Techniques de modélisation transactionnelle pour le dimensionnement des futurs systèmes de radiocommunication mobiles

Anthony Barreteau

UNIVERSITÉ DE NANTES

Ecole polytechnique de l'Université de Nantes

ÉCOLE DOCTORALE

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET MATHÉMATIQUES

Année 2010

N° attribué par la bibliothèque

TECHNIQUES DE MODELISATION TRANSACTIONNELLE POUR LE DIMENSIONNEMENT DES FUTURS SYSTEMES DE RADIOCOMMUNICATION MOBILES

THÈSE DE DOCTORAT

Discipline – Spécialité : Électronique

Présentée
et soutenue publiquement par

Anthony BARRETEAU

le 8 décembre 2010, devant le jury ci-dessous

Président	Michel AUGUIN, Directeur de Recherche CNRS, LEAT, Université Nice Sophia Antipolis
Rapporteurs	Guy GOGNIAT, Professeur, Lab-STICC, Université de Bretagne Sud
	Dominique HOUZET, Professeur, GIPSA-Lab, Institut Polytechnique de Grenoble
Examinateurs	Jean-François DIOURIS, Professeur, IREENA, Polytech'Nantes
	Sébastien LE NOURS, Maître de Conférences, IREENA, Polytech'Nantes
	Christophe MOY, Professeur SUPELEC, IETR, Supélec Rennes

Directeur de thèse : Jean-François DIOURIS
Encadrant : Sébastien LE NOURS
Laboratoire IREENA EA1770

ED 503-116

Remerciements

Je tiens tout d'abord à remercier Monsieur Jean-Paul Calvez de m'avoir accueilli dans son équipe en tant que doctorant. Je lui exprime toute ma reconnaissance et mon estime de ses qualités tant scientifique qu'humaine. Je lui souhaite une retraite paisible et heureuse.

J'adresse ensuite toute ma gratitude à Monsieur Sébastien Le Nours pour la qualité de son encadrement. Ce travail a en effet pu être mené dans de bonnes conditions du fait de sa disponibilité, de sa qualité d'écoute, des échanges constructifs que nous avons pus avoir, et des encouragements qu'il a sus m'apporter dans les moments de doute que l'on peut rencontrer au cours de cette expérience. Pour tout cela, je souhaite le remercier.

Je souhaite également remercier Monsieur Olivier Pasquier et Monsieur Takieddine Majdoub pour les échanges que nous avons pus avoir aussi bien d'un point de vue scientifique qu'amical au sein de notre équipe.

Je remercie aussi Monsieur Jean-François Diouris d'avoir accepté d'être mon directeur de thèse lors de cette dernière année.

Mes remerciements s'adressent également aux membres du jury qui ont accepté de consacrer du temps à la lecture et à l'examen de ce travail de recherche.

Je tiens aussi à remercier toutes les personnes du laboratoire IREENA qui ont participé au bon déroulement de mes travaux de thèse et que j'ai eu plaisir à côtoyer. Merci aussi aux enseignants du département ETN de Polytech'Nantes et aux ingénieurs de Synervia pour leur bonne humeur au quotidien.

Enfin un immense merci à mes parents pour leur amour et le soutien qu'ils m'ont apporté tout au long de mes études. Un grand merci aussi à toute ma famille, mes amis sur qui je peux toujours compter. Si j'écris aujourd'hui ces quelques lignes de remerciement témoignant de l'aboutissement de mon travail de doctorant, c'est bien sur un peu grâce à vous.

Table des matières

Liste des figures

Liste des tableaux

Introduction

Les systèmes de téléphonie mobile n'ont cessé d'évoluer au cours des dernières années de manière à favoriser le transfert d'informations à des débits de plus en plus élevés et à assurer une mobilité accrue de l'utilisateur. Ces évolutions ont permis de diversifier les applications pouvant être accessibles via différents réseaux. La télévision, la radio, l'accès aux contenus disponibles sur Internet, la visiophonie, le GPS (*Global Positioning System*) sont autant d'applications qui commencent ainsi à enrichir les offres initialement proposées sur les premiers téléphones portables comme la communication vocale ou l'envoi de SMS (*Short Message Service*). Les dernières générations de terminaux mobiles sont donc en train de devenir de véritables couteaux suisses de l'information et de la communication.

La prochaine génération de réseaux de radiocommunication mobiles, communément appelée 4G pour 4ème génération, doit permettre d'améliorer les conditions d'utilisation de ces applications sur les futurs terminaux mobiles. L'objectif est de proposer à l'utilisateur une qualité de service (*QoS*) optimale quels que soient le lieu où il se trouve, le moyen qu'il emploie pour se déplacer ou les applications qu'il utilise. Pour cela, les différents réseaux de communication sans fil actuels et futurs vont devoir converger au sein d'une infrastructure de communication globale. Les prochaines générations de terminaux mobiles seront alors dotées de nouvelles fonctionnalités qui leur permettront d'interagir avec cette infrastructure afin d'identifier et de mettre en œuvre la connexion radio la plus appropriée pour acheminer les données en fonction de l'évolution des besoins de l'utilisateur et de ses déplacements. Pour cela, ces systèmes sont amenés à supporter une large gamme de standards de communication.

Suite à l'accroissement du nombre de fonctionnalités devant être intégrées et en raison des contraintes de plus en plus fortes en termes de coût, de surface, de consommation et de délai de mise sur le marché à respecter, le travail de conception des futurs terminaux mobiles se révèle de plus en plus complexe à réaliser. Des propositions doivent donc être faites pour accompagner les concepteurs de ces systèmes afin d'améliorer leur productivité.

Une des problématiques importantes qu'il semble nécessaire d'appréhender se rapporte au travail de dimensionnement mené par l'architecte système. Ce travail consiste à correctement définir les ressources de calcul, de communication et de mémorisation requises pour pouvoir mettre en œuvre les différentes fonctionnalités devant être supportées par les futurs terminaux mobiles. Actuellement cette activité repose le plus souvent sur une approche pragmatique et incrémentale basée sur une expérience préalablement acquise dans la définition des ressources appropriées selon les différentes contraintes imposées. En raison de la multiplication des fonctionnalités à mettre en œuvre et suite aux nouvelles possibilités offertes par les ressources proposées pour les exécuter, ce travail devient de plus en plus complexe à réaliser avec les méthodes actuelles.

La tendance constatée consiste à favoriser l'utilisation de modèles définis à un haut niveau d'abstraction pour permettre une évaluation rapide et efficace des différentes solutions pouvant être envisagées pour réaliser un système. Le concept de modélisation « transactionnelle » aussi appelée TLM pour *Transaction-Level Modeling* offre actuellement des possibilités intéressantes pour la définition de modèles dans le cadre de ce travail. En effet, cette approche définit un ensemble de niveaux d'abstraction qui se différencient par le niveau de détail à considérer pour décrire les différentes propriétés d'un système. Les modèles définis à ces niveaux d'abstraction offrent alors un compromis intéressant entre temps de simulation requis et précisions des résultats obtenus. De plus, il est actuellement

possible de s'appuyer sur des langages tels que SystemC et SystemVerilog pour la description et l'exécution de modèles définis à différents niveaux d'abstraction. Dès lors, il convient de favoriser l'utilisation de ces modèles, et ce par la définition de méthodes correctement formalisées pour faciliter leur création.

Les contributions originales de cette thèse ont pour objectif d'améliorer la productivité de l'architecte système dans le cadre du dimensionnement des futurs terminaux mobiles. Ces travaux portent donc sur la définition d'une démarche guidant l'architecte système jusqu'à la définition précise des ressources matérielles et logicielles satisfaisant aux exigences imposées par ces systèmes. Cette démarche repose pour cela sur un ensemble d'étapes visant à accompagner l'architecte système lors de la création de modèles exécutables de niveau transactionnel. Un modèle générique est proposé de manière à faciliter la définition de ces modèles. Des propositions sont faites afin de diminuer les temps de simulation des modèles utilisés. Des techniques sont également introduites pour capturer les propriétés essentielles des futurs systèmes de radiocommunication mobiles en vue de leur dimensionnement.

Ce mémoire de thèse est organisé comme suit. Il se décompose en 4 chapitres.

Le chapitre 1 introduit et précise le contexte applicatif de cette thèse. La première partie porte sur les nouveaux usages et services associés aux prochaines générations de terminaux mobiles. Nous précisons ensuite les différentes avancées envisagées au sein des infrastructures des réseaux sans fil pour améliorer les conditions d'utilisation des applications proposées sur ces systèmes. Nous présentons enfin les nouvelles capacités d'adaptation devant être supportées par les futurs terminaux mobiles afin de pouvoir utiliser à tout moment le standard de communication le plus approprié pour répondre aux besoins applicatifs de l'utilisateur. Dans la deuxième partie de ce chapitre, nous décrivons les évolutions envisagées au niveau des architectures matérielles et logicielles pour développer les futurs terminaux mobiles. Nous présentons ensuite le processus actuel de conception de ces systèmes en positionnant le rôle de l'architecte système. La troisième partie de ce chapitre permet au final de présenter les problématiques à adresser dans le cadre du dimensionnement des futurs terminaux mobiles.

Le chapitre 2 concerne l'analyse des travaux effectués sur le dimensionnement des systèmes embarqués. Ce chapitre permet de positionner les différents niveaux d'abstraction utilisés dans le cadre de la conception des systèmes embarqués. Un état de l'art est ensuite présenté sur les différentes propositions faites actuellement pour réaliser le travail de dimensionnement des ressources matérielles et logicielles requises. Les différents langages et outils, sur lesquels il est possible de s'appuyer pour créer des modèles en vue du dimensionnement de ces systèmes, sont aussi présentés. Á partir des observations faites, nous introduisons alors les différentes contributions que nous souhaitons apporter.

Le chapitre 3 présente notre contribution pour la description des propriétés fonctionnelles et non fonctionnelles d'un système au sein de modèles de niveau transactionnel. Les propriétés non fonctionnelles se rapportent aux ressources nécessaires pour la mise en œuvre du système. Ce chapitre est divisé en trois parties. Dans la première partie, nous décrivons l'approche proposée ainsi que la notation utilisée pour représenter les propriétés considérées au sein de modèles pour le dimensionnement de systèmes. Des techniques sont également proposées afin de réduire les temps de simulation nécessaires à l'exécution de modèles de niveau transactionnel. Nous précisons aussi le formalisme du modèle d'exécution générique défini pour faciliter la description des modèles utilisés. Nous montrons ensuite comment il est possible de capturer et de simuler les modèles proposés à l'aide de l'outil CoFluent Studio. Dans la deuxième partie, une expérimentation est

présentée afin d'illustrer la mise en œuvre de notre approche sur un cas d'étude. Nous présentons dans la dernière partie comment cette approche peut s'inscrire dans une démarche de conception plus générale.

Le chapitre 4 présente les deux études de cas proposées pour illustrer les contributions amenées dans le cadre de ces travaux. La première partie de ce chapitre montre la mise en œuvre de notre approche pour le dimensionnement des ressources nécessaires à la réalisation des traitements associés à la couche physique du standard de communication LTE (*Long Term Evolution*). Nous évaluons ici trois solutions d'implantation. La première est composée d'un ensemble de ressources matérielles spécifiques. La deuxième repose sur l'utilisation d'un processeur logiciel. La dernière architecture évaluée correspond à une solution mixte associant un processeur logiciel avec des ressources matérielles spécifiques. Le but est d'estimer les coûts associés en termes de mémoires et de ressources de calcul. Le deuxième cas d'étude permet d'illustrer la mise en œuvre de notre approche pour le dimensionnement d'un terminal mobile adaptatif multiservice. Les techniques de modélisation proposées pour représenter les spécificités de ces systèmes sont présentées. Les résultats de simulation obtenus doivent permettre de caractériser les ressources matérielles requises pour réaliser ce système.

Enfin, une conclusion générale résume les principales contributions de ce travail portant sur la proposition de techniques de modélisation transactionnelle en vue du dimensionnement des futurs terminaux mobiles. Différentes perspectives à ce travail sont également évoquées.

Chapitre 1
Description du contexte applicatif : les systèmes de radiocommunication mobiles de 4ème génération

Sommaire

Ce premier chapitre permet d'introduire le contexte applicatif dans lequel s'inscrit ce travail. Il s'agit ici de présenter les différentes évolutions constatées dans le domaine des systèmes de radiocommunication mobiles. Les attentes des utilisateurs en termes d'applications et de qualité de service sont ainsi introduites. Les progrès envisagés au sein des infrastructures des réseaux sans fil pour répondre à ces besoins sont ensuite présentés. Il en ressort l'émergence d'une nouvelle génération de terminaux mobiles supportant une grande variété d'applications et dotée de nouvelles capacités d'adaptation des interfaces de communication. Les différentes architectures matérielles et logicielles envisagées pour développer ces systèmes sont ensuite présentées. Le processus de développement de ces systèmes est aussi introduit. Ce dernier point permet de positionner le travail de dimensionnement de terminaux mobiles dans le processus de conception de ces systèmes et d'identifier les différentes problématiques qui semblent émerger et qu'il s'agit d'adresser.

1.1 Observations sur l'évolution des services et des infrastructures des réseaux mobiles

1.1.1 Analyse et évolution des besoins des utilisateurs

Dans un passé très proche, un service était associé à un média. La télévision était proposée sur un téléviseur, la radio s'écoutait sur un récepteur du même nom, la communication vocale était réalisée avec un téléphone et Internet était accessible via un ordinateur. Ces différents services étaient donc proposés sur des équipements spécifiques et étaient consommés dans le lieu où se trouvait le moyen d'accès. La tendance actuelle consiste à proposer des solutions qui permettent aux utilisateurs de profiter de ces différents services sur un seul et même équipement, devenant ainsi multimédia et pouvant l'accompagner dans tous ses déplacements. Dans ce contexte, de nouveaux terminaux mobiles multimédias commencent à émerger pour répondre à ces nouveaux besoins en termes de convergence des services et de mobilité. Á terme, ces terminaux permettront ainsi aux utilisateurs d'accéder n'importe où et n'importe quand à une grande variété de services. Le tableau 1.1 [**1**] présente les différents services qui sont ou qui vont être proposés sur ces équipements dans les dix prochaines années ainsi que les évolutions envisagées dans leur utilisation.

Années	Services			
	Appel voix	Communication vidéo	Courrier électronique	Téléchargement audio et vidéo
	Durée appel par mois (en minutes)		Sessions par mois	
2012	600	42	39	12
2020	600	250	50	78

Années	Services			
	Diffusion audio et vidéo en continu	Accès Internet	Jeux en réseau	Géolocalisation, navigation
	Sessions par mois			
2012	5	49	26	8
2020	78	52	52	64

Tableau 1.1 – Illustration de l'évolution de l'utilisation des services mobiles sur les terminaux mobiles.

La communication vocale va continuer à être consommée de la même façon. En revanche, un certain nombre de services vont émerger. Ainsi, le temps passé à regarder ou à écouter des contenus vidéo ou audio via un terminal mobile va être multiplié par quinze en dix ans. Les services de géolocalisation et de navigation vont connaître un réel succès puisqu'ils seront utilisés en moyenne deux fois par jour en 2020.

L'accroissement du nombre de services devant être supportés implique de satisfaire de plus en plus de contraintes diverses et variées lors de la transmission des données associées sur l'interface radio. Ces contraintes à respecter se caractérisent généralement avec les critères suivants :

- le débit définit la quantité d'informations, exprimée en nombre de bits, pouvant être transmise par unité de temps,
- la latence caractérise le temps écoulé entre l'émission et la réception d'un paquet,

- la gigue désigne un écart de délai entre deux paquets consécutifs émis,
- le taux d'erreur binaire exprime le rapport entre le nombre de bits erronés et le nombre de bits transmis,
- le taux de perte représente le pourcentage de paquets de données perdus par rapport au nombre de paquets reçus.

Le tableau 1.2 [**2**] présente les valeurs associées à ces différents critères pour assurer un niveau de qualité de service considéré comme satisfaisant aux utilisateurs. La dernière colonne fait référence à la sensibilité des services aux deux derniers critères évoqués précédemment.

Classes de services	Exemples de services	Débit	Latence	Sensibilité aux erreurs
Conversationnelle	Appel voix	20 kbit/s	50 ms	Faible
	Communication vidéo	5 Mbit/s	20 ms	Faible
Interactive	Navigation Internet	500 kbit/s	200 ms	Forte
	Jeux en réseau	500 kbit/s	20 ms	Forte
Flux	Télévision et radio	2-20 Mbit/s	100 ms	Faible
Arrière plan	Courrier électronique Téléchargement	5-50 Mbit/s	< 2s	Forte

Tableau 1.2 – Critères de qualité de service pour les services mobiles.

La qualité de service proposée aux utilisateurs pour ces différents services est fortement liée aux conditions de transmission proposées par les réseaux d'accès radio pour acheminer les données sur les terminaux mobiles. Les réseaux d'accès radio vont ainsi devoir évoluer pour favoriser l'usage de ces nouveaux services sur ces équipements.

1.1.2 Évolution des infrastructures des réseaux sans fil

Au cours des vingt dernières années, différentes catégories de réseaux sans fil ont été proposées pour permettre l'accès à un ensemble de services via des dispositifs spécialisés et sur des zones de couverture bien définies. La figure 1.1 présente les différents réseaux sans fil qui composent actuellement notre environnement radio.

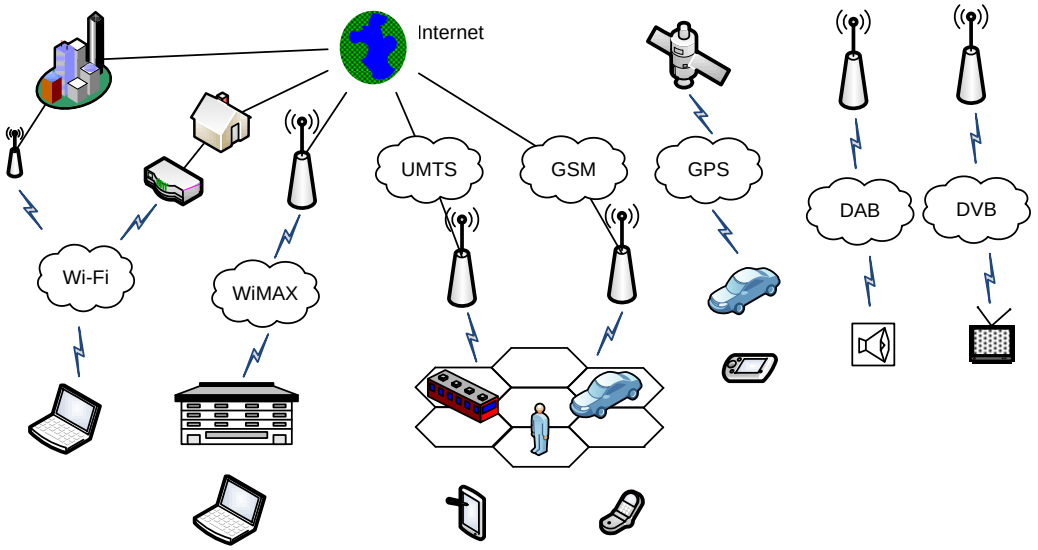

Figure 1.1 – Infrastructures des réseaux sans fil hétérogènes.

Le réseau de télécommunication numérique sans fil de deuxième génération connu sous le nom de GSM [**3**] a vu le jour au début des années 90. Actuellement, il offre la possibilité de pouvoir réaliser de simples communications vocales et l'échange de données de faible volume comme le SMS avec des débits de l'ordre du kbit/s. Une des caractéristiques importante de ce réseau réside dans le fait qu'il assure une continuité de services lors des déplacements de l'utilisateur. Ce type de réseau propose en effet une gestion du *handover* horizontal qui assure le changement de cellule sans perte de connexion lors des déplacements des utilisateurs. Différentes solutions ont ensuite été déclinées afin de diversifier l'offre des services. Le GPRS et l'EDGE [**4**], puis la norme de troisième génération de télécommunication mobile UMTS [**5**] et son évolution HSPA [**6**], ont permis successivement d'améliorer les performances proposées sur les liaisons radio. Les débits atteints à l'heure actuelle sont de l'ordre du Mbit/s. La nouvelle génération de téléphonie mobile connue sous le nom de LTE [**7**] va être déployée dans les prochaines années et devrait encore améliorer les performances des liaisons radio en proposant des débits de 100 Mbit/s. Cette solution devrait favoriser l'usage des services multimédias sur les réseaux cellulaires.

Parallèlement les réseaux informatiques locaux et métropolitains sans fil se sont développés depuis le début des années 2000. Ces réseaux permettent d'offrir une connexion Internet sans fil de quelques dizaines de Mbit/s à des utilisateurs d'ordinateurs portables. Á la différence des réseaux de télécommunications, ces réseaux n'assurent pas de *handover* horizontal entre les différents points de connexion. Les réseaux locaux proposent une zone de couverture de quelques dizaines de mètres autour du point d'accès grâce aux technologies d'accès radio 802.11 [**8**] connues sous le nom de Wi-Fi. Les réseaux métropolitains couvrent quant à eux des zones de quelques kilomètres autour d'une station de base avec le WiMAX et la famille de normes 802.16 [**9**]. Á l'heure actuelle, la tendance pour les réseaux locaux consiste à améliorer les débits sur les liens radio. La dernière technologie 802.11n [**10**] standardisée en 2009 permet par exemple d'atteindre des débits de 300 Mbit/s. Pour les réseaux métropolitains, une nouvelle technologie est en train d'être proposée, le 802.16e [**11**]. Elle doit offrir aux utilisateurs la mobilité déjà offerte dans les réseaux de télécommunications sans fil.

D'autres systèmes ont été proposés pour permettre aux utilisateurs d'accéder à des services bien identifiés. Le système de positionnement mondial GPS par exemple a été développé de manière à offrir des services de navigation et de localisation à des utilisateurs via l'utilisation de terminaux du même nom. Les systèmes DVB [**12**] et DAB [**13**] ont quant à eux été spécifiquement conçus pour la diffusion de la télévision et la radio numérique avec une norme dédiée aux récepteurs portables : le DVB-H [**14**].

La figure 1.2 [**15**] permet d'illustrer les possibilités offertes en termes de débit par ces différentes technologies d'accès radio en fonction de la mobilité des utilisateurs.

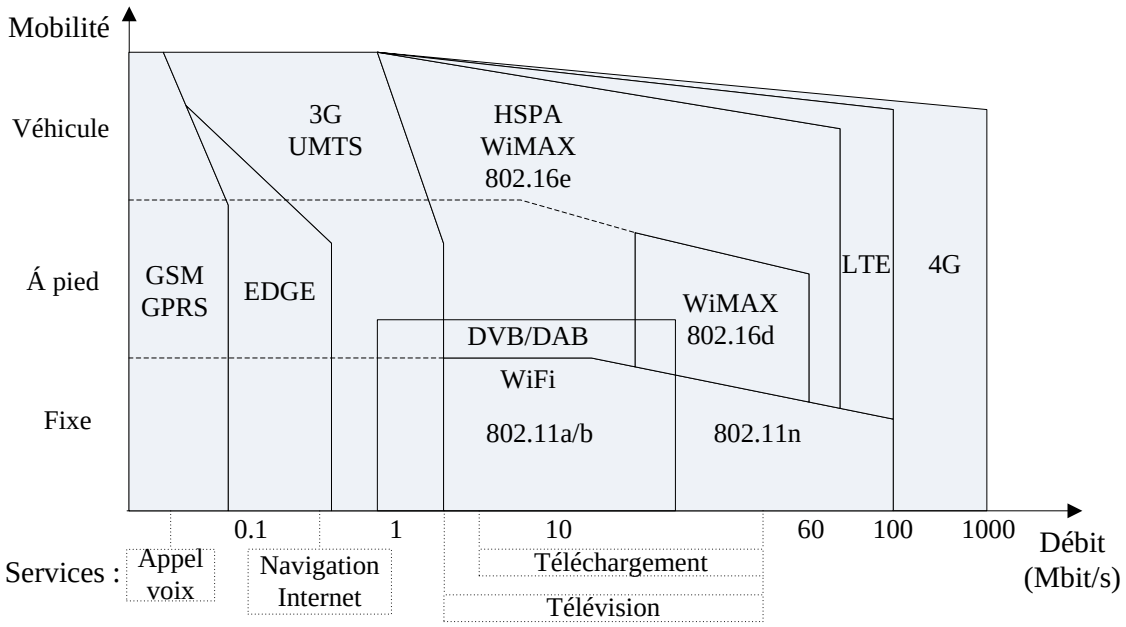

Figure 1.2 – Performances hétérogènes des différentes technologies d'accès radio.

Pour répondre aux besoins croissants en termes de débit des nouveaux services proposés, l'union internationale des télécommunications travaille depuis 2008, dans le cadre du projet IMT-Advanced [**16**], sur la prochaine génération de réseaux Internet haut débit mobile connue sous le nom de 4G. Il est ainsi prévu qu'une nouvelle technologie d'accès radio soit définie en complément de celles existantes afin d'anticiper les besoins de la prochaine génération de services multimédias. Elle devra proposer des débits de 100 Mbit/s en situation de mobilité et de 1 Gbit/s en mobilité réduite afin de pouvoir supporter par exemple de nouveaux services comme la télévision HD. Deux standards de communication sont actuellement en cours de développement par des organismes comme le 3GPP (*3rd Generation Partnership Project*) avec le LTE-Advanced et par le groupe de travail 802.16m avec le Gigabit WiMAX [**17**].

Il est aussi envisagé que le futur système de communication 4G sera capable d'exploiter au mieux la complémentarité des réseaux qui le compose afin d'utiliser la connexion radio la plus adaptée pour répondre aux besoins de l'utilisateur dans un contexte donné. Pour cela, de nouvelles solutions sont envisagées pour gérer une connexion sans discontinuité entre les différents réseaux sans fil. Cette gestion du *handover* dit vertical est déjà proposée entre certains réseaux. On peut citer comme exemple les technologies UMA (*Unlicensed Mobile Access*) [**18**] et I-WLAN (*Interworking-Wireless Local Access Network***)** [**19**] qui assurent respectivement la continuité de services entre le réseau GSM et les réseaux locaux, et entre le réseau 3G et les réseaux locaux. Cependant elle reste limitée à un nombre restreint de réseaux de communication. Ainsi, d'autres standards plus ambitieux comme le 802.21 [**20**] sont en cours de développement pour permettre un *handover* transparent entre les réseaux cellulaires, locaux et métropolitains.

Dès lors, il s'agit de favoriser la définition d'une nouvelle génération de terminaux capables de profiter de la gestion de la mobilité offerte au sein de l'infrastructure des réseaux sans fil de manière à améliorer la qualité de service proposée aux utilisateurs.

1.1.3 Vers l'émergence de terminaux cognitifs

De nos jours, les terminaux mobiles les plus évolués proposent les différents standards de communication GSM, GPRS, EDGE, UMTS et HSPA permettant de se connecter aux réseaux de télécommunications mobiles. Ils supportent aussi d'autres normes comme le Wi-Fi et le GPS afin d'étendre leur offre de services. L'utilisateur a ainsi la possibilité de choisir le réseau de communication qu'il souhaite employer pour accéder aux services qu'il demande. Depuis 2004, de grands projets européens comme End-to-End Reconfigurability (E^2R) [**21**] et son successeur End-to-End Efficiency (E^3) [**22**] sont menés de manière à offrir aux utilisateurs de terminaux mobiles un accès transparent aux services en situation de mobilité. L'objectif est de doter les terminaux mobiles de moyens leur permettant de choisir, parmi les différents réseaux d'accès radio présents dans leur environnement, celui qui est le plus approprié pour proposer les différents services demandés par les utilisateurs avec un niveau de qualité de service toujours satisfaisant. Ce type d'évolution fait référence au concept de la radio cognitive introduit par J. Mitola en 1999 dans [**23**]. La radio cognitive est décrite comme une approche visant à intégrer au sein des objets communicants et des réseaux d'accès radio l'intelligence nécessaire à la prise de conscience des besoins de l'utilisateur en fonction du contexte, et de fournir les ressources de communication permettant de satisfaire ces besoins. Dans le cadre de cette approche, le terminal mobile est donc conscient de l'environnement dans lequel il évolue. Il est doté de moyens lui permettant d'adapter dynamiquement ses interfaces de communication en fonction des évolutions des besoins de l'utilisateur, de l'autonomie de sa batterie et de la charge de trafic supportée par chaque réseau présent dans son environnement. Le scénario nommé « services omniprésents » présenté dans le projet E^2R permet d'introduire les nouvelles fonctionnalités à intégrer au sein des terminaux mobiles pour mettre en œuvre cette approche [**24**]. Ce scénario est illustré par la figure 1.3.

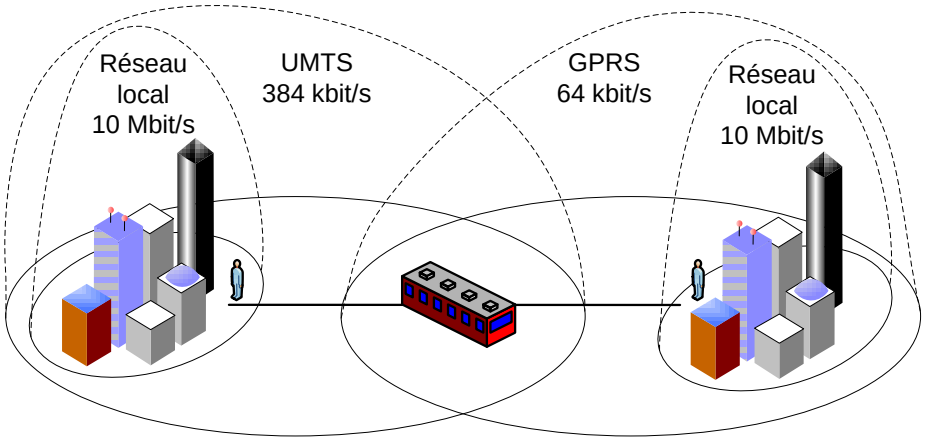

Figure 1.3 – Scénario illustrant les besoins auxquels doivent répondre les futurs terminaux mobiles.

Dans ce scénario, un utilisateur voyage en train et accède à un service de visioconférence via son terminal mobile. Au cours de son trajet, différentes technologies d'accès radio sont présentes dans son environnement. Le terminal mobile dispose alors de moyens lui permettant de passer d'un réseau d'accès radio à un autre de façon transparente pour l'utilisateur afin de lui garantir un niveau de qualité de service satisfaisant tout au long du voyage. Le terminal mobile opère ici en suivant un cycle cognitif en trois phases. Une première phase d'observation permet au terminal de récupérer des informations de plusieurs types :

- les préférences de l'utilisateur dans lesquelles est indiqué le niveau de qualité de service exigé pour le service demandé,
- les paramètres du terminal mobile dans lesquels se retrouvent les informations concernant son niveau de batterie, le taux d'occupation de ses ressources de calcul et de mémorisation, ainsi que les différents standards de communication supportés,
- les paramètres réseaux qui donnent des informations concernant les réseaux d'accès radio disponibles, les standards à utiliser pour établir les connexions radio ainsi que les performances proposées pour délivrer les différents services.

Lors de la deuxième phase de décision, ces informations sont analysées et un standard de communication est choisi afin d'offrir à l'utilisateur l'accès au service demandé avec un niveau de qualité de service satisfaisant. La dernière phase d'exécution consiste alors à changer la configuration des ressources de calcul proposées au sein du terminal mobile de façon à pouvoir exécuter les traitements associés à la chaîne de réception du nouveau standard de communication à utiliser. Les changements de standards de communication que le terminal mobile initie tout au long du voyage sont mis en œuvre de manière à ce que l'utilisateur ne constate aucune dégradation ou interruption de son service de visioconférence.

Depuis 2009, différents organismes comme l'IEEE SCC41 avec le groupe de travail IEEE 1900.4 [**25**] et l'ETSI avec le comité de standardisation RRS [**26**] travaillent sur la spécification des nouvelles fonctionnalités à intégrer au sein des futurs terminaux mobiles et des infrastructures des réseaux sans fil pour proposer ce type d'évolution. Ces fonctionnalités sont décrites sur la figure 1.4 [**27**].

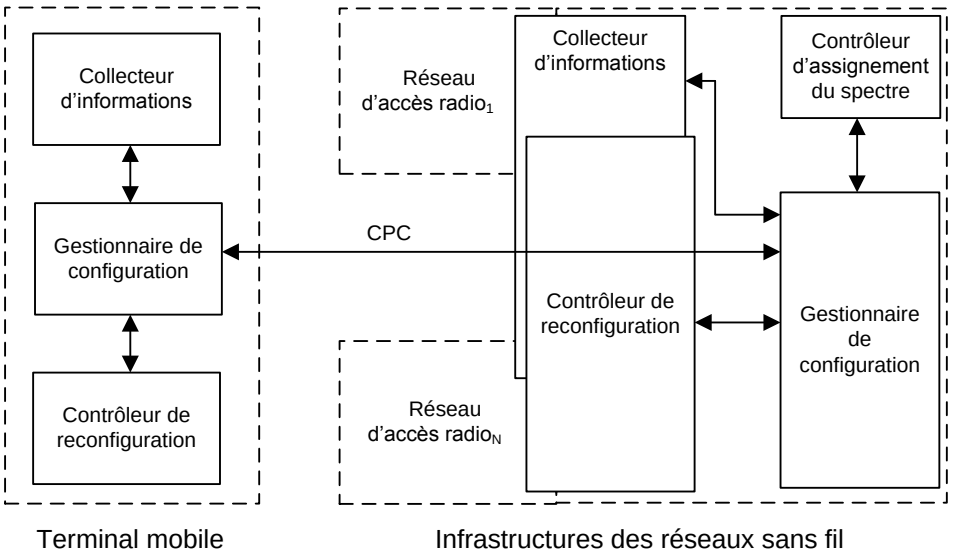

Figure 1.4 – Distribution des nouvelles fonctionnalités au sein des infrastructures des réseaux sans fil et du terminal mobile.

Au sein du terminal mobile, une fonction « Collecteur d'informations » est proposée pour récupérer les informations concernant les capacités de ce dernier et les préférences de l'utilisateur. La fonction « Gestionnaire de configuration » permet elle de décider de la technologie d'accès radio la plus appropriée pour répondre aux besoins de l'utilisateur. Ce choix est fait en analysant les informations collectées au sein du système. Le terminal mobile utilise aussi les informations transmises par la fonction « Gestionnaire de

configuration » proposée au sein des infrastructures des réseaux sans fil. Ces informations précisent les conditions de transmission offertes par les différents réseaux d'accès radio disponibles dans son environnement. Elles sont transmises à la fonction « Gestionnaire de configuration » du terminal mobile en utilisant un canal spécifique dénommé *Cognitive Pilot Channel* ou *CPC*. Ce canal est actuellement en cours de définition par l'ETSI [**28**]. La fonction « Contrôleur de reconfiguration » permet elle de configurer les ressources nécessaires à la mise en œuvre du standard de communication choisi pour réaliser la communication.

Les futurs terminaux mobiles vont ainsi être amenés à supporter de nouvelles fonctionnalités pour améliorer les conditions d'utilisation des services multimédias en situation de mobilité. Cette évolution vers une flexibilité accrue des terminaux mobiles dans la gestion des interfaces de communication implique également des évolutions dans la conception de ces systèmes.

1.2 Observations sur l'évolution de la conception des terminaux mobiles

1.2.1 Constat sur l'évolution des architectures matérielles et logicielles

La section précédente a permis de montrer qu'au cours des quinze dernières années, les terminaux mobiles ont été conçus de manière à proposer toujours plus de standards de communication et à offrir une gamme d'applications élargie à l'utilisateur. De ce fait, les besoins en termes de puissance de calcul pour réaliser les différents traitements associés n'ont cessé d'augmenter. Le tableau 1.3 [**29**] permet de constater que depuis le milieu des années 90, chaque nouvelle génération de terminaux mobiles proposée a exigé dix fois plus de puissance de calcul que la génération précédente. Cette tendance devrait se confirmer pour les prochaines générations de terminaux mobiles 3,9G et 4G qui nécessiteront respectivement des puissances de traitement de l'ordre de 100 et 1000 GOPS.

Années	1995	2000	2005	2010	2015
Générations de terminaux mobiles	2G	2.5-3G	3.5G	3.9G	4G
Standards de communication	GSM	GPRS, UMTS	HSPA	LTE	LTE-Advanced
Puissance de calcul requise [GOPS]	0,1	1	10	100	1000
Nombre de cœurs programmables	1	2	4	8	16
Taille du logiciel [MB]	0,1	1	10	100	1000

Tableau 1.3 – Illustration de l'évolution des besoins pour la conception de terminaux mobiles.

Le tableau 1.3 relève également l'évolution des architectures matérielles proposées en indiquant le nombre de cœurs de processeurs programmables utilisés afin de supporter les traitements associés aux standards de communication et aux applications pour l'utilisateur. Les premières générations d'architectures matérielles reposaient sur l'association de deux types de processeur [**30**]. Un cœur programmable de type RISC permettait de réaliser les traitements liés à l'interface utilisateur et au standard GSM tandis qu'une unité de traitement spécialisée de type DSP était employée pour mettre en œuvre les opérations de traitement du signal associées au service voix. La diversité des traitements proposés au fur et à mesure de l'évolution des besoins applicatifs a ensuite conduit à accroître le nombre de cœurs utilisés et également à diversifier leur nature. Cette hétérogénéité a été utilisée afin de répondre au mieux aux exigences de capacité de traitement et de consommation associées à ces systèmes [**31**].

La figure 1.5 présente un exemple de plate-forme (OMAP 4) et d'unités spécifiques définies par Texas Instrument [**32**] pour développer la prochaine génération de terminaux mobiles 3,9G.

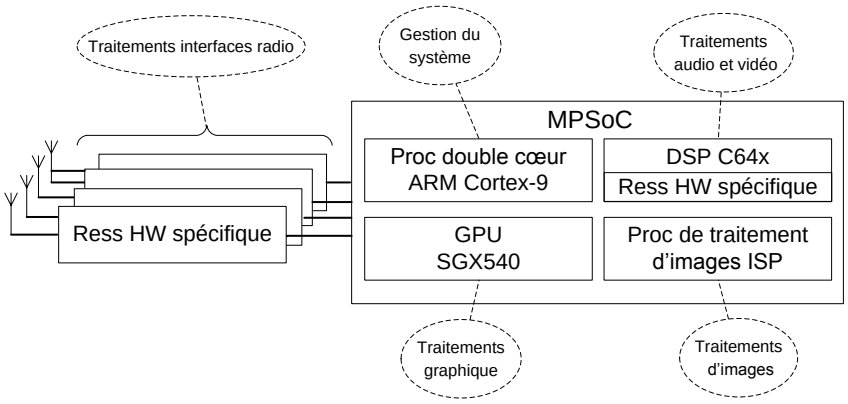

Figure 1.5 – Architecture multiprocesseur hétérogène pour terminaux mobiles 3,9G.

La partie droite de la figure 1.5 présente l'organisation des ressources de traitement hétérogènes proposées au sein d'un MPSoC (*Multiprocessor System-on-Chip*). Elles doivent permettre de supporter les traitements associés aux différentes applications utilisateur. Le processeur graphique SGX540 est proposé pour réaliser les opérations avancées d'affichage sur l'interface utilisateur requérant des traitements spécifiques. Une ressource matérielle spécifique assure les traitements intensifs d'encodage et de décodage des flux vidéo selon les différents codecs actuels. Le processeur DSP C64x est défini pour réaliser le traitement de flux audio et vidéo. Son caractère programmable offre la possibilité de pouvoir supporter de nouvelles normes vidéo. Le processeur ISP a été spécifiquement développé pour réaliser les traitements évolués pouvant être appliqués sur des images. Enfin, le processeur double cœur ARM Cortex 9 doit permettre de supporter le service de navigation Internet et la gestion de la mise en œuvre des traitements sur les différentes ressources de la plate-forme. La partie gauche de la figure 1.5 représente quant à elle les ressources matérielles spécifiques définies pour réaliser les traitements associés aux différentes familles de standards de communication existants (GPS, WLAN, Bluetooth et 3G/3,9G). Ce type de solution d'implantation représente une alternative intéressante à la juxtaposition de circuits dédiés pour la réalisation des traitements associés à chaque standard de communication. Elle vise à favoriser le partage des ressources entre les différents standards de manière à respecter la contrainte forte en termes de surface qui s'applique à ces systèmes.

Cette approche s'inscrit dans le cadre du paradigme de la radio logicielle introduit par J. Mitola en 1995 [**33**]. L'objectif est de tirer parti de la flexibilité offerte par des ressources programmables pour ajouter ou mettre à jour un standard de communication par simple reprogrammation du processeur. L'idée de base de la radio logicielle consiste à réaliser en numérique l'ensemble des traitements associés aux différents standards de communication. Dans sa version idéale, ces traitements sont exécutés via un seul circuit de type processeur. Chaque standard de communication est alors développé en logiciel. Actuellement, en raison du compromis capacité de traitement/consommation offert par les processeurs [**34**], une telle mise en œuvre n'est pas envisageable. En se référant aux besoins en termes de capacité de traitement pour la 3,9G, de l'ordre de 100 GOPS, et de consommation à respecter d'environ un watt [**29**], il n'est pas envisageable d'utiliser un seul processeur de type RISC fonctionnant à une fréquence de 100 GHz. De la même façon, l'association d'une centaine de processeurs fonctionnant à une fréquence de 100 MHz n'est pas viable en raison des

contraintes fortes en termes de surface associées à ces systèmes. L'alternative proposée et désignée par l'appellation radio logicielle « restreinte » ou SDR (*Software Defined Radio*) consiste à utiliser un ensemble de ressources hétérogènes afin d'offrir la puissance de calcul et la flexibilité requises pour réaliser un terminal mobile à un coût acceptable en termes de consommation et de surface. Dans [**29**], une organisation possible des différentes ressources matérielles hétérogènes proposées pour développer les dernières générations de terminaux mobiles est présentée. Elle est décrite sur la figure 1.6.

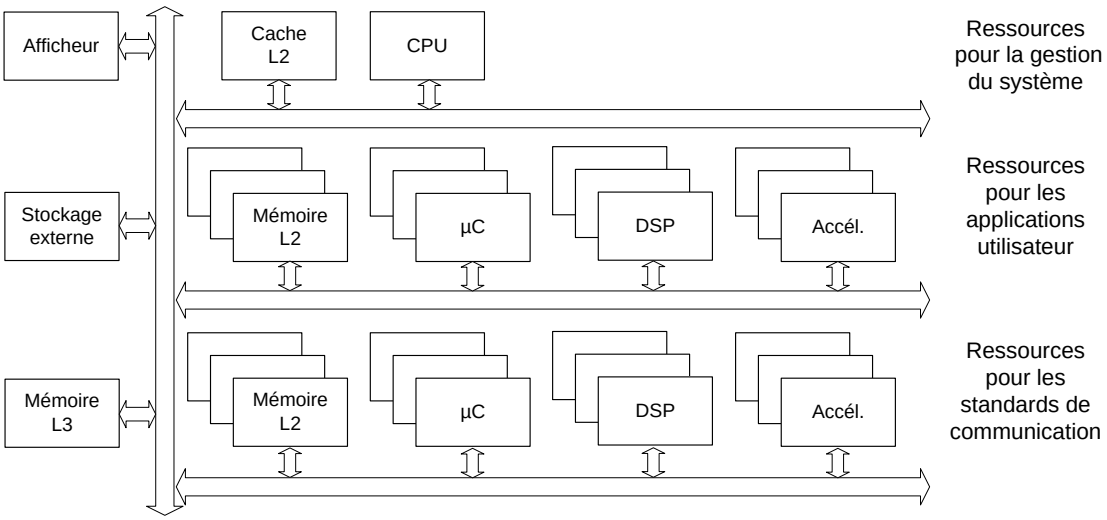

Figure 1.6 – Exemple d'architecture matérielle proposée pour réaliser un terminal mobile.

Sur cet exemple, l'architecture est composée de plusieurs domaines de ressources matérielles hétérogènes permettant de réaliser les différentes catégories de traitements. La gestion du système est assurée par un processeur à usage général. Différentes ressources de calcul et de mémorisation sont définies pour mettre en œuvre les traitements associés aux applications multimédias. Les traitements relatifs aux différents standards de communication sont eux aussi réalisés sur des ressources programmables plus ou moins spécialisées. Les fonctions nécessitant des calculs intensifs sont exécutées via des accélérateurs matériels dédiés. Toutes ces ressources de calcul et de mémorisation sont interconnectées via une architecture de communication de plus en plus évoluée. Différents circuits proposant ce type de ressources hétérogènes pour réaliser les traitements associés aux interfaces de communication des terminaux mobiles sont actuellement commercialisés. On peut citer par exemple le MPSoC SB3500 de la société Sandbridge [**35**], l'architecture à base de processeurs EVP de ST-Ericsson [**35**] ou encore le circuit X-Gold SDR 20 d'Infineon [**36**]. Ils ont été définis pour apporter plus de flexibilité à l'exécution des traitements associés aux dernières générations de standards de communication 3,5G et 3,9G. Les ressources proposées peuvent en effet être partagées entre les différents standards de communication.

Afin d'apporter encore plus de flexibilité à ces architectures, la tendance actuelle consiste à remplacer les accélérateurs matériels dédiés à la réalisation d'une certaine catégorie de traitements par des ressources matérielles offrant des capacités de reconfiguration dynamique. Une ressource matérielle reconfigurable dynamiquement est composée d'un ensemble de ressources opératoires dont la configuration peut être changée au fil du temps afin de pouvoir réaliser plusieurs catégories de traitements [**37**]. Le niveau de granularité associé aux éléments opératoires proposés définit alors la capacité de cette ressource à pouvoir réaliser efficacement différents traitements. Cette efficacité se caractérise par la diversité des traitements qu'elle est capable de mettre en œuvre et par le

temps nécessaire pour effectuer un changement de configuration. Des opérateurs à grain fin comme des LUT, ou table de scrutation, permettent par exemple de réaliser n'importe quelles fonctions booléennes. Ces opérateurs nécessitent cependant des temps de reconfiguration plus long que pour des opérateurs à gros grain correspondant à des unités de traitement plus évolués. Ils permettent de réaliser des calculs arithmétiques [**38**]. La figure 1.7 permet d'illustrer l'évolution des configurations sur une ressource matérielle offrant des capacités de reconfiguration dynamique.

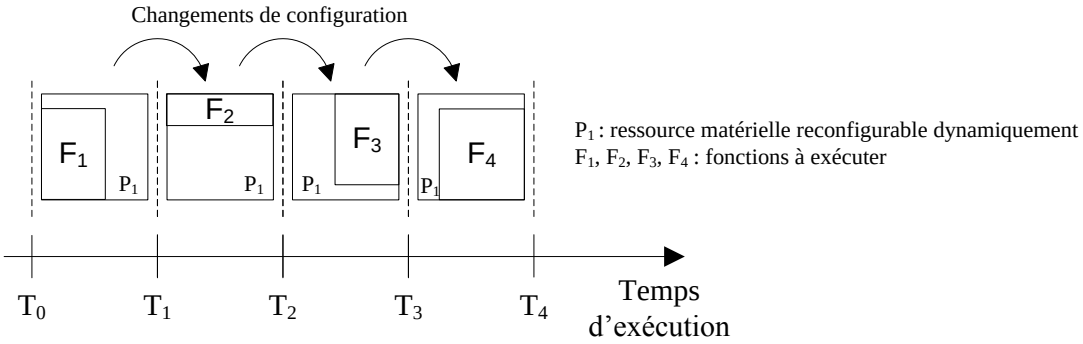

Figure 1.7 – Flexibilité offerte par la reconfiguration dynamique.

