République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique
et de la Technologie

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Cycle de Formation Doctorale
dans la Discipline Génie
Electrique
*Ingénierie des Systèmes
Informatiques*

Thèse de *DOCTORAT*
N° d'ordre: 2009– 96

THESE

Présentée à

L'École Nationale d'Ingénieurs de Sfax

En vue de l'obtention du

DOCTORAT

Dans la discipline Génie Electrique
Ingénierie des Systèmes Informatiques

Par

Jalel KTARI

(Ingénieur Génie Electrique)

APPROCHE ET ENVIRONNEMENT D'EXPLORATION ARCHITECTURALE BASSE CONSOMMATION

Soutenu le 5 Mars 2009, devant le jury composé de :

M.	**Mohamed Adel ALIMI**	*Président*
M.	**Slim BEN SAOUD**	*Rapporteur*
M.	**Jean-Luc DEKEYSER**	*Rapporteur*
M.	**Nouri MASMOUDI**	*Examinateur*
M.	**Mohamed ABID**	*Directeur de thèse*

DEDICACE

A mon père & à ma mère

Auxquels

Je dois ce que je suis
Que dieu vous protége
Et vous prête une bonne santé

Et une vie heureuse

A toute ma famille
Pour les encouragements continus

A tous mes amis

A tous ceux que j'aime
Et qui m'aiment

A toi

REMERCIEMENTS

C’est avec un grand plaisir que je réserve ces lignes en signe de gratitude et de reconnaissance à tous ceux qui ont contribué de prés ou de loin à l’élaboration de ce travail.

Je tiens à exprimer ma vive gratitude à mon directeur de thèse, Monsieur Mohamed ABID, Professeur à l’Ecole Nationale d’Ingénieurs de Sfax. Ses conseils et encouragements m’ont permis de surmonter toutes les difficultés du parcours. Qu’il trouve ici l’expression de mes sentiments sincères.

Mes remerciements s’adressent aussi à Monsieur Mohamed Adel ALIMI, Professeur et Directeur de l’Ecole Nationale d’Ingénieurs de Sfax pour l’intérêt qu’il a porté à ce travail en acceptant de me faire l’honneur de présider le jury de ma soutenance.

J’exprime ma reconnaissance à Monsieur Slim BEN SAOUD, Maître de Conférences à l'Institut National des Sciences Appliquées et de Technologie, ainsi qu’à Monsieur Jean-Luc DEKEYSER, Professeur à l’Université des Sciences et Technologies de Lille d’avoir accepté d’être rapporteurs de ma thèse.

Je remercie également Monsieur Nouri MASMOUDI, Professeur à l’Ecole Nationale d’Ingénieurs de Sfax, pour sa participation à mon jury.

Je remercie également Monsieur Mourid MARRAKCHI, Maître Assistant à l’Ecole Nationale d’Ingénieurs de Sfax, pour ses précieux conseils.

Par ailleurs, je remercie mon ami Dr. Issam MAALEJ qui m’a largement soutenu. Je lui suis très reconnaissant pour son aide, ses conseils et ses encouragements.

Je tiens également à remercier tous mes collègues et tous les membres de l’unité de recherche CES, à qui je souhaite une bonne continuation.

SOMMAIRE

Introduction générale

La complexité des systèmes embarqués est en pleine croissance afin de répondre aux exigences et aux critères de performances des nouvelles applications. Ceci rend la conception de plus en plus difficile en intégrant une multitude de fonctionnalités tout en respectant les contraintes de l'application. Par ailleurs, l'avènement des nouvelles technologies met en évidence la nécessité d'établir de nouvelles méthodologies de conception à un haut niveau d'abstraction. Ces méthodologies servent à mieux guider le concepteur lors du choix de la solution architecturale afin d'atteindre les objectifs souhaités entre autre la basse consommation.

En fait, la consommation de puissance et d'énergie est devenue une des contraintes principales lors de la conception des systèmes embarqués. En effet, les nouvelles applications nécessitent de plus en plus de puissance de calcul et de précision et par conséquent une augmentation de la consommation d'énergie et de puissance. Par ailleurs, vu le caractère de mobilité dans ces applications fonctionnant avec des batteries, la maîtrise de la consommation s'impose lors de la conception. Ceci permet d'une part d'augmenter la durée de vie des batteries qui se traduit par un gain en temps de communication d'un GSM par exemple, et de minimiser la dissipation thermique des composants qui influe sur la fiabilité du système d'autre part.

Objectif du sujet

- Les nouvelles applications embarquées ont clairement montré l'évolution croissante de la consommation. Si la consommation n'est pas réduite et maîtrisée, la complexité ou bien la performance des applications devra être réduite afin d'envisager une solution embarquable. D'où une dégradation importante, voire inacceptable pour l'utilisateur, de la qualité de service (QoS) associée aux services fournis.
- On sait que les systèmes mobiles devront atteindre des performances élevées : 100 Mops/mW pour les nœuds mobiles et 10 à 50 Mops/mW pour les stations de base. Si de telles performances peuvent être atteintes par les ASIC, cela représente une efficacité 10 fois supérieure aux DSP actuels, 100 fois supérieure aux microprocesseurs généralistes. Pour garantir l'adaptabilité des systèmes à un environnement hétérogène, il faut donc prospecter de nouvelles solutions architecturales logicielles et matérielles garantissant une performance

élevée, une grande flexibilité et une faible énergie. Seule une approche globale de haut niveau permet de caractériser et d'optimiser efficacement la consommation (Benoit et al., 2004).

- L'étape d'exploration architecturale est une étape critique dans le flot de mise en œuvre du produit. En effet, pour une application donnée, il est possible d'avoir une multitude d'architectures. Cet espace contient des architectures qui sont non réalisables, d'autres qui ne satisfont pas les contraintes, d'autres qui les satisfont et des architectures qui ont des performances optimales. Il incombe à l'étape d'exploration de trouver pour l'application, une architecture adéquate qui satisfait les contraintes et qui optimise au mieux sa consommation.

Organisation et contribution de ce rapport

Afin de maîtriser l'aspect faible consommation dans les systèmes embarqués, et afin de voir la possibilité de l'intégration de cet aspect dans l'exploration de l'espace des solutions à différents niveaux d'abstractions, une étude est faite sur les divers outils et méthodologies. De plus, une approche d'exploration basse consommation est proposée et validée.

Dans le premier chapitre du rapport, on présente les diverses techniques et méthodologies d'estimation de la consommation dans les systèmes embarqués à différents niveaux d'abstractions. Le deuxième chapitre est consacré à l'état de l'art de l'exploration de l'espace des solutions. On présente aussi les besoins, la nécessité et la complexité de l'exploration basse consommation lors de la conception.

Dans le troisième chapitre du rapport, on présente notre approche pour faire face aux difficultés d'exploration et de conception basse consommation. Elle consiste à proposer des modèles de performances riches et une technique d'exploration dite basse consommation. Un modèle complet est proposé afin de déduire les performances globales du système qui seront utilisées lors de l'exploration à travers une technique basée sur le recuit simulé. Cette heuristique permet d'exploiter la technique d'exploration selon plusieurs niveaux de granularité et ce afin de pouvoir choisir le niveau qui permet d'assurer une exploration précise et rapide.

Le dernier chapitre est consacré à l'étude et la modélisation de haut niveau du filtre FIR, de la FFT et des différentes fonctions de l'application MPEG-2 sur diverses architectures cibles. Les modèles établis serviront comme bibliothèque de modèles utiles pour bien mener le choix stratégique de l'architecture cible adéquate lors de l'exploration basse consommation. Et afin de valider l'approche, une étude probabiliste est proposée.

Chapitre

I

La conception faible consommation

(Techniques et Outils)

Chapitre I. La conception faible consommation

I.1 Introduction

Historiquement, les contraintes majeures lors de la conception des systèmes embarqués étaient essentiellement la performance et le coût. Avec la tendance vers les applications portatives et la forte densité d'intégration, l'énergie est devenue un facteur critique lors de la conception. En effet, la consommation est actuellement l'une des importantes métriques lors de la conception des systèmes embarqués. Il est à signaler que l'objectif essentiel de la conception faible consommation est non seulement minimiser la puissance, mais aussi augmenter la durée de vie des batteries et éviter les systèmes de refroidissement encombrants. Pour cela, plusieurs techniques et technologies sont développées et appliquées dans ce contexte. Ces différentes techniques seront présentées dans ce chapitre.

I.2 Les exigences des systèmes embarqués

La conception logiciel/matériel de haut niveau permet de maîtriser la conception des systèmes complexes (Figure 1) et d'approuver leurs continuités. L'analyse des performances pendant une phase avancée de conception et avant la fabrication permet une exploration rapide de plusieurs alternatives d'architecture, ce qui offre au concepteur une meilleure visibilité et une grande réactivité vis-à-vis des changements technologiques (fiabilité, optimisation, flexibilité, migration etc.).

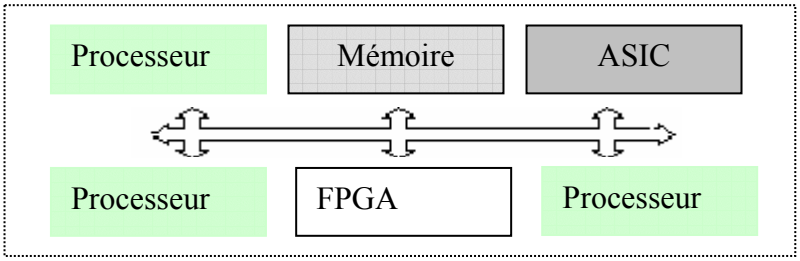


Figure 1. *Architecture d'un système embarqué mixte (HW/SW)*

Les prévisions pour l'évolution des applications de traitement de signal et d'image, pour les systèmes mobiles par exemple, montrent leur impact sur la conception des circuits et des systèmes embarqués. En particulier, à l'heure actuelle, il est indispensable de tenir compte de la consommation (en puissance et en énergie) comme critère de développement d'un système (Tableau 1) au même titre que la surface et la vitesse. En effet, la fréquence de

fonctionnement élevée, le nombre de ressources mises en œuvre et le degré d'intégration, contribuent à atteindre les limites physiques supportables par les circuits. La maîtrise de la consommation est donc un problème majeur dans la conception des systèmes embarqués.

On sait que si l'optimisation de la consommation doit intervenir à chaque niveau de la conception, c'est cependant aux plus hauts niveaux que les gains attendus sont les plus importants : 20 à 50% au niveau technologique contre 10 à 20x au niveau système (Rabaey et al., 1996) (Fei et al., 2003). Comme la partie logicielle représente au moins 70% du coût du développement des systèmes complexes et que la taille du code des applications TDSI double tous les deux ans, on conçoit qu'il est indispensable de disposer, au plus tôt dans la conception, de métriques de consommation fiables caractérisant aussi bien les parties logicielles et les parties matérielles (ITRS, 2004).

Tableau 1. *Les exigences fonctionnelles des systèmes PDA (ITRS, 2004)*

Année	2004	2007		2012	2015
Technologie (nm)	90	65		45	32
Voltage (V)	1.2	0.8		0.6	0.5
Fréquence (MHz)	300	600		900	1200
Application	Traitement Image	Vidéo temps réel Codec (MPEG4)		Interprétation temps réel	
	Web E-Mail	TV Téléphone reconnaissance de voix Cryptage		TV Téléphone reconnaissance de voix	
Performance (GOPS)	0.3	2	14	77	461
Puissance (W)	0.1	0.1	0.1	0.1	0.1
Puissance Standby (mW)	2	2	2	2	2
Capacité batterie (Wh/Kg)	120	200		400	

I.3 Système faible consommation

Diverses méthodes existent afin d'obtenir un système à faible consommation :

- Modéliser le logiciel pour diminuer le coût énergétique de son exécution : optimisation du code des applications, étude du comportement des applications et des processeurs en fonction des paramètres du code.

- La seconde méthode consiste à concevoir de composants spécifiques pour consommer le minimum d'énergie. En fait, c'est avec l'explosion du marché des systèmes embarqués et l'apparition des problèmes de surchauffe des composants, que cet aspect a été

pris en compte au niveau de la conception. Comme techniques de limitation de consommation des composants, on cite la diminution de la tension d'alimentation, l'activation séparée des blocs logiques et le contrôle du taux d'activité des données.

- La troisième méthode consiste à réaliser une collaboration entre le logiciel et le matériel afin d'optimiser la consommation totale du système. Elle s'appuie sur des mécanismes matériels pour diminuer la consommation et utilise le logiciel pour une meilleure décision en activant ou non ces mécanismes. Cette méthode est décrite dans le cadre de la mise en veille des périphériques ainsi que sur l'adaptation dynamique de la vitesse du processeur, etc.

I.4 La maîtrise de la consommation

Les principales sources de dissipation de puissance dans un circuit numérique sont données dans l'équation suivante (équation 1).

La puissance statique P_s peut être, dans certains cas, négligée pour des circuits de type CMOS. Mais avec les taux d'intégration actuels et les nouvelles technologies, elle est loin d'être négligeable.

$$P_{moy} = P_d + P_{cc} + P_f = \alpha . C . Vdd^2 . f + Vdd . I_{cc} + Vdd . I_f \quad (1)$$

où P_{moy} est la puissance moyenne dissipée par le circuit, P_d la puissance dynamique causée par la charge et la décharge de la capacité C. P_{cc} et P_f reflètent la puissance dissipée due aux courants de court-circuit et de fuite respectivement.

Avec Vdd la tension d'alimentation, f la fréquence d'horloge, C la capacité physique du circuit et α le taux de commutation du circuit. Ces deux derniers paramètres sont souvent regroupés en un terme C_{eff}.

I.5 La réduction de la consommation

La consommation moyenne dans un circuit dépend de :

- La tension d'alimentation, son impact quadratique en ce qui concerne la consommation permet d'envisager un gain important. Il faudra cependant rester avec V_{dd} > 2*la tension de seuil (V_T) (Rabaey et al.,1996) ou même plus actuellement. Et ceci pour éviter une augmentation importante du temps de propagation, qui est proportionnel à $C_l*V_{dd}/(V_{dd}-V_T)^2$, ce qui cause un ralentissement du fonctionnement.
- La fréquence d'horloge peut être réduite en se basant sur des techniques d'optimisation du chemin critique logique (arbre de l'horloge) ou bien des méthodes

(parallélisme, pipeline) aux niveaux algorithmiques et architecturaux. De cette manière, on évite le fonctionnement à des fréquences plus élevées.

- L'activité du circuit qui se manifeste par le nombre de commutations au niveau des portes. Afin de réduire cette activité, il est utile de bien codifier les données de telle façon qu'on minimise les changements de niveau logique et éventuellement minimiser les commutations parasites.
- La capacité effective qui est un paramètre technologique : les connexions aux composants externes ont typiquement une capacité beaucoup plus élevée que les connexions aux ressources sur la puce. Donc pour économiser l'énergie, la minimisation de l'utilisation des accès externes et de la commutation est sollicitée. En fait, l'accès à la mémoire externe consomme beaucoup d'énergie. Ainsi, une façon de réduire la capacité est de réduire ces accès et d'optimiser le système en employant des ressources internes comme la mémoire cache et les registres.

I.6 Méthodologies de réduction de la consommation

Il est possible d'utiliser des techniques de conception basse consommation à différentes étapes de la conception d'un système. La conception descendante consiste à partir du niveau le plus abstrait d'atteindre le niveau le plus bas. Pour ces systèmes, on distingue cinq niveaux différents: (Havinga et al., 2000) (Nikolaidis et al., 2005)

- Le niveau système
- Le niveau algorithmique
- Le niveau architectural (RTL: Register Transfert Level)
- Le niveau logique
- Le niveau électrique et physique

Par exemple, au niveau système : les modules inactifs peuvent être désactivés afin de minimiser les pertes énergétiques. Au niveau architectural, le parallélisme matériel peut être employé pour réduire les interconnections sans dégrader la sortie du système. Au niveau technologique, plusieurs optimisations peuvent être appliquées au niveau porte.

Les gains en consommation à chaque niveau de conception sont détaillés avec les outils dans le tableau 2: (Rabaey et al., 1996) (Laurent, 2002) (Beak et al.,2004) (Sequence, 2005) (Stanley et al.,2004) (Xilinx, 2006) (Minh et al., 2003) (Shin et al., 2002).

Tableau 2 : *Techniques et outils de réduction de la consommation*

Niveau	Domaine	Outils	Intervention
Système	Processeur travaillant avec plusieurs fréquences	- *ORINOCO*	-Mises en veille -Voltage scaling -Partitionnement (hw/sw) -Contrôle d'activité
Algorithmique	-Sources de consommation mal maîtrisées -Architecture peu connue	*-GAUT_LP* *-Orinoco (Offis)* *-SoftExplorer* *-ePRO* *-SES*	-Réduire les accès mémoire -Réduire les ruptures de pipeline -Diminuer le taux de défaut de cache
RTL	-Sources de consommation à peu près maîtrisées -Implantation réelle inconnue	*-Design Powe (Synopsys)* *-Power Theater Sequence* *-GAUT_LP* *- Wattch, Simplepower* *-DSP-PP*	-Parallélisme/Pipeline -Encodage de bus -Eteindre les modules inutilisés
Logique	-Nature des signaux entrés influence fortement la consommation du circuit	*-Cooltime* *-QuickPower (Mentor)* *-DesignPower(Synopsys)* *PowerTheater(Sequence)* *-Accupower* *-Xpower(Xilinx)*	-Format de codage -Extraction des sous-fonctions communes -Partage de ressources -Eliminer les transitions parasites
Physique	Source de consommation clairement identifiée	*-Coolpower* *-Spice* *-Accupower*	-Transistor sizing -Actions sur les seuils -Actions sur Vdd

Etant donné une spécification de conception, le concepteur a plusieurs choix sur les différents niveaux d'abstraction. Le designer doit choisir par exemple un algorithme particulier, concevoir ou exploiter une architecture qui peut être employée, et déterminer les divers paramètres influants comme la tension et fréquence d'horloge. Cet espace de conception multidimensionnel offre une grande gamme de compromis possibles. Et l'influence la plus remarquable sur la propriété de conception est obtenue aux plus hauts niveaux.

Donc les décisions de conception les plus efficaces dérivent du choix et de l'optimisation des architectures et des algorithmes aux niveaux les plus hauts. Plusieurs chercheurs (Havinga et al., 2000)(Garcia et al., 2005) ont montré que la conception au niveau architectural et système peut avoir un fort impact sur la consommation. Cependant en concevant au niveau système, le problème est de prévoir l'efficacité des décisions de conception. En fait, les détails d'implémentation peuvent être exactement estimés seulement au niveau technologique et pas aux niveaux d'abstraction plus hauts.

I.7 Techniques et outils d'estimation

I.7.1 Techniques

Diverses techniques d'estimation sont basées sur une méthode appelée Analyse de la Puissance au niveau Instructions (ILPA). L'ILPA a été développé à l'université de Princeton par Vivek Tiwari et al (Tiwari et al.,1996). Cette méthode était souvent considérée comme référence dans l'estimation de la consommation. Elle est applicable théoriquement à tous les processeurs que ce soit les processeurs généraux (Pentium, Athlon,…) ou les processeurs spécifiques (DSP).

Dans les systèmes basés sur microprocesseur, on peut modéliser la dissipation comme une fonction du logiciel (des instructions) étant exécutée sur une plate-forme matérielle. Les techniques d'évaluation de consommation du logiciel dans la littérature peuvent être triées dans ces catégories :

I.7.1.1 Analyse de la puissance au niveau instructions (ILPA)

Cette méthode est proposée afin d'évaluer la dissipation d'une partie du logiciel. L'idée de base est d'associer la puissance consommée avec l'exécution d'une instruction individuelle. La modélisation de cette méthode est décrite généralement par :

$$E=\sum_{i}(B_i*N_i)+\sum_{i,j}(O_{ij}*N_{ij})+\sum_{K}S_k \qquad (2)$$

Avec B_i l'énergie dissipée par l'instruction individuelle i, O_{ij} reflète la puissance due au changement entre deux instructions consécutives (i,j). En effet, il apparaît un surcoût dû au passage d'une instruction à l'autre c à d qu'un certain nombre de bits commutent du fait du changement du code de l'instruction (Brandolese et al., 2000). Le modèle doit donc tenir compte de cette consommation inter-instruction ce qui oblige à mesurer les consommations de toutes les combinaisons possibles deux à deux. S_k représente l'énergie due aux ruptures de pipeline et aux défauts de cache.

La figure 2 représente une plate forme utilisée afin de matérialiser cette méthode (Russell et al.,1998). Pour mesurer la puissance, une résistance de précision a été placée en série avec l'alimentation du processeur.

La consommation du processeur est calculée par cette formule (3) :

$$P(t)=I(t)V(t)=\frac{V_1(t)-V_2(t)}{R}*V_2(t) \qquad (3)$$

Pour déterminer Bi et Oij de l'équation 2 il faut faire un ensemble de tests des cas possibles. Mais ça engendre un nombre important de mesures ≈110 000 mesures pour un processeur Intel de 331 instructions (Li et al., 2003).

