 10^{-6}$ |
| Chip size | 3.85 $mm^2$. |
| Density of transistors | 3.3 k/$mm^2$. |
| Technology | 0.6µm-CMOS |

The generalised architecture can employ a n-operand adder module M realised using carry save adders where the longest path in the circuit from input to output involves only six gate delays. M is a dynamic range and $n = \log_2 M$. The architecture has the advantage that is easily expandable to larger bit-widths. For long word length numbers, ie, 512 bits the CPA array must be implemented using tree structures or Carry Look Ahead adders. Carry Look Ahead adders technique is more attractive if performance must be improved.

### 3.3.8. Conclusions.

An alternative architecture for computing modular multiplication based on Montgomery's algorithm has been presented. A 12x12-bits modular multiplier prototype has been designed and fabricated using AMS-0.6 µm CMOS technology. The architecture requires the modulus to be odd and the size of the modulus to be $2^n > M > 2^{n-1}$. Due to its simple logic, the proposed architecture present a good performance. This implementation requires only six gate delays. Carry delay must be added if a carry propagate adder is used. In order to optimise the critical paths, the layout of the prototype presents a high density standard cell based bit-slice guided layout structure. Including bonding pads the chip size is 3.85 $mm^2$. The active area size is 1.33 x 0.93 $mm^2$ containing about 4100 transistors. CADENCE tools version 4.4.1 were used.

In this architecture, the hardware clock speed is limited by the longest path in the circuit from input to output, being the shortest possible execution cycle approximately the sum of delay times associated with the gates on such path.

This implementation take the one argument in serial bit, least significant bit first, and produce the partial product in bit parallel form, giving the final result in $\theta(n)$ execution cycles where each one takes at most three clock cycles.

From experimental results we must conclude that an execution cycle of 42ns has been obtained. It means that the multiplier was found to be functional at a maximum frequency of 71 MHz. The whole process takes a time proportional to the number of digits in X. The standard cell gates used to implement the modular multiplier operate at a supply voltage of 5V.

The power dissipation is 93.5 mW at the frequency of 70 MHz. This architecture has the advantage that they are very simple, allowing a cellular construction and are easily expandable to larger bit-widths. For this reason, this architecture can be easily used in implementing cryptography systems to execute modular exponentiation of long wordlength numbers.

### 3.4. References.


[1] S. Eldridge, C. Walter, Hardware Implementation of Montgomery's Modular Multiplication Algorithm, IEEE Transaction on Computers, Vol. 42, No. 6, June 1993.

[2] E.F. Brickell, A Fast Modular Multiplication algorithm with applications to two key cryptography, Advances in cryptology, Proc. of Crypto 82, New York, 1982, pp51 - 60.

[3] C. Walter, S. Eldridge, A Verification of Brickell's Fast Modular Multiplication Algorithm, Inter. Journal Computer Math., Vol. 33, 1990, pp. 153-169.

[4] S. ELdridge, A Faster Modular Multiplication Algorithm, Inter. Journal Computer Math., Vol. 40, 1991, pp. 63-68 .

[5] C. Walter, Fast Modular Multiplication using 2-Power Radix, Inter. Journal Computer Math., Vol. 39, 1991, pp. 21-28.

[6] M. Kameyama, S. Wei, T. Higuchi, Design of an RSA encryption processor based on signed-digt multivalued arithmetic circuits, Syst. Comput. Japan, Vol. 21, pp. 21-31, 1990

[7] C. Walter, Fast Modular Multiplication by Operand Scaling, Crypto'91, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara California 1991, pp. 313-323.

[8] S. Even, Systolic Modular Multiplication, Crypto'90, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara California 1990, pp. 619-624.

[9] H. Morita, A Fast Modular Multiplication Algorithm based on a Higher Radix, Crypto'89, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara California 1989, pp. 387-399.

[10] M. Abe, H. Morita, Higher Radix Nonrestoring Modular Multiplication Algorithm and Public-key LSI Architecture with Limited Hardware Resources, Proc. Anacrypt'94, 1994, pp. 363-375.

[11] Charles C. Wang, T. K. Truong, Howard M. Shao, Leslie J. Deutsch, Jim K. Omura, Irving S. Reed: VLSI Architectures for Computing Multiplications and Inverses in GF(2m). IEEE Transactions on Computers 34(8): 709-717, 1985

[12] H. Sedlak, The RSA Cryptography Processor, Advances in Cryptology, Eurocrypt 87, June, 1987

[13] P. Barret, Implementing the Rivest, Shamir, and Adleman, Public Key Encryption Algorithm on a Standard Digital Processor, Advances in Cryptology, Crypto 86, pp. 311-323, 1986

[14] P.L. Montgomery, Modular Multiplication without trial division, Mathematics of Computation , 44, 1985, pp 519 - 521.

[15] J. Morechand, Projet Cryptographie: analyse de cryptoprocesseurs, Internal Report, Telecom, Paris, 1996.

[16] P.A. Findlay, B.A. Johnson, Modular Exponentiation Using Recursive Sums of Residues, Advances in Cryptology - Crypto'89, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara, California, 1989, pp. 371-386.

[17] C. Walter, Optimal parameters for on-line Arithmetic, Inter. Journal Computer Math.,Vol. 56, 1995, pp. 11-18.

[18] J.C. Bajard, L.S. Didier, P. Kornerup, A RNS Montgomery's Modular Multiplication, Journal IEEE Transaction on Computers, Vol. 47, No. 7 July, 1998.

[19] R. Bouraoui, A. Guyot, K. Khoumsi, Prototype of a circuit for the GCD and Extended GCD of very Large Numbers, Proc. of the 1991 International Conference on Microelectronics, ICM'91, Cairo, Egypte, 1991, pps. 83-86.

[20] R. Geiger, P. Allen, N. Strader, VLSI Design Techniques for Analog and Digital Circuits, McGraw Hill, New York, 1990.

[21] I. Koren, Computer Arithmetic Algorithms, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[22] A. Bernal, A. Guyot, Hardware for Computing Modular Multiplication Algorithm, 24$^{TH}$ European Solid-State Circuits Conference, La Hague, The Netherlands, 1998.

[23] A. Bernal, A. Guyot, Design of a Modular Multiplier based on Montgomery's Algorithm, XIII Conference on Design of Circuits and Integrated, Madrid, Spain, Nov., 1998.

# 4. Architecture for computing the Modular Exponentiation.

## 4.1 Introduction

As has been reviewed, modular exponentiation operation is the main and more frequently used function to process hidden information, it is a basic operation widely used in cryptography and constitutes a computational bottleneck in many protocols. It plays important roles in several public key cryptosystems where encryption and decryption processes require the modular exponentiation arithmetic function which is executed as multiple repetitive modular multiplication.

In this chapter, a summary of the most known modular exponentiation algorithms will be presented to finally describe the overall circuit implementing the algorithm for bit modular exponentiation. An alternative architecture for computing modular exponentiation based on generalised square-multiply binary method and using Montgomery's algorithm will be presented.

The architecture used to execute the modular exponentiation was verified by an experimental 32-bits exponentiation prototype which was designed, fabricated and tested using 0,6 um CMOS-AMS technology and including 8400 equivalent gates into an active area of 2.30 x 1.73 $mm^2$. In this chapter, the architecture, its operation, some simulation and experimental results for exponentiation operations are presented. The evaluation is provided according to functionality, power consumption and performance under the condition of 5V supply voltage.

## 4.2. Exponentiation algorithms.

Modular exponentiation of integers is the operation most widely used for several well known signature [1][2][3][4] and encryption [5][6][7] protocols in public-key cryptosystems [8][9]. Different approaches for executing modular exponentiation have been published and several algorithms for performing it already exist. Some authors have proposed multiple bit scan techniques [10][11] while others have used redundant number representation system [12] in order to avoid carries and implementations to reduce the computational complexity [13]. Similarly to multiplication case, the different algorithms can be divided into algorithms that are suitable for hardware implementation, such as [14][15][16], and those that are suitable for software implementation [15][17][18]. Due to the fact that our goal is oriented toward hardware implementation, we will focus on the more popular hardware suitable implementation algorithms.

Most of the common algorithms for modular exponentiation are based on the square-and-multiply method, such as the binary method. In general, exponentiation of the form $X^Y \text{mod} M$ is performed by repeated squaring operations, with conditional multiplication by the original X. So, if we can express the standard exponentiation function as:

$X^Y = (X^{Y/2})^2$ -----> if Y is even $\qquad$ $X^Y = X.X^{Y-1}$ -----> if Y is odd

Considering modular arithmetic properties, we can write the modular exponentiation as following:

$$X^Y \bmod M = ((X^{Y-I} \bmod M) \cdot (X^I \bmod M)) \bmod M \qquad (1)$$

Now, repeated squaring operations would be executed of that way:

$$(X^Y)^Z \text{mod} M = (X^Y \text{mod} M)^Z \text{mod} M \qquad (2)$$

So,

$$X^Y \text{mod} M = (X^2 \text{mod} M)^{Y/2} \text{mod} M$$
$$X^Y \text{mod} M = (X^2 \text{mod} M)^2 \text{mod} M)^{Y/4} \text{mod} M$$

For i iterations :

$$X^Y \text{mod} M = (((....(X^2 \text{mod } M)^2 \text{mod } M)^2 \text{mod } M)^2 ... i .... \text{mod } M)^2 \text{mod } M)^2)^{Y/2^i} \text{ mod } M$$

Some of the most known algorithms for calculating the modular exponentiation will be briefly reviewed.

**4.2.1. Square and Multiply algorithm or Binary method**

The best known algorithm for computing the modular exponentiation function $Z = X^y \text{mod} M$, is called the binary method [19], which is based on repeated squaring of X and multiplication whenever the corresponding bit of Y is 1. The binary method given below scans the bits of the exponent Y from left to right. Another version of this algorithm scans the bits of Y from right to left. This algorithm is briefly described:

Let n be the number of bits of Y, X and M

$$X = \sum_{i=0}^{n-1} x_i 2^i$$

$$Y = [y_{n-1}\, y_{n-2} y_{n-3} \ldots\ldots y_1 y_0] \qquad Y = \sum_{i=0}^{n-1} y_i\, 2^i$$

$y_i \in \{0,1\}$ and $n = [\log_2 Y] + 1$. The multiplication can be expressed as: $X^Y = \prod_{i=0}^{n-1} (x^{2^i})^{y_i}$.

where $y_{n-1}$ is the most significant bit, then the left to right version of the algorithm works as follows:

Input**:** X, Y, M, n where $n = \lceil \log_2 Y \rceil + 1$
Output: $Z = X^Y \text{ mod } M$

```
If y_{n-1} = 1 Z = X else Z = 1
        For i = n - 2 to  0  do {
                Z = Z*Z modM;
                if y_i = 1 then Z = Z*X modM;
                }
```

This algorithm requires on average 1.5n modular multiplication for an n-bit exponent. In the case of 512 bit integers, the algorithm performs on average 766 modular multiplication of 512 bits number [20].

### 4.2.2. M-ary method (MM)

The binary method can be generalised to the m-ary method [21] which scans the digits of Y expressed in radix *m*. Restricting the attention to the case when $m = 2^d$, the algorithm can be briefly described as follows: the exponent Y can be partitioned into k sections of *d* bit each for kd = n. If d does not divide n , the exponent is padded with at most d-1 zeros. So, we have:

$$X = \sum_{i=0}^{n-1} x_i 2^i \qquad Y = \sum_{i=0}^{n-1} y_i 2^i = \sum_{i=0}^{k-1} F^{(i)} 2^{id}$$

where $F^{(i)} = [y_{id+d-1}\, y_{id+d-2}\, y_{id+d-3} \ldots\ldots y_{id}] = \sum_{t=0}^{d-1} y_{id+t} 2^t$

First, the values of $V_j = V^j \bmod M$ are computed for $j = 2, 3 \ldots 2^d-1$ Then, the bits of Y are scanned *d* bits at a time from the most significant to the least significant, then the left to right version of the algorithm works as follows: Inputs: X, Y, M, n and *d*, where $n = [\log_2 Y]+1$ and n=k*d*, for $k \geq 1$.

Set $V_0 = 1$ and $V_1 = X$
  From j = 2 to $(2^d-1)$ do {
    $V_j = V_{j-1} X \bmod M$;
  From i=k-1 to 0 do
    $F^{(i)} = \sum_{t=0}^{d-1} y_{id+t} 2^t$
  Set $Z = V_{F^{(k-1)}}$
  From i=k-2 to 0 do
    From j=0 to (*d*-1) do
      $Z = Z * Z \bmod M$
    If $F^{(i)} \neq 0$ then $Z = Z * V_{F^{(i)}} \bmod M$
Halt

### 4.2.3. Koç's algorithm

Koç [21], has proposed an algorithm that makes use of high radix and bit recoding techniques to perform modular exponentiation. The algorithm is based on high radix representations due

to the fact that high radix methods with optimal choice of the radix provide significant reductions in the number of multiplication required for modular exponentiation. Besides, bit recoding techniques applied of Y are used to further reduce the total number of multiplication when Y has several hundred bits. This algorithm requires fewer multiplication than the binary method but suffers from excessive latency and a slow clock [19].

### 4.2.4. Findlay

Findlay [16] presents a method for computing a modular exponentiation useful in performing the RSA public key algorithm. The method uses conventional multiplication followed by a partial modular reduction based on sums of residues. The hardware implementation uses a serial data and one-dimensional semi-systolic array. A serial multiplier array is coupled with a unique serial sum-of-residues reduction array. The partitioned sums-of residues method can use look-up tables (table of residue values) which allow to speed-up the reduction calculation or use an additional architecture for sum-of-residues calculation; in each case the hardware overhead is important.

### 4.2.5. Brickell's algorithm

Considering that by storing a set of precomputed values it is possible to reduce the global number of multiplication needed, Brickell et al. [22] proposed a method of speeding up the modular exponentiation operation precomputing some specific values. So, precomputing and storing $X^{m_0}, X^{m_1}$ ……, $X^{m_{r-2}}, X^{m_{r-1}}$, for some integers $m_0$, $m_1$…… $m_{r-2}$, $m_{r-1}$, and finding a decomposition expression taking the form:

$$Y = \sum_{i=0}^{r-1} a_i \, m_i$$

where $0 \le a_i \le h$ for $0 \le i \le m$, then it is possible to compute:

$$X^Y = \prod_{d=1}^{h} C_d^{\,d} \qquad \text{where} \qquad C_d = \prod a_i = d^{X^{m_i}}.$$

### 4.2.6. Rooij algorithm

Rooij [23] presents an algorithm for exponentiation with precomputation which is based on two approaches, the first one splits the exponentiation into the product of a number of exponentiation with smaller exponents. The second one uses the techniques of vector addition chains to compute this product of powers. Mixing these approaches is possible to speed-up

the multiplication and squaring procedures. Nevertheless, this algorithm is slower than the method from Brickell.

### 4.2.7. Hamano's algorithm.

A Θ(n)-depth polynomial-size combinational circuit algorithm was proposed by Hamano [24] for computing n-bit modular exponentiation. The algorithm is a generalisation of the square and multiply method. The principal drawback consists in implementing the modular reduction. Additionally, a RNS to binary conversion must be done in each round.

### 4.2.8. Yongfei

The systolic modular exponentiation system presented by Yongfei [19] is based on k-*SR* representations and fast modular multiplication. The central point in the systolic approach is to ensure that once an information item is brought into the system it can be used effectively and repetitively while is being "pumped" from cell to cell through the system. The scheme exploits the fact that the k-*SR* algorithm is faster than the signed-digit algorithm because it needs less modular multiplication and no pre-computation of $M^{-1}$.

In table I, a comparison of some parameters between different mentioned algorithms is done.

Table I. Features of some modular exponential algorithms.

| **Algorithm** | **Depth** | **Notation** | **Technique** |
|---|---|---|---|
| Brickell | Ø(logn/loglogn) μp | Binary | Precomputation |
| Yongfei | …… | *k*-SR | Systolic arrays |
| Koç | …… | Not redundant | High radix-Bit recoding |
| Findlay | Ø(n) | Binary | Serial multiplier array |
| Hamano | (1+α)n/α | RNS | MODEXP based alg. |
| Square-bin | Ø(nlogn) | Binary | Repeated squaring |

### 4.3. Hardware for computing modular exponentiation.

The problem of efficiently evaluating powers has been widely studied. In the case of the RSA public key cryptosystem, where the algorithm is independent of X but depends on Y, the problem consists in executing the function $Z = X^Y \text{mod} M$, where M = pq for primes p and q. The modular exponentiation of the plain text (message) X, produces the cipher text Z using the encryption key Y. The decryption process is also a modular exponentiation using the

secret key. As mentioned, RSA cryptosystem is considered secure if both integers X and M have several hundred decimal digits.

Many research activities on hardware implementation oriented to speed up public key cryptosystems have also been done to introduce new strategies. Methods based on a higher radix [25], systolic architectures [19] or generalisation of the binary method [24] have been studied. Also, methods to reduce the computational complexity of the algorithms. Another one consists in utilising parallel techniques to perform faster implementation. Nevertheless, some of the proposed architectures demand excessive hardware resources or sophisticated implementation which is a constraint to install it in small size hardware.

As can be seen from table II, since some of the chip manufacturers give their speeds assuming the use of the Chinese Remainder Theorem while others do not, it is often difficult to compare the performance of the different chips.
In table II, a survey of hardware implementation of some RSA chips and their performance features is shown. More recently hardware implementations are presented in section 4.7.

Table II. A survey of Hardware implementations [26].

| Company | Year | Tech. | Bits/chip | Clock | baudrate | Clk/512bits |
|---|---|---|---|---|---|---|
| Sandia | 1981 | 3 µm | 168 | 4 MHz | 1.2k (336) | $4.0 \times 10^6$ |
| Bus. Sim. | 1985 | G. Array | 32 | 5 MHz | 3.8k (512) | $0.67 \times 10^6$ |
| AT &T | 1987 | 1.5 µm | 298 | 12 MHz | 7.7k (1024) | $0.4 \times 10^6$ |
| Cylink | 1987 | 1.5 µm | 1024 | 16 MHz | 3.4k (1024) | $1.2 \times 10^6$ |
| Cryptech | 1988 | G. Array | 120 | 14 MHz | 17k (512) | $0.4 \times 10^6$ |
| CENT | 1988 | 1.0 µm | 1024 | 25 MHz | 5.3k (512) | $2.3 \times 10^6$ |
| Brit. Tel. | 1988 | 2.5 µm | 256 | 10 MHz | 10.2k (256) | $1.0 \times 10^6$ |
| Plessy | 1989 | ------ | 512 | ------- | 10.2k (512) | ------- |
| Sandia | 1989 | 2.0 µm | 272 | 8 MHz | 10k (512) | $0.4 \times 10^6$ |
| Philips | 1989 | 1.2 µm | 512 | 16 MHz | 2k (512) | $4.1 \times 10^6$ |

The security of the cryptosystems is based on the difficulty of factoring integers. So, the word lengths and key lengths in modular exponentiation should be significantly greater than those used in conventional general purpose computer hardware, requiring typical word length around 256 bits or more, and it will grow in the future as the cryptanalysis makes progress. The requirements of the lengths makes RSA slow. When bulk data are transmitted in mobile telecommunication systems, cryptographic algorithms are also required to be fast and cheap

for the encryption and decryption of bulk data. Hence it is quite natural to speed up modular exponentiation.

The main idea to speed up modular exponentiation is reducing the number of multiplication and the depth of the path used to execute modular multiplication. For this reason it is important to investigate the smallest circuit depth achievable for this operation. As known, radix 2 arithmetic has a very short critical path. Based on mentioned idea, selecting radix 2 and mixing a generalised binary method with a three operands Carry Save Adder modular multiplier system functioning at high frequency, it is possible to meet the required features.

In next section we will present an alternative implementation describing the overall circuit to execute the algorithm for bit modular exponentiation. Operands are expressed in binary representation (radix 2). A generalised binary method to reduce the number of multiplication and a three operands carry save adder modular multiplier architecture to reduce the depth achievable for this operation are mixed. Some numerical example of modular exponentiation using this architecture, some simulation and experimental results are also presented.

### 4.3.1. Hardware implementation.

The core operation of exponentiation is modulo multiplication and as mentioned in last chapter, it can be performed using conventional multiplication where the multiplication and reduction can be combined or considered as separate tasks. That means that module reduction can be performed by division, which is slow, or by trial subtractions incorporated into the multiplication, that modify the partial products formed in the multiplication process. But it is suggested that neither of these is used as a reduction method [16].

Another method consists in using modular reduction. This method makes necessary some procedures to convert an integer to an M-residue field and vice versa. Module reduction is associative, so can be carried out at each stage to prevent the intermediate results from growing too large. Multiplication is best performed in a bit-serial form using a multiplier as described in last chapter. So, hardware implementation will be described for executing the

fast modular exponentiation based on both the extension of binary method and modular multiplication Montgomery's algorithm. A small depth combinational architecture is proposed for n-bit modular exponentiation.

In figure 4.1, a general block diagram of the modular exponentiation is shown. As can be seen, the system is based on a modular exponentiation operator which requires five inputs: X, Y, M, the partial products and the reduction factor. However, using multiplexing techniques, it is possible to use carry save adders for only three operands.

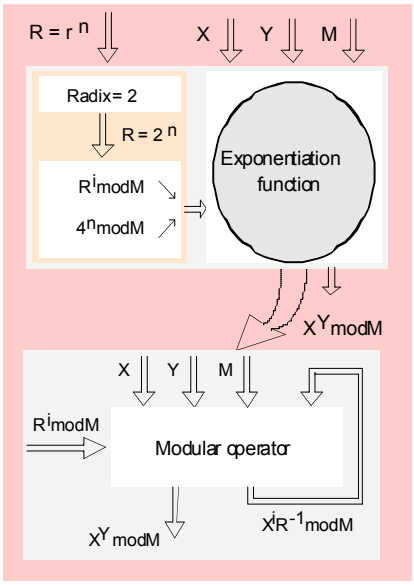



Figure 4.1. Block diagram of the modular exponentiation operator.

Before presenting the proposed architecture, it is necessary to review both the mathematical concepts used in the general procedure and the selected algorithms to be mixed.

### 4.3.2. Dynamic of the procedure.

The computation of $X^Y \text{mod} M$ can be executed for arbitrary integers X, Y and M represented as n-bit binary integer positives, within bounds $2^{n-1} < M < 2^n$ and $0 < X, Y < M$. For cryptography applications M is an odd integer. $M = pq$ being $p$ and $q$ very large random primes.

$R = 2^n$ - R is a power of radix and is prime to M. R is an integer satisfying $RR^{-1} \text{mod} M = 1$.

It means that inverse multiplicative of RmodM exists. See section 2.5.3.2 in chapter two.

Binary representation are denoted as:

$$M = \sum_{i=0}^{n-1} m_i 2^i \qquad M = ( m_0, m_1, .. m_{n-1} ) \text{ where each } m_i \text{ is 0 or 1}$$

$$X = \sum_{i=0}^{n-1} x_i 2^i \qquad X = ( x_0, x_1, .. x_{n-1} ) \text{ where each } x_i \text{ is 0 or 1}$$

$$Y = \sum_{i=0}^{n-1} y_i 2^i = \sum_{i=0}^{k-1} F^{(i)} 2^{id} \qquad Y = ( y_0, y_1, .. y_{n-1} ) \text{ where each } y_i \text{ is 0 or 1}$$

where $F^{(i)} = [y_{id+d-1}\, y_{id+d-2}\, y_{id+d-3} \ldots\ldots\, y_{id}] = \sum_{t=0}^{d-1} y_{id+t} 2^t$

In general, using the square-multiply method, the number of multiplications required is equal to the number of nonzero bits in the binary representation of Y. For this reason, it seems thus worthwhile to investigate techniques to recode Y in order to increase the number of zero bits in its representation. In this case, Y is expressed in radix *m*, we are going to restrict the attention to the case when $m = 2^d$. The algorithm is again presented below.

Set $V_0 = 1$ and $V_1 = X$

From j = 2 to $(2^d-1)$ do

$V_j = V_{j-1} X \bmod M$

From i=k-1 to 0 do

$F^{(i)} = \sum_{t=0}^{d-1} y_{id+t} 2^t$

Set $Z = V_{F(k-1)}$

From i=k-2 to 0 do

From j=0 to (*d*-1) do

Z = Z*ZmodM

If $F^{(i)} \neq 0$ then $Z = Z*V_{F^{(i)}}$ modM

Halt

From square-multiply binary method, the number of multiplications required by the algorithm to compute the modular exponentiation consists in two parts. First, $X^2$, $X^4$, $X^6$,….,$X^{2^{n-1}}$ (modM). This step requires (n-1) multiplication (squaring) operation. Second, we must continue to compute the exponentiation multiplying the partial result with $2^i$ power of X if the *i*th bit of the exponent is nonzero, this step requires also (n-1) multiplication. So, binary method requires in general a maximum number of multiplication equivalent to 2(n-1). Generalised method uses a smaller number of multiplications as will be discussed later.

Because of, modular exponentiation will be executed as a series of modular multiplications XWmodM, where X=W, it is necessary to define the bounding of the operands involved into the operation. So, if W denotes the partial product which is fed back to be multiplied by itself or by X, we can extend the Montgomery's restrictions to the previous partial product W:

So, let $W = (w_0, w_1, ....., w_{n-1})$ be such that $0 \leq W < M$, and let M(X.W) denote a Montgomery multiplication of X and W, now $M(X.W) \equiv XWR^{-1}$modM defines an *M-residue* to be a residue class modulo M. So, if we have $X' \equiv XR$modM, (which convert an integer X to an M-residue X') we can compute $Z^{-1}$, if we have the expression: $Z \equiv XW$modM; and T=X'W'; then $T \equiv XWR^2$modM.

The algorithm will compute $XWR^{-1}$mod M. This result is a Montgomery number. The generated sequence $S_i$ must satisfy the condition: $0 \leq M + W \leq 2M$. Each sequence $S_i$ represents the partial product result for each $x_i = 1$ of the multiplicand. So , we have:

$S_0 := 0$

If $(S_i + x_i W)$ is even.

$S_{i+1} = (S_i + x_i W)$ div 2

else

$$S_{i+1} = (S_i + x_i W + M) \text{ div } 2$$

By induction:

$$2^i \times S_i \equiv \sum_{j=0}^{n-1} (X_j\, 2^j)\, W \bmod M, \qquad S_i + 1 \text{ is an integer for } i = 0, 1, 2 \ldots\ldots n-1,$$

Consequently, the last expression can also be written as $2^i \times S_i \equiv (x_{i-1}\, x_{i-2} \ldots\, x_0) W \bmod M$, obtaining that:

$RS_n \equiv XW \bmod M$. and therefore $XWR^{-1} \bmod M$ is either $S_n$ or $S_n - M$,
with $0 \leq S_i < M + W < 2M$.

As was shown in chapter three, this method introduces an unwanted factor of $R^{-1} \bmod M$ into the product A.B. So, to convert a M-residue to an integer a division by $R^{-1} \bmod M$ is executed calculating $((XWR^{-1} \bmod M)(R^2 \bmod M)) \bmod M$. The final result would be $XW \bmod M$ which is a non Montgomery number. For radix=2 then: $R^2 \bmod M = (2^n)^2 \bmod M = 4^n \bmod M$.

The concluding multiplication by $R^{-1} \bmod M$ could be done using the same algorithm again by taking the output Sn and $R^{-1} \bmod M$ as the new multiplicands, the latter having been previously calculated once and for all by some other means. However, when further modular arithmetic is involved, it is better to start by using the algorithm to premultiply all inputs using $R^{-1} \bmod M$ as the other input.

Due to the fact that in each case the previous partial product is fed back to be multiplied by itself or by X. we have that, for Y even the next partial result to the power of four would be:

$X^4 \bmod M = ((X^2\, 2^{-n} \bmod M)(X^2\, 2^{-n} \bmod M) \bmod M$. Then: $Z = ((X^2\, 2^{-n}.X^2\, 2^{-n})\, 2^{-n} \bmod M$

When the partial result is fed back to the modular multiplier, the new result to the power of four would be n+2 size, then, we have:

$$= ((X^4\, 2^{-2n}.2^{-(n+1)}) \bmod M$$

The result to the power of eight would be n+3 size, then, we have:

$= (X^4\ 2^{-2n}.2^{-(n+1)}\ X^4\ 2^{-2n}.2^{-(n+1)}\ 2^{-(n+2)}) \mathrm{mod} M$

$= (X^8\ 2^{-4n}.2^{-2(n+1)}\ 2^{-(n+2)}) \mathrm{mod} M$

The result to the power of 16 would be n+4 size, then, we have

$= ((X^8\ 2^{-4n}.2^{-2(n+1)}\ 2^{-(n+2)})\ (X^8\ 2^{-4n}.2^{-2(n+1)}\ 2^{-(n+2)})\ 2^{-(n+3)}) \mathrm{mod} M$

$= (X^{16}\ 2^{-8n}.2^{-4(n+1)}\ 2^{-2(n+2)}\ 2^{-(n+3)}) \mathrm{mod} M$

The result to the power of 32 would be n+4 size, then, we have

$= ((X^{16}\ 2^{-8n}.2^{-4(n+1)}\ 2^{-2(n+2)}\ 2^{(n+3)})\ (X^{16}\ 2^{-8n}.2^{-4(n+1)}\ 2^{-2(n+2)}\ 2^{-(n+3)})\ 2^{-(n+4)}) \mathrm{mod} M$

$= (X^{32}\ 2^{-16n}.2^{-8(n+1)}\ 2^{-4(n+2)}\ 2^{-2(n+3)}\ 2^{-(n+4)}) \mathrm{mod} M$

In general, the result to the power of $2^i$ and with a size of n + i bits:

$(X^{2^i}\ 2^{-n2^{(i-1)}} 2^{-(n+1)\,2^{(i-2)}}\ 2^{-(n+2)\,2^{(i-3)}}\ \ldots\ldots\ 2^{-2(n+i-2)}\ 2^{-(n+i-1)}) \mathrm{mod} M$

From the last expression we obtain the bits number of the result vs. powering size as is shown in table III:

Table III. Bit number of the result vs. powering size.

| **Power($2^i$)** | **k** | **Function** | **Nbits** |
|---|---|---|---|
| 2 | 1 | $2^{-n}$ | n+1 |
| 4 | 2 | $2^{-2n}2^{-(n+1)}$ | n+2 |
| 8 | 3 | $2^{-4n}2^{-2(n+1)}2^{-(n+2)}$ | n+3 |
| 16 | 4 | $2^{-8n}2^{-4(n+1)}\ 2^{-2(n+2)}\ 2^{-(n+3)}$ | n+4 |
| 32 | 5 | $2^{-16n}2^{-8(n+1)}2^{-4(n+2)}2^{-2(n+3)}2^{-(n+4)}$ | n+5 |
| $2^i$ | i | $2^{-n2^{(i-1)}}2^{-(n+1)2^{(i-2)}}2^{-(n+2)2^{(i-3)}}$ ..…… $2^{-2(n+i-2)}2^{-(n+i-1)}$ | n+i |

So, for the big word length we can infer the values shown in table IV.

Table IV. Function for big word length

| **Power($2^i$)** | **i** | **Function** | **Nbits** |
|---|---|---|---|
| 128 | 7 | $2^{-2^6n}2^{-(n+1)2^5}2^{-(n+2)2^4}$ ……..$2^{-2(n+5)}2^{-(n+6)}$ | n+7 |
| 256 | 8 | $2^{-2^7n}2^{-(n+1)2^6}2^{-(n+2)\,2^5}$ ……..$2^{-2(n+6)}2^{-(n+9)}$ | n+8 |
| 512 | 9 | $2^{-2^8\,n}2^{-(n+1)2^7}2^{-(n+2)2^6}$ ……..$2^{-2(n+7)}2^{-(n+8)}$ | n+9 |
| 1024 | 10 | $2^{-2^9n}2^{-(n+1)2^8}2^{-(n+2)\,2^7}$ ……..$2^{-2(n+9)}2^{-(n+10)}$ | n+10 |

| | | | |
|---|---|---|---|
| 2048 | 11 | $2^{-2^{10}n}2^{-(n+1)2^9}2^{-(n+2)2^8}$ ……..$2^{-2(n+9)}2^{-(n+10)}$ | n+11 |
| 4096 | 12 | $2^{-2^{11}n}2^{-(n+1)2^{10}}2^{-(n+2)2^9}$……..$2^{-2(n+10)}2^{-(n+11)}$ | n+12 |

So, it is possible to convert a M-residue partial product to an integer multiplying each time by $4^n$modM and then to execute the next step. However, is also possible to execute the repeated squaring operations and then to multiply the final result by a factor $2^{p(n,k)}$, where k is the power number. In table V, some corrective factors are listed.

Table V. Corrective factors.

| i | Power($X^i$) | Corrective factor | Nbits |
|---|---|---|---|
| 2 | $X^2 = X^1 X^1$ | $R^2$modM | n+1 |
| 3 | $X^3 = X^2 X^1$ | $R^3$modM | n+1 |
| 4 | $X^4 = X^2 X^2$ | $R^4$modM | n+1 |
| 16 | $X^{16} = X^4 X^4$ | $R^{16}$modM | n+1 |
| 17 | $X^{17} = X^{16} X^1$ | $R^{17}$modM | n+1 |
| n | $X^n = X^{n-1} X^1$ | $R^n$modM | n+1 |

### 4.3.3. Architectural implications

The modular exponentiation system must compute $X^Y$modM. The common method for performing this operation is the *square and multiply* algorithm. In the square and multiply algorithm, the number of multiplication required for computing modular exponentiation is equal to the number of nonzero bits in the binary representation of Y. Some authors [21], are researching for new recoding strategies of the exponent in order to reduce the number of non zero values in Y, however the overhead hardware requirements for doing that are not attractive.

So, considering that a generalised square multiply method requires a smaller number of multiplication for executing the modular exponentiation, in this work an alternative hardware implementation which performs in effect modular reduction systems based on Montgomery's method is proposed. A modular exponentiation function mixes both the generalised method and the proposed alternative architecture used to execute the Montgomery's algorithm.

The design of a chip for performing cryptographic operations based on this architecture, is simple and allows very high clock rates. These two advantages make such a chip competitive with currently known designs.

As can be seen from the algorithm, the exponent Y expressed in radix *m* is partitioned into k sections of *d* bit each for k*d* = n. If *d* does not divide n, the exponent is padded with at most d-1 zeros.

First, the values of $V_j = V^j \text{mod} M$ are computed for j = 2 , 3 ..$2^d$-1. Then, the bits of Y are scanned *d* bits at a time from the most significant to the least significant.

As will be discussed later, due to the fact that it is possible to select *d* for a given **n** such that the maximum number of multiplications is minimised, the cache memory size required for storing the partial modular products computed for j = 2 , 3 ..$2^d$-1 can be also defined. The size of the cache memory and the value of *d* optimum denoted as *d** for a given **n** can be seen in table VI [28]. Note that the cache memory does not require a very large storage capacity. The dependence between **n** and *d** will be discussed later.

Table VI. Cache size

| **n** | ***d**** | **cache** |
|---|---|---|
| 32 | 3 | 8 |
| 64 | 3 | 8 |
| 128 | 4 | 16 |
| 256 | 4 | 16 |
| 512 | 5 | 32 |
| 1024 | 6 | 64 |
| 2048 | 6 | 64 |

Additionally, as modular multiplication is done by repeated cycles involving shifting and addition together with a simultaneous modular subtraction in order to satisfy the condition $0 \leq S_i < M + W < 2M$, a comparator/subtractor is needed. In the modular multiplier both the modulus (M) and the multiplier (X) are fed parallelly to datapath while the multiplicand (W) is fed serially, so a parallel to serial converter is also required.