La mise en œuvre d'un processus de reconfiguration dynamique permet ainsi d'adapter la configuration des éléments de base de cette ressource matérielle afin de pouvoir réaliser successivement les fonctions de traitement F1, F2, F3 puis F4. Différents travaux dans le domaine académique sont actuellement menés afin de pouvoir tirer parti de la flexibilité et des performances offertes par les ressources matérielles reconfigurables dynamiquement. On peut citer les travaux menés par l'équipe SCEE (*Signal Communication et Electronique Embarquée*) de Rennes qui visent à identifier les différents opérateurs de calcul pouvant être partagés par les différents standards de communication pour mettre en œuvre les différentes fonctions de traitement associées [**39**]. Des propositions d'architectures matérielles associant plusieurs types de processeur avec des ressources matérielles reconfigurables commencent aussi à émerger. On peut citer les propositions faites dans le cadre des travaux menés dans des laboratoires de recherche comme l'IMEC avec leurs architectures ADRES [**40**] qui associent des processeurs évolués de type VLIW à des zones de reconfiguration à gros grain. Le CEA-LETI a aussi proposé différentes générations d'architectures MPSoC comme FAUST [**41**], ALPIN [**42**] et MAGALI [**43**]. Elles sont composées d'un ensemble de ressources de traitement hétérogènes pouvant être interconnectées via un réseau sur puce. Les différentes IP matérielles proposées peuvent être modifiées pour réaliser les différents traitements des standards de communication.

Dès lors, les concepteurs sont amenés à développer de nouvelles architectures logicielles pour tirer parti de la flexibilité offerte par les ressources matérielles proposées pour réaliser les prochaines générations de terminaux mobiles. Les terminaux mobiles vont en effet être dotés de moyens leur permettant de gérer l'exécution des traitements associés aux différents standards de communication à différents instants sur ces différentes ressources matérielles. Les travaux les plus avancés sur ces aspects ont été réalisés dans le cadre du projet JTRS initié par le département de la défense américaine. Une architecture logicielle de référence dénommée SCA (*Software Communications Architecture*) a été proposée [**44**]. Elle définit un ensemble de services et de règles pour la conception d'architectures logicielles utilisées au sein de systèmes SDR dans le domaine militaire. D'autres travaux sur ces aspects ont été initiés par l'agence de défense européenne dans le cadre du projet ESSOR [**45**]. Dans le cadre du projet européen E^2R [**46**], les travaux menés ont permis de spécifier les différentes

entités à mettre en œuvre au sein des prochaines générations de terminaux mobiles. La figure 1.8 présente de façon schématique l'architecture logicielle composée de trois entités qui a été proposée.

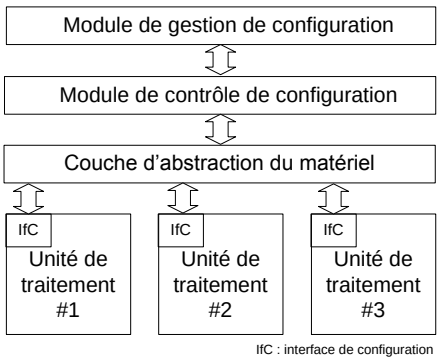

Figure 1.8 – Organisation des ressources logicielles pour la réalisation des traitements associés aux interfaces radio sur des ressources matérielles hétérogènes flexibles.

Une entité dénommée « Module de gestion de configuration » regroupe tous les services permettant au terminal de décider de la configuration à mettre en œuvre. Cette entité a accès à des informations concernant les ressources à utiliser, les interconnexions à réaliser, les algorithmes à exécuter et l'ordonnancement de ces traitements. Une entité « Module de contrôle de configuration » est ensuite en charge d'initier, contrôler et exécuter les configurations des ressources disponibles. La configuration des différentes unités de traitement se fait alors par le biais des interfaces de configuration qu'elles proposent et d'une couche d'abstraction du matériel. Á partir de ces spécifications, différentes propositions d'architectures ont vu le jour. Dans le cadre des travaux de recherche menés par l'équipe SCEE sur l'étude et la conception des futurs systèmes de communication, plusieurs évolutions d'architectures adaptées aux équipements radio logiciels ont été proposées. Un gestionnaire de configuration hiérarchique a dans un premier temps été présenté pour permettre le déploiement et la réalisation des traitements associés à différents standards de communication sur une architecture matérielle hétérogène et reconfigurable. Des extensions ont ensuite été proposées pour que cette architecture puisse répondre aux besoins de la radio cognitive. L'architecture logicielle a ainsi été dotée d'une capacité de prise de décision ainsi que de moyens pour obtenir des informations pour l'aider dans ces choix. Des informations plus détaillées sur ces architectures sont disponibles dans les thèses de doctorat de Jean Philippe Delahaye [**47**] et Loïg Godard [**48**]. D'autres travaux comme ceux menés dans le cadre de projets comme OverSoC [**49**] ou FOSFOR [**50**] visent à intégrer ces nouveaux services au sein d'un exécutif temps réel afin de pouvoir gérer dynamiquement l'exécution de traitements aussi bien sur des unités de traitements logicielles que matérielles.

Le développement des futurs terminaux mobiles implique donc l'utilisation de nouvelles architectures reposant sur un ensemble de ressources matérielles et logicielles. Ces ressources pouvant être programmables et reconfigurables de manière à apporter la flexibilité requise pour exécuter les différentes fonctions de traitement associées à l'ensemble des standards de communication devant être supportés. Dès lors, des évolutions doivent être apportées au processus de conception de ces systèmes de manière à prendre en compte ces nouveaux aspects.

1.2.2 Processus actuel de conception des terminaux mobiles

Le processus de développement d'un terminal mobile se décompose en une série d'étapes organisées comme décrit sur la figure 1.9.

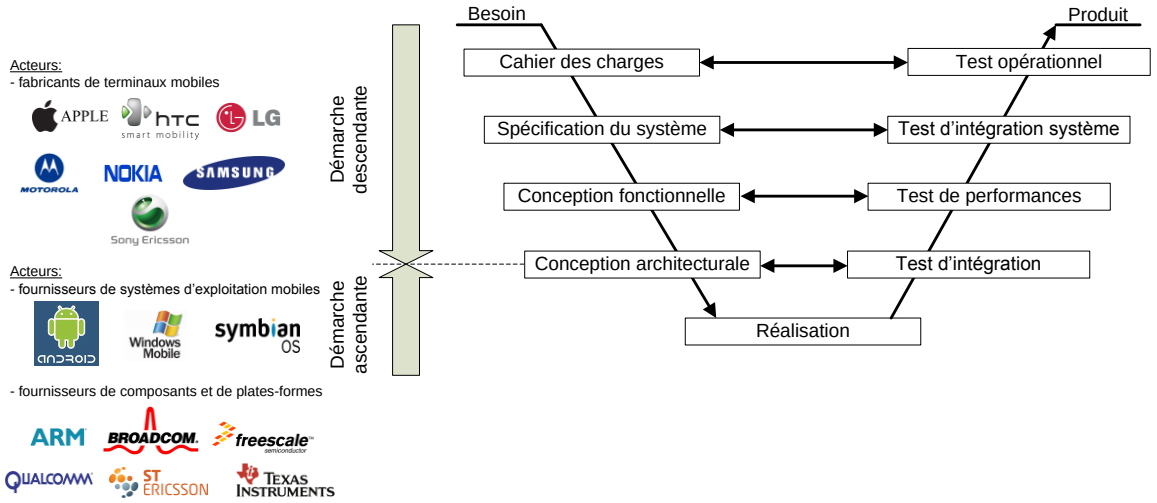

Figure 1.9 – Acteurs et processus de conception des terminaux mobiles.

La première étape du processus de conception consiste à exprimer le besoin auquel le système doit répondre. Les différentes étapes de spécification, conception, réalisation et vérification permettent d'aboutir à un produit opérationnel répondant au besoin initialement exprimé par le client dans le cahier des charges. Cette organisation temporelle du développement d'un terminal mobile s'appuie actuellement sur une approche orientée plate-forme. Cette approche permet de réduire le temps et les coûts associés à la conception et à la validation d'architectures matérielles et logicielles [**51**]. Une plate-forme peut être vue comme un ensemble de ressources matérielles et logicielles qui sont associées afin de pouvoir développer une gamme de terminaux mobiles. Plusieurs acteurs interviennent dans le cadre de la définition d'une plate-forme. Les fabricants de plates-formes utilisent des ressources matérielles préconçues (processeurs, IP pour *Intellectual Property*, mémoires,…) en interne ou par d'autres sociétés. Les architectures de plates-formes consistent alors en l'assemblage des différentes ressources nécessaires compte tenu des besoins associés à une gamme de produits. Pour faciliter l'interopérabilité entre ces différentes ressources, le consortium SPIRIT [**52**] travaille actuellement sur la proposition d'un standard nommé IP-XACT. Les fabricants de plates-formes développent également les couches logicielles de bas niveau associées. Ils intègrent enfin sur les terminaux mobiles les plus évolués un système d'exploitation développé par une société tierce pour faciliter le développement d'applications sur leur plate-forme. Les fabricants de terminaux mobiles s'appuient ensuite sur ces plates-formes de base pour mettre au point les différents types de produit qu'ils souhaitent commercialiser. Le dimensionnement de la plate-forme utilisée se fait alors en lien avec les applications à supporter.

Le travail de conception de plate-forme pour les terminaux mobiles repose sur une approche pluridisciplinaire où différents corps de métier doivent interagir comme on peut le voir sur la figure 1.10 [**53**].

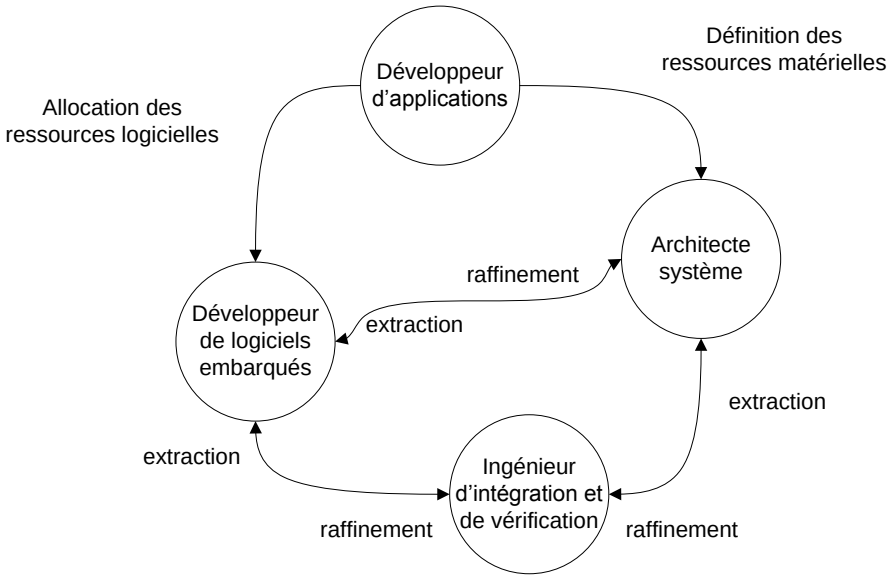

Figure 1.10 – Corps de métier pour la conception des terminaux mobiles.

Quatre corps de métier se distinguent dans le cadre de la conception des architectures numériques pour terminaux mobiles. Le développeur d'applications est en charge de mettre au point l'application supportée par le système, et ce de manière indépendante de toutes considérations technologiques. La description de l'application se fait le plus souvent à l'aide de langages procéduraux (C), objets (C++) ou d'outils spécifiques comme Matlab. Cette description forme une spécification exécutable de l'application servant de référence pour les autres étapes du processus de conception. Le développement des ressources logicielles du système consiste à déployer l'application sur les ressources matérielles identifiées. Pour ce faire, il peut s'avérer utile de disposer de 'prototypes virtuels' correspondant à une représentation simulable des ressources matérielles. Cette approche permet d'anticiper le développement du logiciel avant la mise à disposition de cartes prototypes. L'intégration et la vérification consiste en une validation conjointe du bon fonctionnement des ressources logicielles et matérielles, et ce de manière précise au niveau cycle. Dans ce cas, il peut s'avérer utile de disposer d'une description de la plate-forme sur la base de simulateurs de jeu d'instruction ou ISS (*Instruction Set Simulator*) pour les processeurs, et de modèles précis au niveau cycle des bus de communication et des IP spécifiques utilisés. Les différents acteurs du processus de conception interagissent ainsi en utilisant les informations amenées à chaque étape. En ce qui concerne l'architecte système, il tend à jouer un rôle clé dans le cadre du processus de conception des terminaux mobiles. Son travail se positionne en effet entre la phase de spécification du système et la phase de réalisation et de vérification. Il a la responsabilité d'identifier et de caractériser les ressources de calcul, de communication et de mémorisation requises pour mettre en œuvre l'application devant être supportée par le système. Au final, une solution architecturale satisfaisant les différentes exigences associées au système doit être identifiée. Les choix opérés par l'architecte système vont ensuite permettre le démarrage des travaux de développement du logiciel embarqué et d'intégration et de vérification du système. La figure 1.11 extraite de [**54**] illustre le fait que ce travail devrait être mené le plus tôt possible dans le processus de conception.

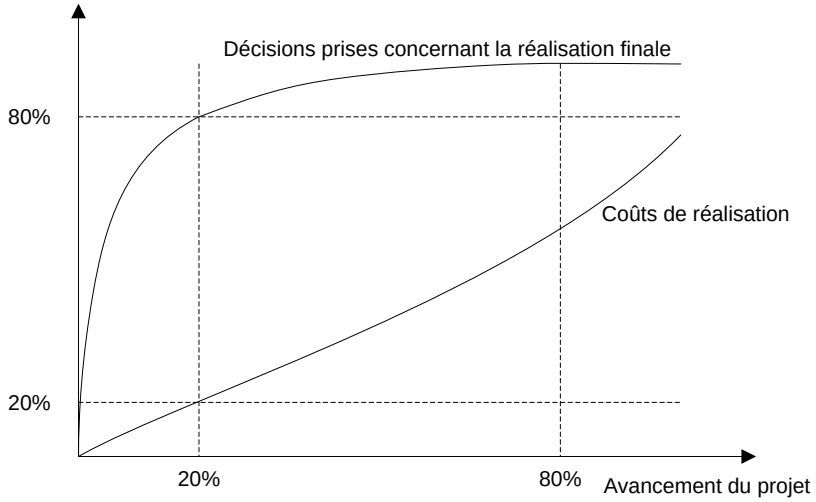

Figure 1.11 – Influence des prises de décisions dans le processus de conception.

Cette figure montre en effet que 80% des décisions concernant la réalisation finale sont généralement prises lorsque les coûts de réalisation sont encore peu élevés, c'est-à-dire de l'ordre de 20% du coût total. Compte tenu de l'influence des décisions prises sur les coûts de développement et des évolutions à considérer pour les prochaines générations de terminaux mobiles, il apparaît essentiel d'assister l'architecte système pour l'aider dans ces choix.

1.3 Problématiques liées au dimensionnement des futurs terminaux mobiles

Le travail de l'architecte système consiste à identifier et caractériser les différentes ressources à utiliser pour réaliser les différentes fonctionnalités devant être supportées par le système. Actuellement, ce travail est mené selon une approche pragmatique et incrémentale basée sur l'expérience préalablement acquise dans la définition des ressources appropriées selon les contraintes imposées [**55**]. L'accroissement du nombre de fonctionnalités à mettre en œuvre et la diversité des ressources proposées pour les réaliser tend à élargir considérablement l'espace de conception de ces systèmes [**55**]. Il s'agit donc d'aider l'architecte système pour qu'il puisse mener à bien ce travail dans le temps qui lui est imparti. Pour cela, il convient de favoriser la définition de modèles de représentation du système lors de cette étape de dimensionnement de manière à lui proposer une aide à la prise de décision. Les approches proposées pour réaliser ce travail de modélisation reposent généralement sur le paradigme en Y formalisé par D. Gajski [**56**] et sur l'approche orientée plate-forme [**51**]. La figure 1.12 permet d'illustrer l'organisation des activités de modélisation à mener en se basant sur ces concepts. Cette organisation vise à clairement séparer la description devant être faite de l'application de celle de la plate-forme.

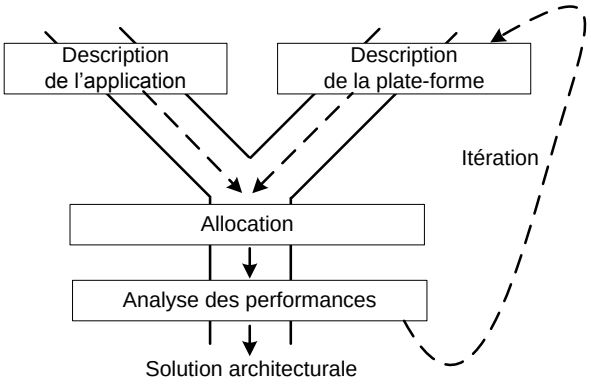

Figure 1.12 – Flot d'exploration basé sur le paradigme en Y et l'approche orientée plate-forme.

La vue purement applicative est utilisée pour décrire ce que doit faire le système sans faire état de tout élément de réalisation. Elle doit ainsi permettre de représenter les différentes fonctionnalités à réaliser, les algorithmes à exécuter ainsi que les différentes relations entre ces fonctions. Des propriétés supplémentaires peuvent être utilisées pour fournir des informations qualitatives et quantitatives sur les différents éléments représentés. Un débit, une latence ou un délai peuvent par exemple permettre de caractériser un élément de l'application. Ces informations sont ensuite utilisées pour correctement dimensionner les ressources nécessaires à la mise en œuvre des différentes fonctionnalités sur le système. La vue orientée plate-forme doit elle permettre de représenter les caractéristiques des différentes ressources pouvant être utilisées pour réaliser le système. Elle doit comprendre la description des différentes ressources de calcul, de mémorisation et de communication proposées. Elle doit aussi permettre de considérer les propriétés non fonctionnelles associées aux différents constituants de la plate-forme. La capacité de traitement et la fréquence de fonctionnement peuvent par exemple être utilisées pour caractériser une ressource de calcul. La capacité de stockage et les temps de lecture et d'écriture d'une information peuvent servir à caractériser une ressource de mémorisation. Une ressource de communication peut quant à elle être caractérisée à partir de propriétés non fonctionnelles comme la taille des données pouvant être échangées et la bande passante proposée. La description du système obtenu en allouant les différents éléments de la vue applicative sur les différents éléments de la plate-forme permet alors à l'architecte système d'analyser d'une part les performances pouvant être obtenues pour un déploiement donné et d'autre part d'estimer les ressources nécessaires à sa réalisation. Á partir de cette description du système, l'architecte système doit être en mesure d'identifier une solution architecturale permettant de répondre aux exigences en termes de temps, de surface, de consommation et de coût devant être satisfaites par le système.

Dès lors, il s'agit d'identifier les différentes problématiques auxquelles l'architecte système va être confronté pour réaliser le travail de dimensionnement des futurs terminaux mobiles. Un premier constat a été fait concernant l'accroissement du nombre de fonctionnalités que les terminaux mobiles vont être amenés à proposer et la grande variété des scénarios de fonctionnement devant être supportés. D'autre part, il a été évoqué que ces systèmes seront réalisés à partir d'un ensemble de ressources matérielles hétérogènes pouvant être programmables et reconfigurables. L'espace de conception de ces systèmes va donc considérablement s'élargir, complexifiant ainsi le travail d'exploration mené par l'architecte système. Pour maîtriser la complexité amenée par ces évolutions, il est nécessaire de proposer aux architectes systèmes une approche correctement formalisée permettant la création de modèles afin de les guider dans leurs choix. Il a été évoqué dans la section 1.2.2 que ces choix devaient être effectués le plus tôt possible dans le processus de

conception de manière à limiter au maximum les risques de surcoût pouvant être induits suites à des mauvaises prises de décisions. Les modèles utilisés pour dimensionner ces systèmes devront ainsi être définis à un niveau d'abstraction plus élevé que celui utilisé lors des phases d'implantation [**57**]. Le niveau d'abstraction considéré pour créer ces modèles devra pour cela permettre de représenter les spécificités liées aux applications devant être supportées par les futurs terminaux mobiles et les caractéristiques des ressources pouvant les exécuter. Il devra d'autre part permettre de limiter la quantité de détails nécessaire à considérer au sein de ces modèles de manière à obtenir des temps de simulation plus courts que ceux nécessaires pour simuler les modèles d'implantation.

Le deuxième constat important se rapporte à la flexibilité devant être apportée à l'exécution des différents standards de communication sur les prochaines générations de terminaux mobiles. L'architecte système va en effet devoir prendre en compte l'influence de l'évolution des caractéristiques de l'application lors du dimensionnement des ressources de la plate-forme. Ce travail de dimensionnement devra donc se faire en lien avec les différents scénarios de fonctionnement du système. Il sera ainsi nécessaire de représenter et d'évaluer l'impact des nouvelles capacités d'adaptation des interfaces de communication par rapport à l'exécution des traitements associés sur les ressources du système. Pour cela, les modèles proposés devront par exemple permettre de représenter les mécanismes de reconfiguration mis en jeu.

Dès lors, il semble nécessaire d'accompagner l'architecte système dans le cadre de ce travail de dimensionnement d'architecture en lui proposant une démarche précise. Cela suppose de définir un ensemble d'étapes clairement identifiées afin de le guider jusqu'à la définition des ressources matérielles et logicielles satisfaisant aux exigences imposées. Á ces étapes doivent être associées un ensemble de modèles reposant sur une sémantique rigoureusement définie. Ces modèles doivent être définis de manière à représenter les propriétés strictement nécessaires et suffisantes du système pour la réalisation du travail d'exploration d'architectures. En effet, il s'agit de proposer à l'architecte système des modèles qui nécessitent des temps de création et de simulation les plus courts possibles afin de lui offrir la possibilité d'analyser un maximum de solutions d'implantation possibles dans le temps qui lui est imparti. La tendance consiste aussi à réaliser ce travail d'exploration le plus tôt possible dans le processus de conception. L'objectif est de prendre les décisions concernant la réalisation finale au début du cycle de conception, lorsque les coûts de réalisation sont encore très faibles. Le paradigme TLM [**58**] a ainsi été introduit au début des années 2000 par des sociétés comme ST Microelectronics, Cadence et ARM afin de favoriser la définition de modèles de représentation du système à un niveau d'abstraction plus élevé que le niveau RTL (*Register Transfer Level*). Le niveau RTL est en effet utilisé lors des phases de développement du matériel. Il propose donc un niveau de détails et des temps de simulation trop importants pour réaliser le travail d'exploration. Le niveau dit transactionnel offre ainsi un compromis intéressant entre la précision des détails nécessaires à la définition d'un modèle et les temps de simulation requis pour pouvoir observer des résultats. Comme cela va être évoqué dans le chapitre 2, ce niveau de représentation offre donc des possibilités intéressantes en vue du dimensionnement d'architectures.

Notre contribution vise donc à tirer parti des bénéfices apportés par le niveau transactionnel et des langages de description comme SystemC pour la création de modèles en vue du dimensionnement d'architectures. Dès lors, il convient de proposer une démarche, des modèles et un ensemble de méthodes afin d'aider l'architecte système dans le cadre du dimensionnement des futurs terminaux mobiles.

Chapitre 2
État de l'art sur les approches de dimensionnement des systèmes embarqués

Sommaire

Pour répondre aux nouveaux besoins applicatifs devant être supportés par les prochaines générations de terminaux mobiles, des évolutions doivent être envisagées dans le cadre du développement de ces systèmes et plus particulièrement au niveau du travail de dimensionnement des ressources matérielles et logicielles requises. L'utilisation de modèles de représentation de ces systèmes est alors considérée comme un moyen pertinent pour permettre à l'architecte système de mener à bien ce travail dans le temps qui lui est imparti. Tout d'abord, le chapitre 2 introduit les différents niveaux d'abstraction couramment utilisés pour modéliser des systèmes embarqués. Nous présentons ensuite les différentes approches qui s'appuient sur ces niveaux d'abstraction en vue du dimensionnement de systèmes. Les différents langages et outils pouvant être employés sont aussi évoqués. Au final, nous introduisons les différentes contributions visées dans ce travail pour améliorer le processus de dimensionnement des terminaux mobiles.

2.1 Niveaux d'abstraction considérés pour la modélisation des systèmes embarqués

L'abstraction permet au concepteur de maîtriser la complexité d'un système en masquant les détails non significatifs à sa description au sein d'un modèle. Un modèle est ainsi généralement défini de manière à représenter une description partielle du système. Les différentes variétés de modèles pouvant être utilisées sont présentées dans [**59**]. Classiquement, ces modèles permettent de représenter trois vues distinctes mais complémentaires [**60**]. La vue structurelle permet de définir l'organisation des constituants du système en faisant apparaître les différentes relations à considérer. La vue comportementale permet de traduire l'aspect dynamique de chaque constituant. Elle exprime par exemple un changement d'état ou une réaction sur des sorties suite à une évolution sur des entrées. La dernière vue dite objet permet de préciser les différents attributs qui permettent de caractériser les différents éléments du modèle. La figure 2.1 présente les différents niveaux d'abstraction considérés tout au long du processus de conception en partant des spécifications pour arriver à la réalisation [**60**]. Ces niveaux sont ici classés en fonction de l'abstraction qui est faite des relations de couplage entre les constituants d'un modèle. Le niveau algorithmique représente ici le niveau le plus abstrait et le niveau RTL le niveau le plus détaillé.

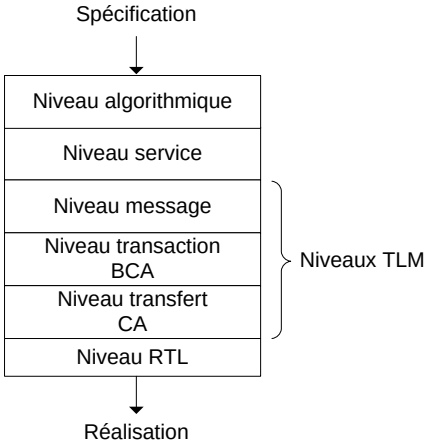

Figure 2.1 – Niveaux d'abstraction du point de vue des communications pour la conception de systèmes.

Le niveau algorithmique permet d'obtenir dans un premier temps une représentation fonctionnelle du système sans aucune notion de temps.

Ensuite, au niveau service, la communication entre les différents constituants est représentée comme une combinaison de requêtes et de services. Un réseau d'interconnexion abstrait permet alors d'assurer le routage des requêtes et des données échangées. Une requête de service peut ainsi être décrite sous la forme d'une primitive de communication. Dans le cas d'une application vidéo, cette primitive de communication peut par exemple être « lire (vidéo) ».

Le niveau message suppose lui que les différents constituants d'un système communiquent via un réseau explicite de canaux de communication. Pour cela, des primitives de haut niveau telles que Send(Mess,Dest) et Receive(Mess,Source) sont alors utilisées pour envoyer et recevoir des messages. Tous les détails qui pourraient être liés à un protocole de communication spécifique sont ainsi masqués.

Au niveau transaction, une topologie d'interconnexion des constituants, des protocoles d'échange de données élémentaires et des synchronisations doivent être définis. Les communications sont ici réalisées par le biais de liens abstraits point à point qui nécessitent l'utilisation de primitives telles que MPutData(Data,Address) et Data:=MGetData(Address) pour l'envoi et la réception d'un paquet de données.

Le niveau transfert permet de rajouter encore plus de précision dans la description d'une communication puisque celle-ci est représentée par un transfert de données élémentaires pendant un cycle d'horloge. Les primitives proposées sont du type Write(Val,Addr) et Val:=Read(Addr). L'adressage est ici explicite puisqu'il peut aussi bien faire référence à une mémoire qu'à un registre.

Le dernier niveau est le niveau dit RTL où les échanges de données sont cette fois-ci réalisés par des fils et des bus physiques. Les primitives de communication sont alors des lectures et des écritures sur des ports et l'échelle de temps est le cycle d'horloge.

Depuis quelques années, nous assistons à l'émergence d'une approche de modélisation dite transactionnelle, aussi appelée TLM pour *Transaction Level Modeling*, dans le cadre de la conception des systèmes embarqués [**53**]. Cette approche s'appuie sur un ensemble de niveaux d'abstraction compris entre le niveau algorithmique et le niveau CA (*Cycle Accurate*). Dans le cadre de cette approche de modélisation, une distinction claire est faite entre la partie traitement et la partie communication d'un système. Le terme transaction est alors employé pour désigner une communication. Dans [**61**], une transaction est définie comme « un transfert de données (communication) ou une synchronisation entre deux modules à un instant déterminé par les spécifications matérielles/logicielles du système ». En ce sens, elle correspond à la notion de message introduite précédemment. Différentes définitions et classifications des niveaux d'abstraction considérés dans l'approche TLM sont disponibles dans la littérature [**58**] [**62**]. Nous nous basons ici sur la vision proposée par STMicroelectronics pour caractériser ces niveaux d'abstraction. La figure 2.2 extraite de [**61**] permet de positionner les différents niveaux d'abstraction de l'approche TLM par rapport aux autres niveaux d'abstraction existant en précisant les niveaux de détail utilisés pour décrire les communications.

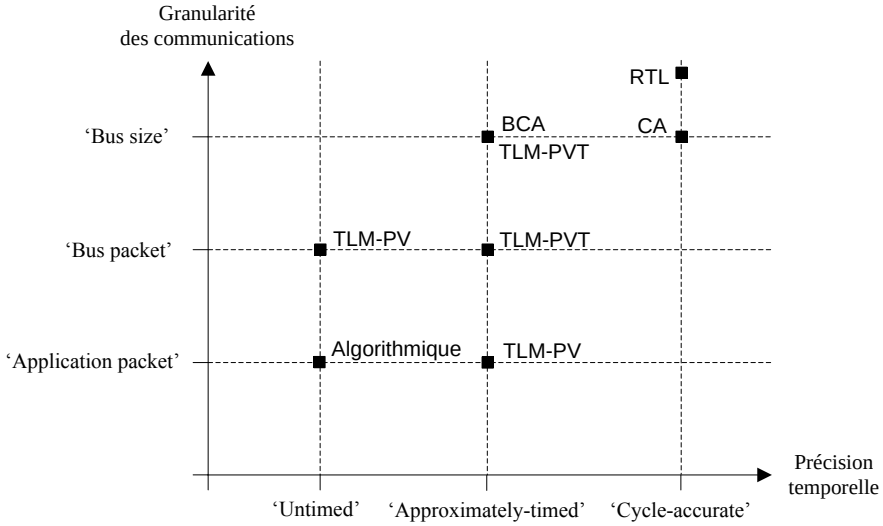

Figure 2.2 – Niveaux d'abstraction et niveaux de détail des communications.

L'axe des abscisses représente les différentes échelles de temps pouvant être employées. Au niveau le plus abstrait, le temps n'est pas pris en compte et les dépendances temporelles sont alors d'ordre causal. Le niveau '*Approximately-timed*' permet lui d'associer des durées

à des opérations de transport des données. Cette durée, définie approximativement, peut par exemple provenir d'une mesure ou d'une estimation. Le niveau de description le plus détaillé dit '*cycle accurate*' permet, comme son nom l'indique, de décrire de façon précise les opérations de transport effectuées à chaque cycle d'horloge.

L'axe des ordonnées est lui utilisé pour évoquer les différents niveaux de granularité qui peuvent être considérés lors d'échanges d'informations entre les différents éléments d'un modèle. Le niveau 'A*pplication packet*' fait référence au niveau de représentation le plus abstrait des communications. En effet, à ce niveau, aucune référence n'est faite sur le protocole de communication mis en jeu pour transférer l'information. Au niveau '*bus packet*', les protocoles de communication sont alors considérés sans pour autant faire référence aux bus de communication pouvant être employés. Les caractéristiques associées au bus de communication sont en effet prises en compte au niveau '*bus size*'.

La figure 2.2 présente ainsi les deux niveaux d'abstraction considérés dans le flot de conception suivi par STMicroelectronics. Le niveau d'abstraction TLM-PV (*Transaction Level Modeling Programmer View*) est utilisé pour décrire une vue purement fonctionnelle d'un système. Á ce niveau d'abstraction, les différents constituants du système et leur couplage sont représentés mais aucune notion de temps n'est alors prise en compte. L'exécution d'un traitement sur une ressource de calcul ou le transfert d'une information sont ici considérés comme instantanés. Le deuxième niveau d'abstraction est appelé TLM-PVT (*Transaction Level Modeling Programmer View plus Timing*). C'est une extension du niveau TLM-PV dans laquelle la modélisation du temps a été ajoutée. L'intérêt porté par les concepteurs sur ces niveaux d'abstraction est lié au fait qu'il nécessite d'une part des temps de simulation beaucoup plus courts que ceux nécessaires avec des modèles définis aux niveaux CA et RTL. En effet, il est estimé dans [**61**] qu'en utilisant une approche de modélisation transactionnelle, il est possible d'atteindre un facteur d'accélération de mille par rapport à un modèle précis au niveau RTL et un facteur cent par rapport à un modèle de niveau CA. Ces niveaux d'abstraction offrent d'autre part la possibilité de définir des modèles permettant de capturer, d'analyser et de vérifier les propriétés d'un système avant la phase de réalisation. Il est en effet évoqué dans [**53**] qu'une description purement fonctionnelle d'un système peut être utilisée dans le cadre des travaux menés lors des étapes de vérification fonctionnelle, de développement du logiciel embarqué et de co-simulation. Une description prenant en compte les propriétés temporelles d'un système peut elle être utilisée pour l'optimisation du logiciel embarqué et l'analyse d'architectures.

Cette approche de modélisation transactionnelle conduit donc à définir un ensemble de niveaux permettant de représenter un système entre le niveau algorithmique et le niveau CA. La section suivante vise à évoquer les différentes approches proposées pour la définition de modèles s'appuyant sur ces niveaux d'abstraction en vue du dimensionnement de systèmes.

2.2 Approches pour le dimensionnement des systèmes embarqués

Nous présentons ci-après différents travaux menés afin de guider l'architecte système dans le cadre du travail de dimensionnement.

Une approche proposée par l'organisme de recherche finlandais VTT pour le dimensionnement des futurs terminaux mobiles est présentée sur la figure 2.3 [**63**].

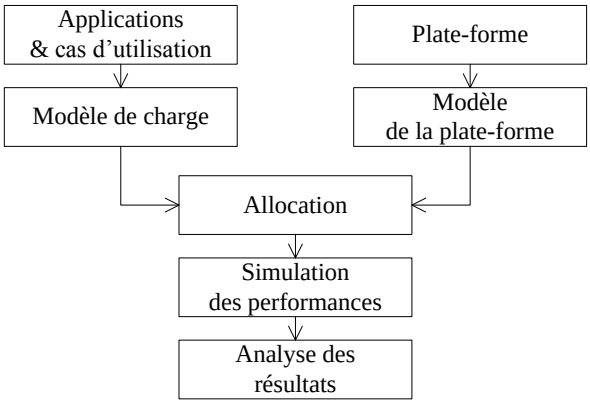

Figure 2.3 – Approche pour le dimensionnement de terminaux mobiles.

Cette approche se base sur le paradigme en Y présenté dans la section 1.3. La distinction est donc ici faite entre la description des applications et la description de la plate-forme sous-jacente. Le modèle des applications est décrit en utilisant le langage UML (*Unified Modeling Language*). Ce modèle représente précisément l'organisation structurelle des différentes applications devant être réalisées par le système. Les charges de calcul, de communication et de mémorisation associées à ces applications sont elles décrites de façon abstraite. Elles peuvent être obtenues analytiquement, à partir de mesures, ou encore en extrayant des informations du code source des applications comme cela est présenté dans [**64**]. La plate-forme est elle modélisée au niveau 'cycle approximé' en utilisant le langage SystemC. Le modèle des applications peut être alloué sur le modèle de la plate-forme et la simulation du modèle obtenu permet d'analyser les performances pouvant être atteintes par le système. Il est en effet possible d'observer les taux d'utilisation des ressources de calcul, de mémorisation et de communication. L'analyse de ces informations peut par exemple permettre à l'architecte système de constater que la fréquence d'un processeur peut être diminuée de manière à optimiser la consommation du système. Ce type d'approche illustre la volonté actuelle d'élever le niveau de représentation des systèmes en vue de leur dimensionnement. En limitant la quantité de détails considérés au sein de ces modèles, il est en effet possible de réduire les temps de simulation nécessaires à leur exécution. L'architecte système a ainsi la possibilité de pouvoir valider plus rapidement ses choix concernant la réalisation de son système. Cependant, il est évoqué que des efforts doivent encore être faits de manière à diminuer les temps de simulation nécessaires à l'exécution de leurs modèles définis au niveau 'cycle approximé'. Il semble aussi essentiel de pouvoir observer les performances obtenues par un système en fonction des différents scénarios de fonctionnement à évaluer. Pour pouvoir réaliser cette analyse le plus rapidement possible, les modèles utilisés sont paramétrables. Cela permet de limiter l'effort de modélisation nécessaire à l'évaluation de différents cas d'utilisation.

Nokia présente dans [**65**] l'approche utilisée pour dimensionner leurs terminaux mobiles. L'originalité de cette approche réside dans le fait que ce travail de dimensionnement est réalisé en utilisant des modèles exécutables de l'environnement du système. Ces modèles permettent de capturer et d'évaluer les performances pouvant être obtenues en fonction des différents cas d'utilisation identifiés pour chaque système. Les cas d'utilisation sont d'abord décrits à l'aide de diagrammes de séquence. Un diagramme de séquence permet d'obtenir une description graphique des interactions entre l'environnement et le système. L'outil CoFluent Studio est ensuite utilisé pour capturer et simuler les modèles du système et de son environnement. Pour évaluer facilement différents scénarios de fonctionnement, le modèle de l'environnement est paramétré par un fichier scénario. Ce fichier contient les différentes

requêtes de services devant être envoyées au système. Le modèle de l'environnement lit alors les informations contenues dans ce fichier puis envoie au cours de la simulation les différentes requêtes de services indiquées. Le dimensionnement du système peut ainsi se faire en lien avec les différents cas d'utilisation envisagés.

Au sein de l'IMEC (*Interuniversity MicroElectronics Centre*), les travaux menés visent aussi à favoriser la définition de modèles exécutables pour le dimensionnement de plate-forme de type SDR. Ils proposent dans [**66**] d'analyser les performances pouvant être obtenues avec une gestion en temps réel des ressources d'une plate-forme de type ADRES pour la réalisation des traitements associés à la couche physique du standard de communication 802.11n. Les modèles ici utilisés sont définis à différents niveaux d'abstraction. La description des couches hautes de ce protocole de communication et des évolutions des conditions de transmission et de réception sur l'interface radio est faite à un niveau de granularité plus élevé que pour la couche physique. La couche MAC (*Medium Access Control*) est par exemple modélisée pour générer les paquets de données devant être traités par la couche physique. La communication entre le modèle de la couche MAC et le modèle de la couche physique se fait au niveau 'Application Packet'. Les instants de production correspondent eux aux instants critiques d'envois de paquets de données entre la couche MAC et la couche physique. Pour dimensionner les ressources nécessaires à l'exécution des traitements de la couche physique du standard 802.11n, un niveau de granularité plus fin est ensuite évoqué pour représenter et analyser l'exécution des traitements associés sur les ressources de la plate-forme.

Les équipes de recherche de l'université de Tübingen et de Karlsruhe indiquent eux dans [**67**], l'importance et la difficulté qu'il peut y avoir à évaluer les propriétés non fonctionnelles d'un système lors de la phase de dimensionnement. L'approche proposée permet de modéliser, simuler et observer les performances temporelles atteintes par le système ainsi que la consommation induite par l'activité des ressources de la plate-forme. L'expérimentation proposée pour illustrer leur approche concerne un décodeur JPEG nécessitant de réaliser quatre fonctions de traitement. Chaque fonction est réalisée sur un processeur de type PowerPC. Les moyens d'observation des résultats proposés permettent d'évaluer si la contrainte de consommation appliquée est bien respectée. Des modèles de niveau TLM-PVT décrits en SystemC sont ici utilisés pour évaluer ces propriétés. Dans [**68**], le CECS (*Center for Embedded Computer Systems*) montre l'intérêt qu'il peut y avoir à utiliser ce niveau de représentation pour, par exemple, dimensionner efficacement et rapidement un bus de communication au sein d'un système sur puce. Dans le cadre de leur expérimentation avec un bus de communication de type AMBA (*Advanced Microcontroller Bus Architecture*), un facteur moyen d'accélération de 55% des temps de simulation est obtenu avec un modèle de niveau TLM-PVT par rapport à un modèle de représentation défini au niveau BCA. Ils indiquent de plus qu'ils arrivent à obtenir le même degré de précision pour les résultats obtenus. Ce type de travaux illustre la tendance actuelle qui consiste à élever le niveau d'abstraction utilisé pour modéliser les systèmes dans le cadre de leur dimensionnement de manière à réduire les temps de simulation nécessaires.

Certains travaux se concentrent sur le dimensionnement d'une catégorie de ressources. Par exemple, dans [**69**], des industriels comme Infineon Technologies et Cadence Design Systems proposent une approche reposant sur l'utilisation de modèles décrits en SystemC au niveau CA pour dimensionner les ressources de mémorisation. Cette approche permet, à partir de critères comme la latence, l'arbitrage mémoire et la synchronisation, d'évaluer les performances obtenues en utilisant différents types de mémoires (locales ou partagées). Le cas d'étude proposé est la chaîne de transmission du standard 802.11n. Dans [**70**], la société Qualcomm s'intéresse elle au dimensionnement des ressources de communication pour un

terminal mobile 4G. L'objectif est de réaliser ce travail en tenant compte à la fois des contraintes en termes de latence liées à l'application et de la consommation induite par l'utilisation des ressources de communication.

On constate dans cette partie la pluralité des approches et des niveaux d'abstraction utilisés pour définir des modèles dans le cadre du dimensionnement de systèmes. Certains travaux de recherche comme ceux menés en France au sein du projet MoPCom [**71**] tentent de formaliser une approche [**72**] pour la création de modèles à différents niveaux d'abstraction (TLM-PV, TLM-PVT, CC pour *Cycle Callable*). L'objectif est de faciliter l'interopérabilité des modèles utilisés pour dimensionner des systèmes à différents niveaux d'abstraction. Le travail d'exploration doit ainsi pouvoir être mené par raffinements successifs des modèles utilisés en partant du niveau d'abstraction le plus élevé pour aboutir à une analyse fine menant à l'implantation du système. Les modèles proposés sont créés à l'aide de profils UML tel que MARTE [**73**] définis spécifiquement pour réaliser ce travail d'analyse de performance et d'exploration d'architectures de systèmes embarqués temps réel. Des informations plus détaillées sur ce profil seront données dans la partie suivante. Une transformation automatique des modèles UML en code SystemC est envisagée afin de permettre leur simulation.

D'autres travaux visent à faciliter la description des nouvelles caractéristiques des plates-formes envisagées pour développer les prochaines générations de systèmes embarqués. On peut citer par exemple les travaux menés par Guy Gogniat au sein du Lab-STICC (*Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance*). Dans [**74**], l'objectif est de faciliter l'utilisation de ressources reconfigurables dynamiquement dans le cadre de la conception de systèmes. Des modèles sont utilisés pour représenter l'application, la plate-forme et l'allocation de plusieurs fonctions de l'application sur une ressource reconfigurable de la plate-forme. Dans le cadre de cette expérimentation, un FPGA (*Field-Programmable Gate Array*) Xilinx est utilisé pour implanter le système. Le système doit supporter une application de suivi d'objets en mouvement sur une vidéo. Pour cela, neuf traitements doivent être effectués. Un co-processeur reconfigurable dynamiquement permet de réaliser trois de ces traitements à différents instants. Des techniques de modélisation sont ici proposées pour capturer les nouveaux services permettant de réaliser en cours de fonctionnement les processus de reconfiguration nécessaires. Cette description en UML est ensuite le point d'entrée des outils Xilinx pour générer automatiquement le code nécessaire à l'implantation du système considéré sur cette cible. Ce type d'approche illustre la volonté actuelle d'utiliser des modèles pour faciliter l'utilisation de ressources reconfigurables dynamiquement dans le cadre de la conception de systèmes embarqués. Par ailleurs, il est évoqué dans cet article la nécessité de pouvoir faire le lien entre les possibilités offertes par ces ressources et les besoins en termes d'adaptabilité d'une application.

2.3 Langages et outils

La présentation des différentes approches proposées actuellement pour le dimensionnement de systèmes embarqués a permis d'évoquer un certains nombre de langages qui tendent à être de plus en plus utilisés pour définir des modèles dans le cadre de ce travail.

UML [**75**] par exemple est un langage de description graphique qui a été normalisé par l'OMG (*Object Management Group*) [**76**] en 1997. Initialement, il a été défini pour faciliter le travail de conception de systèmes logiciels à haut niveau. C'est un langage formel qui permet en fait de modéliser un large éventail de systèmes relevant de différents domaines d'application. Ce langage est organisé autour de diagrammes qui sont répartis en trois

catégories : les diagrammes structuraux, comportementaux et d'interactions. Ces diagrammes permettent de représenter les différentes spécificités d'un système. La notion de profil a ensuite été introduite afin d'offrir aux utilisateurs de ce langage une spécialisation d'UML pour un domaine particulier. Des profils tels que SysML [**77**] et MARTE [**73**] ou encore UML for SystemC [**78**] ont par exemple été définis pour faciliter l'utilisation de modèles UML dans le cadre de la conception de systèmes embarqués temps réel. Le profil SysML a été proposé pour faciliter la spécification, l'analyse, la vérification et la validation de systèmes embarqués. Le standard MARTE propose lui un ensemble d'éléments de notation qui facilite l'utilisation d'UML pour la modélisation et l'analyse des systèmes embarqués temps réel. Les modèles décrits à partir du profil MARTE peuvent ainsi être annotés avec des propriétés non fonctionnelles de façon à pouvoir caractériser les contraintes qui sont associées aux éléments qu'ils représentent. Il offre par ailleurs la possibilité de pouvoir représenter l'application et son placement sur la plate-forme. Dans [**79**] sont présentés différents conseils pour utiliser ces deux profils dans le cadre d'un travail d'exploration de l'espace de conception d'un système embarqué temps réel. UML for SystemC a lui été proposé de manière à faciliter la génération de code SystemC à partir d'une description d'un système faite en UML. L'intérêt est d'obtenir une description exécutable du système à partir d'un modèle de représentation décrit en UML. Des projets comme MARTES [**80**] ont pour cela été menés pour faire le lien entre les modèles de représentation basés sur UML et les modèles SystemC qui permettent leur exécution tout au long du processus de conception.

Le langage SystemC [**81**] [**82**] a lui été défini par l'OSCI (*Open SystemC Initiative*) [**83**] qui rassemble une multitude de sociétés et de laboratoires de recherche comme Cadence, Forte, Synopsys, Philips,… Depuis décembre 2005, SystemC est standardisé auprès de l'IEEE sous le nom de IEEE 1666™-2005. Il est basé sur une extension du langage de programmation C++ à laquelle ont été ajoutées différentes bibliothèques permettant de représenter les différents aspects d'un système matériel comme sa structure, son comportement et ses caractéristiques temporelles. Il est alors possible pour le concepteur de pouvoir utiliser un même langage pour la modélisation des parties matérielles et logicielles d'un système à différents niveaux d'abstraction. Une librairie intitulée TLM 2.0 [**84**] a par ailleurs été définie par l'OSCI pour faciliter la création de modèles de niveau transactionnel avec le langage SystemC. D'autres langages tels que SpecC [**85**] issu des travaux menés par le professeur Gajski à l'université de Californie à Irvine ou encore SystemVerilog [**86**] peuvent aussi être employés pour la définition de modèles de niveau transactionnel.

Par ailleurs, un grand nombre d'outils d'aide à la conception de systèmes ont été proposés au cours des dernières années. Un état de l'art de ces différents outils a été réalisé à l'université de Californie située à Berkeley [**87**]. Ils sont classés en fonction des activités définies dans le paradigme en Y qu'ils supportent. Neuf outils, parmi les quatre-vingt-dix référencés, proposent au concepteur un flot unifié permettant de réaliser la description fonctionnelle du système sans aucune considération technologique, puis la description des ressources de la plate-forme et enfin l'allocation des différentes fonctions sur les ressources de la plate-forme. On peut citer les outils CoFluent Studio [**88**], MLDesigner [**89**] et System Studio [**90**] qui sont actuellement commercialisés par des industriels et Artemis [**91**], Mescal [**92**] et Metropolis [**93**] qui sont proposés dans le domaine académique. L'outil CoFluent Studio a par exemple été développé suite aux travaux menés par le professeur Calvez à l'université de Nantes. Il est actuellement commercialisé par la société CoFluent Design. Cet outil permet de capturer et de simuler les modèles utilisés dans le cadre de l'approche MCSE (*Méthodologie pour la Conception des Systèmes Electroniques*) [**94**]. Il offre un environnement de saisie principalement graphique qui permet de décrire des modèles de

niveau transactionnel conformément à l'approche en Y. Á partir de cette description, un modèle SystemC peut ensuite être généré puis simulé.