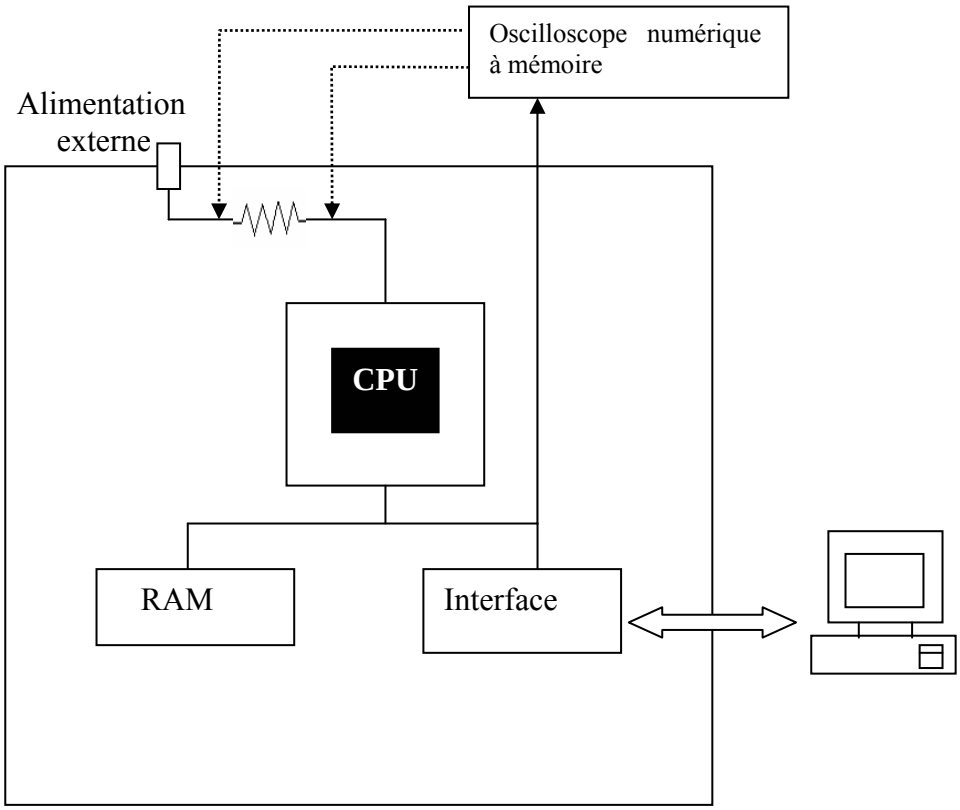


Figure 2 : *Plate forme expérimentale pour la mesure*

V. Tiwari dans (Tiwari et al.,1996) a fait l'expérience sur les processeurs commerciaux :

- Intel 486 DX2-S, 40 MHz, c'est un processeur CISC (Complex Instruction Set Computer) basé sur l'architecture du X86.
- FujiTsu SPARClite MB86934, 20 MHz, c'est un processeur 32 bits, RISC (Reduced Instruction Set Computer)

L'expérience est faite par le procédé décrit au-dessus, le tableau 3 montre quelques résultats obtenus :

Tableau 3 : *Le coût des instructions de bases du 486 et '934*

	Intel 486DX2				FujiTsu SPARClite MB86934			
N°	Instruction	Courant (mA)	Cycles	Energie(nJ)	Instruction	Courant (mA)	Cycles	Energie(nJ)
1	NOP	276	1	22.7	NOP	198	1	32.6
2	Mov dx,bx	302	1	24.9	St %io,[%IO]	346	2	114
3	Jmp	373	3	92.3	Ld [%IO],%io	213	1	35.1
4	Add dx,bx	314	1	25.9				

Cette méthode est assez efficace et fiable pour les architectures ayant un jeu d'instructions limité. Par contre, pour les architectures VLIW (Very Long Instruction Word), le nombre de mesures à réaliser devient important [$O(N^{2k})$ avec N le nombre d'instructions et k l'ordre du VLIW]. Pour cela d'autres méthodes ont été développées afin de surmonter ce problème.

I.7.1.2 Caractérisation par macro-modélisation

Au lieu d'évaluer la puissance au niveau instruction, le niveau fonctionnel logiciel, qui est une technique de macro-modélisation, traite les fonctions ou les sous-routines comme "des boîtes noires" et construit les macro-modèles qui corrèlent la puissance avec un jeu de caractéristiques. Telles caractéristiques de puissance peuvent être obtenues et rassemblées en employant une structure de simulation d'énergie à bas niveau (Li et al., 2003). La puissance dans ce cas est :

$$P = \sum_i W_i * C_i \qquad (4)$$

Avec Wi sont les coefficients du macro-modèle à déterminer. Des méthodes mathématiques de régression sont mises en jeu afin d'identifier les Wi optimales, basées par l'application de paires d'entrée/sortie bien connues. Le problème-clé de cette macro-modélisation est comment choisir les Ci qui représentent une corrélation de la puissance avec ces boites noires, qui peuvent efficacement capturer les caractéristiques de puissance d'une sous-routine logicielle donnée dans des circonstances diverses.

I.7.1.3 FLPA

Les techniques d'estimation classiques ont souvent leurs limites. En effet, avec ces méthodes, la consommation due à la communication avec l'extérieur (accès à la mémoire externe, défaut de cache) n'est pas considérée. Par ailleurs, ces techniques sont assez complexes à exploiter pour les nouvelles architectures ayant un pipeline profond. Afin de surmonter ces problèmes, (Laurent et al., 2007) (Julien et al., 2004) a proposé une méthode permettant de réduire la complexité de l'estimation. Cette méthode est basée sur une analyse fonctionnelle de la cible du point de vue consommation (Functional Level Power Analysis : FLPA) ; elle est indépendante du niveau d'abstraction (assembleur ou C). Grâce à cette analyse, un nombre limité de mesures suffit pour déterminer le modèle de consommation de la cible. De plus, elle prend en compte toutes les fonctions du processeur que ce soit le contrôle du pipeline, les unités de traitement, les mémoires internes ainsi que les défauts de cache, ce qui n'est pas le cas avec les méthodes au niveau instruction.

I.7.1.3.1 Méthodologie

La méthodologie d'estimation de la consommation FLPA, illustrée par la figure 3, est constituée de deux parties : la définition du modèle et le processus d'estimation.

- La définition du modèle de puissance du processeur est réalisée une seule fois par cible. Elle est basée sur l'analyse fonctionnelle de l'architecture cible de point de vue consommation. Cette analyse permettra de déterminer un modèle de puissance basé sur des lois de consommation qui représentent le comportement en courant du cœur du DSP. Ces lois sont des fonctions mathématiques déterminées à partir d'un nombre réduit de mesures physiques réalisées sur la cible et dépendant de paramètres algorithmiques et de configuration. La FLPA permet de déterminer quels sont les paramètres pertinents de point de vue consommation pour un processeur donné. Par ailleurs, les paramètres algorithmiques sont des ratios (variant de 0 à 1) qui représentent le taux d'activité entre chaque bloc fonctionnel du DSP; par exemple, le taux de parallélisme, le taux de défaut de cache…Les paramètres de configuration sont définis par le concepteur.

- Le processus d'estimation est réalisé à chaque fois que la consommation d'un algorithme doit être déterminée. Au niveau C, les paramètres algorithmiques sont estimés en utilisant un modèle de prédiction. Il suffit ensuite d'utiliser les lois de consommation établies pour l'architecture cible pour connaître la consommation de l'application. (Laurent et al., 2002), (Ktari et al., 2005)

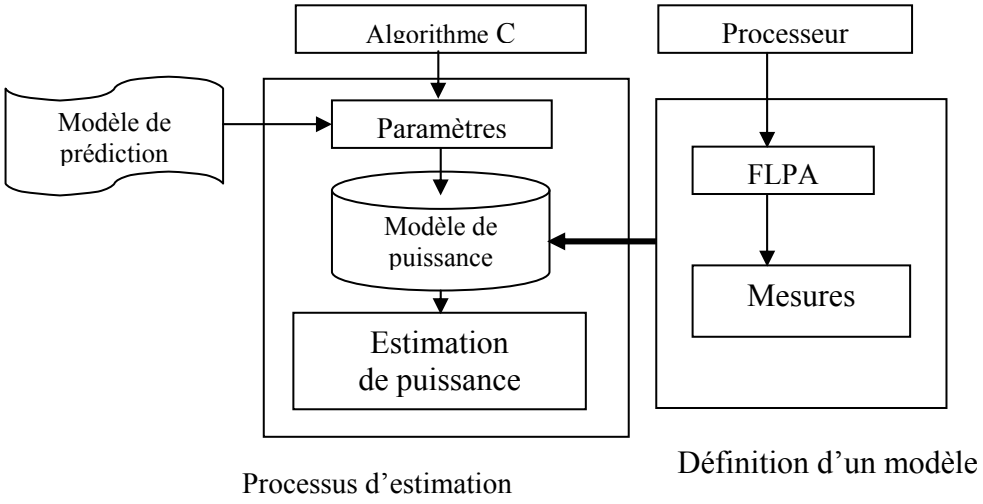


Figure 3 : *Méthodologie de l'estimation FLPA*

I.7.1.3.2 Outil : SoftExplorer

Cet outil (Laurent et al., 2007) automatique a comme entrée le code C ou ASM d'une application pour estimer sa consommation en terme de puissance et énergie. La structure de SoftExplorer est illustrée dans la figure 4. Il est basé sur trois modèles complémentaires :

- Le modèle du processeur : également appelé le modèle de puissance représente la manière dont la consommation du processeur change avec son activité.

- Le modèle de l'algorithme : représentant le lien entre l'algorithme et l'activité qu'il induit dans le processeur.

- Le modèle du compilateur : également appelé le modèle de prédiction qui représente le comportement du compilateur qui dépend des options choisies par le programmeur pendant la compilation, avec un fort impact sur le code généré, et ainsi sur l'activité du processeur.

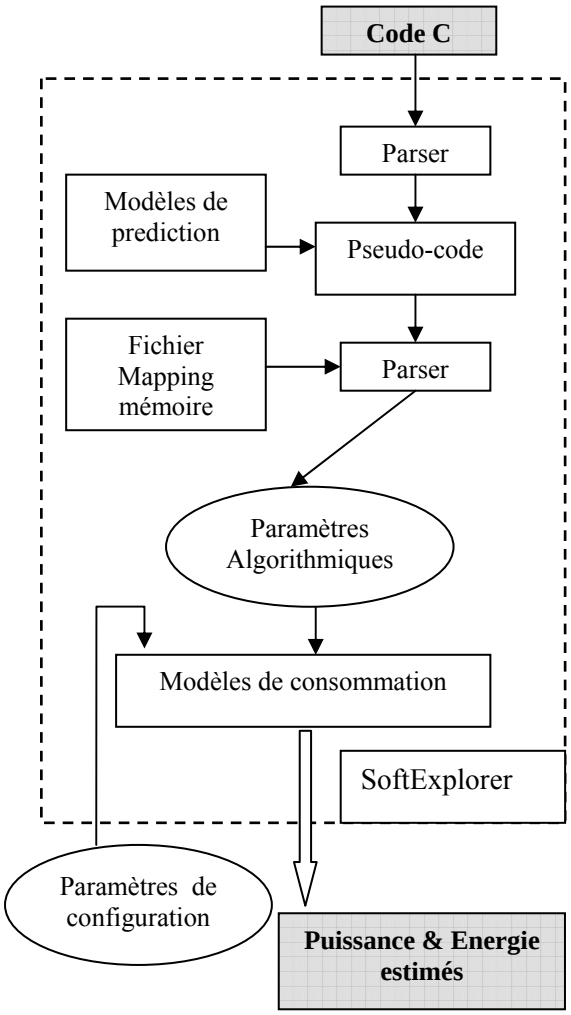


Figure 4 : *Flot d'estimation avec SoftExplorer*

Cet outil disponible sera exploité dans ce travail (Ktari et al., 2007) lors de la modélisation des applications écrites en ANSI-C. Et ceci en parallèle avec les mesures réalisées sur cartes DSP.

I.7.2 Plate-forme SEQUENCE

Sequence Design, Inc fournit un ensemble d'outils commerciaux complémentaires avec le flot de conception existant (Synopsys, Cadence). (Figure 5)

Cette plate-forme permet aux concepteurs des systèmes sur puce de réduire le coût du temps de mise sur le marché (*time to market*) en améliorant la performance et en limitant la consommation. En effet, elle permet de tenir compte de l'aspect faible consommation (analyse et optimisation de puissance) dans tous les niveaux d'abstractions du flot de conception. (Sequence, 2005)

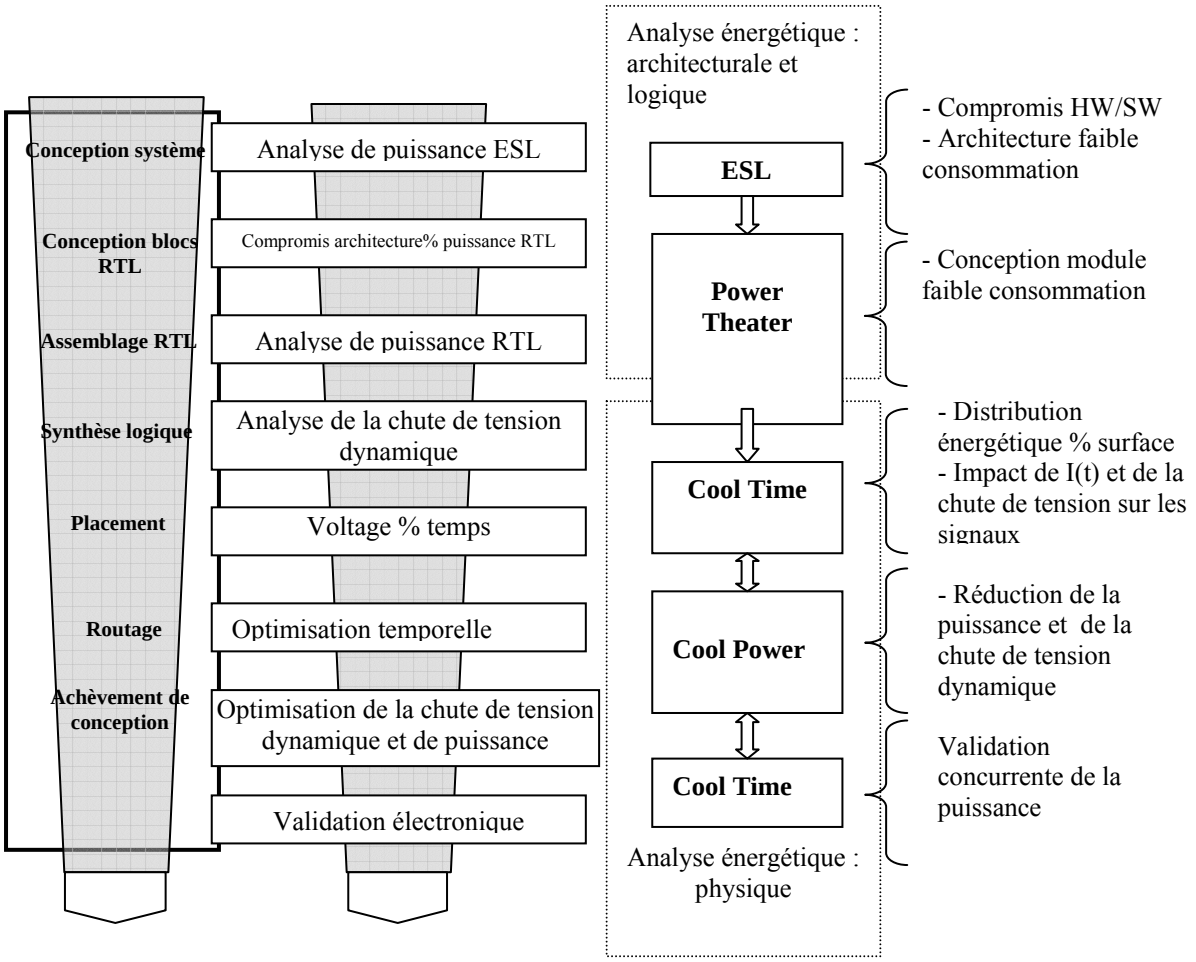


Figure 5 : *Flot de conception Sequence design*

I.7.2.1 ESL : (Electronic System Level)

I.7.2.1.1 Introduction

Le défi majeur lors de la conception des SoCs est de concevoir et d'implémenter une architecture sur puce optimale qui soit performante, faible consommation et de surface réduite dans une plate-forme matérielle/logicielle. Cet acte d'équilibrage entre ces diverses contraintes nécessite l'exploration des architectures aux niveaux d'abstraction plus haut que le niveau RTL. C'est pour cette raison, un intérêt est accordé pour adopter SystemC comme langage de conception au niveau système électronique (Electronic System Level : ESL).

La technique d'estimation de puissance ESL doit répondre à des exigences critiques. En effet, les architectures des SoCs sont soumises à diverses contraintes. Pour cela, les concepteurs ont besoin des résultats rapidement ainsi que le *feedback* sur la puissance et l'énergie consommées de tous les modules de la puce.

I.7.2.1.2 Caractéristiques

La technologie d'estimation ESL fournit les informations énergétiques critiques requises par le concepteur de SoC lors de l'évaluation de la consommation au niveau Système-C. En effet, elle intègre avec le synthétiseur de Système-C des outils d'estimation exploités par les programmeurs pour optimiser les algorithmes. (Figure 6)

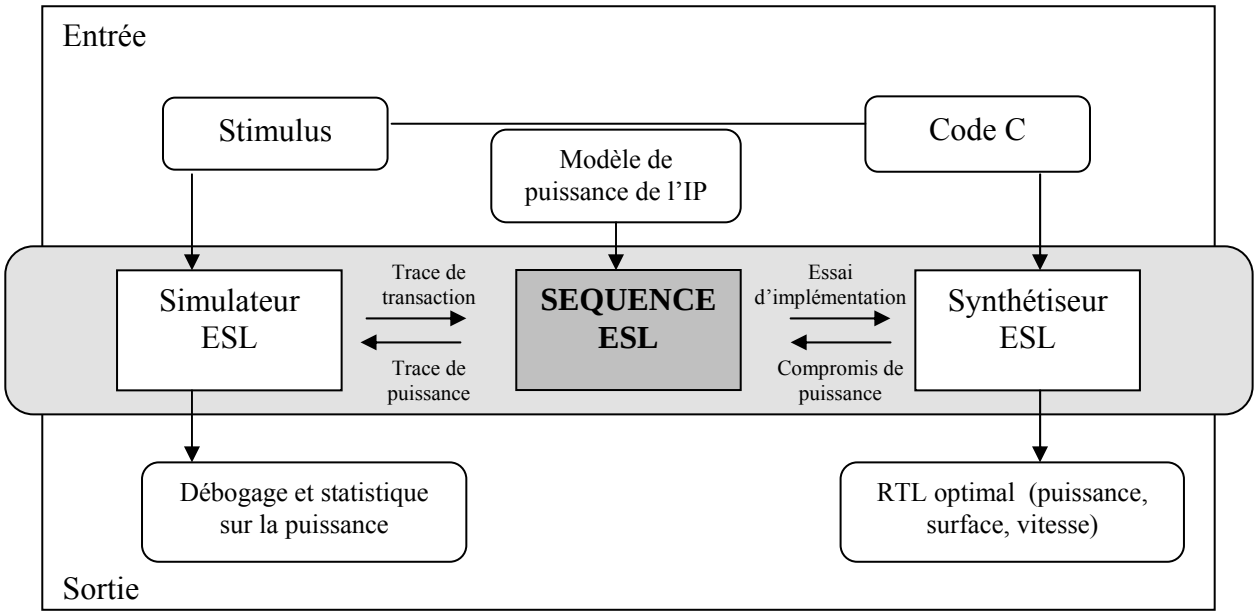


Figure 6: *ESL: Flot de conception*

I.7.2.2 Power Theater

Réduire la consommation si tôt, avant la synthèse, économise considérablement la dissipation de puissance. Power theater est un ensemble d'outils garantissant une efficacité maximale de la puissance lors de la conception. (Sequence, 2005)

À travers Power Theater, Sequence offre des solutions rapides et précises aux niveaux RTL et logique qui analysent et réduisent la dissipation de puissance lors de la conception d'un SoC. Cet outil analyse, affiche et aide l'utilisateur à réduire la puissance dissipée dans la puce entière et au niveau de chaque module. Il a comme entrée du code Verilog, VHDL et Synopsys Liberty (.LIB)

I.7.2.2.1 Avantages

Power Theater offre aux concepteurs divers avantages :

- Analyse de la puissance et de l'énergie assez tôt dans le flot de conception facilitant ainsi la réduction de la dissipation et le contrôle du coût énergétique.

- Assurer l'analyse « Zero-sim » de la puissance des modules RTL, sans recours aux vecteurs de tests : *testbenchs*.

- Voir l'impact simultané de tous les modules communiquant (mémoire, I/O, contrôle du chemin des données) dans la puce.

- Déterminer les cycles critiques de dissipation.

I.7.2.2.2 Méthodologie de réduction

Cet outil intègre des agents de réduction de la consommation au niveau RTL appelés WattBots qui mesurent de façon automatique l'impact de tout changement potentiel de l'architecture sur la dissipation. (Figure 7)

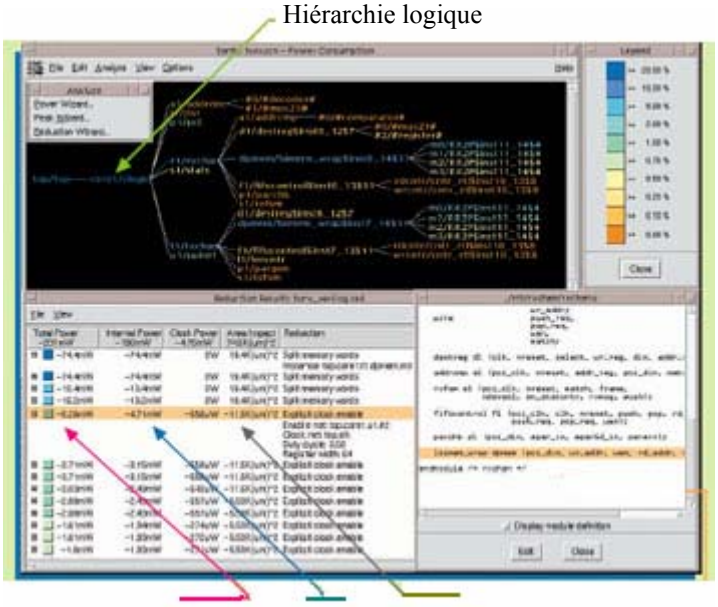


Figure 7 : *Analyse de puissance avec le débogueur RTL Cool PowerTheater*

Chaque WattBots est conçu pour identifier un type spécifique de possibilité de réduction énergétique. En effet, WattBots intègre tous les principaux types de circuits pouvant être exploité lors de la conception, y compris le contrôle, le chemin de données, l'E/S, la mémoire et l'horloge. Et pour chaque possibilité identifiée, Power Theater propose :

-Des modifications RTL spécifiques,

-Une quantification du gain énergétique résultant, en tenant compte des diverses contraintes mises en jeu comme la surface.