### 4.3.4. Modular exponentiation architecture.

The architecture of the proposed hardware is very simple and it uses not long distance communications. Basically, the architecture presents as data inputs X, Y, M, R and four control signals: *Reset, clk* and *start* which are used to initialise the procedure, and *Load_inv* signal used to load the inverse multiplicative number. Left to right version of the algorithm requires as inputs: X, Y, M, n and *d*, where $n = \lceil \log_2 Y \rceil + 1$ and $n = kd$, for $k \geq 1$, nevertheless, it is possible to define an optimum value of *d* (*d**) for different n = 4, 8, 16…1024 in order to obtain a minimum number of multiplications required by the algorithm, we can consider that this value is predefined according to the bits length.

One output signal *Stop* is supplied by the control part when modular exponentiation is computed. This flag indicates that the modular reduction must be done, it means the final result must be multiplied by the corrective factor. In figure 4.2, the external signals are shown.

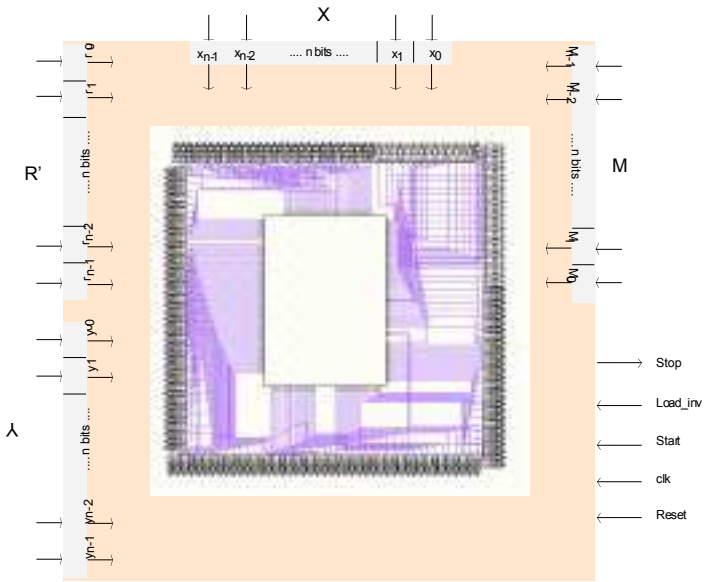



Figure 4.2. External signals.

All flip-flops are set to reproduce a *zero* output, at time t = 1, by a Reset signal. The *start* synchronised signal initialises the control part to generate the internal control lines necessary to execute the modular exponentiation.

A block diagram of a practical hardware modulo exponentiation system is presented in figure 4.3. The architecture consists of a set of registers/multiplexer, a CLA comparator/subtractor, a small size RAM, a parallel to serial transformer, a modular multiplier and the control part.

From binary algorithm it is possible to observe that two steps can be clearly defined. In the first one, the values $Vj = V^j \text{mod} M$ are computed for $j = 2, 3 .. 2^d-1$. It means $2^d-1$ modular multiplication could be done by taking X and each partial product $X^j$ as the new multiplicands.

Also, in this step each partial product $V^j$ would be stored into the intermediary RAM using as field address the content of F(i), as can be seen in figure 4.4. So, $2^d-2$ read memory operations must be done. The partial products will be used to calculate $Z * V_{F(i)} \text{ mod} M$.

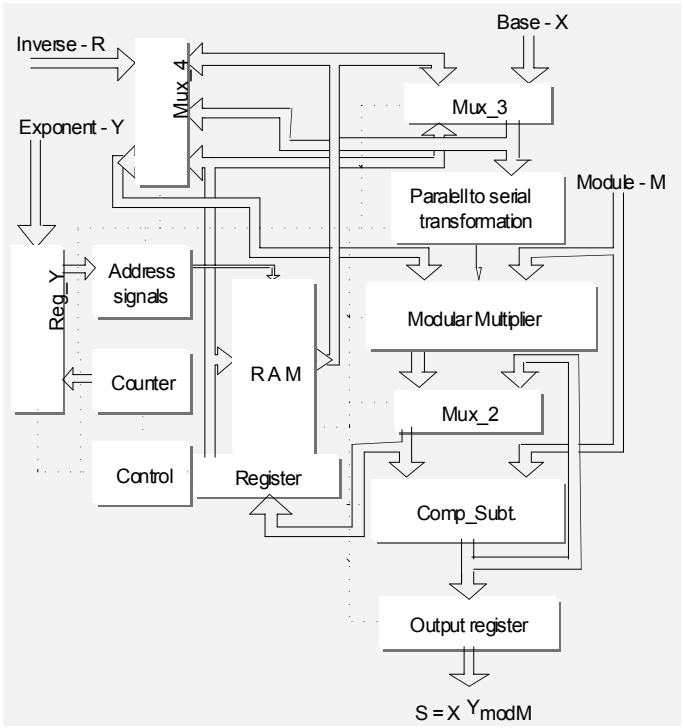



Figure 4.3. Modular Exponentiation System Block diagram.

In the second step, each partial result is raised to the $2^d$ power, requiring (k-1)*d* multiplication and then is multiplied with $V_{F(i)}$, where $F^{(i)}$ is the value of the current bit section of the

exponent Y. If $F^{(i)} = 0$, modular multiplication is not executed, so, this procedure requires at most (k-1) multiplication.

Summing the number of multiplication required, we obtain:.

$$T^{max}(n, d) = 2^d - 2 + (k-1)d + (k-1) = n + n/d + 2^d - d - 3 \qquad \text{since } n = kd$$

From this expression, it is possible to select $d$ for a given n such that $T^{max}(n, d)$ is minimised. Derivating the expression with respect to $d$ and in order to find the optimal value of $d$ minimising the maximum number of multiplication executed by the algorithm, we need to solve:

$$\partial T^{max}(n, d)/\partial d = - n/d^2 + 2^d \log_e 2 - 1 = 0$$

By enumeration, a value of $d = d^*$ optimum can be found such that $\partial T^{max}(n, d^*)$ is as close to zero as possible.

In table VII, the optimum values of $d^*$ for n =4, 8, 16…1024, together with the values of $T^{max}(n)$ and $T^{max}(n, d^*)$ for binary method (BM) and generalised method (MM) are shown. $T^{max}$ represents the maximum number of multiplications required by the radix m algorithm.

Table VII. Maximum number of multiplications.

| | **BM** | **MM** | |
|---|---|---|---|
| N | $T^{max}(n)$ | **d*** | $T^{max}(n,$ **d*)** |
| 4 | 6 | 2 | 5 |
| 8 | 14 | 2 | 11 |
| 16 | 30 | 2,3 | 23 |
| . | . | . | . |
| . | . | . | . |
| 128 | 191 | 3,4 | 167 |
| 256 | 383 | 4 | 325 |
| 512 | 767 | 5 | 635 |
| 1024 | 2046 | 6 | 1250 |

| | | | |
|---|---|---|---|
| . | . | . | . |
| n | 2(n-1) | | $n+n/2+2^d-d-3$ |

Considering that, if $F^{(i)} = 0$, modular multiplication is not executed, it is possible to find an expression to denote the average number of multiplications and subsequently minimised in order to find an optimum d*****. In table VIII, the average number of multiplications required by algorithms binary method (BM) and generalised square multiply (MM) with the optimum values of d, are presented [21].

Table VIII. Average number of multiplications.

| | **BM** | **MM** | |
|---|---|---|---|
| n | $T^{ave}$(n**)** | **d*** | $T^{ave}$(n, **d***) |
| 4 | 5 | 2 | 5 |
| 8 | 11 | 2 | 11 |
| 16 | 23 | 2,3 | 23 |
| . | . | . | . |
| . | . | . | . |
| 128 | 191 | 3,4 | 167 |
| 256 | 383 | 4 | 325 |
| 512 | 767 | 5 | 635 |
| 1024 | 1535 | 6 | 1250 |
| . | . | . | . |
| n | 3/2(n-1) | | $n+[(n/d)-1][1-(1/2^d)]+2^d- d - 2$ |

Square and multiply algorithm for modular exponentiation requires on average 1.5n modular multiplication for an n-bit exponent [20]. In the case of 512-bit integers, the algorithm performs on average 766 modular multiplications.

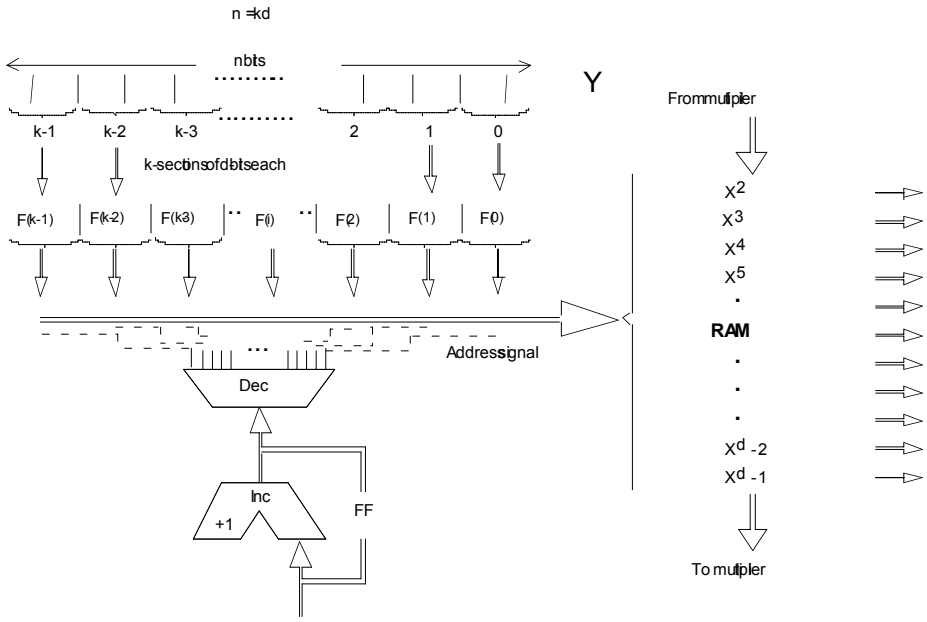



Figure 4.4. Exponent register.

As is shown in figure 4.4, the content of each n field $F^{(k-i)}$ is interpreted as an address signal for storing the partial modular product. A decoder and an incrementer can be used to facilitate this procedure. As mentioned, the number of multiplication required in this step for any value of the exponent is $2^d$ - 2, for this reason $2^d$ - 2 access cache memory must be done for storing $V^j$modM values coming from the comparator/subtractor. The incrementer/decoder block allows scan each one of the k-sections into the exponent Y to generate the RAM accessing addressing signal where each $V^j$modM partial product will be stored.

The modulus (M) and the partial products (W) are fed concurrently to modular multiplier while the multiplicand which can be either X or $V^j$ or $V_{F(i)}$, is fed serially through Mux_3, being processed only for the parallel to serial transformation unit. The parallel to serial transformation unit is responsible for generating one output serial signal which is one of the operands to be fed to modular multiplier. The parallel to serial transformation and modular multiplication are concurrently performed.

Each partial result of the modular multiplication is compared with modulus using the overflow signal supplied by subtractor. Mux_2 and a Carry Look Ahead (CLA) subtractor act

when running the modular multiplication algorithm, performing extra subtractions of the original modulus as necessary after the main loop to obtain the least non-negative residue, the result can be stored into the RAM or fed back to modular multiplier through selectors Mux_3 and Mux_4. Nevertheless, Montgomery multiplication produces a result less than twice the modulus, so only at most one subtraction is needed

During the second step and depending of the $F^{(i)}$, several cache memory access are executed. In general, access memory are done if $F^{(i)} \neq 0$. When $F^{(i)} = 0$, modular multiplication is not performed. If $F^{(i)} = 1$, a modular multiplication occurs, but cache memory is not accessed, due to the new operand is X. In this step, each access memory requires a write operation in order to transfer the data from RAM, through Mux_4, and through the parallel to serial transformation unit in sequence to be finally supplied to modular multiplier. As can be seen, at most (k-1) access memory must be executed.

Finally, a flag (*Stop* signal) is used to indicate the *end* status of the modular exponentiation function validating the input signal *Load_inv* to load the corrective factor.

### 4.4. 32-bits prototype design.

In order to validate the architecture, a 32-bits modular exponentiation system has been designed and fabricated by using 0.6um CMOS-AMS technology. In figure 4.5, a capture schematic of the system is shown.

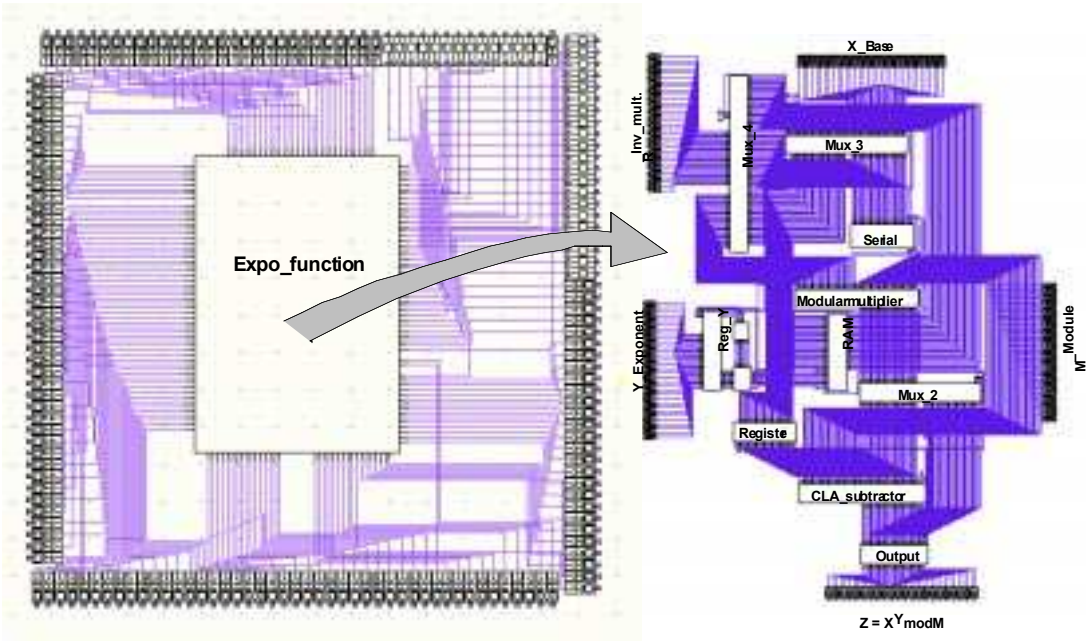


Figure 4.5. Schematic capture of the system.

The evaluation of the architecture will be provided according to functionality, power consumption and performance under the condition of 5V supply voltage. Excepting registers and multiplexers blocks which present a standard operation, all other functional blocks will be briefly discussed as follows:

### 4.4.1. Modular Multiplier.

Modular exponentiation is executed by repeated modular multiplication. As mentioned in chapter 3, the multiplier design is oriented to fast execution of modular multiplication. As can be seen from figure 4.6, two operands are fed parallely while one second operand coming from *partoser1* unit is fed serially. The modulus is always fed in parallel.

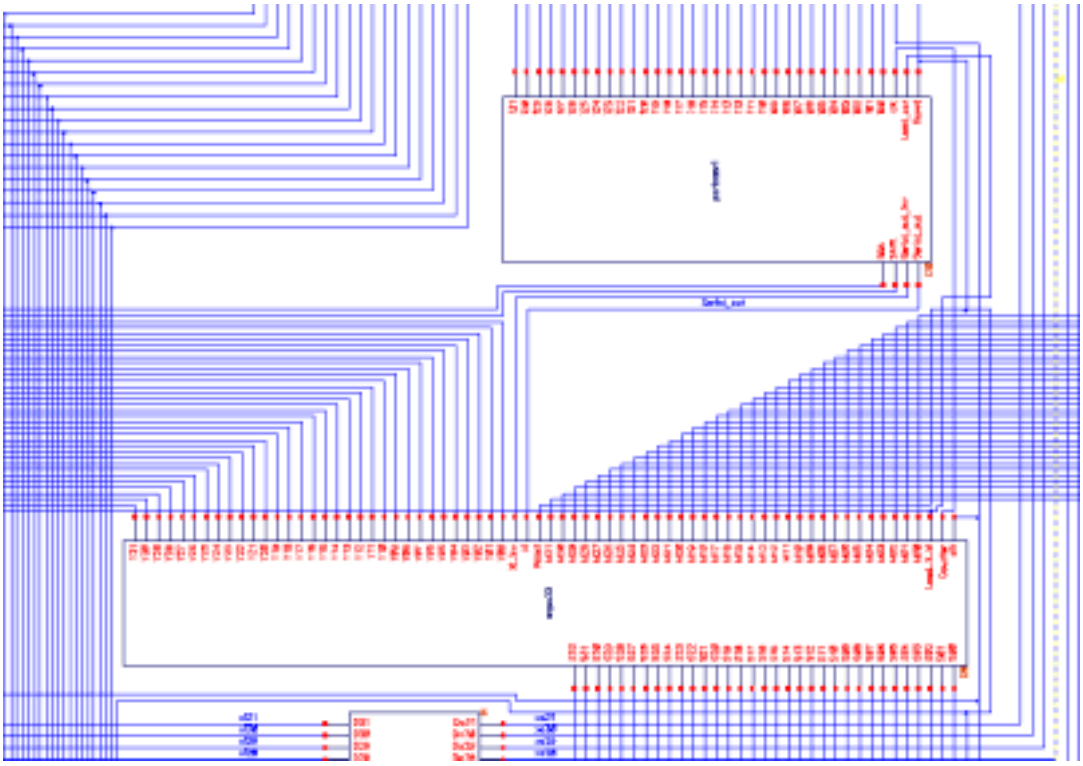


Figure 4.6. Multiplier and parallel to serial unit.

The modular multiplier (*expo32*) consists of 32 identical cells as can be seen in figure 4.7. The design of the cell is depicted in figure 3.8(a), in chapter 3. The inputs to multiplier are supplied from different functional blocks.

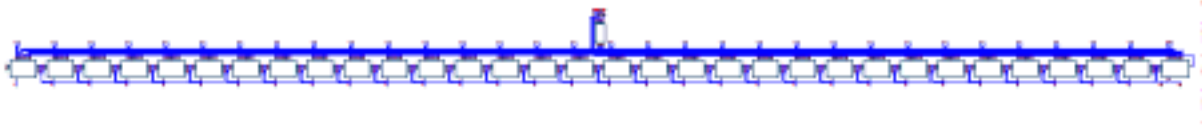


Figure 4.7. Modular multiplier diagram.

During the first step of the algorithm, the operands involved are X, which is an external input, and $V^j$ which is the output of the CLA_subtractor. In the second step, the operands are $V_{F^{(i)}}$ and depending of $F^{(i)}$, $V^j$ or X. In each case, X (first case) or $V_{F^{(i)}}$ (second one) is fed parallely to *partoser1* conversion unit, and for every $0 \leq i \leq n-1$, an output serial signal (*serial_out*) is carried out to modular multiplier. The logic needed to implement this conversion is shown in figure 4.8.

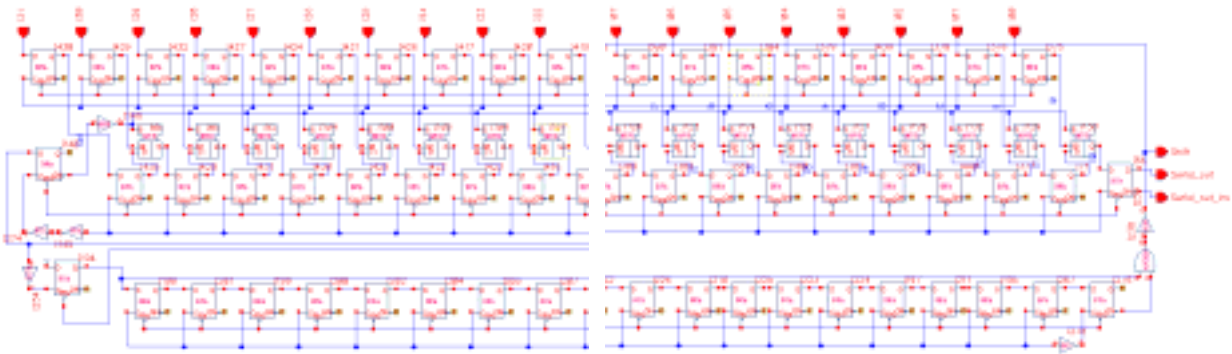


Figure 4.8. Parallel to serial unit

Once, the bits of X or $V^j$ are available in the input register, a control signal (*Load_X_M*) loads the partial product and the *shift* internal signal is activated to start the conversion procedure. In Figure 4.9 it is possible to see both partial product (*Part_P* - in binary) and Serial_out signals (*al_out*) which correspond to the serial output related to *Part_p* parallel input. As can be seen, each serial bit, -least significant first-, is used to produce in parallel a multiplication partial product. Thus a flag is used to indicate to the control part the end of the operation (*Mult*). This flag is activated by shift internal signal.

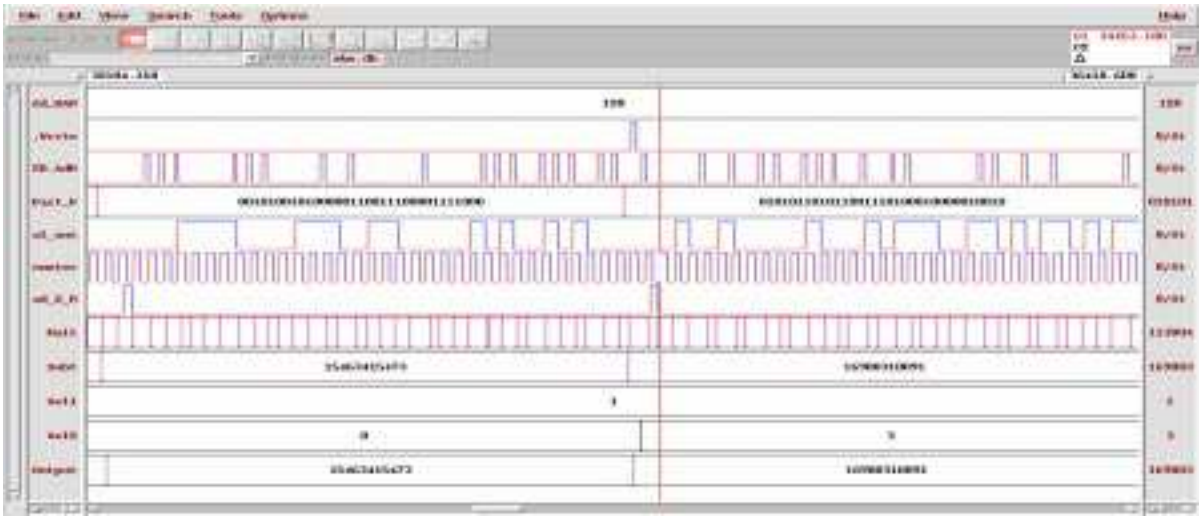


Figure 4.9. Parallel to serial conversion.

### 4.4.2. Exponent Y register and control part.

Specifications for the operation of the exponentiation system are more frequently given for the exponent Y register. The exponent Y defines the number of multiplications, the number of memory accesses for executing read or write operation. The connectivity between the exponent Y register, RAM and the control part is presented in figure 4.10.

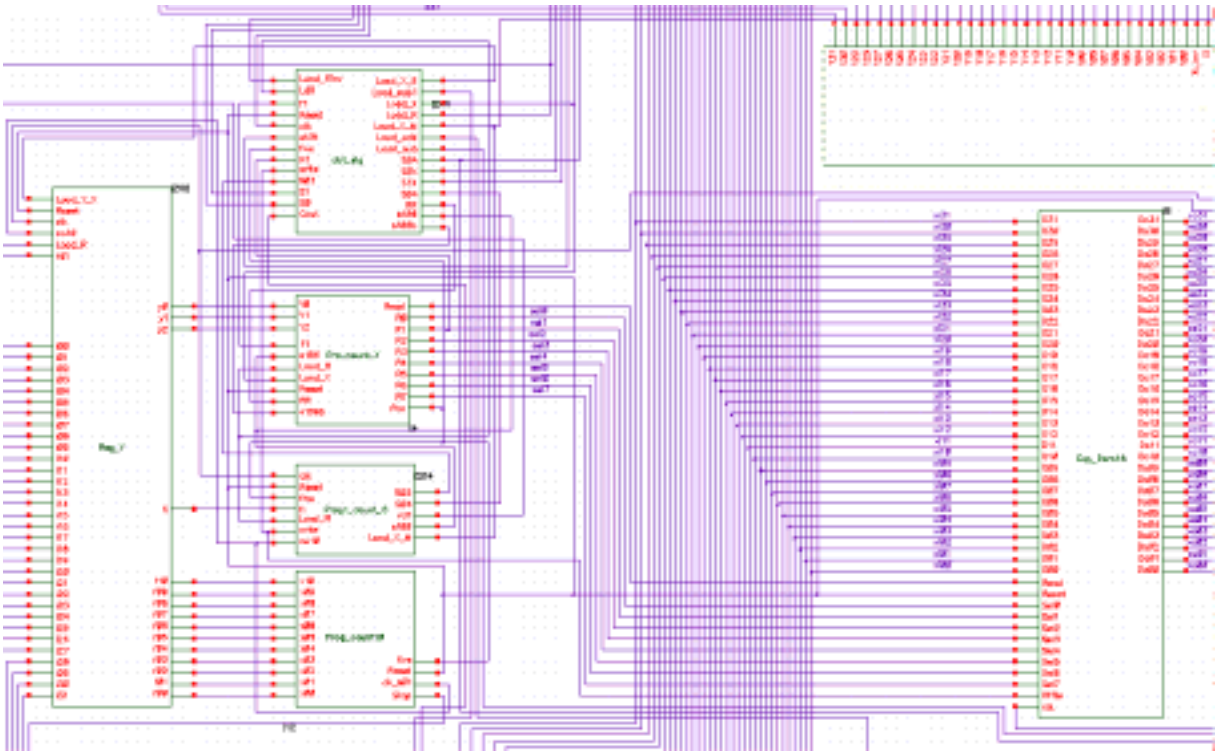


Figure 4.10. Exponent register, control part and intermediary memory.

This part of the system is composed by sequential machines implemented as forward path containing combinational logic and several feedback paths that include storage elements as flip-flops and latches. The control part is composed by five different state machines included into four blocks: *Prog_count10*, *Pro_count_Y, Prog1_count_6*, and *ctrl_sig*. These state machines include clocked input drivers and clocked output buffers. They generate all required control signals for operation.

In its simplest form, *Ctrl_sig* block is responsible for generating the main clocked control signals needed to realise correctly the required function. Register and multiplexers load signals, multiplexer selection signals, write and several secondary control signals are distributed from it to overall system. Secondary control signals are used as command lines to activate sequentially other state machines. In this block are generated six primary load

signals: *Load_X*, *Load_R*, *Load_X_M*, *Load_sub*, *Load_add* and *Load_sub1* which are used to load new data into the registers.

Besides, selection signals of the multiplexers: *S1, S0, S1X, S0X* and *S0A* are also generated. Each one of this load and selection signals represented as Sel1 [*S1, S0*], Sel2 [*S1X, S2X*] and *S0A* are generated according to several feedback informations coming from other functional blocks as multiplier (*Parity*), Subtractor (*Cout*) or Serial-bit processing unit (*Shift*) between others. In figure 4.11, the main control signal are illustrated.

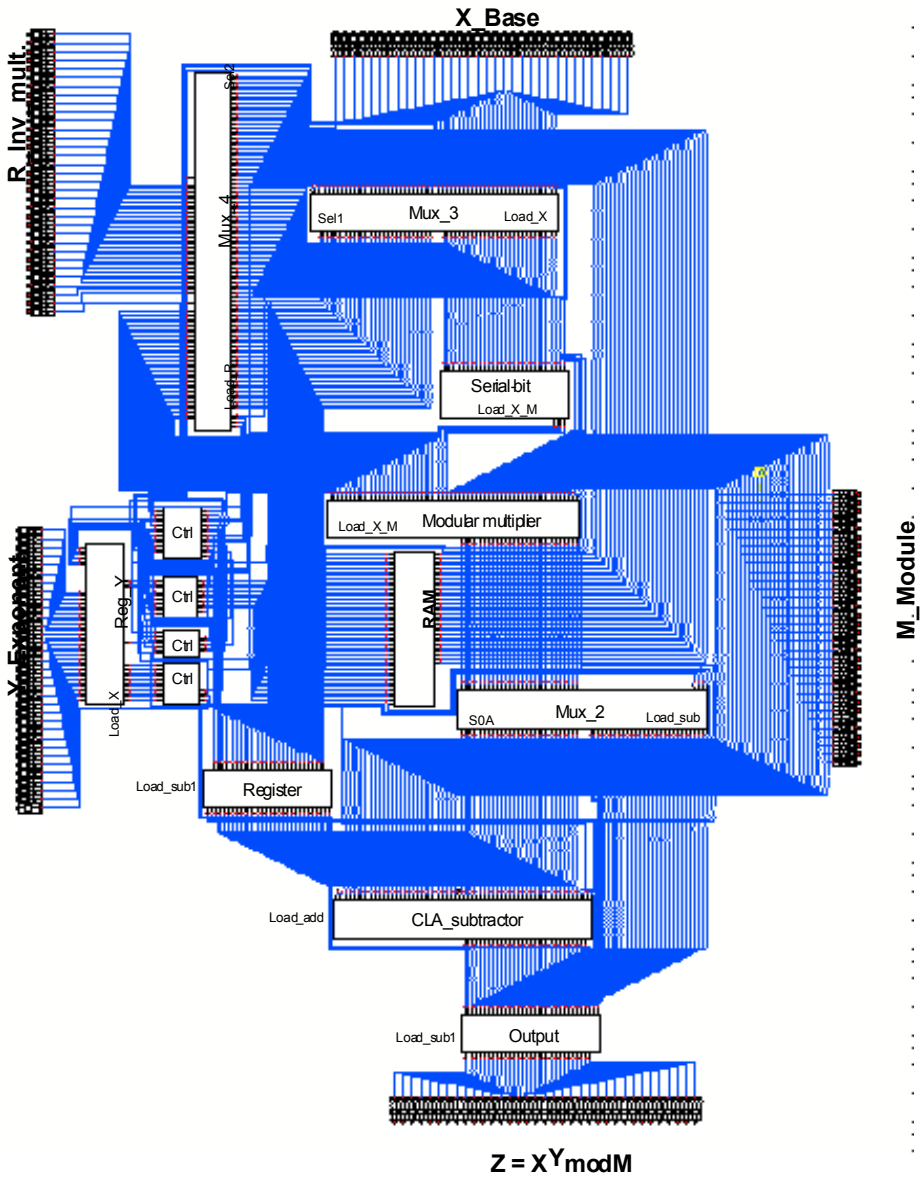
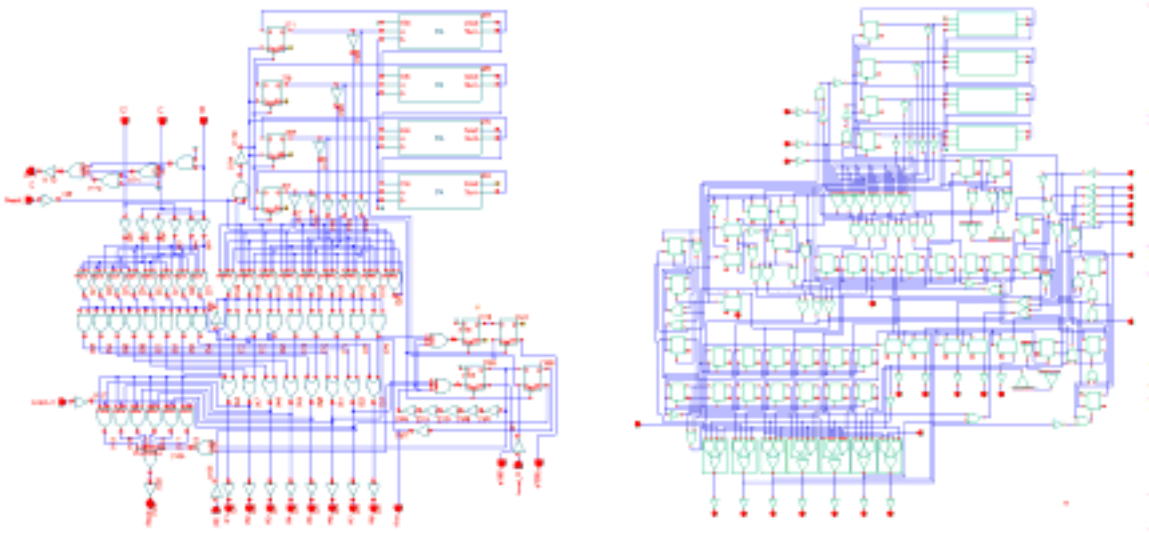



Figure 4.11. Schematic illustrating main control signals.

During the first step of the algorithm, the second component *Pro_count_Y* is a structure that allows the contents of all $V^j$ partial products be calculated and stored into the RAM. This state

machine includes a NOR decoder to generate sequentially the select lines required for reading operations. Once a given row is selected, a *Read* signal is activated in order to store data from *Register* executing a read operation.
Additionally, as shown in chip architecture, 3 bits coming from Reg_Y must be also decoded to generate 8 row select lines. If $F^{(i)} = 0$ or $F^{(i)} = 1$, write operation is not executed, signals *fi* and *f1* detect those conditions respectively. In figure 4.12, schematics of *Pro_count_Y* and *ctrl_sig* subsystems are depicted.



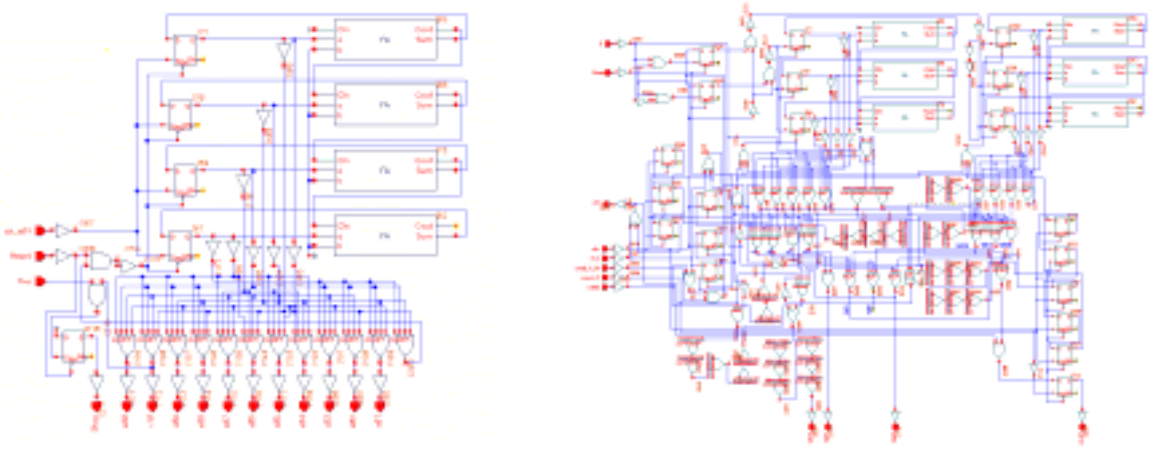
(a) (b)

Figure 4.12. Schematics of (a) *Ctrl_sig* and (b) *Pro_count_Y*

The exponent Y register has 32-bits input partitioned into 11 sections of 3 bits, so, we have that k*d* = 33. The 33-th bit is zero. Each one of the sections of the exponent is scanned by the state machine *Prog_count10,* its output signals are sequentially generated in order to activate in Reg_Y the group of bits $[F^{(i)}]$ to be scanned. The content of this group $Y_2, Y_1, Y_0$ is carried out to *Prog_count_Y*.