2.4 Contributions visées pour le dimensionnement des futurs terminaux mobiles

Le travail de dimensionnement mené par un architecte système a pour but d'identifier une solution de réalisation permettant de satisfaire les exigences fonctionnelles et non fonctionnelles associées à ce système. Les propositions faites actuellement visent donc à favoriser la création de modèles permettant une exploration de l'espace de conception. Ces modèles doivent pour cela capturer les propriétés de l'application et de la plate-forme afin d'analyser les performances pouvant être obtenues pour différentes solutions d'implantation. Nous avons constaté l'intérêt porté pour le langage UML et plus particulièrement sur des profils spécifiques comme MARTE et SysML pour la description de modèles dans le cadre de ce travail [**79**]. Cependant le manque d'usage de ces langages par les architectes systèmes et le manque de modèles de référence tendent à limiter l'utilisation de ces profils.

Les travaux évoqués précédemment permettent aussi de constater qu'il semble essentiel de pouvoir élever le niveau d'abstraction considéré pour représenter un système. L'accroissement du niveau d'abstraction porte aussi bien sur les propriétés de l'application que sur celles des ressources de la plate-forme. Certains travaux sont proposés pour prendre en compte ces propriétés à différents niveaux d'abstraction. D'autres visent à diminuer les temps de simulation nécessaires à l'exécution des modèles à chaque niveau d'abstraction. Le niveau transactionnel avec des propriétés temporelles a été identifié comme offrant un bon compromis précision/temps de simulation. Cependant, peu de démarches clairement définies visent à favoriser la création de modèles de niveau transactionnel pour le dimensionnement de systèmes.

Notre contribution, qui sera développée dans la partie suivante, porte donc sur la proposition d'une démarche visant à accompagner l'architecte système dans l'exploration de l'espace de conception pour les futurs terminaux mobiles. Cette démarche repose sur **un ensemble d'étapes pour la création de modèles de niveau transactionnel intégrant les propriétés fonctionnelles et non fonctionnelles du système à dimensionner**. Un modèle d'exécution générique est proposé **afin de réduire les temps de création de ces modèles**. Une technique d'observation des propriétés non fonctionnelles conduit à **une réduction significative des temps de simulation nécessaires**. Par la suite, nous validons notre approche en utilisant l'outil CoFluent Studio basé sur SystemC pour la simulation.

Dans le chapitre 4, différentes techniques de modélisation sont également proposées de manière à faciliter la description au niveau transactionnel des différentes fonctionnalités devant être supportées par les futurs terminaux mobiles. Il a en effet été évoqué précédemment que ces systèmes vont devoir proposer d'une part, une offre diversifiée d'applications à l'utilisateur et d'autre part, une gamme élargie de standards de communication. Une technique de modélisation a ainsi été proposée afin de réduire l'effort de modélisation nécessaire à la description de ces systèmes de radiocommunication multiservices. Ces systèmes seront aussi dotés d'une nouvelle fonctionnalité en charge de supporter les différents mécanismes permettant d'adapter les interfaces de communication en fonction des besoins applicatifs de l'utilisateur. Une technique de modélisation est ainsi proposée de manière à pouvoir représenter au niveau transactionnel ces différents mécanismes. Une technique de modélisation de l'environnement du système est aussi

proposée afin de pouvoir dimensionner les ressources requises en tenant compte des différents scénarios de fonctionnement devant être supportés.

Chapitre 3
Contributions pour la prise en compte des propriétés non fonctionnelles au sein de modèles de niveau transactionnel

Sommaire

Il a été constaté précédemment la nécessité de disposer de modèles permettant une exploration architecturale efficace dans le cadre de la conception des futurs terminaux mobiles. Les critères d'efficacité portent sur la rapidité à créer et à simuler les modèles utilisés. Nous présentons donc ici notre proposition d'une approche et de techniques en vue de la création de modèles de niveau transactionnel pour l'exploration d'architectures. Les contributions portent d'une part sur une technique d'abstraction des propriétés non fonctionnelles en vue de diminuer les temps de simulation nécessaires à l'exécution de modèles. D'autre part, nous proposons un modèle d'exécution générique visant à faciliter la création des modèles. L'expérimentation et la validation de notre approche est effectuée en utilisant l'outil CoFluent Studio. La deuxième partie de ce chapitre vise à illustrer la mise en œuvre de notre approche sur l'exemple d'un algorithme de transformée de Fourier rapide devant être implanté sur une architecture matérielle. Les résultats obtenus permettent d'évaluer la puissance de calcul requise pour exécuter cet algorithme en tenant compte des propriétés liées à l'application et à l'architecture considérée. La dernière partie présente la démarche proposée afin d'assister l'architecte système dans le processus d'exploration d'architectures.

3.1 Proposition de techniques de modélisation au niveau transactionnel pour l'exploration d'architectures

3.1.1 Objectifs de l'approche

Le travail d'exploration architecturale consiste à analyser les différentes possibilités de déploiement d'une application sur une plate-forme et à fixer les propriétés des différents constituants de l'architecture. Ce travail s'effectue en évaluant les performances pouvant être obtenues selon ces différentes propriétés et également selon les différentes distributions possibles. Dans le cadre de nos travaux, nous cherchons à proposer une approche permettant de capturer au sein de modèles les propriétés nécessaires à l'évaluation de ces performances. La figure 3.1 présente un exemple de déploiement d'une application sur une plate-forme qu'il s'agirait d'évaluer dans le cadre du travail d'exploration architecturale.

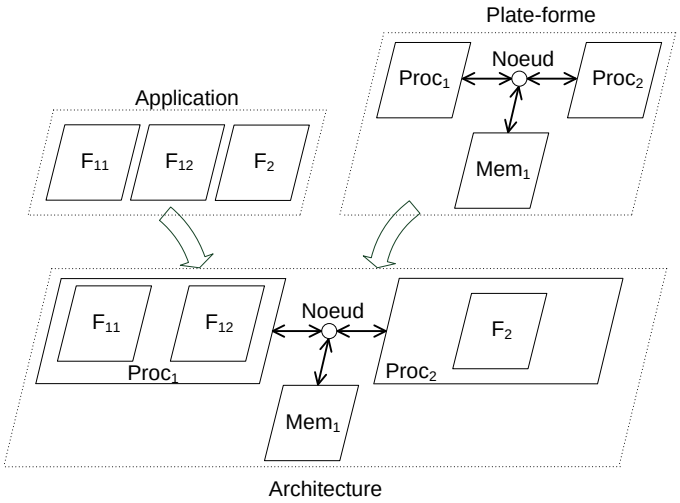

Figure 3.1 – Exemple de déploiement d'une application sur une plate-forme.

L'application est ici composée de trois fonctions désignées par les termes F_{11}, F_{12} et F_2. La plate-forme regroupe les ressources pouvant être utilisées pour réaliser le système. Elle est ici composée de deux ressources de traitement $Proc_1$ et $Proc_2$, d'une ressource de communication Nœud et d'une ressource de mémorisation Mem_1. La partie basse de la figure 3.1 présente un exemple de déploiement de l'application sur les ressources de la plate-forme. Les fonctions F_{11} et F_{12} sont exécutées sur la ressource $Proc_1$ et la fonction F_2 sur la ressource $Proc_2$. Le dimensionnement de cette architecture se fait en considérant les propriétés dites fonctionnelles et celles dites non fonctionnelles. Les propriétés fonctionnelles se rapportent à la complexité des algorithmes à exécuter, à la taille des données échangées entre les fonctions ou encore aux temps de réponse à respecter. Elles sont donc indépendantes de toutes considérations technologiques. Nous désignons ici par le terme propriétés non fonctionnelles les caractéristiques associées aux ressources de la plate-forme ainsi que les propriétés émergentes qui découlent de l'exécution de l'application sur la plate-forme. Une liste non exhaustive des différentes propriétés non fonctionnelles pouvant être associées à une architecture est présentée dans le tableau 3.1.

<table>
<tr><td colspan="2">Catégories de ressources</td><td>Ressource de traitement</td><td>Ressource de mémorisation</td><td>Ressource de communication</td></tr>
<tr><td rowspan="6">Propriétés non fonctionnelles</td><td rowspan="2">Caractéristiques de la plate-forme</td><td>Fréquence d’horloge</td><td>Temps d’accès</td><td>Bande passante</td></tr>
<tr><td>Capacité de traitement</td><td>Capacité de mémorisation</td><td></td></tr>
<tr><td rowspan="2">Propriétés émergentes locales</td><td>Consommation</td><td>Consommation</td><td>Consommation</td></tr>
<tr><td>Taux d’utilisation</td><td>Taux d’occupation</td><td>Débit</td></tr>
<tr><td rowspan="3">Propriétés émergentes globales</td><td colspan="3">Coût</td></tr>
<tr><td colspan="3">Consommation</td></tr>
<tr><td colspan="3">Surface</td></tr>
</table>

Tableau 3.1 – Propriétés non fonctionnelles associées à une architecture.

Le travail d’exploration architecturale consiste à analyser l’impact de ces propriétés sur les performances de l’architecture. Il s’agit de vérifier qu’une configuration donnée des ressources de la plate-forme permet de répondre d’une part aux exigences applicatives exprimées à travers les propriétés fonctionnelles. D’autre part, il est nécessaire de s’assurer que les propriétés émergentes respectent bien les contraintes en termes de coût, de consommation et de surface qui s’appliquent sur le système considéré.

Il a été constaté précédemment que l’utilisation de modèles, dans le cadre de ce travail, offre un moyen pertinent pour l’évaluation des performances de différentes solutions architecturales. Les propositions faites actuellement dans les travaux de recherche considérés reposent généralement sur la définition de modèles distincts de l’application et de la plate-forme, et ce selon différents niveaux d’abstraction possibles. L’originalité de nos travaux se rapporte à la volonté d’exprimer au sein d’un modèle commun de niveau transactionnel les différentes propriétés nécessaires au dimensionnement d’une architecture. Il s’agit en effet de pouvoir représenter à la fois les propriétés fonctionnelles liées à l’application et les propriétés non fonctionnelles liées à la plate-forme et à l’architecture analysée. Dans la partie suivante, nous présentons la notation proposée pour représenter au sein d’un même modèle les propriétés requises pour l’évaluation des performances d’une solution architecturale.

3.1.2 Notation utilisée

La notation retenue s’inspire du formalisme défini dans la méthodologie MCSE [**95**]. La figure 3.2 introduit les différents éléments graphiques qui permettent de représenter les différentes propriétés d’un système à modéliser ainsi que les entités qui composent son environnement. Sur cette figure, le système est en interaction avec deux entités pouvant être décrites en utilisant les notations introduites par la suite.

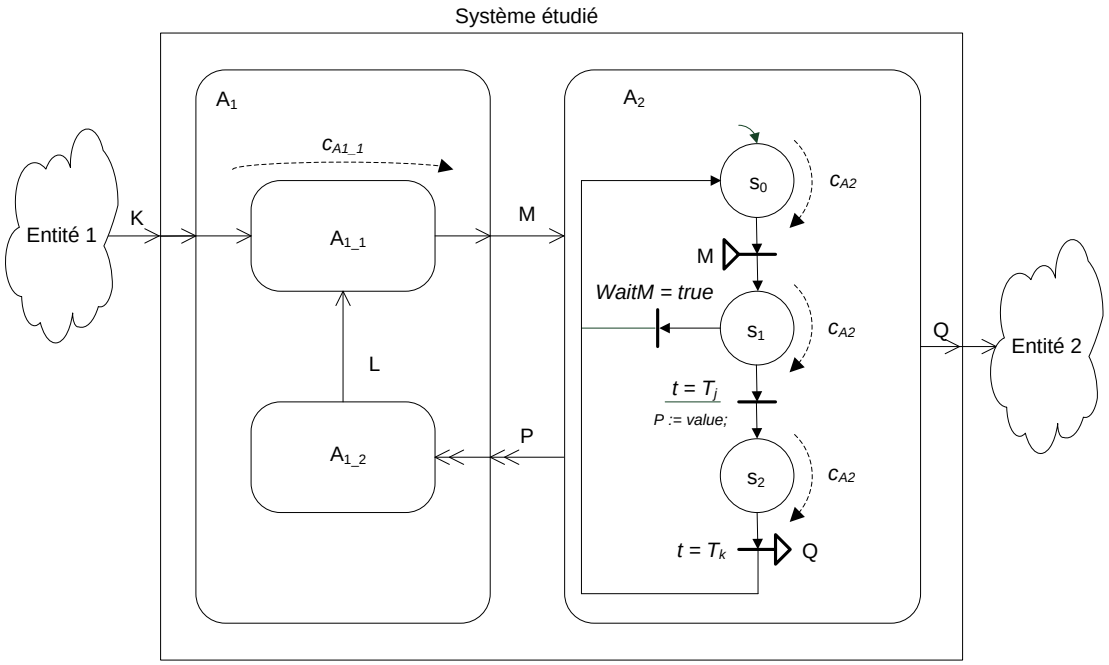

Figure 3.2 – Notation utilisée pour la description des propriétés d'un système.

La vue structurelle du système est décrite sous la forme d'un diagramme d'activités. Ce type de représentation favorise la structuration du comportement global du système en un ensemble de comportements partiels. Chaque activité considérée au sein du système étudié est ici représentée par un rectangle avec des coins arrondis. La notion de hiérarchie entre activités peut être prise en compte. Ainsi, sur la figure 3.2, l'activité A_1 est raffinée à l'aide de deux activités A_{1_1} et A_{1_2}. Les interactions entre les différentes activités sont aussi exprimées. Elles se représentent graphiquement par l'emploi d'une simple ou double flèche. Une simple flèche fait référence à une relation de type événementiel. Un événement auquel est associé un contenu spécifique est alors appelé information. La double flèche représente quant à elle le partage entre plusieurs activités d'une donnée sans aucune relation d'ordre.

La vue comportementale du système est représentée à l'aide d'une notation spécifique proche du langage de modélisation graphique de systèmes réactifs StateChart [**96**]. L'activité A_2 fait apparaître les différents éléments graphiques qui sont utilisés pour traduire le comportement d'une activité. Chaque rond représente un état dans lequel peut se trouver une activité. Le passage d'un état à un autre est lié à une condition de transition. Différentes conditions de passage d'un état à un autre peuvent être utilisées. Elles peuvent se rapporter à une condition d'attente d'une information comme c'est le cas entre l'état S_0 et S_1. Le passage de l'état S_0 à S_1 se fait en effet suite à l'arrivée d'une information M émise par A_{1_1}. Le protocole de type rendez-vous est ici utilisé pour caractériser les communications entre activités. Dans le cadre de ce protocole, aucun stockage ne peut alors être effectué au sein des relations. L'activité A_2 reste ainsi bloquée dans l'état S_0 tant qu'elle n'a pas reçu l'information M devant être transmise par l'activité A_{1_1}. L'activité A_{1_1} passe elle dans l'état suivant dès lors que l'information M a été lue par l'activité A_2. Une information M se rapporte en ce sens à la notion de transaction introduite dans la section 2.1. Elle correspond en effet à un transfert de données dans le cadre d'une communication bloquante où la condition temporelle de production peut être considérée. Nous utiliserons donc par la suite le terme transaction pour ce type de relation entre activités. Une condition de transition entre deux états peut aussi se rapporter à une variable interne. Cette variable peut être de type booléen comme cela est illustré sur la figure 3.2 avec la condition WaitM entre l'état S_1 et S_0. Elle peut aussi se rapporter à une information temporelle. Ce type de condition peut par

exemple être employé pour exprimer une durée pendant laquelle une activité doit rester dans un état comme c'est le cas sur l'exemple de la figure 3.2 pour les états S_1 et S_2. Au sein de modèles de niveau transactionnel, ces propriétés temporelles sont utilisées pour représenter les caractéristiques des différents constituants d'une architecture à analyser. Une action peut aussi être associée à une condition. Le passage de l'état S_1 à l'état S_2 entraîne par exemple une mise à jour de la valeur de la variable P qui est partagée par les activités A_2 et A_{1_2}. Une condition de transition peut de plus entraîner la production d'une information. Une information Q est par exemple ici produite dès lors que la condition temporelle $t = T_k$ est vérifiée.

Une troisième vue permet d'exprimer les contraintes s'appliquant sur les différents constituants du modèle. Elles peuvent par exemple se rapporter à une contrainte temporelle, à une consommation d'énergie à respecter ou encore à une puissance de calcul à ne pas dépasser. Ces contraintes sont représentées sous la forme d'annotations sur les deux autres vues. Une contrainte peut ainsi être associée à un état ou à une activité. Une flèche avec un trait discontinu est alors utilisée pour les exprimer. Sur la figure 3.2, C_{A1_1} se rapporte à une contrainte devant être respectée par l'activité A_{1_1}. C_{A2} représente elle une contrainte qui s'applique sur les états S_0, S_1 et S_2 de l'activité A_2.

3.1.3 Approche considérée

L'approche que nous proposons vise donc à faciliter la définition d'un modèle de représentation d'une architecture en se basant sur le formalisme présenté dans la partie précédente. La figure 3.3 permet d'illustrer notre approche.

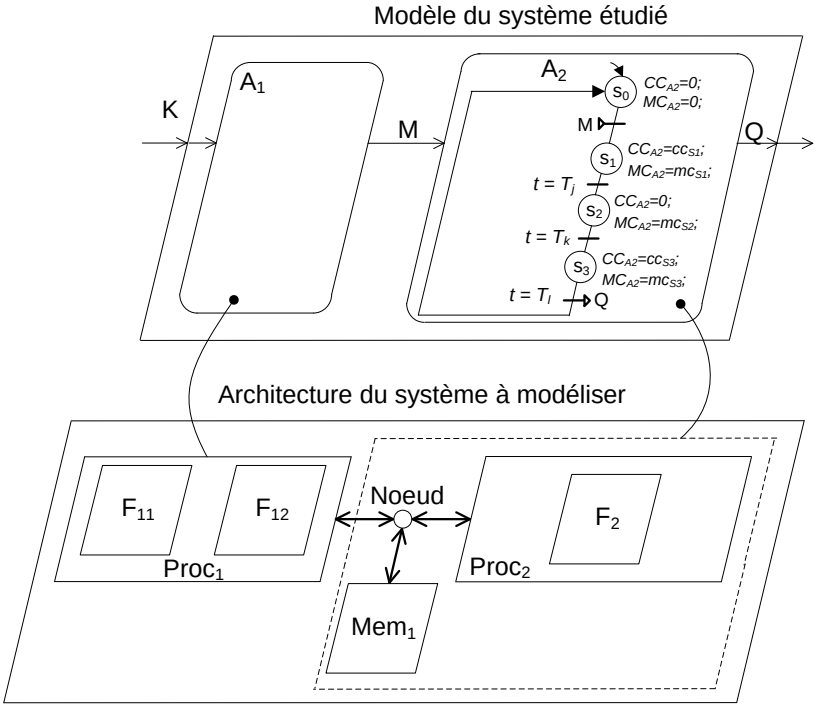

Figure 3.3 – Approche proposée pour la création de modèles dans le cadre du travail d'exploration d'architectures.

On retrouve sur la partie basse de la figure 3.3, l'exemple de l'architecture présentée précédemment sur la figure 3.1. La partie haute de la figure 3.3 illustre la représentation considérée pour décrire les propriétés nécessaires à l'analyse des performances obtenues

avec cette architecture. Les activités A_1 et A_2 sont respectivement définies pour analyser les performances obtenues dans le cas où les fonctions F_{11} et F_{12} sont allouées sur la ressource $Proc_1$ et la fonction F_2 sur la ressource $Proc_2$. La vue comportementale de l'activité A_2 est précisée sur la figure 3.3. Cette description permet de représenter d'une part la réception et la production des données associées aux traitements de la fonction F_2. Les différents états définis permettent quant à eux de traduire l'utilisation qui est faite de la ressource de calcul $Proc_2$ et de la ressource de mémorisation Mem_1 pour exécuter la fonction F_2. Sur la figure 3.3, la variable CC_{A2} décrit la puissance de calcul mise en jeu pour exécuter les traitements de la fonction F_2 sur la ressource $Proc_2$. La variable MC_{A2} permet elle de représenter l'évolution de la quantité d'informations stockée par la ressource de mémorisation Mem_1. La description de l'activité des ressources $Proc_2$ et Mem_1 pour l'exécution de la fonction F_2 se fait alors de la manière suivante. L'état initial S_0 représente un état d'attente d'une transaction M. Dans cet état, la ressource de mémorisation Mem_1 est vide et aucun traitement n'est effectué. Dans ce cas, les variables CC_{A2} et MC_{A2} prennent la valeur zéro. La réception d'une transaction M entraîne ensuite le stockage d'une quantité d'informations mc_{S1} et la nécessité de mettre en œuvre une puissance de calcul de cc_{S1} opérations par seconde. La variable MC_{A2} est mise à jour en utilisant l'information concernant la quantité de données transmises qui est indiquée dans la transaction M. La puissance de calcul CC_{A2} mise en œuvre pour réaliser le traitement est elle obtenue à partir d'une expression analytique. Cette expression prend en compte la complexité de l'algorithme à exécuter, la contrainte de temps à respecter et la capacité de traitement offerte par la ressource utilisée. La condition $t=T_j$ indique que le temps nécessaire à la réalisation des traitements par la ressource $Proc_2$ est écoulé. L'état S_2 représente ensuite un nouvel état pendant lequel la ressource $Proc_2$ est inactive. CC_{A2} prend donc la valeur zéro. MC_{A2} prend elle la valeur mc_{S2} qui correspond à la nouvelle quantité d'information stockée en mémoire après l'exécution du traitement. Lorsque la condition $t=T_k$ est vérifiée, l'activité A_2 passe dans l'état S_3. Les nouvelles valeurs cc_{S3} et mc_{S3} devant être affectées à CC_{A2} et MC_{A2} sont alors calculées. La condition $t=T_l$ indique la fin de l'exécution d'un deuxième traitement devant être réalisé. Une transaction Q correspondant au résultat obtenu est alors produite en sortie. L'activité A_2 repasse enfin dans l'état S_0 en attente d'une transaction M.

L'approche de modélisation proposée sert à représenter les propriétés qui caractérisent les différents constituants d'une architecture. Les modèles décrits permettent ensuite d'observer les performances obtenues lors de l'exécution de l'application sur les ressources de la plate-forme. Afin de favoriser l'utilisation de ces modèles, nous avons constaté qu'il était nécessaire d'adresser deux problématiques importantes. La première se rapporte à la difficulté qu'il peut y avoir à décrire au sein de modèles les différentes propriétés à analyser d'un système. Cette difficulté porte sur la diversité des applications et des architectures à modéliser. Pour maîtriser cette complexité, nous proposons donc un modèle de référence permettant, par simple paramétrage, d'évaluer les performances obtenues avec différentes architectures. Ce modèle d'exécution générique paramétrable est présenté dans la partie 3.1.5. La deuxième problématique se rapporte au temps nécessaire à l'extraction d'informations issues du modèle. Ces informations sont généralement obtenues par simulation. Les temps nécessaires à la simulation des modèles proposés doivent donc être suffisamment courts pour pouvoir analyser efficacement l'espace de conception identifié. Nous présentons donc dans la partie suivante la proposition qui a été faite pour diminuer les temps nécessaires à la simulation des modèles créés selon l'approche considérée.

3.1.4 Technique proposée pour le calcul des propriétés non fonctionnelles

Le principe considéré ici consiste en une abstraction des propriétés non fonctionnelles via un accroissement du niveau de granularité au sein de modèles de niveau transactionnel. Il s'apparente au principe d'abstraction fait entre le niveau '*cycle accurate*', qui donne une description précise des opérations de transport effectuées à chaque cycle d'horloge, et le niveau '*Approximately-timed*' où des durées sont associées aux opérations de transport des données. Ce principe d'abstraction est ici appliqué sur les propriétés non fonctionnelles en vue de l'évaluation des performances du système. En utilisant des langages comme SystemC, il est avéré que le temps de simulation d'un modèle dépend du nombre de changements de contexte effectués pendant son exécution [**97**]. Un changement de contexte intervient dès lors qu'une transaction entre deux modules SystemC est effectuée. Il s'agit donc de limiter au maximum le nombre de transactions initiées pour pouvoir réduire les temps de simulation de nos modèles. La figure 3.4 permet d'illustrer la méthode que nous proposons d'utiliser.

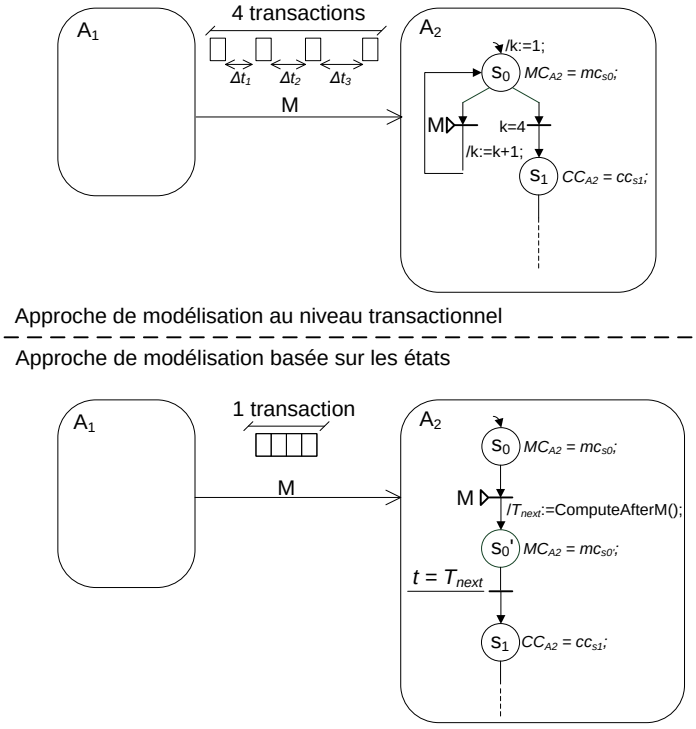

Figure 3.4 – Principe de la méthode de calcul des propriétés non fonctionnelles.

La figure 3.4 présente deux activités désignées A_1 et A_2. A_1 est une activité émettrice et A_2 est une activité réceptrice. Sur la partie haute de la figure, l'activité A_2 est en attente de quatre transactions M pour pouvoir passer de l'état S_0 à l'état S_1. Δt_1, Δt_2 et Δt_3 sont ici représentatifs des délais qui sont induits par le nœud de communication utilisé pour transférer les données. En définissant M à un niveau de granularité plus élevé, il est possible de remplacer ces quatre transactions par une seule transaction. De manière à conserver un comportement temporel équivalent pour l'activité A_2, un état S_0' est rajouté. Le temps de transfert associé à la réception des données est alors calculé lors de la réception de la transaction M. Ce calcul nécessite que les délais Δt_1, Δt_2 et Δt_3 soient ajoutés dans le contenu de la transaction M. L'instant de réception est alors utilisé comme instant de référence et les délais Δt_1, Δt_2 et Δt_3 lui sont ajoutés pour calculer l'instant de fin de réception. L'état S_0' est alors introduit afin de prendre en compte le temps de transfert estimé des données.

Une expérimentation a été faite pour justifier du gain en termes de temps de simulation pouvant être obtenu en réduisant le nombre de transactions initiées lors de l'exécution d'un modèle. La figure 3.5 présente le diagramme d'activités du modèle proposé pour réaliser cette expérimentation.

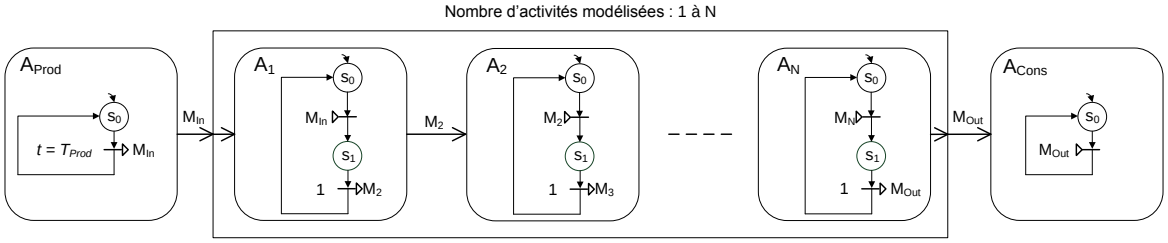

Figure 3.5 – Modèle pour l'évaluation du gain de temps de simulation en fonction du facteur de réduction du nombre de changements de contexte.

Chaque activité décrite ici représente un module SystemC. L'activité A_{Prod} est définie de manière à assurer la production d'une transaction M_{In} toutes les 100 ns. Le comportement décrit pour les activités A_1 à A_N permet ensuite d'assurer la retransmission des transactions reçues en entrée sur la sortie. Aucun traitement n'est ici considéré entre la réception et la production d'une transaction. L'activité A_{Cons} vient elle consommer les transactions M_{Out} en sortie.

L'outil CoFluent Studio a été utilisé dans le cadre de cette expérimentation pour capturer et simuler ce modèle. Il offre en effet la possibilité de capturer graphiquement des modèles de niveau transactionnel. Il génère ensuite automatiquement une description en SystemC du modèle qui peut alors être simulée pour analyser son comportement et les performances obtenues. Pour cette expérimentation, nous avons fixé un temps simulé de 11 ms. L'activité A_{Prod} génère alors 110000 transactions M_{In} durant la simulation du modèle. Nous avons simulé notre modèle une première fois en considérant une seule activité entre A_{Prod} et A_{Cons}. Nous avons alors observé le temps nécessaire à la simulation du modèle. Une activité intermédiaire entre A_{Prod} et A_{Cons} a ensuite été successivement ajoutée et nous avons mesuré les temps de simulation requis. Le but était d'évaluer l'influence du nombre de changements de contexte sur le temps de simulation du modèle. La figure 3.6 montre les résultats concernant les gains en temps de simulation observés.

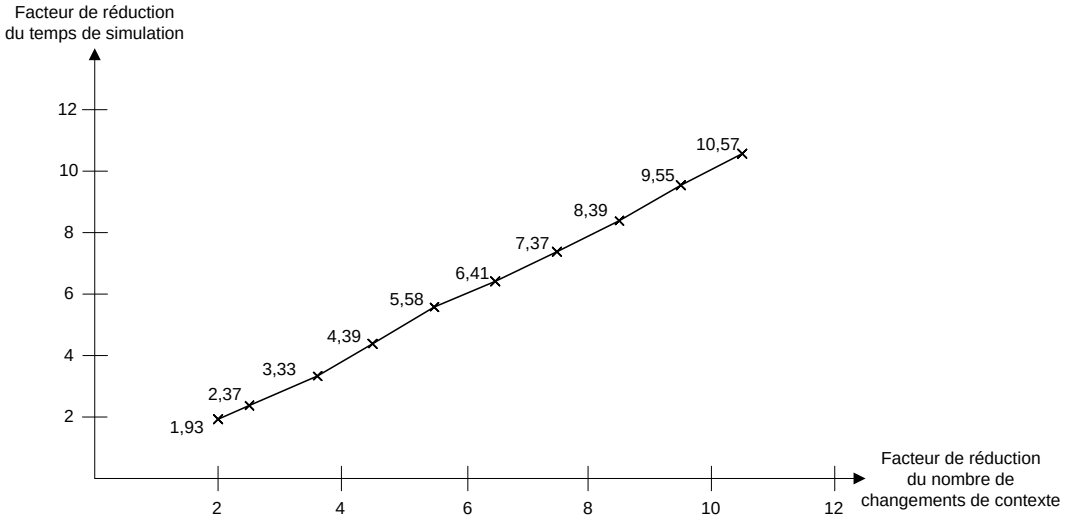

Figure 3.6 – Gain de temps de simulation en fonction du facteur de réduction du nombre de changements de contexte.

L'axe des abscisses représente le rapport de commutations économisées entre un modèle avec N activités et un modèle avec une activité entre A_{Prod} et A_{Cons}. Pour une activité, deux changements de contexte sont effectués pour une transaction initiée par $A_{Prod.}$ Pour un modèle avec N activités, N+1 changements de contexte sont nécessaires. Le facteur de réduction du nombre de changement de contexte se calcul alors avec l'expression (N+1)/2, avec N le nombre d'activités considérées entre A_{Prod} et A_{Cons}. L'évaluation a porté successivement sur la prise en compte de 3, 4, 6, 8, 10, 12, 14, 16, 18 et 20 activités entre A_{Prod} et A_{Cons}. L'axe des ordonnées représente lui le facteur de réduction de temps de simulation en fonction du facteur de réduction du nombre de changements de contexte. Nous vérifions ainsi à travers cette expérimentation que le facteur de réduction du temps de simulation est quasiment proportionnel au facteur de réduction du nombre de changements de contexte au sein d'un modèle. Cette observation confirme l'intérêt de diminuer le nombre de transactions nécessaires à l'évolution du comportement d'un modèle lors de son exécution afin de limiter le temps de simulation requis. Il s'agira par la suite d'évaluer l'influence des algorithmes pouvant être ajoutés au sein des modules SystemC sur le temps de simulation.

Sur la partie haute de la figure 3.4, la propriété non fonctionnelle MC_{A2} considérée évolue dès lors qu'une transaction M est reçue en entrée. Cette propriété permet en effet d'observer l'évolution du coût mémoire induit par le stockage des informations contenues dans M. Une méthode spécifique est alors requise pour pouvoir observer l'évolution de cette propriété non fonctionnelle avec le niveau de granularité des transactions considéré sur la partie basse de la figure 3.4. Nous présentons dans la partie suivante la proposition d'un modèle générique permettant de décrire à la fois le comportement d'une activité et l'évolution des propriétés non fonctionnelles associées. Ce modèle générique est défini de manière à bien séparer la description qui est faite de l'évolution temporelle de l'activité liée aux instants de consommation et de production des transactions, et l'évolution temporelle des propriétés observées. Nous qualifions cette approche de modélisation de *state based* en ce sens qu'elle exprime les caractéristiques de l'architecture à partir d'états significatifs. Elle se différencie d'une approche *event based* qui elle exhibe l'ensemble des transactions nécessaires à l'évolution du système. Dans la partie suivante, nous proposons donc de modéliser une activité en se basant sur cette approche orientée *state based* de manière à pouvoir représenter d'une part la partie consommation et production de transactions et d'autre part l'évolution des propriétés non fonctionnelles analysées.

3.1.5 Formalisation du modèle d'exécution générique

Dans le cadre de notre approche, l'analyse des performances obtenues avec une architecture se fait en décrivant les états qui permettent de représenter l'évolution des propriétés non fonctionnelles caractéristiques des ressources de la plate-forme. Pour faciliter ce travail de modélisation, nous avons formalisé un modèle d'exécution qui permet de décrire comment l'état et les sorties du modèle vont être modifiés. Dès lors, nous proposons une représentation générique afin de faciliter la création des modèles. Cette création consiste à instancier le modèle générique et à combiner les différentes instances du modèle générique obtenues afin d'évaluer différentes solutions architecturales. Pour décrire notre modèle d'exécution générique, nous nous appuyons ici sur la notation présentée sur la figure 3.7 pour décrire une machine séquentielle décrite au niveau cycle d'horloge.

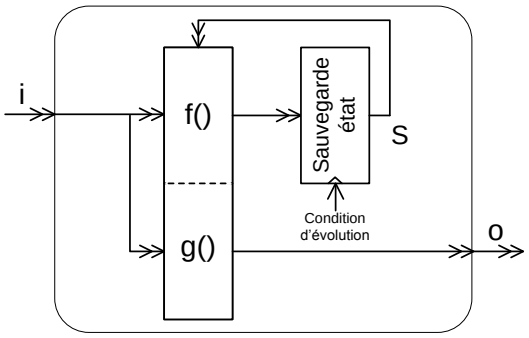

Figure 3.7 – Principe d'évolution d'une machine séquentielle au niveau cycle d'horloge.

Pour des modèles précis au niveau cycle, l'évaluation de l'état courant est faite de manière systématique relativement à une horloge servant de référence temporelle. La fonction *f* désigne la fonction d'évolution de l'état. Cette évolution dépend de l'état courant et des entrées. La *condition d'évolution* correspond ici à la période d'horloge utilisée comme référence temporelle pour faire évoluer l'état. La fonction *g* définit elle la fonction d'évolution des sorties. Les entrée/sortie *i* et *o* représentent à ce niveau des signaux logiques.

Au sein des modèles transactionnels que nous considérons, tel qu'illustré sur la figure 3.3, l'évolution des activités repose sur les transactions échangées (K, M et Q) et également sur les conditions temporelles (T_j, T_k et T_l) délimitant les états identifiés afin de rendre compte de l'utilisation faite des ressources matérielles et logicielles. Pour ces modèles transactionnels contenant des informations temporelles, il est possible de considérer que les évolutions de *f* et de *g* sont donc sensibles, soit à la présence des transactions d'entrée, soit aux conditions temporelles identifiées au sein du comportement de chaque activité. Ces constatations conduisent à faire évoluer la description donnée en figure 3.7.

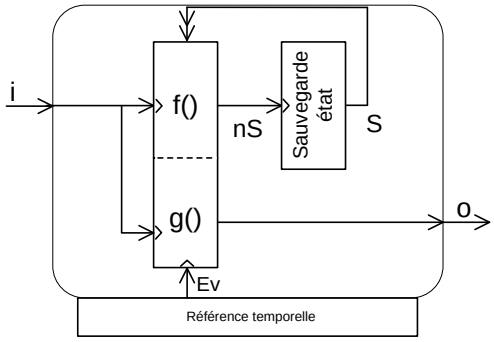

Figure 3.8 – Principe d'évolution d'une machine séquentielle dans une approche de modélisation transactionnelle.

Selon la notation présentée sur la figure 3.8, les fonctions *f* et *g* évoluent selon les transactions *i* reçues en entrée. Á partir de l'information reçue et de l'état *S* dans lequel se trouve la machine, *f* détermine le nouvel état *nS* (*newState*) dans lequel doit évoluer la machine. Les conditions temporelles d'évolution sont elles évaluées vis-à-vis d'une référence temporelle fournie par le simulateur. Les événements temporels correspondant aux instants d'évolution sont symbolisés par *Ev*.

Par ailleurs, comme cela est illustré sur la figure 3.3, deux catégories de sorties sont considérées au sein des modèles proposés. Une première catégorie se rapporte aux transactions produites. La deuxième concerne les observations faites sur les propriétés non fonctionnelles. La description présentée sur la figure 3.9 permet de distinguer ces deux catégories de sorties.

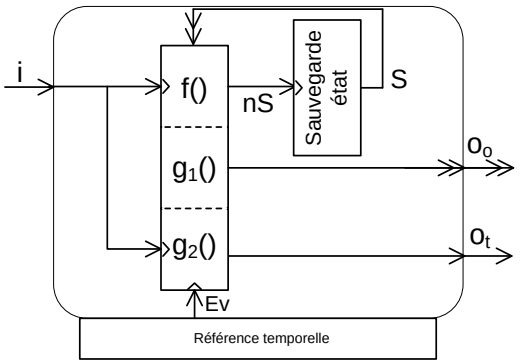

Figure 3.9 – Principe d'évolution d'une machine séquentielle dans une approche de modélisation transactionnelle.

g_1 désigne ici la fonction d'évolution des observations notées o_o, associant la valeur de la propriété non fonctionnelle observée et l'information temporelle associée. Sur la figure 3.3, cette association peut être illustrée par le couple (T_j, mc_{s2}) qui traduit une évolution de la propriété non fonctionnelle MC_{A2}. g_2 désigne elle la fonction définissant la production des transactions de sortie notées o_t.

Il est aussi possible de constater que certaines observations o_o peuvent ne pas dépendre de transactions en entrée ou n'entraînent pas de transactions en sortie. Ce cas se retrouve sur la figure 3.3, lors du passage de l'état S_1 à S_2 pour les observations de CC_{A2} et MC_{A2}. Dès lors, en utilisant le simulateur SystemC, il peut ne pas être nécessaire d'utiliser de primitive *sc_wait()* pour effectuer ces observations. Ainsi, lorsque les états pour lesquels les observations o_o ne dépendent pas des transactions d'entrée *i* et n'impliquent pas de sortie o_t, l'exécution de la machine séquentielle peut se faire en un temps nul vis-à-vis du simulateur.

Suite à ce constat, nous introduisons les notions de temps observé et de temps simulé :

- le temps observé désigne une information associée à une condition temporelle qui ne nécessite pas d'être prise en compte par le simulateur,
- le temps simulé représente à l'inverse une information au sein d'une condition temporelle entrainant une évolution du temps au niveau du simulateur.

Nous considérons, dans le cadre de l'approche expliquée dans la partie 3.1.3 que les temps observés et simulés peuvent être calculés au sein même de la machine. Ces calculs se font à partir de l'instant de réception d'une transaction en entrée et selon les propriétés temporelles associées à l'architecture analysée. Nous proposons donc de rajouter une sortie $T_{NextSimu}$ sur la représentation donnée sur la figure 3.10.

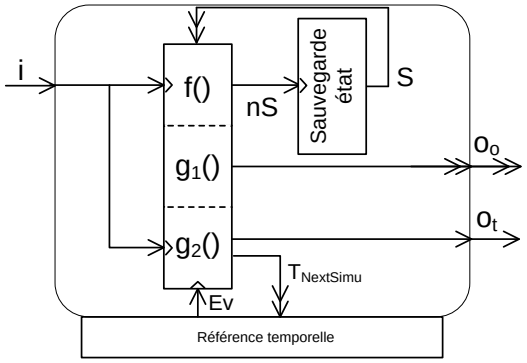

Figure 3.10 – Principe d'évolution d'une machine transactionnelle.

Une fois l'instant $T_{NextSimu}$ calculé, le simulateur doit alors faire évoluer la simulation jusqu'à cet instant. Toujours en utilisant le simulateur SystemC, ce délai peut être pris en compte en utilisant la primitive *sc_wait(time).*

Nous retenons ainsi le principe d'évolution présenté sur la figure 3.10 pour notre modèle d'exécution de niveau transactionnel. Il permet d'une part de considérer la consommation et la production de transactions en entrée et en sortie. Il permet d'autre part d'observer l'évolution des propriétés non fonctionnelles de l'architecture analysée pendant les intervalles de temps qui séparent la réception et l'envoi de transactions. Ces observations sont effectuées en temps nul vis-à-vis du simulateur de manière à réduire les temps de simulation nécessaires à l'exécution de nos modèles. Nous proposons au final de qualifier une telle machine de machine transactionnelle (ou machine basée sur les transactions) du fait que son évolution est guidée par les transactions reçues.

La figure 3.11 présente le modèle d'exécution générique à une entrée et une sortie que nous proposons pour faciliter la description de notre machine transactionnelle. Il pourrait bien sûr être étendu pour représenter une activité à N entrées et M sorties.

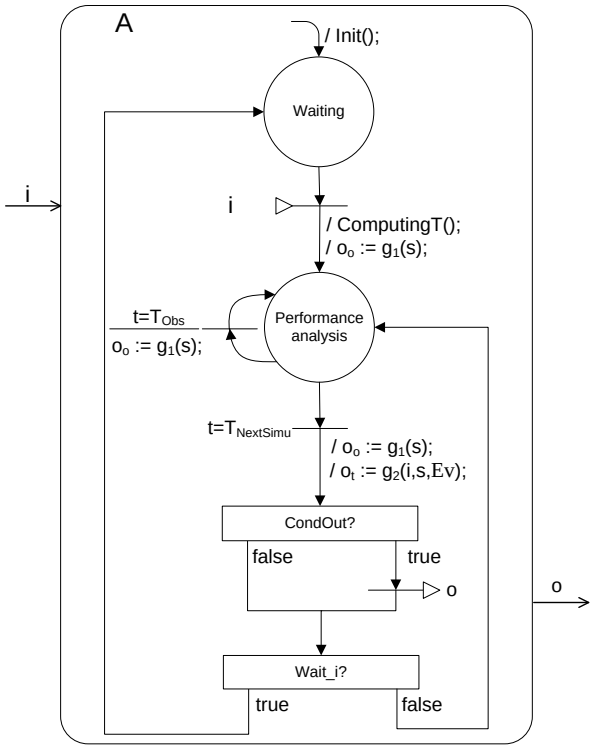

Figure 3.11 – Modèle générique retenu.

L'état *Waiting* représente ici l'état d'attente d'une transaction *i* en entrée. Á la réception de cette transaction, la fonction *ComputingT()* calcule les instants d'évolution $T_{NextSimu}$ vis-à-vis du simulateur ainsi que les instants significatifs T_{Obs} d'évolution des propriétés non fonctionnelles. L'état *Performance analysis* permet de gérer le passage entre les différents états décrits. Le passage d'un état à un autre peut se faire par l'intermédiaire de la condition $t=T_{Obs}$ qui est évaluée en temps nul vis-à-vis du simulateur. Dans ce cas, cette condition temporelle correspond à un instant où la valeur d'une ou plusieurs propriétés non fonctionnelles doit évoluer. L'action $O_o := g_1(s)$ détermine alors les nouvelles valeurs associées aux propriétés non fonctionnelles évaluées à afficher en fonction de l'état *S* dans lequel se trouve la machine. Le passage d'un état à un autre peut aussi se faire par l'intermédiaire de la condition $t=T_{NextSimu}$. Cette condition temporelle peut correspondre soit à l'instant où une transaction o doit être produite en sortie, soit à l'instant à partir duquel une

transaction *i* peut être reçue en entrée. Cette condition entraîne d'une part la mise à jour des nouvelles valeurs associées aux propriétés non fonctionnelles évaluées en fonction du résultat de l'action $O_o := g_1(s)$. Elle entraîne d'autre part la production ou non d'une transaction *o* en sortie et l'attente ou non d'une transaction *i* en entrée en fonction du résultat de l'action $O_t := g_2(i,s,Ev)$. La gestion de la production d'une transaction de sortie et l'attente d'une transaction en entrée se fait par l'intermédiaire des variables *CondOut* et *Wait_i* de type booléen.

Pour faciliter l'obtention d'une telle machine, nous recommandons d'exprimer les fonctions d'évolution *f*, g_1 et g_2 sous la forme d'une table de transition. Le tableau 3.2 décrit la forme générale retenue pour la représentation d'une table de transition. Nous nous sommes appuyés sur la notation proposée par le professeur Gajski dans [**98**].

État présent	État futur		Action	
	condition	état	condition	action
S_0				
S_1				
…				
S_N				

Tableau 3.2 – Table de transition proposée.

La partie gauche de la table permet de décrire la fonction d'évolution *f*. Les différents états parcourus sont indiqués au sein des colonnes notées *État présent* et *État futur*. La condition de passage d'un état présent à un état futur doit être précisée. Elle peut se rapporter à l'attente d'une transaction en entrée ou à une condition d'évolution temporelle. La partie droite se rapporte aux fonctions d'évolution des sorties g_1 et g_2. Les deux colonnes permettent d'exprimer les conditions d'évolution des sorties de la machine transactionnelle et les actions à initier suite à un changement d'état. Une action peut faire référence à la production de transactions en sortie (O_t) ou à des évolutions des valeurs associées aux propriétés non fonctionnelles évaluées (O_o).

Le tableau 3.3 montre un exemple de table de transition décrite à partir de ce formalisme pour l'activité A_2 présentée sur la figure 3.3.

État présent	État futur		Action	
	condition	état	condition	action
S_0	M	S_1	-	$CC_{A2}=cc_{s1}$; $MC_{A2}=mc_{s1}$;
S_1	$t=T_j$	S_2	-	$CC_{A2}=0$; $MC_{A2}=mc_{s2}$;
S_2	$t=T_k$	S_3	-	$CC_{A2}=cc_{s3}$; $MC_{A2}=mc_{s3}$;
S_3	$t=T_l$	S_0	-	Q $CC_{A2}=0$; $MC_{A2}=0$;

Tableau 3.3 – Exemple de table de transition.