De cette façon, le concepteur a la possibilité de choisir l'architecture la mieux adaptée aux contraintes de conception.

I.7.2.3 Cool Time

I.7.2.3.1 Avantages

CoolTime (Sequence, 2005) est un moyen facilitant l'analyse simultanée de l'intégrité de la puce: la tension d'alimentation électrique et la synchronisation. En regroupant divers outils, CoolTime rend l'analyse des effets électriques interdépendants précise et convergente. Il partage une plate-forme commune avec CoolPower pour assurer une conception rapide qui tient compte des chutes de tension, courants de fuite et la synchronisation. (Figure 8)

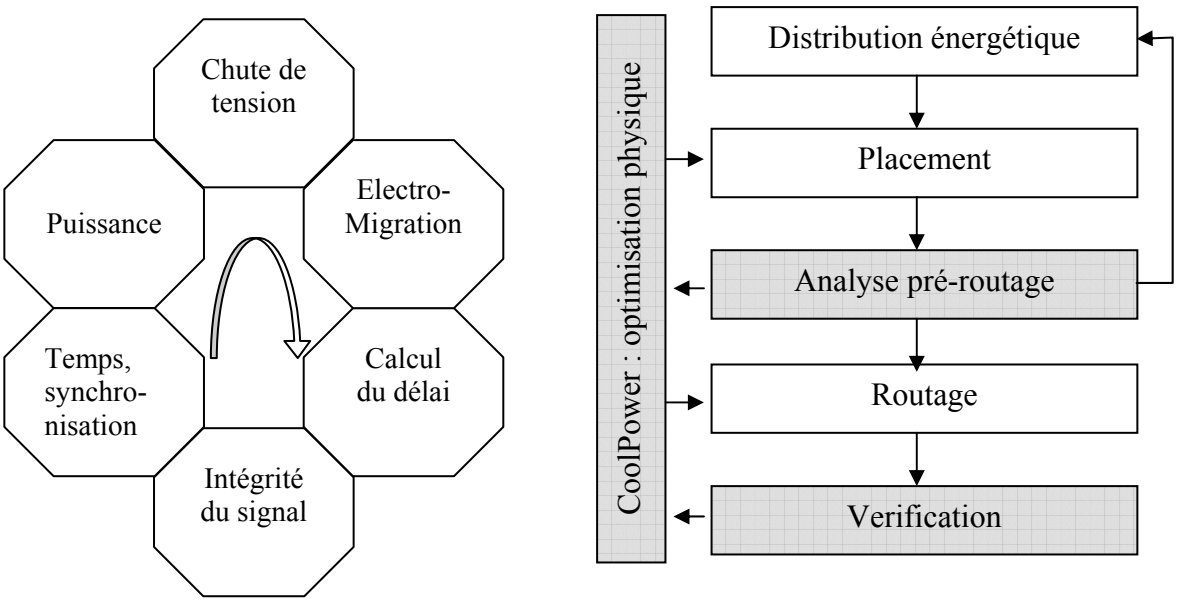


Figure 8 : *Analyse électrique concurrente avec CoolTime*

I.7.2.3.2 Caractéristiques

- CoolTime peut être utilisé si tôt dans le flot de conception réduisant ainsi les changements exigés dans les étapes postérieures. En partant du placement initial, l'analyse du courant statique peut être faite afin de valoriser la consommation lors du routage. Ainsi on assure une chute de tension dynamique dans des marges acceptables lors de la conception.

- CoolTime permet l'analyse de la chute de tension dynamique. En effet, un modèle de caractérisation intégré dans l'outil génère des modèles de courant sous forme d'onde. Et avec une étude concurrente, CoolTime analyse les effets interdépendants des événements, des courants et des tensions lors du régime transitoire.
- Il supporte les techniques de simulation (testbenchs) avec ou sans vecteurs de simulation. Contrairement aux techniques probabilistes, l'algorithme de création des stimulus prévoit la chute de tension au pire des cas.
- L'outil crée un modèle RLC complet de la puce ainsi que des parasites. Il supporte l'inductance mutuelle et les sources contrôlées de courant et de tension.

I.7.2.4 Cool Power

CoolPower (Sequence, 2005) prévoit et améliore la conception avant et après le routage. En effet, il offre à l'utilisateur la capacité d'optimiser de façon interactive la conception hiérarchique des millions de portes au niveau blocs.(Figure 9)

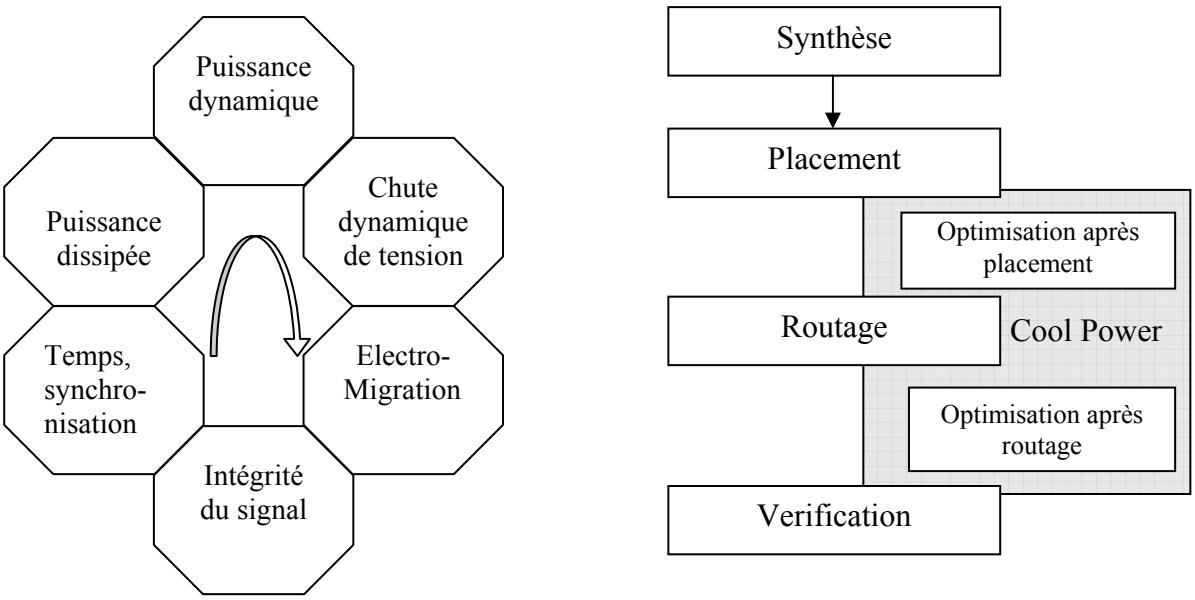


Figure 9 : *Analyse et optimisation électrique concurrente avec CoolPower*

En plus, l'outil réduit la dissipation statique et dynamique au niveau physique à l'aide de l'analyse concurrente de CoolTime tout en respectant la synchronisation et l'intégrité du signal. En effet, il réduit la puissance dynamique en appliquant et en testant divers changements sur le *netlist* qui optimisent les pertes énergétiques. Par ailleurs, CoolPower fixe automatiquement les problèmes provoqués par la chute de tension dynamique en insérant des capacités de découplage et en modifiant les placements pour éviter les points chauds.

En résumé, la plateforme Sequence offre une gamme d'outils payants orientée vers la conception et l'estimation des performances des SoC et des ASIC, mais non pas vers les DSPs commerciaux. Par ailleurs, cette plateforme reste encore un produit en cours de développement.

I.7.3 SES Scanner

SES (Seoul national university Energy Scanner) (Shin et al., 2002) est un outil qui fournit des informations sur la puissance et l'énergie consommées par le programme embarqué au niveau cycle, en visant son optimisation. Il associe ces informations récupérées par mesure sur carte avec le code C. Avec le débogueur GNU, jouant le rôle d'interface, diverses informations sur la consommation sont extraites au niveau C facilitant à l'utilisateur d'identifier les points chauds de l'application embarquée. SES comporte 3 modules logiques : l'estimateur d'énergie, l'analyseur et l'interface utilisateur. (Figure 10)

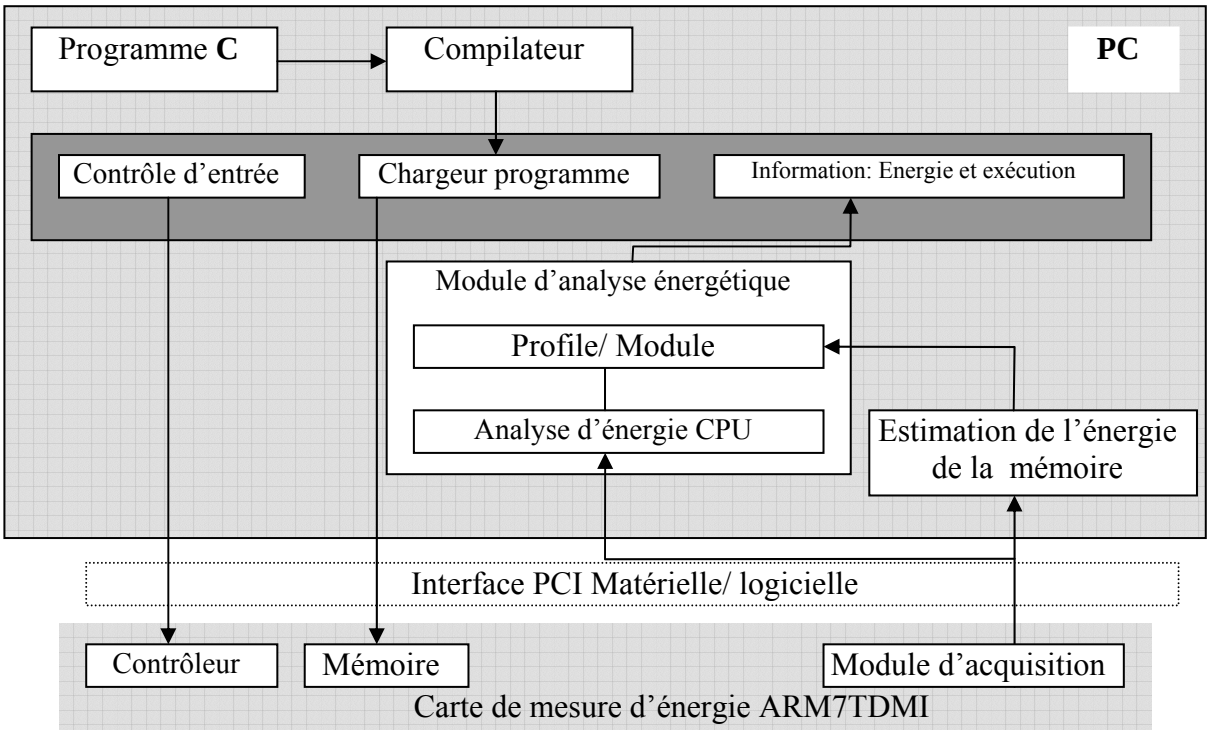


Figure 10 : *Architecture du SES*

I.7.3.1 Module d'estimation d'énergie

Ce module comporte une carte de mesure et un estimateur de consommation de la mémoire. La carte est connectée au port PCI exploitant un module d'acquisition du profil en

temps réel facilitant la collection de la trace précise du système au niveau cycle. Cette carte comporte un cœur de processeur ARM7TDMI avec son contrôleur, le module d'acquisition et la mémoire programme.

Le module d'acquisition comporte un circuit de mesure de l'énergie au niveau cycle, une mémoire d'acquisition et le contrôleur du profil. La carte de mesure d'énergie fonctionne comme un émulateur du processeur équipé d'un circuit de mesure de la consommation au niveau cycle. Les informations collectées comportent une trace énergétique du cœur du processeur ainsi que la trace de la mémoire au niveau cycle. Une fois la trace est collectée, elle est transférée via le bus PCI vers l'estimateur de consommation de la mémoire.

Ce dernier, fonctionnant sur la machine hôte, est un simulateur logiciel qui intègre des modèles de consommation au niveau cycle de divers types de caches, de bus et de mémoires. Il génère un profil général de la consommation.

I.7.3.2 Module d'analyse et interface utilisateur

Ce module lie le profil énergétique du processeur cible et la mémoire avec le code source à différents niveaux C/ASM. Grâce à l'interface graphique, l'utilisateur peut avoir recours à diverses options : compilation, spécification de la partie du code à profiler, ajout des *breakpoints,* et chargement du programme. (Figure 11)

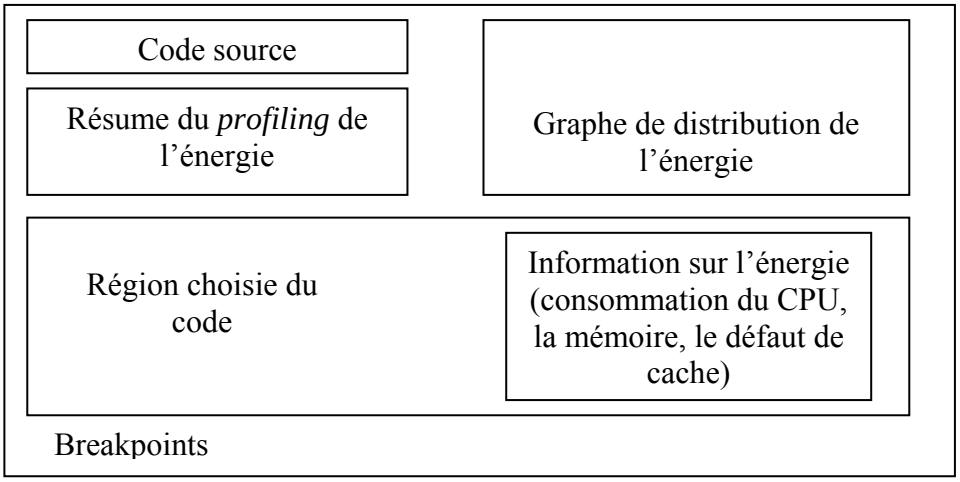


Figure 11 : *L'interface graphique de SES*

La limitation de cet outil est la dépendance envers la plateforme de mesures. En fait, pour tout type d'application, l'outil a recours à la carte d'acquisition d'infos sur la consommation. Ceci paraît un facteur limitatif pour son utilisation lors de la conception des applications.

I.7.4 ORINOCO

La méthodologie de conception au niveau système de ChipVision permet d'identifier les points chauds de l'application aussi tôt dans le processus de conception, spécifiquement au niveau ESL (Electronic System Level). Pour cela, l'outil ORINOCO (Stanley et al., 2004) est employé dans le flot de conception HW. Il est conçu spécifiquement pour les applications de traitement de données et de signal.

I.7.4.1 ORINOCO : Analyse de la puissance au niveau ESL

L'approche traditionnelle de la conception faible consommation est d'estimer et d'analyser la puissance au niveau RTL ou porte, et de modifier la conception en conséquence. Dans le meilleur cas, les blocs fonctionnels RTL sont modifiés, et re-synthétisés. Ce processus est répété jusqu'à ce que les résultats désirés soient réalisés. Toutefois, les réductions désirées en puissance peuvent souvent être réalisées en modifiant seulement l'architecture ou bien l'algorithme. Cependant, les modifications à ce niveau affectent non seulement la puissance, mais également d'autres métriques de performance ou bien le coût de la puce.

Avec ORINOCO, c'est possible d'optimiser la consommation au niveau système. La figure 12 montre le flot de conception et d'optimisation. La spécification du système est écrite à un niveau d'abstraction assez élevé (C/C++ ou SystemC). En partant de ces spécifications, des algorithmes réalisant la fonctionnalité du système sont développés et optimisés, généralement avec le même langage. La description algorithmique se compose d'une spécification exécutable ou d'une description fonctionnelle. L'architecture (mémoire, contrôleur et la structure de chemin de données) est générée pour implémenter les algorithmes. Lors de ce développement, diverses contraintes se présentent comme la puissance, la performance et la surface. Le problème qui se pose s'est qu'il y a peu d'outils structurés disponibles pour effectuer une telle analyse.

Au niveau ESL, ORINOCO choisi la plate-forme optimale selon cette méthodologie : Les divers algorithmes candidats sont analysés en terme de consommation et points chauds. Les algorithmes les plus prometteurs sont alors choisis et optimisés. Ceci est alors suivi par la création d'une architecture optimale. Les fonctions dont la consommation est optimale sont alors transformées en matériel. Ce processus d'estimation et optimisation de la consommation est itératif, et chaque itération nécessite des minutes ou des heures.

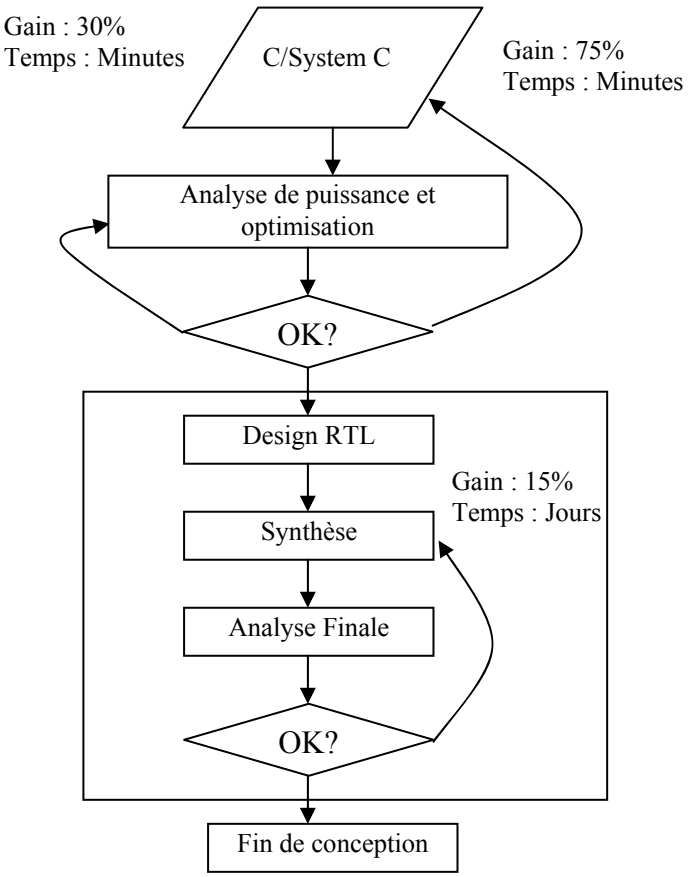


Figure 12: *flot de conception ESL*

I.7.4.2 Analyse d'algorithme et Optimisation

Avec ORINOCO, le « meilleur » algorithme est choisi parmi divers « candidats ». La puissance de chacun est identifiée selon la méthodologie représentée sur la figure 13. La spécification en C/SystemC est d'abord compilée puis profilée. Les informations ainsi générées sont réintégrées dans le code source. Les algorithmes sont alors exécutés, et les données résultantes du profil d'activité sont employées pour annoter une représentation appropriée de contrôle du flot de données (CDF).

Une architecture optimisée en consommation peut être tirée avec ORINOCO sans nécessité de synthèse complète grâce aux modèles de puissance créés pour chaque composant au niveau RTL. Ces modèles dépendent des données en entrée, des caractéristiques des composantes telles que le nombre de bits et l'architecture, et de la technologie des cellules. Ainsi, les modèles de puissance peuvent être générés automatiquement pour une technologie donnée. Et en utilisant l'activité des composants et les modèles de puissance, la puissance d'un composant peut être estimée.