Scanning section tasks are synchronised by a signal *ck_S01* which is generated by *Prog1_count_6*. This block includes two state machines, its function is to count the number of multiplications to be executed for each i-section, so if $F^{(i)} \neq 0$, then four multiplications will be required, on the contrary only three modular multiplications will be executed. When this flag is active (*ck_S01)*, an evaluation of a new section of the exponent Y is started. Signal *Write*, is also generated at the beginning of $4^{th}$ multiplication only if $F^{(i)} \neq 0$ and $F^{(i)} \neq 1$, it means if internal signals fi = 1 and f1 =1, respectively.

All that procedure is executed during the second step of the algorithm, signal *five* is used for all state machines to identify which step of the algorithm is being executed. In figure 4.13, schematic corresponding to *Prog1_count_6* and *Prog_count_10* state machines are presented.



(a) (b)

Figure 4.13. Schematics of (a) *Prog_count_10* and (b) *Prog_count_6*

Several software tools provide the opportunity of creating dense array of small memory cells in order to optimise the RAM design and improving the data storage capability. Usually, RAM generators are available for large size memories. Due to small storage requirements, the intermediary RAM was implemented using both registers and latches.

This array which is presented in figure 4.14, is organised so that 32 bits along the selected row enabled by the row address are accessed simultaneously. The effective address calculation is executed by *Pro_count_Y* coding a 3-bit field statement coming from the exponent register. This field is used to specify indexed address mode through sel[0-7] signals.

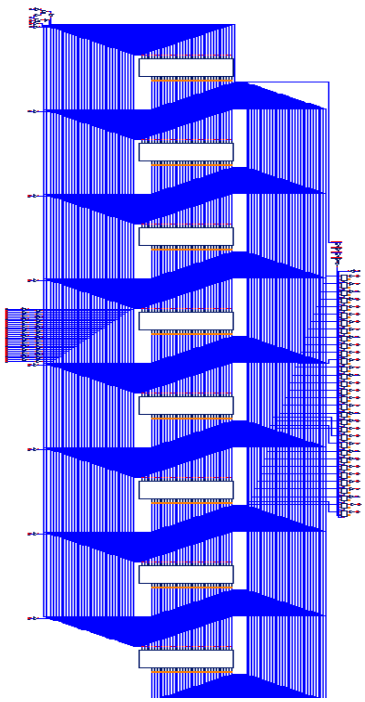


Figure 4.14. Registers array.

### 4.4.3. Comparator subtractor.

Due to the fact that a bit-serial multiplier architecture adds the respective input summands to compute a set of partial product bits, where each partial product must satisfy the condition $0 \leq S_i < M+W < 2M$, so, we can conclude that modular multiplication is executed by repeated cycles involving shifting and addition and usually together with simultaneous modular subtraction in order to satisfy the condition $S_n < M$. For this reason, a subtraction function is required in order to establish the comparison and subtraction if needed.

The comparator/subtractor is the other important part of the data path to be discussed. As its names suggests, the comparator/subtractor must provide comparison and subtraction operations on data furnished from modular multiplier. A block diagram of a 32-bits wide subtractor showing the inputs coming from Module Register and Modular Multiplier outputs passing through a multiplexer is given in figure 4.15. Two parallel 32-bit buses feed the functional block. This allows both inputs to receive data simultaneously, one of the two parallel buses is used to carry out the result to the register *word_32* once the $S_n < M$ condition is verified.

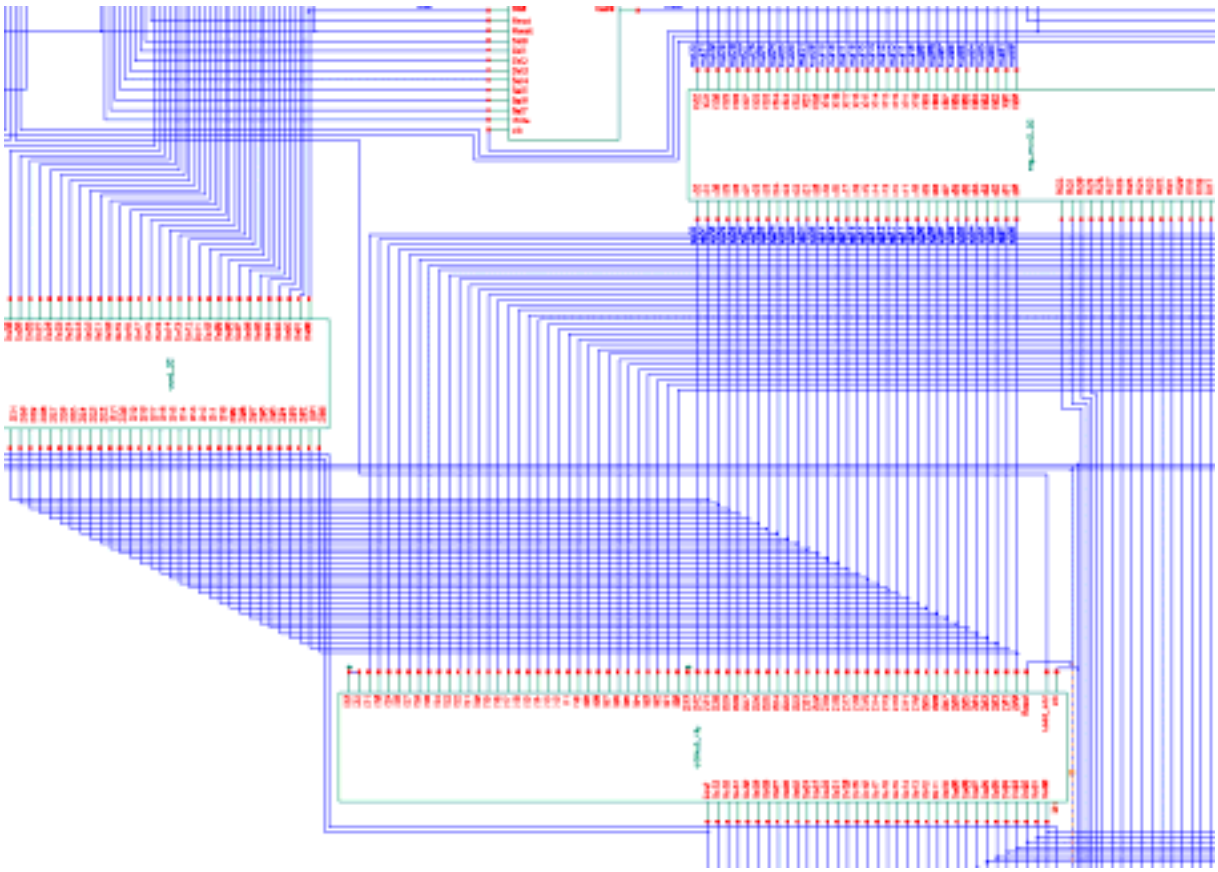


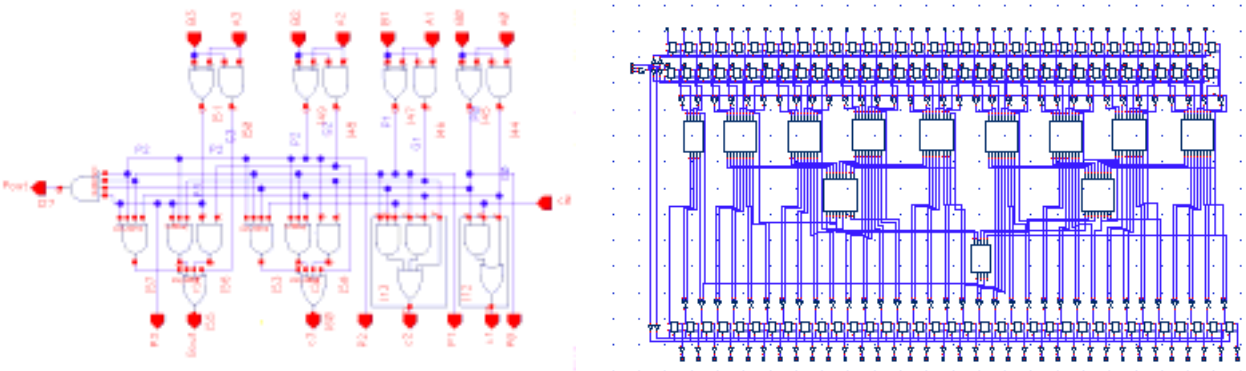
Figure 4.15. Multiplier/ Subtractor loop.

The subtractor execution time may limit the maximum clock frequency of the system unless special care is taken for arithmetic operations. These operations are slowed by carry or borrow propagation delays across the width of the subtractor. So, in this implementation we use a Carry Look Ahead approach in order to speed carry propagation across groups of adjacent stages. As only subtraction operation is required, a Carry Look Ahead (CLA) adder structure will be used [29]. The Carry Look Ahead approach allows calculate early the value of the carry rather than to propagate it. This implementation provides an important speed enhancement. The Carry Look Ahead Unit (CLU) used in this implementation is depicted in figure 4.16(a).

The subtraction is executed encoding the two's complement operand in order to reduce the number of loops to be done. This makes the operator faster and uses less hardware. To realise the two's complement of a binary number, first the number is inverted and then a logic "1" is added to it. The adder circuit includes the "1" to be added if necessary The analysis of the 32-bits Carry Look Ahead adder, shown in figure 4.16(b), can be extended to n-bits and applied to an implementation of the CPA array of the multiplier.

In this case, each bit of M must be inverted and an input carry = 1 must be set. Using XOR logic operators with each $m_i$ and the input carry =1, a representation two's complement of module is obtained. So, the operation to be executed will be Sn + (-M +1). Figure 4.17, shows a detailed diagram of this architecture. The adder must execute a fast subtraction of $S_n$ - M, only if $S_n$ > M condition is detected, This is announced by an overflow situation which is flagged by *Cout* signal. Overflow condition is implemented by using the XOR logic operation with the two last Carry signals.

*Cout* signal is sensed by the control part. If *Cout* = 1 condition is detected, the state machine generates the control signal required to execute subtraction operation loops before continuing with the general procedure. In figure 4.18, this situation is illustrated.



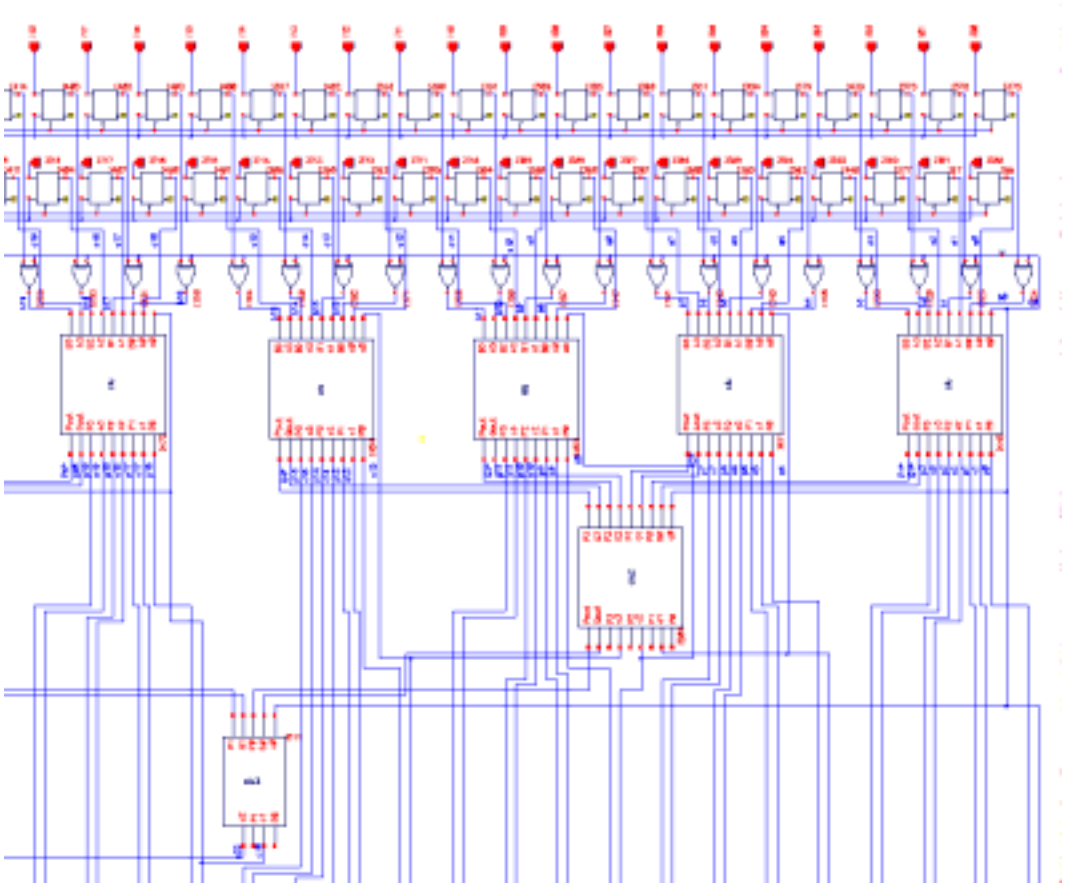


Figure 4.16. (a) CLU scheme. (b) 32-bits Carry Look Ahead subtractor.

Figure 4.17.Schematic of 32-bit CLA Subtractor.

When Cout = 1, both *Load_sub* and *Load_add* load signals are repeated while the generation of all not required control signals is suspended. As can be seen from figure 4.11, these signals allow load the partial product to be fed back to subtractor. During this procedure, a select signal S0A is activated in order to transfer the partial product to be subtracted.

In this case, a product partial 3168643542 > 3100785095 is obtained. *Cout* signal is "one", a subtraction using new data (3168643542 - 3100785095 = 67858447) as new partial product is executed. Once *Cout* = 0 condition is achieved, the state machine continues to generate the previous control sequencing.

This Carry Look Ahead architecture can execute 32-bits subtraction in 5 ns, so, a subtraction operation does no take a significant time speed to overall time. Only at most one subtraction

is needed each time, because Montgomery multiplication produces a result less than twice the modulus.

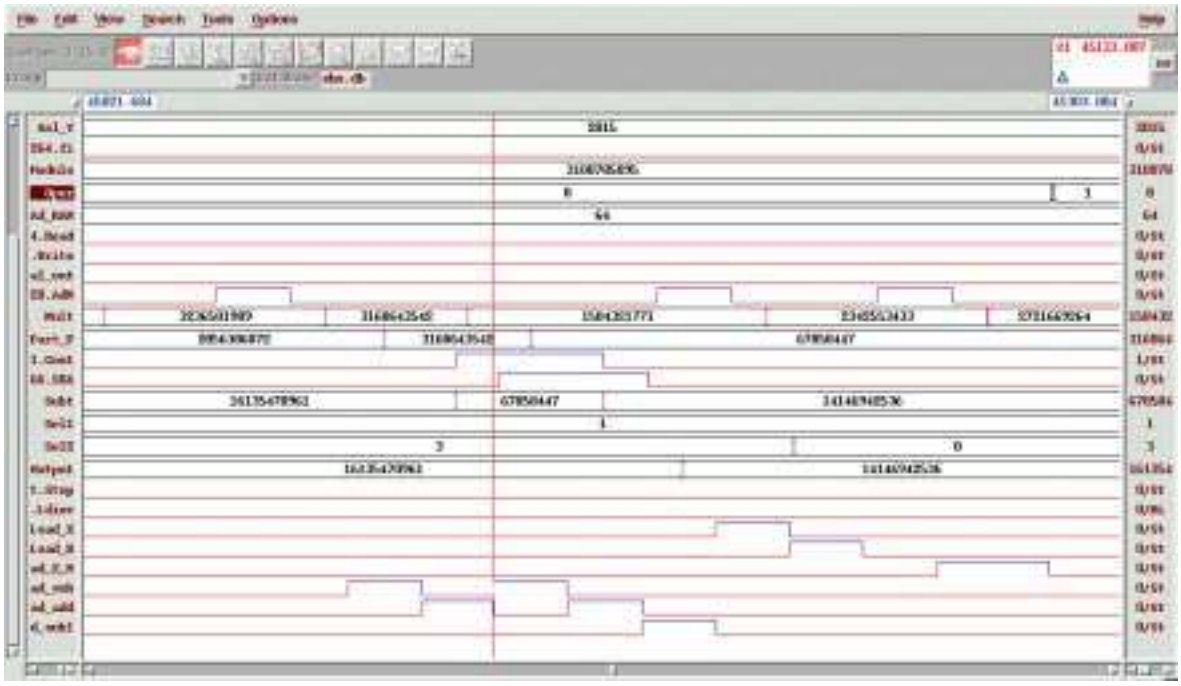


Figure 4.18. Overflow detection and subtraction operation.

### 4.5. Simulation results

Several simulations of the exponentiation system using typical parameters were done. Post-simulations were executed using Verilog simulator. This architecture is fast because, first, the number of multiplications is reduced, and second because steps for addition, shifting and eventually subtraction present a good performance, leading to have a compact operator with n-bit-serial word. The system requires small amount of hardware, it executes the exponentiation by using only selectors and carry save adders which have no carry propagation. So, this architecture allow speed-up the multiplication function. In order to analyse the simulation results, an example will be studied. Let calculate $X^Y$modM where the operands are presented in table IX:

Operation: $((297606960^{417292808})\bmod 3100785095) = 1336894240$

$$R = 2^n \text{mod} M \qquad R` = (2^n)^2 \text{mod} M$$

Table IX. Operands of the exponentiation

| op | Decimal | Binary ($2^i$) - i | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | 297606960 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| Y | 417292808 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| M | 3100785095 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| R | 1194182201 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| R` | 1915139751 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| d | 3 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Z | 1334514530 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

Applying generalised square multiply algorithm, it is possible to execute the exponentiation performing 39 modular multiplications. During first step six multiplications are executed while 33 are executed in the second one. In table X, the number of required multiplications for executing modular exponentiation is shown.

Table X. All multiplication are Montgomery's number.

| | C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| J | i = 8 | i = 7 | i = 6 | i = 5 | i = 4 | i = 3 | i = 2 | i = 1 | i = 0 |
| 0 | $X^{6}$ | $X^{48}$ | $X^{396}$ | $X^{3182}$ | $X^{24468}$ | $X^{203756}$ | $X^{1630050}$ | $X^{13040400}$ | $X^{104323202}$ |
| 1 | $X^{12}$ | $X^{96}$ | $X^{7920}$ | $X^{63640}$ | $X^{50936}$ | $X^{407512}$ | $X^{3260100}$ | $X^{26080800}$ | $X^{208646404}$ |
| 2 | $X^{24}$ | $X^{192}$ | $X^{1584}$ | $X^{12728}$ | $X^{101872}$ | $X^{815024}$ | $X^{6520200}$ | $X^{52161600}$ | $X^{417292808}$ |
| $F^{(i)}$ | 0 | $6 \neq 0$ | $7 \neq 0$ | $6 \neq 0$ | $6 \neq 0$ | $1 \neq 0$ | 0 | $1 \neq 0$ | 0 |
| C | - | $X^{192}X^{6}$ | $X^{1584}X^{7}$ | $X^{12728}X^{6}$ | $X^{101872}X^{6}$ | $X^{815024}X$ | - | $X^{52161600}X$ | - |
| Cf | $X^{24}$ | $X^{198}$ | $X^{1591}$ | $X^{12734}$ | $X^{101878}$ | $X^{815025}$ | $X^{6520200}$ | $X^{52161601}$ | $X^{417292808}$ |

In tables XI and XII, the values of $V_j$ and the content of the different sections of the exponent Y are shown. As can be seen, only the first calculated values of $V_i$ must be stored. For 128-bits length numbers, only seven values must be stored. This method does not require much store overhead. In general $(2^d-2)$ elements must be stored. It is possible to store only the odd powers through use of a sliding-windows method, but the system will become more complex.

Table XI. From j = 2 to $2^d$-1.

| | | |
|---|---|---|
| 7 | $V_7 = X^6.X \bmod M$ | $X^7 \bmod M$ |

| j | $V_j$ | $CV_j$ |
|---|---|---|
| | | |
| 0 | $V_0 = 0$ | 1 |
| 1 | $V_1 = 1$ | $X \bmod M$ |
| 2 | $V_2 = X.X \bmod M$ | $X^2 \bmod M$ |
| 3 | $V_3 = X^2.X \bmod M$ | $X^3 \bmod M$ |
| 4 | $V_4 = X^3.X \bmod M$ | $X^4 \bmod M$ |
| 5 | $V_5 = X^4.X \bmod M$ | $X^5 \bmod M$ |
| 6 | $V_6 = X^5.X \bmod M$ | $X^6 \bmod M$ |

Table XII. Exponent register fields.

| $F^{(i)}$ | Bin | Dec |
|---|---|---|
| $F^{(9)}$ | 011 | 3 |
| $F^{(8)}$ | 000 | 0 |
| $F^{(7)}$ | 110 | 6 |
| $F^{(6)}$ | 111 | 7 |
| $F^{(5)}$ | 110 | 6 |
| $F^{(4)}$ | 110 | 6 |
| $F^{(3)}$ | 001 | 1 |
| $F^{(2)}$ | 000 | 0 |
| $F^{(1)}$ | 001 | 1 |
| $F^{(0)}$ | 000 | 0 |

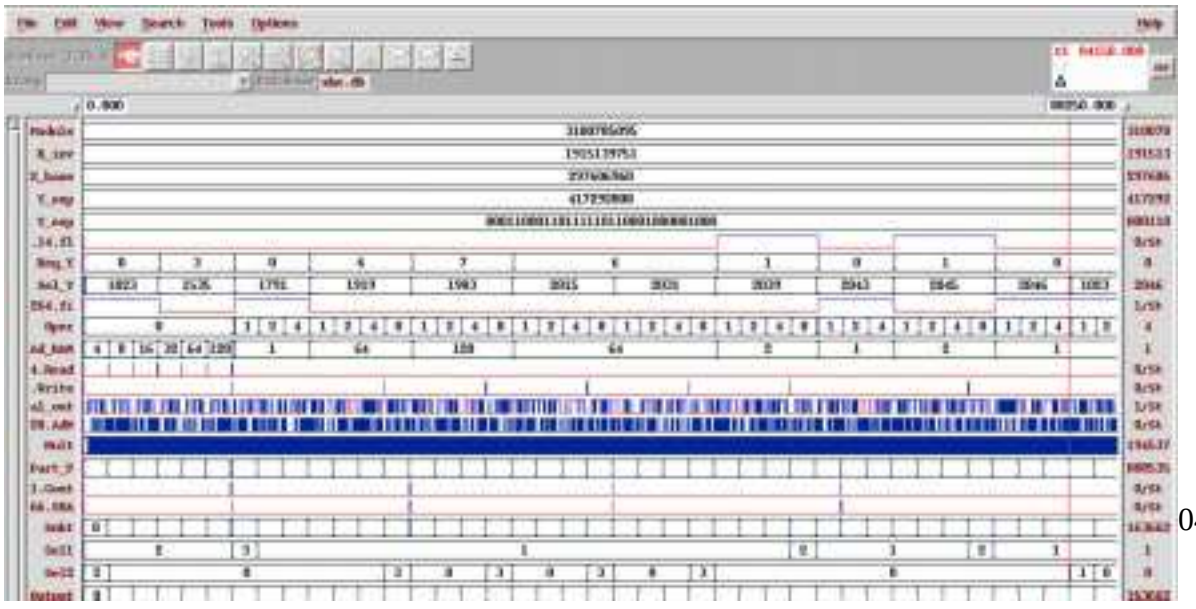
In table XIII, a detailed description of the number of multiplications to be executed are presented. Multiplications are classified according to steps of the algorithm. As shown, *Read* operations are required only during first step of the algorithm while *Write* operations are executed in the second one.

Table XIII. Decimal and modular products

| **Product** | **Decimal Representation** | **Product** | |
|---|---|---|---|
| **2 to $2^d$-1** | | **Decimal** | **Modular** |
| $X^2$modM | (297606960 x 297606960) | 88569902640441600 | 1220819910 |
| $X^3$modM | (1220819910 x 297606960) | 363324502122573600 | 2446309785 |
| $X^4$modM | (2446309785 x 297606960) | 728038818332103600 | 588220545 |
| $X^5$modM | (588220545 x 297606960) | 175058528206993200 | 230579675 |
| $X^6$modM | (230579675 x 297606960) | 68622116114538000 | 2421749705 |
| $X^7$modM | (2421749705 x 297606960) | 720729567585946800 | 1088410215 |
| **I = 8** | | | |
| $X^6$modM | (2446309785 x 2446309785) | 5984431564186746225 | 2421749705 |
| $X^{12}$modM | (2421749705 x 2421749705) | 5864871633667587025 | 959486135 |
| $X^{24}$modM | (959486135 x 959486135) | 920613643257238225 | 2845492880 |
| $F^{(8)} = 0$ | | | |
| $X^{24}$modM | ---------- | 920613643257238225 | 2845492880 |
| **I = 7** | | | |
| $X^{48}$modM | (2845492880 x 2845492880) | 8096829730130694400 | 3054400745 |
| $X^{96}$modM | (3054400745 x 3054400745) | 9329363911056555025 | 279690895 |
| $X^{192}$modM | (279690895 x 279690895) | 78226996745901025 | 2771104150 |
| $F^{(7)} = 6$ | | | |
| $X^{198}$modM | (2771104150 x 2421749705) | 6710920657786775750 | 1055815655 |
| **I = 6** | | | |
| $X^{396}$modM | (1055815655 x 1055815655) | 1114746697343079025 | 2931251230 |
| $X^{792}$modM | (2931251230 x 2931251230) | 8592233773376512900 | 2396536315 |
| $X^{1584}$modM | (2396536315 x 2396536315) | 5743386309113779225 | 757893075 |
| $F^{(6)} = 7$ | | | |
| $X^{1591}$modM | (757893075 x 1088410215) | 824898564707761125 | 2019467110 |
| **I = 5** | | | |
| $X^{3182}$modM | (2019467110 x 2019467110) | 4078247408371752100 | 1512481145 |
| $X^{6364}$modM | (1512481145 x 1512481145) | 2287599213980511025 | 2408263975 |
| $X^{12728}$modM | (2408263975 x 2408263975) | 5799735373282800625 | 1276250550 |

| | | | |
|---|---|---|---|
| $F^{(5)} = 6$ | | | |
| $X^{12734} \text{mod} M$ | (1276250550 x 2421749705) | 3090759392968587750 | 1299764255 |
| **I = 4** | | | |
| $X^{24468} \text{mod} M$ | (1299764255 x 1299764255) | 1689387118575705025 | 116562455 |
| $X^{50936} \text{mod} M$ | (116562455 x 116562455) | 13586805915627025 | 2841312675 |
| $X^{101872} \text{mod} M$ | (2841312675 x 2841312675) | 8073057717115655625 | 830139625 |
| $F^{(4)} = 6$ | | | |
| $X^{101878} \text{mod} M$ | (830139625 x 2421749705) | 2010390391952560625 | 427086680 |
| **I = 3** | | | |
| $X^{203756} \text{mod} M$ | (427086680 x 427086680) | 182403032233422400 | 184917350 |
| $X^{407512} \text{mod} M$ | (184917350 x 184917350) | 34194426331022500 | 864799135 |
| $X^{815024} \text{mod} M$ | (864799135 x 864799135) | 747877543896748225 | 2340178210 |
| $F^{(3)} = 1$ | | | |
| $X^{815025} \text{mod} M$ | (2340178210 x 297606960) | 696453322936341600 | 1397806095 |
| **I = 2** | | | |
| $X^{1630050} \text{mod} M$ | (1397806095 x 1397806095) | 1953861879219149025 | 3080141940 |
| $X^{3260100} \text{mod} M$ | (3080141940 x 3080141940) | 9487274370546963600 | 2053533270 |
| $X^{6520200} \text{mod} M$ | (2053533270 x 2053533270) | 4216998890996892900 | 2326786995 |
| $F^{(2)} = 0$ | | | |
| $X^{6520200} \text{mod} M$ | ---------- | 4216998890996892900 | 2326786995 |
| **I = 1** | | | |
| $X^{13040400} \text{mod} M$ | (2326786995 x 2326786995) | 5413937720101130025 | 1701027820 |
| $X^{26080800} \text{mod} M$ | (1701027820 x 1701027820) | 2893495644413952400 | 2013237725 |
| $X^{52161600} \text{mod} M$ | (2013237725 x 2013237725) | 4053126137363175625 | 1416995150 |
| $F^{(1)} = 1$ | | | |
| $X^{52161601} \text{mod} M$ | (1416995150 x 297606960) | 421707618926244000 | 2592698160 |
| **I = 0** | | | |
| $X^{104323202} \text{mod} M$ | (2592698160 x 2592698160) | 6722083748867385600 | 2227692430 |
| $X^{208646404} \text{mod} M$ | (2227692430 x 2227692430) | 4962613562679304900 | 1693762415 |
| $X^{417292808} \text{mod} M$ | (1693762415 x 1693762415) | 2868831118466632225 | 1334514530 |
| $F^{(0)} = 0$ | | | |
| $X^{417292808} \text{mod} M$ | ---------- | 2868831118466632225 | 1334514530 |

Simulation results of the exponentiation function is depicted in figure 4.19. This figure shows



04

a window of the computation of $X^Y$modM where X,Y and M are given in table IX .

Figure 4.19. Modular exponentiation simulation results.

"*Serial_out"* signal corresponds to a serial-bit input of the modular multiplier coming from serial-bit conversion, while "*Mult*" and "*Output*" are the partial multiplication and partial exponentiation results respectively.

Signal "*Cout"* indicates a Sn > M condition starting extra subtractions of the original modulus. *"Oper"* set signals is generated by the state machine for counting the number of multiplications to be executed in each cycle and addressing a cache memory when F(i)=0. From behavioural simulation results using the 0.6µm standard cell library, the execution cycle delay time is at most 54 ns. So, a minimum clock period of 18ns was observed giving a maximum clock frequency of 55 MHz using typical parameters.

Several simulations considering a reasonable number of inputs were done. In figure 4.20, simulation results for a new calculation are presented.

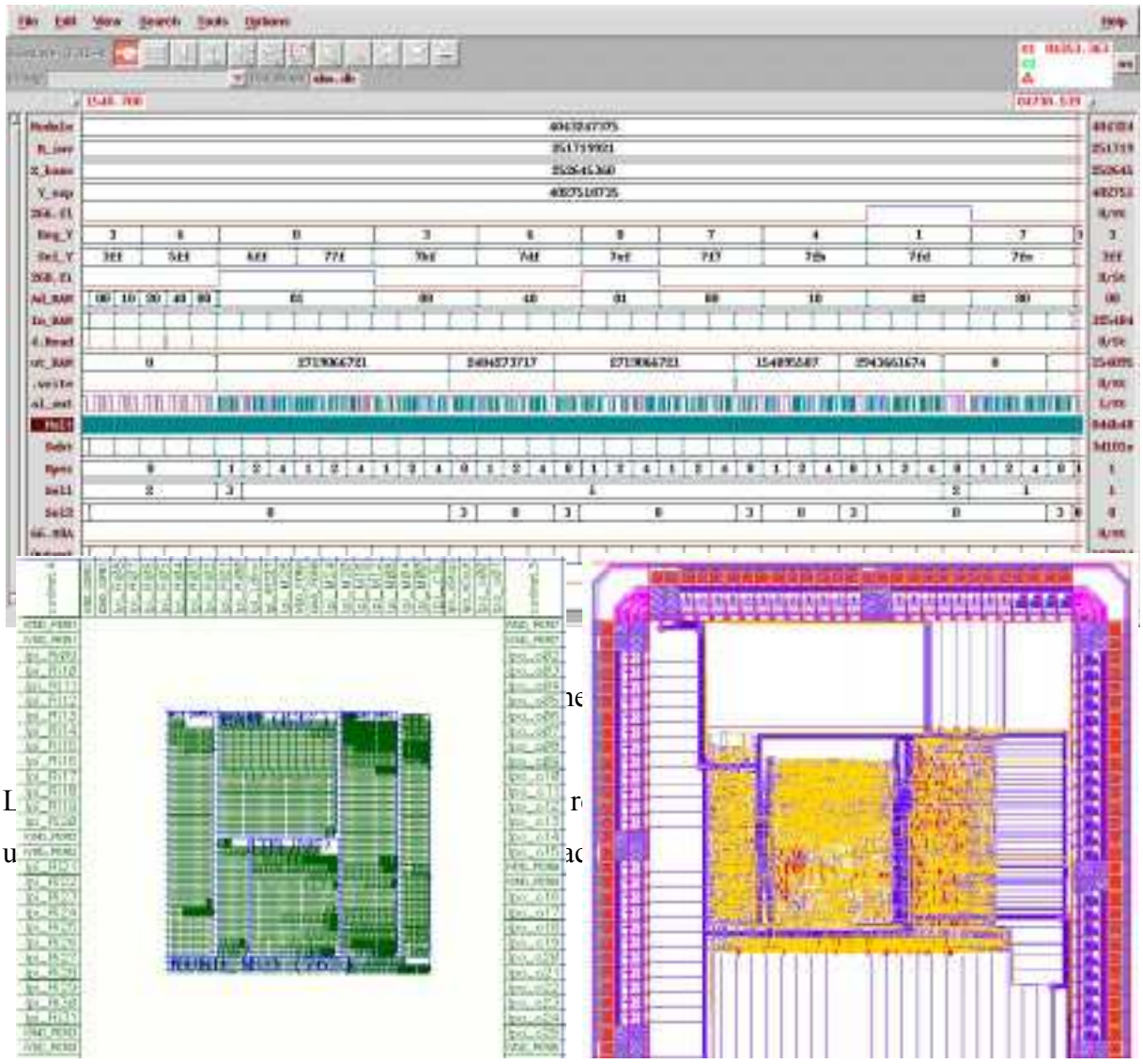


Figure 4.21. (a) Automatic placement (b) Final layout.

### 4.6. Experimental results

As mentioned in last chapter, from experimental results, the execution cycle delay time for a single multiplication was about 42 ns, each execution cycles needs at most three clock cycles, so it means that for a multiplier a maximum clock frequency of about 70 MHz was obtained. However, considering all interconnection of the data path in the modular exponentiation function, a reduction of clock frequency was verified obtaining a clock frequency of 55 MHz.

The architecture includes 8400 equivalent gates into an active area of 2.30 x 1.73 $mm^2$, giving a density of 2107 equivalent gates/$mm^2$. The layout prototype shown in figure 4.22 presents block, soft block and placement class region mixed layout strategies. Design size is limited by theI/O pad ring composed by 116 pads occupying an area of 4 x 3.9 $mm^2$.