La condition de passage entre l'état S_0 et S_1 se rapporte à l'arrivée d'une transaction M en entrée. Les passages entre les états S_1 et S_2, S_2 et S_3, et S_3 et S_0 se font suite à la vérification d'une condition temporelle. T_j et T_k représentent des temps observés puisque les conditions temporelles associées n'impliquent pas la production d'une transaction Q en sortie où l'attente d'une transaction M en entrée. T_l représente un temps simulé puisqu'il entraîne la production d'une transaction Q en sortie. L'évolution des valeurs des propriétés non fonctionnelles CC_{A2} et MC_{A2} est représentée dans la colonne *action*. Cette évolution est décrite en considérant les actions effectuées lors du passage des transitions. Cette convention est adoptée dans la suite du document. Il serait également possible de considérer des actions comme effectuées pendant la durée des états. Enfin, seul le passage de l'état S_3 à l'état initial S_0 entraîne ici la production d'une transaction Q en sortie.

Nous présentons dans la partie suivante la mise en œuvre proposée du modèle générique au sein de l'outil de saisie CoFluent Studio.

3.1.6 Mise en œuvre dans l'outil CoFluent Studio

Les modèles proposés dans le cadre de notre approche sont capturés et simulés à l'aide de l'outil CoFluent Studio. D'autres outils permettant la création de modèles en SystemC pourraient aussi être utilisés pour simuler les modèles proposés. L'outil CoFluent Studio propose un éditeur qui permet d'obtenir une description structurelle et comportementale d'un système à partir des différents éléments graphiques présentés sur la figure 3.12.

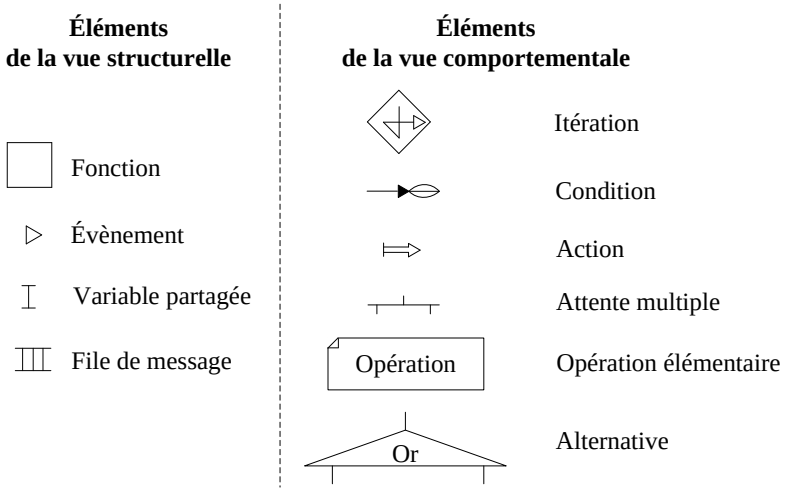

Figure 3.12 – Éléments graphiques pour la modélisation d'un système avec l'outil CoFluent Studio.

La saisie de la représentation structurelle se base sur les quatre éléments définis sur la partie gauche de la figure 3.12. Á partir de ces éléments, il est possible de représenter les différentes fonctions mises en jeu ainsi que les relations qui les lient. Trois types de relation sont possibles pour connecter les fonctions entre elles :

- l'évènement pour synchroniser les fonctions,
- la variable partagée pour échanger des données sans synchronisation,
- la file de message pour l'échange de données de type producteur/consommateur.

La partie droite de la figure 3.12 présente les six éléments graphiques qui permettent ensuite d'associer un comportement à chaque fonction considérée. Le comportement des opérations élémentaires peut ensuite être détaillé avec un éditeur d'algorithmes et différents attributs peuvent être utilisés pour caractériser les différents éléments du modèle. Á partir de cette description, l'outil peut générer un modèle SystemC exécutable pour pouvoir observer les propriétés d'un système.

La partie droite de la figure 3.13 présente la description définie pour capturer avec l'outil CoFluent Studio le modèle générique de la machine transactionnelle proposé et présenté sur la partie gauche.

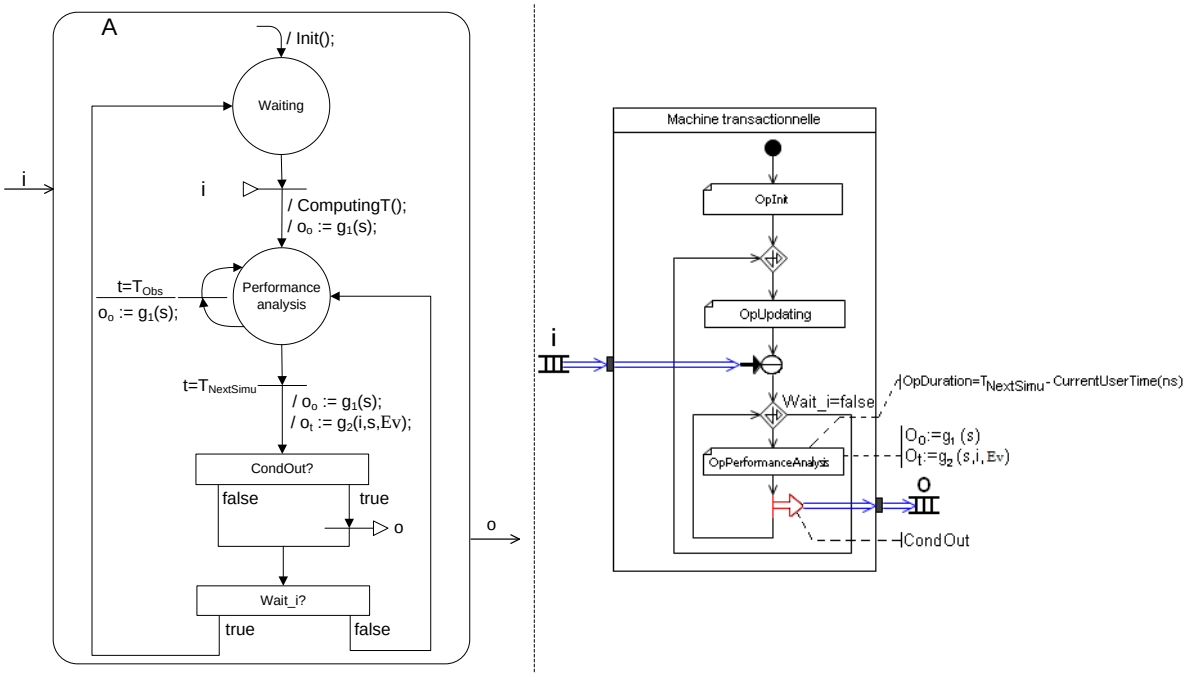

Figure 3.13 – Modélisation dans l'outil CoFluent Studio du modèle générique d'une machine transactionnelle à une entrée et une sortie.

L'opération *OpInit* permet de réaliser la phase d'initialisation. Ensuite la fonction rentre dans une première boucle itérative. L'opération *OpUpdating* permet de mettre à jour la condition *Wait_i* pour que, suite à la réception d'une transaction *i*, elle puisse rentrer dans la deuxième boucle itérative. Une condition d'attente permet d'assurer la réception des transactions *i* en entrée. Á la réception d'une transaction *i* via la file de message du même nom, la fonction rentre alors dans la deuxième boucle itérative. La taille de la file de message *i* est ici fixée à zéro de façon à correctement représenter le protocole de rendez-vous voulu et le caractère bloquant associé. L'opération *OpPerformanceAnalysis* contient l'algorithme proposé pour décrire l'évolution du comportement de la machine transactionnelle. Cet algorithme permet d'une part d'afficher les observations faites sur les propriétés non fonctionnelles considérées. Il permet d'autre part de définir la durée OpDuration de l'opération *OpPerformanceAnalysis* à chaque itération. Cette durée peut correspondre au délai devant s'écouler avant la production d'une transaction de sortie. L'autre cas se rapporte au délai devant s'écouler avant de pouvoir recevoir une nouvelle transaction *i* en entrée. L'expression suivante *OpDuration=*$T_{NextSimu}$*-CurrentUserTime(ns)* permet de calculer cette durée. La primitive *CurrentUserTime(ns)* retourne l'instant où se trouve le simulateur et $T_{NextSimu}$ représente l'instant futur devant être pris en compte par le simulateur. La génération ou non d'une transaction de sortie lors d'une itération se fait à partir de l'action conditionnée *CondOut* sur la file de message *o*. Pour qu'une fonction revienne en attente d'une transaction *i* en entrée et donc sorte de la boucle itérative, la condition booléenne *Wait_i* doit prendre la valeur *true*.

La figure 3.14 illustre une façon de décrire sous forme algorithmique une table de transition capturée au sein de l'opération *OpPerformanceAnalysis* à partir de l'exemple de la table de transition présentée dans le tableau 3.3.

```
OpPerformanceAnalysis
(
    ccs1 = NbOpToPerform1/Tproc1;
    mcs1 = M.Size;
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", CurrentUserTime(ns), 0); // (1)
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", CurrentUserTime(ns), ccs1);// (2)
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", CurrentUserTime(ns), 0);
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", CurrentUserTime(ns), mcs1);
    Tj = CurrentUserTime(ns) + Tproc1;
    mcs2 = Result1.Size;
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", Tj, ccs1);// (3)
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", Tj, 0);// (4)
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", Tj, mcs1);
    cofDisplay("Tobs=%0.f ns;MCA2=%d op", Tj, mcs2);
    ccs3 = NbOpToPerform2/Tproc2;
    mcs3 = mcs2 +P.Size;
    Tk = M.Treception2;
    cofDisplay("Tobs=%0.f ns; CCA2=%d op/s", Tk, 0);// (5)
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", Tk, ccs3);// (6)
    cofDisplay("Tobs=%0.f ns;CCA2=%d bits", Tk, mcs2);
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", Tk, mcs3);
    Tl = Tk + Tproc2;
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", Tl, ccs3);// (7)
    cofDisplay("Tobs=%0.f ns;CCA2=%d op/s", Tl,0);// (8)
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", Tl, mcs3);
    cofDisplay("Tobs=%0.f ns;MCA2=%d bits", Tl, 0);
     Q.Size = Result2.Size;
    CondOut=true;
    Wait_i=true;
    OpDuration= Tl-CurrentUserTime(ns) ;
    }
```

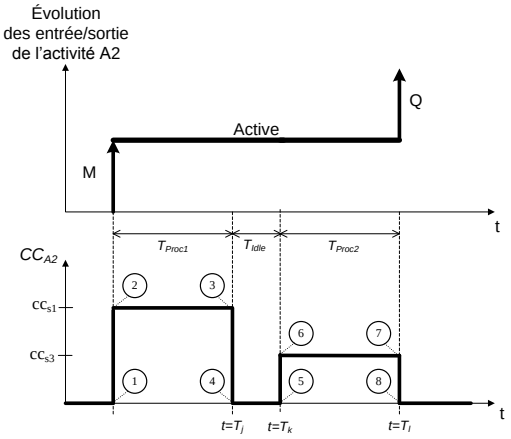

Figure 3.14 – Algorithme associé à l'opération *OpPerformanceAnalysis* et observations obtenues.

Sur cet exemple, à la réception d'une transaction *M*, les nouvelles valeurs cc_{s1} et mc_{s1} traduisant l'évolution des propriétés non fonctionnelles CC_{A2} et MC_{A2} sont tout d'abord calculées. Le calcul de cc_{s1} se fait à partir des paramètres T_{Proc1} et $Nb_{OpToPerform1}$ définis. T_{Proc1} correspond au temps nécessaire à la réalisation du traitement. $Nb_{OpToPerform1}$ représente le nombre d'opérations à réaliser durant cette période. L'instant de réception de *M* est ici utilisé comme instant de démarrage des traitements. L'instant de fin de traitement T_j est lui calculé avec le paramètre T_{Proc1}. L'affichage des observations est réalisé à l'aide de la primitive *cofDisplay()*. Cette primitive permet d'afficher une fonction de type y=f(x) sur un graphique. Dans notre cas, x correspond à un instant d'évolution et y à une valeur d'une propriété non fonctionnelle à afficher. Les paramètres x et y sont respectivement placés en première et deuxième position dans la primitive *cofDisplay()*. La partie droite de la figure 3.14 permet d'illustrer le lien entre les primitives *cofDisplay()* présentent dans l'algorithme et l'affichage de l'évolution de la propriété non fonctionnelle CC_{A2}. Nous illustrons ensuite avec l'instant T_k comment l'accroissement du niveau de granularité d'une transaction *M* peut permettre de réaliser des observations en temps nul vis-à-vis du simulateur. L'instant T_k est en effet calculé à partir d'une information contenue dans *M* indiquant l'instant de réception d'un deuxième bloc de données à traiter. L'instant T_l est enfin pris en compte par le simulateur pour produire une transaction *Q* en sortie. Une durée *OpDuration*, correspondant au temps devant s'écouler avant l'envoi de cette transaction, est associée à l'opération *OpPerformanceAnalysis*. La gestion de la production de Q se fait en mettant la condition booléenne *CondOut* à *true*. Le retour sur la condition d'attente d'une transaction *M* se fait en mettant la condition *Wait_i* à *true*. Sur cet exemple, l'opération *OpPerformanceAnalysis* n'exhibe pas d'états. Cependant nous verrons par la suite qu'il peut être nécessaire de décrire plusieurs états au sein de cette opération.

3.2 Cas d'étude pour la prise en compte des propriétés non fonctionnelles au sein des modèles transactionnels

Dans cette partie, nous illustrons notre approche pour la prise en compte des propriétés non fonctionnelles au sein des modèles transactionnels. Notre expérimentation porte sur une fonction de transformée de Fourier rapide devant être implantée sur une architecture pipeline. L'analyse porte sur la puissance de calcul nécessaire à la mise en œuvre de cette fonction.

3.2.1 Description du cas d'étude

L'algorithme de transformée de Fourier rapide ou FFT (*Fast Fourier Transform*) a été introduit par Cooley et Tukey [**99**] en 1965. Il convertit N échantillons d'un signal capturé dans le domaine temporel en une série de N échantillons dans le domaine fréquentiel. Cet algorithme permet de décomposer une transformée de Fourier discrète (TFD) à N échantillons en N/2 transformées minimales, à deux échantillons chacune, répétées sur $\log_2(N)$ étages. Il est employé dans de nombreuses chaînes de communications numériques pour réaliser par exemple les fonctions d'égalisation et de démodulation OFDM. La figure 3.15 présente le calcul de la transformée de Fourier rapide réalisé sur huit échantillons que nous considérons dans le cadre de notre cas d'étude.

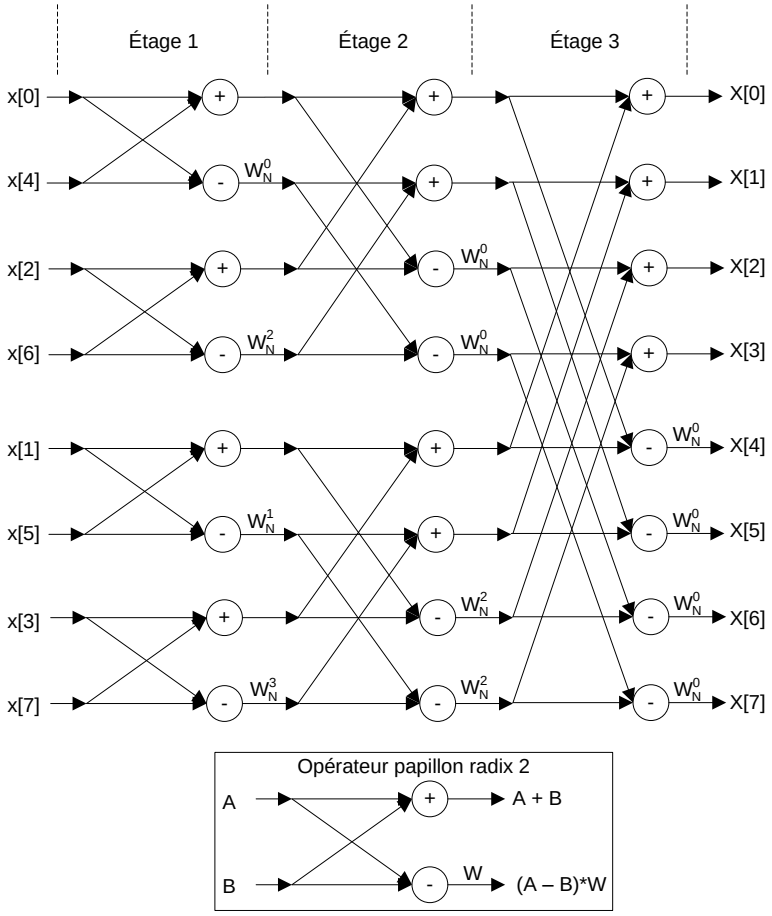

Figure 3.15 – Calcul d'une transformée de Fourier rapide sur huit points.

La partie basse représente l'opérateur élémentaire papillon radix 2 pour la décimation en fréquence utilisé pour réaliser le calcul de la FFT sur huit points. Il met en œuvre une

addition, une soustraction et une multiplication de deux nombres complexes soit dix opérations sur deux nombres réels. Pour exécuter une FFT sur huit points, il est nécessaire de réaliser douze fois les traitements associés à un opérateur radix 2. Chaque opérateur radix 2 de chaque étage nécessite de disposer des résultats de l'étage précédent pour démarrer ses traitements. Il est possible d'utiliser différentes architectures parallèles pour implanter cet algorithme. Nous considérons pour notre cas d'étude une des structures matérielles proposée par le fabricant Xilinx avec leur IP FFT pour réaliser cet algorithme [**100**]. La figure 3.13 décrit l'organisation des ressources utilisées.

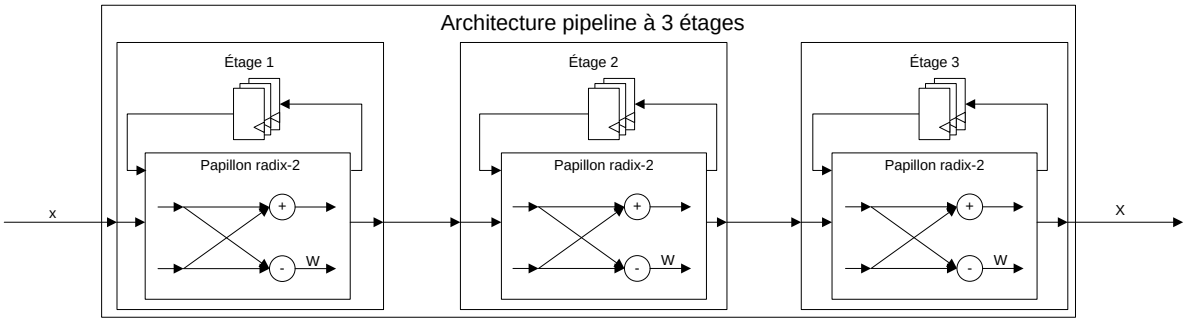

Figure 3.16 – Organisation des ressources de l'architecture considérée.

Cette architecture repose sur la mise en cascade de trois opérateurs papillons radix 2. Chaque opérateur permet de réaliser les quatre traitements devant être effectués par chaque étage. Un banc mémoire est proposé sur chaque étage de manière à pouvoir stocker les données reçues et les résultats des calculs intermédiaires.

Nous proposons ici d'analyser la puissance de calcul requise compte tenu de la contrainte de temps imposée par l'application et du paramétrage de la fréquence d'horloge des opérateurs de traitement. La puissance de calcul est ici exprimée en nombre d'opérations par seconde sur des nombres réels. Cette évaluation pourrait par exemple être menée dans le cadre du dimensionnement de cette architecture pour une implantation sur un FPGA. Il s'agirait en effet d'évaluer la quantité de ressources logiques nécessaires afin de satisfaire aux contraintes imposées.

3.2.2 Application de l'approche au cas d'étude

La partie basse de la figure 3.17 présente l'architecture pipeline à trois étages qu'il s'agit de modéliser en suivant notre approche. La partie haute de la figure 3.17 présente les trois activités « Pipeline stage 1 », « Pipeline stage 2 » et « Pipeline stage 3 » et les relations de couplage *SymbolOFDM*, *InStage2*, *InStage3* et *SymbolOFDMDemapped* considérées pour décrire le modèle d'évaluation des performances de cette architecture.

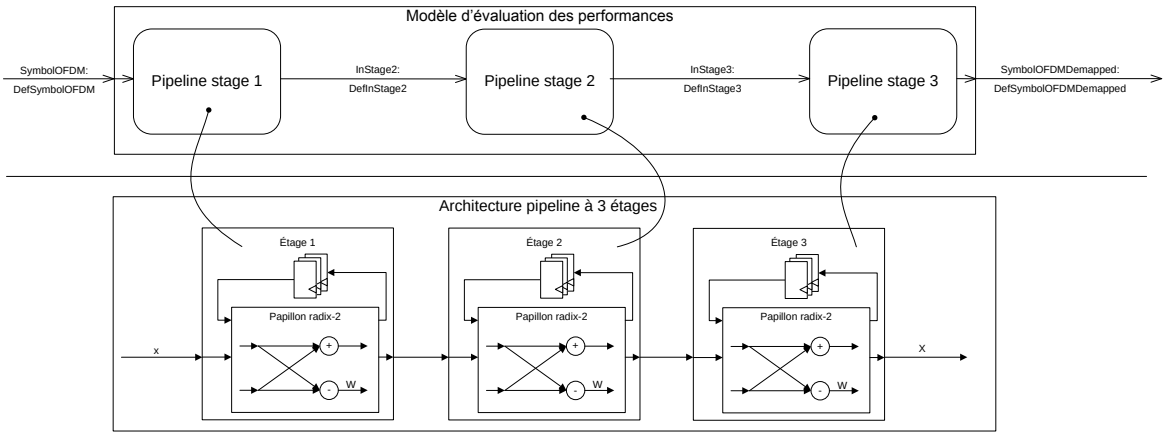

Figure 3.17 – Modèle de performance de l'architecture pipeline à trois étages.

La première étape de notre approche consiste à identifier les transactions nécessaires et suffisantes pour faire évoluer le comportement de chaque activité proposée afin de traduire l'évolution de la puissance de calcul requise à chaque instant pour chaque étage de pipeline. La figure 3.18 exhibe les relations entre activités à représenter.

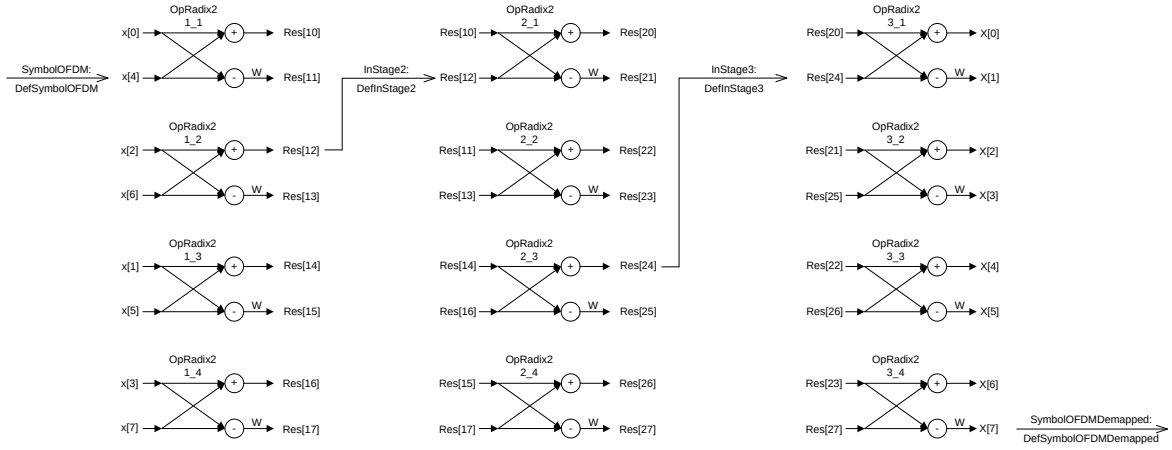

Figure 3.18 – Relations de couplage du modèle de performance.

Sur cette figure sont décrites les quatre itérations de l'opérateur papillon radix 2 que chaque étage de pipeline doit réaliser pour effectuer une transformée de Fourier rapide sur 8 points. Les données consommées et produites à chaque itération sont aussi représentées. Pour la mise en œuvre considérée, le démarrage des traitements sur le premier étage de pipeline s'effectue lorsque les symboles à valeurs complexes x[0] et x[4] sont présents en mémoire. Une transaction *SymbolOFDM* sera donc transmise à l'activité «Pipeline stage 1» à l'instant où le symbole complexe x[4] est reçu. Les traitements sur les deuxième et le troisième étage de pipeline démarrent eux lorsque les résultats Res[12] et Res[24] ont été calculés par les étages précédents. Des transactions *InStage2* et *InStage3* sont donc initiées par les activités «Pipeline stage 1» et «Pipeline stage 2» aux instants où ces données sont disponibles. La dernière activité «Pipeline stage 3» produit elle une transaction *SymbolOFDMDemapped* lorsque l'ensemble des échantillons de sorties X[0]… X[7] ont été calculés. Le niveau de granularité considéré en sortie permet de limiter le nombre de transactions produites. Des informations concernant les instants de production de chacun de ces échantillons de sortie pourraient être ajoutées dans la transaction *SymbolOFDMDemapped*. Ces informations pourraient en effet être utiles pour décrire par

exemple l'évolution de la puissance de calcul requise pour réaliser le traitement suivant sur ces échantillons.

La deuxième étape de notre approche consiste à identifier les paramètres ayant une influence sur la puissance de calcul que devra supporter le circuit d'implantation avec cette architecture. Dans le cadre de cette étude, l'opérateur de base utilisé par chaque étage de pipeline est l'opérateur papillon radix 2. La figure 3.19 présente les paramètres permettant de caractériser cet opérateur.

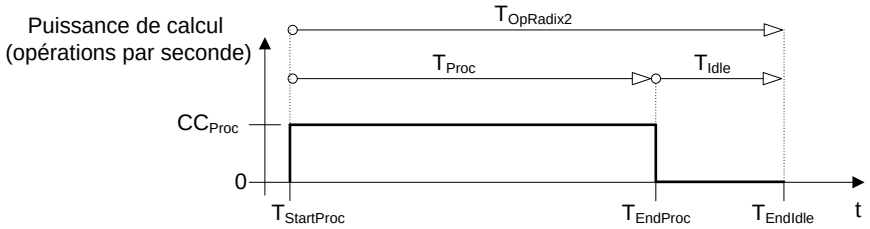

Figure 3.19 – Paramètres associés à l'opérateur papillon radix 2.

$T_{OpRadix2}$ correspond au temps maximal pouvant être alloué pour réaliser une itération de l'opérateur papillon radix 2. Cette durée peut être divisée en deux fenêtres temporelles T_{Proc} et T_{Idle}. T_{Proc} représente le temps défini pour que l'opérateur papillon radix 2 exécute les traitements qui lui sont associés. T_{Idle} correspond lui à la durée pendant laquelle l'opérateur radix 2 est inutilisé. CC_{Proc} représente le nombre d'opérations par seconde sur des nombres réels nécessaires à l'exécution des traitements associés à l'opérateur papillon radix 2.

Le choix des valeurs pouvant être attribuées à ces paramètres se fait alors en respectant la contrainte suivante. On considère ici un intervalle de temps $T_{SymbolOFDM}$ séparant deux séquences de huit échantillons à traiter. C'est cette contrainte de temps dépendante de l'application considérée qui est utilisée pour ensuite paramétrer $T_{OpRadix2}$. En effet, le premier opérateur radix 2 de l'architecture doit être en mesure d'effectuer les quatre itérations de l'opérateur papillon radix 2 pendant cet intervalle de temps, soit :

$$T_{OpRadix\ 2} \leq \frac{T_{SymbolOFDM}}{4}.$$

Il s'agit ensuite de répartir cette contrainte de temps entre T_{Proc} et T_{Idle} de manière à ce que :

$$T_{Proc} + T_{Idle} \leq \frac{T_{SymbolOFDM}}{4}.$$

Nous considérons ici que chaque opérateur radix 2 de chaque étage de pipeline fonctionne avec la même fréquence d'horloge. Les valeurs appliquées sur T_{Proc} et T_{Idle} sur le premier étage sont donc répercutées sur les paramètres T_{Proc} et T_{Idle} du deuxième et troisième étage. L'expression analytique suivante est utilisée pour calculer la puissance de calcul requise pour exécuter les traitements sur chaque étage de pipeline :

$$CC_{Proc} \leq \frac{10}{T_{Proc}}.$$

La valeur définie au numérateur correspond aux nombres d'opérations devant être réalisées sur deux nombres réels par l'opérateur radix 2 à chaque itération. Le paramètre T_{Proc} est utilisé dans le cadre de cette analyse pour évaluer l'influence de la fréquence d'horloge sur la quantité de ressources logiques requise pour implanter cette architecture.

La figure 3.20 présente les activités proposées avec les comportements associés pour décrire l'activité des ressources de calcul de chaque étage de pipeline.

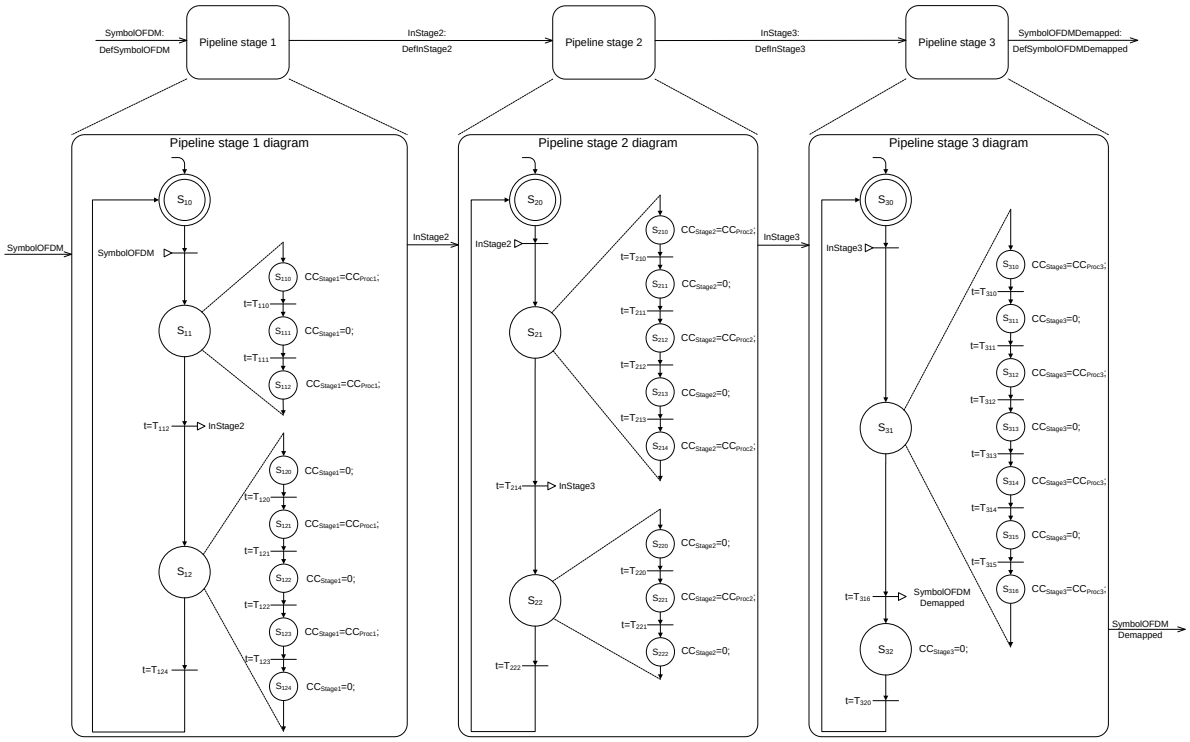

Figure 3.20 – Diagramme d'activités du cas d'étude.

Chaque activité reçoit ainsi en entrée une transaction indiquant l'instant de démarrage des quatre itérations de l'opérateur papillon radix 2. Les instant T_{xxx} correspondent aux instants successifs de démarrage et d'arrêt des traitements pour chaque opérateur de chaque étage de pipeline. Ils sont calculés à partir de l'instant de réception de la transaction d'entrée et des paramètres T_{Proc} et T_{Idle} définis. La puissance de calcul mise en jeu est elle exprimée à travers la variable CC_{StageX} avec X l'indice de l'étage.

Á partir de cette description, il est ensuite possible de définir les tables de transition des automates associés à chaque activité représentée sur la figure 3.20. Les tables de transition présentées dans les tableaux 3.4, 3.5 et 3.6 traduisent respectivement le comportement des activités « Pipeline stage 1 », « Pipeline stage 2 » et « Pipeline stage 3 ».

État présent	État futur		Action	
	condition	état	condition	action
S_{10}	SymbolOFDM	S_{110}	-	$CC_{Stage1}=CC_{Proc1}$;
S_{110}	$t=T_{110}$	S_{111}	-	$CC_{Stage1}=0$;
S_{111}	$t=T_{111}$	S_{112}	-	$CC_{Stage1}=CC_{Proc1}$;
S_{112}	$t=T_{112}$	S_{120}	-	InStage2 $CC_{Stage1}=0$;
S_{120}	$t=T_{120}$	S_{121}	-	$CC_{Stage1}=CC_{Proc1}$;
S_{121}	$t=T_{121}$	S_{122}	-	$CC_{Stage1}=0$;
S_{122}	$t=T_{122}$	S_{123}	-	$CC_{Stage1}=CC_{Proc1}$;
S_{123}	$t=T_{123}$	S_{124}	-	$CC_{Stage1}=0$;
S_{124}	$t=T_{124}$	S_{10}	-	-

Tableau 3.4 – Table de transition de l'activité « Pipeline stage 1 ».

État présent	État futur		Action	
	condition	état	condition	action
S_{20}	InStage2	S_{210}	-	$CC_{Stage2}=CC_{Proc2}$;
S_{210}	$t=T_{210}$	S_{211}	-	$CC_{Stage2}=0$;
S_{211}	$t=T_{211}$	S_{212}	-	$CC_{Stage2}= CC_{Proc2}$;
S_{212}	$t=T_{212}$	S_{213}	-	$CC_{Stage2}=0$;
S_{213}	$t=T_{213}$	S_{214}	-	$CC_{Stage2}=CC_{Proc2}$;
S_{214}	$t=T_{214}$	S_{220}	-	InStage3 $CC_{Stage2}=0$;
S_{220}	$t=T_{220}$	S_{221}	-	$CC_{Stage2}=CC_{Proc2}$;
S_{221}	$t=T_{221}$	S_{222}	-	$CC_{Stage2}=0$;
S_{222}	$t=T_{222}$	S_{20}	-	-

Tableau 3.5 – Table de transition de l'activité « Pipeline stage 2 ».

État présent	État futur		Action	
	condition	état	condition	action
S_{30}	InStage3	S_{310}	-	$CC_{Stage3}=CC_{Proc3}$;
S_{310}	$t=T_{310}$	S_{311}	-	$CC_{Stage3}=0$;
S_{311}	$t=T_{311}$	S_{312}	-	$CC_{Stage3}= CC_{Proc3}$;
S_{312}	$t=T_{312}$	S_{313}	-	$CC_{Stage3}=0$;
S_{313}	$t=T_{313}$	S_{314}	-	$CC_{Stage3}= CC_{Proc3}$;
S_{314}	$t=T_{314}$	S_{315}	-	$CC_{Stage3}=0$;
S_{315}	$t=T_{315}$	S_{316}	-	$CC_{Stage3}= CC_{Proc3}$;
S_{316}	$t=T_{316}$	S_{32}	-	SymbolOFDMDemapped $CC_{Stage3}=0$;
S_{32}	$t=T_{320}$	S_{30}	-	-

Tableau 3.6 – Table de transition de l'activité « Pipeline stage 3 ».

Le comportement de chaque étage de pipeline décrit sous la forme de tables de transition est ensuite traduit sous forme algorithmique. Chaque algorithme décrit l'évolution des sorties de chaque étage de pipeline en fonction de l'état dans lequel il se trouve et des événements d'entrées considérés. Il permet aussi l'affichage de la propriété non fonctionnelle considérée qui se rapporte à la puissance de calcul à mettre en œuvre.

La figure 3.21 présente la description structurelle et comportementale de l'architecture pipeline à trois étages obtenue en utilisant le modèle générique proposé.

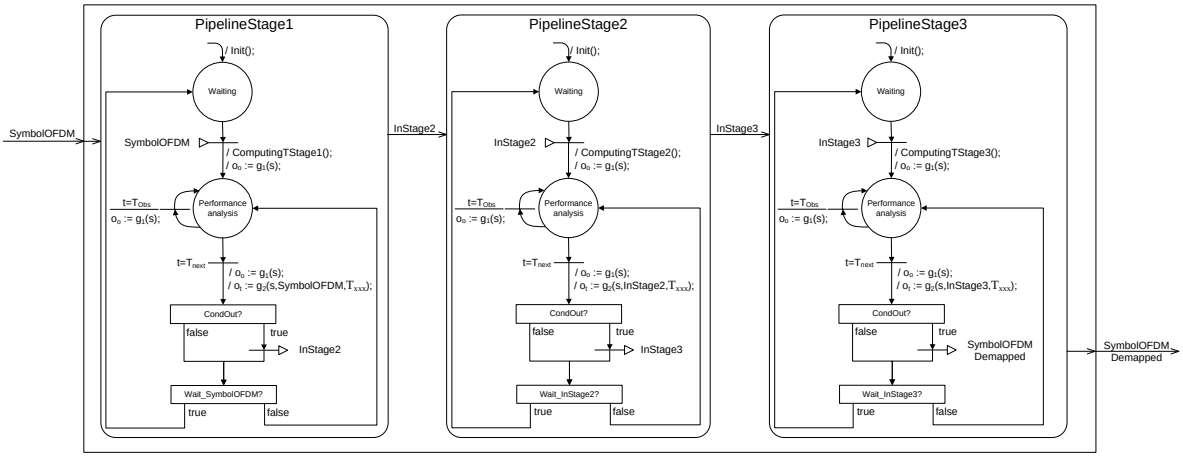

Figure 3.21 – Modèle transactionnel pour l'évaluation des performances d'une architecture pipeline à 3 étages réalisant une FFT sur huit points.

Les instants T_{112}, T_{124}, T_{214}, T_{222}, T_{316} et T_{320} décrits sur la figure 3.20 correspondent à des instants d'évolution $T_{NextSimu}$ vis-à-vis du simulateur. Les conditions temporelles associées entraînent en effet la production d'une transaction en sortie ou l'attente d'une transaction en entrée. Les autres instants précisés correspondent à des instants T_{Obs} d'évolution des propriétés non fonctionnelles CC_{StageX}. Le passage d'un état à un autre se fait alors en temps nul vis-à-vis du simulateur.

Ce modèle a ensuite été capturé avec l'outil CoFluent Studio. Le modèle obtenu est présenté sur la figure 3.22.

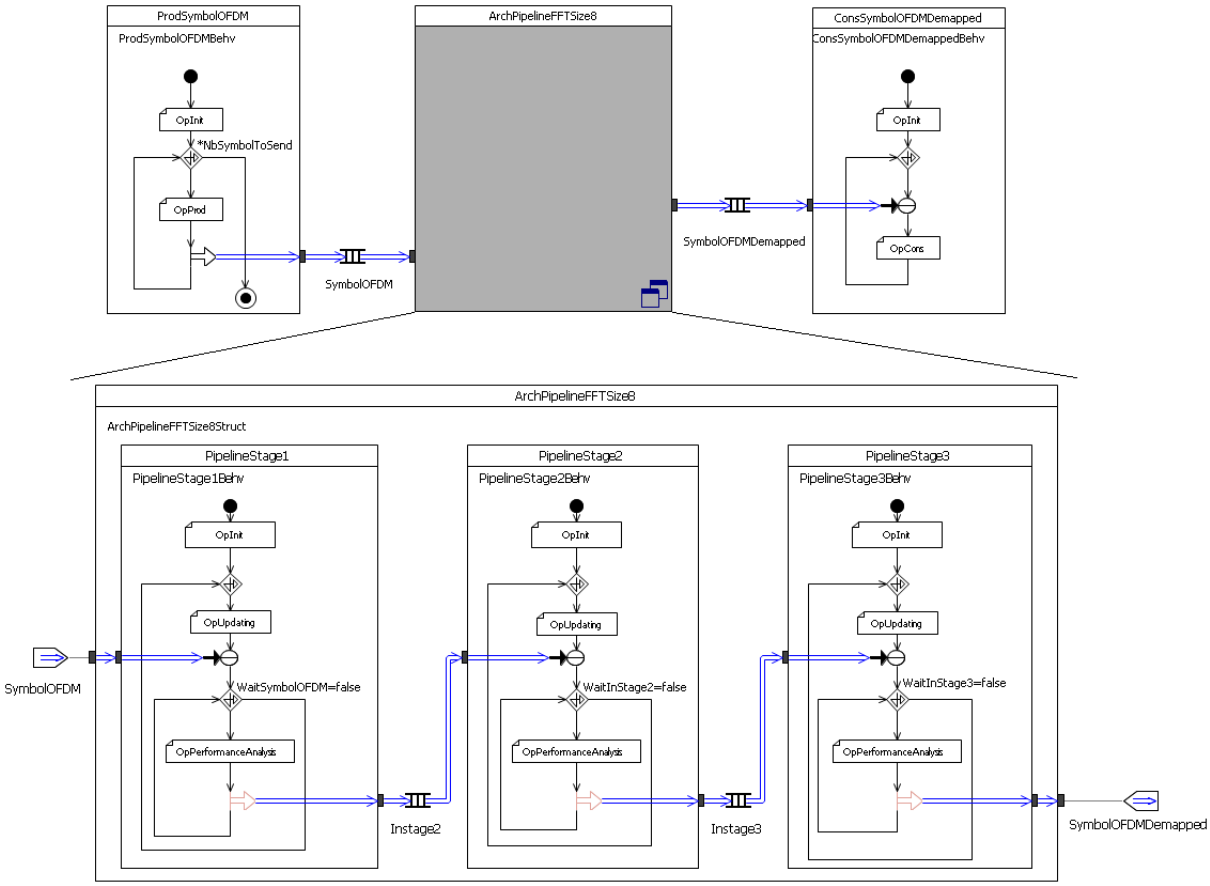

Figure 3.22 – Saisie dans l'outil CoFluent Studio du modèle d'exécution de la machine transactionnelle de la FFT sur huit points.

La partie haute de la figure 3.22 fait apparaître l'environnement de test proposé pour analyser les performances de l'architecture considérée. La fonction « ProdSymbolOFDM » est utilisée pour générer des transactions *SymbolOFDM* aux instants correspondant à la présence du symbole à valeur complexe d'indice quatre et donc au début des traitements sur le premier étage. La fonction « ConsSymbolOFDMDemapped » permet elle de consommer les résultats envoyés par la fonction « ArchPipelineFFTSize8 ». La partie basse de la figure 3.22 présente le modèle de l'architecture analysée. Les algorithmes définis à partir des tables de transition sont saisis au sein des opérations élémentaires *OpPerformanceAnalysis* de chaque étage de pipeline. Le modèle généré à l'aide de l'outil CoFluent Studio représente 1257 lignes de code SystemC dont 43% sont insérées automatiquement par l'outil.

3.2.3 Évaluation avec l'outil

L'exécution du modèle permet au final d'analyser la puissance de calcul requise pour effectuer une FFT sur huit points avec cette architecture pour différents paramétrages de T_{Proc} et T_{Idle}. Nous présentons dans cette partie les résultats obtenus en considérant une période d'arrivée $T_{SymbolOFDM}$ de huit échantillons en entrée de 1 ms. Le paramétrage de T_{Proc} et T_{Idle} doit donc se faire de manière à ce que :

$$T_{Proc} + T_{Idle} \leq 250\ \mu s.$$

Les figures 3.23 et 3.24 présentent des exemples de résultats pouvant être observés en appliquant un taux d'utilisation de 50% de la contrainte de temps sur le paramètre T_{Proc} de chaque étage de pipeline, soit :

$$T_{Proc} = 125\ \mu s, T_{Idle} = 125\ \mu s.$$

L'environnement de test est configuré de manière à simuler trois itérations successives de la FFT sur huit points. La figure 3.23 présente le profil de la puissance de calcul requise pour réaliser les traitements sur le premier étage de pipeline pour cette configuration.

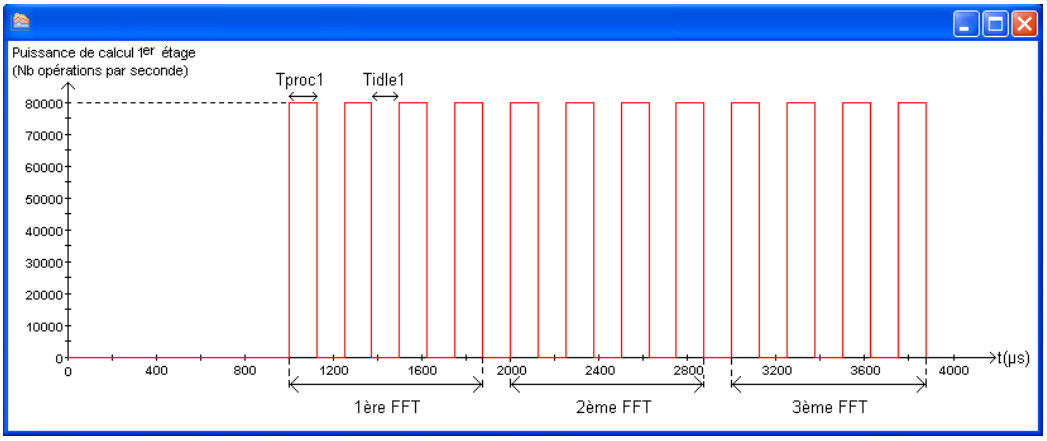

Figure 3.23 – Profil de la puissance de calcul évaluée pour le premier étage de pipeline.

Ce résultat de simulation permet d'observer qu'une puissance de calcul maximale de quatre-vingt mille opérations par seconde est nécessaire.

La figure 3.24 montre elle le profil globale de la puissance de calcul requise par l'architecture pour cette configuration.

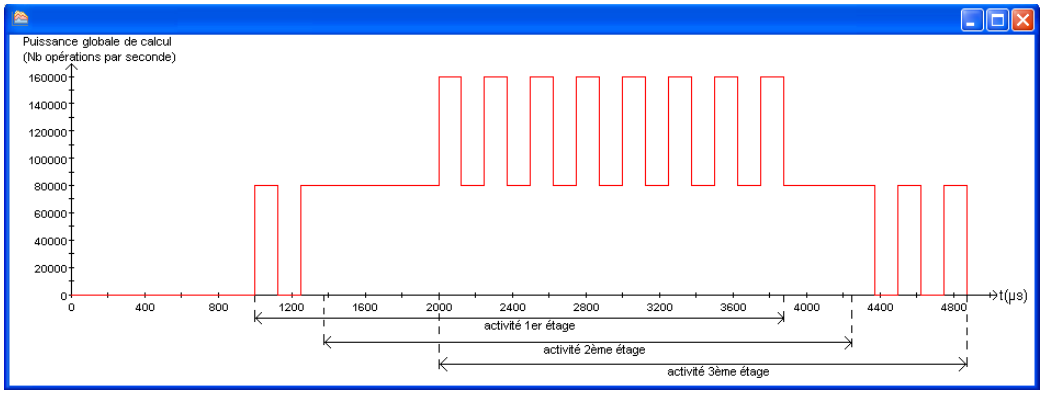

Figure 3.24 – Profil globale de la puissance de calcul requise par l'architecture.

Ce profil est obtenu à partir des informations utilisées pour afficher le profil de chaque étage de pipeline. Un algorithme a en effet été proposé pour permettre à la fin de la simulation d'afficher ce résultat. Ce résultat permet ainsi d'estimer la puissance de calcul devant être supportée par le circuit qui devra implanter cette architecture avec cette configuration.

La figure 3.25 montre le profil de la puissance de calcul mise en jeu pour réaliser l'ensemble des traitements sur chaque étage de pipeline en utilisant un autre paramétrage de T_{Proc} et T_{Idle}. T_{Proc} est ici paramétré de manière à évaluer l'application de la contrainte maximale $T_{OpRadix2}$.