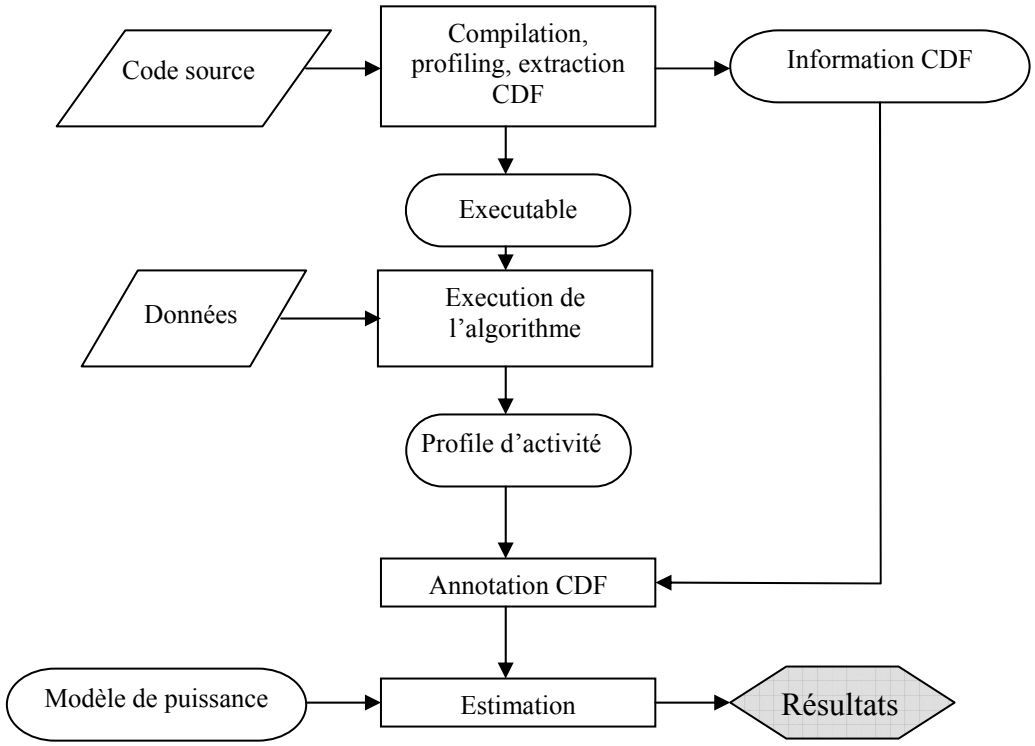


Figure 13: *Algorithme d'estimation de la puissance avec le flot de contrôle de donnée*

Par ailleurs, les mémoires sont utilisées dans ce cas pour le stockage intermédiaire de l'information et la communication inter-blocs. Ainsi, elles ont un effet significatif sur la consommation. Diverses techniques d'optimisation des accès aux mémoires incluant les transformations de boucles et le co-emplacement de code peuvent être affectées simplement en réécrivant le code. Toutefois, l'outil n'est pas conçu pour estimer la consommation des processeurs et des bus commerciaux, de plus il ne tient pas compte de l'interaction avec l'environnement externe.

I.7.5 EPRO

EPRO (Beak et al., 2004) est un outil fournissant des informations sur la performance et la consommation pour diverses applications réelles embarquées. Avec cet outil, des études de cas ont montré une réduction de 5,4 % sur la consommation de l'énergie et une amélioration de 4,4% au niveau performance. L'avantage majeur de ePRO est l'aptitude de faire le *co-profiling* (consommation et performance) en même temps.

I.7.5.1 Architecture générale

EPRO nécessite trois modules physiques : la cible embarquée, un multimètre numérique et la machine hôte. Cet outil est conçu pour le processeur PXA255 de la famille TynuxBox basée sur l'architecture Xscale. L'avantage de cette architecture est la présence d'une unité monitrice de performance (PMU) présente au niveau architectural. Ceci mène à bien contrôler divers comportements du système : efficacité de la mémoire cache (instruction, donnée), la latence de l'étage *fetch.*

Afin d'effectuer des profils matériels sur l'énergie, on doit avoir recours à un multimètre digital fonctionnant à des fréquences élevées. Ce dernier est contrôlé à la fois par la machine hôte et la carte cible afin de collecter le profile énergétique via des *triggers* E/S.

EPRO inclus 3 modules : le *profiler* d'énergie, le *profiler* de performance et l'interface graphique utilisateur (GUI). (Figure 14)

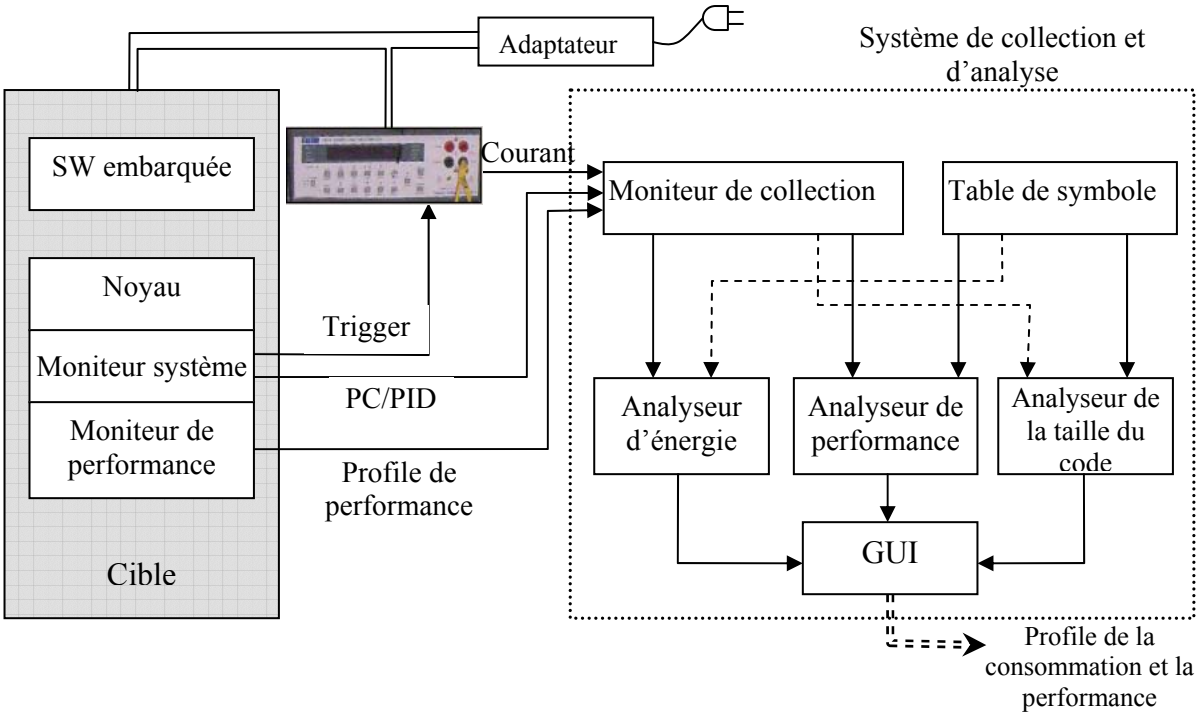


Figure 14: *Architecture de ePRO*

I.7.5.2 Le *profiler* d'énergie

Ce *profiler* d'énergie comporte un moniteur système, un multimètre et un analyseur d'énergie. La trace énergétique est faite en deux étapes :

- La collection de données : Durant cette phase, le moniteur système collecte périodiquement les valeurs échantillonnées du courant pour les envoyer à la machine hôte via

une interface Ethernet. En plus, il collecte les informations sur le système comme le compteur programme (PC), l'identificateur de processus (PID) à la même période.

- L'analyseur des données : Durant cette phase, l'analyseur énergétique analyse les valeurs échantillonnées du courant et les informations du système pour générer la trace énergétique. Et à l'aide d'une table de symbole générée par le compilateur croisé, la trace est *mappée* avec les diverses fonctions de l'application embarquée.

I.7.5.3 Le *profiler* de performance

L'analyseur de performance comporte un modificateur de code qui a comme entrée un code à un haut niveau d'abstraction comme le langage C, et a comme sortie un code modifié. Le PAC (Analyse de Performance de Code) est inséré à ce nouveau code.

Le code généré par le PAC est compilé et exécuté sur le processeur cible. Ceci permet de générer le profil de performance (efficacité de la cache d'instruction et de donnée, les requêtes sur le bus). (Figure 15)

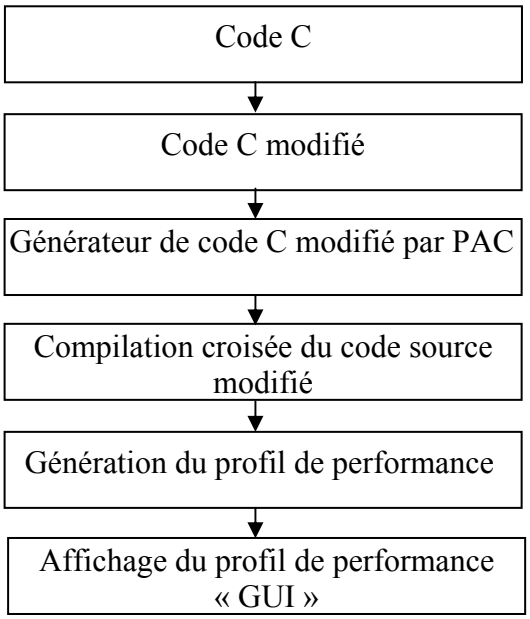


Figure 15: *Organigramme global de profil de performance*

I.7.5.4 L'interface utilisateur

Les traces de l'énergie et de performance sont présentées à travers une interface graphique facilitant l'identification des points chauds de l'application embarquée. En effet, les caractéristiques énergétiques de chaque fonction sont présentées à travers cette interface.

L'inconvénient majeur de cet outil est son recours à la plateforme pour le *profiling*. Ce qui impose sa présence dans tous les tests.

I.7.6 DSP-PP

Il a été largement accepté que la simulation, au niveau porte et circuit, est infaisable pour évaluer la consommation d'une exécution logicielle pour des systèmes de calcul complexes. Pour cela, un jeu complémentaire d'approches basé sur l'utilisation de simulateurs architecturaux de consommation au niveau cycle précis est apparu. Ces simulations peuvent être applicables aux processeurs modernes super scalaire (avec les pipelines assez profonds). (Li et al., 2003).

On peut citer le DSP-PP (Minh et al., 2003) qui est un outil de simulation RTL permettant l'estimation de la puissance dissipée pour les DSPs. Il est écrit en C++ afin de profiter de ce haut niveau d'abstraction. (Figure 16)

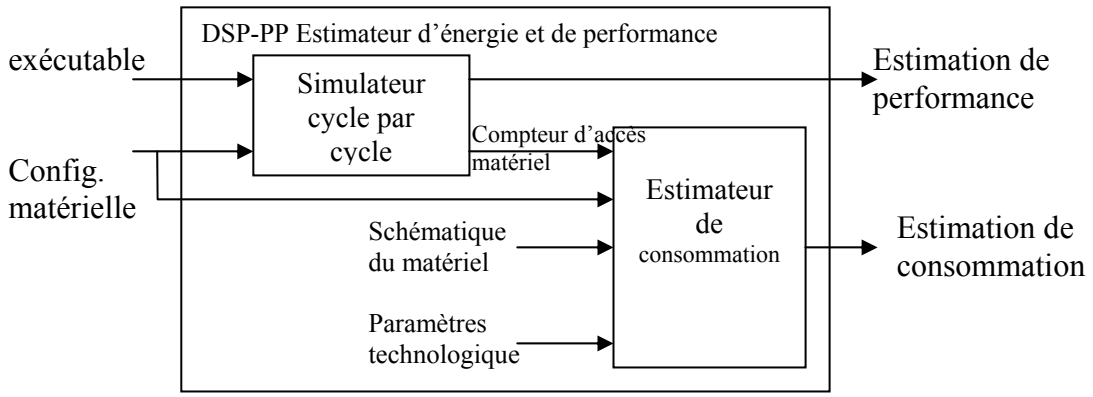


Figure 16: *Diagramme de l'estimateur DSP-PP*

DSP-PP emploie la simulation détaillée au niveau cycle de tous les composants du DSP: les chemins de données et l'interconnexion et estime exactement la valeur de puissance dynamique, de court circuit et de fuite de chaque composant du DSP. Les composants du DSP sont modélisés comme des objets intégrants le modèle de consommation.

Le DSP-PP est composé de deux composants: le simulateur de performance au niveau cycle (CPS) et l'estimateur de dissipation de puissance (PDE).

- Le CPS est un simulateur "piloté par l'exécution" il accepte comme entrée l'exécutable obtenu par compilation et la configuration architecturale du DSP. Il simule cycle par cycle l'exécution de l'instruction ainsi que les données. Il génère comme sortie des statistiques sur la performance, et le nombre d'accès matériels cycle par cycle.

- Le PDE est constitué de modèles de consommation des différents composants. Il accepte comme entrée le nombre d'accès matériels du CPS et la configuration de l'architecture du DSP afin de générer une estimation de la puissance

Cependant, la simulation au niveau cycle-précis entraîne une vitesse de simulation extrêmement lente, empêchant l'efficacité de la recherche d'espace de conception. C'est particulièrement vrai en simulant des applications complexes employant des modèles de processeur détaillés. À cause de cela, la simulation basée sur le modèle de puissance ne peut pas être employée pour l'évaluation de puissance de logiciel pendant l'exécution.

I.8 Interprétation

De nombreux travaux se sont focalisés sur la modélisation de la consommation au niveau instruction, où la consommation du code est obtenue en estimant celle de chaque instruction du code. Ces méthodes sont généralement inefficaces pour les architectures complexes ayant diverses unités de traitement communicantes. De plus, certains outils ne ciblent pas les DSP commerciaux traités dans ce travail (C62, C55 et le C67). Par ailleurs, les gains importants en terme de consommation sont réalisés aux hauts niveaux d'abstraction où les décisions d'implantations logicielles et/ou matérielles sont faites. Ceci permet de compenser la complexité croissante des applications et d'intégrer la consommation au début du flot de conception mixte. Par ailleurs, afin de pouvoir dimensionner le système dés le début du flot de conception, des modèles de performances temporelles et énergétiques sont nécessaires afin de caractériser l'influence des paramètres de l'application et de l'architecture sur la consommation.

I.9 Conclusion

Dans ce chapitre, on a présenté les diverses techniques et outils d'estimation et d'optimisation de la consommation. Grâce à ces outils, le concepteur a la possibilité d'exploiter plus de ressources 'faible consommation' et d'offrir les performances maximales aux utilisateurs. Par ailleurs, les décisions de conception les plus efficaces dérivent du choix et de l'optimisation des architectures et des algorithmes aux niveaux les plus hauts. Et afin de mieux guider le concepteur lors de la conception des architectures complexes, il est nécessaire d'adopter une méthodologie d'exploration de l'espace des solutions possibles qui soit plus globale (toute l'architecture). Cette méthodologie permettra d'avoir une solution optimale qui respecte les diverses contraintes de l'application.

Chapitre

II

Exploration de l'espace des solutions

Chapitre II. Exploration de l'espace des solutions

II.1 Introduction

Les futurs systèmes mobiles ne cessent d'évoluer en intégrant de plus en plus de nouvelles fonctionnalités. A titre d'exemple, les téléphones portables actuels intègrent de la musique, la vidéo, les jeux, le GPS, la capture d'image, l'accès à Internet, le stockage de données, etc. Et ceci, tout en gardant une bonne autonomie et une masse ne dépassant pas les 150 grammes. Cette multitude de fonctionnalités favorise généralement des architectures multiprocesseurs ayant des performances élevées. Pour garantir la faisabilité de ces systèmes, il faut donc prospecter de nouvelles solutions architecturales logicielles et matérielles garantissant une performance élevée, une grande flexibilité et une faible consommation (Figure 17). Seule une approche globale permettra de caractériser et d'optimiser efficacement ces systèmes. (Maalej, 2007)

De plus, l'optimisation d'un système ne s'effectue pas seulement au niveau de la conception de ces composants, mais également au niveau du choix d'une architecture générale. Le choix d'une technologie par rapport à une autre et le choix des unités de traitement, peuvent devenir des problèmes critiques lorsque la gestion de la consommation représente un critère essentiel. Ainsi au lieu de rechercher un périphérique, entre autre une architecture, plus performante, il peut parfois être préférable de changer complètement de technologie.

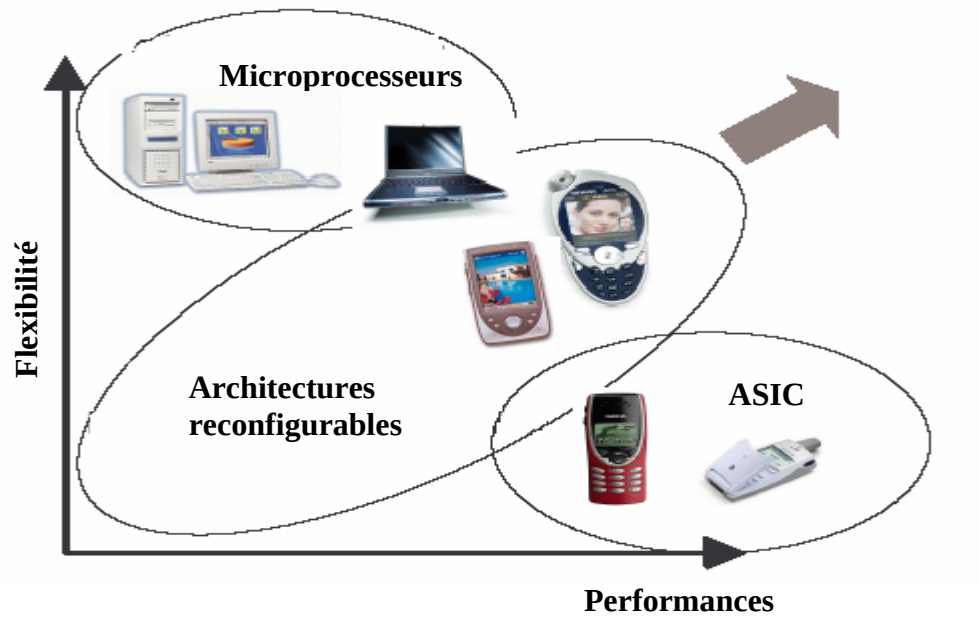


Figure 17 : *Flexibilité des architectures vis à vis de leurs performances* (Benoit et al., 2004)

Dans ce chapitre, on présente les méthodes d'exploration de l'espace des solutions, le flot de conception général des systèmes embarqués en terme d'outils, de métriques et de modèles d'exploration architecturale (Hw/Sw). Ce chapitre s'intéresse aussi aux algorithmes d'exploration de l'espace des solutions.

II.2 Conception mixte

L'exploration de l'espace de conception est entreprise à partir des descriptions de haut niveau tel que le C/VHDL en tenant compte éventuellement d'une architecture cible. Cela permet de considérer un compromis des réalisations logicielles/matérielles pour satisfaire les performances et les contraintes à travers les bons choix de l'architecture logicielle/matérielle.

Par ailleurs, l'exploration de l'espace des solutions est parmi l'une des étapes nécessaires lors de la conception des systèmes embarqués. Elle permet de surmonter le problème de la complexité de l'espace afin d'atteindre la solution adéquate rapidement. Il est à signaler que le choix de la meilleure solution à un haut niveau d'abstraction n'est pas assez simple vu le nombre de combinaisons architecturales possibles. Par ailleurs, la complexité de l'exploration est liée à la complexité de l'application. En effet, pour une application contenant n tâches fonctionnant sur une architecture monoprocesseur, le nombre de solutions possibles U est établi par la loi suivante (Bagdadi et al.,2002):

$$U_n = \sum_{q=1}^{n} \sum_{i=0}^{q} \frac{(-1)^{q-i} i^n}{(q-i)!i!} \qquad (5)$$

Pour le cas d'une architecture multiprocesseur (p processeurs), le problème se complique plus. Le nombre de solutions possibles sera :

$$U_n^p = \sum_{q=1}^{n} p^q \sum_{i=0}^{q} \frac{(-1)^{q-i} i^n}{(q-i)!i!} \qquad (6)$$

Prenons le cas d'un problème composé de 10 tâches fonctionnant sur 3 processeurs, l'espace des solutions possibles dépasse dans ce cas 10^7 combinaisons possibles. Le concepteur d'un tel système est incapable généralement de gérer et d'évaluer cet ensemble de solutions ni manuellement ni d'une manière exacte.

Il est à signaler que la problématique d'exploration de l'espace des solutions ne se limite pas à l'étude des combinaisons possibles (ordonnancement et partitionnement). En fait, les systèmes embarqués sont souvent soumis à diverses contraintes. Le temps réel et la

consommation sont parmi les exigences de ces systèmes vu qu'ils sont toujours en interaction avec leur environnement extérieur et qu'ils nécessitent beaucoup de calcul.

II.3 Éléments caractéristiques des flots de codesign

L'exploration de l'espace des solutions basse consommation nécessite un certain nombre d'informations relatives à l'application d'une part et aux modèles de performances d'autre part. Par ailleurs, un besoin de modèles d'estimation et de performance suffisamment riches et paramétrables s'impose lors de l'exploration. Dans ce qui suit, on présente les points clés de cette dernière:

- La spécification : elle est un point important du flot de conception dont le choix peut avoir un impact sur les résultats. Il existe une variété de modèles et de langages utilisés (Lustre, SDL, CDFG, System-C, etc). Ils permettent d'exprimer des notions telles que : la concurrence, la hiérarchie, les communications, la synchronisation, le temps, etc.

- La simulation : elle se retrouve à plusieurs niveaux d'abstraction dans le flot de conception. Le niveau de détails est plus important au niveau d'abstraction le plus bas et il diminue au fur et à mesure que l'on remonte vers le niveau système. Plus le niveau de détails est élevé plus les temps de simulation sont longs. Au niveau système, la simulation est de type fonctionnel, elle permet de vérifier que le système est fonctionnellement correct sans se préoccuper des détails d'implantations.

- L'architecture : Un flot de codesign est souvent orienté vers un type d'architecture. La modélisation des architectures permet d'évaluer les performances et les coûts d'implémentation de la spécification et ainsi de guider les choix de partitionnement. Les premières méthodes de codesign considéraient des modèles d'architecture simples composés d'un processeur et d'un accélérateur matériel dédié (ASIC). D'autres méthodes permettent de cibler des architectures hétérogènes composées de plusieurs types de processeurs et d'accélérateurs ainsi que des hiérarchies de mémoires (Bianco *et al.,* 1998), (Auguin *et al*., 2001), (Marteil et al,.2006). Enfin, il existe des méthodes permettant de gérer les architectures reconfigurables (souvent à base de FPGA), (Li *et al*., 2000), (Bossuet *et al*., 2003) (Elleouet et al., 2006).