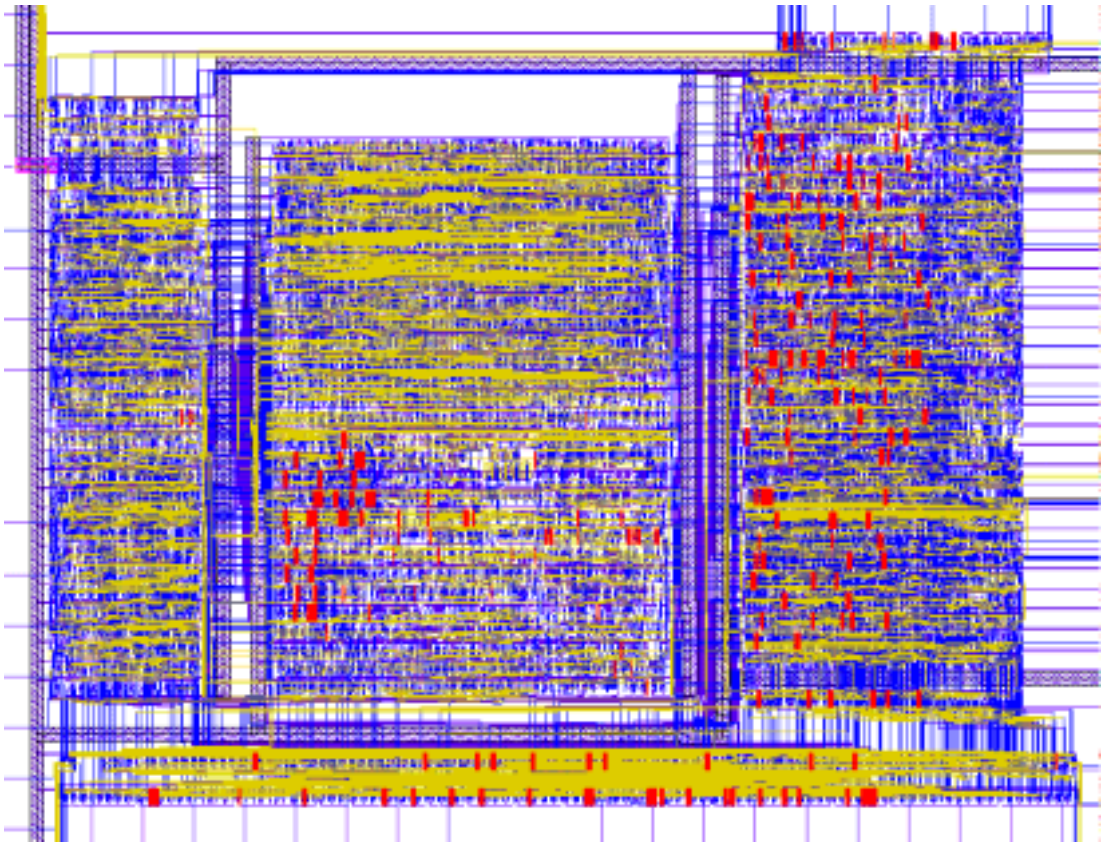


Figure 4.22. Core of the cryptosystem layout.

The average number of multiplications is given by $n+[(n/d)-1][1-(1/2^d)]+2^d-d-2$, where each modular multiplication takes ***n*** execution cycles. So, a 32-bits modular exponentiation is executed in average 44 multiplications each one using three clock periods. Each restoring operation takes a half clock period and each subtraction operation takes one clock period. Computing 32-bits modular exponentiation takes in average 2.3 μs with twelve cache memory accesses.

In general a n-bits modular exponentiation will take in average :

$$n[n+[(n/d)-1][1-(1/2^d)]+2^d-d-2] \text{ execution cycles.}$$

Main characteristics of the modular exponentiation architecture are given in table XIV. The generalised architecture can employ a n-operand adder module M realised using carry save adders and tree structures to substitute a CPA array.

Table XIV. Architecture performance [32].

| | |
|---|---|
| Modulus (M) | 32 bits $2^n > M > 2^{n-1}$ |
| Multiplicand (X) | 32 bits (Bin) |
| Multiplier(Y) | 32 bits (Bin) |
| Product | n + 1 bits ($X^Y$modM) |
| Exponentiation time | 1.7 μs |
| Active area size | 2.30 x 1.73 $mm^2$ |
| Equivalent gates | 8400 |
| Frequency | 55 MHz |
| 1024 bit [seg] (average) | $73.6 \times 10^{-6}$ |
| Chip size (bonding pads) | 15.6 $mm^2$. |
| Density of transistors | 8.4 $k/mm^2$. |
| Technology | 0.6μm-CMOS |
| Algorithm | M-ary/Montgomery |
| Cache memory | Eight words |

### 4.7. State of the art

In the interests of cryptographic application, estimates should be given for the speed of cryptographic operations such as RSA or Diffie-Hellman. The new designs are fast and smaller than previous ones. In table XV, is contained some of RSA chips and their main characteristics.

Table XV. State of the art of the cryptosystems [30].

| Chip | RAM | Tech. (μm) | Fabric. | Clock (MHz) | n (bits) | Algorithm |
|---|---|---|---|---|---|---|
| ST16CF54 | 352 | 1.2 | SGS | 5 | 768 | Montgomery |
| ST16KL74 | 608 | 1.2 | SGS | 5 | 1024 | Montgomery |
| SLE44C200 | 256 | 1.0 | Siemens | 5 | 540 | Sedlak |
| P83C855 | 512 | 1.2 | Philips | 10 | 648 | Quisquater |
| MC68HC5S | 512 | 1.2 | Motorola | 5 | 1328 | Quisquater |
| SCALPS | 128 | 1.5 | UCL | 10 | 512 | Montgomery |
| CY512i | 768 | 1.5 | Cylink | 15 | 512 | Massey/Omura |
| CRIPT | ext. | 1.2 | CNET | 25 | 1024 | bit wise |

In recent years, public-key cryptosystems has gained increasing attention from both companies and end users who wish to use this technology to add security to a wide variety of applications. One consequence of this trend has been the growing importance of public key smart cards to store a user's private key and to provide a secure computing environment for the private key operation.

Many chip manufacturers are therefore proposing ever better and faster implementations of public key algorithms using dedicated crypto-coprocessors on their chips. The most widely used smart cards with crypto-coprocessors are listed in table XVI. In this table we presents the standard field size of such chips in terms of on-board memory sizes (RAM), operating voltage and frequency, and the maximum public key size supported for RSA or DSA public modulus and elliptic curves.

Table XVI. Technical characteristics of some Smart Cards Cryptoprocessors [31].

| Name | Manuf. | Max | RAM | Voltage | Clock Ext. | Clock Int. | Tech |
|---|---|---|---|---|---|---|---|
| H8/3111 | Hitachi | 576 | 800B | 3v & 5v | 10Mhz | 10Mhz | 0.8μm |
| H8/3112 | Hitachi | 576 | 1312B | 3v & 5v | 10Mhz | 10Mhz | 0.8μm |
| H8/3113 | Hitachi | 1024 | 1.5KB | 3v & 5v | 10Mhz | 14Mhz | 0.5μm |
| T6N29 | Toshiba | 1024 | 512B | 3v & 5v | --- | --- | 0.6μm |
| T6N37 | Toshiba | 1024 | 512B | 3v & 5v | --- | --- | ---- |
| T6N39 | Toshiba | 1024 | 512B | 3v & 5v | --- | --- | ---- |
| T6N42 | Toshiba | 2048 | 512B | 3v & 5v | --- | --- | ---- |
| ST16CF54 | SGS-Th | 512 | 512B | 5v±10% | 5Mhz | 5Mhz | --- |
| ST19CF68 | SGS-Th | 512 | 960B | 3v,5v±10% | 10Mhz | 10Mhz | 0.6μm |
| ST19KF16 | SGS-Th | 1088 | 960B | 3v,5v±10% | 10Mhz | 10Mhz | 0.6μm |
| P83W854 | Philips | 2048 | 800B | 2.7v to 5.5v | 8Mhz | --- | ---- |
| P83W858 | Philips | 2048 | 800B | 2.7v to 5.5v | 8Mhz | --- | ---- |
| P83W8516 | Philips | 2048 | 2304B | 2.7v to 5.5v | 8Mhz | --- | ---- |
| P83W8532 | Philips | 2048 | 2304B | 2.7v to 5.5v | 8Mhz | --- | --- |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SMARTXA | Philips | 2048 | 1.5/2K | --- | --- | --- | --- |
| SLE44CR80S | Siemens | 540 | 256B | 3v to 5v | 7.5Mhz | 7.5Mhz | 0.7µm |
| SLE66CX160S | Siemens | 1100 | 1280B | 2.7v to 5.5v | 7.5Mhz | 7.5Mhz | 0.6µm |
| µPD789828 | NEC | 2048 | 1KB | 1.8v to 5.5v | 5Mhz | 40Mhz | 0.35µm |

Those chips are becoming bigger, more versatile faster and increasingly secure. The new range of public key sizes for RSA or DSA is now generally up to 1024 bits and some chips can even handle 2048-bit computation. Every architecture has its own optimizations for computing modular multiplication and exponentiation, naturally the best internal architecture of a co-processor relies strongly on the choice of modular multiplication algorithm. Table XVII, lists the computation times (measured in ms) on different chips for different public key algorithms.

Table XVII. Comparison of computation times [31].

| **Name** | **Application** | H8/3111-2 | H8/3113 | ST16CF54B | ST19CF68 | ST19KF16 | PP883388554 8 | PP883388551362 | SLE44CR80S | SLE66CX160 | µPD789828 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DES | 64 bits | --- | --- | 10 | --- | --- | 10 | 10 | 3.7 | 3.7 | 4 |
| SHA | 512bits | --- | --- | 15.2 | 8.2 | 8.2 | 10 | 5 | 5.6 | 5.6 | 2 |
| MDS | 512 bits | --- | --- | 12 | --- | --- | --- | --- | 9 | 9 | --- |
| RSA 512 | Sign with CRT | 202 | --- | 142 | 70 | 20 | 45 | 37 | 60 | 37 | 16 |
| RSA 512 | Sign without CRT | 514 | 68 | 389 | 195 | 55 | 140 | 93 | 220 | 110 | 52 |
| RSA 512 | Verify | --- | --- | 9 | 4.5 | 2 | 22 | 10 | 20 | 10.3 | 2 |
| RSA 768 | Sign with CRT | --- | --- | 377 | 189 | 50 | 182.5 | 88 | 250 | 124 | 52 |
| RSA 768 | Sign without CRT | --- | 210 | --- | --- | 165 | 385 | 220 | --- | 437 | 164 |
| RSA 768 | Verify | --- | --- | 190 | 100 | 3 | 36 | 18 | --- | 18.4 | 4 |
| RSA 1024 | Sign with CRT | --- | --- | 800 | 400 | 110 | 250 | 160 | 450 | 230 | 100 |
| RSA 1024 | Sign without CRT | --- | 480 | --- | --- | 380 | 800 | 400 | --- | 880 | 360 |
| RSA 1024 | Verify | --- | --- | 265 | 150 | 5 | 50 | 25 | --- | 24 | 7 |
| RSA 2048 | Sign with CRT | --- | --- | --- | --- | 780 | 2180 | 1100 | --- | 1475 | 750 |
| RSA 2048 | Sign without CRT | --- | --- | --- | --- | --- | 21 s | 6.4 s | --- | 44 | --- |
| RSA 2048 | Verify | --- | --- | --- | --- | 100 | 156 | 54 | --- | 268 | 45 |
| DSA 512 | Sign | --- | --- | 163 | 84 | 25 | 75 | 58 | 95 | 50 | 31 |
| DSA 512 | Verify | --- | --- | 283 | 146 | 40 | 115 | 82 | 175 | 90 | 70 |
| DSA 768 | Sign | --- | --- | --- | --- | 50 | 145 | 100 | --- | --- | 57 |
| DSA 768 | Verify | --- | --- | --- | --- | 80 | 230 | 145 | --- | --- | 150 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DSA 1024 | Sign | --- | --- | --- | --- | 100 | 215 | 150 | --- | 143 | --- |
| DSA 1024 | Verify | --- | --- | --- | --- | 160 | 355 | 225 | --- | 271 | --- |
| ECDSA 135 | Sign | --- | --- | --- | --- | --- | --- | --- | 185 | 185 | --- |
| ECDSA 135 | Verify | --- | --- | --- | --- | --- | --- | --- | 360 | 360 | --- |
| ECDSA 255 | Sign | --- | --- | --- | --- | --- | --- | --- | --- | --- | 81 |
| ECDSA 255 | Verify | --- | --- | --- | --- | --- | --- | --- | --- | --- | 380 |
| Internal Clock Frequency (MHz) | | 3.57 | 14 | 5 | 10 | 10 | Ind. | Ind. | 5 | 5 | 40 |

### 4.8. Conclusions.

An alternative and very general modular architecture has been proposed for performing modular exponentiation based on a generalised square-multiply binary method and Montgomery's algorithm using serial data. This implementation requires a smaller number of modular multiplications than the well known binary method. The system consists of a serial multiplier array coupled with a small size cache memory and some multiplexers.

Based on this architecture, it is possible to build an easily expandable RSA engine with an average number of multiplications given by $n+[(n/d)-1][1-(1/2^d)]+2^d-d-2$, where each modular multiplication takes ***n*** execution cycles. The calculation time is estimated by considering the number of average multiplications required for computing the exponentiation and the number of clock cycles needed for each single multiplication execution cycle.

This implementation takes the one argument in serial bit, least significant bit first, and produces the partial product in bit parallel form. The architecture has been presented and verified and the efficiency of hardware implementation discussed. Due to its simple logic, the proposed architecture presents a good performance. In general a n-bits modular exponentiation will take in average: $n[n+[(n/d)-1][1-(1/2^d)]+2^d-d-2]$ execution cycles.

From experimental results we must conclude that at most an execution cycle of 54ns is required using a clock cycle of 18ns. It means that the modular exponentiation system is functional at a maximum frequency of 55 MHz. The whole process takes a time proportional to the number of digits in X. The standard cell gates used to implement the modular multiplier operate at a supply voltage of 5V. The architecture has the advantage that it is very simple, allowing a cellular construction and is easily expandable to larger bit-widths. So, it can be easily used in implementing cryptography systems to execute modular exponentiation of long

word length numbers. It can be installed in some cellular phones which operate at about 15 MHz or into some smart cards.

The generalised architecture can employ a n-operand adder module M realised using carry save adders where the longest path in the circuit from input to output involves only six gate delays and tree structures to substitute a CPA array. The architecture has the advantage that it is very simple, allowing a cellular construction and is easily expandable to larger bit-widths. So, it can be easily used in implementing cryptography systems to execute modular exponentiation of long word length numbers.

This architecture is fast because, first, the number of multiplications is reduced, and second because steps for addition, shifting and eventually subtraction present a good performance, leading to have a compact operator with n-bit-serial word. The system requires small amount of hardware, it executes the exponentiation by using only selectors and carry save adders which do not have carry propagation. So, this architecture allows speed-up the multiplication function. The architecture can be installed in some cellular phones which operate at about 15 MHz or into a smart card which has available only 5x5 $mm^2$ and operates at about 4MHz.

**4.9. References.**


[1] E.F. Brickell, K.S. McCurley, An interactive identification scheme based on discrete logarithms and factoring. J. Cryptology, Vol. 5, No. 1, pp. 29-39, 1992.

[2] C.P. Schnorr, Efficient signatures generation for smart cards. J. Cryptology, Vol. 4, No. 3, pp. 161-174, 1991.

[3] Sung-Ming Yen, Chi-Sung Laih, Improved digital signature suitable for batch verification, IEEE Transaction on computers,Vol. 44, No. 7, pp. 957-959, July, 1995.

[4] W. Diffie, M. Hellman, Privacy and Authentication: An Introduction to Cryptography, Proc. of the IEEE, v. 67, No. 3, Mar 1979, pp. 397-427.

[5] J.J. Quisquater, C. Couvreur, Fast dechipherment algorithm for RSA Public-key Cryptosystem, Electronics Letter, vol.18, pp. 905-907, 1982.

[6] A. Fiat, A. Shamir, How to improve yourself: Practical solutions to identification and signature problems. Advances in Cryptology - Proc. of Crypto'86, pp. 186-194, 1986.

[7] T. ElGamal, A Public Key Cryptosystem and a Signature Scheme Based on Discrte Logarithms, IEEE Transactions of Information Theory, v. IT-31, No. 4, Jul. 1985, pp. 469 - 472.

[8] W. Diffie, M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, IT-22, No. 6, Nov 1976, pp. 644-654.

[9] R. Merkle, Secure communication over an insecure channel, Communications of the ACM, pp, 1976.

[10] P. Barret, Implementing the Rivest, Shamir, and Adleman, Public Key Encryption Algorithm on a Standard Digital Processor, Advances in Cryptology, Crypto 86, pp. 311-323, 1986

[11] H. Sedlak, The RSA Cryptography Processor, Advances in Cryptology, Eurocrypt 87, June, 1987

[12] M. Abe, H. Morita, Higher Radix Nonrestoring Modular Multiplication Algorithm and Public-key LSI Architecture with Limited Hardware Resources, Proc. Anacrypt'94, 1994, pp. 363-375.

[13] L.C.K. Hui, K.Y.Lam., Fast square-and multiply exponentiationfor RSA, Electronic letters, 30 (17), pps 1396-1397.

[14] P.W. Baker, Fast computation of ABmodN, Electron. Letters, Vol. 23, No. 15, pp. 794-795, July, 1985.

[15] E.F. Brickell, D.M. Gordon, K.S. McCurley, D.B. Wilson, Fast exponentiation with Precomputation (Extended abstract), Advances in Cryptology, Eurocrypt'92, Workshop on the theory and Applications of Cryptographic techniques. Balatonfured, Hungary, 1992, pp. 201-207.

[16] P.A. Findlay, B.A. Johnson, Modular Exponentiation Using Recursive Sums of Residues, Advances in Cryptology - Crypto'89, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara, California, 1989, pp. 371-386.

[17] G.R. Blakley, A Computer Algorithm for Calcualting the Product ABModulo, IEEE Transactions on Computers, Vol. C-32,No. 5, May,1983 pp. 497-500.

[18] A. Selby, C. Mitchell, Algorithms for software implementations of RSA, IEEE Proc., Vol. 136E, pp. 166-170, 1989.

[19] Y. Han, C.J. Mitchell, D. Gollmann, A Fast modular Exponentiation for RSA on Systolic Arrays, Inter. Journal Computer Math., Vol. 63, 1997, pp. 215-226.

[20] V. Bokio, M. Peinado, R. Venkatesan, Speeding up Discrete Log and Factoring Based Schemes via Precomputations, 17th International Conference on the theory and Applications of Cryptographic techniques. Spoo, Findland, 1998, pp. 221-235.

[21] C. Koç, High -Radix and Bit Recoding Techniques for Modular Exponentiation, Inter. Journal Computer Math., Vol. 40, 1991, pp. 139-156.

[22] E.F. Brickell, D.M. Gordon, K.S. McCurley, D.B. Wilson, Fast exponentiation with Precomputation (Extended abstract), Advances in Cryptology, Eurocrypt'92, Workshop on the theory and Applications of Cryptographic techniques. Balatonfured, Hungary, 1992, pp. 201-207.

[23] P. Rooij, Efficient Exponentiation using Precomputation and Vector Addition Chains, Eurocryp'94, Workshop on the theory and Applications of Cryptography Techniques, Perugia, Italy, May, 1994, pp. 389-399.

[24] T. Hamano, N. Takagi, S. Yajima, F. Preparata, "O(n)-Depth Circuit Algorithm for Modular Exponentiation, 1995, pp. 188-192.

[25] H. Morita, A Fast Modular Multiplication Algorithm based on a Higher Radix, Crypto'89, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara California 1989, pp. 387-399.

[26] E. Brickell, A Survey of Hardware Implementations of RSA, Advances in Cryptology - Crypto'89, International Conference on the theory and Applications of Cryptography and Information security, Santa Barbara, California, 1989, pp. 368-369.

[27] I. Koren, Computer Arithmetic Algorithms, Prentice Hall, New Jersey, 1993.

[28] D. E. Knuth, Seminumerical algorithms, Second edition, The art of Computer Programming, Addison Wesley Publishing Co., v. 2 , Massachusetts, 1981.

[29] J.M. Muller, Arithmétique des ordinateurs, Masson, Paris, 1989.

[30] D. Naccache, D. M'Raihi, Arithemetic co-processors for Public-key cryptography: The state of the Art, IEEE Micro, pp.14-24, June, 1996.

[31] Handschuh H, Paillier P. Smart, Card Crypto-Coprocessor for Public-Key Cryptography, CryptoBytes, Vol. 4, Number 1, Summer 1998.

[32] A. Bernal, A. Guyot, Hardware Implementation of M-ary Modular Exponentiation Algorithm, To be published in XIV Conference on Design of Circuits and Integrated Systems, DCIS'99, Palma ed Mallorca, Spain, Nov., 1999.

# 5. Low Power GaAs Methodologies.

## 5.1 Introduction.

Super fast systems with sub-nanoseconds cycle time, and multi-gigabit per second telecommunication systems were the motivations behind the development of high speed VLSI circuits, where speed was the main constraint and power consumption was not a limiting factor. As mentioned before, MOS is by far the most often used technology for VLSI circuits. Currently, in order to obtain high speed and MOS ICs high density scaling methods are used. MOS technology becomes more competitive than other technologies (ECL, I2L) when scaling process are applied. Scaling MOS technology shows lowest speed-power product.

However, when device miniaturisation is continued, the second order effects on device characteristics become significant, making it non-viable at a certain geometry. Additionally, channel length reduction must be accompanied by a supply voltage reduction, causing a narrow noise margin and high sensitivity to variations in the supply voltage. The mentioned drawbacks in MOS sub-micron geometry for Ultra-High Speed VLSI circuits lead to seek other technologies to obtain faster devices which can be used in designing more sophisticated systems.

Some cryptography applications use satellite communication where, high speed and principally radiation tolerant integrated circuits are needed. Recently, advances in high speed VLSI circuits and with the development of portable telecommunication and multimedia systems, which demand high clock frequency.

Considering mentioned reasons, GaAs technology becomes an excellent candidate to implement ultrahigh speed cryptographic applications. GaAs logic families are considered as an attractive alternative, if both high speed and radiation tolerance are required. Today’s modern electronic

communication systems with a need for very high levels of performance consider GaAs components. However, although GaAs logic families have better power-delay products than others logic families, their power consumption is still large. This in particular prevents the realisation of VLSI circuits in GaAs to be used in portable communication.

Smart cards are plastic credit cards containing built in electronics. They are widely used in Europe, and are likely to become similarly pervasive in the US. Recently, some smart cards have been also designed with RF communication representing a convergence of the RF ID and smart card concepts. In the future, the convergence of RF ID and smart cards technology will probably continue and these cards may gradually take over most monetary transactions, greatly reducing the need for card currency.

On the one hand, GaAs is one of the technologies more widely used in RF communication applications. On the other hand, cryptographic techniques show that a VLSI circuit capable of performing long wordlength (>256 bits) modulo multiplication at very high speed must form part of any high speed cryptosystem. The encryption algorithms built into smart cards protect them from unauthorised use, yet allow many clever ways in which the owner may use them in place of cash. So, a compact cryptosystem design combined with a regular architecture which takes advantage of superior performance of GaAs technique attract much interest.

Due to long word length modulo multiplication has applications in other secret communication problems and other cryptographic methods, a functional design of GaAs modular exponentiation would be available for incorporation into a variety of other systems that will attract outside interest.

Nevertheless, the smart cards usually have small electrical contacts so that the cards reading machine can provide the necessary electrical power, excessive power dissipation create a technical barrier for high integration on a single chip.

These consideration reveal the urgency of reducing the power dissipation in GaAs integrated circuits and more specifically in functions required for implementing cryptosystems.

In other words, a low power GaAs cryptochip would demonstrates the well-known potential advantages of Gallium Arsenide VLSI technology and would be very attractive for several high speed cryptosystems applications. For that reason, currently the cryptography systems are also included into the market behaviour of GaAs digital integrated circuits.

The development both of efficient low-power GaAs logic cells and novel design strategies to achieve low power consumption have been lately started. Low Power Gallium Arsenide technology, has been proposed as a viable alternative which overcomes both the MOS limitations in Ultra-High Speed applications and significant standard GaAs power dissipation of VLSI ICs.

This chapter is focused on the reviewing of GaAs technology to be used in designing low power functional blocks of the cryptosystem. Besides, some power reduction strategies will be presented. These strategies will be further used for implementing GaAs low power by mixing mentioned approaches. Nevertheless, the GaAs foundry fix the technology parameters, reductions of the supply voltage level can be also done.

So, for implementing a low power GaAs cryptosystem, low power GaAs level structures for typical logic level functions like flip flops, muxes, full adders, etc, must be studied. In next chapters, low power GaAs level structures for intermediary memories are presented. Additionally, a simple, but very power efficient logic style, the branch based logic for designing full adders is discussed.

## 5.2 Gallium Arsenide Technology

Gallium Arsenide is a compound semiconductor that has been widely used since the late 1960 for microwave application and light emission. Its dominance in the microwave area is still retained. The use of GaAs MESFET for digital applications began in 1974 with some relatively high power, high speed SSI divider circuits [1]. The technology developments over the years have allowed to design integrated circuits which have been well characterised by higher speeds and power levels comparable with silicon MOSFET.
To explore the potential of the GaAs technology, a direct comparison between GaAs and silicon must be done. Considering the electrical properties of the two materials, -which are summarised in table I,- is possible to define briefly the advantages of GaAs over silicon as a base material for Ultra-High integrated circuits implementation. In table I, $m_0$ represents the free electron mass while $N_A$ and $N_D$ are the acceptors and donors concentration respectively.

Table I. GaAs / Silicon Electrical Properties at 300°K [2].

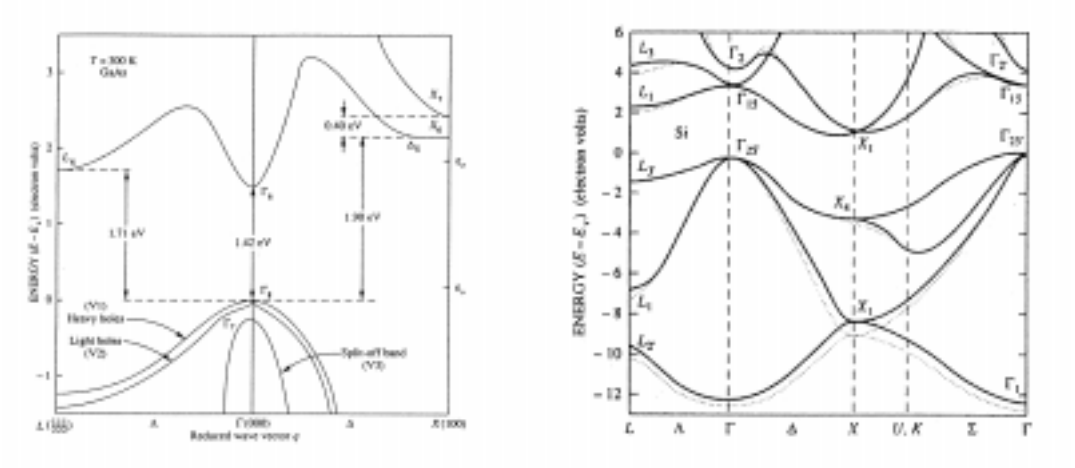
| Properties | Units | GaAs | Silicon |
|---|---|---|---|
| Effective mass electron | | 0.063 $m_o$ | 0.33 $m_o$ |
| Effective mass hole | | 0.090 $m_o$ | 0.16 $m_o$ |
| Electron mobility (at $N_D=10^{17}$ $cm^{-3}$) | $cm^{-2}$/v-s | 6000 | 1200 |
| Hole mobility (at $N_A=10^{17}$ $cm^{-3}$) | $cm^{-2}$/v-s | 350 | 480 |
| Maximum electron drift velocity | cm/s | $8 \times 10^6$ | $6.5 \times 10^6$ |

| | | | |
|---|---|---|---|
| Dielectric constant | | 13.1 $e_o$ | 11.9 $e_o$ |
| Energy gap | ev | 1.42 | 1.12 |
| Type of gap | | Direct | Indirect |
| Density of states in conduction band | $cm^{-3}$ | $5 \times 10^{17}$ | $3 \times 10^{19}$ |
| Density of states in valence band | $cm^{-3}$ | $7 \times 10^{18}$ | $1 \times 10^{19}$ |
| Maximum resistivity | Ω-cm | $10^{9}$ | $10^{5}$ |
| Minority carrier life time | sec | $10^{-8}$ | $2.5 \times 10^{-3}$ |
| Surface state density | $cm^{-2}$ | $10^{12}$ | $10^{10}$ |
| Breakdown field | kv/cm | $4 \times 10^{5}$ m | $3 \times 10^{5}$ |

### 5.2.1. Band diagrams.

One of the main advantage of GaAs over silicon for high speed FETs can be understood by considering the energy band diagrams features. On the one hand, GaAs is a direct gap semiconductor, that means, that the minimum energy separation between the conduction band minimum and valence band maximum occurs at the same momentum **k,** at the Brillouin zone centres, as can be seen from figure 1.1 (a). On the other hand silicon is an indirect-gap semiconductor due to its conduction band minimum is separated in momentum from the valence band maximum figure 5.1 (b).

Th energy band structure of GaAs is responsible of its virtues. Narrow and sharply valleys in the band structure correspond to electrons with low effective mass state, while wide and gentle curvature valleys correspond to electrons with larger effective mass. Those characteristic lead to some desirable consequences for the electron transport such as: higher mobility for low energy electrons and appropriate velocity-electric field characteristics.




(a) (b)

Figure 5.1. Energy band structures of GaAs (a) and silicon (b) [3].

### 5.2.2. Electron mobility.

Mobility depends upon concentration of impurity and temperature and varies inversely with electron effective mass [2]. As can be seen from table I, for GaAs the effective mass of these electrons is 0.063 times the mass of a free electron. For that reason the low energy electrons in GaAs show a higher mobility than the more energetic electrons.

From an electrical point of view the principal advantage that GaAs has over silicon is that mobility of electrons in GaAs is six to ten times higher than in silicon. Therefore, transit times as short as 15 - 10 picoseconds, corresponding to current gain-bandwidth products in the range 15 - 25 GHz can be obtained for GaAs transistors for typical gate lengths of 0.5 - 1.0 μm. That represents three to five times improvement over silicon devices. In figure 5.2, the mobility values for both electrons and holes for each material as a function of impurity atom concentration are shown. Notice further that the hole mobility in GaAs is significantly less than that in silicon.

### 5.2.3. Velocity-Field Relation.

The band structure of GaAs material leads to the velocity-electric field characteristic for electrons show in figure 5.3 and 5.4. The characteristic is obtained through simulation at different values of donor concentration. In these figures the same characteristics for silicon are given.

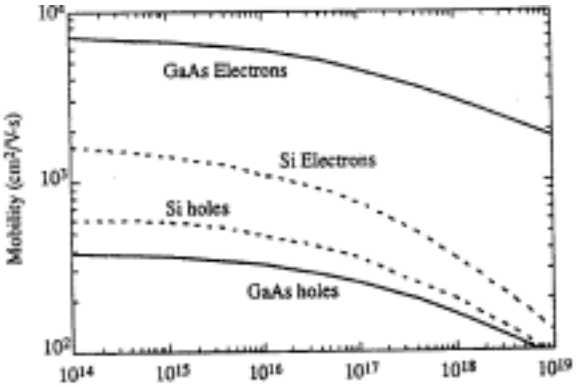
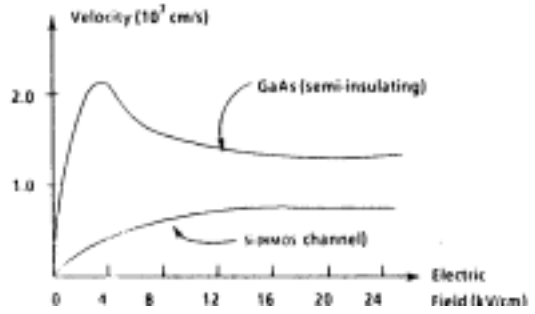
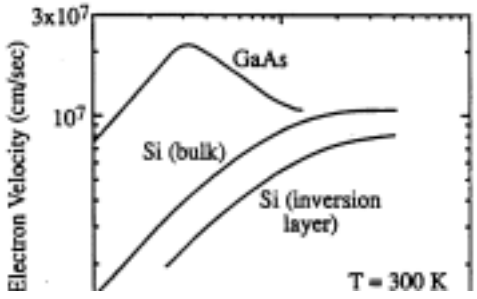



Figure 5.2. Carrier mobility in GaAs and silicon [2].

In GaAs the electron velocity is a non linear function of the electric field. The velocity in semi-insulating GaAs reaches its peak value of about 2.2 x $10^7$ cm/s at approximately 3 kv/cm and



decays to a saturation value of about 1.4 x $10^7$ cm/s. Thus, at a doping level of 1 x $10^{17}$ $cm^{-3}$, the electron drift mobility is quite high for a low electric field [4]. This provide low resistivity in thin film layers and high electron velocity at low applied voltage. Nevertheless at normal doping levels the saturation drift velocity for GaAs and silicon are almost equal, the saturation velocity in GaAs is achieved at electrical fields about four times lower than in silicon.

Fig. 5.3. Steady state electron in GaAs/Si [5]. Fig. 5.4. Steady state electron in AsGa/Si [6].

### 5.2.4. Schottky Junction (Barrier heights).

The height of the depletion region under the metal-semiconductor interface is known as Schottky junction. The height of the depletion region depends upon the gate voltage. Schottky barrier diode is one of the high performance components available in GaAs technology and is extensively used in the design of GaAs circuits.

It may find application as a level-shifting element, limited to forward bias operation emphasising low series resistance and high junction capacitance. In other case, can be used as logic-switching element where both forward and reverse-bias operation are needed, obtaining low series resistance and low capacitance. Also, is often used by itself as a diode for both level and logic shifting.

In opposition to silicon, the barrier height of the metal/GaAs junction for a n-type device is nearly independent of the work function of the metal that form the junction. For that reason, Schottky barriers can be realised on GaAs with large variety of metals (eg. aluminium, platinum, titanium) leading to high quality Schottky junctions with excellent ideally factors, see table II.

Table II. Barrier height in volts for various types of semiconductor [5].

| Semicond. | Type | Ag | Al | Au | Pt | Ti | W |
|---|---|---|---|---|---|---|---|
| Silicon | n | 0.78 | 0.72 | 0.80 | 0.90 | 0.50 | 0.67 |
| Silicon | p | 0.94 | 0.58 | 0.34 | | 0.61 | 0.45 |
| GaAs | n | 0.88 | 0.80 | 0.90 | 0.84 | | 0.80 |
| GaAs | p | 0.63 | | 0.42 | | | |

### 5.2.5. Depletion heights and capacitance.

GaAs field effect transistor does not have any p-n junction around its drain and source terminals and therefore the interelectrode capacitance in a GaAs device is much smaller.

### 5.2.6. Current Flow Across a Schottky junction.

Since the current density Js (1), (2) in both GaAs and MOS device is proportional to the electron velocity, the amount of current available to charge or discharge a particular load capacitance in a GaAs device is three to five larger and the switching speed is therefore three to five higher than in a silicon device with the same dimension, it means, with the same channel length, channel width and channel electron concentration. On the other hand, fairly low reverse currents are obtained ($J_S < 1\ \mu A/cm^2$ ).

$$J_S = qnv = qn\mu E \qquad (1)$$

$$J_S = [R_* T^2 \exp ( - q\emptyset_{Bn} / KT)][ \exp (qv / nKT) -1] \qquad (2)$$

### 5.2.7. Resistivity.

Gallium Arsenide is capable of being grown in a high resistivity form which is called as semi-insulating GaAs. The semi-insulating property of GaAs material (resistivity in the range of $10^7$ - $10^9$ Ω-cm at room temperature) is another advantage for high performance devices. It not only minimises the parasitic capacitance for interconnections on the GaAs surface but also allows for easy electrical isolation of multiple devices on a single substrate.

The low capacitance presented by an interconnect line on a GaAs substrate can be much less than that of an interconnect line on silicon, due to the fact that silicon is fairly conductive. This is other reason why digital GaAs gates can switch faster than their bulk silicon counterparts.

### 5.2.8. Radiation resistance.

The nuclear radiation encountered in certain military and space environments and in the nuclear industrial field is the most demanding ambient which semiconductor devices are exposed. For these type of applications (space and military), radiation hardness is a desirable property. In the study of radiation effects in semiconductor devices, several types of radiation, radiation damage and semiconductor devices must be considered. Electronic circuits are affected by several radiation types such as neutrons, protons, gamma rays, cosmic rays and electrons. These radiation types induce three fundamental effects : displacement of atoms from their lattice sites, ionisation of atoms and internal energy changes.