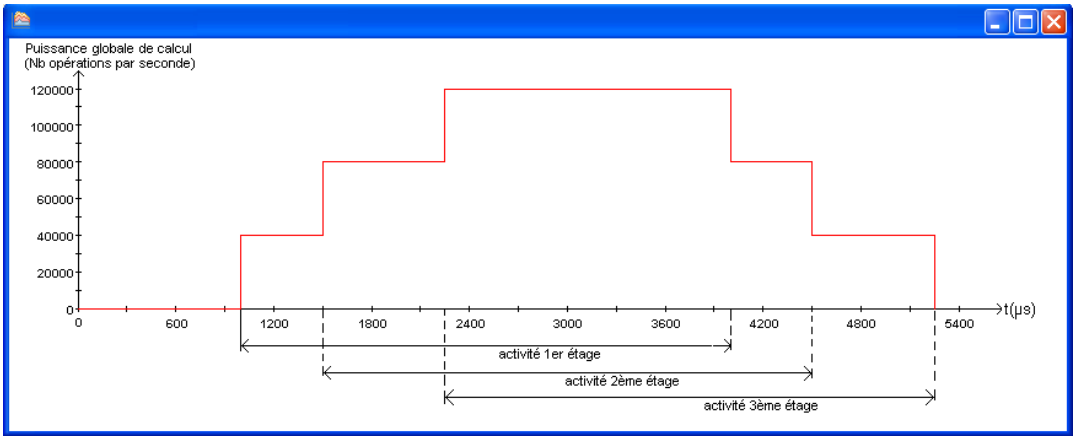

Figure 3.25 – Profil global de la puissance de calcul requise par l'architecture.

Cette figure permet de constater une diminution de la puissance de calcul requise par rapport à la configuration précédente. Il serait intéressant par exemple dans le cadre de cette étude, d'évaluer d'autres propriétés non fonctionnelles. Les observations faites sur ces propriétés permettraient en effet d'évaluer pour différentes configurations, le meilleur compromis consommation/performance/surface pouvant être obtenu.

Une deuxième expérimentation a été menée dans le cadre de ce cas d'étude afin d'évaluer les temps de simulation qui auraient été requis si un modèle exhibant les différentes transactions représentatives de l'ensemble des échanges de symboles à valeurs complexes avait été utilisé. Un deuxième modèle a ainsi été défini en considérant cette fois-ci les trente-deux transactions devant être initiées pour chaque évaluation d'une itération de la FFT sur huit points sur l'architecture considérée. Le tableau 3.7 permet de comparer les temps de simulation nécessaires pour exécuter les deux modèles proposés en considérant à chaque fois le même nombre d'itérations de la FFT.

Nb itérations FFT	Modèle simulé	Nb transactions	Temps de simulation sans observation	Gain de temps de simulation	Temps de simulation avec observations	Gain de temps de simulation
1000	A	4000	79 ms	6,13	8672 ms	7,67
	B	32000	484 ms		66515 ms	
2000	A	8000	140 ms	6,92	20406 ms	11,56
	B	64000	969 ms		235828 ms	
3000	A	12000	188 ms	7,64	34266 ms	14,83
	B	96000	1437 ms		508281 ms	
4000	A	16000	250 ms	7,62	40063 ms	21,97
	B	128000	1906 ms		880265 ms	

Tableau 3.7 – Comparaison des temps de simulation obtenus.

Le modèle A fait référence au premier modèle présenté et le modèle B au deuxième. La colonne *Nb transactions* indique le nombre de transactions initiées durant la simulation. Un facteur de réduction théorique de huit existe donc entre les deux modèles au niveau des transactions produites. La colonne *Temps de simulation sans observation* indique les temps

de simulation nécessaires pour exécuter les deux modèles dont le comportement a été défini de manière à simplement décrire l'évolution de chaque activité en fonction des transactions consommées et produites. Selon le nombre d'itérations considéré, un facteur de réduction supérieur à sept du temps de simulation peut être observé entre les deux modèles. Cette information permet de corroborer les résultats présentés sur la figure 3.6 en tenant compte cette fois-ci de l'influence de l'algorithmie sur les temps de simulation des modèles.

La deuxième observation sur les temps de simulation se fait cette fois-ci en considérant l'affichage des transactions initiées et l'évolution des propriétés non fonctionnelles avec les différents moyens d'affichage des résultats proposés par l'outil CoFluent Studio. Les observations sur les transactions échangées durant la simulation ont une influence non négligeable sur les temps de simulation des modèles. Cependant, le premier modèle proposé en utilisant notre approche permet de réduire de manière significative cet accroissement du temps de simulation.

Ce cas d'étude a permis de présenter les différents résultats pouvant être obtenus, en mettant en œuvre notre approche, et en exécutant les modèles décrits avec l'outil CoFluent Studio. Nous avons aussi pu constater à travers cette expérimentation les gains en temps de simulation obtenus en minimisant le nombre de transactions nécessaire à l'observation de l'évolution des propriétés non fonctionnelles d'une architecture.

3.3 Proposition d'une démarche pour le dimensionnement des futurs terminaux mobiles

Le dimensionnement opportun des ressources matérielles et logicielles des futurs terminaux mobiles implique une exploration rapide et efficace de l'espace de conception. Pour ce faire, il est essentiel d'assister l'architecte système dans le processus d'évaluation des performances en définissant une démarche systématique d'exploration. Une telle démarche consiste en un ensemble d'étapes, de modèles et de techniques à utiliser afin de mener à bien le travail d'exploration. La démarche présentée au sein de cette section s'appuie sur les propositions précédemment introduites au sein de ce chapitre. Notamment, elle repose sur l'approche présentée, consistant à modéliser au sein d'une même représentation les différentes propriétés d'une architecture, et ce afin de pouvoir évaluer rapidement les différentes propriétés émergentes.

La démarche proposée s'inscrit dans une démarche de conception plus générale partant de l'expression des besoins au sein d'un cahier des charges jusqu'à la réalisation finale du système. La démarche d'exploration se positionne telle qu'illustrée sur la figure 3.26.

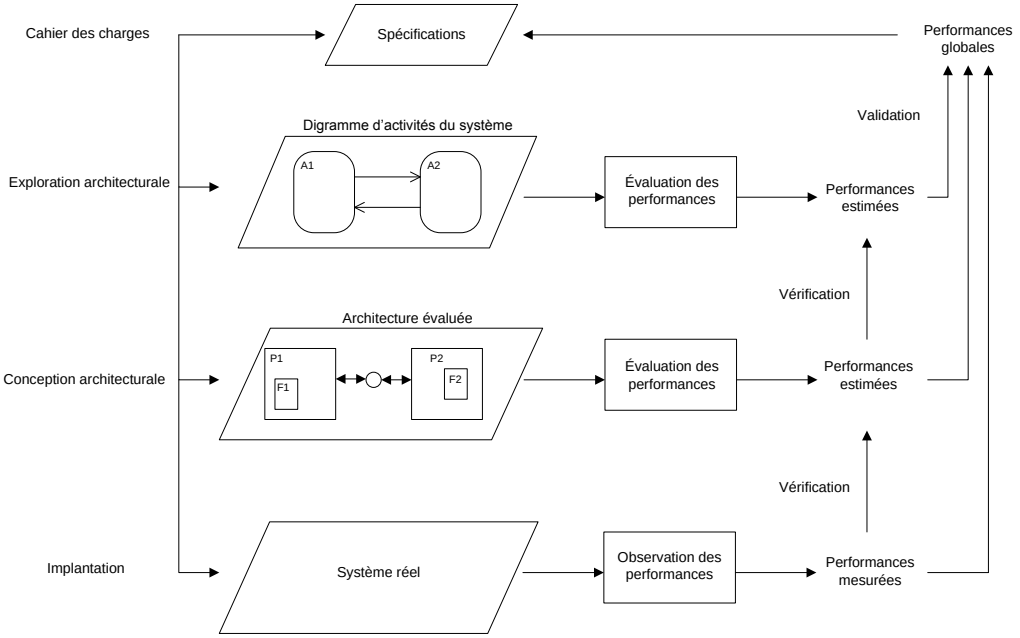

Figure 3.26 – Intégration de la démarche d'exploration architecturale au sein d'une démarche de conception.

L'exploration architecturale prend donc place une fois les spécifications du système établies. Le résultat de cette exploration résulte d'une comparaison entre les performances estimées et les performances globales à satisfaire. La conception architecturale désigne alors l'étude faite pour un niveau d'analyse plus précis de l'architecture, et ce en disposant de modèles raffinés pour l'ensemble des fonctions de l'application et des ressources de la plate-forme.

Le positionnement présenté sur la figure 3.26 relève les différentes étapes identifiées pour la démarche préconisée d'exploration architecturale :

- **La description du système étudié**

Elle consiste à représenter les différentes propriétés du système étudié, et ce selon trois vues complémentaires :

- La vue structurelle consiste à décrire la décomposition du système étudié sous la forme d'un diagramme d'activités. La notation précédemment introduite au sein de ce chapitre est considérée. Cette vue permet notamment de définir les transactions strictement nécessaires à l'analyse du système. Il a été montré de quelle façon il était possible de réduire le nombre de transactions nécessaires à l'analyse d'un système, et ce dans le but de réduire les temps de simulation associés.
- La vue comportementale vise à exprimer le comportement associé à chaque activité. Le comportement décrit tient aussi bien compte des caractéristiques temporelles à satisfaire du point de vue de l'application supportée par le système que de l'utilisation faite des ressources de la plate-forme considérée. Afin de faciliter cette description, il est recommandé d'utiliser la représentation sous forme de table de transition tel qu'introduit précédemment.
- La vue comportant l'expression des propriétés émergentes observées définit le lien entre les caractéristiques observées et les propriétés étudiées. Par exemple, la consommation associée à une architecture peut s'exprimer en fonction du nombre d'opérations exécutées et de l'utilisation faite des ressources de mémorisation.

- **L'évaluation des performances**

Elle consiste à analyser les résultats obtenus lors de l'exécution d'un modèle du système, et ce compte tenu du paramétrage opéré pour ce modèle. Les différentes expressions fournies au sein des tables de transition servent de base pour le paramétrage des modèles créés. La simulation de ces modèles permet de relever et de comparer les différentes performances atteintes.

- **L'exploration**

Elle suppose de pouvoir modifier aisément les paramètres des modèles afin d'étudier différentes configurations d'architectures. Les différentes alternatives à analyser portent sur :

- le découpage structurel du système,
- les caractéristiques des éléments de plate-forme,
- la distribution des fonctions au sein des ressources de traitement de la plate-forme,
- la répartition des contraintes de temps.

Selon l'approche proposée, nous considérons qu'une telle exploration peut être menée en apportant des modifications au sein des tables de transition et selon les contraintes de temps considérées.

L'exploration suppose de pouvoir simuler les modèles créés. Dans nos travaux, nous avons considéré l'utilisation de l'outil CoFluent Studio. Pour autant, les principes proposés peuvent être utilisés afin de créer des modèles SystemC indépendamment de cet outil.

Une telle démarche doit à terme permettre d'améliorer le processus d'exploration en établissant un ensemble d'étapes à suivre pour l'architecte système. Il en ressort la nécessité d'assister le processus de création des modèles utilisés. L'utilisation du modèle d'exécution générique permet d'envisager la création aisée de modèles pour lesquels il s'agirait de pouvoir définir un jeu approprié de paramètres dans le but d'instancier une architecture particulière à analyser. Les éléments introduits précédemment ont permis de définir le principe de fonctionnement de machines dites transactionnelles et la méthode de définition de ces machines. La définition d'un outil visant à créer des modèles sur la base du modèle d'exécution générique proposé permettrait de réduire les temps de création des modèles. Les fonctions assurées par cet outil permettraient la saisie des différentes propriétés de l'architecture et la génération de la description associée. La sémantique claire du modèle d'exécution générique permet d'envisager l'obtention de descriptions pouvant ensuite être traduites dans un langage de simulation tel que SystemC. Ce processus de création peut s'apparenter au processus mené lors de la synthèse de machines séquentielles pour lesquelles une expression donnée d'un automate dans un langage de description de matériel conduit à une mise en œuvre sous la forme de machines de Moore ou de Mealy.

On peut relever que l'approche proposée se distingue des approches en Y classique en ce sens qu'elle privilégie la création d'un modèle unique associant au sein d'une même vue les différentes propriétés relatives au système. Une telle approche ouvre la voie à différents aspects particulièrement essentiels à étudier pour l'architecte de systèmes. Un de ces aspects porte sur la répartition opportune des contraintes de temps entre les fonctions d'une application et la façon de mener le raffinement de la description.

Chapitre 4
Études de cas : Dimensionnement de systèmes de radiocommunication mobiles

Sommaire

Ce chapitre présente une illustration des différentes contributions apportées dans le cadre de ces travaux de thèse. Deux études de cas distinctes et complémentaires ont été définies. La première se rapporte au dimensionnement d'une architecture devant supporter les traitements associés à la couche physique de la partie réception du futur standard de communication LTE. Nous présentons ici le travail de modélisation effectué dans le but d'explorer différentes solutions architecturales. Le deuxième cas d'étude concerne un terminal mobile devant supporter une gestion dynamique des interfaces de communication afin d'assurer un niveau de qualité de service toujours satisfaisant pour l'utilisateur pour les différentes applications supportées. Nous présentons les techniques proposées pour représenter et dimensionner ce système en définissant un modèle de niveau transactionnel avec des informations temporelles. Les résultats de simulation obtenus permettent d'analyser et de vérifier les performances pouvant être atteintes par le système.

4.1 Modélisation et dimensionnement de l'architecture d'un récepteur LTE

Cette partie traite de l'application de l'approche présentée dans le chapitre 3 dans le cadre du dimensionnement des ressources nécessaires pour la mise en œuvre d'un récepteur LTE. Nous présentons le travail de modélisation effectué dans le but d'explorer différentes solutions architecturales et d'en évaluer les performances. L'objectif est de pouvoir définir les ressources de calcul et de mémorisation requises afin de satisfaire aux contraintes de temps liées au système étudié.

4.1.1 Contexte de l'étude

Le système étudié doit permettre d'assurer la réception au niveau de la couche physique de trames radio émises par le futur réseau de communication LTE de 3,9G. Le LTE est le nom du projet mené par le 3GPP pour définir les spécifications techniques [**101**] de cette nouvelle génération de réseau mobile. La nouvelle technologie d'accès radio E-UTRA (*Evolved Universal Terrestrial Radio Access*) associée à ce réseau d'accès radio a été définie de manière à supporter des débits de transmission pouvant atteindre jusqu'à 100 Mbit/s sur le lien radio descendant. Nous expliquons tout d'abord les différents traitements associés à la couche physique de ce standard par la présentation de la partie émission. La figure 4.1 donne une vue schématique des différents traitements réalisés au niveau de la couche physique d'une station de base appelée eNode pour transmettre des données à un terminal mobile.

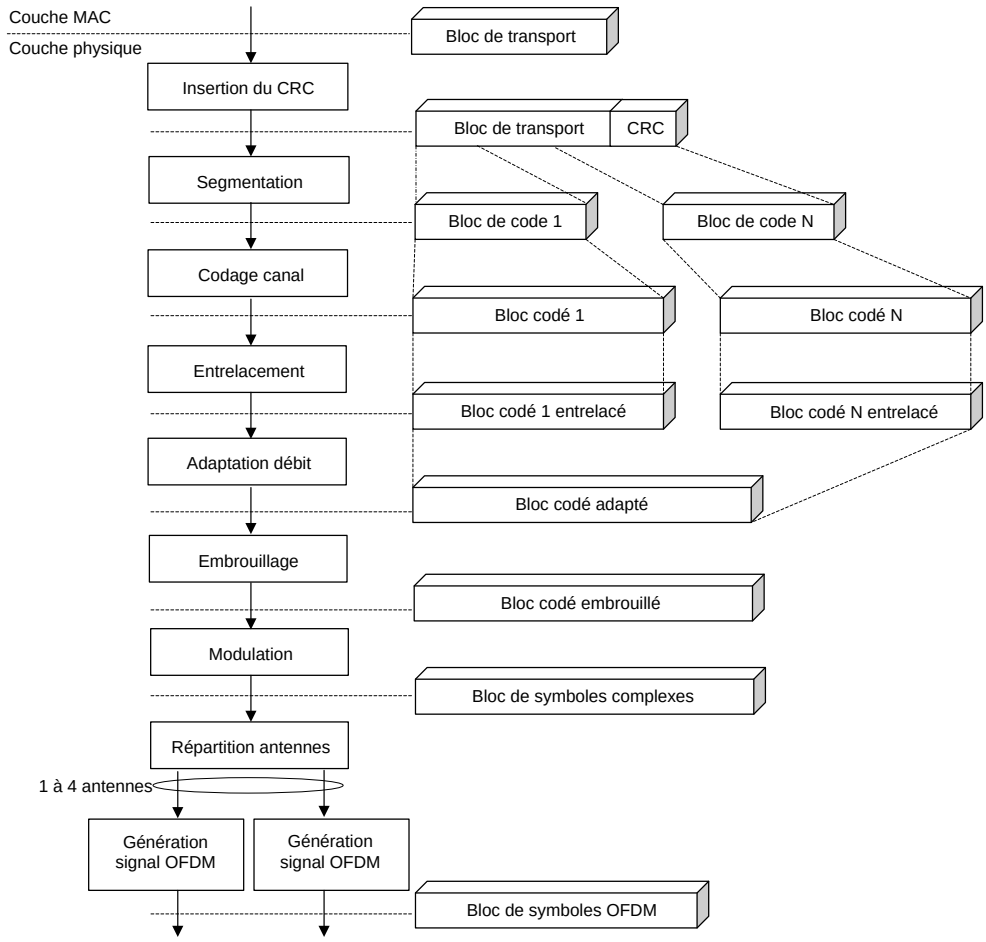

Figure 4.1 – Traitements au niveau de la couche physique d'une station émettrice LTE.

La couche MAC (*Medium Access Control*) assure la gestion de l'accès au support physique de transmission. Elle transmet à la couche physique toutes les millisecondes un bloc de transport ou TB (*Transport Block*) contenant les informations associées aux applications demandées par un utilisateur. Le premier traitement réalisé par la couche physique sur le TB consiste à lui ajouter un en-tête de données de contrôle de 24 bits appelé CRC (*Contrôle de Redondance Cyclique*). Le TB avec son en-tête CRC est ensuite segmenté en plusieurs blocs de code ou CB (*Code Block*) si sa taille est supérieure à 6144 bits [**102**]. Cette taille correspond à la quantité de données maximales qu'un codeur turbo peut supporter. Les CB sont ensuite encodés par un codeur turbo avec un taux de redondance d'un tiers. Les bits de sortie du codeur turbo sont alors entrelacés séparément avant de passer par une fonction d'adaptation de débit qui va permettre d'ajuster la quantité des bits transmise sur le canal physique. Puis, le CB passe par une fonction d'embrouillage qui consiste à multiplier élément par élément les données binaires par une séquence pseudo aléatoire donnée. Les données sont ensuite modulées selon la technique qui a été définie par la couche MAC. Trois types de modulation peuvent être employés : QPSK (*Quadrature Phase-Shift Keying*), 16QAM (*Quadrature Amplitude Modulation*) ou 64QAM. Il en résulte un CB de symboles à valeurs complexes. Il s'agit ensuite dans le cas d'une transmission multiantenne de répartir les différents symboles à valeurs complexes sur les différentes antennes de la station émettrice. La dernière opération consiste à réaliser une transformée de Fourier rapide inverse ou IFFT (*Inverse Fast Fourier Transform*) sur N points pour effectuer la transmission des symboles à valeurs complexes sur le support physique. La taille de l'IFFT mise en œuvre dépend de la bande passante allouée à l'utilisateur toutes les millisecondes. La bande passante peut être de 1.4, 3, 5, 10, 15, 20 MHz.

La figure 4.2 présente le format de trame utilisé pour assurer l'échange d'informations entre une station de base et un terminal mobile.

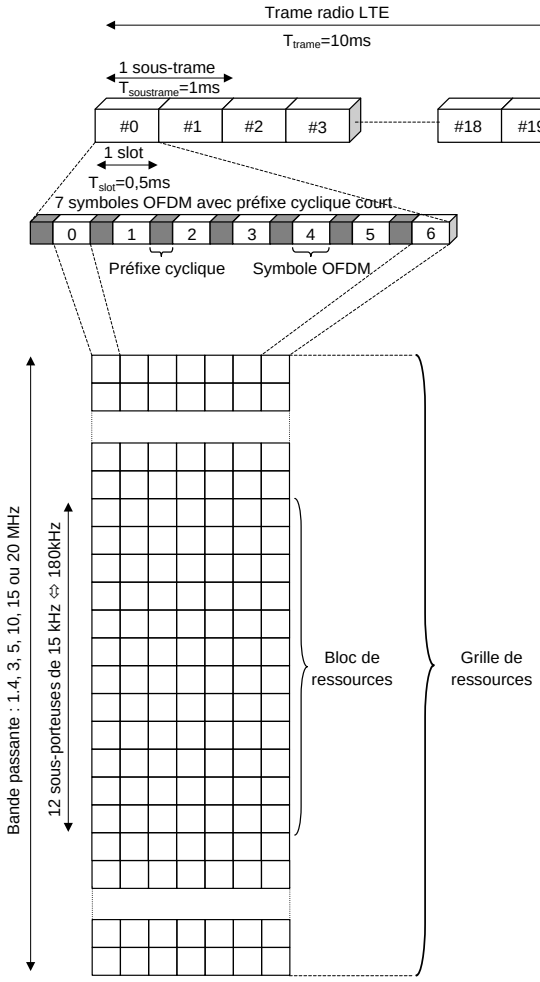

Figure 4.2 – Format de trame radio LTE.

Une trame radio LTE a une durée de 10 ms. Elle est divisée en dix sous-trames d'une durée de 1 ms. Cette durée d'1 ms corrrespond à la période d'envoi d'un TB par la couche MAC à la couche physique. Cette sous-trame est elle-même découpée en deux intervalles de temps (*slots*) de 0,5 ms. Un intervalle de temps est composé de six ou sept symboles OFDM en fonction du type de préfixe cyclique utilisé : long ou court.

La partie basse de la figure 4.2 donne une représentation schématique des ressources fréquentielles et temporelles utilisées. Le signal transmis dans chaque intervalle de temps pendant une durée de 0,5 ms est décrit sous la forme d'une grille de ressources dont la taille varie d'un point de vue fréquentiel en fonction de la bande passante allouée à la station émettrice. Cette grille de ressources est composée de blocs de ressources. Un bloc de ressources représente l'association de douze sous-porteuses consécutives d'une largeur de bande de 180 kHz pour un intervalle de temps. Le bloc de ressources est l'unité d'allocation la plus petite qui est utilisée par la station émettrice pour envoyer des données à un terminal mobile. Il est composé de quatre-vingt-quatre éléments de ressources contenant chacun un symbole à valeur complexe.

Le schéma de transmission utilisé dans le standard LTE se base sur la méthode d'accès OFDMA **(***Orthogonal Frequency-Division Multiple Access*). La technique de modulation OFDMA consiste à allouer un ou plusieurs blocs de ressources à chaque utilisateur toutes les millisecondes en fonction de la quantité d'informations à transmettre et des contraintes de temps à respecter pour assurer un niveau de QoS satisfaisant.

Le tableau 4.1 synthétise les différents paramètres utilisés par la couche MAC pour configurer les fonctions de traitement réalisées au niveau de la couche physique. La configuration utilisée peut être modifiée à chaque sous-trame envoyée.

Nombre de symboles OFDM dans une sous-trame	14					
Taux de codage	1/3					
Type de modulation	QPSK, 16QAM, 64QAM					
Bande passante (MHz)	1.4	3	5	10	15	20
Taille IFFT	128	256	512	1024	1536	2048
Nombre maximum de blocs de ressources par intervalle de temps	6	15	25	50	75	100
Nombre maximum de sous-porteuses occupées	72	180	300	600	900	1200

Tableau 4.1 – Paramètres de configuration de la couche physique du standard LTE.

Dans le cas d'étude considéré, il s'agit d'analyser les ressources requises en réception sur un terminal mobile supportant ce standard, et ce compte tenu des évolutions possibles au cours du temps des configurations de sous-trames radio LTE. Les traitements associés à la couche physique d'un récepteur LTE doivent permettre de reconstituer toutes les millisecondes le TB transmis. Nous proposons ici d'effectuer le travail d'exploration d'architectures devant permettre d'identifier et de caractériser les ressources nécessaires à la mise en œuvre de ces traitements.

4.1.2 Spécification de l'étude de cas

Afin de limiter la complexité du cas d'étude, nous nous focalisons ici sur une configuration avec une seule antenne à l'émission et à la réception. L'objectif est d'analyser les performances pouvant être obtenues par le système en évaluant différentes alternatives d'architectures et différentes répartitions des contraintes temporelles sur les fonctions de traitement à réaliser. Nous proposons pour cela de définir un modèle permettant d'obtenir par simulation les observations nécessaires pour comparer différentes distributions des fonctions de traitement sur différentes catégories de ressources.

La figure 4.3 représente les six activités décrivant les six fonctions de traitement considérées et mises en jeu au niveau de la couche physique pour assurer la réception des données sur un terminal mobile supportant le standard de communication LTE.

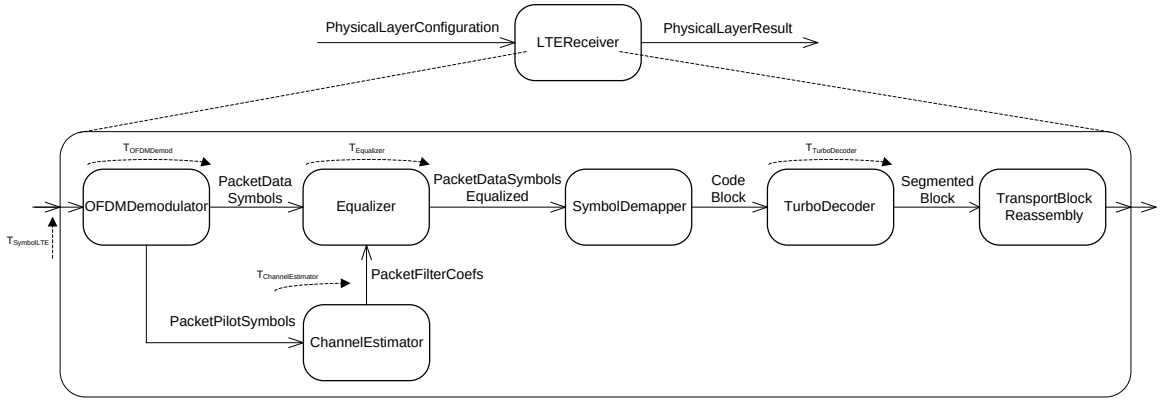

Figure 4.3 – Diagramme d'activités d'un récepteur LTE.

L'information d'entrée véhiculée par le lien *PhysicalLayerConfiguration* correspond à chaque symbole OFDM composant une sous trame-radio LTE. Le lien *PhysicalLayerResult* véhicule lui les blocs de transport reconstitués par la couche physique et envoyés à la couche MAC.

Les activités « OFDMDemodulator », « Equalizer », « Channel Estimator » et « TurboDecoder » sont proposées pour décrire l'utilisation des ressources de calcul nécessaires à l'exécution des traitements associés. Les activités « SymbolDemapper » et « TransportBlockReassembly » décrivent elles l'utilisation des ressources de mémorisation requises pour stocker les données associées.

Les relations entre ces activités sont définies de manière à exprimer les dépendances de données à considérer entre les différentes fonctions de traitement. Le lien *PacketDataSymbols* permet de transmettre à l'activité « Equalizer » le résultat du calcul de la FFT réalisée sur chaque symbole OFDM reçu. Le lien *PacketPilotSymbols* véhicule vers l'activité « ChannelEstimator » en une seule transaction les symboles pilotes introduits dans certains symboles OFDM. L'activité « ChannelEstimator » génère ensuite des paquets de données contenant chacun les coefficients nécessaires à la réalisation d'un processus d'égalisation sur un symbole OFDM démodulé via le lien *PacketFilterCoefs*. Le lien *PacketDataSymbolsEqualized* entre l'activité « Equalizer » et « SymbolDemapper » permet de transmettre un paquet de données contenant le résultat de l'égalisation d'un symbole OFDM. L'activité « SymbolDemapper » génère elle via le lien *CodeBlock* les blocs de données devant être décodés. Enfin l'activité TurboDecoder transmet à l'activité « TransportBlockReassembly » par l'intermédiaire du lien *SegmentedBlock* le résultat de l'opération de turbo décodage effectuée sur chaque bloc de données reçu. Des informations sont également transmises à travers ces relations de manière à pouvoir paramétrer chacune de ces activités.

Les contraintes de temps $T_{OFDMDemod}$, $T_{ChannelEstimator}$, $T_{Equalizer}$ et $T_{TurboDecoder}$ associées à chaque activité représentent les temps maximum pouvant être utilisés pour réaliser les traitements sur les données reçues. Ces contraintes de temps sont définies en tenant compte des caractéristiques de l'application (latence, débit,…) et de l'architecture proposée pour réaliser les traitements associés.

Nous présentons par la suite le travail mené pour représenter le comportement associé à chacune de ces activités.

- **Démodulateur OFDM**

La première activité « OFDMDemodulator » décrite se rapporte à la première fonction de traitement exécutée lors de la réception d'un symbole OFDM contenu dans une sous-

trame LTE. Elle consiste à réaliser une transformée de Fourier rapide afin de ramener le signal reçu du domaine temporel dans le domaine fréquentiel. Le démodulateur OFDM utilisé doit ici être capable d'exécuter le calcul d'une FFT sur 128, 256, 512, 1024, 1536 et 2048 points. Nous considérons ici que le calcul de la FFT est réalisé en utilisant l'algorithme radix 2 introduit dans la partie 3.2. Afin d'évaluer la puissance de calcul requise pour réaliser ce traitement, nous utilisons l'expression analytique suivante :

$$CC_{FFT} = \frac{5N \log_2(N)}{T_{ProcOFDMDemod}}.$$

Cette expression permet de calculer la puissance de calcul requise, en nombre d'opérations par seconde sur des nombres réels, en fonction de la taille N de la FFT à réaliser et du temps de traitement $T_{ProcOFDMDemod}$ défini. Ce temps correspond à la durée qui s'écoule entre, l'instant à partir duquel le calcul de la FFT sur le symbole OFDM peut démarrer, et l'instant où l'ensemble des résultats du calcul sont disponibles. Ce temps de traitement doit être défini de manière à respecter la période d'arrivée des symboles OFDM $T_{SymbolLTE}$ de 71428 ns. Il doit d'autre part rendre compte de l'architecture utilisée pour réaliser ce traitement.

La figure 4.4 présente le comportement associé à l'activité « OFDMDemodulator » proposée afin d'analyser la puissance de calcul à mettre en œuvre pour réaliser le calcul de la transformée de Fourier rapide en fonction du temps de traitement $T_{ProcOFDMDemod}$ défini et de la taille de la FFT à exécuter.

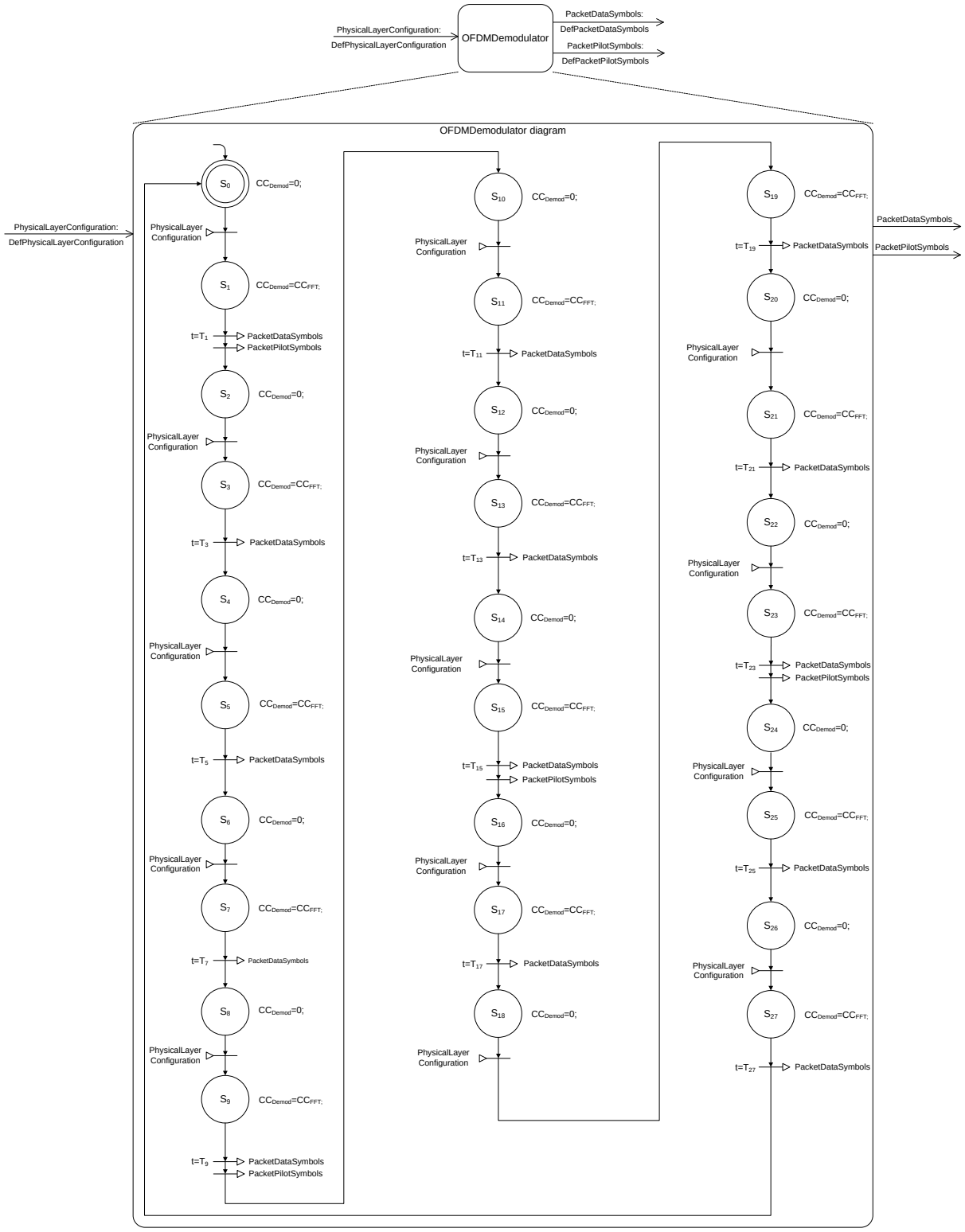

Figure 4.4 – Comportement associé à l'activité « OFDMDemodulator ».

Ce comportement est proposé pour décrire les traitements successifs réalisés sur les quatorze symboles OFDM constituants chaque sous-trame LTE. La variable CC_{Demod} décrit l'évolution de la puissance de calcul mise en jeu. Les états S avec des indices pairs correspondent aux états d'attente des symboles OFDM à traiter. Durant ces états, aucune ressource de calcul n'est utilisée. Les états S avec des indices impairs correspondent aux

instants pendant lesquels est effectué le calcul de la FFT sur le symbole OFDM reçu. Les instants de démarrage et d'arrêt du traitement sont liés aux caractéristiques et aux paramétrages des ressources utilisées. L'instant de production T_x du résultat de la FFT est donc calculé à partir de l'instant de démarrage et du temps de traitement $T_{ProcOFDMDemod}$ considérés.

Le tableau 4.2 présente la table de transition décrite pour l'activité « OFDM demodulator ».

État présent	État futur		Action	
	condition	état	condition	action
S_0	PhysicalLayerConfiguration	S_1		$CC_{Demod}=CC_{FFT}$;
S_1	$t=T_1$	S_2		$CC_{Demod}=0$; PacketDataSymbols PacketPilotSymbols
S_2	PhysicalLayerConfiguration	S_3		$CC_{Demod}=CC_{FFT}$;
S_3	$t=T_3$	S_4		$CC_{Demod}=0$; PacketDataSymbols
S_4	PhysicalLayerConfiguration	S_5		$CC_{Demod}=CC_{FFT}$;
S_5	$t=T_5$	S_6		$CC_{Demod}=0$; PacketDataSymbols
S_6	PhysicalLayerConfiguration	S_7		$CC_{Demod}=CC_{FFT}$;
S_7	$t=T_7$	S_8		$CC_{Demod}=0$; PacketDataSymbols
S_8	PhysicalLayerConfiguration	S_9		$CC_{Demod}=CC_{FFT}$;
S_9	$t=T_9$	S_{10}		$CC_{Demod}=0$; PacketDataSymbols PacketPilotSymbols
S_{10}	PhysicalLayerConfiguration	S_{11}		$CC_{Demod}=CC_{FFT}$;
S_{11}	$t=T_{11}$	S_{12}		$CC_{Demod}=0$; PacketDataSymbols
S_{12}	PhysicalLayerConfiguration	S_{13}		$CC_{Demod}=CC_{FFT}$;
S_{13}	$t=T_{13}$	S_{14}		$CC_{Demod}=0$; PacketDataSymbols
S_{14}	PhysicalLayerConfiguration	S_{15}		$CC_{Demod}=CC_{FFT}$;
S_{15}	$t=T_{15}$	S_{16}		$CC_{Demod}=0$; PacketDataSymbols PacketPilotSymbols
S_{16}	PhysicalLayerConfiguration	S_{17}		$CC_{Demod}=CC_{FFT}$;
S_{17}	$t=T_{17}$	S_{18}		$CC_{Demod}=0$; PacketDataSymbols
S_{18}	PhysicalLayerConfiguration	S_{19}		$CC_{Demod}=CC_{FFT}$;
S_{19}	$t=T_{19}$	S_{20}		$CC_{Demod}=0$; PacketDataSymbols
S_{20}	PhysicalLayerConfiguration	S_{21}		$CC_{Demod}=CC_{FFT}$;
S_{21}	$t=T_{21}$	S_{22}		$CC_{Demod}=0$; PacketDataSymbols
S_{22}	PhysicalLayerConfiguration	S_{23}		$CC_{Demod}=CC_{FFT}$;
S_{23}	$t=T_{23}$	S_{24}		$CC_{Demod}=0$; PacketDataSymbols PacketPilotSymbols

S_{24}	PhysicalLayerConfiguration	S_{25}		$CC_{Demod}=CC_{FFT}$;
S_{25}	$t=T_{25}$	S_{26}		$CC_{Demod}=0$; PacketDataSymbols
S_{26}	PhysicalLayerConfiguration	S_{27}		$CC_{Demod}=CC_{FFT}$
S_{27}	$t=T_{27}$	S_0		$CC_{Demod}=0$; PacketDataSymbols

Tableau 4.2 – Table de transition associée à l'activité « OFDMDemodulator ».

Cette représentation permet de décrire l'évolution de la variable CC_{Demod} ainsi que l'évolution des sorties *PacketDataSymbols* et *PacketPilotSymbols*.

- **Estimateur de canal**

Un processus d'égalisation est ensuite exécuté de manière à réduire les effets de distorsion induits par le canal de transmission. Ce traitement nécessite d'avoir effectué au préalable une phase d'estimation qui va permettre de calculer les coefficients d'égalisation associés à chaque sous-porteuse utilisée. L'estimation de canal se fait à partir de symboles appelés pilotes qui sont insérés dans la grille de ressources à des endroits bien spécifiques. Le nombre de symboles pilotes dépend donc du nombre de blocs de ressources alloués à un utilisateur. La figure 4.5 montre le positionnement de symboles pilotes dans deux blocs de ressources composant une sous-trame LTE.

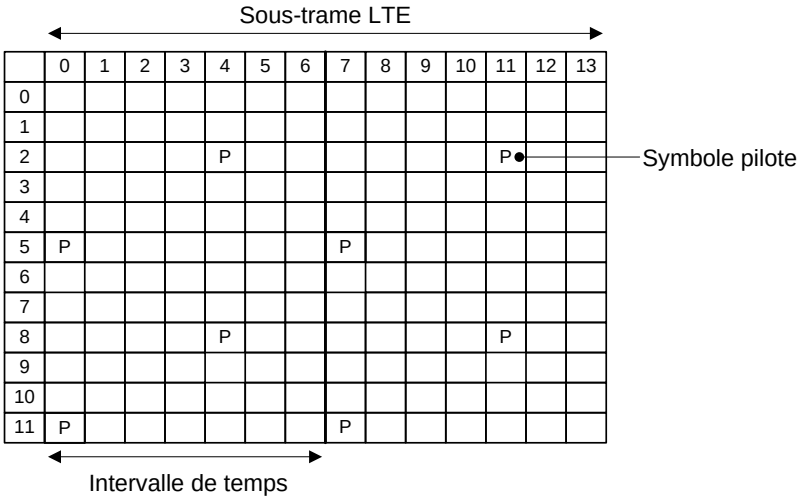

Figure 4.5 – Position des symboles pilotes dans deux blocs de ressources d'une sous-trame LTE.

Les indices horizontaux correspondent aux indices des symboles OFDM contenus dans une sous-trame LTE. Les indices verticaux font références aux indices des sous-porteuses associées aux blocs de ressources. Les symboles OFDM d'indices 0, 4, 7, 11 contiennent les symboles pilotes utilisés pour réaliser l'estimation de canal. Une première interpolation linéaire en fréquence permet ainsi de calculer les coefficients associés aux sous-porteuses utilisées pour transmettre un symbole OFDM. Une interpolation linéaire en temps permet ensuite d'estimer les coefficients associés aux sous-porteuses utilisées pour transmettre les deux ou trois symboles OFDM précédents. Le nombre de coefficients d'égalisation à calculer par l'estimateur de canal pour chaque symbole OFDM dépend donc du nombre de blocs de ressources utilisés pour transmettre des données à un utilisateur dans une sous-trame radio LTE.

Nous nous basons ici sur l'expression analytique définie dans [**103**] pour exprimer la puissance de calcul requise pour mettre en œuvre les traitements associés à cette fonction d'estimation de canal :

$$CC_{Coefs} = \frac{8 * N_f * \left(N_{pilot} + N_{data} - N_{pilot}\right) + Nb_{InterpolationTemps} * 8 * N_t * N_{data}}{T_{ProcChannelEstimator}}.$$

Cette expression permet de calculer la puissance de calcul requise, en nombre d'opérations par seconde sur des nombres réels, pour réaliser une interpolation linéaire en fréquence et $Nb_{InterpolationTemps}$ interpolations linéaires en temps. $Nb_{InterpolationTemps}$ prend la valeur deux dans le cas des symboles OFDM d'indices 0 et 7 et trois dans le cas des symboles OFDM d'indices 4 et 11. Compte tenu de l'organisation de la sous-trame LTE, nous considérons N_f=12 et N_t=4. Ces paramètres correspondent respectivement aux longueurs des filtres d'interpolation linéaire en fréquence et en temps fixés. N_{pilot} représente le nombre de symboles pilotes à traiter dans un symbole OFDM. N_{data} représente le nombre de sous-porteuses utilisées pour transmettre le signal. N_{data} et N_{pilot} dépendent donc du nombre de blocs de ressources alloués à un utilisateur dans un intervalle de temps.

Le temps de traitement $T_{ProcChannelEstimator}$ correspond à la durée qui s'écoule entre l'instant à partir duquel l'interpolation linéaire en fréquence peut démarrer et l'instant où les résultats de la dernière interpolation linéaire en temps sont disponibles. Ce temps de traitement doit être défini de manière à respecter la période d'arrivée des symboles OFDM démodulés contenant des symboles pilotes. Il doit d'autre part rendre compte de l'architecture utilisée pour réaliser ces traitements.

La figure 4.6 présente le comportement associé à l'activité « ChannelEstimator » proposée afin d'analyser la puissance de calcul à mettre en œuvre pour réaliser le calcul des coefficients d'égalisation de chaque symbole OFDM en fonction du temps de traitement $T_{ProcChannelEstimator}$ défini et du nombre de blocs de ressources contenus dans la sous-trame LTE traitée.

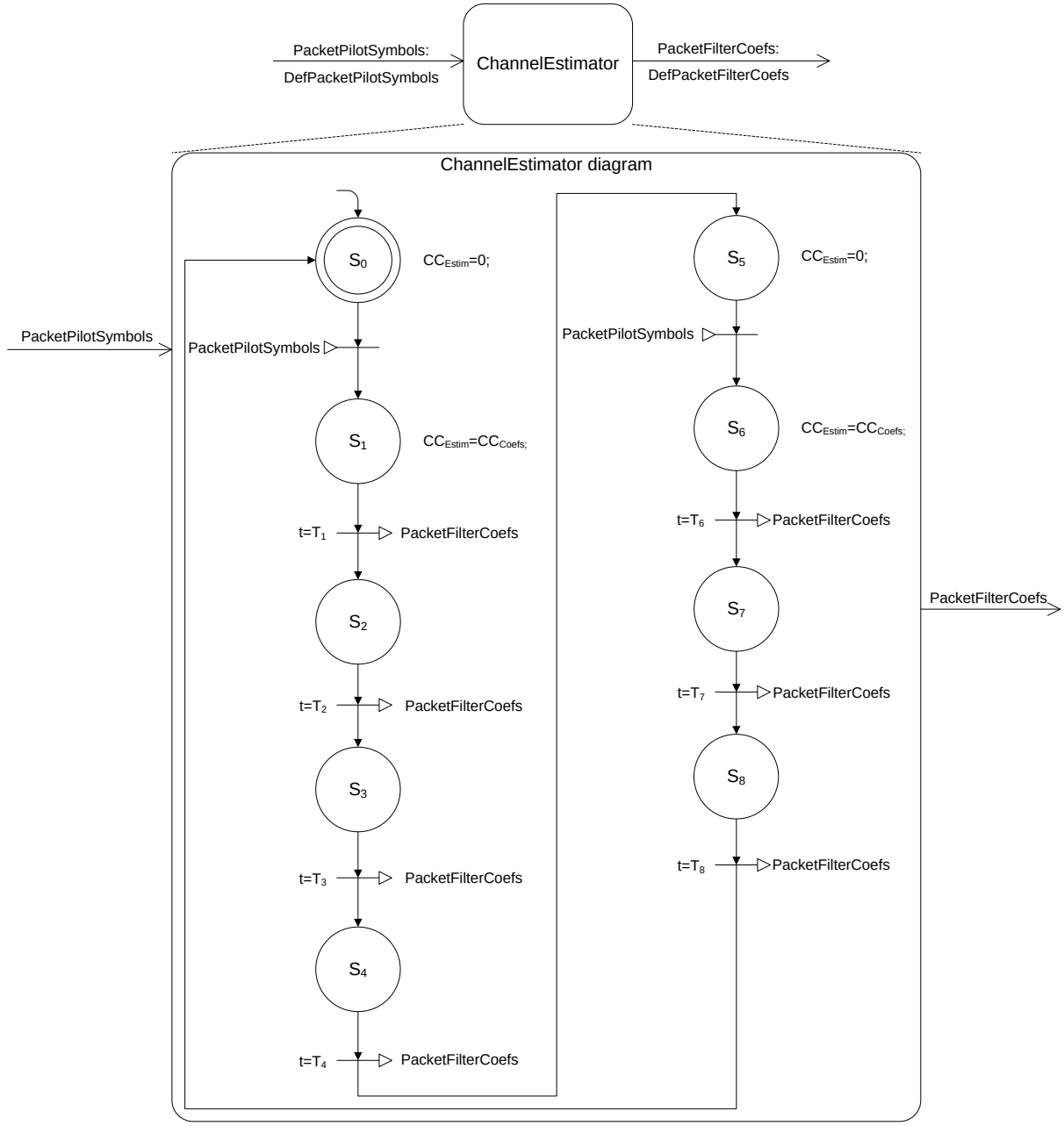

Figure 4.6 – Comportement associé à l'activité « ChannelEstimator ».

Ce comportement est proposé pour décrire les traitements successifs réalisés pour obtenir les coefficients d'égalisation associés à chaque symbole à valeur complexe contenu dans une sous-trame LTE. La variable CC_{Estim} décrit l'évolution de la puissance de calcul mise en jeu. Les états S_0 et S_5 correspondent respectivement aux états d'attente des symboles OFDM démodulés d'indices 4,11 et 0,7 contenant des symboles pilotes. Durant ces états, aucune ressource de calcul n'est utilisée. Les états S_1, S_2, S_3, S_4 et S_6, S_7, S_8 correspondent aux instants pendant lesquels sont effectués l'interpolation linéaire en fréquence et les $Nb_{InterpolationTemps}$ interpolations linéaires en temps. Les instants de démarrage et d'arrêt des traitements sont liés aux caractéristiques et aux paramétrages des ressources utilisées. Les instants de production T_x des résultats des interpolations linéaires en temps et en fréquence sont calculés à partir des instants de démarrage et du temps de traitement $T_{ProcChannelEstimator}$ considéré.