- Les métriques : Les étapes du processus d'exploration et plus particulièrement le partitionnement, peuvent être guidées par des métriques caractérisant l'application. Ces métriques permettent de guider le concepteur et les outils quant aux choix de l'architecture et du partitionnement. Dans (Sciuto *et al*., 2002), des métriques relativement fines sont définies

pour caractériser l'affinité entre les fonctions d'une application et trois types de cibles architecturales : processeur à usage général (GPP), processeur de signal numérique (DSP) et ASIC. Les métriques, qui résultent de l'analyse du code 'C' de l'application, permettent de repérer les séquences d'instructions qui peuvent être soit orientées DSP, orientées ASIC (instructions de niveau bit,...) ou orientées GPP (structure de test, ratio d'instructions d'entrées/sorties,...). Ces métriques servent ensuite pour guider l'outil de partitionnement logiciel/matériel.

- Partitionnement : que celle-ci soit manuelle ou automatique, son but est de répartir les "fonctions" de l'application sur les parties logicielles et matérielles de l'architecture cible. Ce processus est réitéré jusqu'à ce qu'une solution ou un ensemble de solutions satisfaisantes ait été trouvé.

- Exploration de l'espace de solutions : Le résultat de l'exploration de l'espace de conception peut prendre la forme, soit d'une solution unique, soit d'un ensemble de solutions. Les méthodes qui fournissent à l'utilisateur une solution unique à partir d'un sous-ensemble restreint de possibilités, visent le plus souvent à trouver la solution optimale au problème alors que les méthodes qui fournissent un ensemble de solutions visent en général à trouver des solutions qui respectent l'ensemble des contraintes sans forcément être optimales. En effet, à un niveau d'abstraction élevé, la précision n'est pas assez suffisante pour garantir avec certitude que la solution soit optimale (car trop de détails d'implantation sont inconnus). Il est ainsi plus judicieux de conserver un ensemble de solutions "prometteuses" qui seront ensuite estimées à un niveau inférieur, conduisant à un ensemble de solutions plus réduit. La réitération de ce principe permet ainsi de converger vers une solution unique.

Dans cette partie, les étapes essentielles du flot de conception mixte typique sont présentées. Dans la section suivante, quelques outils de codesign seront présentés.

II.4 Méthodologies et outils

II.4.1 Introduction

Bien que les méthodes de réduction de puissance et d'énergie soient plus efficaces une fois adressées le plutôt possible dans le processus de conception (globalement au niveau système); la majorité des travaux existants sur l'optimisation de puissance adressent séparément les parties logicielles, matérielles et de communications après avoir décidé de l'architecture cible du système. Quelques approches de co-design tiennent compte d'un tel but à un niveau d'abstraction plus élevé. Ces approches commencent en général par une étape

d'estimation de la consommation des parties du système (tâches, fonctions, communications etc.) pour déterminer ensuite, et le plutôt possible, la consommation totale du système.

II.4.2 Outils de codesign

Le tableau 4 présente quelques environnements de codesign développés soit par les groupes de recherche universitaires, soit dans l'industrie

Tableau 4 : *Comparatif des environnements de codesign (Abdennour, 2004)*

	Type d'application	**Langage de spécification**	**Approche de conception**	**Architecture cible**
Ptolemy	TSI	Flot de donnée synchrone	Découpage automatique orienté logiciel	Multi-processeur + ASIC
Vulcan	Systèmes temps réel	HardwareC (extension de C)	Découpage automatique orienté matériel (contrainte de temps et coût)	Mono-processeur, ASIC, bus et mémoire
Cosyma	systèmes temps réel embarqué	Cx (extension de C)	Découpage automatique orienté logiciel (contraintes de temps+ coût en surface)	processeur+ co-processeur ou multi-processeur et Mémoire partagée pour les Coms
SpecSyn	Système de contrôle et de communication	SpecCharts	-pré-estimation - découpage automatique ou manuel - raffinement - implantation	Multi-processeur processseurs+ ASICs+ASIP + et Bus, mémoire pour lesComs
Polis	systèmes temps réel embarquées de contrôle	CFSMs	- découpage manuel, -ordonnancement, - implantation	Multi-processeur microcontrôleurs avec des RTOS + ASICs et Ports d'E/S (Automatiquement générés) pour les coms
Cosyn	Système temps réel embarqués	Graphe de tâche s	- Estimation -Découpage automatique contraintes de temps + coût en surface et en consommation	Multi-processeur processeurs+ ASICs+FPGAs et point-à-point, bus, réseaux de communication locale LAN) pour les coms
CODES	Les systèmes de communication	RDP, StateCharts	- modélisation -partitionnement - simulation - intégration	Multiprocesseur Processeurs + mémoire + ASICs + FPGA
CoWare	Traitement de signal et système de communication	C (SystemC) HDL (VHDL, Verilog)	- découpage manuel - synthèse d'interface, - ordonnancement, - cosimulation	Multiprocesseur Processeurs + DSPs+ ASICs et des coms Point-à-point avec un protocole de rendez-vous

Dans ce tableau les aspects de comparaison sont :

- Les types d'applications ciblées,
- Le langage de spécification du système,
- L'approche de conception,
- L'architecture cible,
- Les types des communications logicielle/matérielle utilisées.

Dans la section suivante, on va présenter en détails d'autres outils d'exploration basse consommation ainsi que les limitations de ces environnements.

II.4.3 L'environnement MOVE

Cet outil permet de réaliser *automatiquement* un ASIP pour le traitement d'image à partir d'une spécification haut niveau (Heikkinen *et al*., 2002). L'architecture cible est la TTA (Transport Triggered Architecture). Son principe est semblable à l'architecture VLIW : elle permet d'effectuer plusieurs opérations en un seul cycle d'horloge en utilisant un réseau de transport (9 bus) de données, couplé à des unités fonctionnelles (multiplieurs, accumulateurs, registres…). Les avantages de cette technique sont la flexibilité, la simplicité et la facilité d'extension du système. La conception se base sur l'utilisation d'une plateforme (appelée MOVE) pour la conception automatique de processeurs, formée par un système de développement HW/SW et un optimiseur. (Figure 18).

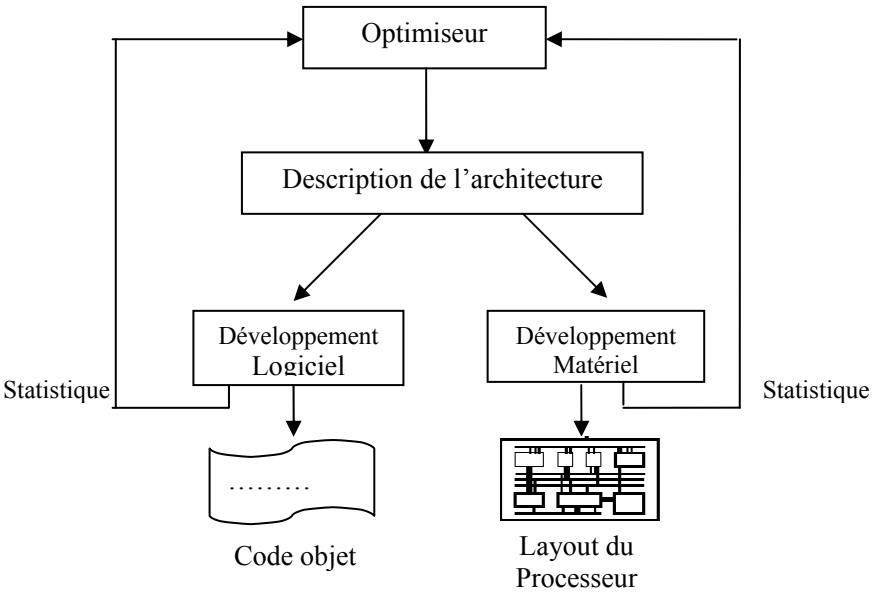


Figure 18 : *L'architecture MOVE*

Le système explore l'espace de solutions (combinaison de plusieurs choix d'architecture) pour trouver la meilleure solution selon le critère rapport coût/performance. Le système permet ainsi d'ordonnancer l'application envisagée sur plusieurs architectures. L'aspect faible consommation n'est pas traité clairement dans cette approche.

II.4.4 L'outil Codef-LP

Codef-LP (figure 19) (Guitton et al., 2003) développé à l'université de Nice propose une méthodologie permettant d'extraire une architecture monoprocesseur qui satisfait les contraintes temps, surface et qui minimise les pics de courants. La description de l'application en graphe de tâche ainsi que les contraintes et la librairie de modèles sont l'entrée de l'outil Codef. Ce dernier génère un *mapping* de l'application en utilisant une heuristique d'exploration basée sur le glouton. Afin d'optimiser davantage la consommation, des raffinements manuels en variant la fréquence sont faits suite à l'exploration automatique. L'estimation de la consommation des modules matériels est faite avec l'outil WattWatcher qui nécessite une synthèse. Alors que celle en logicielle est faite avec Vestim qui est basé sur la méthode d'estimation au niveau assembleur ce qui nécessite la compilation du code C. Par ailleurs, le modèle de consommation considère que la consommation suit la loi $P=KCFreqV^2$. L'exploration dans cette méthodologie se limite au changement manuel de la fréquence/Vcc ou au changement du processeur.

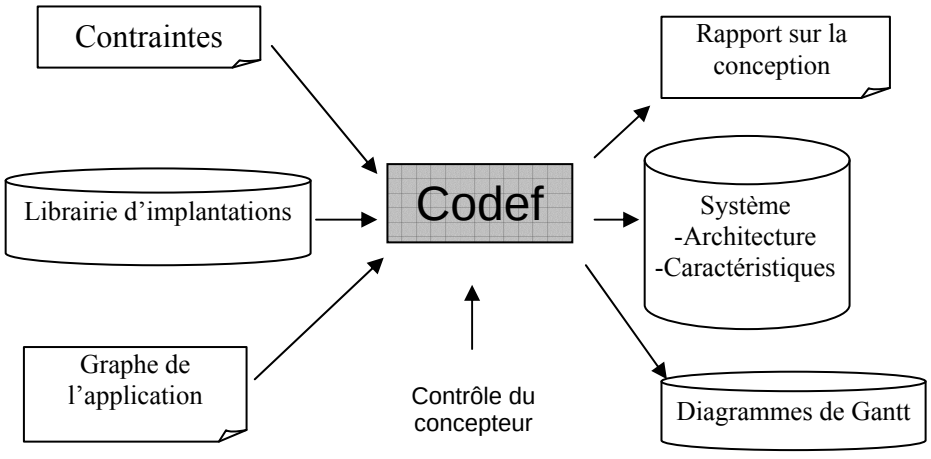


Figure 19 : *Description de l'outil Codef*

Toutefois, cet outil n'est pas disponible, il traite des architectures monoprocesseur, il ne tient pas compte de la communication. De plus, il se base sur une méthode au niveau

assembleur lors de l'estimation de la consommation de la partie logicielle, ce qui est compliqué pour les architectures VLIW. Par ailleurs, l'heuristique du glouton adoptée pour l'exploration n'est pas assez efficace vu qu'elle construit la solution pas à pas sans revenir sur ses décisions de choix. En plus, l'outil repose sur une librairie de modèles non paramétrable. En effet, une tâche peut avoir plusieurs modèles de performances selon les paramètres algorithmiques et/ou architecturales (taille de l'image, fréquence, cadence, etc.).

II.4.5 L'outil Mogac

Mogac (Dick et al., 1998) développé à l'université de Princeton est un outil de co-synthèse logicielle/ matérielle. Il *partitionne* et ordonnance la spécification de l'application décrite en graphe de tâche périodique. Pour cela l'outil exploite un algorithme génétique multiobjectif qui échappe des minimums locaux. Le coût et la consommation sont optimisés sous des contraintes de temps réel. Par ailleurs, l'outil tient compte de la consommation de la communication. Par contre, l'outil ne tient compte ni de l'influence de la tension d'alimentation ni de la fréquence. Par ailleurs, il ne traite ni la surface ni la consommation des mémoires. (Figure 20)

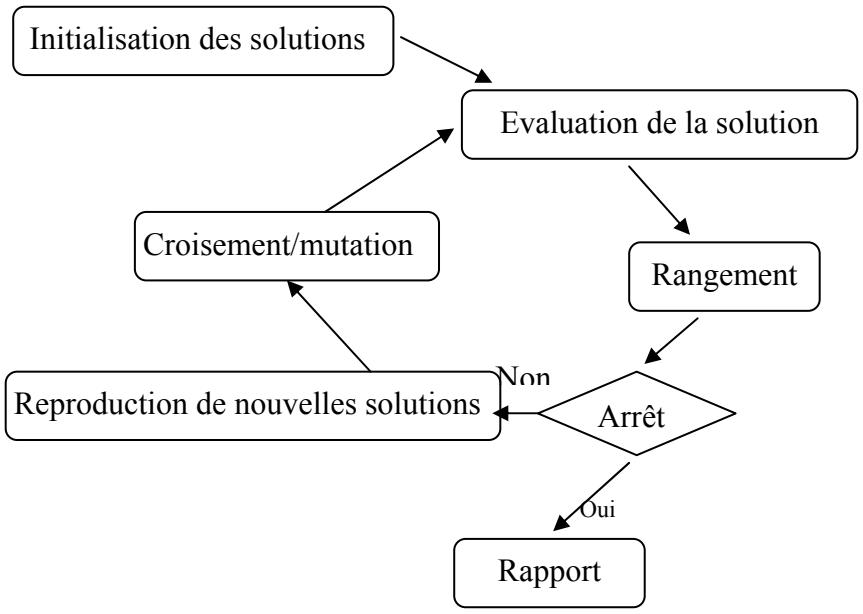


Figure 20 : *Description de l'outil Mogac*

Avec l'outil Mogac (Figure20), une exploration multi-objective est faite en terme de coût et de consommation. Les solutions pareto-optimales sont extraites par l'outil grâce à l'algorithme génétique. Toutefois, cet outil, conçu au départ pour optimiser essentiellement le

coût, n'est pas disponible. Aucune indication sur les méthodes d'estimation des performances n'est indiquée ni au niveau tâche ni au niveau système.

II.4.6 L'outil Cosyn-LP

L'outil Cosyn-LP (Bharat et al., 1999) développé à l'université de Princeton est un outil de co-synthèse qui part d'une spécification en graphe de tâche périodique avec des contraintes de temps réel afin de générer une architecture à faible coût qui respecte les contraintes. Ces contraintes ainsi qu'une bibliothèque de modèles sont fournies à l'outil. L'approche se base sur une combinaison d'ordonnancement préemptif et non préemptif afin d'ordonnancer les tâches. La technique d'exploration se base sur le regroupement de tâches (Clustering). (Figure 21)

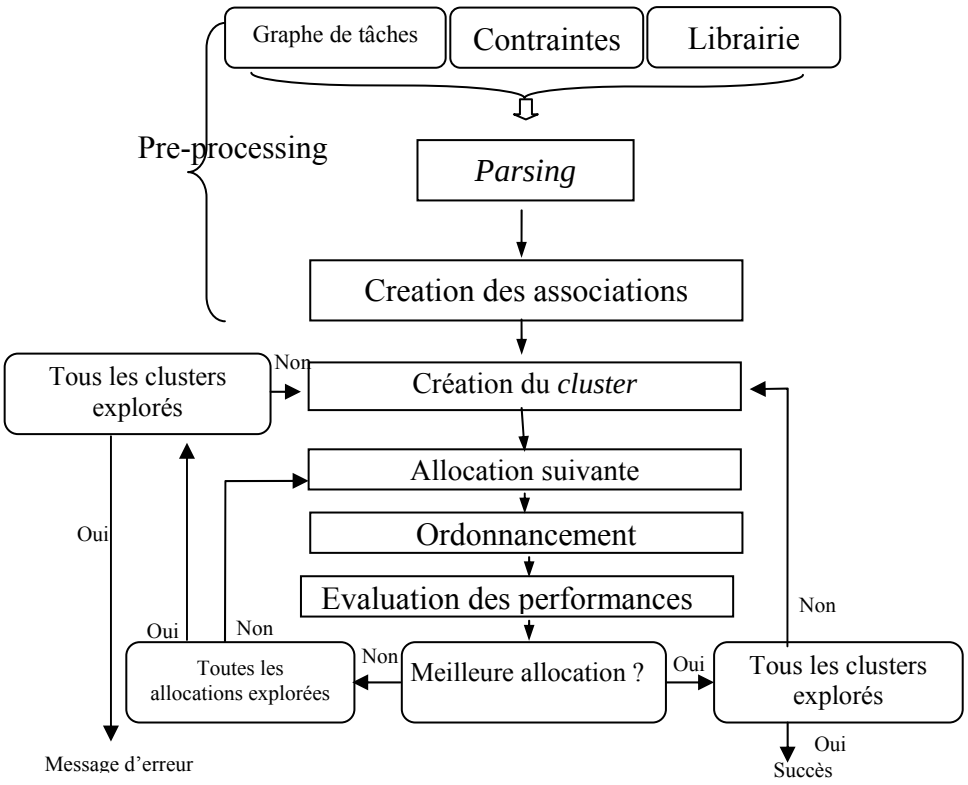


Figure 21 : *Description de l'outil Cosyn-LP*

L'étude exploratrice est faite sur des processeurs de type Motorola 68360, 68040, des ASICs et un FPGA Xilinx 3195. Vu que la technique est basée sur le clustering, tous les clusters et les allocations sont considérés lors de l'exploration. Ceci rend la méthode exacte et par conséquent assez complexe. Par ailleurs, le concepteur de cet outil suppose que les modèles de performance sont déjà prêts et non paramétrables. Donc pour chaque tâche s'exécutant sur une cible donnée, lui correspond une performance temporelle et énergétique

unique. En plus, la granularité des modèles de performances de l'application n'est pas considérée dans cette approche.

II.4.7 Méthodologie de Ghali

Cette méthodologie d'exploration développée à l'université Paris XI Orsay (Ghali et al., 2004), repose sur l'évaluation du temps, de la surface et de la consommation d'une architecture monoprocesseur. L'évaluation du temps d'exécution est déduite par simulation grâce à l'outil SimpleScalar ou par exécution directe. La surface silicium est évaluée avec les deux outils CACTI et FUPA pour estimer la surface du cache d'instruction et la surface des unités de calcul flottant respectivement. Les outils SimplePower et Xpower sont exploités pour évaluer la consommation des architectures superscalar et des FPGA respectivement. Il est à noter que cette méthodologie repose sur l'exploration avec les algorithmes génétiques multi objectives NSGA II (Figure 22). Et afin d'extraire les solutions les plus prometteuses, le concepteur a eu recours à 100 stations de travail pour émuler les diverses solutions.

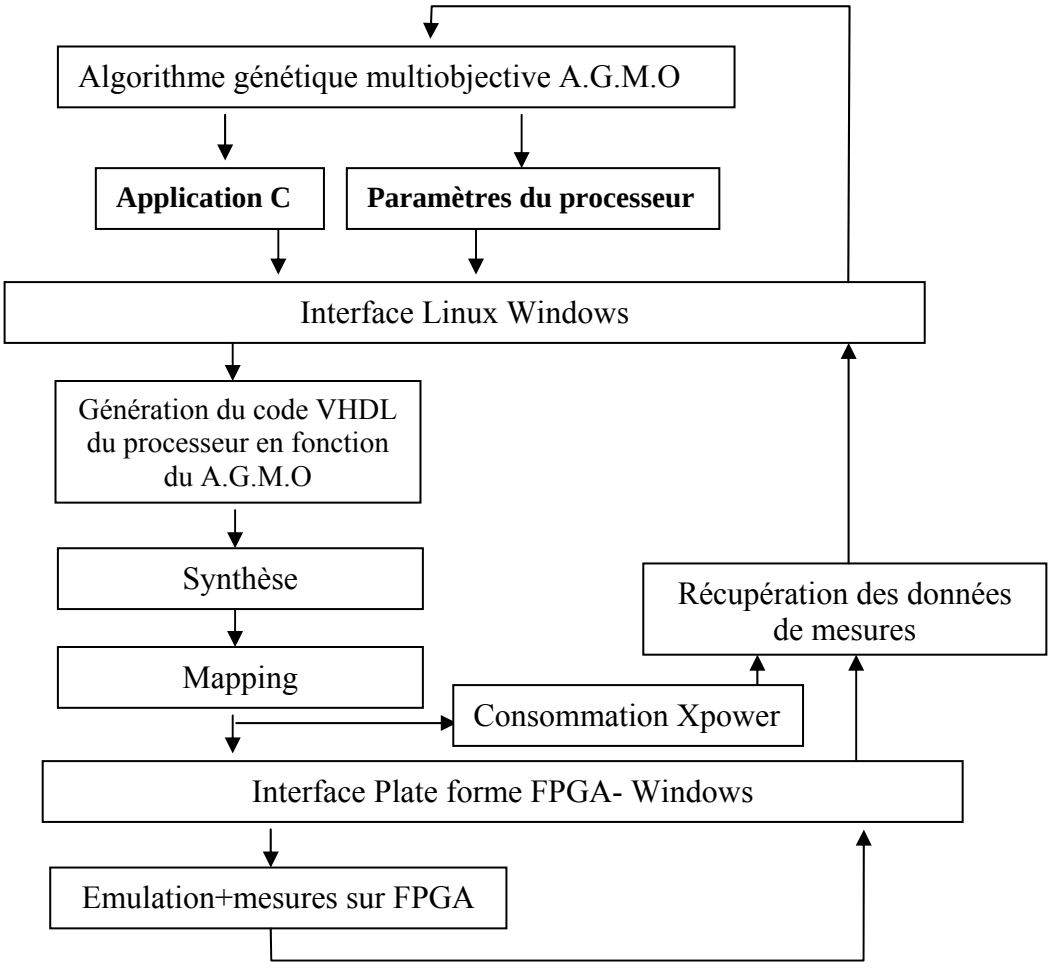


Figure 22 : *Description de la méthodologie*

A travers cette méthodologie, la possibilité de tailler le processeur selon les besoins et les contraintes se présente. En fait, l'algorithme génétique permet de modifier les caractéristiques de la cible (taille de la cache de données et d'instructions, le nombre de ALU, de MULT, de registre RUU, de pipeline) pour étudier leurs influences sur les performances. Aucune considération des paramètres algorithmiques de l'application n'est faite. Par ailleurs, cette exploration se limite au niveau configuration architecturale d'un processeur SuperScalaire générique ainsi que la configuration d'un processeur particulier « Leon ».