The two basic damage mechanism: displacement and ionisation which result from cumulative exposure to fast neutrons, generate defects in the crystal lattice by displacing lattice constituents, thus introducing additional energy states in the energy band gap. This effect is negligible for the GaAs MESFET. Different experimental test data and theoretical analyses have demonstrated the tolerance of GaAs discrete JFETs and MESFETs and planar integrated circuits to fast neutrons

and ionising radiation both under transient and cumulative conditions. Outstanding total dose ionising radiation behaviour is attributed to the p-n junction or Schottky barrier gate structure which is free of charge build-up [7]. In other words, GaAs is more radiation resistant than silicon due to the absence of gate oxide.

### 5.2.9. Reliability.

Traditionally, reliability has been expressed in terms of failure rates and is the form of expressing reliability today. But, in GaAs technology the suppliers do not measure failure rates directly but a distribution of failures which are measured by life testing and characterising distribution parameters. The lifetime and the spread of the distribution are two commonly parameters which are measured using highly accelerated test. The temperature is the primary method used to accelerate the test.

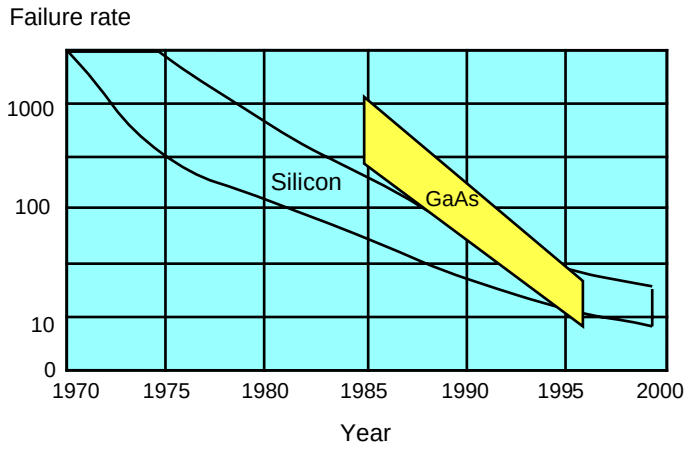



Figure 5.5. Improvements in GaAs MESFET Reliability compared to silicon [8].

As can be seen from the figure 5.5, GaAs reliability relative to silicon is growing because of the lifetimes have exhibited improvements over time at a rate apparently superior to changes in silicon technology. The evidence of the inherent superiority of MESFETs and gold-based interconnects may be accepted before the 21st century.

Marginal improvements are generally made as a result of other changes such as: process and control improvements, design rule changes, assembly and packaging control, electrical measurements guard-bands, overall product maturity, changes in handling and shipping procedures and changes in application conditions. The key to make a change toward the next generation of GaAs reliability focus has to move from measuring to controlling. The emphasis must move to understanding and especially controlling each of the parameters which determine premature failures [8].

Finally, the direct band gap of GaAs allows efficient radiative recombination of electrons and holes allowing to use the p-n junctions as light emitters and in consequence an efficient integration of electrical and optical functions can be achieved.
Additionally, because of its larger band gap, GaAs devices can operate over a wider temperature range (from -200° to 200°).

Considering that remarkable advantages over silicon, much effort is being dedicated to the development of GaAs ICs technologies. Therefore, GaAs technology maturity in the processing of digital integrated circuits in early 90's was equivalent to silicon technology maturity of the mid 1970's. Nevertheless, the expected higher performance of GaAs compared with silicon should be studied not only on the basis of the material properties but also in terms of the actual logic gates and integrated circuits implemented in GaAs technologies and their inherent power dissipation.

### 5.3. A brief review of GaAs Logic families.

There are several device choices for high-speed GaAs ICs, each with certain advantages and disadvantages. The most mature of these device technology is the depletion-mode FET (DFET). This device has a large current drive capacity per unit device width, contributing to its high speed, low fan-out sensitivity and higher power dissipation.

Enhancement-mode FET (EFET) is another device which is obtained by increasing the pinch-off voltage of the DFET. This device shows a low current and a low power dissipation. On the other hand, when the Schottky barrier of the EFET is replaced with an implanted p-region that forms a p-n junction for the gate, the result is a junction EFET, known as E-JFET. Today, GaAs MESFET are far more widely used than GaAs JFETs because lower parasitic result in superior high-frequency performance. Several logic GaAs families which are based on the mentioned devices have been proposed. The most popular approaches to high speed GaAs logic circuits will be briefly mentioned.

The Buffered-FET Logic (BFL) [1] was developed by Hewlett Packard in 1974. This approach used single and dual-gate FET as the switching transistors with an active load and level shifting diodes and a source follower in the output circuits, it represents the fastest gate for a reasonable fan-outs, but dissipates the most power. The function of the source follower with level shifting diodes consists in restoring the required logic levels voltages (+0.7V and to -Vth or below) required by the inputs FETs.

Unbuffered FET Logic (UFL) [9], was obtained from design variations of BFL family. Using a quite different circuit structure where the load driver source follower is omitted, the new circuit configuration consumes less power. In this case, the circuit shows higher sensitivity to high fanout because there is no buffer between the switching transistor and the output node.

In order to reduce the BFL power consumption several approachs have been proposed. One of them is the Capacitor Diode FET Logic (CDFL) [10] which add a Schottky diode in the voltage-shift section of the circuit causing that this section is always reversed-biased. The Schottky diode acts as a capacitor providing capacitive coupling between stages through which the high frequency signal is transmitted.

Another approach to minimise area and power dissipation is known as Schottky-Diode FET Logic (SDFL) [11][12][13][14], proposed in 1978, used a small Schottky barrier switching diodes as logic inputs and a single driven D-MESFET for output inversion and buffering This approach dissipates about one-fifth the power of the BFL, however, it is slower by about a factor of two. This approach offers saving s in circuit area, since the logic is implemented using diodes that occupy a smaller area than FETs. Both BFL and SDFL families were extensively employed for the design of depletion-mode GaAs integrated circuits.

Buffered Diode FET Logic (BDFL) [15], family resulted from some variations of SDFL such as buffered inputs. This configuration shows low input capacitance, buffered inputs and also no dc current loading on the output. However, the configuration circuit, if not buffered is sensitive to fanout.

Another GaAs FET logic approach developed in 1977 and called Direct Coupled FET Logic (DCFL) [16][17], is the most common circuit design approach based on E-MESFETs. In this approach enhancement-mode rather depletion-mode FETs are used. Using this configuration no level shifting is required. The main characteristic is that the power dissipation can be very low. Since enhancement -mode FETs are used, the logic swings are much smaller than for BFL and are also less than SDFL. For this reason, material and processing requirements are more severe due to require strict uniformity control of the device threshold in a logic structure [18]. From a static point of view, DCFL has very good fanout capability determined by the very low leakage currents. From a dynamic point of view, the switching speed of DCFL gates is reduced by the gate capacitance loading of the output node.

Additionally, the small width D-MESFET used in DCFL cause that the output rise time of the circuit with high fanout is slower.

Super Buffer FET Logic (SBFL) [19] use a quasi-complementary output driver in order to improve the peak load current-static current ratio of DCFL family. Due to the fact that fan-out and capacity load sensitivity for the DCFL is high, the use of the output driver (superbuffer) allow implement circuits with higher performance and better peak load current-static current ratio. In figure 5.6, a SBFL three input NOR is shown.

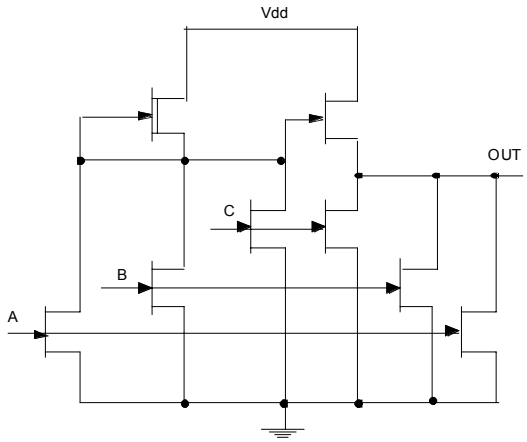



Figure 5.6. SBFL three input NOR

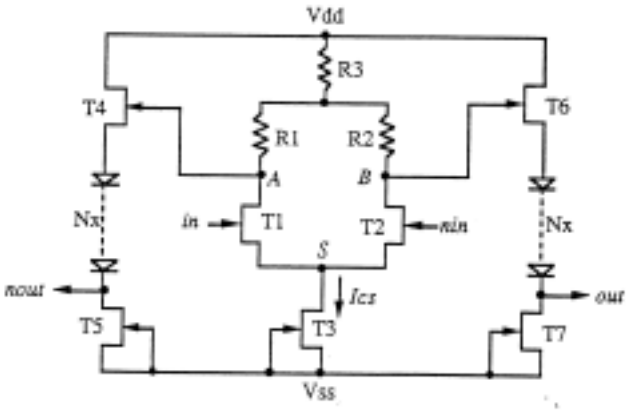



Figure 5.7. SCFL Logic structure.

In order to overcome the limitations of FET threshold control in DCFL, the Capacitor-Coupled FET Logic or Source Coupled FET logic SCFL [20][21] was proposed. A differential amplifier and two follower buffers with diode level shifters composed the inverter configuration. This logic family has been used to design circuits which have demonstrated a wide range of tolerance to threshold voltage and partial immunity to temperature variation. Figure 5.7 shows a SCFL logic structure.

Some SCFL family characteristics make it suitable for implementing high speed, low power circuits. First one, the gate-drain capacitance is small because the drain voltage at the ON state may be higher than any other logic family by design. Second, the discharge time of the differential amplifier outputs is short because the discharging current is dominated by the saturation current of the switching transistors. Another important characteristics that SCFL shows a good fan-out capability. However, Source Follower Direct Coupled FET Logic SCFL family show unacceptable levels of power consumption for complex portable telecommunication and multimedia systems applications.

All logic families and in fact all GaAs MESFET logic families thus far reported, dissipate static power. Therefore, their performance is tied to constant power delay curves. Another important characteristic of these logic families is that their logic swing is determined by the width and

length ratios of load and switching FETs, limiting both gate fan-in and circuit densities. Additionally, all those logic families are consumptive of power and consequently the scale integration of circuits made from these gates is limited by the power budget divided by the power dissipation per gate.

Power Rail Logic [22] was proposed as a new logic style which offers smaller area and lower dissipation than DCFL while its speed is quite similar. Topologically, the gate is identical to a DCFL gate with an input signal that is used to control the power rail of the gate, as can be seen in figure 5.8. The style allows to break down to ground the remaining gates that are not used in the information processing reducing the global power dissipation.

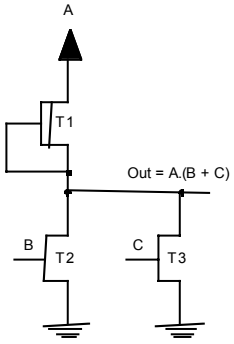



Figure 5.8. PRL schematic.

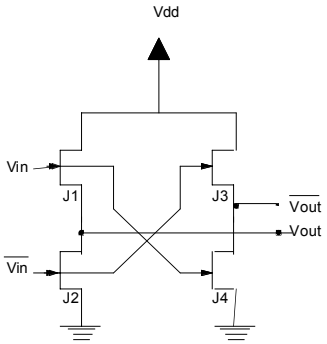



Figure 5.9. PCFL schematic.

A pseudo-Complementary FET logic (PCFL) low power family [23] uses a complementary signals to control the pull-up -and the pull-down networks of the gate. For this reason the gate delivers complementary output signals. Using this approach, the circuit itself must be duplicated. The low threshold voltage of enhanced-mode transistors would limit the degradation of the high logic level. This family allows design high speed GaAs digital systems with reasonable power consumption. A PCFL schematic is depicted in figure 5.9.

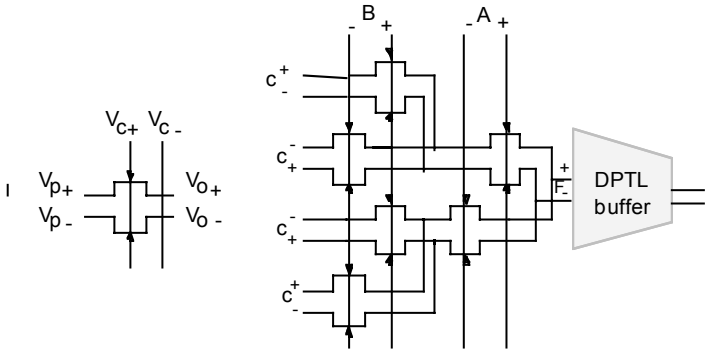



Figure 5.10. DPTL schematic.

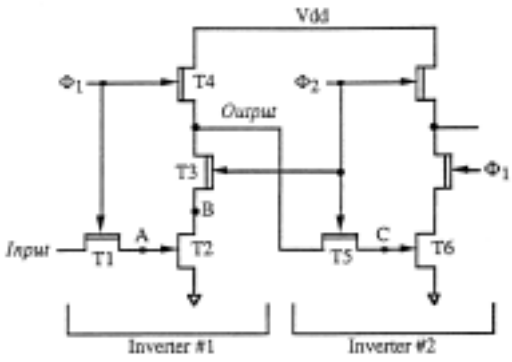



Figure5.11. Two TDFL inverters

While static circuits in Gallium Arsenide MESFET technology have been quite exhaustively studied, dynamic circuits have been relatively unexplored. Only, a relatively low number of dynamic GaAs logic circuits works have been reported. Dynamic logic gates topologies are enough different from static gates. Dynamic gates require a clock to perform combinational logic. Additionally, dynamic gates can be non ratioted, it means, the logic levels are not determined by length and width ratios of load and switching FETs.

Differential Pass Transistor Logic (DPTL) [24] approach, which was derived from DPTL CMOS technology, has shown reasonable results. DPTL GaAs technology shows significant advantages in speed, area and power consumption. Its operation is similar to the NMOS pass element. As well as other pass transistor approachs, GaAs DPTL makes extensive use of complementary input signals. A differential pass element and a three input xor logic are shown in figure 5.10.

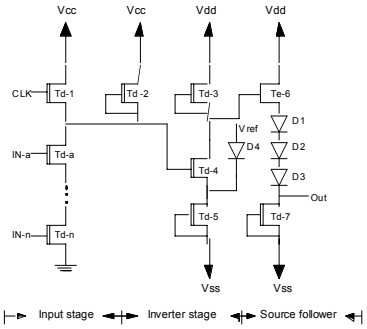



Figure 5.12 TTDL schematic.

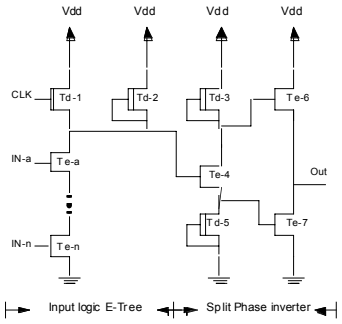



Figure 5.13. SPDL schematic.

Within the dynamic families Two-Phase Dynamic FET Logic [25] (TDFL - figure 5.11) shows a configuration which is based on an extension of NMOS dynamic circuit techniques. This approach offers advantages in power dissipation but is limited to basic logic gate implementations. Another drawback of the TDFL gate is that it is sensitive to clock skew andclock feed through problems which may result in errors in logic evaluation.

Trickle Transistor Dynamic Logic (TTDL) [26] (figure 5.12) was proposed as an alternative logic configuration. This approach uses a self-biased transistor to compensate for leakage loss and an external voltage reference to control the operation of diode inserted in the inverter stage. The performance of the logic gate using this configuration could be affected by variations in the external voltage reference.

Additionally, the necessity for several supplies complicates the design. In order to overcome those limitations, Law [27] proposed in 1994 its Split Phase Dynamic Logic (SPDL - figure 5.13). In this new configuration the diode controlled buffer is replaced with a split phase inverter eliminating also the external voltage reference.

However, there are several drawbacks associated with dynamic gates: the need to supply clock signals to every gate in the circuit. Also, dynamic circuits have a minimum frequency of operation associated with the leakage of charge from isolated nodes in the circuit. Besides, two factors degrade performance as chip complexity is increased. First, a higher percentage of the chip area must be devoted to interconnect wiring and the increased capacitive load on the logic gate degrades speed performance. Second, limits on the chip thermal dissipation restrict the drive current available from each logic.

The gate delay caused by the capacitive loading of the wiring demand on the logic gate is a function of the current drive of the logic gate, the logic swing and the average wiring capacitance per cell. The first two factors depend upon the type of GaAs logic gate used while the third factor can be estimated as a function of gate count [28].

### 5.4. Available technologies.

The choice of a particular FET device for implementing integrated circuits is dependent on the circuit performance requirements of the integrated circuits and the fabrication process of the device. If the circuit requirements are high speed and low power, similar device characteristics must be desirable. Table III relates the circuit requirements and consequent device features.

Table III. Circuit and device requirements for Very High Speed and Low Power ICs [29].

| **Circuit requirements.** | **Device features.** |
|---|---|
| Small logic voltage swing.<br>Low Power.- $1/2\ C\ \Delta V^2$ | Very uniform threshold voltages for active devices. |
| Low device and parasitic capacitance | Low input capacitance devices.<br>Semi-insulating substrate for low parasitic |
| High switching speeds with reasonable Fanout loading at low switching voltages | Very high current gain bandwidth.<br>Very high power bandwidth.<br>Fast increase in transconductance above threshold. |

| **Device electric characteristics.** | **Physical parameters.** |
|---|---|
| High transconductance at control voltages Low above threshold. | High carrier mobilities. |

| | |
|---|---|
| Very uniform threshold voltages. | Very low threshold voltage sensitivity to horizontal Geometry variations.<br>Low threshold voltage sensitivity to vertical Geometry variations and doping variations. |
| Very low input capacitance. | Small geometries and low carrier storage effects. |
| High current and power gain bandwidths. | High carrier mobilities and saturation velocities.<br>Small geometries.<br>Good thermal design. |

Considering device and physical features we can note that the appropriated process selection is strongly important. A validated technology intended for easy structuring and improved reliability featuring small sensitivity to temperature and process variation is required. These issues, as well as performance goals in terms of small area, low power and high speed are very important for widespread industrial acceptance of GaAs technology.

Gallium Arsenide integrated circuits and systems require quite different design approaches from these that are common in silicon. System clock speeds running at several Gigahertz mean that every element of the system including the interconnect wiring itself must be designed and treated as a complex circuit element.

Although, in last years the technology was confronted with some problems and constraints, the recent advances in material processing, fabrication, testing and packaging have brought about an environment whereby it is now possible to design in this medium. More number of foundries providing Gallium Arsenide fabrication are now in operation such as: Vitesse Semiconductor Corporation, Systems and Process Engineering Corporation - SPEC, Anadigics, TriQuint semiconductor, Motorola Inc. Additionally, many other computer companies such as Digital Equipment Corporation, Compaq, Computer and others, are working in partnership with GaAs vendors to develop devices for speed critical bottlenecks such as data encryption, error detection and correction and cache control motivated by demands placed by ever-faster microprocessors [30]. All that, makes possible the realisation of a new generation of products with lower levels of power dissipation.

Digital GaAs emerged as the starting material for integrated circuits with one million or more transistors per chip. Particularly, Vitesse semiconductor Corp. produces GaAs ICs that form part of a supercomputers central processing unit. These chips use four levels of metal and contain more than 1.2 million of transistors [31]. HGaAs III Vitesse process is one of the most mainly used technology in digital GaAs applications. The Gallium Arsenide foundry “Vitesse Semiconductor Corporation” has finished 1996 with sales of 65 million of dollars, which represents about 51% of growing respect to 1995. This data is due to the fact digital GaAs to communication application [32].

The GaAs competition in those applications is up-market silicon bipolar technology rather CMOS. TriQuint semiconductor Inc, also makes GaAs ICs for both computers and digital to communication applications [33]. In Europe, it is now six year since the EuroGaAs Initiative was launched. That EC supported programme was oriented to enhance the manufacturing capability of the seven major European GaAs component manufacturers. In table IV, the progress and the impact on the world wide merchant market is shown.

Table IV. European GaAs Foundries - World wide merchant market [34]

| Year. | World Wide Sales (million) | European Sales (million) | Market share. (%) |
|---|---|---|---|
| 1992 | 87 | 1.8 | 2 |
| 1993 | 100 | 6 | 6 |
| 1994 | 123 | 12.3 | 10 |
| 1995 | 208 | 28 | 14 |

The Vitesse H-GaAs III technology is offered as an advanced process for the fabrication of high performance GaAs VLSI digital circuits. Five levels of interconnection are provided in order to design low power dissipation and high speed at high levels of integration, (more than 1.000.000 active devices can be implemented).

The designers can use two versions of the H-GaAs III process: the first one uses polyamide as the intermetal insulator reducing the routing capacitance and leading faster loaded gate delays, its maximum temperature operation is 100° C. The second version, uses $SiO_2$ as the intermetal dielectric obtaining circuits that are ~20% slower but can operate over a wider temperature range (~125° C).

Three active devices are available: an enhancement mode MESFET, a depletion mode MESFET, and a Schottky-barrier diode. The transistors are used for switching and as active loads while the Schottky barrier diodes are used primarily for level shifting. The minimum sized devices have nominal gate lengths of 0.6 μm [35].

Additional, H-GaAs III is a self-aligned process which reduces the effects of series resistance. Vitesse H-GaAs III allows five levels of interconnection where M1, M2 and M3 levels are typically used for signal routing while M4 and M5 are used for power and ground busses. Circuits containing up to several thousand gates can be designed using a two version using only 9 mask layers.

Currently, GaAs VLSI manufacturing has more in common with silicon CMOS technology than with earlier GaAs. However, fewer steps are required to produce GaAs circuitry due to only four mask levels are needed to define the GaAs transistor compared with six for the silicon

device. A cross sectional representation of a FET after completion of the fabrication cycle is shown in figure 5.6.

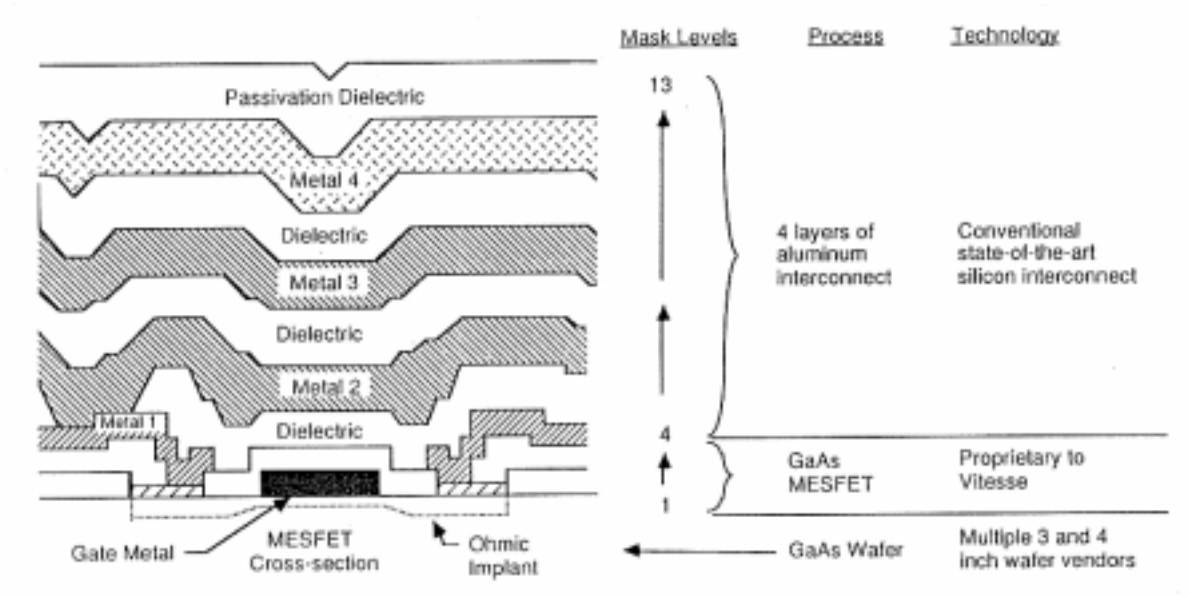


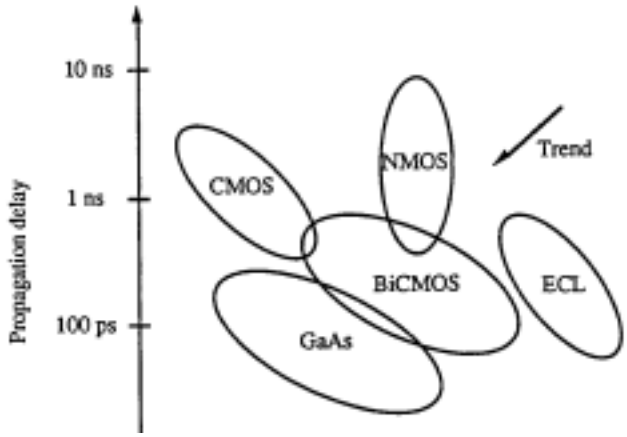

Figure 5.14. GaAs MESFET structure of Vitesse technology [35].

In table V, some MESFET model parameters of the Vitesse process are shown.

Table V. MESFET Model Parameters.

| Device | Param. | Units | Slow | Typ. | Fast |
|---|---|---|---|---|---|
| E-JFET | $V_{T0}$ | V | 0.35 | 0.22 | 0.15 |
| | $I_{DS[max]}$ | μA | 100 | 180 | 600 |
| | $R_S$ | Ω | <170 | | |
| D-JFET | $V_{T0}$ | $A/V^2$ | -0.63 | -0.87 | -1.0 |
| | $I_{DS[max]}$ | μA | 200 | 600 | 1400 |
| | $R_S$ | Ω | | | |

Currently, in communications, a prime use for high-performance digital electronics is in the interface to the optical-fibre trunk lines of serial data links. At this interface, digitised phone conversations, e-mail, bank transactions and more are multiplexed into a single serial high-frequency data stream, converted into optical pulses and transmitted along optical fibre cable, an application that is ideal for GaAs VLSI with its optical affinities. An important criterion of the



device technology, but not the only criterion, is the gate delay at a given power. In figure 5.7 is shown the performance projection for some technologies.

Figure 5.15. Speed-Power Performance of Si. and GaAs ICs. [17].

In the area of the devices for handy phone and PCS/PCN application, GaAs MESFETs, HEMTs and HBTs have been actively investigated to have low voltage operation and low power consumption. The market has started to expand. In the field of the data processing, GaAs LSIs have established its position for the highest speed logic in a computer system..

A GaAs device is becoming inevitably necessary for the equipment we have to use in daily life such as for communication, broadcasting and data processing. In the field of a supercomputer, GaAs LSIs have been used in high speed data processing and will be used widely in the near future [36].

The design of power-efficient high performance digital electronics is a relatively new area of interest, driven by the recent growth in battery-powered computer based products. Power-efficient design will play a key role in making these portable products feasible. Battery technology is being improved, but it is unlikely that a dramatic solution to the power problem is forth coming. This puts a sever constraint on the power that may be consumed by these devices.
Since electrical power is a physical quantity and since it is converted into heat in the resistive switches and wires of the physical device a thorough understanding the physical effects is necessary. Focuses on the design for low power at the circuit and logic level, we must determine the main reasons of power dissipation and the best form of overcoming those problems: drawbacks as clock frequency, delay and complex architectures. Besides, some strategies oriented to reduce the circuit activity or capacity without sacrificing performance will be briefly reviewed.

## 5.5. Low Power GaAs Strategies.

Four approaches are commonly used to overcome the problem of power dissipation: dynamic switching power due to charging/discharging circuit capacitances, leakage currents power from reverse biased diodes and subthreshold conduction, short circuits current power due to finite signal rise/fall times and static biasing power found in some types of logic styles. Each topic is briefly analysed.

### 5.5.1. Dynamic Switching Power.

The principal requirements of high speed VLSI circuits are: a small feature size, high process yield and extremely low dynamic switching energy. This last characteristic becomes the most important of all. The **dynamic switching energy** or **power-delay product**, is the minimum energy that a gate can dissipate during a clock cycle. So, a power dissipation for a chip with $\mathbf{N_g}$ gates with an average gate clocking frequency $\mathbf{F_c}$ will be:

$$P_{(chip)} = 2 \times N_g \times F_c \times (P_d \times t_d) \qquad (3)$$

where: $P_d$ ------> Dynamic power dissipation

$t_d$ ------> Clock period

Dynamic switching energy requirements for high speed VLSI are quite severe. For that reason, in several technologies many efforts are being done to reach a lower speed-power product. So, technologies allowing combine high levels of integration with high speed performance are very attractive.
One possibility to accelerate the switching is the reduction of the maximum input level, for example by reduction of the input voltage range. Another possibility is to reduce the capacitive loads at the switches using smaller transistors widths. A third alternative is the optimisation of the switch or the application of a new circuit concept.

The logic switching speeds and speed-power products of the FET gate are dramatically in GaAs. For the same logic voltage swing, a GaAs MESFET would give about 4 - 6 times higher switching speeds than silicon counterpart. The factor of switching reduction is approximately 1 / N, where N is the number of loading gates.

From the point of view of dynamic switching, a larger gain in power reduction can be achieved at the register transfer and higher design levels being the task of the design the definition of the clocking scheme and the datapath architecture, the designer's freedom lies in the utilisation of parallelization, sequentialization and pipelining.

Power efficient state encoding is the consideration of state and state transition probabilities; the number of transition can be reduced by using combinational circuit optimisation: technology dependent and technology independent techniques for the minimisation of the power consumption of combinational circuits. In many cases the power can be further reduced by state reencoding or retiming, for that reason, several authors have used RNS notation to represent the operands. Also power reduction is achieved by shutting down parts of the circuit

### 5.5.2. Short-circuit current Power.

Second, the speed or for that matter the delay of this circuit type is unaffected by power supply voltages down to about 1V, meaning that power dissipation can be reduced without degrading performance. This fact is consequential, for as features size are reduced in any VLSI technology, the external voltages must be reduced, otherwise the electric fields internal to the transistor will not remain below breakdown. This fact will loom larger as VLSI feature sizes descend into the deep submicrometer range, below 0.5 μm. This is not possible to do in CMOS technology. In CMOS chips, a shrinking supply voltage slows the signal unless offset by shrinking transistors in successive generations of technology.

### 5.5.3. Leakage current Power.

Since the dynamic power consumption of a circuit is proportional to a $CV^2f$, low power and low voltage digital design has to be performed at several levels such as architecture, logic and basic cell levels in order to minimise these three terms, capacitance, voltage and frequency. However, activity must be also considered as well as the static power consumption due to switching and leakage currents.

The basic physical mechanism for subthreshold current, that is thermionic emission/diffusion over a potential barrier, is the same for both MOSFETs and MESFETs. However, due to the larger threshold voltage of typical MOSFETs compared with threshold voltage of E-type MESFET (0.7V vs 0.2V respectively) and the small logic low level obtained by CMOS, a more negative $V_{gs}$ - $V_t$ is obtained in the off state for CMOS than for DCFL GaAs circuits. Therefore, the observed drain current in the off state is five to six orders of magnitude larger than that of the MOSFETs. This implies that the subthreshold leakage currents easily dominates circuit operation. On the other hand, due to the Schottky barrier in the gate electrode of the MESFET, leakage current flows into the gate.

Low standby power consumption with impact in power/sleep down modes and their wake up latency, higher Vt for process to minimise the transistor leakage currents at the expense of reduced speed and possible constraint for lower system speed, are also some proposed techniques. The main goals of such design methods are the activity reduction as well as the capacitance and leakage currents reduction. However, supply voltage reduction is the most effective way to save power.

### 5.5.4. Static biasing Power.

Considering that in the electric field regime of below $10^4$ V/cm, electrons in GaAs move at three to five times the velocity of electrons in silicon. This advantage will be most apparent in low voltage applications. The power consumption of digital integrated circuits, such as logic, memories and digital signal processors, is directly proportional to the square of supply voltage. The supply voltage squarely influences the dynamic power consumption and for this reason it is obvious to look for the lowest possible supply voltage. Therefore, reducing supply voltages, the power consumption of digital electronics is strongly reduced .
In summary the characteristics of a power-efficient system as: a small area, low voltage operation, high code density, flexible clocking and goods design tools, are also very important factors to achieve power reduction.

Low power reduction can be achieved minimising activity and capacitance in digital logic modules as well as in optimised libraries. The low power optimisation criteria must be well specified, since the adequate and relevant solutions, circuit techniques or software tools can be very different. It means, optimised design of architectures and cells is mandatory, but it is not necessarily the case that the optimum organisation for performance is the same as the optimum organisation for power-efficiency. First one, involves techniques to reduce the dissipated power based on the technology used, like as device scaling or variations of the supply voltage.

Another one, consider different strategies or design style, e.g., static versus dynamic logic, synchronous versus asynchronous circuits. Also are included the architectural conception which reduce the global power consumption of the system optimising critical paths or applying several pipeline stages. This approach, contains different techniques for reducing power consumption by means of improving the underlying algorithms which looking for the optimisation of the computation complexity.

### 5.5.5. Asynchronous design.

Since clocked circuits waste power by clocking all parts of the chip whether or not they are doing useful work, recently new alternatives to fully synchronous electronic circuits have been studied. Those alternatives are known as asynchronous circuits. In several applications asynchronous circuits are more power-efficient than their clocked equivalents.

Additionally, clocked design assumes global synchrony, and any deviation from this assumption must be compensated by margins whitin each clock cycle, reducing the circuit's performance. On high-performance chips the clock drivers are also responsible for a significant proportion of the total power consumption.

Although gated clocks can reduce the wastage, they are generally only practical at a coarse granularity. Besides, simple clocked design requires the same clock to be applied to all parts of the circuit, but its frequency can only be optimised for one function. So, many parts of the circuit must operate at a higher frequency than is necessary, wasting power. Asynchronous circuits, on the other hand, are inherently data driven and are active only when doing useful work. Parts of the circuit that receive less data will automatically operate at a lower average frequency. So, low average consumption with impact on reduction of overall power consumption will be achieved.

The basic idea behind low-power processor, coprocessors or memories is to reduce the number of basic steps and clock cycles for the execution of a given task. In addition to these architectural issues, important power savings are obtained by lowering the supply voltage.

In next chapters some low power considerations will be taken into account to design, fabricate and test a high speed and low power Gallium Arsenide prototypes, exploring new low power GaAs approaches. Mainly, two functional blocks used in performing modular exponentiation at very high speed. The significance of the work lies in the combination of a new architecture for accelerating modular exponentiation and low power GaAs design strategies.

All aspects of design for low power in GaAs based systems ranging from process technology over circuit techniques up to architecture and systems design are considered. Like as CMOS technology, a decreasing power dissipation per logic function has become as primary concern in virtually all GaAs chips designed today.

## 5.6. References


[1] R.V. Tuyl, C. Liechti, “High Speed Integrated Logic with GaAs MESFETs”, IEEE Journal of Solid-State Circuits. Vol.SC-9, pp.269-76, October, 1974.

[2] S. M. Sze, “Physics of Semiconductors Devices”, (2 Ed.), Wiley Pub., New York, 1988.

[3] S. I Long, S. E. Butner, “Gallium Arsenide Digital Integrated Circuit Design”, Mc-Graw-Hill Publishing Company, 1990.

[4] J.S. BLakemore, “Semiconductor and other Major Properties of Gallium Arsenide”, Journal of Applied Physic. Vol.53, pp.R123-R181, October , 1982.