Le tableau 4.3 présente la table de transition associée à l'activité « ChannelEstimator ».

État présent	État futur		Action	
	condition	état	condition	action
S_0	PacketPilotSymbols	S_1		$CC_{Estim}=CC_{Coefs}$;
S_1	$t=T_1$	S_2		PacketFilterCoefs
S_2	$t=T_2$	S_3		PacketFilterCoefs
S_3	$t=T_3$	S_4		PacketFilterCoefs
S_4	$t=T_4$	S_5		$CC_{Estim}=0$; PacketFilterCoefs
S_5	PacketPilotSymbols	S_6		$CC_{Estim}=CC_{Coefs}$;
S_6	$t=T_6$	S_7		PacketFilterCoefs
S_7	$t=T_7$	S_8		PacketFilterCoefs
S_8	$t=T_8$	S_0		$CC_{Estim}=0$; PacketFilterCoefs

Tableau 4.3 – Table de transition associée à l'activité « ChannelEstimator ».

Cette représentation permet de décrire l'évolution de la variable CC_{Estim} ainsi que l'évolution de la sortie *PacketFilterCoefs*.

- **Égaliseur**

L'activité « Equalizer » permet ensuite de représenter la mise en œuvre du processus d'égalisation de l'effet du canal pour chaque symbole OFDM démodulé. Cette opération s'effectue sur les symboles à valeurs complexes contenus dans un symbole OFDM ne se rapportant pas à un symbole pilote. Cette fonction nécessite de réaliser une multiplication de chaque symbole à valeur complexe par le coefficient d'égalisation associé. Ceci revient à réaliser six opérations sur des nombres réels. L'expression analytique suivante permet de calculer la puissance de calcul requise, en nombre d'opérations par seconde sur des nombres réels, pour réaliser le processus d'égalisation d'un symbole OFDM démodulé :

$$CC_{Equal} = \frac{6 * (N_{data} - N_{pilot})}{T_{ProcEqualizer}}.$$

L'expression N_{data}-N_{pilot} permet d'obtenir le nombre de symboles à valeurs complexes contenus dans le symbole OFDM démodulé qu'il s'agit d'égaliser. Le temps de traitement $T_{ProcEqualizer}$ correspond à la durée qui s'écoule entre, l'instant à partir duquel le processus d'égalisation d'un symbole OFDM démodulé peut démarrer, et l'instant où les résultats de ce calcul sont disponibles. Ce temps de traitement doit être défini de manière à respecter la période d'arrivée des coefficients d'égalisation. Il doit d'autre part rendre compte de l'architecture utilisée pour réaliser ce traitement.

La figure 4.7 présente le comportement associé à l'activité « Equalizer » proposée afin d'analyser la puissance de calcul à mettre en œuvre pour réaliser la fonction d'égalisation en fonction du temps de traitement $T_{ProcEqualizer}$ défini et du nombre de symboles à valeurs complexes à égaliser au sein de chaque symbole OFDM démodulé reçu.

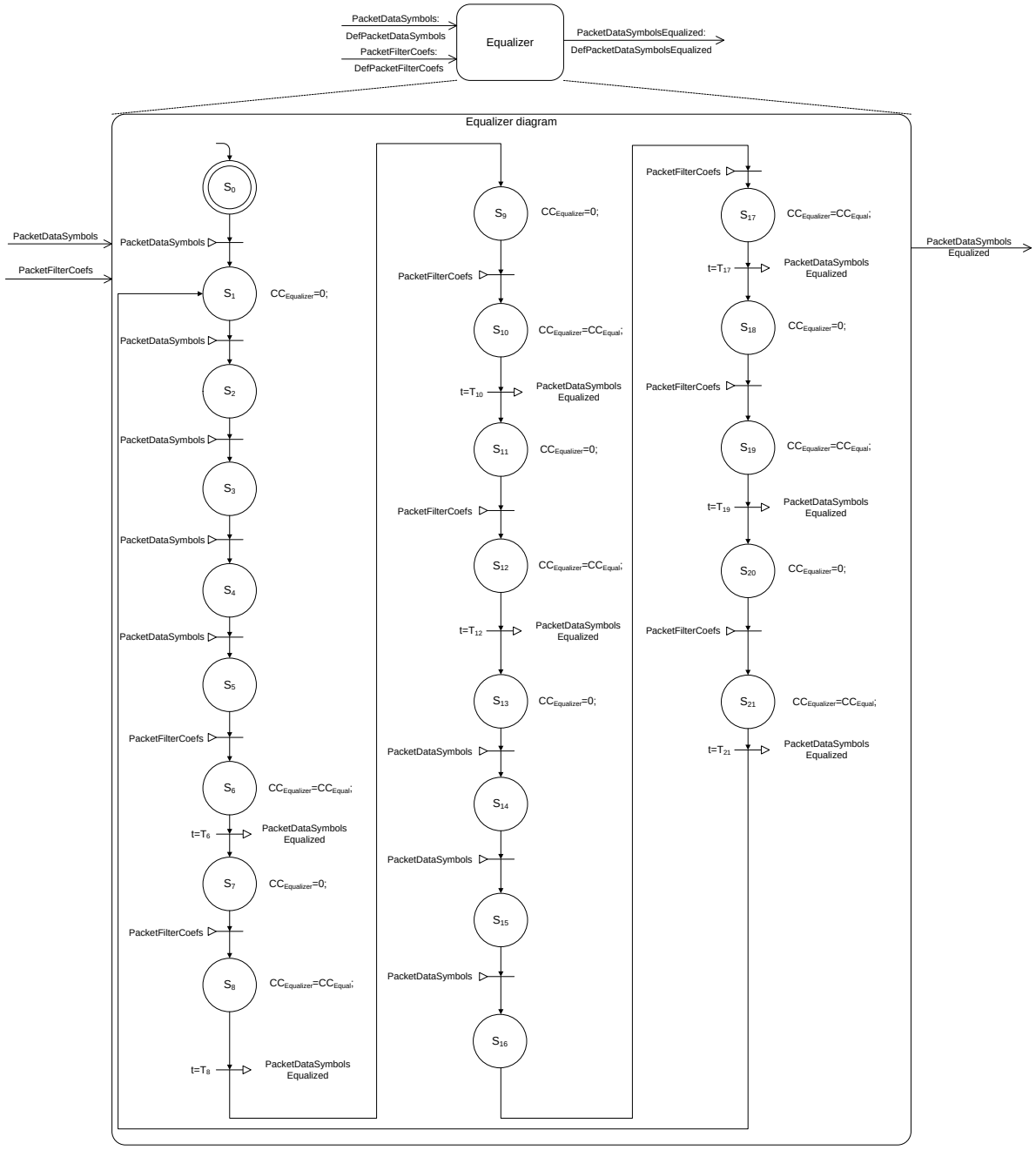

Figure 4.7 – Comportement associé à l'activité « Equalizer ».

Ce comportement est proposé pour décrire le processus d'égalisation réalisé sur les quatorze symboles OFDM d'une sous-trame radio LTE. La variable $CC_{Equalizer}$ décrit ici l'évolution de la puissance de calcul mise en jeu. Les états S_0, S_1, S_2, S_3, S_4, S_{13}, S_{14} et S_{15} correspondent aux états d'attente des symboles OFDM démodulés. Durant ces états, aucune ressource de calcul n'est utilisée. Les états S_6, S_8, S_{10}, S_{12}, S_{17} S_{19} et S_{21} traduisent eux respectivement l'activité des ressources de calcul pour exécuter le processus d'égalisation d'un symbole OFDM dès lors que le paquet de coefficients d'égalisation associé est reçu. Les instants de démarrage et d'arrêt du traitement sont liés aux caractéristiques et aux paramétrages des ressources utilisées. L'instant de production T_x du résultat du processus d'égalisation est calculé à partir de l'instant de démarrage et du temps de traitement $T_{ProcEqualizer}$ considérés.

Le tableau 4.4 présente la table de transition décrite pour l'activité « Equalizer ».

État présent	État futur		Action	
	condition	état	condition	action
S_0	PacketDataSymbols	S_1		$CC_{Equalizer}$=0;
S_1	PacketDataSymbols	S_2		
S_2	PacketDataSymbols	S_3		
S_3	PacketDataSymbols	S_4		
S_4	PacketDataSymbols	S_5		
S_5	PacketFilterCoefs	S_6		$CC_{Equalizer}$=CC_{Equal};
S_6	t=T_6	S_7		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_7	PacketFilterCoefs	S_8		$CC_{Equalizer}$=CC_{Equal};
S_8	t=T_8	S_9		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_9	PacketFilterCoefs	S_{10}		$CC_{Equalizer}$=CC_{Equal};
S_{10}	t=T_{10}	S_{11}		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_{11}	PacketFilterCoefs	S_{12}		$CC_{Equalizer}$=CC_{Equal};
S_{12}	t=T_{12}	S_{13}		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_{13}	PacketDataSymbols	S_{14}		
S_{14}	PacketDataSymbols	S_{15}		
S_{15}	PacketDataSymbols	S_{16}		
S_{16}	PacketDataSymbols	S_{17}		$CC_{Equalizer}$=CC_{Equal};
S_{17}	t=T_{17}	S_{18}		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_{18}	PacketFilterCoefs	S_{19}		$CC_{Equalizer}$=CC_{Equal};
S_{19}	t=T_{19}	S_{20}		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized
S_{20}	PacketFilterCoefs	S_{21}		$CC_{Equalizer}$=CC_{Equal};
S_{21}	t=T_{21}	S_1		$CC_{Equalizer}$=0; PacketDataSymbolsEqualized

Tableau 4.4 – Table de transition associée à l'activité « Equalizer ».

Cette représentation permet de décrire l'évolution de la variable $CC_{Equalizer}$ ainsi que l'évolution de la sortie *PacketDataSymbolsEqualized*.

- **Démodulateur QPSK, 16QAM, 64QAM**

L'activité « SymbolDemapper » permet de représenter la mise en œuvre du processus de démodulation QPSK, 16QAM et 64QAM. Cette fonction est analysée du point de vue des ressources de mémorisation nécessaires pour le stockage des résultats. L'expression analytique suivante correspond à la quantité d'informations stockée en bit à chaque fois qu'un processus de démodulation est exécuté :

$$MC_{\text{SymbolDemap}} = Nb_{\text{DemodSymbol}} * Nb_{\text{BitPerDemodSymbol}} .$$

$Nb_{DemodSymbol}$ représente le nombre de symboles à valeurs complexes reçus dans le paquet de données *PacketDataSymbolsEqualized*. $Nb_{BitPerDemodSymbol}$ fait référence au nombre de bits par symbole à valeur complexe à démoduler : deux pour une modulation QPSK, quatre pour une démodulation 16QAM et six pour une démodulation 64QAM.

La figure 4.8 présente le comportement associé à l'activité « SymbolDemapper » proposée pour analyser le coût mémoire induit par cette fonction.

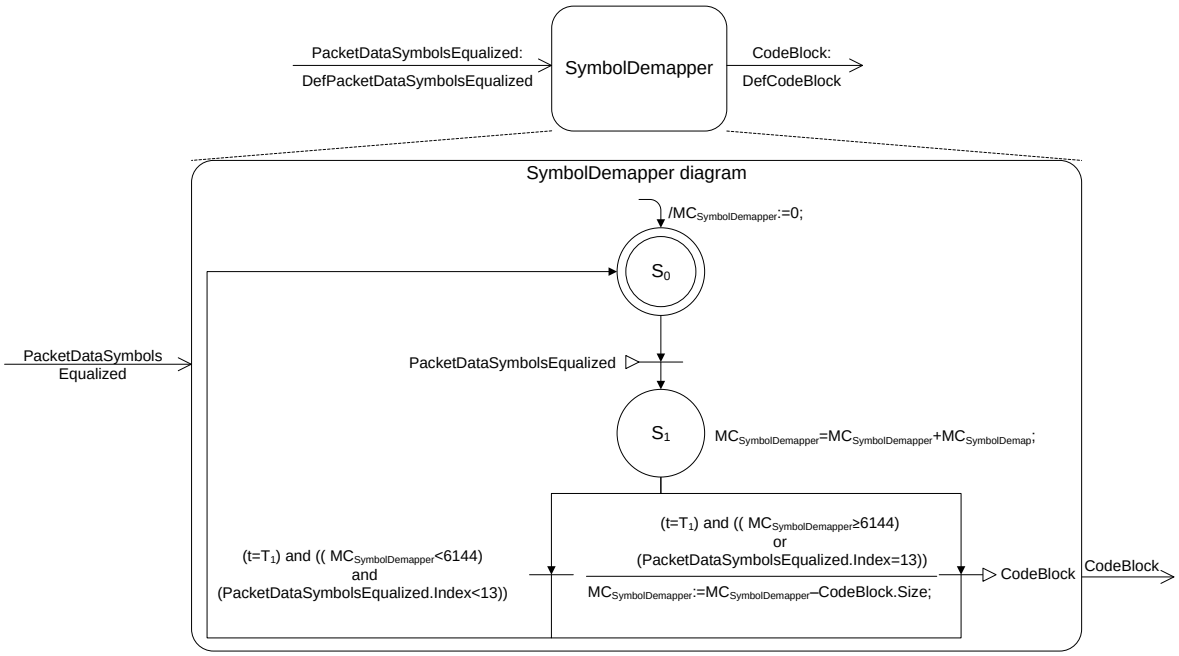

Figure 4.8 – Comportement associé à l'activité « SymbolDemapper ».

Ce comportement est proposé pour décrire le stockage successif des résultats des opérations de démodulation effectués sur chaque symbole OFDM égalisé reçu. La variable $MC_{SymbolDemapper}$ décrit l'évolution de la quantité de données stockées en mémoire. L'état S_0 correspond à l'état d'attente d'un symbole OFDM égalisé. L'état S_1 représente l'instant pendant lequel est effectué le stockage du résultat du processus de démodulation. L'instant de production T_x correspond à l'instant où un ou deux blocs de données à décoder peuvent être envoyés sur la sortie *CodeBlock*. Un ou plusieurs paquet de données *CodeBlock* sont produits dès lors que 6144 bits ont pu être démodulés ou que les quatorze symboles OFDM de la sous-trame LTE ont été démodulés.

Le tableau 4.5 présente la table de transition associée à l'activité « SymbolDemapper ».

État présent	État futur		Action	
	condition	état	condition	action
S_0	PacketDataSymbols Equalized	S_1		$MC_{SymbolDemapper}$= $MC_{SymbolDemapper}$+ $MC_{SymbolDemap}$;
S_1	$t=T_1$	S_0	($MC_{SymbolDemapper}\geq 6144$) or (PacketDataSymbols Equalized.Index=13) ($MC_{SymbolDemapper}<6144$) and (PacketDataSymbols Equalized.Index<13)	CodeBlock $MC_{SymbolDemapper}$:= $MC_{SymbolDemapper}$– CodeBlock.Size;

Tableau 4.5 – Table de transition associée à l'activité « SymbolDemapper ».

Cette représentation permet de décrire l'évolution de la variable $MC_{SymbolDemapper}$ ainsi que l'évolution de la sortie *CodeBlock*.

- **Décodeur de canal**

L'activité « TurboDecoder » permet d'analyser la puissance de calcul à mettre en œuvre pour réaliser la fonction de décodage canal en utilisant un décodeur turbo. Le décodeur turbo doit être capable de décoder un bloc de données dont la taille maximale peut être de 6144 bits. Nous considérons ici que le décodage de canal est réalisé en utilisant l'algorithme MAP (*Maximum A Posteriori*) présenté dans [**104**]. Pour évaluer la puissance de calcul requise pour décoder chaque bloc de données, nous utilisons l'expression analytique suivante :

$$CC_{TurboD} = \frac{Nb_{BitsCodeBlock} * Nb_{Loop} * 132}{T_{ProcTurboDecoder}}.$$

Elle permet de calculer la puissance de calcul requise, en nombre d'opérations par seconde, en fonction de la taille du bloc de données à décoder et du temps de traitement $T_{ProcTurboDecoder}$ défini. Ce temps correspond à la durée qui s'écoule entre, l'instant à partir duquel le décodage du bloc de données peut démarrer, et l'instant où les résultats du calcul sont disponibles. Ce temps de traitement doit être défini de manière à respecter la période d'arrivée des blocs de données à décoder. Il doit d'autre part rendre compte de l'architecture utilisée pour réaliser ce traitement. Dans cette expression, $Nb_{BitsCodeBlock}$ représente le nombre de bits contenus dans le bloc de données à décoder. Nb_{Loop} représente le nombre d'itérations du turbo décodeur fixé pour réaliser le décodage du bloc de données.

La figure 4.9 présente le comportement associé à l'activité « TurboDecoder » proposée afin d'analyser la puissance de calcul à mettre en œuvre pour réaliser la fonction de turbo décodage en fonction du temps de traitement $T_{ProcTurboDecoder}$ défini, de la quantité de données contenue dans le bloc à décoder, et du nombre d'itérations du turbo décodeur à effectuer.

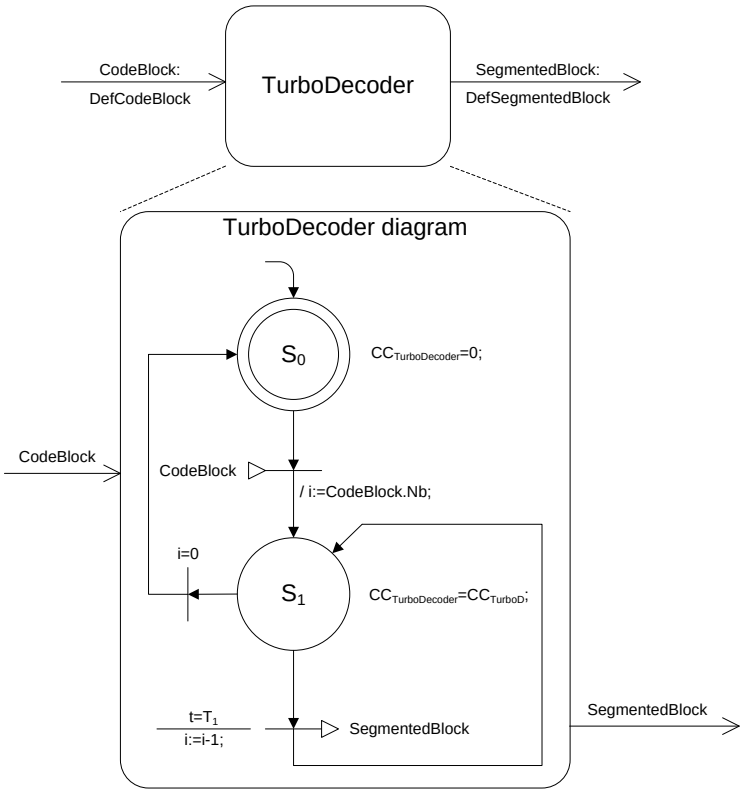

Figure 4.9 – Comportement associé à l'activité « TurboDecoder ».

Ce comportement est proposé pour décrire les traitements successifs réalisés sur les blocs de données envoyés par l'activité « TurboDecoder ». La variable $CC_{TurboDecoder}$ décrit l'évolution de la puissance de calcul mise en jeu. L'état S_0 représente l'état d'attente des blocs de données via la relation *CodeBlock*. Deux blocs de données peuvent au maximum être contenus dans une transaction *CodeBlock*. Durant cet état, aucune ressource de calcul n'est utilisée. L'état S_1 correspond à l'instant pendant lequel est effectué le turbo décodage de l'information contenue dans un bloc de données reçu. Les instants de démarrage et d'arrêt du traitement sont liés aux caractéristiques et aux paramétrages des ressources utilisées. L'instant de production T_x du résultat de ce traitement est calculé à partir de l'instant de démarrage et du temps de traitement $T_{ProcTurboDecoder}$ considérés.

Le tableau 4.6 présente la table de transition associée à l'activité « TurboDecoder ».

État présent	État futur		Action	
	condition	état	condition	action
S_0	CodeBlock	S_1		i:=CodeBlock.Nb $CC_{TurboDecoder}$= CC_{TurboD}
S_1	$t=T_1$ i=0	S_1 S_0		i:= i-1 SegmentedBlock $CC_{TurboDecoder}$=0

Tableau 4.6 – Table de transition associée à l'activité « TurboDecoder ».

Cette représentation permet de décrire l'évolution de la variable $CC_{TurboDecoder}$ ainsi que l'évolution de la sortie *SegmentedBlock*.

- **Fonction de reconstitution de blocs de transport**

La dernière activité proposée se rapporte à la fonction d'assemblage des paquets de données décodés de manière à reconstituer le bloc de transport devant être transmis à la couche MAC du terminal mobile. Cette fonction est analysée du point de vue des ressources de mémorisation nécessaires pour le stockage des données reçues. L'expression suivante représente la quantité d'informations stockée en bit à chaque fois qu'un nouveau bloc de données décodé est reçu :

$$MC_{\text{TBReassemb}} = Nb_{\text{BitSegmentedBlock}} \quad .$$

$Nb_{BitsSegmentedBlock}$ représente le nombre de bits contenus dans le bloc de données décodé reçu.

La figure 4.10 présente le comportement associé à l'activité « TransportBlockReassembly » proposée pour analyser le coût mémoire induit par cette fonction.

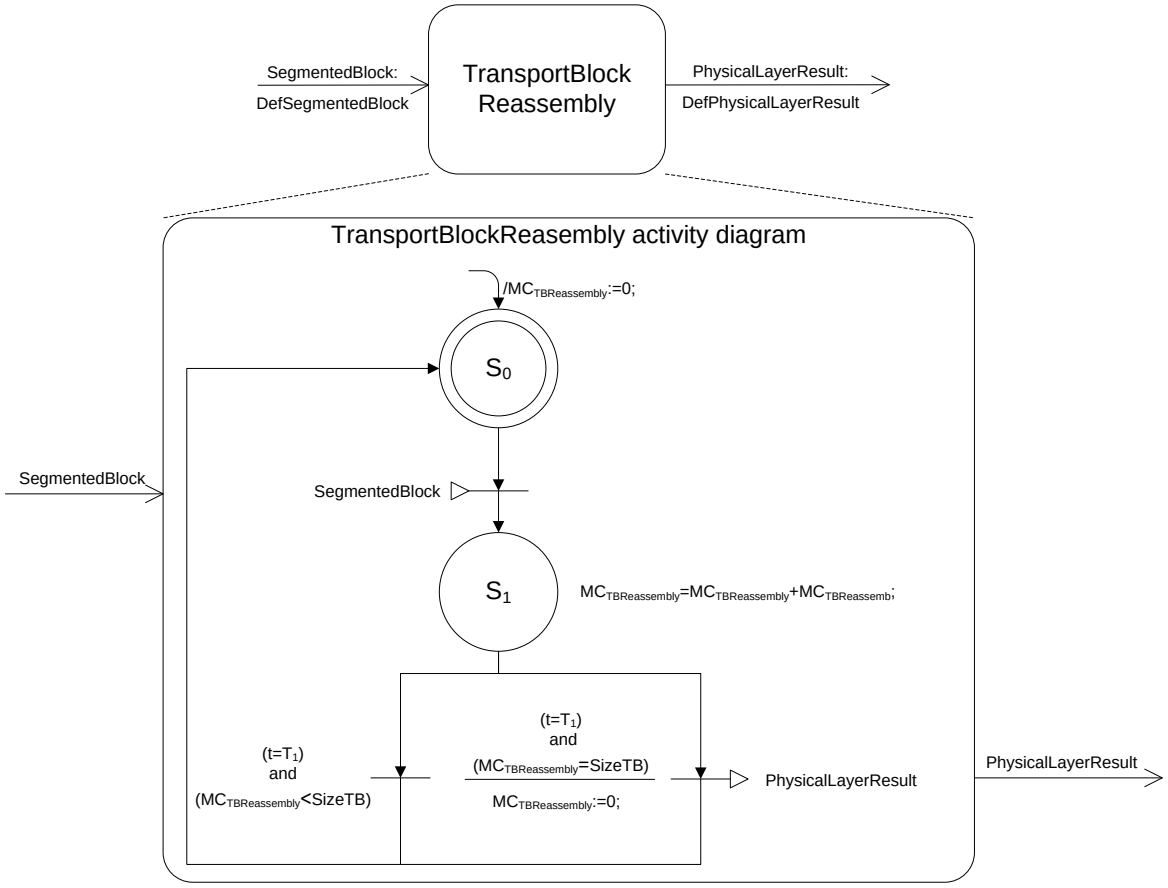

Figure 4.10 – Comportement associé à l'activité « TransportBlockReassembly ».

Ce comportement est proposé pour décrire le stockage successif des paquets de données segmentés envoyés par l'activité « TurboDecoder ». La variable $MC_{TBReassembly}$ décrit l'évolution de la quantité de données stockées en mémoire. L'état S_0 représente l'état d'attente d'un bloc segmenté. L'état S_1 représente l'instant pendant lequel est effectué le stockage du paquet de données reçu. L'instant T_1 peut correspondre à l'instant de production d'une transaction *PhysicalLayerResult* si un bloc de transport est reconstitué et peut donc être transmis à la couche MAC.

Le tableau 4.7 présente la table de transition associée à l'activité « TransportBlockReassembly ».

État présent	État futur		Actions	
	condition	état	condition	actions
S_0	SegmentedBlock	S_1		$MC_{TBReassembly}= MC_{TBReassembly} +MC_{TBReassemb}$;
S_1	$t=T_1$	S_0	$MC_{TBReassembly} =SizeTB$ $MC_{TBReassembly} <SizeTB$	PhysicalLayerResult $MC_{TB}:=0;$

Tableau 4.7 – Table de transition associée à l'activité « TransportBlockReassembly ».

Cette représentation permet de décrire l'évolution de la variable $MC_{TBReassembly}$ ainsi que l'évolution de la sortie *PhysicalLayerResult*.

- **Saisie du modèle**

Dans le cadre de notre approche, il s'agit ensuite de s'appuyer sur les tables de transition définies pour chacune des activités considérées pour opérer le paramétrage du modèle d'exécution générique proposé. La figure 4.11 présente le modèle obtenu avec l'outil CoFluent Studio pour réaliser le travail d'exploration d'architectures pour le système considéré.

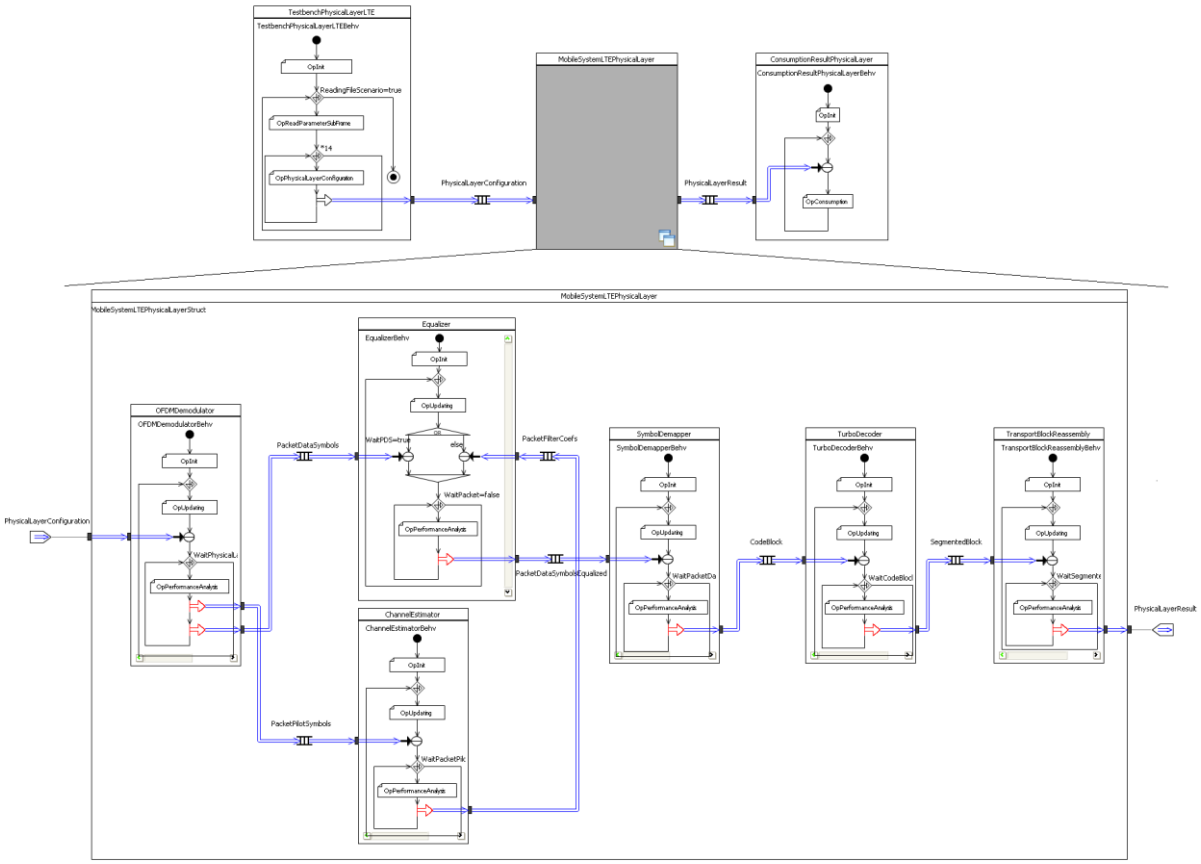

Figure 4.11 – Saisie dans l'outil CoFluent Studio du modèle d'évaluation des performances de la couche physique du récepteur LTE.

Le même modèle est utilisé pour capturer chacune des activités précédemment décrites. Pour l'activité « OFDMDemodulator », le modèle est étendu au cas où deux sorties sont nécessaires. Pour l'activité « Equalizer », le modèle est étendu au cas où deux entrées doivent être considérées. La partie haute de la figure 4.11 fait apparaître l'environnement de test proposé. La fonction « TestbenchPhysicalLayerLTE » est utilisée pour traduire l'envoi des symboles OFDM contenus dans une sous-trame LTE. Une transaction *PhysicalLayerConfiguration* comporte les informations concernant le nombre de blocs de ressources contenus dans la sous-trame considérée, la taille de la FFT à réaliser, le type de démodulation à exécuter et le nombre d'itérations du turbo décodeur à effectuer. Ces informations sont ensuite transmises à travers les différentes files de messages et permettent de prendre en considération les différentes configurations de sous-trames radio LTE. Il est alors possible d'observer la puissance de calcul et les ressources de mémorisation requises en fonction de la configuration de la sous-trame LTE traitée.

Dans le cadre du travail d'exploration présenté dans les trois parties suivantes, l'environnement de test est configuré pour envoyer successivement trois sous-trames radio LTE avec les paramètres présentés dans le tableau 4.8.

N° de la sous-trame LTE	Délai d'envoi (ms)	Nb de blocs de ressources	Taille FFT	Type de démodulation	Nb itérations du turbo décodeur
1	1	12	128	QPSK	1
2	1	50	512	16QAM	2
3	1	200	2048	64QAM	1

Tableau 4.8 – Configurations des sous-trames définies pour évaluer les performances de l'architecture analysée.

La fonction « ConsumptionResultPhysicalLayer » permet elle de consommer les résultats envoyés par la fonction « MobileSystemLTEPhysicalLayer ». La partie basse de la figure 4.11 présente le modèle obtenu pour réaliser le travail d'exploration d'architectures pour ce système. Les algorithmes définis à partir des tables de transition ont ensuite été saisis au sein des opérations élémentaires *OpPerformanceAnalysis* de chaque fonction. Le modèle généré à l'aide de l'outil CoFluent Studio représente 3842 lignes de code SystemC dont 22% sont insérées automatiquement par l'outil.

4.1.3 Exploration architecturale : 1ère configuration

La première architecture évaluée correspond à une architecture basée sur des ressources matérielles spécifiques. Chacune de ces ressources matérielles est ici utilisée pour réaliser les traitements associés à une des fonctions de la couche physique du récepteur LTE. La mise en œuvre analysée est illustrée sur la figure 4.12.

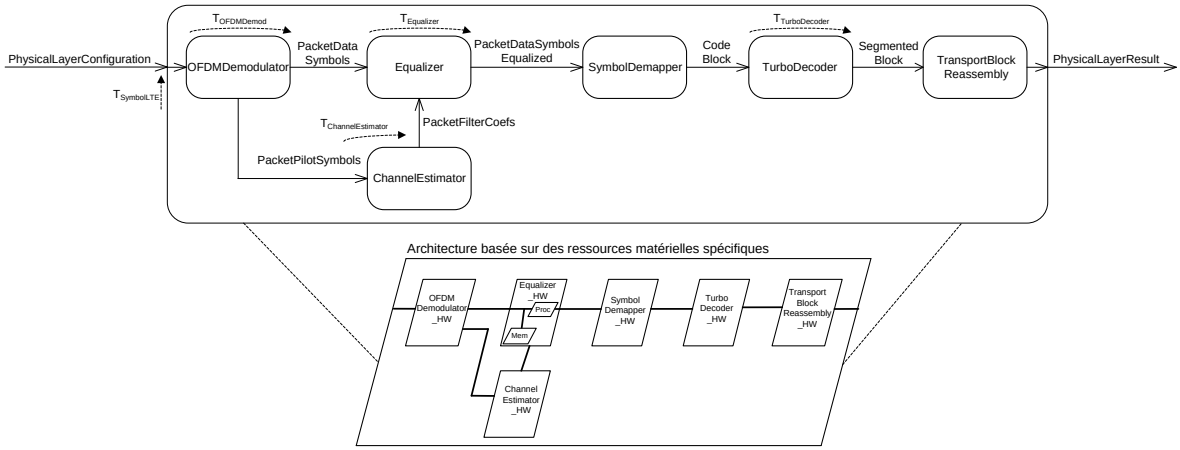

Figure 4.12 – Architecture évaluée basée sur l'utilisation de ressources matérielles spécifiques.

Les communications entre ressources correspondent ici à des communications point-à-point pour lesquelles les ressources de mémorisation sont situées au sein des ressources considérées. Comme illustrée par le bloc *Equalizer_HW*, chaque ressource s'organise autour d'une unité de traitement et d'une ressource de mémorisation. Pour une telle mise en œuvre, toutes les ressources de calcul considérées peuvent potentiellement évoluer en parallèle. Il s'agit alors d'exprimer les contraintes de temps devant être respectées avec cette mise en œuvre pour réaliser les différents traitements considérés.

Pour limiter les contraintes de temps possibles, nous ne considérons pas ici d'évolution pipeline des ressources.

Pour l'activité « OFDMDemodulator », la contrainte de temps $T_{OFDMDemod}$ à respecter est liée à la période d'arrivée des symboles OFDM contenus dans une sous-trame LTE qui est de $T_{SymbolLTE}$=71428 ns. Le calcul de la transformée de Fourier rapide doit donc s'effectuer pendant cet intervalle de temps soit :

$$T_{OFDMDemod} \leq T_{Symbol\ LTE}.$$

Pour les activités « ChannelEstimator » et « Equalizer », trois itérations des traitements qui leur sont associées sont à réaliser pendant l'intervalle de temps $T_{SymbolLTE}$ suite à la réception d'une transaction *PacketPilotSymbols* d'indices 0 ou 7 et quatre dans le cas d'indices 4 ou 11. Les temps de traitement pour les activités « ChannelEstimator » et « Equalizer » doivent donc être définis de manière à respecter les contraintes suivantes avec N le nombre d'itérations évoqué précédemment :

$$N * T_{ChannelEstimator} + T_{Equalizer} \leq T_{SymbolLTE},$$

$$T_{Equalizer} \leq T_{ChannelEstimator}.$$

Pour l'activité « TurboDecoder », au maximum deux blocs de données doivent être décodés suite à la démodulation d'un symbole OFDM. Le temps alloué pour réaliser le turbo décodage d'un bloc de données doit alors respecter la contrainte de temps suivante :

$$T_{TurboDecoder} \leq \frac{T_{Equalizer}}{2}.$$

Une fois ces différentes contraintes de temps fixées, il est possible d'analyser l'évolution de la puissance de calcul et du coût mémoire pour les différentes ressources considérées. Cette analyse est menée en évaluant différentes configurations de sous-trames radio LTE et en comparant différents paramétrages des ressources de traitement. L'influence de ce paramétrage est considérée en appliquant différentes valeurs au temps de traitement T_{Proc} associé à chaque activité. Ces temps de traitement sont définis de manière à respecter les contraintes de temps fixées précédemment.

Les figures 4.13, 4.14 et 4.15 donnent des exemples de résultats de simulation obtenus à partir du modèle capturé. Les contraintes de temps maximales sont ici appliquées pour les temps de traitements associés à chacune des fonctions considérées:

$$T_{ProcOFDMDemod} = 71428\ ns, T_{ProcC\ hannelEsimator} = 14285\ ns,$$

$$T_{ProcEqualizer} = 14285\ ns, T_{ProcTurboDecoder} = 7142\ ns.$$

Ces simulations ont été réalisées sur un poste de travail doté d'un processeur Intel Core 2 duo ayant une fréquence d'horloge de 2,66 GHz. L'exécution du modèle nécessite un temps de simulation de l'ordre de 140 ms. L'instrumentation proposée au sein de l'outil pour permettre l'observation des résultats présentés sur les figures 3.14, 3.15 et 3.16 n'entraîne pas ici de surcoût sur le temps de simulation.

La figure 4.13 montre l'évolution de la puissance de calcul nécessaire à la réalisation du processus de turbo décodage pour les trois configurations de sous-trames radio LTE.

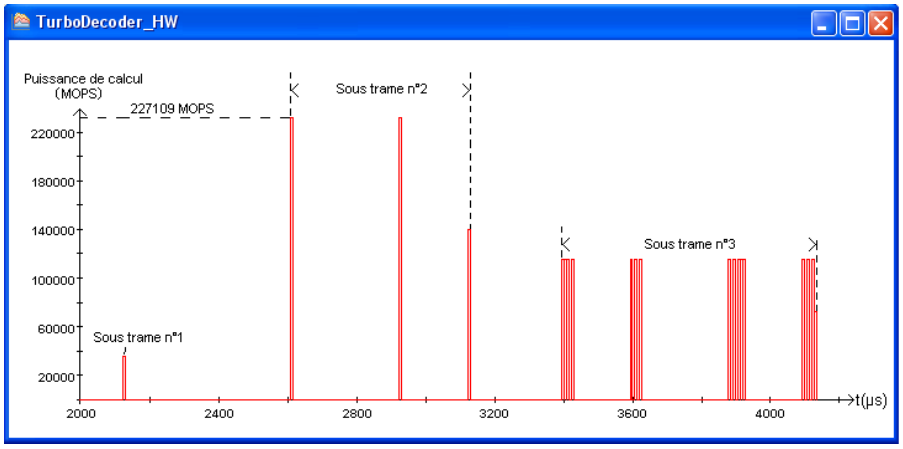

Figure 4.13 – 1ère configuration : évolution de la puissance de calcul requise pour le turbo décodeur.

La figure 4.14 montre le coût mémoire associé au stockage des données en bit pour réaliser la fonction de démodulation avec ce scénario de fonctionnement.

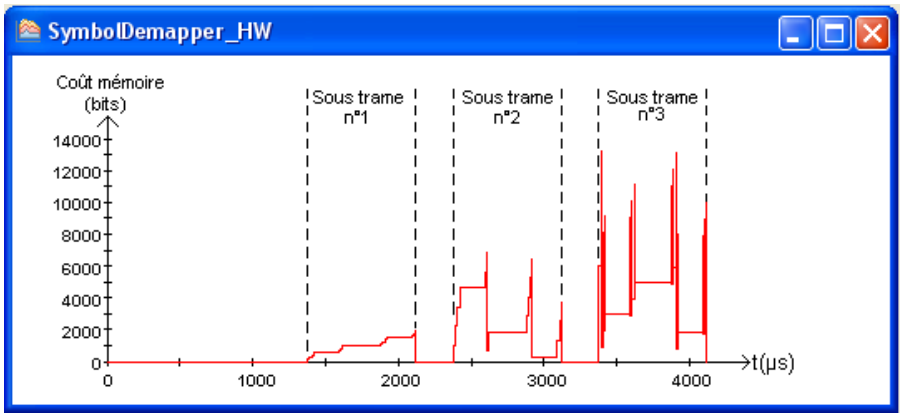

Figure 4.14 – 1ère configuration : évolution du coût mémoire pour la fonction de démodulation.

La figure 4.15 montre elle la puissance globale de calcul observée pour exécuter l'ensemble des traitements associés à la couche physique du standard LTE avec le jeu de paramètres considéré et les configurations de sous-trames LTE évaluées.

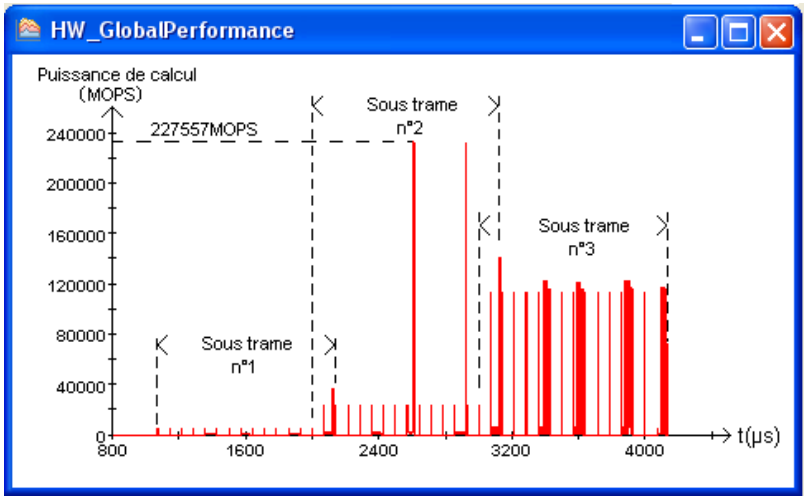

Figure 4.15 – 1ère configuration : évolution de la puissance de calcul requise pour le récepteur LTE.

Les trois figures précédentes présentent des exemples d'observation pour un jeu de paramètres donné et pour trois configurations de sous-trames radio LTE. Ces résultats servent de point de comparaison avec d'autres résultats pouvant être obtenus en appliquant différentes valeurs aux paramètres T_{Proc} associés à chaque fonction, et ce afin de déterminer un compromis satisfaisant.

4.1.4 Exploration architecturale : 2ème configuration

La deuxième architecture évaluée correspond à une architecture basée sur un processeur logiciel unique qui réaliserait l'ensemble des traitements associés à la couche physique du standard LTE. Cette mise en œuvre est illustrée sur la figure 4.16.

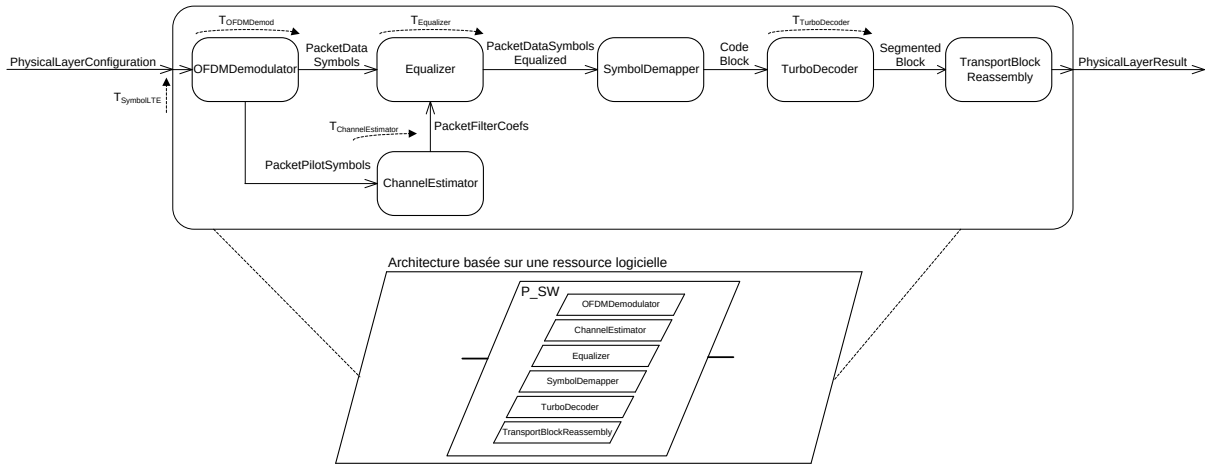

Figure 4.16 – Architecture évaluée basée sur l'utilisation d'un processeur logiciel.

Pour une telle mise en œuvre, les fonctions considérées s'exécutent séquentiellement au sein du processeur. Il s'agit alors de rendre compte de cette exécution en tenant compte des contraintes de temps à considérer. Pour ce type d'implantation, l'exécution de l'ensemble des traitements associés à chacune de ces fonctions doit donc s'effectuer pendant l'intervalle de temps $T_{SymbolLTE}$ qui sépare l'arrivée de deux symboles OFDM consécutifs. L'expression suivante traduit cette contrainte :

$$T_{OFDMDemod} + T_{ChannelEstimator} + T_{Equalizer} + T_{TurboDecoder} \leq T_{SymbolLTE} .$$

Il s'agit alors d'évaluer différentes répartitions de contraintes de temps sur les différentes fonctions de traitement considérées. Á partir des contraintes de temps fixées, il convient ensuite d'analyser différents pourcentages d'utilisation de ces temps pour réaliser les traitements associés à chaque fonction considérée.

La figure 4.17 permet d'observer l'évolution de la puissance globale de calcul mise en œuvre par le processeur logiciel avec la séquence de sous-trames radio LTE décrite dans le tableau 4.8 en appliquant la configuration présentée dans le tableau 4.9.

Fonctions	OFDMDemodulator	ChannelEstimator	Equalizer	TurboDecoder
Répartition de la contrainte de temps $T_{SymbolLTE}$	10%	10%	10%	70%
Taux d'utilisation pour le traitement	100%	100%	100%	100%

Tableau 4.9 – 2ème configuration : paramétrage du processeur logiciel.

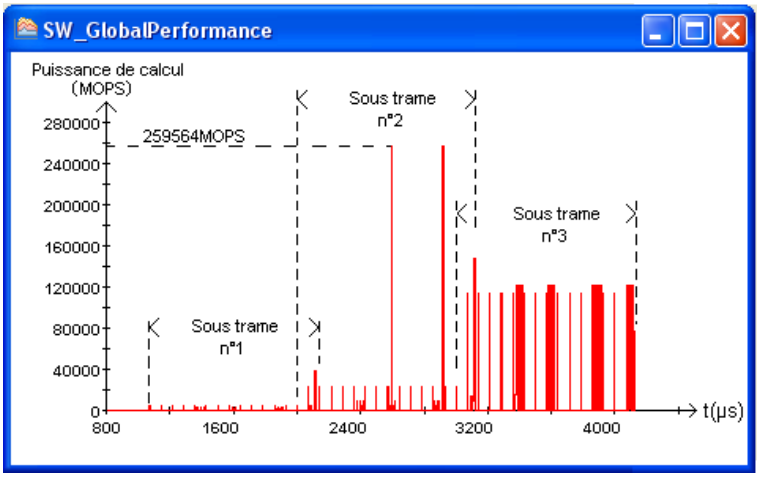

Figure 4.17 – 2ème configuration : évolution de la puissance de calcul requise pour le récepteur LTE.

Á partir des résultats présentés sur les figures 4.15 et 4.17, il est alors possible de comparer les puissances de calcul requises par l'architecture composée de ressources matérielles spécifiques et par le processeur logiciel. Sur les exemples proposés, le paramétrage des ressources a été défini de manière à maximiser le temps de traitement alloué à la fonction de turbo décodage qui nécessite le plus de puissance de calcul. Selon la répartition considérée des contraintes de temps, une puissance globale de calcul maximale de l'ordre de 227 GOPS est requise pour l'architecture reposant sur un ensemble de ressources matérielles spécifiques tandis qu'une puissance globale de calcul de l'ordre de 259 GOPS doit être supportée par le processeur logiciel.