II.5 Discussion

En se basant sur ces travaux ci dessus, diverses constatations sont établies :

1. **Disponibilité :** les outils déjà présentés ne sont pas généralement disponibles pour être exploités. Ce qui nécessite le développement d'un environnement d'exploration basse consommation qui intègre les modèles de performances souhaités afin d'extraire la solution qui répond aux besoins. Ainsi l'outil peut être étendu et enrichi avec d'autres modèles selon les besoins.
2. **Architecture cible** : la plus part de ces approches explorent une architecture prédéfinie ou monoprocesseur, c'est le cas de l'outil Codef et la méthodologie de ghali. Par ailleurs avec l'outil Codef-LP, on peut explorer l'espace des solutions mais sans tenir compte de la consommation de la communication qui peut être significative. Codef-LP se limite lors de l'optimisation de la consommation à la partie logicielle en variant la tension et la fréquence du processeur sans toucher explicitement la partie matérielle.
3. **Modèles de performances** : pour une architecture multiprocesseur, 2 outils existent pour explorer l'espace des solutions (Cosyn & Mogac). Chaque outil gère les lois de consommation en se basant sur des outils ou sur des modèles prêts non paramétrables. Concernant l'intégration d'un modèle de consommation de la communication dans le modèle général, la question qui se pose : est-ce qu'elle est significative par rapport à la consommation de la plate-forme. Si la consommation de la communication est de l'ordre de quelques µW par rapport à la consommation des DSP (quelques mW), serait t'il intéressant d'y tenir compte. Par ailleurs, il est à signaler que les paramètres de l'application et de l'architecture influent sur la performance du système.

Ainsi, il paraît important d'étudier et de mettre en place une méthodologie d'exploration d'architecture multiprocesseur qui tient compte des diverses contraintes entre autre la consommation à partir d'une description au niveau système. Le travail consiste donc à :

- **Niveaux de granularité** : il s'agit de spécifier l'application et les contraintes à plusieurs niveaux de granularités. Cela permet de donner la possibilité d'explorer l'espace des solutions plus efficacement.

- **Modèles paramétriques de consommation** : il s'agit d'établir des modèles paramétriques qui englobent la consommation de tout le système logiciel/matériel/communication. En fait, les paramètres de l'application ainsi que ceux de l'architecture seront considérés dans les modèles de performance afin d'avoir des modèles assez riche.

- **Exploration efficace d'architecture basse consommation :** il s'agit d'être capable de choisir d'une façon efficace la solution architecturale adéquate où le nombre de ressources à exploiter n'est pas connu à priori. En effet, le concepteur peut être confronté au problème du choix du nombre de ressources : est ce qu'il implémente son application sur 2 ou 3 DSPs par exemple.

II.6 Conclusion

Dans cette étude, on a présenté l'impact du choix aux différents niveaux d'abstraction sur la consommation (Système, algorithme, RTL, logique, physique). Étant donné que la consommation a un fort impact sur la conception des systèmes embarqués, un intérêt est accordé pour limiter cette dissipation à travers des méthodologies de réduction de la consommation et des outils mis en œuvre. Ces divers outils d'estimation et d'optimisation de la consommation sont évalués dans ce chapitre. Par ailleurs, afin de répondre à l'ensemble des contraintes de plus en plus pressantes, de nouvelles méthodes de conception doivent être utilisées. Ces méthodes doivent permettre l'adéquation entre l'application et son architecture cible pour bien exploiter les caractéristiques de l'application et garantir une bonne performance du système, ainsi que l'adaptation à l'environnement. Le problème de l'exploration de l'espace de conception logiciel/matériel a été étudié. Cette étude a permis de dégager les caractéristiques et les mécanismes nécessaires afin de formuler une approche globale de l'exploration de l'espace des solutions.

Dans le chapitre suivant, on présente la nouvelle méthodologie d'exploration basse consommation ainsi que l'environnement paramétrique développé.

Chapitre

III

Approche et Méthodologie d'exploration

Chapitre III. Approche et Méthodologie d'exploration

III.1 Introduction

L'objectif de ce chapitre est de présenter la méthodologie et l'approche d'exploration basse consommation. Des modèles de performances riches et paramétriques ainsi qu'une technique d'exploration dite basse consommation sont proposés. Cette approche permet de considérer un certain nombre de paramètres algorithmiques et architecturaux sur la consommation. Un modèle complet est proposé afin de déduire les performances globales du système qui seront utilisées lors de l'exploration à travers une technique basée sur le recuit simulé. Cette heuristique permet d'exploiter la technique d'exploration selon plusieurs niveaux de granularité et ce afin de pouvoir choisir le niveau qui permet d'assurer une exploration précise et rapide.

III.2 Modèles de performances et technique d'exploration

L'exploration de l'espace des solutions basse consommation nécessite un certain nombre d'informations relatives à l'application d'une part, et des modèles de performances d'autre part. En fait, il s'agit de déduire les informations nécessaires sur l'application pour bien mener la phase d'exploration. Par ailleurs, on a besoin de modèles d'estimation et de performance suffisamment riches et paramétrables.

Cette section traite successivement le modèle du graphe et de l'architecture, la méthodologie d'obtention des modèles de performances temporels et énergétiques ainsi que le modèle du coût. On présente aussi la méthode d'estimation et la technique d'exploration de l'espace de solution adoptée.

III.2.1 Modèle de graphe

Le modèle de spécification doit permettre de décrire le fonctionnement de toute l'application tout en étant indépendant de son implémentation finale. Le modèle de spécification est utilisé par le concepteur pour décomposer le système en un ensemble de sous systèmes (le modèle du graphe de tâche décrit le système par des tâches). Ensuite, chaque sous système peut être décrit par le concepteur par un langage de spécification. Les sous systèmes résultant d'un modèle de spécification peuvent être décrits par plusieurs langages de spécification, c'est le cas des spécifications hétérogènes. Concernant la spécification de l'application, elle est souvent représentée à base de graphe de tâche (Guitton et al., 2003)

(kappagantula et al., 2003) (tmar et al., 2007) (Abdennour et al., 2002). Cette représentation permet de modéliser les tâches ainsi que les dépendances inter-tâches de l'application qui sont nécessaires lors de l'ordonnancement et l'estimation des performances. De telles informations sont généralement définies par le concepteur à partir du cahier de charge de l'application afin de développer l'application, ou bien fournies dans le datasheet de l'application.

Dans l'approche proposée, on part d'une spécification de l'application sous forme de graphe de tâches acyclique orienté (DAG) constitué par les tâches Ti du système (les nœuds du graphe) et les dépendances entre elles (les arcs). On associe à chaque arc Aij du graphe, la quantité de données en octets que la tâche Ti doit transférer à la tâche Tj. (Figure23)

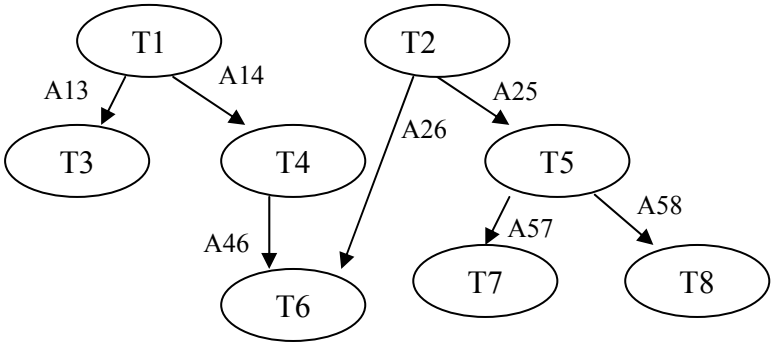


Figure 23 : *Graphe de tâches*

Avec ces informations présentent dans le graphe, les dépendances entre les tâches sont considérées. Ceci servira à l'ordonnancement des tâches et à l'extraction du temps d'exécution total de l'application. Par ailleurs, pour chaque tâche du graphe, des modèles d'estimation sont associés. Ces modèles paramétrables, présentés sous forme de valeurs de performances temporelles et énergétiques, sont attribués à chaque tâche de l'application.

III.2.2 Modèle d'architecture

- L'architecture cible sera une architecture hétérogène (majoritairement du Soft et du Hard) sous forme de composants discrets (DSPs & FPGA) communiquant via un bus et possédant une mémoire commune (Figure 24). Le nombre d'unité de traitement entre autre les DSPs sera paramétrable. Ainsi, le concepteur ne sera pas dans l'obligation de travailler sur une architecture prédéfinie et figée. C'est à l'approche de choisir la solution architecturale adéquate en fixant le nombre de ressources utiles ainsi que le mapping des tâches.

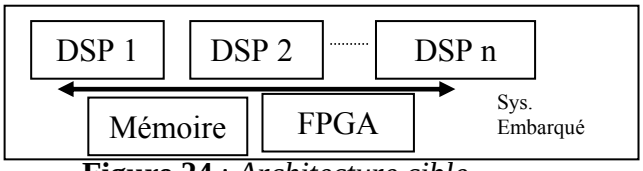


Figure 24 : *Architecture cible*

- Les Divers DSPs de la plate-forme peuvent fonctionner à diverses fréquences chacun.
- Les travaux d'exploration de l'espace des solutions, traitent la consommation du bus par des modèles de type ($P=K.V^2.F.C$). Ce modèle peut être exploité dans les futurs travaux. En fait, vu le niveau d'abstraction assez élevé de l'exploration, un tel modèle serait une solution envisageable.
- La re-configuration dynamique du matériel ne sera pas tenue en compte pour le moment.

Comme déjà cité, cette cible n'exclut pas la possibilité d'intégrer des tâches en matériels. C'est le cas de l'exemple MPEG2, où on peut implanter l'estimation du mouvement en matériel et explorer les différentes possibilités des autres tâches.

III.2.3 Approche

NB: Comme déjà cité, dans ce travail, on cible une architecture essentiellement logicielle (DSP1, DSP2,..DSPn). On traite en premier lieu ce type d'architecture pour profiter du travail établi précédemment au niveau de l'équipe consommation. En fait, les travaux déjà faits au cours de cette thèse se sont focalisés sur les cibles logicielles, et exploitent des plates-formes logicielles. En plus, ce type d'architecture est utilisé dans les logiciels embarqués notamment par les concepteurs des automobiles. Ce travail peut être ensuite étendu à une architecture mixte.

L'approche repose sur une spécification en graphe de tâches de l'application (Figure 25-A). Pour chaque tâche présente dans la spécification de l'application, la connaissance des paramètres et des performances est nécessaire.

Concernant l'estimation de la consommation des tâches (Figure 25-B), chaque tâche est évaluée en terme de temps et de consommation en fonction de la cible et de ses paramètres. L'étape suivante (Figure 25-C) consiste à élaborer une bibliothèque de modèles de temps, consommation des diverses tâches de l'application sur diverses cibles. Une librairie de modèles de performances temporelles et de consommation sera mise en place. Un exemple de librairie de modèles de performances est proposé dans (ktari et al., 2005). Cette librairie traitant la partie logicielle, est établie à travers la méthodologie FLPA (Functional Level Power Analysis) étendue.

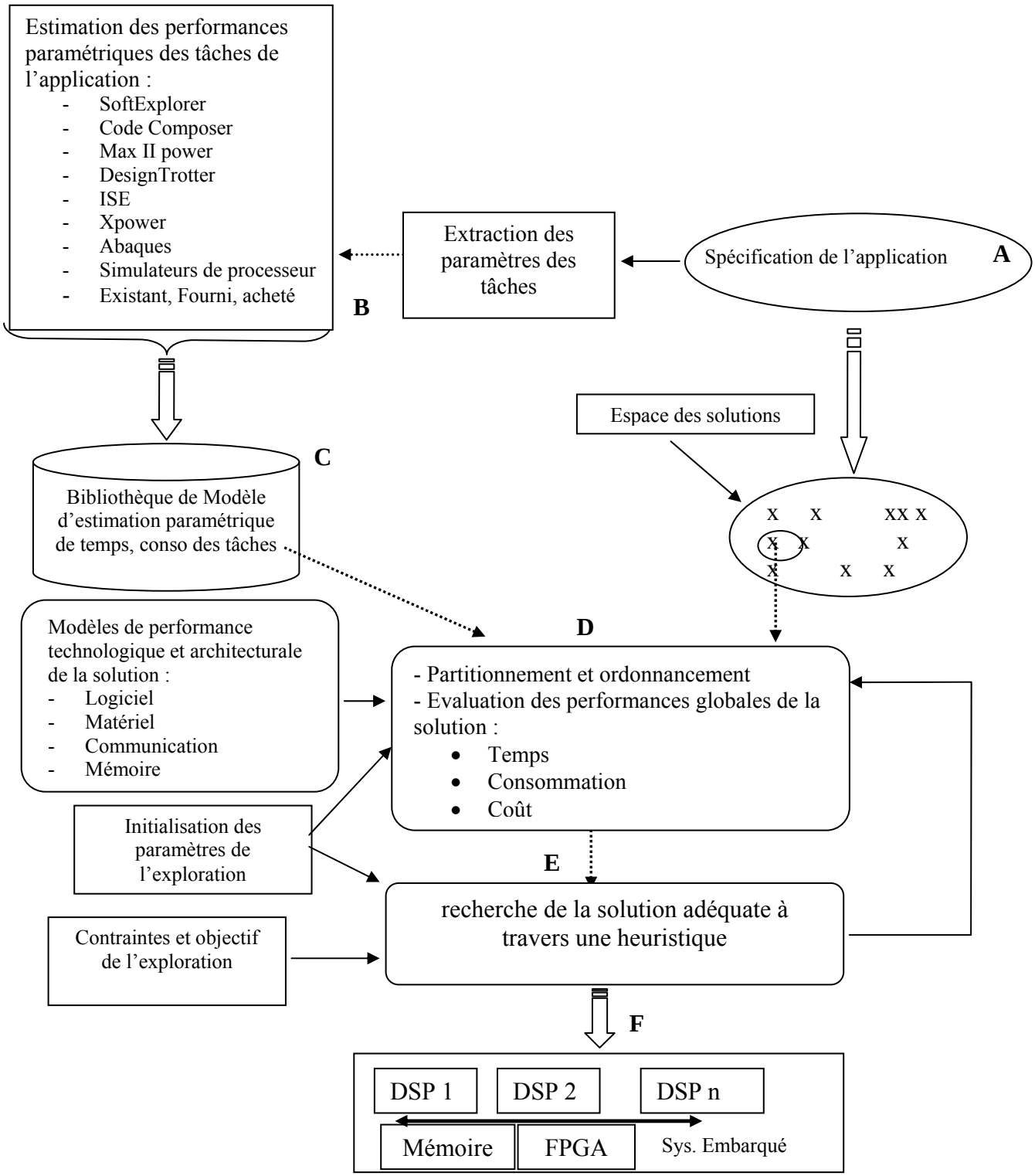


Figure 25 : *Flot d'exploration de l'espace*

D'autres modèles de consommation matériels développés peuvent être exploités comme (Elleouet et al., 2006) (Garcia et al., 2005). Par ailleurs, ces informations peuvent être récupérées manuellement ou à travers d'autres outils d'estimation logicielles et/ou matérielles.

L'étape suivante est l'exploration des solutions architecturales (Figure 25-D) : Elle est basée sur l'analyse des solutions disponibles et la recherche d'une solution qui répond à l'objectif. L'analyse des solutions consiste à évaluer chaque solution à part et à estimer sa performance et sa consommation (Figure 25-E). Bien entendu des modèles de performance des unités de traitement et de communication (DSP, FPGA, mémoire, communication) sont nécessaires afin d'évaluer la performance de tout le système.

Suite à l'analyse des diverses solutions, il faut rechercher la solution « optimale » (Figure 25-F) qui minimise la consommation et respecte les contraintes. En effet, une ou plusieurs architectures peuvent être éligibles et respectent les contraintes du temps réel, de la surface et de la consommation. C'est à ce moment que l'algorithme d'exploration intervient afin de choisir la meilleure solution.

III.2.4 Modèles de performance temporelle

Concernant les performances temporelles, on introduit le partitionnement et l'ordonnancement afin d'extraire le modèle temporel. En fait, le partitionnement et l'ordonnancement de tâches sont deux problèmes récurrents dans le domaine des systèmes temps réel. Le partitionnement consiste à attribuer à chacune des tâches d'un programme un processeur sur lequel se fera l'exécution. On désigne par ordonnancement, le fait d'allouer des ressources et du temps aux tâches sur un processeur donné, de telle manière que certaines conditions soient remplies. Souvent, dans les systèmes embarqués, chaque processeur dispose de son propre ordonnanceur (Noseda, 2002).

Actuellement divers travaux (Bandyopadhyay et al., 2004) (Emmanuel et al., 2001) (Ktari et al, 2008a) traitent ce problème pour des architectures multiprocesseurs. Dans cette étude, l'exploitation d'un ordonnanceur prêt et validé est une solution envisageable. En effet, le gain en consommation paraît surtout lors du partitionnement en attribuant les tâches de calcul aux processeurs faible consommation. Le temps d'exécution total qui dépend de l'ordonnancement et du partitionnement influe sur la consommation totale de l'application.

III.2.5 Modèles de performance énergétique

Dans cette section on propose des modèles de consommation paramétriques. Ces modèles peuvent tenir compte de divers paramètres :

- Les caractéristiques des ressources en veille,
- La fréquence de fonctionnement

- La tension d'alimentation, la fréquence et la taille du bus,
- La taille des données à transmettre,
- La consommation statique et dynamique des modules matériels

Afin de faciliter la compréhension, on l'introduit à travers un exemple. En effet, prenons le cas du graphe suivant:

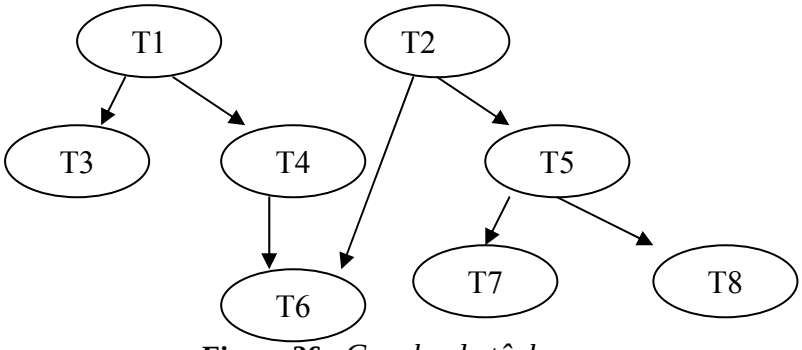


Figure 26 : *Graphe de tâche*

On a ici 8 tâches dépendantes, essayons de proposer manuellement un partitionnement et un ordonnancement pour cette application qu'on désire implanter sur trois DSPs et un FPGA. Les temps d'exécution et la consommation de chaque tâche sont fournis pour chaque cible.

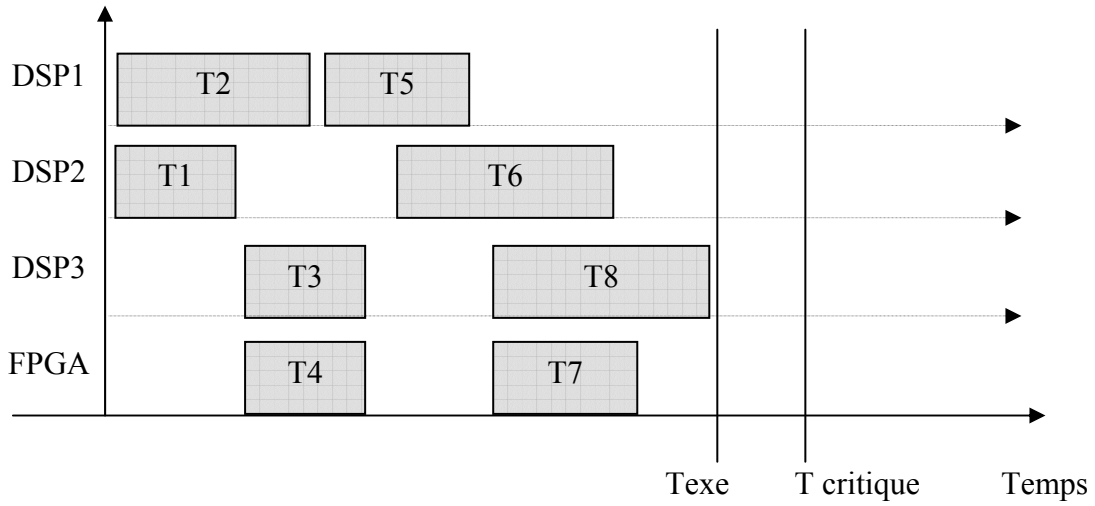


Figure 27 : *Chronogramme*

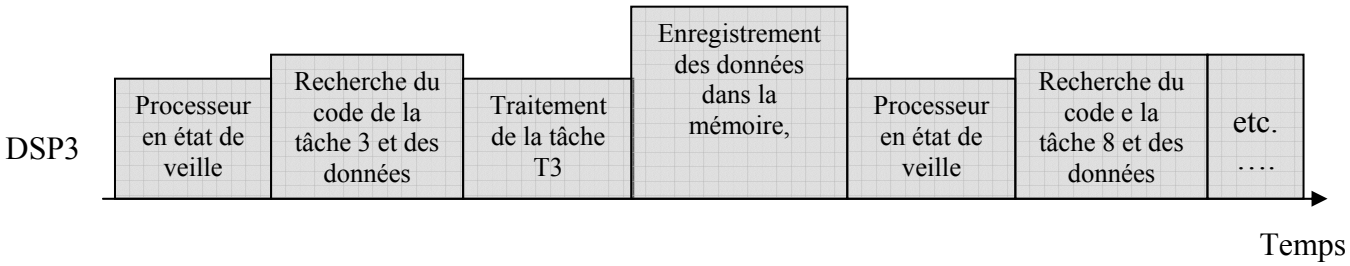
Ce partitionnement & ordonnancement (8 tâches | 4 cibles) n'est pas obligatoirement une solution optimale. Le but de ce chronogramme est d'étudier les performances et la consommation de la solution. Suite à cette étape de partitionnement & ordonnancement qu'on va détailler par la suite, on a la possibilité de confirmer si l'application respecte les contraintes

de temps réel ou non. On a aussi les informations concernant la répartition des charges sur les différentes cibles, le temps de début et de fin de chaque tâche.