[5] O. Wing, “Gallium Arsenide Digital Circuits”, Kluwer Academic Publishers, London, 1990

[6] J.A. Cooper, Jr and D.F. Nelson, “Measurement of the High-Field Drift Velocity of electrons in Inversion Layers of Silicon”, IEEE Electronic Device Letters. Vol.EDL-2, No. 7, pp.171-173, , July, 1981

[7] R. Zuleeg, “Radiation Effects in GaAs Integrated Circuits”, VLSI Electronics: Microstructure Science, Vol. 11, Academic Press, Orlando 1985.

[8] W. Roesch, “GaAs IC Reliability, The Next generation”, Proc. IEEE Gallium Arsenide Symposium, USA, 1993.

[9] A. Barna, C. Liechti, “Optimization of GaAs MESFET Logic Gates with Subnanoseconds Propagation Delays”, IEEE Journal of Solid-State Circuits. Vol.SC-14, pp.708-15, August, 1979.

[10] R.C. Eden, “Capacitor Diode FET Logic (CDFL) Circuit Approach for GaAs d-MESFET ICs”, Gallium Arsenide IC Symposium Proc., Boston, Mass., pp 11-14, October, 1984.

[11] R.C. Eden et al, “Low Power Depletion Mode Ion-Implanted GaAs FET Integrated Circuits”, IEEE Transaction on Electron Devices , Vol.ED-24, pp.1209, September, 1977.

[12] S.I. Long et al, “MSI High Speed, Low Power, GaAs Integrated Circuits using Schottky Diode FET Logic”, IEEE Trans. on Microwave Theory and Techniques, Vol.MTT-28, No. 5, pp.466, May, 1980.

[13] P.J.T. Mellor, A.W. Livingstone, “Capacitor-Coupled Logic using GaAs Depletion mode FETs”, Electron Letters, Vol.16, pp.749, September, 1980.

[14] A.D. Welbourn et al, “A High Speed GaAs 8-Bit Multiplexer using Capacitor-Coupled Logic”, IEEE Journal of Solid-State Circuits. Vol.SC-18, No. 3, pp.359, June, 1983.

[15] R.C. Eden, B.M. Welch, R. Zucca, S.I. Long, “The Prospects for Ultrahigh-Speed VLSI GaAs Digital Logic ”, IEEE Transaction on Electron Devices, Vol.ED-26, pp.299-317, April, 1979.

[16] A. Peczalsky et al, “Design Analysis of GaAs Direct Coupled Field Effect Transistor Logic”,, IEEE Trans. on Computer Aided Design, Vol.CAD-5, No. 2, April, 1986.

[17] K. Esrhaghian, “Design Methodology and Layout Style for Very High Speed Circuits and Subsystems”, Internal Report, University of Adelaide South Australia, January, 1992.

[18] W.R. Wisseman, W.R. Frensley, “GaAs Technology Perspective”, VLSI Electronics Microstructure Science, Vol. 11, Academic Press, Orlando, 1985.

[19] H. Nakamura et al. “A 390 ps 1000 Gate Array Using GaAs Super-Buffer FET Logic”, Dig. of Tech. Papers, Int. Solid State Circuits Conference, pp. 204-205, February, 1985.

[20] S. Katsu, S. Nambu, A. Shimano, G. Kano, “A Source Coupled FET Logic - A New Current Mode Approach to GaAs Logics”, IEEE Trans. Electron Devices, vol. ED-32, pp1114-18, June, 1985.

[21] A. Tamura et al, “High Speed GaAs SCFL Divider”, Electronic letters, Vol. 21, No. 5, pp.605-606, July, 1985.

[22] Chandna, A.; Brown, R.B.; Putti, D. & Kibler, C.D. Power Rail Logic: a low power logic style for digital GaAs circuits. IEEE Journal of Solid-State Circuits. Vol.30, No.10, Oct.95, pp.1096-100.

[23] R. Kanan, B. Hochet, M. Declercq, Pseudo-Complementary FET Logic (PCFL): A Low-Power Logic Family in GaAs”; IEEE Journal of Solid-State Circuits. Vol.31, No.7, pp.992-999, July, 1996.

[24] J.H. Pasternak, C.A.T. Salama, “GaAs MESFET Differential Pass-Transistor Logic”, IEEE, Journal Solid State Circuits, Vol. SC-26, pp. 1309-1316, September, 1991.

[25] K.R. Nary, “Gallium Arsenide Metal-Semiconductor Field Effect Transistor Dynamic Logic Gate Topologies”, Ph.D. These, December, 1992.

[26] Hoe, D.H.K. & Salama, C.A.T. “GaAs Trickle Transistor Dynamic Logic”. IEEE Journal of Solid-State Circuits. Vol.26, No.10, pp.1441-48, Oct. 1991

[27] Law, O.M.K. & Salama, C.A.T. “GaAs Split Phase Dynamic Logic”. IEEE Journal of Solid-State Circuits. Vol.29, No.5, pp.617-22, May, 1994.

[28] P.T. Greiling, C.F. Krumm, “The Future Impact of GaAs Digital Integrated Circuits”, VLSI Electronics Microstructure Science, Vol. 11, Academic Press, pp. 133-171, Orlando, 1985.

[29] N. Kanopoulos, “Gallium Arsenide Digital integrated Circuits: A System Perspective”, Prentice Hall, North Carolilna, 1988.

[30] I. Deyhimy, "Gallium Arsenide Joins the Giants", IEEE Spectrum, February, 1995.

[31] S. Lande, "Customer acceptance of GaAs ICs", Proc. IEEE European GaAs and related III-V Compounds Application Symposium, Torino, Italy, 1994.

[32] “Circuits GaAs: Ventes en hausse de 51% pour Vitesse Semiconductor”, Electronic International, HEBDO, No. 237, Octobre 10, 1996.

[33] T. Smith, "Wafer Fab Line Yield Improvement at TriQuint Semiconductor", Proc. IEEE European GaAs and related III-V Compounds Application Symposium, Torino, Italy, 1994.

[34] J. Turner, “The competitiveness of European GaAs Foundries”, Proc. IEEE European Gallium Arsenide and related III-V Compounds Applications Symposium, Paris, France, 1996.

[35] Vitesse Foundry Design Manual, Vitesse Semiconductor Corporation, May, 1995.

[36] M. Fukuta, "Recent Development of GaAs Devices in Japan", Proc. IEEE European GaAs and related III-V Compounds Application Symposium, Torino, Italy, 1994.

# 6. A Low Power Two-Single Port GaAs Memory Cell.

## 6.1. Introduction

The second great functional block of the exponentiation system when implemented in GaAs technologie responsible of a considerable power consumption would be the cache RAM. In this chapter a new low power and high speed GaAs memory cell is presented. The cell memory was conceived to be used in small size cache memory.

Due to a great demand for low power and high speed digital system, low power GaAs LSI technology is becoming an important and growing area of electronics. In particular, GaAs SRAM is an area of this technology in which considerable attention has been focused [1][2][3]. For GaAs SRAMs there has been also a strong requirement for low power. For that reason, in early days, GaAs SRAM development has been focused on low power applications, especially with very low standby and data retention power.

Much effort has been dedicated to the development of GaAs SRAMs and some remarkable progress in power reduction [4][5], performance [6][7], radiation [8][9] and temperature [10] tolerance have been obtained. Nowadays, more emphasis has been placed on low-power, high-speed rather than large memory capacity, primarily led by cache applications in high speed microprocessors. Consequently, some of currently developed GaAs MESFET static memories are restricted to small static memories [11][12]. Several high-speed on-line GaAs memories are being designed to be applied to high-speed GaAs microprocessors which use small amount of memory on-chip in order to exploit the hierarchical high-speed memory benefits.
Six transistors conventional memory cell usually has been used to implement static RAM, however this cell presents important limitations to implement GaAs SRAM structures. On the one hand, high speed GaAs direct coupled conventional cell configuration require high power consumption. On the other hand, as is shown in figure 6.1, there are some additional problems in using that conventional cell. First one, when the word line level is "high", the low and high

nodes of the cell become capacitevely coupled to the bit lines (i). Current is also injected into the cell through the direct-biased gate-source diode of the access transistor (ii), causing one of the more important mechanism that can generate destructive readout which is itself an strong yield limiting factor for GaAs SRAMs.

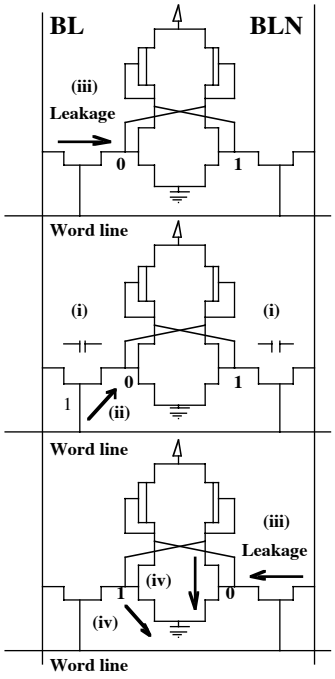



Figure 6.1. Conventional memory cell limitations.

In addition, MESFET leakage current flows through the "low" node from the bit line in non selected cells, then the number of memory cells and the combination of the stored data in each column define the sum of the leakage currents per bit line and not only the leakage currents in the individual access transistor (iii). Moreover, a reduction of the "high" internal node level can be caused by the increase both of the drain to source leakage current in the driver enhancement FET and of the Schottky current from the gate to the source in the driver enhancement FET in the succeeding stage (iv).

Due to the temperature variation, the bit-line potential, a stability of the memory cell and in consequence the circuit operation of GaAs SRAM are strongly affected by the leakage current increment in the access transistors of memory cells.
Several authors have proposed diode or ground shifting techniques to reverse bias the non selected access MESFET [7][8][10][12], in order to limit the leakage current flowing through the transistor. Other have applied built-in redundancy [2] or current mirror techniques [13][14] to GaAs SRAM [11], but additional control logic or several voltage levels are required increasing the complexity and the access time. As is known, there should preferred be only one supply voltage to minimise the access time.

Recent exploratory achievements in the movement toward low voltage operation seemingly give promise of future improvements. Low voltage operation has already been one of the most important design issues for integrated circuits, since it is essential not only to further reduce power dissipation, but also to ensure reliability for miniaturised devices.

In this chapter the characteristics of an experimental and novel low power high-speed GaAs Two-Single Port static memory cell [15] which allows significant power dissipation reduction by reducing both its operating voltage and leakage current flow are discussed. The memory cell has been implemented using a mix of several techniques already published in order to overcome some of their principal drawbacks related to ground shifting, destructive readout and leakage current effects.

The cell size is 36 x 37 $\mu m^2$ using a 0.6 $\mu$m GaAs MESFET technology. An experimental 32 word x 32 bit array has been designed. From simulation results, an address access time of 1ns has been obtained.

A small 8 words x 4 bits protoype was fabricated. The cell can be operated at the single supply voltage from 1V up to 2V. The evaluation is provided according to the functionality and power dissipation. Measured results show a total current consumption of 14 $\mu$A/cell when operated at 1V.

Good performance and operational margin over a reasonable temperature range are its principal features. The final 1 kbit SRAM array can be used in high speed systems with sub 2 ns on-line memories requirements. The cell structure, its operation and some experimental results are presented.

## 6.2. Memory cell design

GaAs digital systems can suffer from several technology related-problems, and RAM memory cells are obviously included. Therefore, the design of the memory cell must be taken into consideration that technology-related issues during the architectural conception phase. The usual design criteria of a static random access memory are density, power consumption and read and write access time. To minimise power dissipation, from point of view of the cell, the leakage currents and operating voltage must be reduced. Leakage current reduction is obtained back biasing the Schottky diodes of the cell rows which are idle at a given time to minimize the Schottky diode currents. So, not significant Schotkky diodes current would be drawn.

From point of view of the decoding and addressing blocks as well as sense amplifier must be disconnected from the supply voltage during its standby state. In order to minimise the access time, the pull-up and pull-down delays of the circuits should be small, as was mentioned, there should preferred be only one supply voltage. All mentioned criteria design are considered in the new cell configuration. The schematic of the proposed high-speed new cell is shown in Figure 6.2 [15].

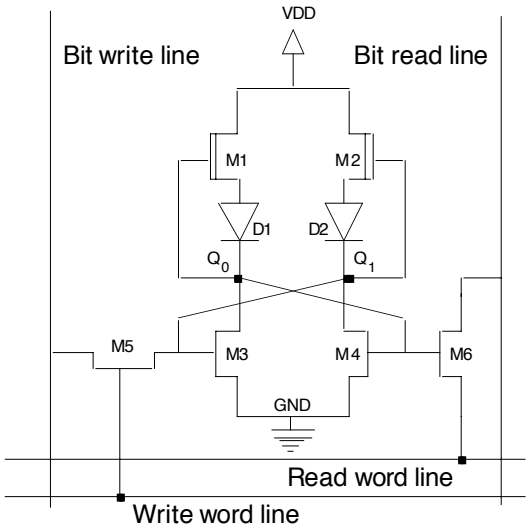



Figure 6.2. New cell diagram.

The cell consists of four enhancement MESFETs, two depletion MESFETs and two diodes. Source-gate back biasing in the depletion transistors *M1* and *M2* are used as sub threshold current reduction circuit in order to reduce the power dissipation of the cell [16]. The back biasing is obtained using *D1* and *D2* diodes. The depletion transistor and diode combination acts as a weak pull-up current supply and must be designed considering the pull-up time requirements, power reduction and the necessary current to compensate the sub-threshold leakage and Schottky currents through the enhancement devices in order to keep the high level in the respective node. The pull-up delay time is defined as the time elapsed when the output voltage reaches some fraction of its steady sate value. From (*eq. 1* ), we can observe that the pull-up delay is proportional to the ratio $W_D$ / $W_E$. So, to reduce the pull-up delay that ratio must be large.

The weak current source formed by *M1-D1* must provide a quite larger equivalent current than the *M4* and *M6* gate to source and gate to drain Schottky inverse currents plus *M3* source to drain sub threshold current. On the other hand, the weak currents source formed by *M2-D2* must provide a quite larger equivalent current than the *M5* and *M4* sub threshold currents (*eq. 2* ) plus *M3* gate Schottky currents (*eq. 3* ) .

All currents depend on the transistor sizes and their biasing voltages. So, a transistor saturation region current (*eq. 4* ) and both direct and reverse diode Schottky current expressions must be considered. The voltages variables in next equations $\{U, U_{ds}\}$ are normalised by thermal voltage.

$$t_{pu} = f(W_D / W_E) \tag{1}$$

$$I_{sub} = C1.W.n_o.(1 - e^{-U_{ds}})/L \tag{2}$$

$$I_{Sh} = W.L.(C2.e^{-U_{ds}})\, e^{U} \tag{3}$$

$$I_{ds} = \beta.(V_{gs} - V_T)^2(1 + \lambda V_{ds})\tanh(\alpha V_{ds}) \tag{4}$$

where $C1 = 2.L_B.q.Dn$ and $C2 = q.N_D.V$

$W_D$ = channel width depletion transistor.

$W_E$ = channel width enhancement transistor.

$n_o$- equilibrium minority carrier concentration;

$L_B$ - the extrinsic Debye length;

$Dn$ - the diffusion constant;

$N_D$ - the doping concentration;

$V$ depends both drift and diffusion velocities.

The latch formed by the cross-coupled transistors *M3* and *M4*, provides a robust storage element with reduced static power dissipation. Transistor *M5* implement one write-only port, while transistor *M6* acts as read-only port.

The operation of the cell is straightforward. The read and write cycles occur on opposite phases of a system clock. The write cycle begin on the rising edge of the clock and the read cycle on the falling edge. The cell mixes the advantages of the conventional and full current mirror cells overcoming some of their drawbacks.

### 6.2.1. Read operation

In order to reduce the power consumption of the non selected cells, the following reading mechanism is used. The cell is read by pulling down the read word line which is maintained at 1V before the read cycle. The word line for selected row is lowered to 0V, while the word lines of the remaining non selected rows are held at 1V. So, it does not make any difference if the stored data into the internal node $Q_o$ corresponds either a low or high logic levels. *M6* transistor Schottky diodes will be always back biased.

In conventional cell during read operation, when the word line level is "low" and the memory cell store "low" data, it should be noted that the leakage currents flows through the access pass transistor, due to the bit line level being at higher potential, and as the number of cells attached to a column is increased, leakage currents through non selected access transistor can overwhelm the active current of the selected cell [17].

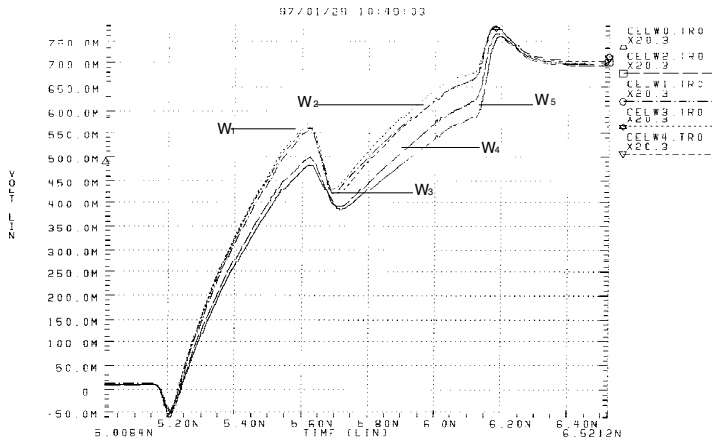
In this configuration the gate-drain and gate-source diodes of the *M6* access transistors of non selected cells are reversed biased appearing as additional capacitance to the storage node overcoming the mentioned conventional cell problem. This capacitive coupling from the read word lines is less than that in conventional cell. On the other hand, the access transistor of the selected cell cannot inject current into the storage nodes causing a non destructive read operation.

If the cell stores a low value at $Q_0$, not significant currents appear through access transistor and the precharged bit line value is held. In this case, precharge operation not only speeds up the read access reading operation of high logic level, but also eliminates the possible charges accumulated on the bit line.

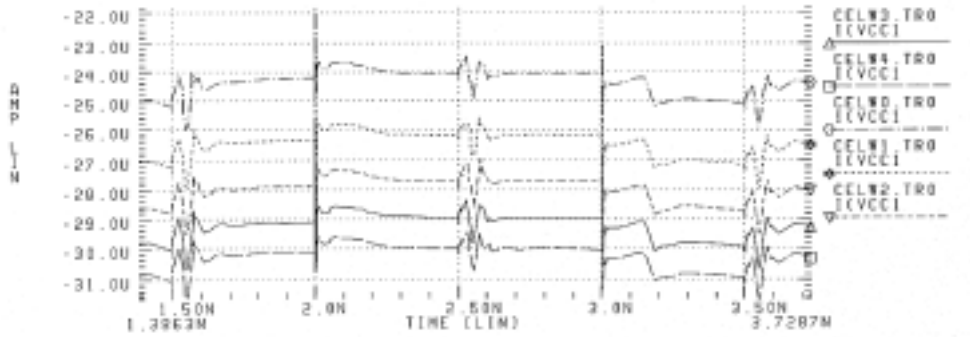
Contrary to high level reading operation, if the cell stores a high value at the internal node $Q_0$ (low in $Q_1$) and the read word line is lowered to 0V, a saturation current flows through *M6* transistor pulling down the bit read line which must be precharged at 1V before each read operation.

This reading mechanism occupies a significant portion of the total time, due to the fact that the amount of drain-to-source current is determined by its drain-to-source and gate-to-source voltages, a reduction of 9% in the voltage values generates a reduction of 30% in the drain current. On the other hand, the MESFET drain current also varies with channel width (W), thus it is possible to avoid an excessive diminution of the drain current by manipulating the parameter W.

Because the read operation is made in single-ended mode and due to both operating voltage reduction and reverse bias of depletion transistor, we can observe a pull-up delay measured from node $Q_0$ slower than that in $Q_1$. As was mentioned before, the pull-up delay reduction is achieved by increasing the ratio $(W_D/ W_E)$. Nevertheless, the current consumption of the cell will be increased as the ratio is increased. This is confirmed in figure 6.3 (a) and 6.3(b) in which the computed results of the pull-up delay and the current consumption for different values of the ratio are shown.




(a)




(b)

Figure 6.3. Pull-up delay (a) and cell power dissipation (b) for different W ratios.

It is seen that the pull-up delay is not a strong sensitive function of the ratio $(W_D/ W_E)$; however some trade-off will have to be made. In recently GaAs applications, an ultrahigh-speed circuit combined with low power strategies is becoming the principal concern. Extra access transistors are added in order to increase the power-delay product [18]. For pull-down delay of the cell, any problem can be present. Unlike the conventional cell, the *M6* access transistor can be dimensioned independently of the driver transistor.

### 6.2.2. Write operation

The write operation is similar to that in a conventional six-transistor cell; data are placed on the bit write line, and the write word line is raised; the cross-coupled transistors force the internal nodes to change to appropriated voltage levels maintaining the state of the cell. In order to obtain a high speed write operation, the access transistor *M5* must be dimensioned respect to *M3* pull down transistor.

Usually, a ratio of *M3 = 3M5* is required. To write a low level, the low voltage bit line is connected through one of the pass transistors to a cell storage node pulling that storage node

low and causing the opposite cell storage node to be driven high. In order to write a high level it must be guarantee that, $V_G \geq V_i + V_{TH}$ if $V_{BL} > V_i$, where $V_G$ is a gate voltage of access transistor, $V_{BL}$ is a bit line voltage level and $V_{TH}$ is the threshold voltage, Vi would be the internal storage node. Using the mentioned voltage levels the write operation is reliable.

Reading data from the cell involves discharging the bit read line through a *M6* access transistor. Transferring data from the bit write line to the cell involves discharging the storage node through a *M5* pass transistor. The resistance of the channel of a transistor is a non linear function of the drain source voltage and is given in the region of operation by:

$$r_{ds} = [\, \partial V_{ds} / \partial I_d \,] \mid V_{gs}=cte \qquad (5)$$

So, neglecting the term $(1+\lambda Vds)$ in (4) the expression would be:

$$r_{ds} = \cosh^2(\alpha V_{ds}) / \beta.\alpha.\, (V_{gs} - V_T)^2 \qquad (6)$$

As the voltage drop across $\mathbf{r_{ds}}$ is reduced, a lower channel resistance is obtained. For cell currents between 10 and 20 $\mu$A, this writing mechanism allow faster write times than writing mechanism used in full mirror cells. Unlike the reported full current mirror cell [4], where the gain in speed of the cell could be a deceptive since the output bit line capacitor must discharge through a number of series connected diodes making the pull-down delay too large, in this configuration no multiple diodes are present in writing process. Besides, the cell grounds not must be driven causing that less control circuitry would be required. In this new cell, only a single voltage of 1V is used, but can also be operated at 2V. In general, the memory cell designed presents good stability and access speed.

The noise margins of the MESFET cell using both typical and slow parameters are shown in figure 6.4. The noise margins were obtained superimposing the simulated transfer curves during the read operation . The *maximum square* noise margin definition [19] was used.

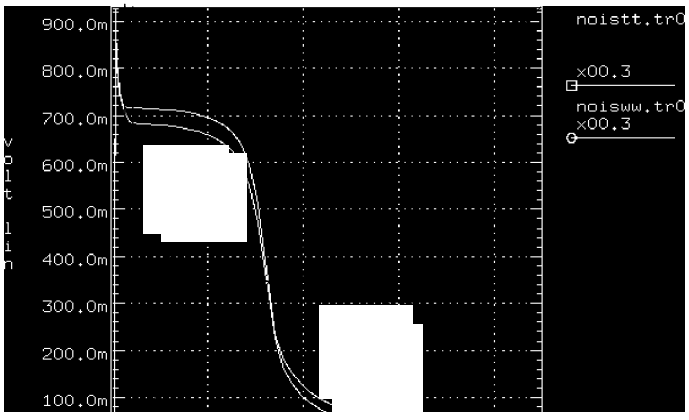


Figure 6.4. Noise margin.

## 6.3. Basic circuit

To analyse the stability of the memory cell HSPICE simulations were carried out. The circuit includes a 32-word x 32-bits memory array, the bit line precharge scheme, I/O circuitry and the sense amplifier. Figure 6.5 shows the 1 kbit RAM block diagram.

The delay time from the address input to the word line is called word-line selection time [20] and is responsible for a large part of the access time. In order to reduce this selection time the following method was applied for the row selection circuit.

A 1 kbit memory array is divided into four 8 x 32 blocks and the address signals are categorised into two groups. The first group ($S_4$, $S_3$), is used for block selection, while the second group ($S_2$, $S_1$, $S_0$) is used for row selection. A hierarchical block decoding method [21] uses power rail logic [22] decoders in order to reduce their power dissipation; when one block has been selected, the remaining three row decoders are disactivated because their power rail control lines are brought down to ground, forcing their unused outputs low.

It is an important technological requirement to reduce the word line RC delay and the array current for preventing the lowering of the high level [23]. Using the above method, a significant reduction in both delay time and power consumption is achieved. As the temperature increases, the high level decreases by the parasitic Schottky diodes in the decoder circuit. The operational margin for the temperature is also improved by this power rail decoding method. To reduce the transient time of the data line signal in *read* operation, column sense amplifiers were used in each column.

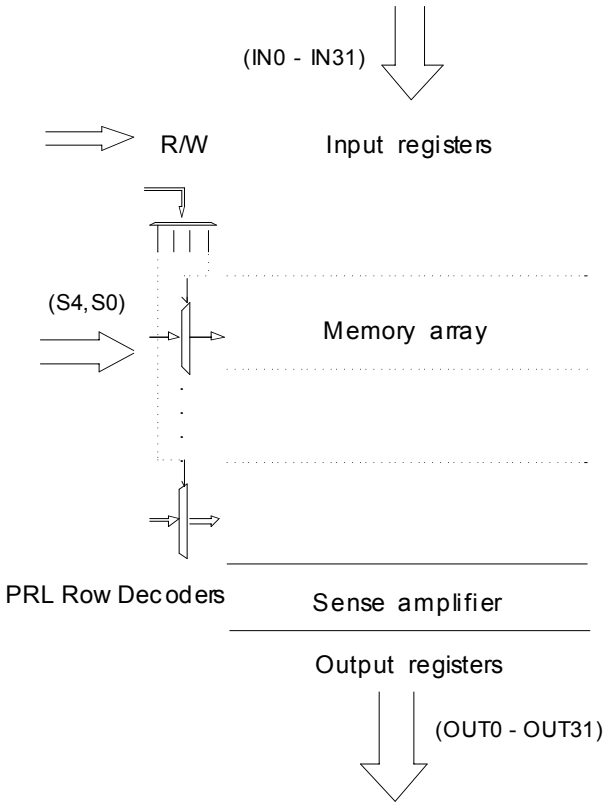
The output stage consists of a register that regenerates and stores the output sense amplifier voltage levels, providing a good fanout and noise margin characteristics. The register limit the output high voltage at 0.7 V to satisfy the input voltage requirements of the driven circuitry.

## 6.4. Sense amplifier

A PRL [22] sense amplifier to achieve lower consumption during no reading operation is proposed. The sense amplifier shown in figure 6.6, consists of a SBFL inverter and two cross-coupled PRL NOR gates. When a read operation is started, the read signal is buffered through the SBFL inverter supplying the power rail of NOR SR latch.

The two cross-coupled transistors *M4* and *M5* avoid charge leakage on the uncharged internal node. This scheme provide a positive feedback which allow to switch rapidly when a small voltage differences are sensed between the output nodes.

Figure 6.5. Block Diagram**.**




Direct and complementary bit read line signals are connected to *M6* and *M3* MESFET's respectively. Since only a single read bit line is required for each memory cell, an inverter is used with the PRL NOR cross-coupled amplifier to generate the complementary signal for the sense operation. The global access time becomes dependent on the threshold level of that inverter. This is the weak point of the sense amplifier configuration used.

The internal voltage levels are then buffered by the push-pull output driver of the sense amplifier offering a suitable fanout capability and furnishing the appropriated voltage levels. All of the logic functions were designed in the Vitesse semiconductor 0.6 $\mu$m process. Representative HSPICE MESFET model parameters are listed in table I.

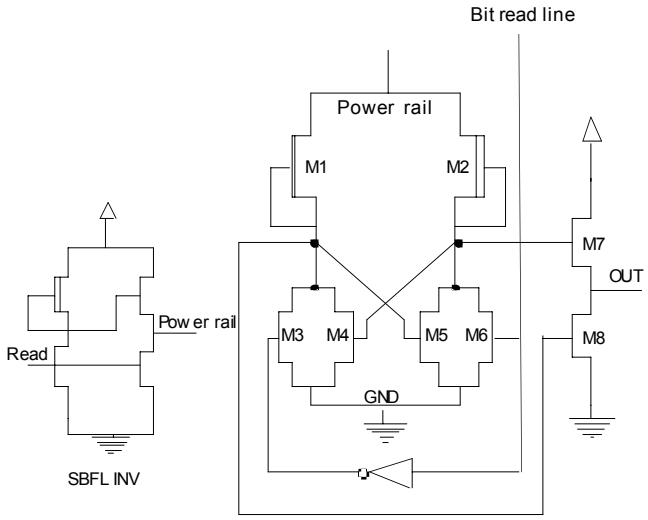



Figure 6.6. PRL sense amplifier.

Table I. MESFET Model Parameters

| Param. | Units | E-JFET | D-JFET |
|---|---|---|---|
| $V_{T0}$ | V | 0.24 | -0.80 |
| Beta | $mA/V^2$ | 2.91 | 2.32 |
| Alfa | Ω-mm | 6.53 | 3.5 |
| Lambda | $V^{-1}$ | 0.072 | 0.050 |
| $G_{DS}$ | ms/mm | 14.5 | 27.0 |
| $R_S$ | Ω-mm | 0.83 | 0.59 |
| $R_D$ | Ω-mm | 0.83 | 0.59 |

## 6.5. Simulation results.

The cell area is 36 x 37 $\mu m^2$ using eight transistors. In figure 6.7, 1 kbit chip layout used for postsimulation analysis is presented. From HSPICE simulation results the total cell read/write access times were found to be 760 ps and 150 ps, respectively. An active current of 20-$\mu$A (at 1Ghz) was obtained. Using this memory cell, the memory array can accommodate 32 cells in a single column. Simulations were done considering arrays with only 32 cells per row and 32 cell per column. The column circuitry of this SRAM include input/output registers and sense amplifiers.

A global write and read access time of 1 ns was measured from the input to the output buffers. The single-ended mode read operation cause that the read access time be longer than the write access time because of the regeneration process necessary to magnify the small bit line voltage difference to full voltage swing. However significant reduction in a global access time has been observed.

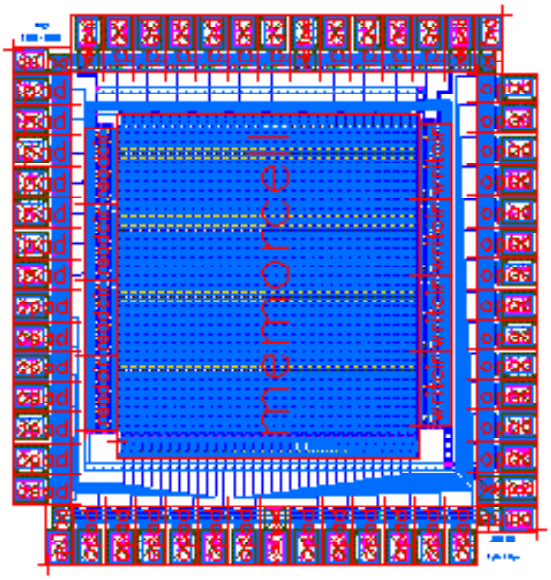


Figure 6.7. 1 kbit layout.

A summary of the memory cell performance (from simulation results) is given in table II. In table III, a comparison between the new cell and some of the reported cells is presented.

Table II. Memory cell performance

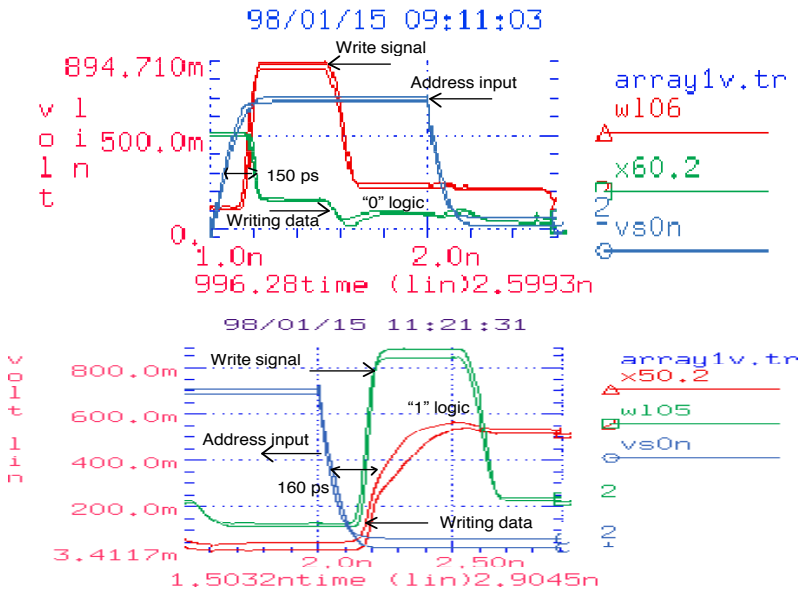
| | |
|---|---|
| Technology | 0.6 $\mu$m - GaAs |
| Chip Organisation | 32-word x 32-bit, 2-port |
| Memory Cell Size | 35.9 x 36.9 $\mu m^2$ |
| Access Time | 1 ns |
| Min. Write Pulse | 150 ps |
| Power Supply | 1 v |
| Core Power Dissipation | 20.5mW |
| Cell Current | 20 $\mu$A |
| Read Time | 760 ps |

Table III. Memory Cells Comparison

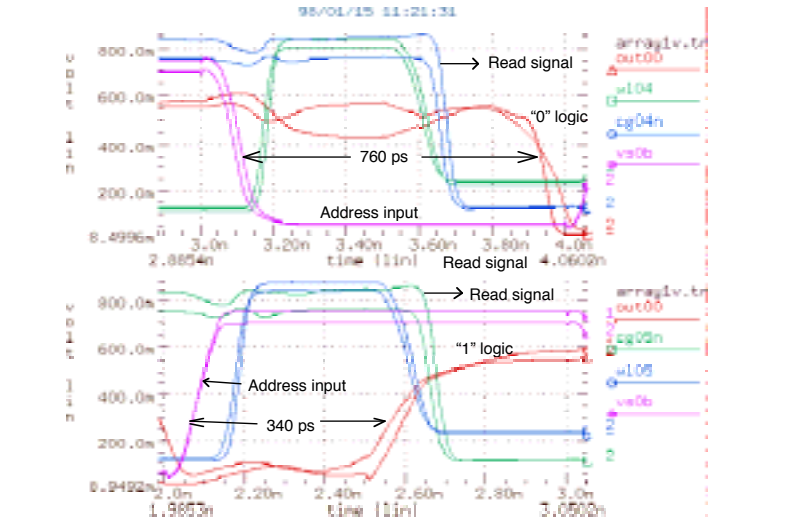
| Cell | SRAM | Access | Tech. | Power | Size |
|---|---|---|---|---|---|
| Tse [2] | 16 KB | 7.5 ns | ------ | 1 w | 33x34 |
| Cha [4] | 1 KB | 2.3 ns | 0.6 $\mu$m | 0.8 w | 18x20 |
| Fie [5] | 4 KB | 3.6ns | 1.0 $\mu$m | 1.9 w | ------ |
| Mat [6] | 16 KB | 5 ns | 0.7 $\mu$m | 2 w | 36x23 |
| Mat [7] | 16 KB | 7 ns | 0.7 $\mu$m | 2.1 w | 36x23 |
| Mak [11] | 4 KB | 7 ns | 1.0 $\mu$m | 0.85 w | 35x29 |
| Law [17] | 1 KB | 2.5 ns | 1.0 $\mu$m | 0.5 w | 26x31 |
| Ber [24] | 1 KB | 1ns | 0.6 $\mu$m | 0.15 w | 36x37 |

High speed and stable operation was accomplished for a temperature range between 5 and 70°C when operated at 1V. In figure 6.8, the address input and data output wave form for a write (a) and read (b) cycle of the memory cell are shown.