4.1.5 Exploration architecturale : 3ème configuration

La troisième architecture évaluée correspond à une architecture hétérogène basée sur l'utilisation d'un processeur logiciel et de ressources matérielles spécifiques selon la répartition illustrée sur la figure 4.18.

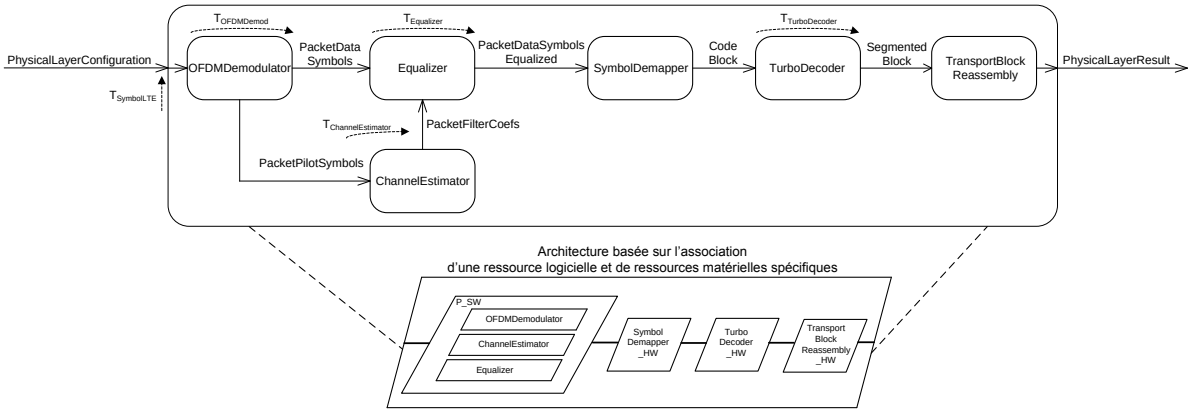

Figure 4.18 – Architecture évaluée basée sur l'utilisation d'un processeur logiciel et de ressources matérielles spécifiques.

Pour ce type d'implantation, l'exécution de l'ensemble des traitements associés aux fonctions de démodulation OFDM, d'estimation de canal et d'égalisation doivent s'effectuer pendant l'intervalle de temps $T_{SymbolLTE}$ qui sépare l'arrivée de deux symboles OFDM consécutifs. L'expression suivante traduit cette contrainte :

$$T_{OFDMDemod} + T_{ChannelEstimator} + T_{Equalizer} \leq T_{SymbolLTE} .$$

Il s'agit alors comme pour la configuration précédente d'évaluer différentes répartitions de contraintes de temps sur les différentes fonctions de traitement considérées. Á partir des contraintes de temps fixées, il convient ensuite d'analyser différents pourcentages d'utilisation de ces temps pour réaliser les traitements associés à chaque fonction considérée. Le tableau 4.9 présente le paramétrage considéré pour le processeur logiciel.

Fonctions	OFDMDemodulator	ChannelEstimator	Equalizer
Répartition de la contrainte de temps $T_{SymbolLTE}$	10%	70%	20%
Taux d'utilisation pour le traitement	100%	100%	100%

Tableau 4.10 – 3ème configuration : paramétrage du processeur logiciel.

Le paramétrage de la ressource matérielle spécifique réalisant la fonction de turbo décodage se fait en fonction du paramétrage proposé pour le processeur logiciel. Le traitement d'au maximum deux blocs de données doit en effet pouvoir être effectué par le

turbo décodeur pendant que s'exécute sur le processeur logiciel un processus d'estimation de canal et d'égalisation d'un symbole OFDM. Le temps de traitement $T_{ProcTurboDecoder}$ pour réaliser le turbo décodage d'un bloc de données a ici été défini de manière à utiliser l'ensemble du temps disponible, soit 8035 ns.

Les figures 4.19 et 4.20 présentent respectivement les puissances de calcul devant être supportées par le processeur logiciel et la ressource matérielle spécifique avec le paramétrage considéré.

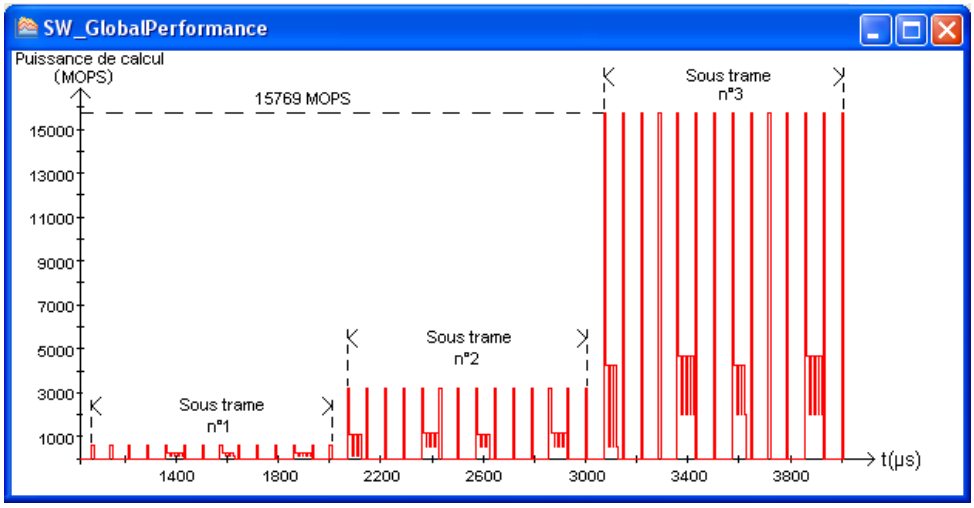

Figure 4.19 – 3ème configuration : évolution de la puissance de calcul requise par le processeur logiciel.

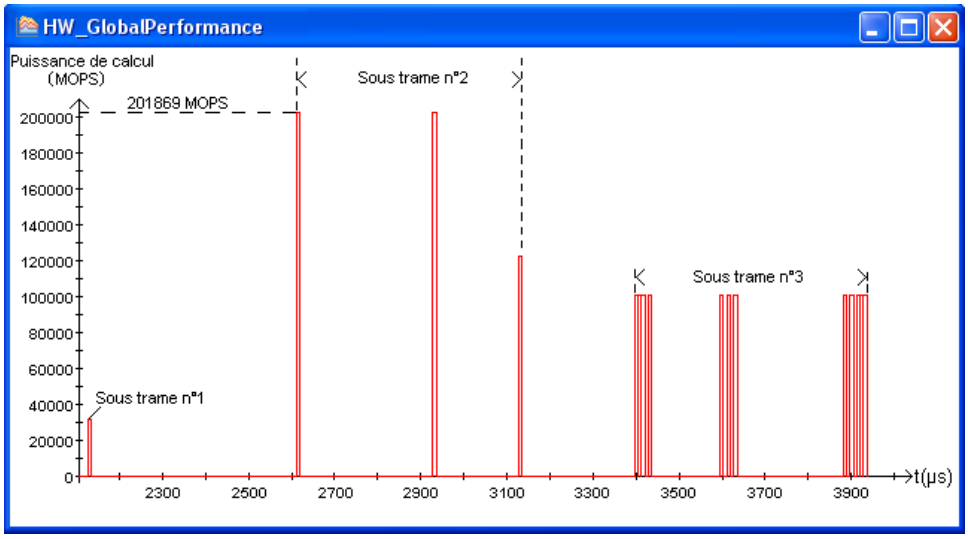

Figure 4.20 – 3ème configuration : évolution de la puissance de calcul requise par la ressource matérielle spécifique réalisant le turbo décodage.

Cette observation permet de constater une nette diminution de la puissance de calcul à mettre en œuvre par le processeur logiciel par rapport à l'expérimentation précédente. La puissance de calcul est ici divisée par un facteur seize avec le paramétrage considéré. La répartition des contraintes temporelles s'est faite ici en allouant le plus de temps possible à la ressource matérielle spécifique qui assure la charge de calcul la plus importante.

4.1.6 Bilan

Ce cas d'étude a permis d'illustrer les possibilités offertes par notre approche. Nous avons ainsi pu mettre en avant l'intérêt qu'il pouvait y avoir à s'appuyer sur un modèle d'exécution générique pour évaluer les performances de différentes solutions architecturales. En effet, le modèle obtenu permet, par simple paramétrage, d'évaluer pour différentes configurations de sous-trames radio LTE différentes configurations d'architectures et différentes répartitions des contraintes temporelles. Les différents résultats de simulation pouvant être observés en exécutant ce modèle permettent de guider le concepteur dans ses choix. L'utilisation du modèle d'exécution générique proposé permet de réduire le temps de création des modèles par rapport à une approche sans modèle de référence. La technique de simulation utilisée permet de plus de réduire les temps de simulation nécessaires à l'exécution du modèle. Par exemple, pour la première configuration de sous-trame radio LTE, soixante-trois transactions sont initiées durant la simulation. Á un niveau de granularité inférieure, par exemple au niveau des symboles à valeurs complexes pour les relations *PhysicalLayerConfiguration*, *PacketDataSymbols*, *PacketPilotSymbols*, *PacketFilterCoefs* et *PacketDataSymbolsEqualized* et au niveau du bit pour les relations *CodeBlock*, *SegmentedBlock* et *TransportBlock*, une dizaine de millier de transactions auraient alors dû être initiées selon nos estimations. Le temps de simulation qui aurait été nécessaire pour évaluer les ressources de calcul nécessaires à la réalisation de cette sous-trame radio LTE aurait ainsi été multiplié par un facteur d'ordre 100.

4.2 Modélisation et dimensionnement d'un terminal mobile adaptatif multiservice

4.2.1 Contexte de l'étude

Les futurs terminaux mobiles devront être capables d'adapter dynamiquement leurs interfaces de communication en fonction des besoins des utilisateurs et des possibilités offertes par l'environnement radio. Cette évolution doit permettre de garantir à l'utilisateur un niveau de qualité de service satisfaisant quels que soient l'endroit où il se trouve et les services qu'il demande. Le comportement d'un tel système va donc être de plus en plus difficile à évaluer avec l'intégration de cette nouvelle capacité d'adaptation. L'objectif de cette étude est donc dans un premier temps de montrer comment il est possible de définir très tôt dans le processus de conception une description exécutable du système considéré afin d'analyser son comportement en fonction des nombreux scénarios de fonctionnement envisageables. Nous proposons ensuite d'utiliser cette description pour dimensionner les ressources de calcul nécessaires pour l'implantation des standards de communication supportés. Nous nous appuyons sur l'approche et la technique de modélisation proposées pour intégrer l'observation des propriétés non fonctionnelles au sein de ce modèle.

4.2.2 Spécification de l'étude de cas

Le cas d'étude que nous considérons ici se veut représentatif des nouvelles caractéristiques des futurs terminaux mobiles introduites dans la partie 1.1.3. Le système étudié propose plusieurs standards de communication. Il supporte ici deux types de technologies d'accès radio ou RAT (*Radio Access Technology*) que sont l'UTRA et le Wi-Fi. Ces technologies permettent respectivement de se connecter au réseau cellulaire UMTS et à des réseaux locaux. Le choix s'est orienté vers ces deux RAT du fait de leur appartenance à deux catégories de réseaux sans fil différents. Le système supporte aussi trois types d'application que sont la communication vocale, la navigation Internet et la

lecture de vidéo en continu. Ce choix s'est fait de manière à considérer des applications représentatives des différents besoins en termes de qualité de service introduits dans la partie 1.1.1. Le système est enfin capable de gérer de façon autonome ses interfaces de communication. Pour cela, il supporte les différents services relatifs au concept de la radio cognitive. Le système a en effet accès à des informations concernant le niveau de qualité de service proposé à l'utilisateur et sur les réseaux d'accès radio disponibles dans son environnement. Á partir de ces informations, il est en mesure de prendre des décisions concernant les RAT à utiliser pour obtenir un niveau de qualité de service satisfaisant pour les applications employées par l'utilisateur. Il supporte enfin les mécanismes nécessaires à la sélection de la RAT à utiliser. La figure 4.21 présente sous la forme d'un diagramme d'activités la structure du terminal mobile adaptatif multiservice étudié. Cette figure fait aussi apparaître les deux entités « User » et « Network environment » considérées pour représenter l'environnement dans lequel évolue le système.

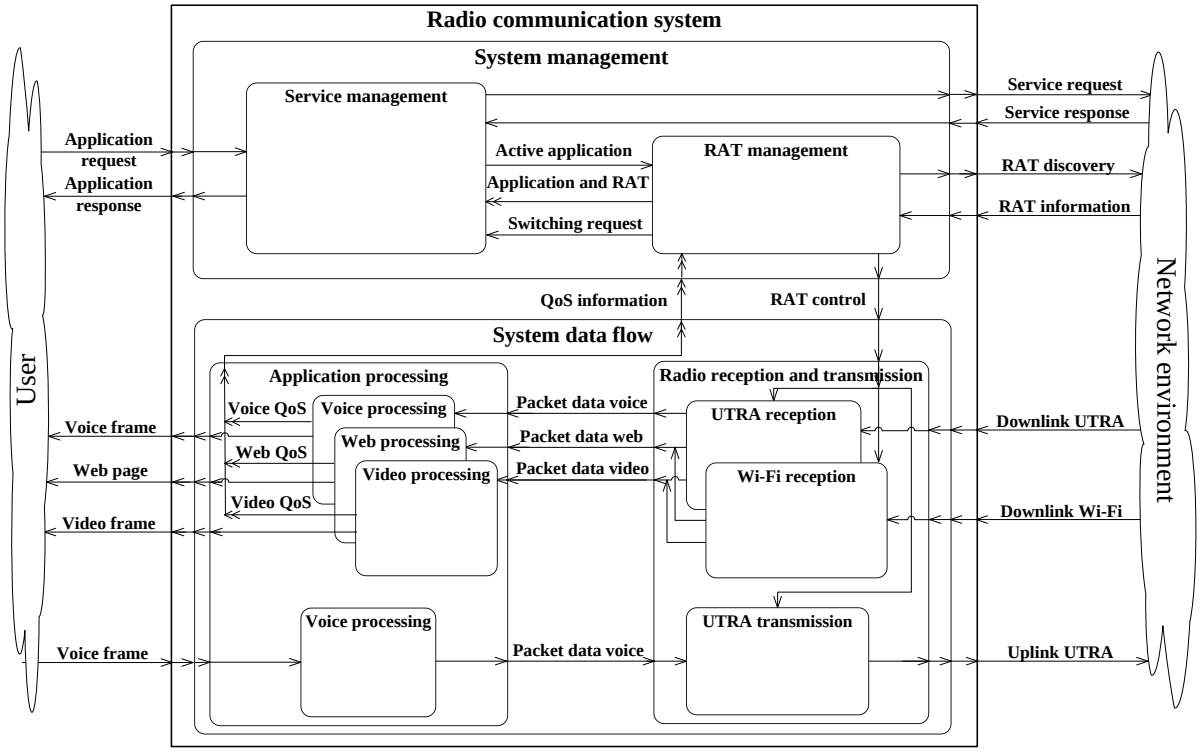

Figure 4.21 – Diagramme d'activités d'un terminal mobile adaptatif multiservice.

L'entité « User » comprend les activités permettant de demander, produire et consommer les données associées aux applications supportées par le système. Le démarrage et l'arrêt d'une application sur le système se fait avec les liens *Application request* et *Application response*. Le système envoie en retour les informations relatives aux trois applications en produisant des transactions *Voice frame*, *Web page* et *Video frame*. Pour chacune de ces applications, le système doit être capable d'offrir à l'utilisateur un niveau de qualité de service satisfaisant. Dans le cas de la conversation téléphonique, une transaction *Voice frame* contenant cent-soixante échantillons de signal voix doit être envoyée à l'utilisateur toutes les 20 ms. Une transaction représente 20 ms d'un signal de parole échantillonné à 8 kHz et quantifié sur 13 bits qu'un décodeur AMR (*Adaptive Multi Rate*) [**105**] est censé produire pour offrir un niveau de qualité de service satisfaisant pour cette application. Pour la navigation Internet, nous considérons qu'une page Web doit pouvoir être affichée dans un intervalle de temps d'une seconde suite à la demande de l'utilisateur.

Une transaction *Web page* est produite à l'instant où la page Web peut être affichée. En ce qui concerne le service de vidéo en continu, une transaction *Video frame* représente l'affichage d'une image de la vidéo. Nous considérons ici que le système doit être en mesure d'afficher vingt images par seconde pour que l'application soit délivrée à l'utilisateur avec un niveau de qualité de service satisfaisant.

L'environnement réseau inclut lui les activités relatives à la production des données associées aux trois applications supportées par le système et à leur transmission sur l'interface radio selon les standards UTRA [**106**] et WiFi [**107**]. Les échanges d'informations entre le système et les réseaux d'accès radio sont décrits avec les relations *Downlink UTRA, Downlink WiFi* et *Uplink UTRA.* Les quantités de données transmises pour chacune des applications via ces relations dépendent des conditions de transmission proposées sur les liens radio. Le lien radio descendant UTRA est supposé pouvoir atteindre des débits de 384 kbit/s et 144 kbit/s dans des environnements radio urbain et rurale. Le Wi-Fi permet lui de supporter des débits pouvant aller jusqu'à 54 Mbit/s.

Le système est représenté en distinguant deux sous-ensembles. La partie basse de la figure 4.21 montre le sous-ensemble « System data flow » qui regroupe les activités liées aux interfaces de communication et aux applications. Les activités « UTRA reception », « WiFi reception » et « UTRA transmission » assurent l'envoi et la réception des données sur les liens radio selon les standards de communication UTRA et Wi-Fi. Les activités « Voice processing », « Web processing » et « Video processing » décrivent elle l'exécution des traitements associés aux trois applications supportées. Le sous-ensemble « System management » présenté sur la partie haute de la figure 4.21 regroupe lui les activités assurant la gestion des demandes d'applications faites par l'utilisateur et le contrôle des interfaces de communication. L'activité « RAT management » permet d'analyser la qualité de service proposée à l'utilisateur à partir des informations transmises via la relation permanente *QoS information.* Dans le cas où le niveau de qualité de service d'une application n'est pas respectée, l'activité « RAT management » peut mettre en œuvre un processus de découverte des réseaux d'accès radio disponibles en utilisant la relation *RAT discovery*. Les informations transmises par l'environnement réseau via la relation *RAT information* sont utilisées pour analyser si une autre RAT peut offrir un niveau de QoS satisfaisant. Si c'est le cas, le changement de RAT s'effectue au sein du système par le biais de la relation *RAT control*. Cette relation permet d'activer ou de désactiver les deux interfaces de communication considérées. L'activité « Service management » prend en charge les communications avec l'environnement radio permettant d'acheminer les données d'une application via le réseau d'accès radio choisi.

Nous proposons donc dans le cadre de cette expérimentation de définir une description exécutable de ce système en utilisant ce niveau de représentation. La partie suivante présente donc les différents aspects originaux qu'il convient de pouvoir modéliser et les propositions qui ont été faites dans ce sens.

4.2.3 Techniques de modélisation proposées

- **Modélisation de l'environnement du système**

Le comportement du système étudié est fortement lié au comportement des entités de l'environnement avec lesquelles il interagit. Afin d'évaluer le caractère adaptatif et les performances pouvant être obtenues par le système, il est nécessaire de disposer d'une description de l'environnement qui permette de tester facilement les différents scénarios de fonctionnement envisageables. Pour appréhender ce problème, une technique proche de celle utilisée par Nokia dans [**65**] est proposée. Cette technique consiste à paramétrer le

modèle des entités de l'environnement avec un fichier scénario. Ce fichier contient les informations nécessaires pour faire évoluer le comportement de chaque entité de l'environnement au cours de la simulation.

La figure 4.22 illustre la technique de modélisation proposée pour tester différents cas d'utilisation du système par l'utilisateur.

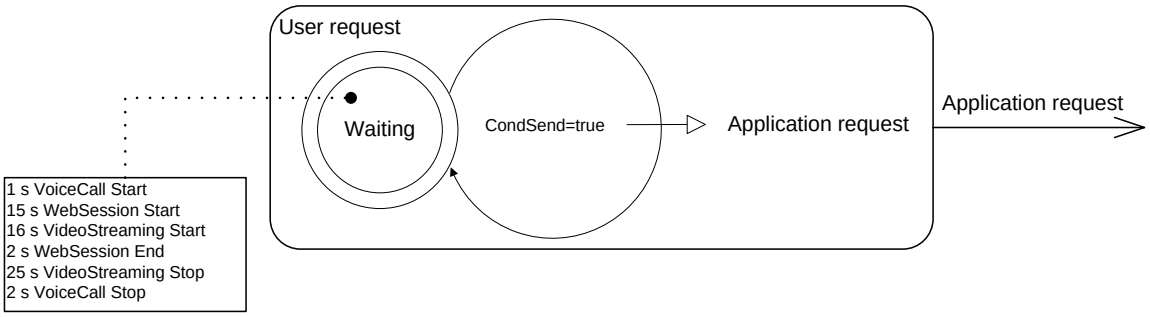

Figure 4.22 – Modélisation du comportement de l'utilisateur.

L'activité « User Request » permet de gérer l'envoi par l'utilisateur des différentes requêtes d'applications au système. Son comportement exhibe un seul état appelé *Waiting*. Sur la partie gauche de la figure 4.22, un exemple du fichier scénario proposé est présenté pour décrire les requêtes d'applications devant être envoyées pour un scénario de fonctionnement donné. Chaque ligne du fichier scénario est composée de quatre champs :

- les deux premiers champs permettent de définir le temps devant s'écouler avant l'envoi de la requête au système via la relation *Application request*,
- le troisième champ indique l'application concernée par la requête. Le paramétrage de ce champ se fait en utilisant les identificateurs VoiceCall, WebSession et VideoStreaming,
- le dernier champ indique si l'application associée à la requête doit être démarrée ou arrêtée par le système.

Le fichier scénario présenté sur la figure 4.22 permet d'évaluer le cas d'utilisation suivant. Au bout d'une seconde, l'utilisateur démarre une conversation téléphonique. Quinze secondes plus tard, il commence à naviguer sur Internet. Après seize secondes de navigation sur Internet, il démarre la lecture d'une vidéo en continu. Les trois autres lignes précisent les instants de fin d'utilisation de chacune de ces applications.

Au cours de l'état *Waiting*, les différentes requêtes d'applications décrites dans le fichier sont lues successivement. L'envoi de la requête se fait dès lors que la durée indiquée est écoulée. Une transaction *Application request* est alors produite avec les informations relatives aux deux derniers champs de la ligne.

La même technique de modélisation est appliquée pour décrire l'évolution du comportement de l'environnement radio. La figure 4.23 montre le modèle proposé pour représenter les fluctuations des conditions de transmission proposées par chacun des réseaux d'accès radio considérés suite aux déplacements de l'utilisateur du système.

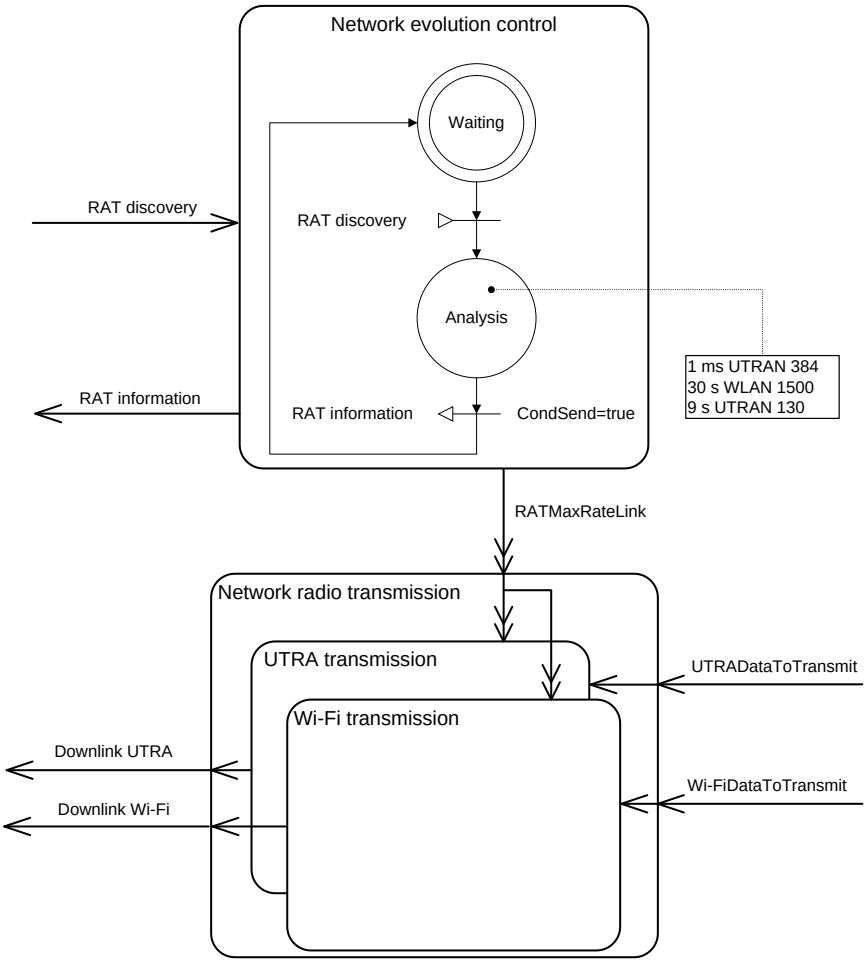

Figure 4.23 – Modélisation du comportement de l'environnement réseau.

L'activité « NetworkEvolutionControl » a été définie de manière à faire évoluer facilement les débits de transmission pouvant être proposés par les deux réseaux d'accès radio UTRAN (*UMTS Terrestrial Radio Access Network*) et WLAN considérés dans notre cas d'étude.

Sur la partie droite de la figure 4.23 est présenté le fichier scénario proposé pour traduire l'évolution de l'environnement radio du système. Chaque ligne du fichier scénario est composée de quatre champs :

- les deux premiers champs décrivent la durée qui doit s'écouler avant de tenir compte de l'évolution des conditions de transmission d'un réseau d'accès radio,
- le troisième champ indique le réseau d'accès radio concerné par cette évolution,
- le dernier champ indique le nouveau débit à considérer.

Dans le fichier scénario présenté, le réseau d'accès radio UTRAN propose dans un premier temps un débit de 384 kbit/s. Ensuite, au bout de trente secondes, un réseau local apparaît dans l'environnement radio du système avec un débit d'1,5 Mbit/s. Puis après neuf secondes, les conditions de transmission se dégradent sur l'interface radio de l'UTRAN. Le débit proposé passe en effet de 384 kbit/s à 130 kbit/s.

Au sein de l'activité « Network evolution control », c'est dans l'état *Analysis* que s'effectue la lecture du contenu du fichier pour appliquer les évolutions des conditions de transmission aux instants fixés au cours de la simulation. La relation permanente *RATMaxRateLink* permet d'indiquer aux activités « UTRA transmission » et « Wi-Fi transmission » le débit pouvant être utilisé pour acheminer les données. Pour effectuer ses choix de RAT, le système peut accéder aux informations concernant les performances pouvant être atteintes sur les liens radio en envoyant une transaction *RAT discovery* à l'activité « NetworkEvolutionControl ». L'activité lui envoie en retour une transaction *RAT information* avec les informations demandées. Ces relations se veulent représentatives du canal spécifique Cognitive Pilot Channel ou CPC introduit dans la partie 1.1.3.

- **Modélisation des interfaces radio du système**

Nous avons constaté précédemment l'intérêt qu'il pouvait y avoir à réduire le nombre de transactions nécessaires à l'évolution du comportement du système modélisé. Dans le cas d'étude présent, nous avons défini un niveau de granularité correspondant aux échanges de données au niveau de la couche MAC entre le réseau d'accès radio et le terminal mobile. La figure 4.24 présente les transactions et le niveau de granularité associé que nous avons considéré pour étudier le comportement des fonctions assurant la réception et la transmission des données sur les liens radio.

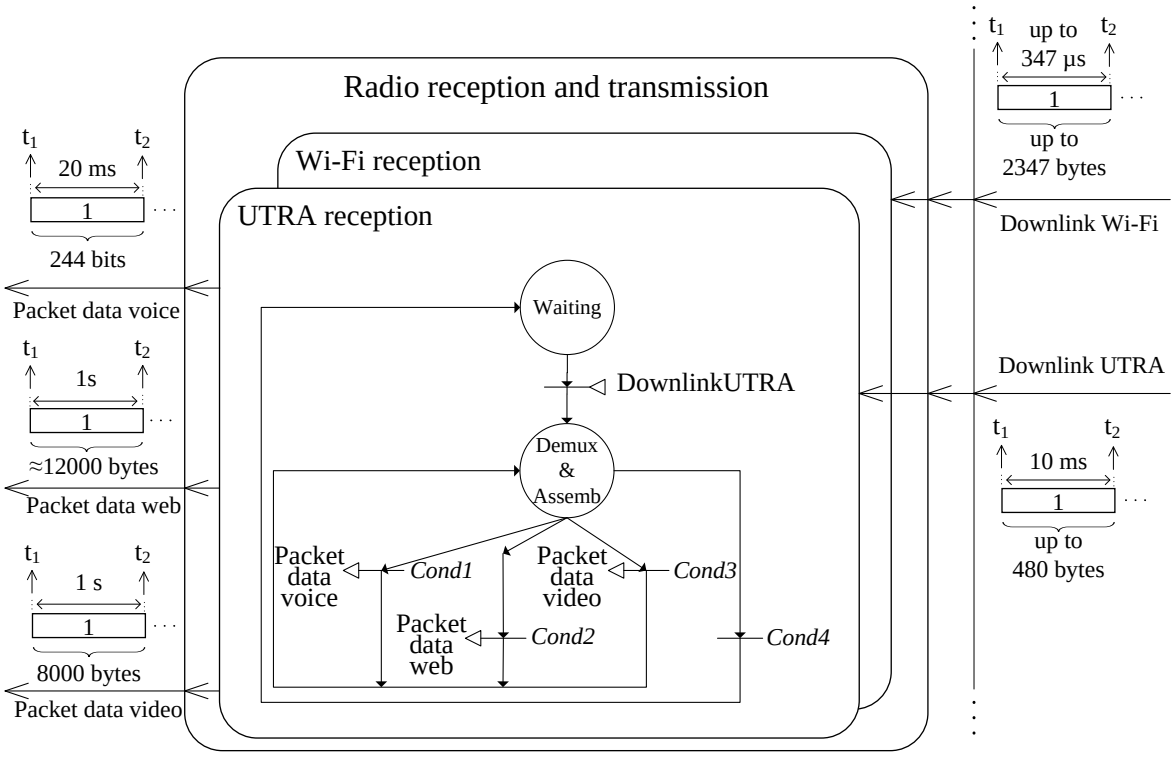

Figure 4.24 – Modélisation des interfaces de communication du système.

Au niveau de modélisation considéré, une transaction *Downlink UTRA* est initiée par le réseau d'accès radio UTRA toutes les 10 ms. Cet intervalle de temps correspond à la période d'envoi d'une trame radio selon les spécifications de ce réseau. La quantité d'informations transmise durant cet intervalle de temps dépend du débit proposé. En considérant un débit maximal de 384 kbit/s, une transaction peut contenir au maximum 480 octets. Pour le réseau WLAN, la taille maximale d'une trame radio est de 2347 octets. Les instants d'envoi ne sont pas périodiques. Si l'on se place dans le cas du débit maximal proposé avec ce standard de communication de 54 Mbit/s, une trame radio Wi-Fi peut être envoyée toutes les 347 µs.

Le comportement des activités « UTRA reception » et « Wi-Fi reception » est décrit de manière à représenter les traitements relatifs aux couches protocolaires MAC et RLC (*Radio Link Control*) de l'UTRA et LLC (*Logical Link Control*) du Wi-Fi. Ces couches protocolaires assurent en réception le démultiplexage des différents flux de données contenues dans une trame. Chaque flux est associé à une application employée par l'utilisateur. Elles réalisent aussi la reconstitution des blocs de données associés à chaque application qui ont été préalablement segmentés pour être transmis sur l'interface radio.

La partie gauche de la figure 4.24 représente les quantités de données devant être reçues dans les intervalles de temps précisés pour pouvoir initier les traitements associés à chaque application. Une transaction *Packet data voice* est ainsi envoyée dès lors qu'un bloc de données de 244 bits est reconstitué. Cette taille de bloc de données correspond à la quantité d'informations encodée par un codeur AMR à un débit de 12,2 kbit/s toutes les 20 ms puis transmise sur l'interface radio. Une transaction *Packet data web* est envoyée lorsque la totalité des informations nécessaires pour l'affichage de la page Web ont été reçues. Il est indiqué dans [**108**] que la taille moyenne d'un fichier HTML (*HyperText Markup Language*) pour la description d'une page Web est de 12000 octets. Une transaction *Packet data video* est initiée lorsqu'un paquet de données correspondant à une seconde de vidéo est reçu. Vingt images par seconde sont affichées dans notre cas d'étude et une image est décrite par un paquet de données constitué de 400 octets. Une seconde de vidéo correspond donc à un paquet de données de 8000 octets.

En utilisant ce niveau de granularité pour les communications, il est possible de décrire le comportement des activités « UTRA reception » et « Wi-Fi reception » de façon identique. La description de ce comportement se fait avec un automate à deux états. Le premier état *Waiting* correspond à l'état d'attente d'une transaction *Downlink UTRA* et *Downlink Wi-Fi*. Le deuxième état *Demux & Assemb* permet lui d'extraire les données contenues dans la trame radio reçue. Lorsqu'un paquet de données associé à une application est reconstitué, l'activité produit en sortie la transaction associée. Ce type de représentation permet de faciliter la capture du modèle qui pourrait supporter une gamme plus étendue de standards de communication. La représentation des activités se rapporte au modèle d'exécution générique proposé dans la partie 3.1.5 avec une entrée et trois sorties. Nous évoquerons dans la partie suivante le calcul des propriétés non fonctionnelles associées à ces activités.

- **Modélisation de la gestion autonome des interfaces de communication du système**

Le système considéré offre la particularité de proposer des services lui permettant de basculer de façon autonome entre les deux réseaux d'accès radio UTRAN et WLAN. L'objectif est de garantir à l'utilisateur que les applications en cours de fonctionnement lui sont délivrées avec la meilleure qualité de service possible. La figure 4.25 présente la description du comportement de l'activité « RAT management » proposée pour représenter ces nouveaux services afin d'analyser leur impact sur le comportement global du système.

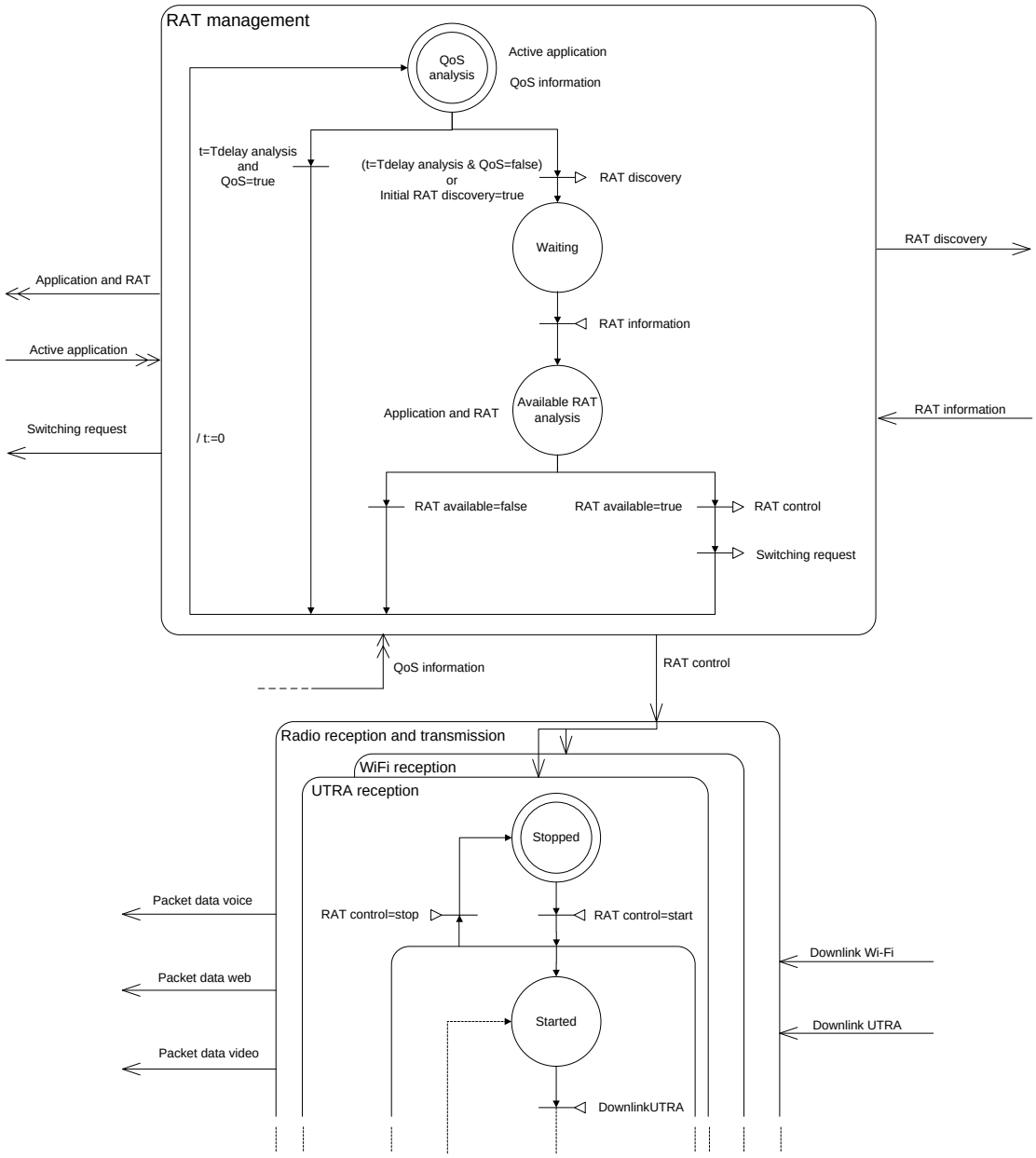

Figure 4.25 – Modélisation de la partie contrôle des interfaces de communication du système.

Le comportement de cette activité est décrit à l'aide d'un automate à trois états. Au démarrage, le système doit décider du réseau d'accès radio auquel il va se connecter en fonction des critères préférentiels définis par l'utilisateur et des performances associées aux réseaux d'accès radio disponibles dans son environnement. Ce comportement a été décrit de la manière suivante.

Une transaction *RAT discovery* envoyée correspond à une requête d'informations faite par le système à l'environnement réseau concernant les RAT présentes dans son environnement avec les performances associées. Une transaction *RAT information* reçue en retour par le système contient les informations demandées. Dans l'état *Available RAT analysis*, le système analyse ces informations. Le démarrage de l'interface de communication choisie se fait ensuite en initiant une transaction *RAT control*.

Le système doit ensuite être capable d'opérer en suivant le cycle cognitif en trois phases décrit dans la partie 1.1.3.

Il doit tout d'abord observer la qualité de service proposée pour les applications en cours de fonctionnement. Pour représenter ce comportement, l'activité « RAT management » analyse périodiquement la QoS proposée pour acheminer les données associées aux applications en cours d'utilisation sur le système en s'appuyant sur les différents critères présentés dans la partie 1.1.1. Cette action se fait dans l'état *QoS analysis* via une opération de lecture du contenu de la relation permanente *QoS information* mise à jour par les activités du sous-ensemble « Application processing ». La période d'analyse du contenu de cette relation permanente *Tdelai analysis* est ici de 50 ms.

Dans le cas où une application n'est pas assurée au niveau de QoS exigée par l'utilisateur, le système doit alors être capable d'identifier un nouveau standard de communication permettant de résoudre ce problème. Pour représenter ce comportement, l'activité « RAT management » initie une transaction *RAT discovery* et reçoit en retour une transaction *RAT information* contenant la nouvelle liste des RATs disponibles dans l'environnement du système.

Si le système identifie dans l'état *Available RAT analysis* une nouvelle RAT permettant d'assurer la QoS requise par l'application considérée comme dégradée, une transaction *Switching request* est alors envoyée à l'activité « Service management ». Elle contient le nom de l'application dégradée et la nouvelle RAT à utiliser. Dans le cas où aucune RAT ne permet d'améliorer la QoS, un nouveau processus de découverte est initié toutes les 50 ms jusqu'à ce qu'une nouvelle RAT soit découverte ou que l'application soit de nouveau proposée avec un niveau de QoS acceptable par l'utilisateur.

En s'appuyant sur ce type de représentation, il est alors possible d'évaluer et de valider les performances pouvant être atteintes avec un tel système.

4.2.4 Résultats issus de la simulation du modèle du système

La description du système et de son environnement a été capturée avec l'outil CoFluent Studio. L'outil permet de générer le code SystemC associé à ce modèle afin de le rendre exécutable. Le modèle généré représente 4421 lignes de code SystemC dont 62% sont insérées automatiquement par l'outil.

La simulation du modèle offre la possibilité d'analyser les réactions du système vis-à-vis des différents cas d'utilisation envisageables. Cette évaluation permet d'une part de vérifier si le système est en mesure de respecter les contraintes temporelles qui lui sont appliquées. Les résultats de simulation obtenus permettent aussi d'évaluer la capacité du système à maintenir un niveau de qualité de service satisfaisant pour l'utilisateur. Le temps de simulation nécessaire à l'exécution du modèle est d'une seconde et demie sur un poste de travail doté d'un processeur Intel Core 2 duo ayant une fréquence d'horloge de 2,66 GHz. L'instrumentation proposée par l'outil CoFluent Studio pour obtenir l'observation présentée sur la figure 4.26 implique un surcoût de temps de simulation de dix-huit secondes.

La figure 4.26 montre un exemple d'affichage de résultats proposé par l'outil CoFluent Studio permettant d'observer l'évolution du comportement du système lors d'une simulation.

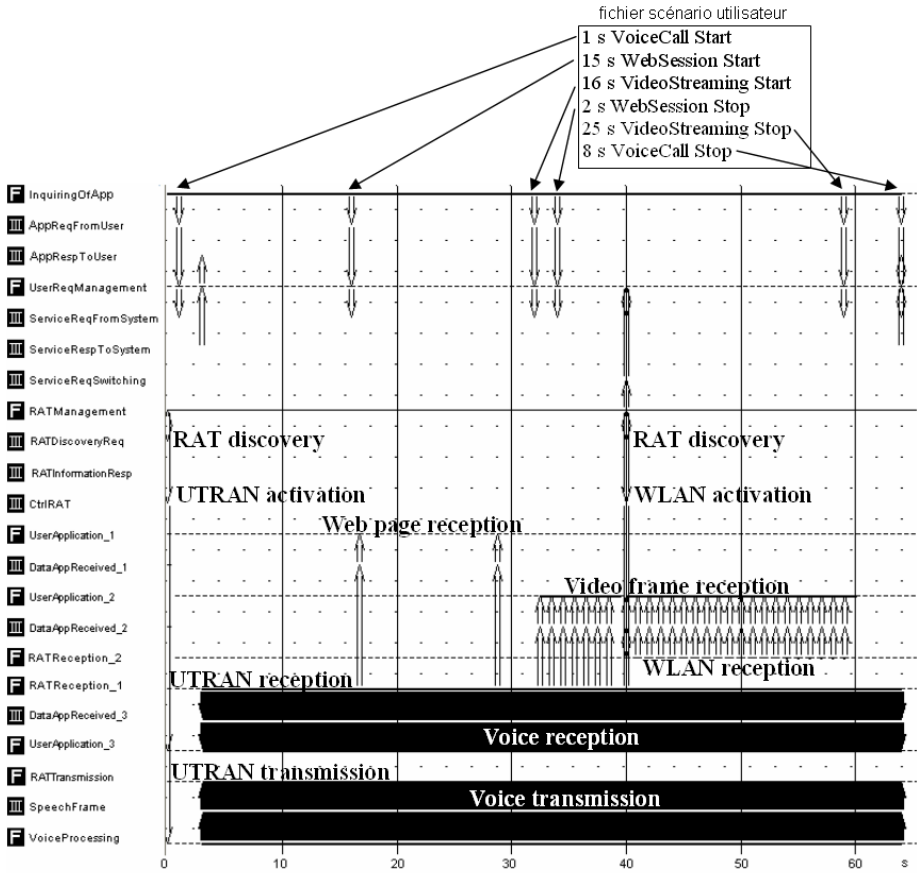

Figure 4.26 – Observation des transactions initiées lors de l'exécution du modèle du système adaptatif multiservice.

Les fonctions modélisées du système et de son environnement ainsi que les relations mises en jeu sont décrites sur la partie gauche de la figure 4.26. Les flèches représentent les transactions initiées lors de la simulation du modèle. Avec ce type d'affichage, les différentes interactions entre les fonctions du modèle peuvent être observées durant la simulation. Il est possible par exemple de visualiser l'envoi par la fonction *InquiringOfApp* des différentes requêtes d'applications conformément aux indications fournies dans le fichier scénario utilisateur présenté sur la partie haute de la figure. Ces trois applications sont dans un premier temps proposées par le système à l'utilisateur via le réseau d'accès radio UTRAN. Sur cette figure, cela se traduit par des transactions entre les fonctions *RATReception_1*, *RAT transmission* et les fonctions de traitement associées aux applications que sont *UserApplication_1*, *UserApplication_2* et *UserApplication_3*. Après trente-neuf secondes, le système initie un processus de découverte. Il constate en effet que l'UTRAN ne permet plus de satisfaire la QoS associée à la conversation téléphonique et à la lecture de la vidéo en continu en cours de fonctionnement. Après la phase de découverte et d'analyse, le système décide alors de basculer l'application de vidéo en continu sur un réseau d'accès radio local qui est présent dans son environnement. Une transaction *RATControl* est en effet envoyée par la fonction *RATManagement* à la fonction *RATReception_2* relative à l'interface de réception Wi-Fi pour l'activer. La fonction UserApplication_2 assurant les traitements de l'application de vidéo en continu reçoit ensuite des transactions provenant de la fonction *RATReception_2*.

La figure 4.27 présente elle un autre moyen de visualisation proposé par l'outil CoFluent Studio que nous utilisons pour observer l'évolution du niveau de qualité de service proposé

à l'utilisateur au cours de la simulation. Le niveau de qualité de service est ici analysé à partir du critère de latence.

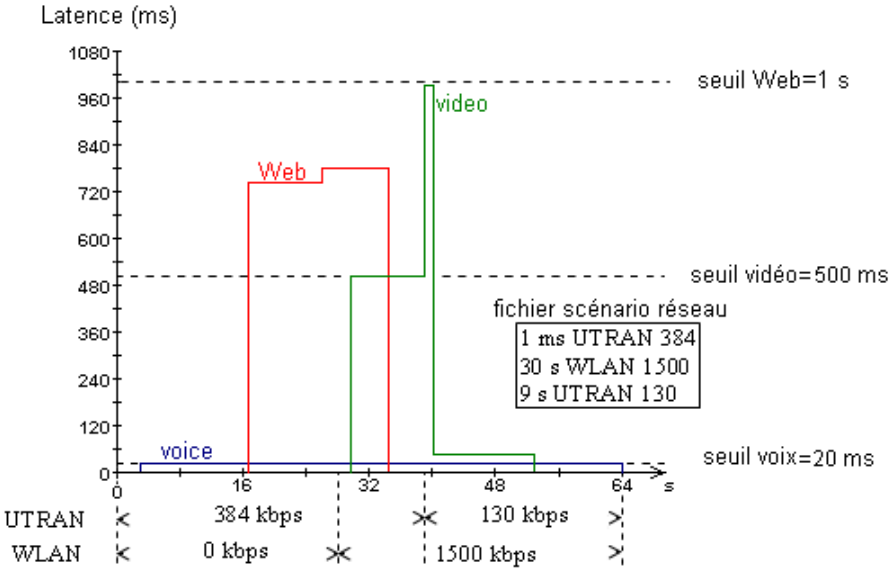

Figure 4.27 – Observation de l'évolution de la qualité de service à partir du critère de latence.