Afin d'évaluer la consommation de l'application, prenons le cas d'un DSP, un FPGA, le bus et la mémoire :

III.2.5.1 Modèle de consommation d'un DSP

Etudions la consommation de ce DSP3 :



- Dans la 1ère phase, le DSP est en état de veille. Aucune tâche ne lui est attribuée, mais il consomme une énergie en veille.
- Dans la 2eme phase, le DSP accède à la mémoire centrale pour rechercher la tâche 3 ainsi que les données fournies par la tâche1.
- Dans la 3ème phase, le processeur traite la tâche 3.
- Dans la 4ème phase, le DSP accède à la mémoire centrale pour enregistrer les données.

etc.

Ainsi, le modèle de consommation de ce DSP, peut être établi de la façon suivante :

Energie(DSP3)=P_Veille*T_veille1+P_tâche3*Texe_T3+P_tâche8*Texe_T8+ P_Veille*T_veille2 (7)

D'une façon plus générale, la consommation d'un DSP(i) sera :

$$\text{Energie(DSPi)}= P\_veille * (Texe\_totale - \sum_{tache(i)} T(i)) + \sum_{Tache(i)} Texe(Ti) * P(Ti) \qquad (8)$$

$$\text{Avec Texe\_totale}= \sum_{Tache(i)} T(i) + \sum T\_veille \qquad (9)$$

Le temps de réveil du processeur n'est pas considéré pour le moment, ce temps ne dépasse pas 10 cycles pour un TMS320C6000 à titre d'exemple.

III.2.5.2 Modèle de consommation de la communication

Dans cette étude, la communication est gérée via un bus partagé. Les travaux de modélisation de la consommation d'un bus discret ne sont pas nombreux. Un modèle de consommation d'un bus PCI décrit dans (Kappagantula et al., 2003) peut être exploité à titre d'exemple.

$$P\_bus=1/2*C\_bus*V^2*N\_bits*M \quad (10)$$

Avec N_bits : la largeur du bus. C_bus, V et N_bits sont fournis par le concepteur du bus. Le temps de communication est :

$$Tcomm= \frac{\frac{2*N\_data}{N\_bit\_bus}+2}{F} \quad (11)$$

Ainsi l'énergie consommée par le bus sera :

$$Energie\_bus= 1/2*C\_bus*V^2*N\_bits*M*\frac{\frac{2*N\_data}{N\_bit\_bus}+2}{F} \quad (12)$$

Vu le niveau d'abstraction assez élevé, il est assez difficile d'avoir un modèle de bus plus précis.

III.2.5.3 Modèle de consommation d'une mémoire

Concernant la consommation de la mémoire, vu que sa taille est en pleine croissance dans les systèmes embarqués (90% de la surface en 2011), il est utile de la modéliser et de l'intégrer dans la consommation générale. En fait, lors de l'accès à la mémoire en lecture ou écriture, ce périphérique va consommer de l'énergie qui s'ajoute à la consommation statique. (Marteil, 2006)

$$Energie\_memoire=Texe\_totale*Pstat+\sum_{N\_accés} P\_accès(R/W)*T\_accès \quad (13)$$

Avec P_accès : la puissance consommée par la mémoire lors de l'accès en lecture ou en écriture. Cette information est généralement fournie par le concepteur de la mémoire.

III.2.5.4 Modèle de consommation du matériel : FPGA

Concernant la consommation des architectures FPGA, le modèle de puissance doit tenir compte de la consommation statique et dynamique. En effet la consommation d'un FPGA est due à la consommation des tâches actives en traitement ainsi qu'à la consommation de toutes les tâches synthétisées que se soit en veilles ou actives.

$$P(FPGA)= \sum_{Tache\_active} P\_dynamique + \sum_{Taches} Pstat \quad (14)$$

$$E(FPGA)= \sum_{Tache\_active(i)} P\_dynamique\_Tache(i) * Texe(i) + \sum_{Taches} Pstat * Texe\_totale \quad (15)$$

III.2.5.5 Modèle global

Le modèle de consommation globale de l'architecture, qui est composé de divers modules, doit tenir compte des diverses sources de dissipation d'énergie. Pour être proche de la réalité, le modèle développé tient compte de la consommation des unités de traitements implantés, du bus et de la mémoire. Ainsi

$$Energie\ (Totale)= \sum_{Cible(i)} Energie(cible(i)) + Energie\_bus + Energie\_memoire \quad (16)$$

Il est à noter que ces équations tiennent compte des divers paramètres de l'application et de l'architecture. En effet, à travers cette méthodologie, les modèles de performances de chaque tâche sont modélisés en fonction de divers paramètres (fréquence, type de DSP, taille de l'image, cadence…). L'avantage de cette méthode consiste à proposer une tâche pouvant avoir plus qu'un modèle de performance sur une cible donnée en jouant sur les paramètres.

III.2.6 Modèle coût

Vu la présence dominante de la partie logicielle (DSPs) vis à vis des FPGA, le coût sera une contrainte un peu figée, qu'on ne peut améliorer qu'on modifiant le nombre de ressources. Par ailleurs, le coût de chaque ressource est pondéré par un coefficient vu la diversité des coûts technologiques (un mm² de surface de DSP peut coûter moins cher que celui d'un FPGA).

$$Cout\_tot= \sum_{Ressources} \alpha i * Surfaces(i) \quad (17)$$

III.2.7 L'exploration basse consommation

La solution architecturale proposée pour cette application a trois caractéristiques : énergie, coût, temps (figure 27). Comme ces trois paramètres interagissent ensemble, il y a une nécessite pour faire une étude exploratrice globale de la solution. Parmi les points clefs de l'exploration de l'espace, on cite la performance globale de l'application. Pour cela, des modèles d'estimation sont nécessaires afin d'évaluer de l'application en sa globalité.

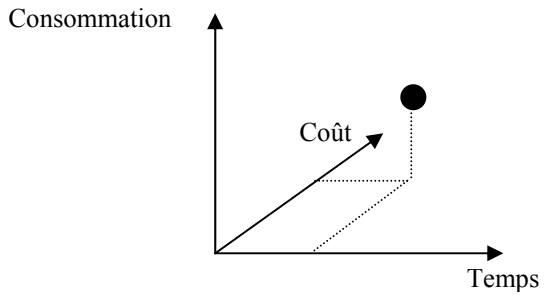


Figure 27 : *Champs d'exploration de l'espace des solutions*

Ces modèles d'estimation seront de haut niveau et tiennent compte des paramètres architecturaux et algorithmiques. Dans le cadre de ce travail, l'étude est menée sur des applications écrites en langage C. Pour cela, la méthode d'estimation basée sur la FLPA étendue, détaillée dans les chapitres précédents, a été exploitée afin de proposer des modèles paramétriques de haut niveau. Ces modèles d'estimation seront présentés dans la section suivante.

III.2.8 Méthodes d'estimation

A partir de l'analyse fonctionnelle de l'application, la méthodologie FLPA permet de développer un modèle paramétrique, qui représente le comportement de consommation d'une cible. (Figure 28)

En fait, cette méthodologie se compose de quatre étapes.

- L'analyse fonctionnelle: qui détermine les paramètres influant sur le modèle de puissance.
- La caractérisation de chaque paramètre est accordée pour qualifier son influence sur la consommation de l'application.
- Le modèle général est établi selon les paramètres disponibles.
- Validation du modèle par mesures.

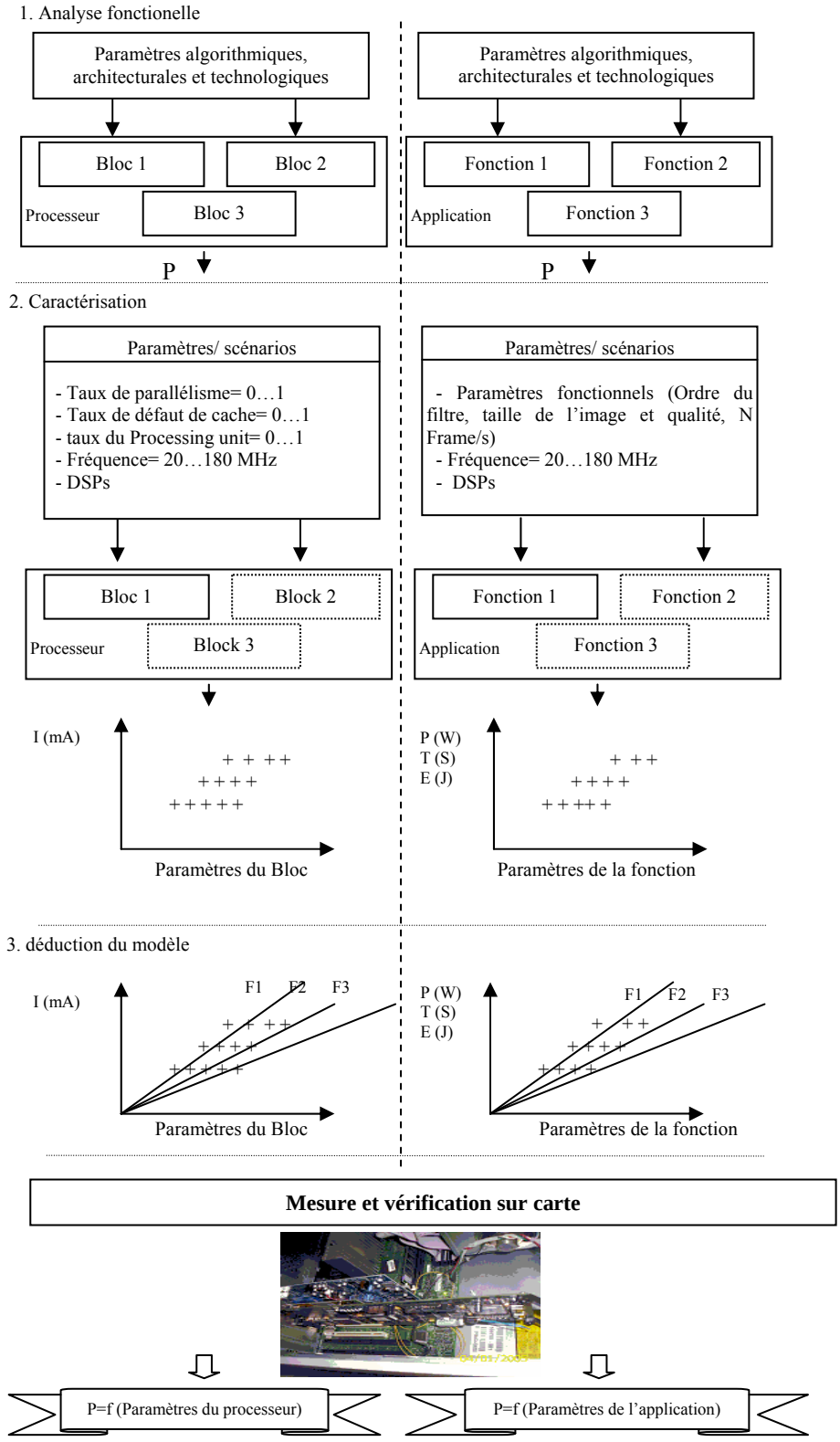


Figure 28 :(a) *FLPA : Méthodologie pour les processeurs* ***(b)*** *FLPA transposée pour IP(SW)*

Ainsi, on peut tenir compte des caractéristiques algorithmiques, afin d'évaluer la consommation à ce niveau selon les variations des paramètres de IP (Intellectual Propriety). En fait, cette méthodologie part de l'extraction des paramètres algorithmiques, architecturaux

et technologiques qui ont une influence directe sur la consommation de l'application (taille de l'image, résolution, nombre d'images par seconde, précision du calcul, DSP cible, fréquence de fonctionnement). L'étape suivante consiste à extraire la variation de la consommation en fonction de chaque paramètre extrait à travers des estimations ou mesures grâce à des scénarios. Enfin, la formulation de lois de consommation mathématiques en fonction de ces paramètres est établie. Une confrontation des modèles établis avec les mesures sur carte DSP est envisageable afin d'avoir une idée sur la précision de ces modèles. Ces modèles établis pour diverses applications de traitement de signal grâce à cette méthodologie seront intégrés dans la librairie de performances nécessaire lors de l'exploration.

Avec cette librairie, on a la possibilité d'avoir des modèles paramétriques des diverses tâches. Ces modèles peuvent tenir compte :

- des paramètres de l'application : taille de l'image, résolution, cadence, chrominance, ordre du filtre,…
- des paramètres architecturales et technologiques : fréquence de fonctionnement, tension d'alimentation de la cible, la cible…

Cette bibliothèque sera une base de modèles prêts et paramétrables exploitable pour les diverses applications. Ainsi, en cas de modification de la spécification d'une tâche de l'application, on n'a pas besoin de tout re-modéliser. Toutefois, cette bibliothèque n'est pas toujours complète et nécessite souvent des mises à jour. En effet, un problème se pose si une nouvelle tâche se présente dans l'application que ce soit par ajout par le concepteur ou par modification. Dans ce cas, diverses solutions sont possibles afin de proposer un modèle d'estimation des performances de la nouvelle tâche et ceci:

- à travers des outils comme Softexplorer, Design trotter, l'environnement ISE-Xpower, Code Composer, Max II power, Quartus power play analyser, les simulateurs des processeurs.
- à travers : les datasheets des cibles indiquant la consommation moyenne, les abaques de consommation, des modèles de consommation simplistes du type : P=0.063 Freq Area. Avec ces techniques, on a la possibilité d'avoir des modèles de performance assez rapidement avec une précision relativement moyenne.
- à travers : des mesures sur carte en cas de disponibilité.

III.2.9 Conclusion

La formulation d'une méthode d'abstraction de l'architecture et de l'application a permis la mise en œuvre de modèles de performances à un haut niveau. Ces modèles de performances temporelles et énergétiques sont combinés et évalués à travers une approche d'exploration de l'espace de solution. Cette étape représente une étude exploratrice du système embarqué en sa globalité permettant d'aboutir à des solutions qui respectent les diverses contraintes en mettant une attention particulière à la consommation. L'environnement d'exploration sera le sujet de la section suivante.

III.3 Outil d'exploration

On présente dans cette section l'environnement d'exploration. La présentation est accompagnée d'un exemple d'illustration. L'environnement repose sur deux outils, le premier sert pour la saisie de la spécification à partir du graphe, et des performances à partir de la librairie, le second pour l'évaluation des performances de toute l'architecture et l'exploration de l'espace des solutions afin d'extraire la solution adéquate (Figure 29).

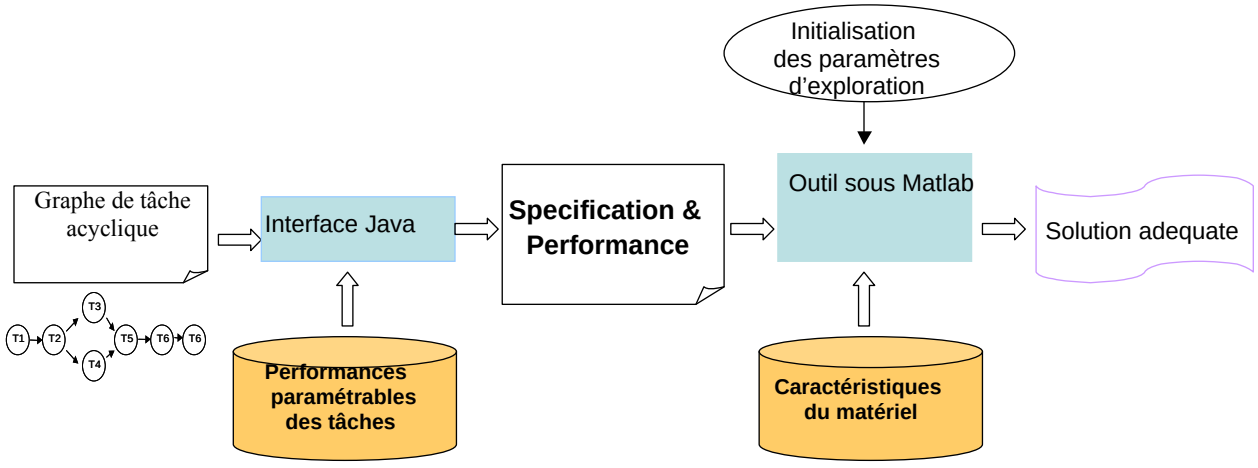


Figure 29 : *Environnement d'exploration*

Les informations nécessaires pour l'exploration englobent les diverses implémentations possibles de chaque tâche. Vu qu'une tâche peut avoir plusieurs performances selon ces paramètres, chaque implantation tient compte des paramètres algorithmiques et architecturaux lors de la saisie du temps d'exécution, la puissance moyenne, la puissance maximale, la taille des données résultantes. Cette description est gérée par une interface graphique écrite en java.

Elle permet de générer un fichier .txt contenant ces informations sur le graphe. Ce fichier de description textuelle de l'application sera l'entrée principale pour le calcul sous Matlab.

Cette interface (Figure 30) permet de simplifier la saisie des informations, et d'avoir un affichage et une lecture plus lisible des informations caractérisant l'application.

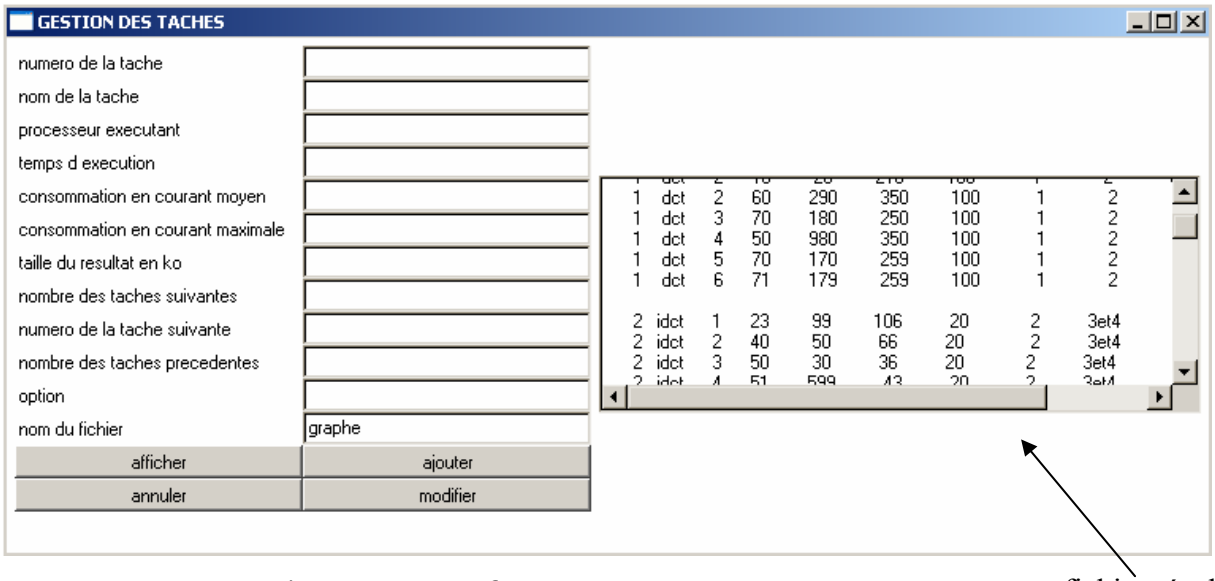


Figure 30 **: *Interface de description de l'application***

Pour le moment, la saisie des performances de chaque tâche est faite manuellement. Une « intégration » automatique des performances des tâches est possible en exploitant directement la base de donnée. (Figure 31)

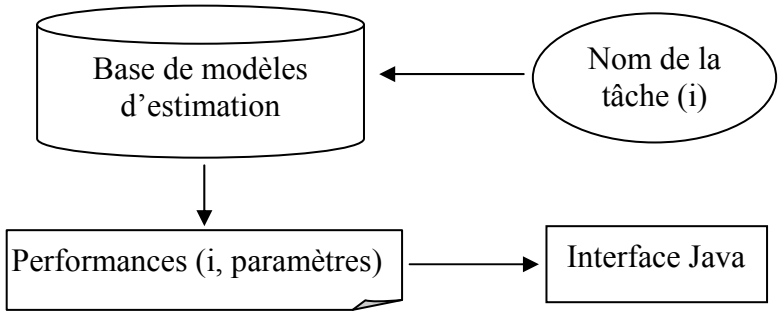


Figure 31 : *génération automatique des performances des tâches*

Par ailleurs, les diverses caractéristiques des cibles (DSPs, FPGA et bus) peuvent être saisies dans un fichier de matériels utilisé comme entrée par Matlab tel que : la puissance en veille du DSP, les caractéristiques spécifiques du bus (Vcc, Freq, capacité).