Figure 6.9 shows the address input and data output wave form for a read and write cycle considering parametric variations (worst case parameters) and operating at 1V. A range the temperature between 20 - 60°C was also considered.




(a)




(b)

Figure. 6.8. Write (a) and Read (b) operations - Wave forms.

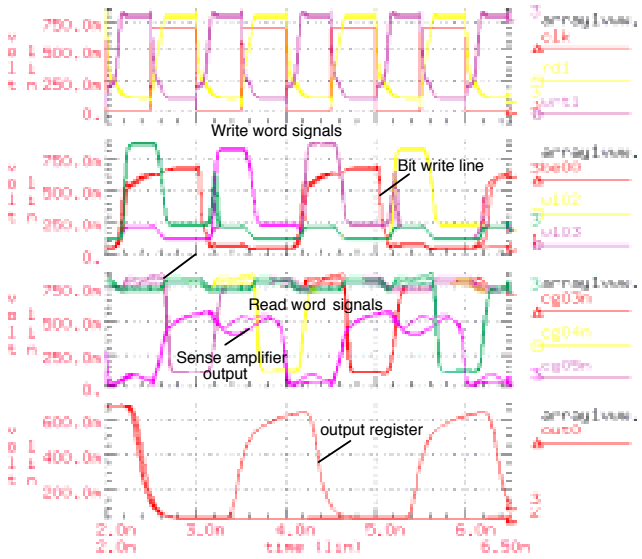



Figure. 6.9. Fully pipelined read/write timing diagram using slow parameters.

### 6.5.1. Worst case

For all the non selected cells, the Read word line voltage is set to 1V with an exception of one active cell for which the voltage on the Read line should be zero. If the voltage in all the internal nodes of the cells ($Q_0$) is high i.e. $VQ_0 = 0.7V$, then all the M6 transistors in the non selected cells (31 in total) are operating in the inversion regime with their source node connected to the bit Read line. They are working as Source followers while the M6 transistors of the active cell is operating in normal mode and should discharge the bit Read line. This operating condition correspond to a worst case. In figure 6.10, simulation results show the read/write timing diagram of the SRAM when fully pipelined [25][26] considering worst case mode operation at 1V. The dependence of the address access time with the temperature is also shown, a range of temperature between 5 - 70° was considered. Write and read consecutive operations for cells attached to two differents column are shown.

## 6.6. Experimental results.

To demonstrate the performance of the cell a 8-words x 4-bits prototype was fabricated using Vitesse III - GaAs technology. A die photo of this experimental circuit is shown in Figure 6.11. The layout of prototype, including bonding pads, occupies an area of 1.15 $mm^2$. The test chip was tested at a power supply voltage of 1V and 2V.

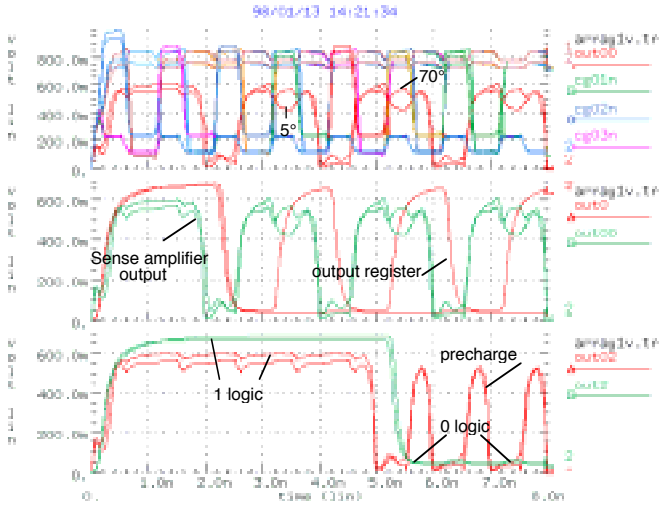



Figure. 6.10. Worst case operating conditions.

First, a simple functional tests at different frequencies were done. A GENRAD LV500 test equipment was used. Figure 6.12, is an oscillograph screen illustrating the functional testing results as well as some internal waveforms using a supply voltage of 1V.

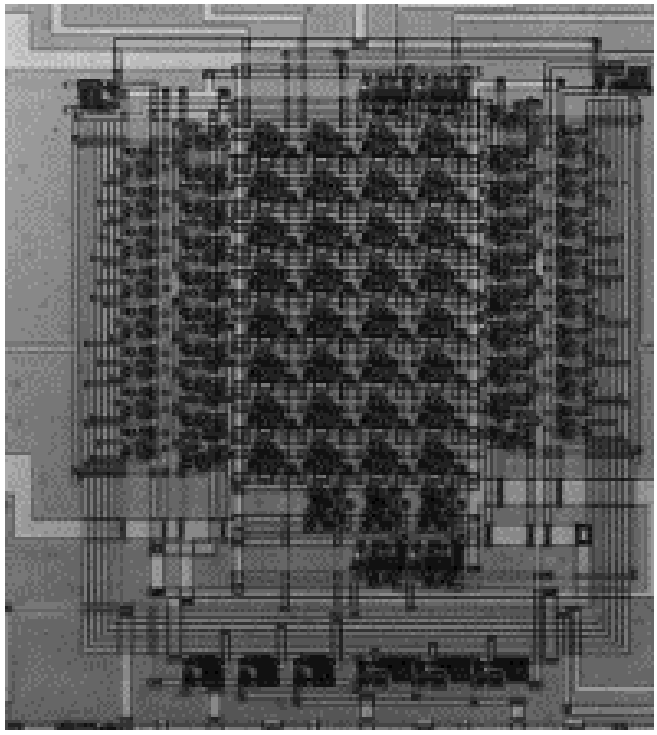


Figure 6.11. Test chip microphoto [27].

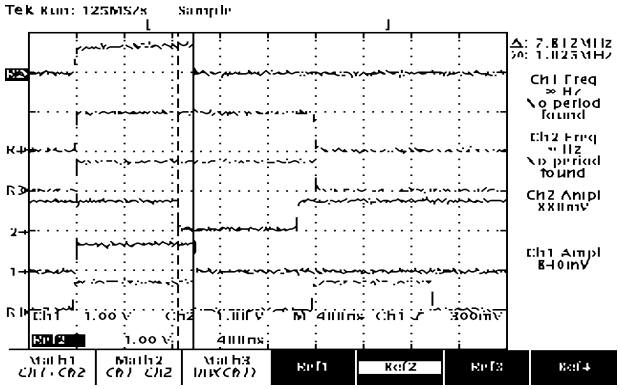



Figure 6.12. Functional testing results.

The test chip was designed using two separate supplies for the cell array core and the control part. Thus, the core was found to be operational over a range of power supply voltages of 1V and 2V. Similarly, the cell was found to operate properly for sense amplifier supplies ranging from 1V to 2V. Five prototypes were tested. The current consumption per cell can be inferred, obtaining 14 $\mu$A/cell at 1V. This result is 30% lower than results obtained through simulation. In table IV, the standby current consumption of the core is shown for supply voltages of 1V and 2V.

Table IV. Core current consumption [mA].

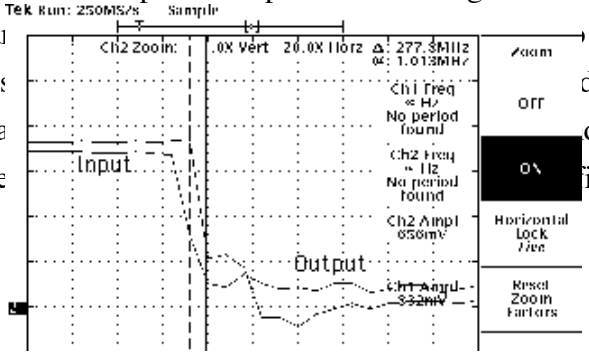
| Supply | Chip1 | Chip2 | Chip3 | Chip4 | Chip5 |
|---|---|---|---|---|---|
| 1V | 0.27 | 0.32 | 0.16 | 0.33 | 0.15 |
| 2V | 0.38 | 0.48 | 0.48 | 0.49 | 0.36 |

In table V, the current consumption of the control part is shown. This current includes the clock and output drivers, the sense amplifiers, I/O registers and the pads. As can be seen the power saving in the control part is not much more significant, using another technique for addressing and decoding recently published [28], more significant power consumption reduction could be achieved.

Table V. Control part current consumption [mA].

| Supply | Chip1 | Chip2 | Chip3 | Chip4 | Chip5 |
|---|---|---|---|---|---|
| 1V | 25.7 | 25.4 | 26.4 | 26.2 | 24.8 |
| 2V | 40.2 | 38.5 | 41.6 | 41.0 | 39.7 |

Due to the test equipment features, the read and write signals could not be synchronised with the clock signal causing the additional delays. Figures 6.13 (a) and 6.13 (b) show the time scale of transients produced in the sense amplifier outputs for each logic level in reading operations. As can be seen, the sense a the positive feedback provided by the two cross delay time elapsed in inverting bit read line signa dent on the threshold level of that inverter, this be figuration.



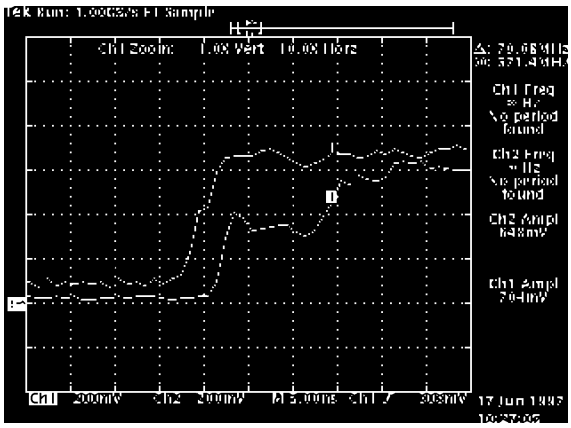
(a)

(b)

Figure 6.13. Sense amplifier outputs.

## 6.7. Conclusions.

A novel low power memory cell structure [29] has been developed to implement static RAM in GaAs technology. The new cell present low power dissipation and high operating speed. The RAM was designed and a test chip fabricated using Vitesse III - GaAs technology.



With the improvement of the structure an address access time of 1ns with a cell power dissipation of 14 $\mu$A/cell has been obtained. The RAM operates at only supply voltage of 1V up to 2V. This RAM can be easily used in implementing high-speed cache memory systems with sub 2 ns on-line memories requirements.

## 6.8. References


[1] W. White, A. Taddiken, H. Shichijo, M. Vernon, Integration of GaAs 4 Kbit Memory with 750-Gate Logic for Digital RF Memory Applications, Proc. 11th GaAs IC Symposium, San Diego California, pp. 37- 40, 1989.

[2] T. Tsen, J. Kwiat, E. Walton, K. Elliot, S. Tiku, A. Cappon, A Low Power 16K GaAs HMESFET Static RAM with Built-in Redundancy, Proc. 12th IEEE GaAs IC Symposium, pp. 155-157, New Orleans, Louissiana, October, 1990.

[3] A. Fiedler, J. Chun, D. Kang, A 3-ns 1K x 4 Static Self-timed GaAs RAM, Proc. 10th IEEE GaAs IC Symposium, Nashville, Tenneessee, pp 67-70, November, 1988.

[4] H. Nakano, M. Noda, M. Sakai, S; Matsue, T. Oku, K. Sumitani, H. Makino, H. Takano, K. Hishitani, A High-Speed GaAs 16Kb SRAM OF 4.4ns/2W using Triple-level metal Interconnection, Proc. 12th GaAs IC Symposium, New Orleans, Louisiana, USA, pp. 151- 154,, 1990.

[5] Y. Kaneko, H. Shimizu, K. Nagata, M. Koyanagi, M; Okamoto, M. Suzuki, S; Yokokawa, S. Shimizu, T. Maejima, J. Wada, H. Kawada, S. Ueno, M. Minamizawa, I. Yaegashi, A 25k-Gate BDCFL G/A with a Differential Push-Pull ECL I/O, Proc. 15th GaAs Symposium, San jose California, pp. 141- 144, 1993.

[6] M. Suzuki, S. Notomi, M. Ono, N. Kobayashi, E. Mitani, K. Odani, T. Mimura, M. Abe, A 1.2-ns HEMT 64-kb SRAM, IEEE Journal of Solid State Circuits , Vol. 26, No. 11, pp. 1571-1576, November, 1991.

[7] S. Matsue, H. Makino, M. Noda, H. Nakano, S. Takano, K. Nishitani, S. Kayano, A 5-ns GaAs 16-kb SRAM, IEEE Journal of Solid State Circuits , Vol. 26, No. 10, pp. 1399-1406, October, 1991.

[8] S. Matsue, H. Makino, M. Noda, N. Tanino, S. Takano, K. Nishitani, S. Kayano, A Soft Error Improved 7ns/2.1W GaAS 16 KB RAM, Proc. 11th IEEE GaAs IC Symposium, pp. 41-44, San Diego, California,1989.

[9] T.R. Weatherford, J.R. Hauser, S.E. Diehl, Comparisons of Single Event Vulnerability of GaAs SRAMS, IEEE Transactions on Nuclear Science. Vol. NS-33, No.6, pp. 1590- 1596, December, 1986.

[10] H. Makino, S. Matsue, M. Noda, N. Tanino, S. Takano, K. Nishitani, S. Kayano, "A 7-ns/850-mW GaAs 4-kb SRAM with Little Dependence on Temperature", IEEE Journal of Solid State Circuits , Vol. 25, No. 5, pp. 1232-1238, 0ctober, 1990.

[11] A. Chandna, R. Brown, An Asynchronous GaAs MESFET Static RAM Using a New Current Mirror Cell, IEEE Journal of Solid State Circuits , Vol. 29, No. 10, pp. 1270-1276, October, 1994.

[12] O.M.K. Law, C.A.T. Salama, GaAs Schmith Trigger Memory Cell Design, IEEE Journal of Solid State Circuits , Vol. 31, No. 8, pp. 1190-1192, August, 1996.

[13] S. Flannagan, P.H. Pelley, N. Herr, B.E. Engles, T. Feng, S. Nogle, J.W. Eagan, R.J. Dunnigan, L.J. Day, R.I. Kung, A 8-ns CMOS 64Kx4 and 256Kx1 SRAM`s, IEEE Journal of Solid State Circuits , Vol. 25, No. 5, pp. 1049-1055, October 1990.

[14] Ch.Ch. Chao, B. A. Wooley, A 1.3-ns 32-Word x 32-Bit Three-Port BiCMOS Register File, IEEE Journal of Solid State Circuits , Vol. 31, No. 6, pp. 758-765, June, 1996

[15] A. Bernal, A. Guyot, New Two-Single Port GaAs Memory Cell, Proc. 23rd European Solid State Circuits Conference, ESSCIRC'97, pp. 180-183, Southampton, England, September, 1997.

[16] K. Itoh, K. Sasaki, Y. Nakagome, Trends in Low-Power RAM Circuit Technologies, Proc. of the IEEE, Vol. 83, No. 4, pp. 524-543, April, 1995.

[17] O.M.K Law, C.A.T. Salama, GaAs Dynamic Memory Design, IEEE Journal of Solid State Circuits , Vol. 31, No. 8, pp. 1193-1196, August, 1996.

[18] E. Busheri, V. Bratov, A. Thiede, V. Staroselsky, D. Clark, Design and Analysis of a Low Power HEMT SRAM Cell, Electronic Letters, 31, (21), pp. 1828-1829, 1995.

[19] S. Long, M. Sundaram, "Noise-Margin Limitations on Gallium-Arsenide VLSI", IEEE Journal of Solid-State Circuits.Vol.23, No.4, pp.893-900, August, 1988.

[20] A Fiedler, D. Kang, A GaAs Pin-for-Pin Compatible Replacement for the ECL 100474 4K SRAM, Proc. 12th IEEE GaAs IC Symposium, New Orleans, Louissiana, pp. 147-150, October, 1990.

[21] T. Hirose, H. Kuriyama, S. Murakami, K. Yuzuriha, T. Mukai, K. Tsutsumi, Y. Nishimura, Y. Kohno, K. Anami, A 20-ns 4 mb CMOS SRAM with Hierarchical Word Decoding Architecture, IEEE Journal of Solid-State Circuits. Vol.25, No.5, pp. 1068- 1073, October, 1990.

[22] A. Chandna, R. Brown, D. Putti, C.D. Kibler, "Power Rail Logic: A Low Power Logic Style for Digital GaAs Circuits", IEEE Journal of Solid-State Circuits. Vol.30, No.10, pp. 1096- 1100, October, 1995.

[23] H. Kawasaki, S.I. Long, A Low-Power 128 x 1-bit GaAs FIFO for ATM Packet Switcher, IEEE Journal of Solid State Circuits, Vol. 31, No. 10, pp. 1547-1555,, October, 1996.

[24] A. Bernal, R. Ribas, A. Guyot, GaAs MESFET SRAM using a New High Speed Memory Cell, Proc. 5th European Gallium Arsenide and Related III-V Compounds Applications Symposium, Bologna, Italy, pp. 255- 258, September, 1997.

[25] T. Chappell, B; Chappell, S. Schuster, J. Allan, S. Klepner, R. Joshi, R. Franch, A 2-ns cycle, 3.8-ns Access 512-kb CMOS ECL SRAM with a Fully Pipelined Architecture, IEEE Journal of Solid-State Circuits. Vol.26, No.11, pp. 1577- 1585, November, 1991.

[26] D. Koe, C. Salama, Pipelining of GaAs Dynamic Logic Circuits, Proc. IEEE International Symposium on Circuits and Systems, San Diego, California, USA, , pp. 255- 258, May, 1992.

[27] A. Bernal, A. Guyot, New High Speed GaAs Memory Cell, Proc. XII Design of Circuits and Integrated Systems Conference, Sevilla, Spain, pp. 441- 446, November, 1997.

[28] J.F. Lopez, K. Esrhraghian, R. Sarmiento, A. Nuñez, D. Abbot, "GaAs Pseudo-Dynamic Latched Logic for High Performance Proccesor Cores", IEEE Journal of Solid-State Circuits.Vol.32, No.8, pp.1096-1100, Aug., 1997.

[29] A. Bernal, A. Guyot, A New Low-Power GaAs Two-Single Port Memory Cell, IEEE Journal of Solid-State Circuits.Vol.33, No.7, pp.1096-1100, July., 1998.

# 7. A Low-Power GaAs Asynchronous Logic.

## 7.1. Introduction

As it has been presented in the previous chapters, the full adder cell is one of the functions that is used with greater recurrence in the architecture of the exponentiation system. It is possible to see it both in the CSA or CPA adjustment. Similarly this operator is used into the control part of the multiplier as well as in the one of the global system. It is also used in the implementation of the substractor. For these reasons, strategies for reducing the power consumption in the full adders implementation using asynchronous topologies were researched. In this chapter we will analize two differents approachs.

In digital circuits, GaAs has been investigated since 70's, but because of the high static consumption, it has not reached similar CMOS widespread use. GaAs synchronous digital circuit associated worrisomes, such as static power dissipation, clock skew and signal synchronization are the greatest barriers to overcome in the face of increasing both operation speed and design complexity.

Nevertheless, being particularly fast, GaAs technology is becoming good candidate for global clock free design (asynchronous design). Due to, asynchronous circuits do not require a clock to govern the timing of state changes, not having skew and signal synchronization problems [1], the use of self-timed differential structures has been proposed as a way to avoid the discussed troubles [2][3].

Currently, GaAs asynchronous design starts to be considered, due to it is possible to avoid high clock distribution troubles and static currents in unused parts of the circuit [4][5][6]. Of this form, asynchronous approach avoids the precharge and discharge of parasitic capacitances in

portions of a unused circuit during the current computation, reducing the global power consumption.

For that application, several differential structures [7][8][9][10], using dual-rail output signals to detect operation completion, have been proposed. These structures conceived in order to increase the gate complexity and reduce consequently, the power dissipation per logic function represent an efficient strategy to build asynchronous circuits.

### 7.2. Asyncrhonous design

The underlying principle of asynchronous circuits consists in detecting logic evaluation detection through acknowledge signals and use that signal to trigger through a request signal evaluation below. The decision of which and when function blocks must operate is taken by an asynchronous interfece device, commonly called *handshake* circuit.

The use of handshake circuit in asynchronous design makes it run as fast as possible, giving automatic adaptation to physical properties and easier design migration due to the less timing considerations [4].

Using that principle of operation is possible to achieve lower levels of power consumption in high-speed ICs design because of asynchronous approach avoids the precharge and discharge of parasitic capacitances in portions of a unused circuit during the current computation; another asynchronous advantage is the average case instead of the worst-case performance, bacause it senses when a computation has been completed while synchronous ones must wait until all possible have completed before latching the results; asynchronous circuits automatically adapt to variations on fabrication, temperature and power-supply voltage; Subsystems can be easily substituted into the asynchronous systems. It would allow increase the global performance of the circuit since other internal circuits or structures would not be affected.

Several techniques have been proposed to generate the acknowledge signals: single rail and dual rail techniques will be briefly reviewed.

#### 7.2.1. Single-rail techniques.

*Sutherland* [11], in its paper presents the technique called *Micropipeline,* which implement delay elements that represent the worst-case propagation delays of function bloks and generate the acknowledge signal. For that, it must be guaranted that only the outputs are ready before the transition of the delay element output; this method is technology independent and specifically in GaAs MESFET micropipelines approach enhances the circuit performance and can be used to

reduce the power consumption cutting-off temporarily the power-supply to portions of the circuit in stand-by.

Asynchronous circuit can be also implemented building functions with ternary logic [12]. Ternary logic gates must take into account three distinct input/output voltage levels corresponding to the values true ('1'), undefined ('u') and false ('0'). Stand-by state and not ready signals are identified by the 'u' condition.

CSCD ciruits ("*Current Sensing Completion Detection*") [13], monitors dynamic currents during logic transition (evaluation) to generate the acknowledge signal when the operation is finished, it means, when no dynamic currents are detected; CSCD GaAs MESFET implementation are unfeasible because of the static currents discussed above.

### 7.2.2. Dual rail technique.

In dual rail signalling, every Boolean variable is encoded onto two wires, called *an encoding pair*. This encoding provides a means for logic functions completion detection as well as a valid data transmission by observing the two wires ($S_t$,$S_f$). The two wires encode ternary values, so let ($S_t$,$S_f$) be the pair of wires:

- Precharge phase or not ready. - ($S_t$,$S_f$) = (0,0);
- Ready data '1' - ($S_t$,$S_f$) = (1,0);
- Ready data '0' - ($S_t$,$S_f$) = (0,1);
- Never used - ($S_t$,$S_f$) = (1,1)

Valid data transmissions are always separated by the intermediate state ($S_t$,$S_f$) = (0,0) to stablish standby state (precharge phase) before the next evaluation in function blocks making possible the completion detection.

The handshake circuit [14] must prevent "runaway" conditions, it means, data overwritten at the input to next block, if that block has a long computation latency. It must also prevent "continual feeding", this is, data computed more than once by next block if previous block has a long latency. So, an acknowledge signal is necessary to indicate when a next block has completed its task and is ready for the next. Handshake circuit must also guarante that the next block is in stand-by condition while the previous block execute the computation.

Dual rail function blocks can be easily built with CMOS standard logic gates by implementing dual-logic cells (direct and complementary logic implementations). In GaAs circuits, the absence of P-type transistors and the difficulty to build NAND gates in the standard DCFL MESFET logic family lead to the dual-logic NOR-NOR PLA configuration. However, differential

structures are widely used not only in the asynchronous CMOS ICs design but also in GaAs technology increasing the speed of operation with similar levels of power consumption [8].

In next section, we analyse from experimental results two asynchronous structures to implement low power self-timed circuits. The first one, a DC$^2$FL structure which was conceived keeping in mind mentioned properties, but at the expense of speed performance and the second one, the Enable/Disable MESFET Differential Logic (EMDL) which is also presented as a solution to implement dual-rail synchronous and asynchronous circuits, with no power consumption in standby state and keeping speed performance comparable to DCFL.

### 7.3. A Low-Power Differential Cross-Coupled FET Logic.

A DC$^2$FL low-power differential MESFET logic structure mixes several features of previous techniques, achieving lower levels of power-delay product than DCFL gates and as low as DCVS ones. Additionally, it can be also used to reduce the static power consumption of asynchronous circuits during the standby state, showing further significant advantages when applied to built self-timed rings [15].
A fully functional 8-bit ripple carry adder was designed by using Vitesse GaAs III technology [16] at 2V power-supply voltage. The structure is totally compatible with DCFL, DPTL and DCVS topologies.

#### 7.3.1. Basic Structure

Unlike the most of previous precharged differential structures presented [8][9][10], DC$^2$FL is based on predischarging internal nodes. The basic structure is illustrated in figure 7.1, the direct and complementary logic function branches are involved in the EFET logic tree, providing two internal outputs *a* and *b* which are required to delay-insensitive asynchronous applications. When *fi* is high (*fin* is low) the predischarge phase is started and both internal nodes are discharged through M2 and M3 transistors, while M1 isolates the logic tree from the supply voltage, so no significant currents flow into the EFET logic tree.

Then, during the evaluation phase (*fi* is low, *fin* is high), either *a* or *b* node is charged up to 0.6V according to the logic tree function and the input variables. The cross-coupled transistors M4 and M5 connected to such nodes, provide a positive feedback avoiding charge leakage on the uncharged internal node, during the evaluation phase, switching fastly when small voltage differences are sensed. The internal voltage levels are after buffered by the output stage, offering a suitable fanout capacity.

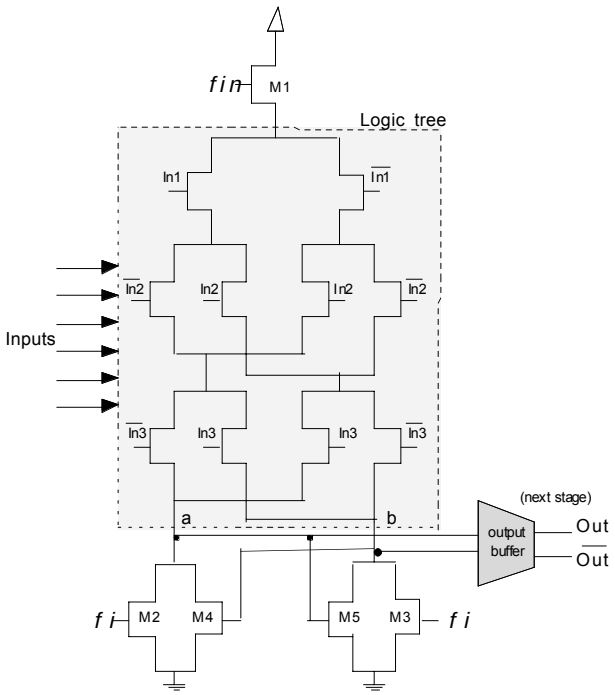



Figure 7.1. - DC$^2$FL structure.

The two predischarge transistors M2 and M3, like as the cross-coupled ones M4 and M5 can be made as small as possible due their size do not affect the global performance. M1 transistor is sized similar to the logic tree transistors because the number of EFET devices involved in the logic part define the current used to charge the respective internal node.

So, as can be seen in figure 7.2, the logic tree transistors size (W) is a compromise between the charging delay (td) responsible for the speed performance, and the current flow through the tree (I), that contributes to the power dissipation. This analysis was realized over a inverter and full adder (Fig. 7.1), but it can be extended to more complex gates.

A constraint of this approach, like as DPTL and DCVS techniques, consists in restricting the high input voltages (Vin < 0.7V) to prevent the forward biasing of the MESFET Schottky barrier in DC$^2$FL tree transistors. Design considerations about the output stage are discussed in next section.

### 7.3.2. Output Stage

The output stage consists of a buffer that regenerates the internal voltage levels, providing a good fanout and noise margin characteristics. Moreover, it must to be no inverting circuit, because these outputs can be connected to either a similar next stage or a DCVS one [10] and they have to be low during the predischarge phase for the proper operation of these circuits. Furthermore, this stage must limit the output high voltage at 0.7V to satisfy the input voltage requirements of the next stage. The evident approach of two cascating DCFL inverters for each output is unsuitable due to the significant power dissipation of this approach.

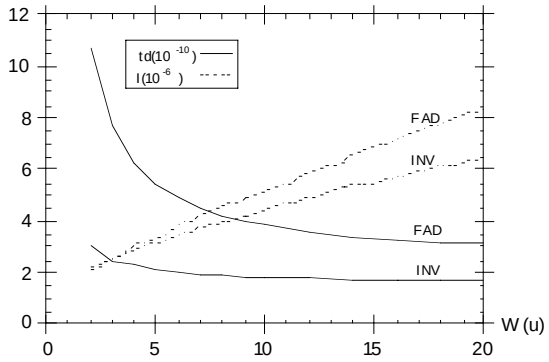



Fig. 7.2. - Charging delay (td) and current consumption (I) vs tree transistor width.

A new and particular output stage to achieve lower consumption during standby state is proposed. The circuit is shown in figure 7.3, and consists of a pullup NOR gate configuration, a SBFL inverter and a NOR SR latch implemented with Power Rail technique [17].

The NOR gate detects when either *a* or *b* node is charged above to approximately 0.2V, and generates a signal which buffered through the SBFL inverter supplies the SR latch. The latch senses small voltage differences in the nodes *a* and *b*, like as DPTL buffer [7], switching fastly and furnishing the appropriated voltage levels 0.1V and 0.7V.

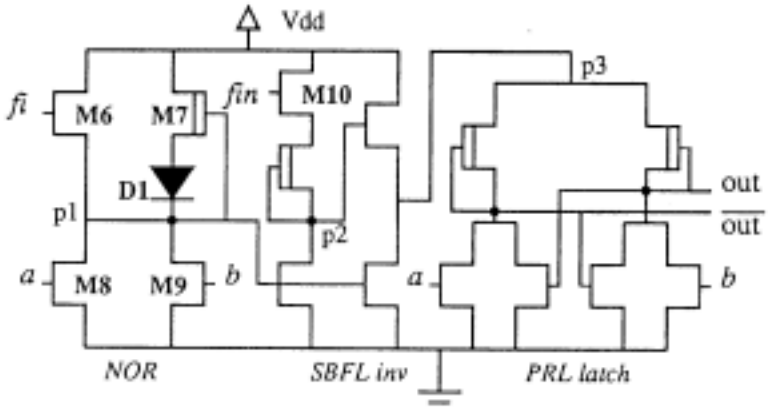



Fig. 7.3. - Output stage schematic.

The pullup NOR output is raised up at 0.6V by *fi* signal during the predischarge phase. The logic tree input variables could not be available or even *a* or *b* nodes not be properly charged when the evaluation phase is started. It would discharge the output NOR leading a bad functioning of the circuit. Then, a weak pullup current source (M7 and D1) is used to compensate the subthreshold currents through M8 and M9 transistors. Such configuration keeps NOR output voltage at approximately 0.6V.

This weak current source must provide an equivalent current to M8 and M9 subthreshold currents ($I_{sub}$) plus SBFL inverter gate currents ($I_{Sh}$). Both currents depend on the transistor sizes and their biasing voltages. So, $I_{M7\text{-}D1} \geq 2.I_{sub} + 2.I_{Sh}$ condition must be satisfied. However, with NOR output at 0.6V, $ISh$ is much lower than $I_{sub}$ and can be neglected. Furthermore, M7 device is always operating in saturation region, then [18]:

$$ß_7(V_{gs}-V_T)^2 tanh(\alpha V_{ds}) \geq C_1.W_8.no(1- e^{-U_{ds}})/L_8 \quad (1)$$

where

*no* - equilibrium minority carrier concentration;

$C_1 = 2.L_B.q.D_n$

$L_B$ is the extrinsic Debye length and

$D_n$ is the diffusion constant.

Considering $W_8=2\mu$ and $L_8=1\mu$, the M7 device size $W_7=4\mu$ and $L_7=2\mu$ is obtained. The dimensions $W_1=2\mu$ and $L_1=2\mu$ of D1 device were found by using the expression [19]:

$$I_{Sh} = W_1.L_n1(C_2.e^{-U_{ds}}) e^{U} \quad (2)$$

Where:

$C_2 = q.N_D.V$

$N_D$ is the dopant concentration and

$V$ depends both drift and diffusion velocities.

The voltages variables $\{U,U_{ds}\}$ are normalized by thermal voltage.

Design considerations of the SBFL and Power Rail latch can be found in [20] and [17], respectively. In figure 7.4, the dc transfer curve of a DC$^2$FL inverter using such output stage is shown.

In the case of latched gates, used frequently to implement asynchronous pipelines and self-timed rings, SR DCFL NOR latch can appropriately replace the DC$^2$FL output stage acting as a buffer and storage element. In DCVS approach, such substitution is not possible because the correspondant SR NAND latch version is difficult to be built with DCFL gates [18].

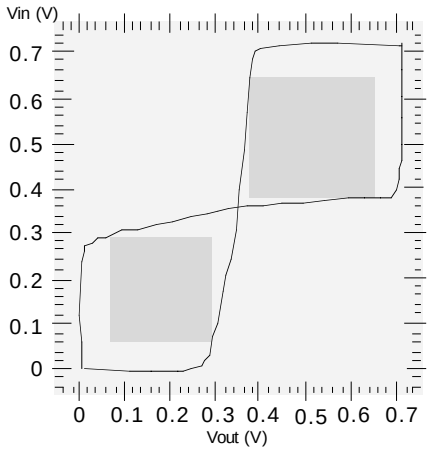



Fig. 7.4. - DC$^2$FL inverter dc transfer curve.

### 7.3.3. Simulation Results

#### 7.3.3.1. Full adder

Full adder circuits were used to evaluate and compare this structure with DCFL, DPTL and DCVS ones. The DC$^2$FL sum circuit is presented in fig. 7.1, HSPICE simulation results at 166 Mhz at a supply voltage of 2V are summarized in table I.

Table I - Full adder simulation results.

| | td (ps) fi-out/in-out | Pot (mW) Prech./eval. | Pot∞td (fJ) | # trans. |
|---|---|---|---|---|
| DCFL | 137.5* | 3.76* | 517.0 | 42 |
| DPTL | 44.3 / - ** | 1.16 / 1.02 | 48.3 | 36 |
| DCVS | 210.4 / 238.8 | 1.15 / 1.02 | 243.7 | 56 |
| DC$^2$FL | 449.4 / 358.5 | 0.47 / 1.05 | 307.0 | 58 |

* In DCFL, there are not *fi* signal and precharge phase.

** In DPTL, the inputs must be available in the evaluation.

The power-delay products (Potxtd) were obtained considering average values. Power dissipation of DPTL, DCVS and DC$^2$FL full adder versions during standby state and evaluation phase are also presented in figure 7.5. Otherwise, latched gates were also taken into account in order to evaluate the differential structure performance, as shown in Table II. For latched DPTL and DCVS gates, it was necessary to add the SR DCFL NOR latches in the outputs to storage

data. A significant power reduction as well as the expected lower power-delay product were verified to DC$^2$FL.