Sur la partie droite de la figure 4.27 se trouvent les niveaux de QoS définis pour chacune des applications proposées par le système. L'évolution de l'environnement réseau durant la simulation est présentée à travers le fichier scénario réseau. Il est possible de constater sur cette figure que l'application de vidéo en continu n'est plus proposée avec un niveau de QoS satisfaisant à l'utilisateur suite à la dégradation des performances de la liaison radio proposée par l'UTRAN après trente neuf secondes de simulation. Le retour à un niveau de QoS satisfaisant se fait suite au basculement de cette application sur un réseau d'accès radio local. La latence associée à cette application passe en effet de 990 ms à une valeur de 42 ms bien inférieure au seuil de 500 ms défini.

Le premier modèle présenté précédemment a été défini de manière à décrire l'ensemble des propriétés fonctionnelles associées au système considéré. En exécutant le modèle obtenu, il est possible d'observer l'évolution du comportement du système compte tenu des propriétés temporelles allouées sur les différents constituants du modèle et des scénarios de fonctionnement testés. En analysant les résultats obtenus, le concepteur est en mesure de valider le comportement souhaité pour le système et de caractériser les différentes contraintes temporelles à respecter.

4.2.5 Prise en compte des propriétés non fonctionnelles

L'expérimentation présentée dans cette partie permet d'illustrer la mise en œuvre de notre approche pour la prise en compte des propriétés non fonctionnelles du système étudié. Nous proposons ici d'effectuer le travail de dimensionnement des ressources nécessaires à la mise en œuvre des fonctions de décodage canal des deux standards de communication considérés en s'appuyant sur le modèle de niveau transactionnel présenté précédemment. Ce modèle a permis dans un premier temps d'observer les performances obtenues par le système en ne considérant que les propriétés fonctionnelles associées. Il s'agit donc cette fois-ci de pouvoir observer les performances obtenues par le système en tenant compte des caractéristiques des ressources utilisées pour exécuter les différentes fonctions demandées.

Dans le cadre de ce cas d'étude, il est considéré que le décodage canal est réalisé avec un décodeur turbo pour l'UTRA [**109**] et avec un décodeur de Viterbi pour le Wi-Fi [**107**].

L'analyse porte ici sur la puissance de calcul devant être mise en œuvre par les ressources utilisées pour réaliser ces deux traitements. Nous considérons pour cette expérimentation que le réseau d'accès radio UTRA encode les données transmises au terminal avec un taux d'encodage de 1/3. Afin d'analyser la puissance de calcul nécessaire pour réaliser le décodage des données contenues dans les trames radio UTRA, nous réutilisons l'expression analytique proposée dans le cas d'étude portant sur le standard LTE puisque le même décodeur est employé [**110**] :

$$CC_{TurboD} = \frac{Nb_{BitsCodeBlock} * Nb_{Loop} * 132}{T_{ProcTurbo}}.$$

Du point de vue du réseau d'accès radio Wi-Fi, nous considérons un taux d'encodage de 1/2. Le codeur convolutionnel est ici considéré avec une longueur de contrainte de 7. L'analyse de l'algorithme de Viterbi [**111**] pouvant être utilisé pour réaliser la fonction de décodage canal au niveau du récepteur Wi-Fi a permis de déterminer l'expression analytique suivante :

$$CC_{ConvolD} = \frac{Nb_{BitsFrame} * 128}{T_{ProcConvol}}.$$

Cette expression permet de calculer la puissance de calcul nécessaire, en nombre d'opérations par seconde, pour réaliser le décodage des $Nb_{BitsFrame}$ bits de données contenus dans une trame radio Wi-Fi.

Deux types d'architectures sont considérés dans le cadre de cette étude. La première repose sur deux ressources matérielles spécifiques réalisant chacune les traitements associés aux deux fonctions de décodage canal pour les deux standards considérés. La mise en œuvre est illustrée sur la figure 4.28.

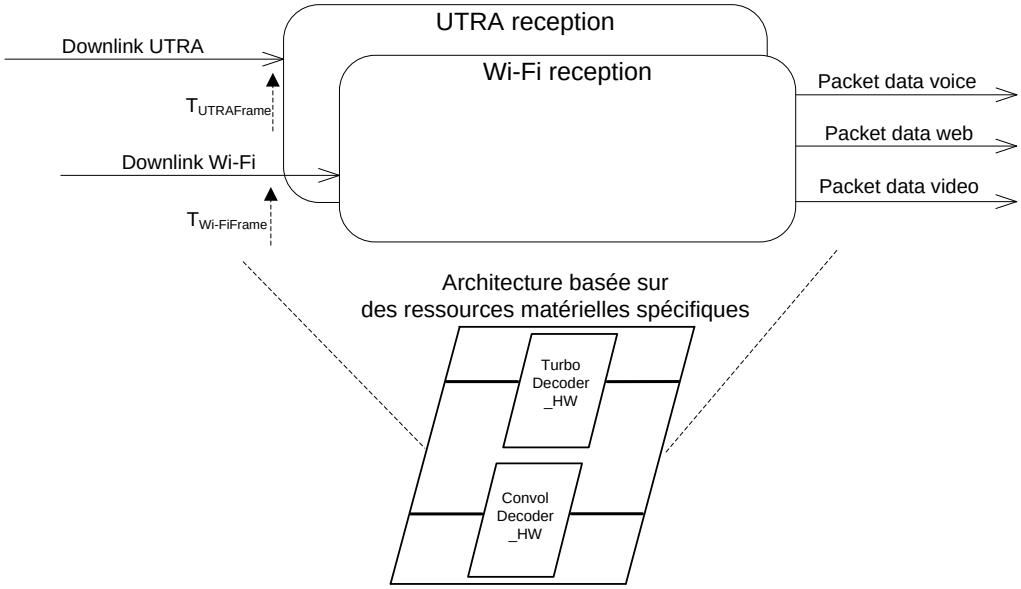

Figure 4.28 – Architecture évaluée basée sur l'utilisation de deux ressources matérielles spécifiques.

Le modèle de base a été enrichi selon la technique présentée dans le chapitre 3 de manière à pouvoir observer les propriétés non fonctionnelles associées aux deux activités relatives aux deux RAT considérées. Dans le cas présenté, seule l'étude des fonctions de décodage canal a été faite. Le paramétrage de $T_{ProcTurbo}$ et $T_{ProcConvol}$ s'effectue de manière à pouvoir réaliser le décodage des données reçues pendant les intervalles de temps $T_{UTRAFrame}$ et $T_{Wi\text{-}FiFrame}$ qui séparent deux trames consécutives UTRA et Wi-Fi.

Les figures 4.29 et 4.30 montrent l'évolution de la puissance de calcul observée pour réaliser le décodage canal sur les interfaces de réception UTRA et Wi-Fi. Pour cette évaluation, le scénario de fonctionnement présenté dans la partie 4.2.4 est repris en appliquant le jeu de paramètres suivant :

$$T_{ProcTurbo} = 2500\ \mu s, T_{ProcConvol} = 4000\ \mu s.$$

Ce paramétrage permet de réaliser le décodage canal des blocs de données reçus dans le cadre du scénario de fonctionnement considéré.

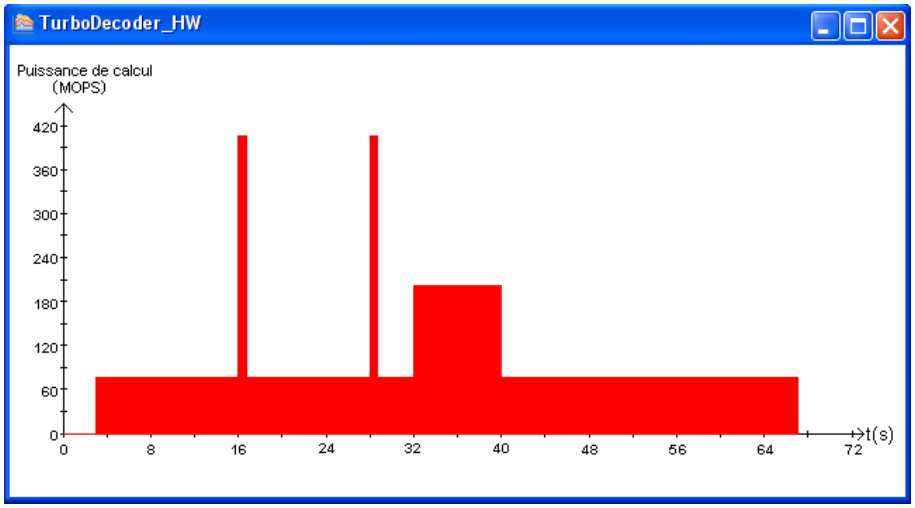

Figure 4.29 – Évolution de la puissance de calcul requise pour le décodage canal du récepteur UTRA.

La puissance de calcul observée pour réaliser le décodage canal des données par le récepteur UTRA fluctue en fonction de la taille des blocs de données à décoder. Les blocs de données contenant les informations associées au service de communication vocale nécessitent par exemple une puissance de calcul de 77 MOPS avec le paramétrage évalué. Les deux pages Web reçues dans le cadre du scénario de fonctionnement considéré requièrent elles une puissance de calcul de 405 MOPS. Enfin l'application de vidéo en continu initialement proposée via le réseau d'accès radio UTRAN demande une puissance de calcul de 202 MOPS. Cette application est ensuite basculée sur le réseau local. La figure 4.30 montre que la réalisation du décodage canal, avec le décodeur convolutionnel et selon le paramétrage considéré, nécessite une puissance de calcul de 1183 MOPS.

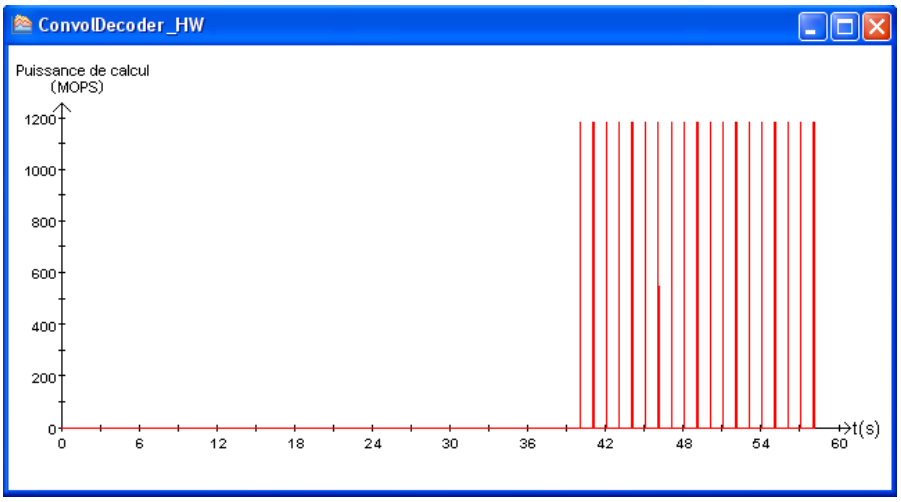

Figure 4.30 – Évolution de la puissance de calcul requise pour le décodage canal du récepteur Wi-Fi.

La prise en compte des propriétés non fonctionnelles au sein de ce modèle et l'instrumentation proposée au sein de l'outil pour permettre leur observation n'entraîne pas ici de surcoût sur le temps de simulation par rapport au modèle qui ne considère pas les propriétés non fonctionnelles.

La deuxième architecture considérée est composée cette fois-ci d'une ressource matérielle reconfigurable supportant les deux fonctions de décodage canal comme celle présentée dans [**112**]. La mise en œuvre est illustrée sur la figure 4.31.

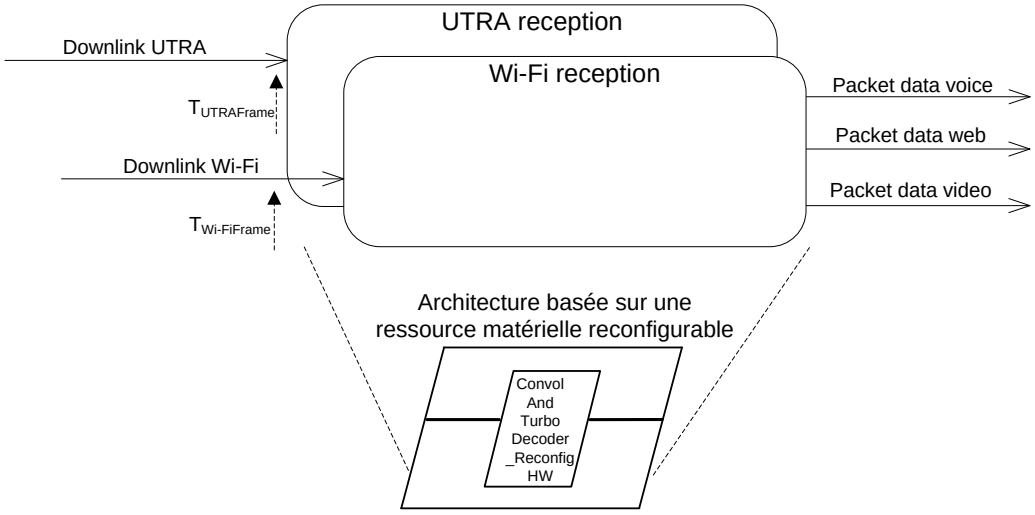

Figure 4.31 – Architecture évaluée basée sur l'utilisation d'une ressource matérielle reconfigurable.

Le partage des ressources de calcul entre ces deux standards doit permettre de réduire le coût en termes de surface occupée sur le circuit final par rapport à la solution précédente pour réaliser ces fonctions. En contrepartie, les deux traitements supportés ne peuvent s'exécuter pendant le même intervalle de temps. De plus, le passage d'un traitement à un autre implique de réaliser un processus de reconfiguration des ressources de calcul. Il est donc nécessaire d'évaluer si les caractéristiques de l'architecture considérée permettent de satisfaire aux exigences de l'application.

La figure 4.32 montre la puissance de calcul observée pour exécuter les traitements de décodage canal des deux interfaces de réception UTRA et Wi-Fi en considérant à nouveau le scénario de fonctionnement illustré dans la partie 4.2.4.

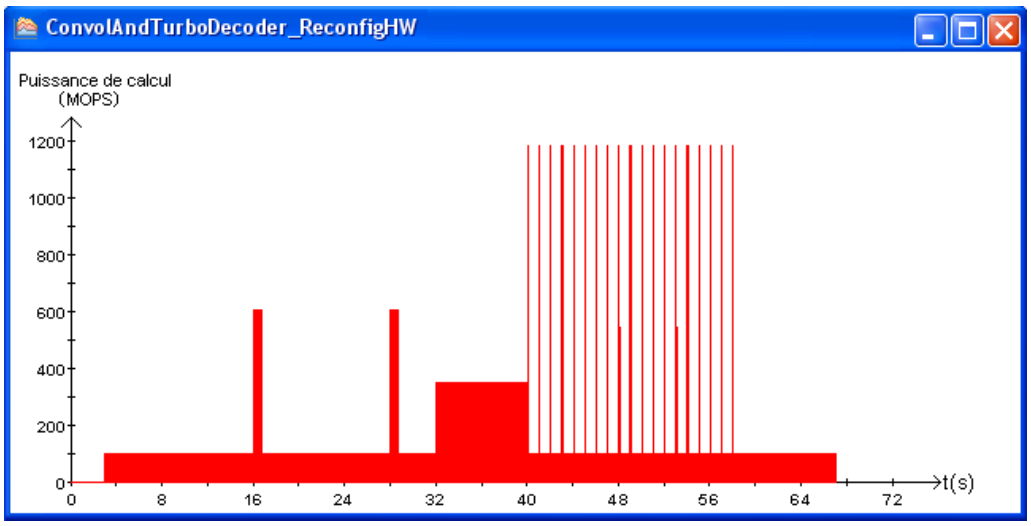

Figure 4.32 – Évolution de la puissance de calcul requise pour le décodage canal du récepteur Wi-Fi et UTRA.

Nous avons conservé pour cette simulation le temps de traitement $T_{ProcConvol}$ utilisé lors de l'expérimentation précédente. Il est alors nécessaire de diminuer le temps de traitement alloué pour les opérations de turbo décodage pour le récepteur UTRA afin de respecter les contraintes temporelles induites par le scénario de fonctionnement évaluée. $T_{ProcTurbo}$ ne doit alors pas dépasser 2000 µs. Cette figure présente l'évolution de la puissance de calcul observée avec cette nouvelle solution d'implantation et pour le paramétrage considéré. En observant les délais qui séparent deux décodages successifs effectués avec deux décodeurs différents, il est possible d'obtenir une approximation de la durée maximale que devrait prendre le processus de reconfiguration des ressources de calcul.

4.2.6 Bilan

Ce deuxième cas d'étude a permis d'illustrer l'intérêt qu'il pouvait y avoir à disposer d'un modèle exécutable défini à un niveau d'abstraction élevé afin d'évaluer et de valider très tôt dans le processus de conception le comportement et les performances d'un système de radiocommunication mobile. Des techniques de modélisation ont été présentées afin de faciliter la description des spécificités de ces systèmes au sein de modèles de niveau transactionnel. Des techniques de modélisation ont aussi été proposées pour décrire l'environnement du système. La simulation du modèle obtenu permet alors d'observer le comportement et les performances du système étudié en fonction des différents scénarios de fonctionnement possibles. Il en ressort aussi de cette expérimentation la possibilité de pouvoir évaluer les ressources nécessaires à la mise en œuvre d'un tel système. Les techniques proposées dans le chapitre 3 ont en effet été utilisées de façon à permettre la prise en compte des propriétés non fonctionnelles. Ces techniques ont été utilisées afin d'évaluer les ressources de calcul nécessaires à la mise en œuvre des RAT.

Conclusion

Les prochaines générations de terminaux mobiles devront être capables, de façon autonome, de déterminer au fil des déplacements de l'utilisateur, les standards de communications les plus adaptés pour garantir un niveau de qualité de service satisfaisant pour les applications en cours de fonctionnement. Ces systèmes proposeront en plus une gamme étendue de standards de communication et une offre diversifiée d'applications. Pour intégrer toutes ces fonctionnalités en respectant les contraintes fortes en termes de coût, de consommation et de surface associées à ces systèmes, de nouvelles architectures composées d'un ensemble de ressources hétérogènes et flexibles devront être utilisées. Pour identifier et caractériser les ressources nécessaires à la mise en œuvre de tels systèmes, les concepteurs devront explorer un espace de conception élargi. Ce travail étant actuellement mené selon une approche pragmatique et incrémentale basée sur l'expérience préalablement acquise par les concepteurs, il semble essentiel de s'appuyer sur des approches efficaces dans le cadre de la conception des prochaines générations de terminaux mobiles. Dans ce contexte, ce travail de thèse visait à améliorer la productivité des concepteurs en proposant une démarche originale pour la définition de modèles de niveau transactionnel en vue du dimensionnement des futurs terminaux mobiles.

Le chapitre 1 a permis d'introduire dans un premier temps le contexte applicatif dans lequel s'est inscrit ce travail. L'analyse de l'évolution des systèmes de radiocommunication et du processus de conception de ces systèmes a permis d'identifier les nouvelles problématiques à adresser dans le cadre du dimensionnement des prochaines générations de terminaux mobiles. L'augmentation du nombre de fonctionnalités à intégrer et l'utilisation de ressources hétérogènes vont entraîner dans le cadre de ce travail un élargissement de l'espace de conception à explorer. La maîtrise de la complexité amenée par ces évolutions implique la définition et l'adoption de nouvelles méthodes, s'appuyant sur l'utilisation de modèles de haut niveau, afin d'identifier et de caractériser rapidement et efficacement une solution architecturale.

Le chapitre 2 a permis de présenter les différents travaux menés actuellement pour favoriser la définition de modèles comme support d'évaluation et de comparaison de différentes solutions d'implantation. En s'appuyant sur le positionnement des différents niveaux d'abstraction pouvant être utilisés pour modéliser ces systèmes, nous avons pu comparer les différentes propositions de modèles faites actuellement. Deux tendances importantes ont alors pu être observées. L'une se rapporte à la volonté d'élever le niveau d'abstraction utilisé pour la définition de ces modèles. Le niveau transactionnel avec des informations temporelles a été identifié comme offrant des possibilités intéressantes dans le cadre de ce travail. L'autre tendance est liée à la volonté de diminuer les temps de simulation nécessaires à l'exécution des modèles utilisés. Lors de cette analyse, un autre constat a aussi pu être fait sur le manque de démarches clairement définies visant à guider le concepteur pour l'utilisation des modèles proposés.

Partant de ces constats, le chapitre 3 a introduit les différentes contributions apportées pour proposer une approche et des modèles correctement formalisés pour permettre une exploration efficace de l'espace de conception pour les futurs terminaux mobiles. Nous nous sommes appuyés sur une notation existante pour permettre la description des différentes propriétés d'un système à modéliser au niveau transactionnel dans le cadre de son dimensionnement. La première contribution a porté sur **la définition d'une approche** permettant de représenter les propriétés fonctionnelles et non fonctionnelles nécessaires à l'évaluation des performances d'un système au sein d'un même modèle, et ce à partir de la

notation proposée. La deuxième contribution a porté sur **la proposition d'une technique de calcul des propriétés non fonctionnelles** permettant de réduire de manière significative les temps de simulation nécessaires à l'exécution des modèles. La troisième contribution a consisté à définir **un modèle d'exécution générique** sur lequel pourrait s'appuyer un concepteur de systèmes pour définir plus facilement des instances de modèles permettant l'évaluation de différentes solutions architecturales envisageables. Une mise en œuvre possible des modèles proposés avec l'outil CoFluent Studio a au final été présentée. Une expérimentation portant sur le dimensionnement d'une architecture devant réaliser un algorithme de transformée de Fourier rapide sur huit points a ensuite été présentée afin d'illustrer la mise en œuvre de notre approche. Le but était d'une part de présenter les observations pouvant être faites suite à l'exécution des modèles obtenus en suivant notre approche. L'autre objectif était d'observer le gain de temps de simulation obtenu en comparant le modèle de niveau transactionnel proposé avec un autre modèle ne mettant pas en œuvre la technique utilisée pour le calcul des propriétés non fonctionnelles.

Le chapitre 4 a permis d'illustrer l'intérêt de notre approche en s'appuyant sur deux cas d'étude bien distincts et complémentaires. Le premier cas d'étude a illustré la mise en œuvre de notre approche dans le cadre du dimensionnement de la couche physique du futur standard de communication LTE. Nous avons présenté dans un premier temps les différentes fonctions de traitement devant être réalisées ainsi que les différentes contraintes à respecter. Un modèle a ensuite été défini en mettant en œuvre notre approche. Ce modèle a été utilisé pour évaluer les performances pouvant être obtenues avec trois implantations possibles. Les résultats de simulation offrent alors au concepteur de systèmes la possibilité de comparer ces différentes solutions d'implantation en se basant sur les observations faites sur les coûts associés en termes de mémoires et de ressources de calcul pour différents paramétrages d'architectures. Le deuxième cas d'étude a permis d'illustrer l'intérêt qu'il pouvait y avoir pour le concepteur à disposer très tôt dans le processus de conception d'un modèle exécutable de son système. **Des techniques de modélisation** ont été proposées pour faciliter la description au niveau transactionnel des différentes caractéristiques des futurs terminaux mobiles introduites dans le chapitre 1. Les résultats de simulation obtenus permettent d'analyser le comportement et les performances atteintes avec un tel système en fonction de différents scénarios de fonctionnement pouvant être envisagés. Nous avons ensuite montré comment il était possible d'observer les performances pouvant être obtenues en tenant compte des caractéristiques de l'architecture considérée pour mettre en œuvre les deux standards de communication. L'expérimentation a porté sur la fonction de décodage canal devant être réalisée par la couche physique de ces deux standards. Deux types d'implantation ont été évalués. Le premier cas consistait à réaliser chaque standard en utilisant deux ressources de calcul bien distinctes. Le deuxième cas reposait sur l'utilisation d'une ressource disposant de la flexibilité requise pour supporter ces deux fonctions.

Perspectives

Á l'issue des travaux menés dans le cadre de cette thèse, différentes perspectives peuvent être envisagées afin d'améliorer les possibilités offertes par l'approche proposée pour correctement dimensionner les prochaines générations de terminaux mobiles.

Un modèle d'exécution générique à une entrée et une sortie a été présenté pour faciliter ce travail. Il conviendrait de poursuivre la définition de ce modèle pour décrire des fonctions nécessitant plusieurs entrées et plusieurs sorties.

Par ailleurs, l'évaluation des performances obtenues avec une architecture se fait actuellement à partir des observations faites sur la puissance de calcul et la quantité de ressources de mémorisation à proposer. Il pourrait être intéressant d'élargir le nombre de propriétés considérées en prenant par exemple en compte la consommation induite par les ressources.

Parmi les différentes propositions qui ont été faites dans ce mémoire, certaines visaient à faciliter la création de modèles de niveau transactionnel pour le dimensionnement de systèmes. Pour continuer dans ce sens, une voie intéressante serait d'automatiser la création de ces modèles sur la base du modèle d'exécution générique proposé. Cet aspect nécessiterait la définition d'un outil permettant la définition de modèles de haut niveau et leur génération vers des outils de simulation. D'autres propositions ont été faites afin de réduire les temps de simulation nécessaires à l'exécution des modèles utilisés. Des améliorations pourraient encore être obtenues en cherchant à optimiser les couplages nécessaires entre les activités.

Il conviendrait aussi d'appliquer les travaux présentés sur des cas d'étude réels impliquant des expérimentations sur des architectures existantes afin de valider notre approche et de justifier de la pertinence des observations obtenues avec des modèles définis au niveau d'abstraction considéré.

Acronymes & Abréviations

3GPP	3rd Generation Partnership Project
ADRES	Architecture for Dynamically Reconfigurable Embedded Systems
ALPIN	Asynchronous Low-Power Innovative NoC
ALU	Arithmetic and Logic Unit
AMR	Adaptive Multi-Rate
ARM	Advanced RISC Machines
BCA	Bus Cycle Accurate
QPSK	Quadrature Phase-Shift Keying
CDMA 2000	Code Division Multiple Access 2000
CA	Cycle Accurate
CB	Code Block
CC	Cycle Callable
CEA-LETI	Commissariat à l'Energie Atomique-Laboratoire d'Electronique et de Technologie de l'Information
CPC	Cognitive Pilot Channel
CRC	Contrôle de Redondance Cyclique
DAB	Digital Audio Broadcasting
DSP	Digital Signal Processing
DVB	Digital Video Broadcasting
DVB-H	Digital Video Broadcasting - Handheld
E^2R	End-to-End Reconfigurability
E^3	End-to-End Efficiency
EDGE	Enhanced Data Rates for GPRS Evolution
ESSOR	European Secure SOftware defined Radio
ETSI	European Telecommunications Standards Institute
EVP	Embedded Vector Processor
FAUST	Flexible Architecture of Unified System for Telecom
FFT	Fast Fourier Transform
FOSFOR	Flexible Operating System FOr Reconfigurable platform
FPGA	Field-Programmable Gate Array
Gigabit WiMAX	Gigabit Worldwide Interoperability for Microwave Access
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile communications
HD	Haute Définition
HSPA	High Speed Packet Access
HTML	HyperText Markup Language
IEEE SCC41	Institute of Electrical and Electronics Engineers Standards Coordinating Committee 41
IFFT	Inverse Fast Fourier Transform
IMEC	Interuniversity MicroElectronics Centre
IMT-Advanced	International Mobile Telecommunications - Advanced
IP	Intellectual Property
ISP	Image Signal Processor
I-WLAN	Interworking - Wireless Local Area Network
JTRS	Joint Tactical Radio System
Lab-STICC	Laboratoire en Sciences et Technologies de l'Information, de la

	Communication et de la Connaissance
LLC	Logical Link Control
LTE	Long Term Evolution
LTE-Advanced	Long Term Evolution-Advanced
LUT	Look-Up Table
MAC	Medium Access Control
MARTE	Modeling and Analysis of Real-Time and Embedded systems
MARTES	Model-based Approach to Real-Time Embedded Systems development
MCSE	Méthodologie de Conception des Systèmes Electroniques
MPSoC	Multi-Processor System-on-Chip
NoTA	Network-On-Terminal Architecture
OFDMA	Orthogonal Frequency-Division Multiple Access
OMAP4	Open Multimedia Application Platform 4
OSCI	Open SystemC Initiative
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RAT	Radio Access Technology
RISC	Reduced Instruction Set Computer
RLC	Radio Link Control
RRS	Reconfigurable Radio Systems
RTL	Register Transfer Level
SCA	Software Communication Architecture
SCEE	Signal, Communication et Electronique Embarquée
SDR	Software Defined Radio
SMS	Short Message Service
SPIRIT	Structure for Packaging, Integrating and Re-using IP within Tool flows
SysML	Systems Modeling Language
TD-SDMA	Time Division Synchronous Code Division Multiple Access
TLM	Transaction-Level Modeling
TLM-PV	Transaction-Level Modeling Programmer's View
TLM-PVT	Transaction-Level Modeling Programmer's View with Timing
UMA	Unlicensed Mobile Access
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network
VLIW	Very Long Instruction Word
VTT	Valtion Teknillinen Tutkimuskeskus
UMA	Unlicensed Mobile Access
UML	Unified Modeling Language
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network

Bibliographie

[1] UMTS Forum, "Magic Mobile Future 2010-2020," 2005.

[2] C. Eriksson, T. Irnich, and M. Mustonen, "IST-4-027756 WINNER II, D 5.10.1 v1.0,The WINNER Role in the ITU Process Towards IMT-Advanced and Newly Identified Spectrum," 2007.

[3] Functionnality in early GSM releases.
[Online]. http://www.3gpp.org/article/functionality-in-early-gsm

[4] GPRS & EDGE, General Packet Radio Service / Enhanced Data rates for Global Evolution. [Online]. http://www.3gpp.org/article/gprs-edge

[5] UMTS, Universal Mobile Telecommunications System.
[Online]. http://www.3gpp.org/article/umts

[6] HSPA, High-Speed Packet Access. [Online]. http://www.3gpp.org/HSPA

[7] LTE, Long Term Evolution. [Online]. http://www.3gpp.org/LTE

[8] IEEE 802.11Wireless Local Area Networks, the Working Group for WLAN Standards. [Online]. http://www.ieee802.org/11/

[9] IEEE 802.16 Wireless Metropolitan Area Networks, the Working Group on Broadband Wireless Access. [Online]. http://www.ieee802.org/16/

[10] IEEE Task Group advances standards process for higer-speed 802.11 Wireless Local Area Networks.
[Online]. http://standards.ieee.org/announcements/pr_tgp802.11n.html

[11] IEEE 802.16e Task Group (Mobile WirelessMAN).
[Online]. http://www.ieee802.org/16/tge/

[12] DVB, The global standard for digital television. [Online]. http://www.dvb.org/

[13] DAB, The World Dab Forum. [Online]. http://www.worlddab.org/

[14] DVB-H, Global mobile TV. [Online]. http://www.dvb-h.org/

[15] D. Bourse and R. Tafazolli, "Beyond 3G / 4G Radio Access Technologies (RATs) and Standards Roadmaps," eMobility Technology Platform Whitepaper, 2007.

[16] ITU-R, "Framework and overall objectives of the future development of IMT-2000 and systems beyond IMT-2000," Recommendation ITU-R M.1645, 2003.

[17] M. Döttling, W. Mohr, and A. Osseiran, *Radio Technologies and Concepts for IMT-Advanced*. John Wiley & Sons, Oct. 2009.

[18] U. M. A. consortium, "Architecture (Stage2), R1.0.4," Technical specification, 2005.

[19] 3GPP, "3GPP system to Wireless Local Area Network (WLAN) interworking; System description," Technical specification 23.234 v8.0.0, Décembre 2008.

[20] IEEE 802.21. [Online]. http://www.ieee802.org/21/

[21] End-to-End Reconfigurability. (2007) IST project summary. [Online]. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/cnt/e2rii_en.pdf

[22] End-to-End Efficiency. [Online]. https://ict-e3.eu/

[23] J. Mitola and G. Q. Maguire, "Cognitive radio : making software radios more personal," *IEEE*, pp. 13-18, 1999.

[24] F. Marx, S. Hope, and A. Delautre, "System scenarios of end to end reconfigurability," *In proceedings of the SDR 04 Technical Conference (SDR'04)*, 2004.

[25] (2009) IEEE 1900.4. [Online]. http://www.scc41.org/

[26] ETSI World Class Standards. (2009) Reconfigurable radio. [Online]. http://www.etsi.org/website/homepage.aspx

[27] ETSI. (2009) Functional Architecture for the Management and Control of Reconfigurable Radio Systems. [Online].
http://www.etsi.org/deliver/etsi_tr/102600_102699/102682/01.01.01_60/tr_102682v010101p.pdf

[28] ETSI. (2009, Sep.) Cognitive Pilot Channel (CPC). [Online]. http://www.etsi.org/deliver/etsi_tr/102600_102699/102683/01.01.01_60/tr_102683v010101p.pdf

[29] K. van Berkel, "Multi-Core for Mobile Phones," *Design, Automation & Test in Europe Conference & Exhibition, DATE '09*, p. 6, 2009.

[30] M. Sauter, *Beyond 3G - Bringing Networks, Terminals and the Web Together: LTE, WiMAX, IMS, 4G Devices and the Mobile Web 2.0*. John Wiley & Sons, 2009.

[31] Y.-H. Park, S. Pasricha, F. J. Kurdahi, and N. Dutt, "A Multi-Granularity Power Modeling Methodology for Embedded Processors," in *International Conference on Hardware Software Codesign*, Atlanta, 2008, pp. 255-260.

[32] Texas Instrument. (2009) OMAP™ 4 Platform: OMAP4430/OMAP4440. [Online].
http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&n

avigationId=12843&contentId=53243

[33] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, pp. 26-38, 2005.

[34] J. Palicot and C. Roland, "La radio logicielle : enjeux, contraintes et perspectives," *Revue de l'électricité et de l'électronique*, 2001.

[35] Wireless Innovation SDR Forum, "Commercial Baseband Technology Overview: The Current State of Technology Development and Future Directions," 2010.

[36] W. Raab, J. Berthold, H. Eisenreich, and J.-U. Schluessler, "Low Power Design of the X-Gold® SDR 20 Baseband Processor," in *Design, Automation & Test in Europe Conference & Exhibition, DATE '10*, Dresden, 2010, p. 2.

[37] L. Bossuet, "Exploration de l'espace de conception des architectures reconfigurables," thèse de doctorat, 2004.

[38] P. Garcia, K. Compton, M. Schulte, E. Blem, and W. Fu, "An Overview of Reconfigurable Hardware in Embedded Systems," *EURASIP Journal on Embedded Systems*, pp. 1-19, 2006.

[39] L. Alaus, D. Noguet, and J. Palicot, "A Bank of coarse grain Common Operators for flexible multi standard SDR terminal," *SDR Forum 2010 European Reconfigurable Radio Technologies*, 2010.

[40] B. Bougard, B. De Sutter, and S. Rabou, "A Coarse-Grained Array based Baseband Processor for 100Mbps+ Software Defined Radio," in *Design, Automation and Test in Europe DATE'08*, Munich, Germany, 2008.

[41] I. Miro-Panades, F. Clermidy, P. Vivet, and A. Greiner, "Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture," *IEEE Computer Society*, pp. 139-148, 2008.

[42] Y. Thonnart, et al., "An asynchronous low-power innovative network-On-chip including design-for-test capabilities," in *International Conference on Advanced Technologies for Communications, ATC '09.*, Hai Phong, 2009, pp. 59-62.

[43] F. Clermidy, R. Lemaire, X. Popon, D. Ktenas, and Y. Thonnart, "An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application," in *12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD'09*, 2009, pp. 449-456.

[44] Joint Tactical Radio System (JTRS), "Software Communications Architecture (SCA)," Standard, 2006.

[45] C. Serra, "ESSOR Architecture and Standard Perspectives," in *SDR'09 Technical Conference*, Washington, 2009.

[46] R. Atakula, K. Boukis, and E. Nicollet, "Specification and preliminary design of mechanisms and processes enabling end to end dynamic reconfiguration," 2006.

[47] J.-P. Delahaye, "Plate-forme hétérogène reconfigurable application à la radio logicielle," Thèse de doctorat, 2007.

[48] L. Godard, "Modele de gestion hierarchique distribuée pour la reconfiguration et la prise de décision dans les equipements de radio cognitive," Thèse de doctorat, 2008.

[49] J. C. Benkhelifa, A. Granado, B. Huck, and E. Miramond, "A Framework for the Exploration of RTOS Dedicated to the Management of Hardware Reconfigurable Resources," in *International Conference on Reconfigurable Computing and FPGAs, 2008. ReConFig '08.*, Cancun, 2008, pp. 61-66.

[50] L. Devaux, D. Chillet, S. Pillement, and D. Demigny, "Flexible communication support for dynamically reconfigurable FPGAs," in *Southern Programmable Logic, SPL'09*, 2009, pp. 65-70.

[51] A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi, "Benefits and challenges for platform-based design," in *Design Automation Conference, DAC'04*, San Diego, 2004, pp. 409-414.

[52] The SPIRIT Consortium. (2006) Enabling innovative IP Re-Use And Design Automation. [Online]. http://www.spiritconsortium.org/home/

[53] CoWare. (2006) TLM peripheral modeling for Platform-Driven ESL Design Using the SystemC Modeling Library.

[Online]. http://www.coware.com/solutions/tlm.php

[54] CoFluent Design. (2004) Guess or model? Your choices drive your success. Application Note.

[55] G. Martin, "Overview of the MPSoC Design Challenge," in *Design Automation Conference, DAC'06*, San Francisco, 2006, pp. 274-279.

[56] D. Gajski and R. Kuhn, "Guest Editors' Introduction New VLSI Tools," in *IEEE Computer*, 1983, pp. 11-14.

[57] M. Office, "European Design Automation Raodmap 6th edition," 2009.

[58] L. Cai and D. Gajski, "Transaction Level Modeling: An Overview," in *International Conference on Hardware/Software Codesign and System Synthesis*, Newport Beach, 2003, pp. 19-24.

[59] R. Zurawski, *Embedded Systems Handbook*. Taylor & Francis Group, 2006.

[60] A.-A. Jerraya and G. Nicolescu, *Spécification et validation des systèmes monopuces*. Lavoisier, 2004.

[61] F. Ghenassia, *Transaction-Level Modeling with SystemC TLM Concepts and Applications for Embedded Systems*. Springer, 2005.

[62] A. Donlin, "Transaction level modeling: flows and use models," in

CODES+ISSS'04, Stockholm, 2004, pp. 75-80.

[63] J. Kreku, M. Hoppari, T. Kestilä, and Y. Qu, "Combining UML2 application and SystemC platform modelling for performance evaluation of real-time embedded systems," *EURASIP Journal on Embedded Systems*, 2008.

[64] J. Kreku, K. Tiensyrä, and G. Vanmeerbeeck, "Automatic workload generation for system-level exploration based on modified GCC compiler," in *Design, Automation and Test in Europe, DATE'10*, 2010, pp. 369-374.

[65] K. Kronlöf, S. Kontinen, I. Oliver, and T. Eriksson, "A Method for Mobile Terminal Platform Architecture Development," in *Forum on specification and Design Langages, FDL'06*, Darmstadt, 2006.

[66] M. Trautmann, et al., "Simulation Framework for Early Phase Exploration of SDR Platforms: A Case Study of Platform dimensionning," in *Design, Automation and Test in Europe, DATE'09*, Nice, 2009, pp. 312-316.

[67] A. Viehl, B. Sander, O. Bringmann, and W. Rosenstiel, "Integrated Requirement Evaluation of Non-Functional System-on-Chip Properties," in *Forum on Specification & Design Languages, FDL'08*, Stuttgart, 2008.

[68] S. Pasricha, N. Dutt, and M. Ben-Romdhane, "Extending the transaction level modeling approach for fast communication architecture exploration," in *Design Automation Conference, DAC'04*, San Diego, 2004, pp. 113-118.

[69] H.-P. Loeb and C. Sauer, "Exploration of Embedded Memories in SoCs using SystemC-based Functional Performance Models," in *Forum on Specification & Design Languages, FDL'09*, Sophia Antipolis, 2009.

[70] R. Beraha, I. Walter, I. Cidon, and A. Kolodny, "Leveraging Application-Level Requirements in the Design of a NoC for a 4G SoC - a Case Study," in *Design, Automation and Test in Europe, DATE'10*, Dresden, 2010, pp. 1408-1413.

[71] Modeling and specialization of platform and components MDA. [Online]. http://www.mopcom.fr/doku.php

[72] A. Koudri, et al., "Using MARTE in a Co-Design Methodology," in *MARTE UML profile workshop co-located with DATE'08*, Munich, 2008.

[73] Modeling and Analysis of Real-time and Embedded systems. [Online]. http://www.omgmarte.org/

[74] J. Vidal, F. de Lamotte, G. Gogniat, J.-P. Diguet, and P. Soulard, "UML design for dynamically reconfigurable multiprocessor embedded systems," in *Design Automation and Test in Europe, DATE'10*, Dresden, 2010.

[75] Object Management Group. The OMG's Unified Modeling Language. [Online]. http://www.uml.org/

[76] OMG. Unified Modeling Language. [Online]. http://www.uml.org/

[77] OMG Systems Modeling Language. [Online]. http://www.omgsysml.org/

[78] E. Riccobene, P. Scandurra, A. Rosti, and S. Bocchio, "A UML 2.0 profile for SystemC: toward high-level SoC design," *International Conference On Embedded Software, EMSOFT'05*, 2005.

[79] M. Mura, L. G. Murillo, and M. Prevostini, "Model-based Design Space Exploration for RTES with SysML and MARTE," *Forum on Specification and Design Languages, FDL'08*, pp. 389-394, 2008.

[80] [Online]. http://www.martes-itea.org/public/news.php

[81] [Online]. http://www.systemc.org/home/

[82] T. Grötker, S. Liao, G. Martin, and S. Swan, *System design with SystemC*. Boston: Springer, 2002.

[83] Open SystemC Initiative. [Online]. http://www.systemc.org/

[84] (2010) TLM-2.0 Standard Now Available.
[Online]. http://www.systemc.org/downloads/standards/tlm20/

[85] D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao, *SpecC: Specification Language and Methodology*. Boston: Springer, 2000.

[86] [Online]. http://www.systemverilog.org/

[87] D. Densmore, R. Passerone, and A. Sangiovanni-Vincentelli, "A platform-based taxonomy for esl design," *IEEE Design and Test of Computers*, vol. 23, no. 5, p. 359–374, 2006.

[88] CoFluent Studio. [Online].
http://www.cofluentdesign.com/index.php/Products_Services/cofluent-studio.html

[89] A. Agarwal, C.-D. Iskander, R. Shankar, and G. Hamza-Lup, "System-Level Modeling Environment: MLDesigner," *Systems Conference, 2nd Annual IEEE*, pp. 1-7, Avril 2008.

[90] Synopsys. Model-Based Algorithm Design, Simulation and Analysis – Get to Results Faster. [Online].
http://www.synopsys.com/TOOLS/SLD/DIGITALSIGNALPROCESSING/Pages/SystemStudio.aspx

[91] A. D. Pimentel, "The Artemis Workbench for system-level performance evaluation of embedded systems.pdf," *International Journal of Embedded Systems 2008 - Vol. 3, No.3*, pp. 181-196, 2008.

[92] A. Mihal, et al., "Developing Architectural Platforms: A Disciplined Approach," *IEEE Design and Test of Computers, vol. 19, no. 6*, pp. 6-16, 2002.

[93] F. Balarin, et al., "Metropolis : An integrated electronic system design environment," *Computer*, 2003.

[94] J. P. Calvez, *Spécification et conception des systèmes, une méthodologie.* Masson, 1991.

[95] J. P. Calvez, *Embedded real-time systems*. Wiley Series In Software Engineering Practice, 1993.

[96] D. Harel and M. Politi, *Modeling Reactive Systems With Statecharts : The Statemate Approach*. New York: McGraw-Hill, 1998.

[97] D. C Black, D. Jack, and B. Bill, *SystemC: From the Ground Up*, 2th ed. Springer, 2008.

[98] D. D. Gajski, *Principles of digital design*. Prentice-Hall, 1996.

[99] J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics Computation*, vol. 19, pp. 297-301, 1965.

[100] Xilinx. LogiCORE Fast Fourier Transform v7.1. [Online]. http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

[101] 3GPP. (2010) LTE. [Online]. http://www.3gpp.org/LTE

[102] 3GPP. (2010, Mar.) TS 36.201, Evolved Universal Terrestrial Radio Access (E-UTRA); LTE physical layer; General description. [Online]. http://www.3gpp.org/ftp/Specs/html-info/36201.htm

[103] J. Berkmann, C. Carbonelli, F. Dietrich, C. Drewes, and W. Xu, "On 3G LTE Terminal Implementation - Standard, Algorithms, Complexities and Challenges," *International Wireless Communications and Mobile Computing Conference. IWCMC '08.*, pp. 970-975, 2008.

[104] Texas Instrument, "Implementing a MAP Decoder for cdma2000 Turbo Codes on a TMS320C62x DSP Device," 2000.

[105] 3GPP. (2007) TS 26.071 Mandatory speech CODEC speech processing functions; AMR speech Codec; General description. [Online]. http://www.3gpp.org/ftp/Specs/html-info/26071.htm

[106] 3GPP, "Technical Specification Group, 25 series: Radio Aspects. (Release 8)".

[107] IEEE Computer Society. (2007) IEEE Standard association. IEEE 802.11-2007, IEEE standard for information technology - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. [Online]. http://www.ahltek.com/WhitePaperspdf/802.11-20%20specs/802.11-2007.pdf

[108] 3GPP. (2003) TR 25.933 version 5.4.0 Release 5, UMTS; IP transport in UTRAN. [Online]. http://ftp.3gpp.org/Specs/html-info/25933.htm

[109] 3GPP. (2002) TS 125.201, UMTS; Physical layer - general description. [Online]. http://www.3gpp.org/ftp/Specs/html-info/25201.htm

[110] 3GPP. (2002, Sep.) TS 125.212, UMTS; Multiplexing and channel coding. [Online]. http://www.3gpp.org/ftp/Specs/html-info/25212.htm

[111] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268-278, Mar. 1973.

[112] G. Krishnaiah, N. Engin, and S. Sawitzki, "Scalable Reconfigurable Channel Decoder Architecture for Future Wireless Handsets," *Design, Automation & Test in Europe Conference & Exhibition, DATE '07*, pp. 1563-1568, 2007.

Publications et communications

Conférences internationales avec actes et comité de lecture

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean Paul Calvez, "Transaction Level Modeling of an adaptive multi-standard and multi-application radio communication system", *Forum of specification and Design Languages (FDL'09)*, Sophia Antipolis, France, 22-24 Septembre 2009.

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean Paul Calvez, "Executable models for performance assessments of adaptive mobile systems", *Software Defined Radio Forum Technical Confernce* (SDR'09), Washington, USA, 1-4 Decembre 2009.

Sébastien Le Nours, Anthony Barreteau, Olivier Pasquier, "Modeling technique for simulation time speed-up of performance computation in transaction level models", *Forum of specification and Design Languages (FDL'10)*, Southampton, Angleterre, 14-16 Septembre 2010.

Colloques nationaux

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean Paul Calvez, "Démarche pour la crétion de modèles transactionnels pour l'évaluation de performances", *3ème colloque du GDR SoC-SiP*, Orsay, France, 10-12 juin 2009.

Anthony Barreteau, Sébastien Le Nours, Jean François Diouris, "Abstraction des propriétés non fonctionnelles au sein des modèles transactionnels", *Ecole d'hiver Francophone sur les Technologies de Conception des systèmes embarqués Hétérogènes (FETCH'10)*, Chamonix, France, 11-13 janvier 2010.

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean François Diouris, "Techniques de modélisation transactionnelle pour le dimensionnement des systems embarqués communicants", *4ème colloque national du GDR SoC-SiP*, Cergy-Pontoise, France, 9-11 juin 2010.

Communications orales

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean Paul Calvez, "Modélisation transactionnelle d'un système de radiocommunication multiapplication et multistandard", *Séminaire SCEE*, Rennes, France, 27 novembre 2008.

Anthony Barreteau, Sébastien Le Nours, Olivier Pasquier, Jean Paul Calvez, "Démarche pour le dimensionnement des futurs terminaux mobiles adaptatifs multiservices", *Journée thématique radio logicielle*, Rennes, France, 26 mars 2009.