Table II - Latched full adder simulation results.

| | td (ps) fi-out/in-out | Pot (mW) Prech./eval. | Pot∞td (fJ) | # transistor |
|---|---|---|---|---|
| DPTL | 181.6 / --- | 1.96 / 1.83 | 344.1 | 50 |
| DCVS | 317.3 / 344.0 | 2.16 / 2.13 | 709.2 | 70 |
| DC$^2$FL | 622.1 / 584.3 | 1.01 / 1.02 | 363.5 | 42 |

### 7.3.3.2. 8-bit ripple carry adder

A 8-bit ripple carry adder circuit was chosen as a vehicle to demonstrate the DC$^2$FL advantages. The Ripple Carry Adder was chosen to be an architecture-neutral evaluation and the worst possible delay for a 8-bits adder. A test chip containing DC$^2$FL and DCVS versions was designed and fabricated by CMP services using Vitesse GaAs III technology. The chip layout is shown in figure 7.6. Table III presents the HSPICE simulation results.

Table III - 8-bit ripple carry adder simulation results.

| | td (ns) | Pot (mW) | Pot∞td (pJ) | # transistor |
|---|---|---|---|---|
| DCVS | 0.903 | 8.93 | 8.06 | 448 |
| DC$^2$FL | 1.641 | 6.08 | 9.98 | 464 |

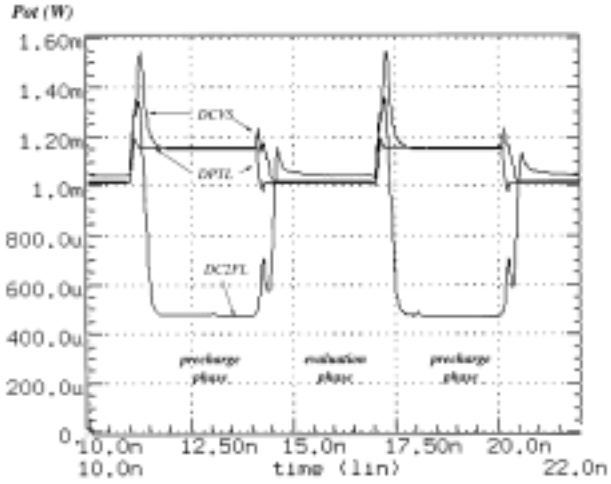



Figure 7.5. Power dissipation graph [21].

The most important feature of DC$^2$FL is its compatibility with DCVS which permit replace some portions of its functional blocks where timing is not critical, saving power on standby state. Such design strategy can be easily extended to synchronous circuits, implementing critical paths of DC$^2$FL circuits using DCVS topology. For example, the same 8-bit ripple carry adder operating in synchronous mode presents approximately 25% power saving in precharge phase

when designed using DC$^2$FL-DCVS mixed circuit, maintaining the same speed performance that DCVS version.

### 7.3.4. Experimental results

To demonstrate the performance of the structure five prototypes were designed and fabricated using Vitesse III GaAs technology. First, an exhaustive functional test was done using a GENRAD LV500 test equipment. The full functionally prototypes were tested at a power supply voltage of 1V, 1,5V, 1,8V and 2V. The exhaustive functional test of each adder was done using the test patterns set shown in table IV. This test pattern set includes all possible required combination to execute the exhaustive functional verification of the adders.

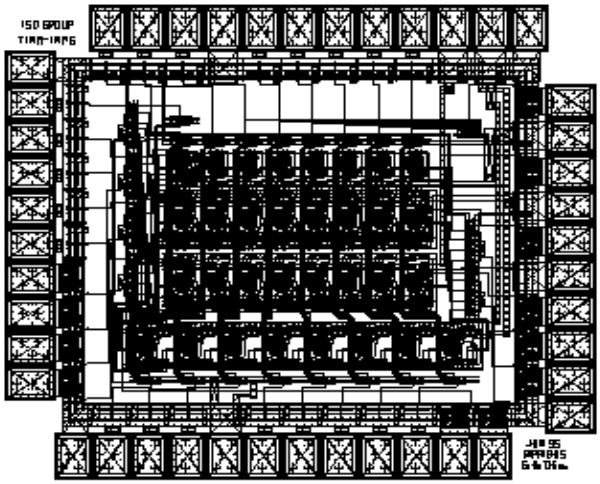


Figure 7.6. - Test chip layout.

The 8-bit adder was implemented in both DCVS and DC$^2$FL structures to compare the power consumption with varing power supply voltage.

Table IV. Test patterns set.

| Hexad. | | Binary Notation | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | B | A7-B7 | A6-B6 | A5-B5 | A4-B4 | A3-B3 | A2-B2 | A1-B1 | A0-B0 | Cin |
| 00 | 00 | 0 - 0 | 0 - 0 | 0 - 0 | 0 -0 | 0 - 0 | 0 - 0 | 0 - 0 | 0 - 0 | 0 |
| aa | aa | 0 - 0 | 1 - 1 | 0 - 0 | 1 - 1 | 0 - 0 | 1 - 1 | 0 - 0 | 1 - 1 | 0 |
| 55 | 55 | 1 - 1 | 0 - 0 | 1 - 1 | 0 - 0 | 1 - 1 | 0 - 0 | 1 - 1 | 0 - 0 | 1 |
| 00 | ff | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 |
| 00 | ff | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 0 - 1 | 1 |
| ff | 00 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 0 |
| ff | 00 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 - 0 | 1 |
| ff | ff | 1 - 1 | 1 - 1 | 1 - 1 | 1 - 1 | 1 - 1 | 1 - 1 | 1 - 1 | 1 - 1 | 1 |

In table V, the current consumption measures of each prototype and of each structure are shown. In figure 7.7, the average power consumption with varing power supply voltage is depicted.

Table V. Current consumption.

| Est.. / Chip | 1.0 V | | 1.5 V | | 1.8 V | | 2.0 V | |
|---|---|---|---|---|---|---|---|---|
| DCVS(mA) | fi= | fi=0 | fi =1 | fi=0 | fi=1 | fi=0 | fi=1 | fi=0 |
| Chip 1 | 1 4.7 | 4.6 | 5.6 | 5.7 | 5.7 | 5.7 | 5.8 | 5.8 |
| Chip 2 | 5.5 | 5.8 | 5.9 | 6.2 | 6.4 | 6.3 | 6.5 | 6.4 |
| Chip 3 | 5.2 | 5.7 | 5.7 | 5.9 | 6.1 | 6.1 | 6.2 | 6.1 |
| Chip 4 | 5.4 | 5.9 | 5.8 | 5.9 | 6.1 | 6.0 | 6.2 | 6.0 |
| Chip 5 | 4.6 | 4.7 | 4.8 | 4.8 | 5.0 | 5.1 | 5.0 | 4.9 |
| DC2FL(mA) | fi= | fi=0 | fi =1 | fi =0 | fi=1 | fi=0 | fi=1 | fi=0 |
| Chip 1 | 1 1.8 | 0.36 | 2.6 | 0.58 | 2.6 | 0.56 | 2.6 | 0.60 |
| Chip 2 | 1.9 | 0.47 | 2.9 | 0.58 | 2.9 | 0.60 | 3.0 | 0.60 |
| Chip 3 | 2.0 | 0.51 | 2.8 | 0.62 | 3.0 | 0.60 | 3.1 | 0.63 |
| Chip 4 | 2.6 | 0.48 | 2.8 | 0.62 | 2.9 | 0.62 | 3.1 | 0.59 |
| Chip 5 | 1.7 | 0.46 | 2.4 | 0.55 | 2.6 | 0.56 | 2.6 | 0.50 |

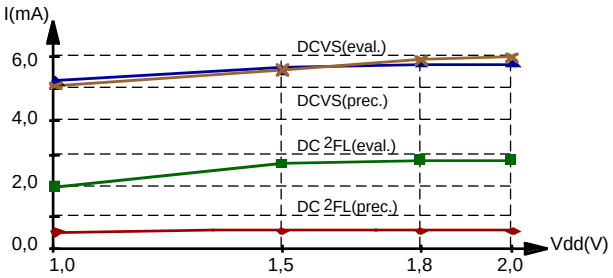
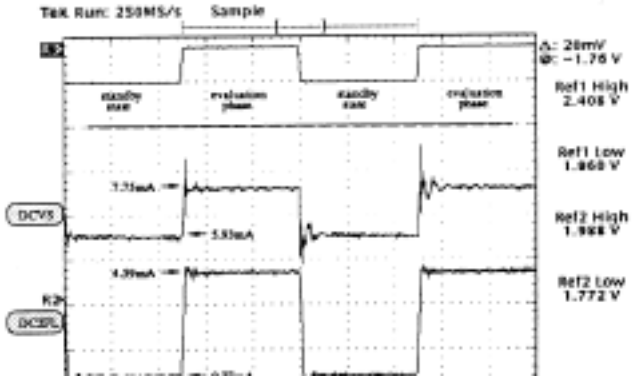



Figure 7.7. Current consumption at different power supply voltage.

The significant power reduction achieved by DC$^2$FL structure during the stand-by state at a power supply voltage of 2V is shown in figure 7.8, which correspond to an oscillograph screen of the equipment test.

The adders were designed to operate with a carrier frequency of 70 MHz. However due to limitations in the test board, reliable measurements were possible only up to 10 MHz. Except for some coupled capacitive effects associated to the input/output pads when high frequencies



were applied, the delay results are predictable.

Figure 7.8. Current consumption waveform.

## 7.4. A Low-Power Enable/Disable GaAs MESFET Differential Logic

In this section, the Enable/Disable MESFET Differential Logic (EMDL) is presented as a solution to implement dual-rail synchronous and asynchronous circuits, with no power consumption in standby state and keeping speed performance comparable to DCFL.

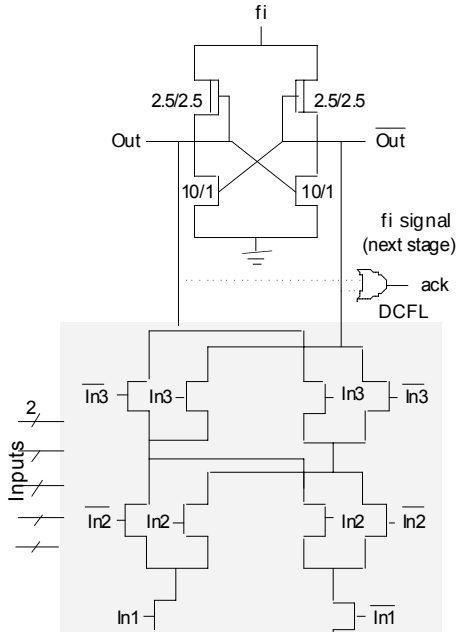
Particularly, iterative networks and micropipelines represent the most attractive architectures to be implemented with EMDL gates, also permitting an easy design migration to or from CMOS ECDL logic without timing constraints. Such an application is exemplified through a fully functional 8-bit ripple carry adder (RCA) fabricated using Vitesse H-GaAs III technology.

### 7.4.1. EMDL Circuit.

The EMDL functionality is similar to CMOS ECDL structure proposed by S. L. Lu [22]. The absence of P-type MESFET is compensated by the Power Rail Logic (PRL) technique, used here to establish the standby and evaluation phases.

#### 7.4.1.1. Basic Operation

The basic structure is shown in figure 7.9, two distinct parts are identified: the logic and load parts. The logic part consists in a E-MESFET logic tree (E-tree) composed by two branches, corresponding to the direct and the complementary logic, necessary to generate the dual-rail outputs. The load part, in turn, is built with two cross-coupled PRL inverters that decide the correct logic evaluation according to the E-tree configuration. The load part also acts as an output buffer. The additional DCFL NOR gate, seen in figure 7.9, is discussed in further



section.

Figure 7.9. Schematic of a EMDL gate.

When the enable signal (*fi*) is brought low to ground, the depletion load transistors in the load part remain on and ensure that the outputs are driven low, independently from the input signal levels. So, the standby state is established and no power-to-ground path is present, avoiding the undesirable static currents. Due to the cross-coupled inverter configuration, when *fien* rises up to approximately 1V, one output is forced to ground while the other goes to 0.7V, according to the E-tree arrangement and the input vector already available.

It is important to observe that the branches in E-tree do not need to discharge internal nodes previously precharged, like in [8][9][10], or vice-versa, that is, to charge such nodes previously predischarged [23]. Small differences in currents are enough to avoid metastable state or mismatches in the load part, during the evaluation. It allows to use ratioless transistors in the E-tree, simplifying the transistor sizing task and providing noise margin immunity.

Moreover, logic trees normally use less transistor than logic networks found in DPTL to build the direct and complementary logic branches, because additional logic is required in order to avoid undefined high-impedance states and guarantee the correct DPTL functionality. Furthermore, because of such additional logic and the source transistor input connections, the fanin in pass-transistor network configurations becomes higher than in logic trees.

Like other MESFET differential structures, the main constraint is the limitation of high input voltage levels at 0.7V to prevent the forward biasing of the MESFET Schottky barrier in the logic part (Vih < 0.7V). Note that, when EMDL stages are cascaded such constraint is respected by the cross-coupled inverter output configuration.

Additionally, as discussed above, the input data must be available before the evaluation phase. Finally, due to a possible performance degradation, the *fi* signal is suggested to be higher than 1V; more details about it are presented bellow.

#### 7.4.1.2. Design Considerations

One of the principal characteristics of EMDL is its straightforward design. The logic part is implemented with well-known logic tree configuration. The load part, in turn, gives the speed and fanout characteristics, since the *fi* signal generator does not limit the charge currents, and the classical equations of the ratioed DCFL could be used to obtain good output voltage levels.

As can be observed in figure 7.10, the *fi* signal high level is a compromise between the delay propagation and the power consumption during the evaluation phase. This enable signal can be driven by a DCFL gate or by any type of super buffer but in a well-controlled manner, i.e., the driver that feeds it cannot also be used to drive other types of logic gates, such as a DCFL or a DPTL one [7]. Figure 7.10, correspond to an EMDL inverter with fanout 1 and 4. The values are normalized (fo=1 / *fi*=1V).

Two procedures to optimize the NMOS tree logic were presented by Chu and Pulfrey [24] and can be directly applied here. Moreover, although the use of ratioless transistors are allowed in the E-tree, minimum transistor dimensions could generate mismatches and might be avoided.

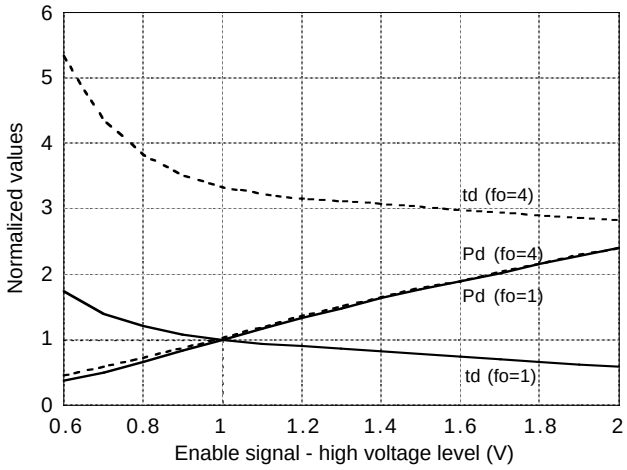



Figure 7.10. Delay and power consumption versus enable voltage level.

When the calculation is started (*fi* goes high) both outputs try to arise until current differences are detected, as illustrated in figure 7.11.

However, depending on the transistor width selected for an E-tree arrangement, parasitic currents in the false path could be momentarily higher than the current flowing to ground, unbalancing the output metastable state in the wrong direction and producing an incorrect evaluation.

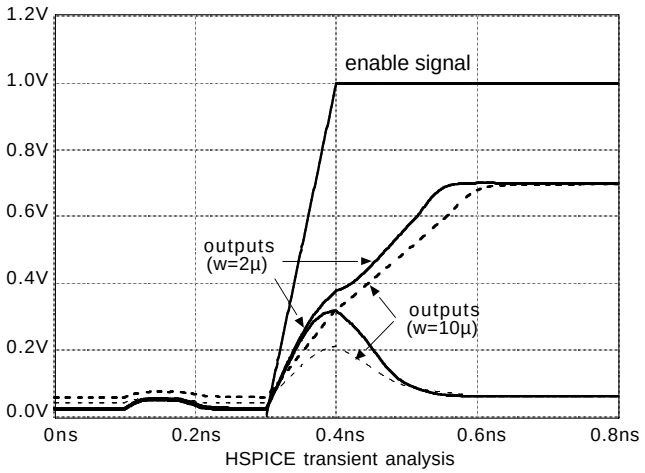



Fig. 7.11. Transient HSPICE simulation: evaluation of the 4-input EMDL NAND gate with 2µm and 10µm E-tree transistor width.

In the MESFET operation, the parasitic source and drain access resistances ($R_s$ and $R_d$) between the source/drain contact and the edge of the gate must be considered; these resistances reduce the externally applied gate-to-source ($V_{gs}$) and drain-to-source ($V_{ds}$) voltages. This situation is described by the equations bellow, where $V_{gsi}$ and Vdsi are the terminal internal voltages [18]:

$$V_{gsi} = V_{gs} - I_{ds}.R_s \quad (3)$$

$$V_{dsi} = V_{ds} - I_{ds} (R_s + R_d) \quad (4)$$

When both outputs try to arise, we consider small values of Vds, and for this reason the E-FETs of the logic tree would operate in linear region, according to ($V_{dsi} < V_{gsi} - V_T$). So, the current through the devices would be:

$$I_{ds} = 2\beta[(V_{gsi}-V_T)V_{dsi}1/2V_{dsi}^2)](1+\lambda V_{dsi})\tanh(\alpha V_{dsi}) \quad (5)$$

The amount of drain-to-source current is determined by its drain-to-source and gate-to-source voltages, a reduction of 9% in the voltage values generates a reduction of 30% in the drain current one. The MESFET drain current also varies with channel width (W), thus it is possible avoid an excessive diminution of the drain current by manipulating the parameter W.

Additionally, the transconductances gm and gds, which relate the increase in Ids to an increase in either $V_{dsi}$ or $V_{gsi}$, are also affected.

$$g_m = [\partial I_d / \partial V_{dsi}] \Big|_{V_{gsi}=cte} \quad (6)$$

$$g_m = [\partial I_d / \partial V_{gsi}] \Big|_{V_{dsi}=cte} \quad (7)$$

The voltage drop across $R_s$ causes a similar effect on the transconductances. Specifically, $g_m$ could be reduced by the small source resistance by almost 30%, reducing the voltage gain and offering smaller current values. The variations of the transconductances according to the drain and source resistances are described by the equations below, where gmi and gdsi are the internal values:

$$g_m = g_{mi} [1 / ( 1 + g_{mi}.R_s )] \quad (8)$$

$$g_{ds} = g_{dsi} [1 / ( 1 + g_{mi}.R_s + g_{dsi}.R_s )] \quad (9)$$

Due to both $g_{ds}$ and $g_m$ are proportional to the transistor width and also are both function of the terminal voltages, an increase of W would lead to keep appropriated voltage gain values [18].

The terminal external voltage reduction is more critical when several series devices are connected. This condition generates a strong reduction of the internal nodes values, obtaining drain currents too close to zero and leading to the mismatch problems discussed. For more than three serie-transistor configurations, a transistor width at least equal to 10µm is suggested in the logic tree.

### 7.4.2. Performance comparison

In order to provide a comparison with other MESFET logic families, a full adder was used as a benchmark circuit. The HSPICE simulation results obtained using Vitesse H-GaAs III typical process parameters, 2V supply voltage and ambient temperature are shown in Table VI. The power-delay products were obtained taking into account the average values.

Note that the use of the PRL technique in the EMDL load part slows it down in respect to DPTL. However, a real consumption reduction is confirmed in standby state, giving a better power-delay product than others. The simple design is also observed with the smallest transistor number. The generation of the enable signals is not taken into account in this analysis.

Table VI. Full adder HSPICE simulation results

| MESFET Structures | td (ps) *fi*-out/in-out | Pd (mW) stand./eval. | Pd X td (fJ) | # trans. |
|---|---|---|---|---|
| DCFL | 137.5* | 3.76* | 517.0 | 42 |
| DPTL | 44.3** | 1.16 / 1.02 | 48.3 | 36 |
| DCVS | 210.4 / 238.8 | 1.15 / 1.02 | 243.7 | 56 |
| DC$^2$FL | 449.4 / 358.5 | 0.47 / 1.05 | 307.0 | 58 |
| EMDL | 99.1** | 0.02 / 0.91 | 46.1 | 26 |

*DCFL is not a dynamic structure (only in-out delay).

**DPTL and EMDL have available inputs in evaluation (*fi*-out delay).

### 7.4.3. Specific applications

The use of EMDL in synchronous circuits is obvious: the clock signal is applied to the enable input establishing the precharge phase between each calculation (evaluation phase), when the input data are already available. In the case of asynchronous design, a request signal replaces the enable one and an additional DCFL NOR gate is necessary to detect the operation completion (ack), as illustrated in figure 7.9. It should be noted, however, that only speed independent circuits are possible because of the input availability when the calculation is started.

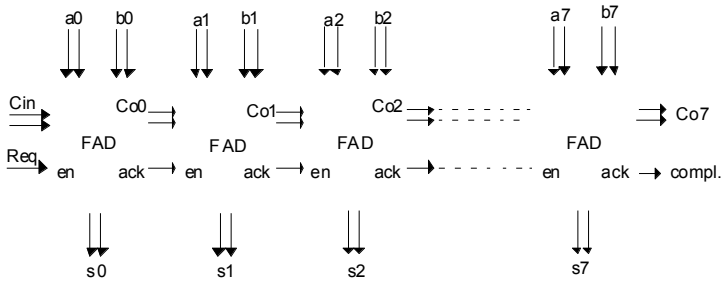
Two interesting applications are iterative networks and micropipelines, discussed in [22] and [25] respectively. An iterative network consists of a one-dimensional array of one or more identical blocks, which are connected only to their neighbors.

Its major advantages are the regular design and the simple connections, and the test easiness. In such applications, the completion signal generated by the DCFL NOR is used to enable the next block. However, shorter data routing delays than combined NOR gate and the completion signal routing delays must be respected to guarantee the correct data flow.

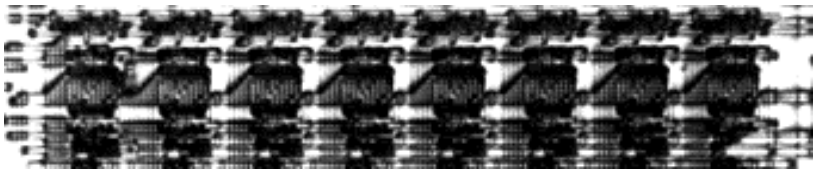
Micropipelines, in turn, can be implemented without explicit delay elements required to satisfy the timing requirements, because the EMDL structure generates its own completion signals, as discussed above. It simplifies the process of synthesizing micropipeline stages and the portability of such circuits between CMOS and MESFET technologies.

### 7.4.4. Experimental results: 8-bit RCA

To demonstrate the EMDL structure functionality, a 8-bit ripple carry adder (RCA) implemented as an iterative network was fabricated through CMP-France services, using Vitesse H-GaAs III technology. The block diagram and the microphotograph of the circuit are illustrated in figure 7.12.




(a)



(b)

Figure 7.12 The 8-bits RCA: (a) block diagram and (b) microphotograph.

HSPICE simulation results are shown in Table VII, and compared with previously published DCVS and DC$^2$FL results [23]. Note that, the DCVS and DC$^2$FL RCA versions are composed by delay-insensitive gates and the request signal is common to all blocks. So, the delay results (td) presented in Table V correspond to average values. In the EMDL version, due to the iterative network configuration, always the worst-case propagation delay gives the speed performance.

All prototypes were verified fully functional and the performance measures were found acceptable. The static currents in standby and evaluation phases were observed at low frequency operation and are shown in figure 7.13, in comparison with the DCVS RCA version. In figure 7.14, a microphotograph of the chip including EMDL, DCVS and DC$^2$FL approachs is shown.

Table VII. 8-bit RCA HSPICE Simulation Results

| | td (ns) | Pd (mW) | Pd X td (pJ) | # transistor |
|---|---|---|---|---|
| DCVS | 0.903 | 8.93 | 8.06 | 448 |
| DC$^2$FL | 1.641 | 6.08 | 9.98 | 464 |
| EMDL | 1.558 | 5.99 | 9.34 | 264 |

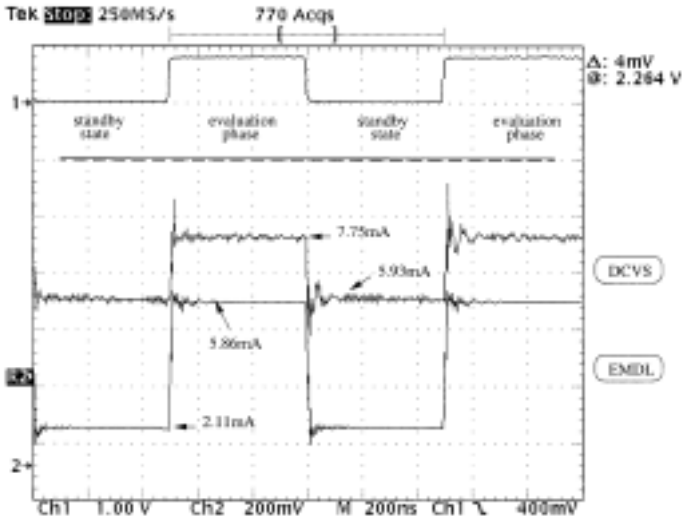



Figure 7.13. Static currents observed in EMDL and DCVS 8-bit RCA circuits [26].

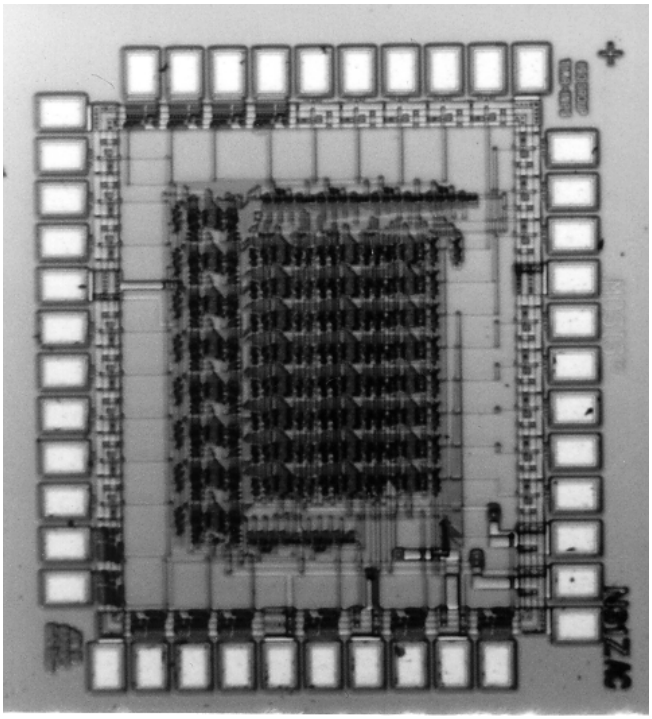


Figure 7.14 EMDL, DCVS and DC$^2$FL 8-bit RCA circuits.

## 7.5. Conclusions

A new GaAs MESFET differential structure for self-timed circuits has been presented. The structure requires only one 2V power supply and uses predischarge technique, achieving lower levels of power consumption. The most important feature of DC$^2$FL is its compatibility with the DCFL, DPTL and DCVS families.Lower levels of power-delay product than DCFL and as low as DCVS are obtained.

The Ripple Carry Adder was chosen to be an architecture-neutral evaluation and the worst possible delay for a n-bits adder. A 8-bit ripple carry adder chip test has demonstrated that DC$^2$FL is an appropriate structure to implement delay-insensitive asynchronous circuits. If extended to synchronous circuits significant power saving ranges are obtained, when mixed circuits are implemented using DC$^2$FL .

The new Enable/Disable GaAs MESFET Differential Logic (EMDL) has been proposed and applied to asynchronous design to show the significant savings in power dissipation. Firstly, because such a differential structure increases the gate complexity, reducing total power consumption per logic function. Secondly, because in asynchronous circuits the functional

blocks are kept in standby state until a calculation is required, avoiding static currents in unused parts of the circuit. The easy design, speed performance characteristics, noise immunity and full compatibility with DCFL, DPTL, DCVS and $DC^2FL$ gates make it also useful to synchronous circuits. A 8-bit ripple carry adder was successfully fabricated and tested, proving the EMDL functionality and features.

A 8-bit ripple carry adder circuit was chosen as a vehicle to demonstrate the advantages of both structures. The Ripple Carry Adder presents the worst possible delay for a n-bits adder. However, other adder topologies with recognized higher performance can be also designed using the mentioned techniques. The pipeline scheme used by two studied asynchronous structures, their performance and power consumption demonstrate the richness of the asynchronous approach. Future projects could be oriented to build long word length adders choosing better adder architectures for this purpose.

## 7.6. References


[1] Hauck, Scott. Asynchronous Design Methodologies: an overview. Proceedings of the IEEE. Vol.83, no.1, Jan.95, pp.69-93.

[2] Ribas, R.P. & Guyot, A. DCFL- and DPTL-based approaches to self-timed GaAs circuits. ESSCIRC 95. Proceedings, Lille, France, Sep.95. pp.186-89.

[3] Chandramouli, V.; Brunvand, E. & Smith, K.F. Self-timed design in GaAs - case study of a high-speed, parallel multiplier. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Vol. 4, no.1, Mar.96, pp.146-49.

[4] Tierno, J.A.; Martin, A.J.; Borkovic, D. & Lee, T.K. A 100-MIPS GaAs asynchronous microprocessor. IEEE Design & Test of Computers. Summer, 94, pp.43-49.

[5] R. P. Ribas, and A. Guyot, "DCFL- and DPTL-based approaches to self-timed GaAs circuits," Proceedings of the 21th European Solid-State Circuits Conference, Lille - France, Sep. 95, pp. 186-189.

[6] G.M. Jacobs, R.W. Brodersen, A Fully Asynchronous Digital Signal Processor using self-timed circuits, IEEE Journal of Solid-State Circuits, Vol. 25, no. 6, Dec. 90, pp. 1526-1537.

[7] Pasternak, J.H. & Salama, C.A.T. GaAs MESFET differential pass-transistor logic. IEEE Journal of Solid-State Circuits. Vol.26, no.9, Sep.91, pp.1309-16.

[8] Hoe, D.H.K. & Salama, C.A.T. GaAs trickle transistor dynamic logic. IEEE Journal of Solid-State Circuits. Vol.26, no.10, Oct.91, pp.1441-48.

[9] Law, O.M.K. & Salama, C.A.T. GaAs split phase dynamic logic. IEEE Journal of Solid-State Circuits. Vol.29, no.5, May.94, pp.617-22.

[10] Chandramouli, V.; Michell, N. & Smith, K.F. A new, precharged, low-power logic family for GaAs circuits. IEEE Journal of Solid-State Circuits. Vol. 30, no.2, Feb.95, pp.140-43.

[11] I.E. Sutherland, Micropipelines, Communications of the ACM, Vol., 32, no. 6, June, 1989, pp. 720-738.

[12] T.H. Meng, Synchronization Design for Digital Systems, Kluwer Academic Publishers, Massachusetts, 1991

[13] M.E. Dean, D.L. Dill, M Horowitz, Self-Timed Logic Using Current Sensing Completion Detection (CSCD), Journal of VLSI Signal Processing, 7, 1994, pp 7-16.

[14] G. Birtwistle, A. Davis, Asynchronous Digital Circuit Design, Springer, 1995

[15] Williams, T.E. Performance of iterative computation in self-timed rings. Journal of VLSI Signal Processing, 7, 1994. pp. 17-31.

[16] Lee, G.; Donckels, B.; Grey, A. & Deyhimy, I. A high density GaAs gate array architecture. IEEE Custom Integrated Circuits Conference. Proceedings. San Diego, CA, 1991.

[17] Chandna, A.; Brown, R.B.; Putti, D. & Kibler, C.D. Power rail logic: a low power logic style for digital GaAs circuits. IEEE Journal of Solid-State Circuits. Vol.30, no.10, Oct.95, pp.1096-100.

[18] Long, S.I. & Butner, S.E. Gallium arsenide digital integrated circuit design. McGraw-Hill Publishing Company . 1990.

[19] Mathieu, H. Physique des semiconducteurs et des composants electroniques. Masson. 1990.

[20] Wing, Omar. Gallium arsenide digital circuits. Kluwer Academic Publishers. 1990.

[21] R. P. Ribas, A. Bernal, and A. Guyot, Low-power differential cross-coupled FET logic for GaAs asynchronous design, Proc. of the European Gallium Arsenide and Related III-V Compounds Application Symposium, Paris - France, Jun. 96, pp. 2A5.

[22] S. H. Lu, Implementation of iterative networks with CMOS differential logic, IEEE Journal of Solid-State Circuits, vol. 23, no. 4, Aug. 88, pp. 1013-1017.

[23] A. Bernal, R.P. Ribas, A. Guyot, Low-Power Differential Logic, Energy and Computation Magazine, Vol. 7, No. 1, Ed.-13, January, pps:13-19, 1998

[24] K. M. Chu, and D. Pulfrey, Design procedures for differential cascode voltage switch circuits, IEEE Journal of Solid-State Circuits, vol. 21, no. 6, Dec. 86, pp. 1082-1087.

[25] S. H. Lu, Implementation of micropipelines in enable/disable CMOS differential logic, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 3, no. 2, Jun. 96, pp. 338-341.

[26] R.P. Ribas, A. Bernal, A. Guyot, A low-Power Enable/Disable GaAs MESFET Differential Logic, Proc. IEEE 18$^{th}$ GaAs IC Symposium, Orlando, USA, Nov., 1996.

## Abstract

Currently. the cryptography processes rest mainly on protocols which use the concept of one-way function. This type of mathematical jamming is frequently carried out by the modular exponentiation function. In this work, the design of an alternative architecture which satisfies the most significant characteristics in order to guarantee the viability of an integrated circuit for calculating the modular exponentiation function is presented. In the architecture, the main advantages of both, generalised square-multiply binary for exponentiation function and the Montgomery's algorithm for modular multiplication are mixed. The architecture is oriented to compute the modular exponentiation of large integer numbers presenting a good performance and a modularity being easily expandable to larger bit-widths. In addition, as several of cryptography applications use satellite communication where high performances but principally radiation tolerant integrated circuits are needed, AsGa become as a suitable technology for the implementation of this type of system. The design of two of the principal blocks of the proposed architecture considering low power strategies consumption are also presented.



**Keywords**: Modular Arithmetic architectures, Modular Exponentiation, Gallium Arsenide.



## Résumé

Les processus de sécurisation d'information privée reposent principalement sur des protocoles qui utilisent le concept de fonction à sens unique ou fonction très difficilement inversible. Ce type de brouillage mathématique est fréquemment réalisé par la fonction exponentielle modulaire. Dans ce travail on présente la conception d'une architecture performante qui satisfait aux caractéristiques les plus importantes afin de garantir la viabilité d'un circuit intégré pour le calcul de la fonction exponentielle modulaire. L'analyse de l'architecture permet d'évaluer les gains en vitesse qu'une réalisation matérielle pourrait permettre par rapport aux algorithmes programmés. L'architecture calcule la fonction exponentielle modulaire des numéros représentés en notation modulaire en combinant les avantages de l'algorithme de Montgomery pour la multiplication et ceux de la méthode généralisée de multiplications répétées, pour l'exponentielle. D'autre part, la nécessité de performances élevées inhérentes aux applications, soit en cartes à puce soit en communication par satellite, a fait considérer l'AsGa comme une technologie appropriée pour l'implémentation de ce type de système. La conception de deux des principaux blocs de l'architecture envisageant la basse consommation a été aussi réalisée.



**Mots clés**: Architectures d'Arithmétique Modulaire, Exponentielle Modulaire, Arséniure de Gallium.

ISBN 2-913329-30-6 broché
ISBN 2-913329-31-4 électronique