Une fois que les diverses informations sont récupérées et intégrées dans Matlab, l'évaluation et l'exploration basée sur l'heuristique du recuit simulé se déroulent.

III.3.1 Stratégie d'exploration

III.3.1.1 Heuristique

Afin d'extraire une solution adéquate parmi celles présentes dans le large espace des solutions tout en respectant les contraintes du système, l'utilisation d'une méta heuristique est nécessaire afin de résoudre ce problème d'optimisation NP complet. En fait, avec les algorithmes d'améliorations itératifs classiques, le processus de recherche est itéré jusqu'à ce que toute modification rende la solution moins bonne. La figure 32 montre que cet algorithme d'amélioration itérative ne conduit pas en général au minimum absolu, mais seulement à un minimum local «A », qui constitue la meilleure des solutions accessibles compte tenu de l'hypothèse initiale.

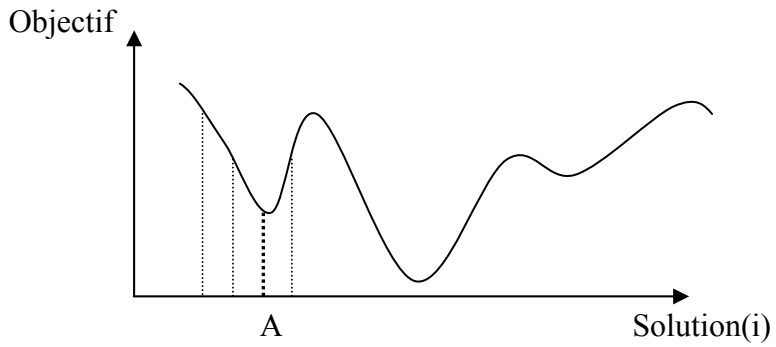


Figure 32 : *Allure de la fonction « objectif » d'un problème d'optimisation difficile*

Avec les métaheuristiques dites de voisinage (recuit simulé, méthode tabou) : il s'agit d'autoriser, de temps en temps, des mouvements de remontée, autrement dit d'accepter une dégradation temporaire de la situation, lors du changement de la configuration courante. Un mécanisme de contrôle des dégradations permet d'éviter la divergence du procédé. Il devient dès lors possible de s'extraire du piège que représente un minimum local, pour partir explorer une autre « vallée » plus prometteuse. (Lacomme et al., 2003)

Au cours de ce travail, on a exploité l'algorithme du « recuit simulé ». L'avantage de cette méthode c'est son aptitude de procurer une solution de bonne qualité. En outre, c'est une méthode générale : elle est applicable et facile à programmer, pour la majorité des problèmes qui relèvent des techniques d'optimisation itérative. Par ailleurs, elle offre une grande souplesse d'emploi, car de nouvelles contraintes peuvent être facilement incorporées. Il s'agit d'une méthode où l'exploration complète du voisinage de la solution actuelle est remplacée par le tirage au sort d'une solution voisine. On passe sur cette solution si la variation D du coût est négative. Sinon, on va quand même sur la solution de coût supérieur avec une probabilité exp(-D/T) paramétrée par un réel positif T appelé température. On

recommence le processus sur la nouvelle solution après avoir baissé légèrement la température T. (Lacomme et al., 2003)

On s'arrête quand T devient négligeable, c'est-à-dire inférieure à un petit réel positif dont la probabilité d'acceptation correspondante soit presque nulle. A ce moment, la probabilité de remonter sur une moins bonne solution est quasi-nulle, et la méthode se comporte comme une recherche locale. Le recuit simulé peut donc échapper aux minima locaux puisqu'il accepte d'augmenter le coût. Il donne de très bons résultats s'il est conduit assez lentement : $T_{n+1}=f(T_n)$. En effet, Le choix du schéma de décroissance est crucial dans cet algorithme car une décroissance trop rapide peut piéger la solution dans le voisinage d'un minimum local.

Algorithme:

```
choose an initial solution (Sol_Initial[1..N ])
choose an initial & final temperature T0 & Tf;

Current_solution=Sol_initial
        While(T(i)<Tf)
{
        New_solution=find a near current_solution
        Calculate Δ cost =Cost(NewSol) - Cost(Current_solution)
                If Δ  cost ≤ 0
                    Current_solution= New_solution
                Else
                  R=rand[0..1];
                  if R≤ exp (- Δcost  /T(i) )
                  Current_solution= New_solution
                  end
                end
   T(i+1)=decreasing function (T(i))          //cooling function
}
```

Pour démarrer la recherche par le recuit simulé, les paramètres de l'algorithme doivent être bien choisis. C'est le cas de la température initiale, le taux de décroissance de la température et le critère d'arrêt du programme.

- Température initiale T0 : on peut la calculer au préalable à l'aide de l'algorithme suivant :

- Faire 100 perturbations au hasard ; évaluer la moyenne Δcost des variations correspondantes.
- Choisir un taux initial d'acceptation R0 de 50% par exemple afin d'explorer le maximum d'espace.
- Déduire T0 de la relation R0= exp (- Δcost /T0)

- Décroissance de la température : peut être effectuée selon la loi géométrique : T(k+1)=0.9*T(k)
- Arrêt du programme : peut être opéré après 2 ou 3 paliers de température successifs sans aucune nouvelle acceptation (figure 33). De cette façon, on garantit que la solution choisie n'est pas locale.
- Vérification indispensables lors des premières exécutions du programme :

- Le générateur de nombre réels aléatoires dans [0,1] doit être bien uniforme.
- La « qualité » du résultat doit varier peu lorsque le programme est lancé plusieurs fois avec des configurations initiales différentes.

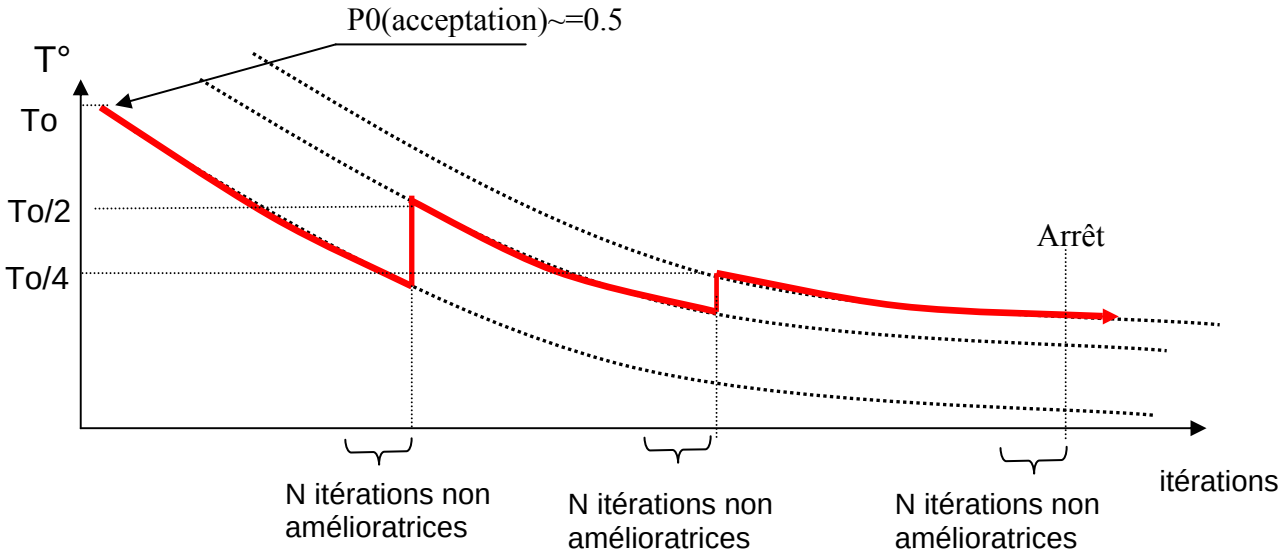


Figure 33 : *Condition d'arrêt du programme*

III.3.1.2 Implémentation

Dans le cadre de l'implémentation de l'outil sous Matlab, une exploration mono-objective est mise en place. Elle a permis selon le choix, d'atteindre des solutions architecturales faible consommation sous des contraintes de temps réel ou bien des solutions performantes en terme de temps d'exécution sous des contraintes énergétiques. Comme le concepteur a la possibilité d'imposer seulement le nombre maximal de processeurs dans l'architecture sans fixer le nombre exact, le nombre de processeurs ne sera pas figé. L'algorithme va explorer les solutions les plus prometteuses parmi celles qui respectent les contraintes de temps réel et le nombre maximal de processeurs (Figure 34). C'est à l'outil

d'extraire le nombre de processeurs utiles ainsi que le *mapping* architectural adéquat et les performances de tout le système. Ce paramétrage du nombre des unités de traitement permettra au concepteur d'une part, de ne pas se limiter à une architecture unique lors de la conception du produit et d'être guidé par l'outil lors du choix de la solution matérielle d'autre part.

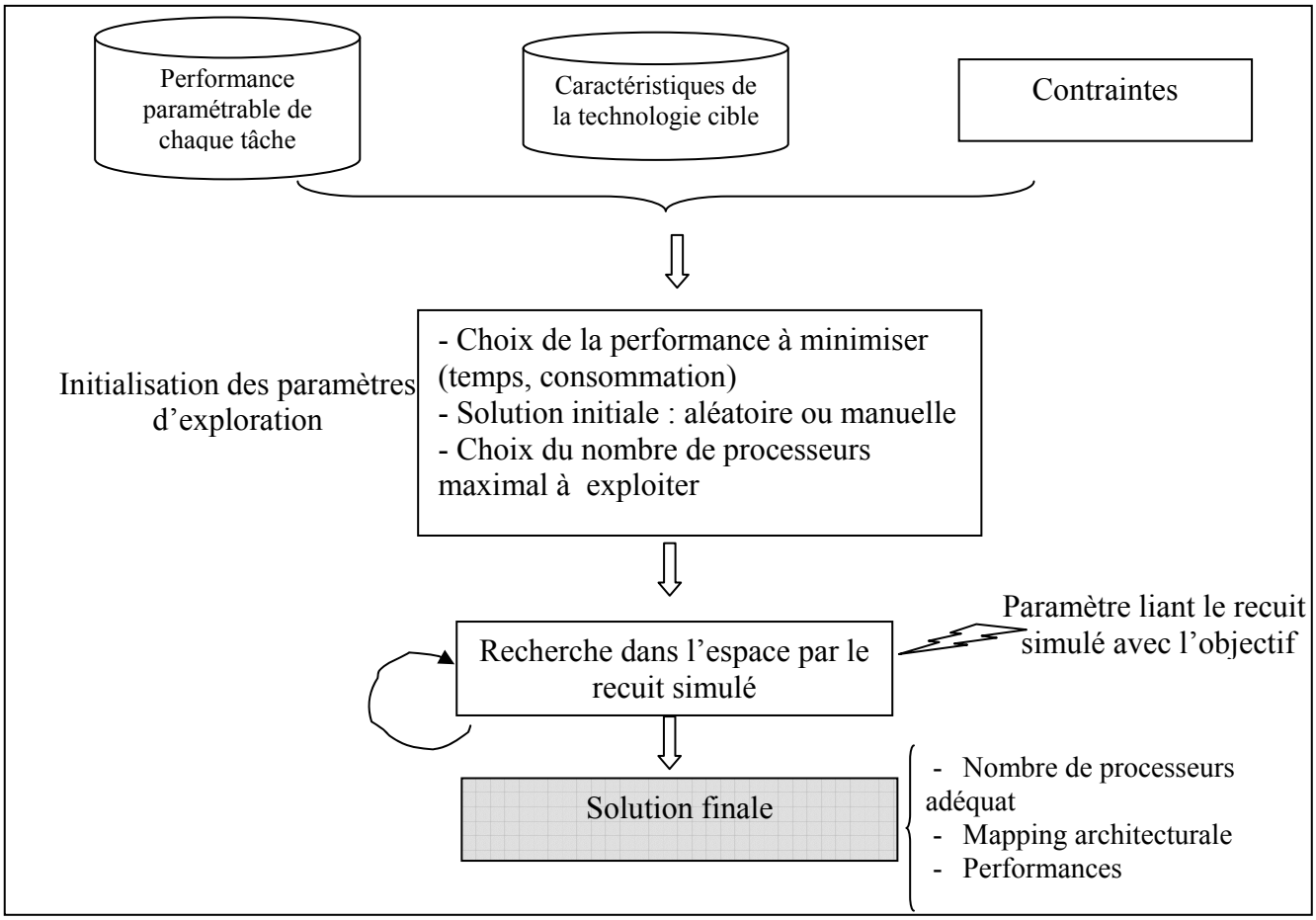


Figure 34 : *Description de l'exploration*

Dans la figure 34, on détaille la méthode exploitée lors de l'implémentation. Cette méthode repose sur :

- *Les performances paramétriques des tâches présentes dans la description textuelle du fichier en entrée*: Avec la diversité des valeurs de performances existantes en fonction des paramètres algorithmiques et architecturales établies, une librairie de modèles peut être analysée par l'outil. Ceci permet l'évaluation d'une multitude de performances pour chaque tâche et l'ajustement de ses paramètres selon l'objectif.

- *Les caractéristiques de la technologie cible:* Les caractéristiques de chaque technologie cible sont nécessaires afin de pouvoir estimer la performance globale de tout le

système. Parmi ces caractéristiques, on peut citer : la tension d'alimentation, la fréquence et la taille du bus, la puissance en veille des ressources à exploiter, etc.

- *Les contraintes* : Les contraintes de l'application sont définies par le concepteur du produit. Elles seront fournies à l'outil afin d'accepter ou de refuser les solutions extraites au cours de l'exploration.

Vu que l'architecture cible est paramétrable avec un nombre de ressources variables et vu les contraintes de surface, de technologie, de coût ou autres, le nombre maximal « Nmax » d'unités de calcul sera fixé dés le départ par le concepteur. Grâce à l'heuristique d'optimisation qui va explorer l'espace de solutions, le nombre de ressources utiles va être choisi. Pour cela, l'outil balaye plusieurs configurations en évaluant la performance globale de chacune en exploitant les informations en entrée. A la fin de l'exploration, l'outil fournit la solution qui répond à l'objectif avec des détails sur sa performance temporelle et énergétique. Le *mapping* architectural est géré par l'outil, il associe à chaque unité de traitement les tâches adéquates afin d'atteindre l'objectif choisi.

Dans la figure 35 on présente, à titre d'exemple, les modèles de performances des tâches et les informations architecturales requises. Pour chaque tâche de l'application, on associe un ensemble de valeurs de performances temporelles et énergétiques relatives à chaque cible. Ces informations seront traitées et exploitées afin d'évaluer la performance de tout le système lors de l'exploration selon l'objectif visé afin d'extraire une solution adéquate.

III.3.2 Résultats et analyse de l'espace d'exploration

Une initialisation des paramètres d'exploration de l'espace de solutions est nécessaire. En fait, l'utilisateur ou le concepteur a la possibilité de choisir une solution initiale aléatoire ou une solution particulière selon ses connaissances sur le comportement de l'application en terme de consommation. Les contraintes du système seront considérées lors de l'évaluation de chaque solution afin de satisfaire le besoin. Dans cette étude de cas, la contrainte de temps ainsi que le nombre maximal de ressources à exploiter sera imposé par le concepteur à l'outil. Ceci va limiter le champ d'exploration à Nmax unités de traitement dés le départ selon les contraintes. Par ailleurs, l'algorithme d'exploration permet d'extraire la solution la plus prometteuse selon l'objectif en précisant les différentes unités et le partitionnement de l'architecture finale qui peut contenir un nombre d'unités N'<Nmax.

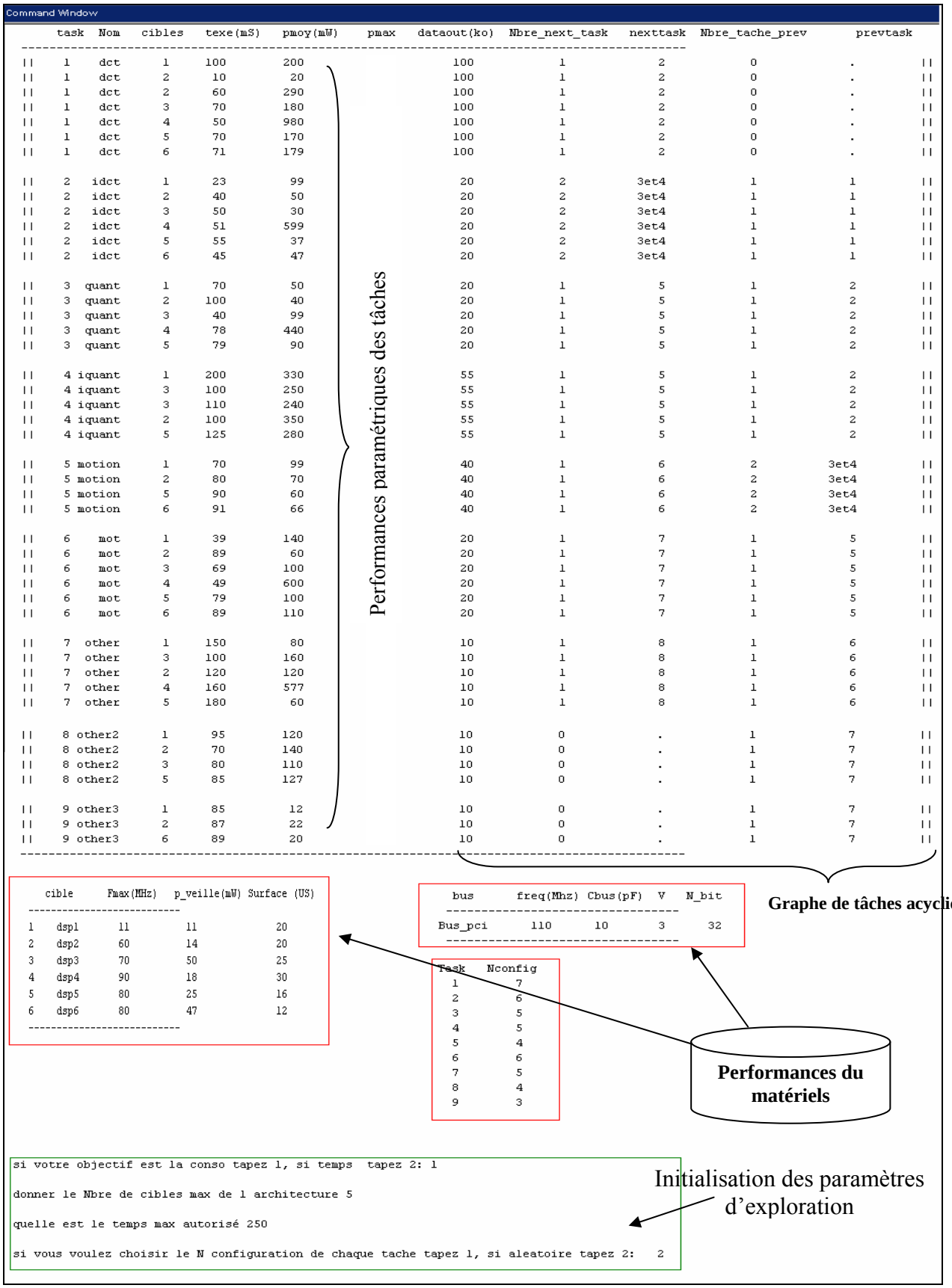


Figure 35 : *Description de l'application sous Matlab*

Ainsi, on extrait le nombre de ressources nécessaires à exploiter pour implémenter l'application. L'outil guide ainsi le concepteur lors du choix de l'architecture cible en terme de nombre et de type de ressources à un niveau avancé lors de la conception du produit.

La figure 36 présente les résultats de l'exploration dans un espace contenant 6 unités de traitement (DSPs) avec un objectif d'extraire une solution faible consommation respectant une contrainte temps réel stricte. L'algorithme converge « rapidement : 1000 itérations» vers une solution contenant 3 processeurs seulement dont la description est présentée dans la figure 36-F. C'est grâce à l'heuristique du recuit simulé que le problème de complexité de l'espace est réduit. Ainsi, l'utilisateur peut à priori connaître le nombre de DSP adéquat à son application et le *mapping* architecturale qui minimisent la consommation de tout le système tout en respectant les contraintes.

Dans la figure 36-A, on présente le résultat de l'exploration de l'espace de solution en se basant sur la heuristique du recuit simulé. L'outil explore l'espace à travers cette heuristique et converge vers une solution dont la consommation est moins de 75 mJ et dont les détails architecturaux sont présentés dans la figure 36-F. Avec cette heuristique d'exploration, l'outil atteint une solution assez bonne et rapidement en la comparant à celle extraite par une recherche « assez complète ». D'ailleurs, dans la figure 36-B, on présente à titre indicatif l'espace de solutions « globale » exploré à travers une recherche aléatoire. Il est à signaler qu'avec cette heuristique « intelligente », un gain de temps dans la recherche de la bonne solution est prouvé. Pour plus de lisibilité, la figure 36-D montre l'évolution de la surface et de la consommation pour diverses solutions architecturales dont le nombre de ressources de traitement est variable : de deux à six processeurs.

La figure 36 C-D montre aussi l'évolution de l'énergie en fonction de la surface et/ou le temps permettant ainsi une connaissance détaillée sur le domaine de variation de la consommation selon le nombre d'unités de la solution.

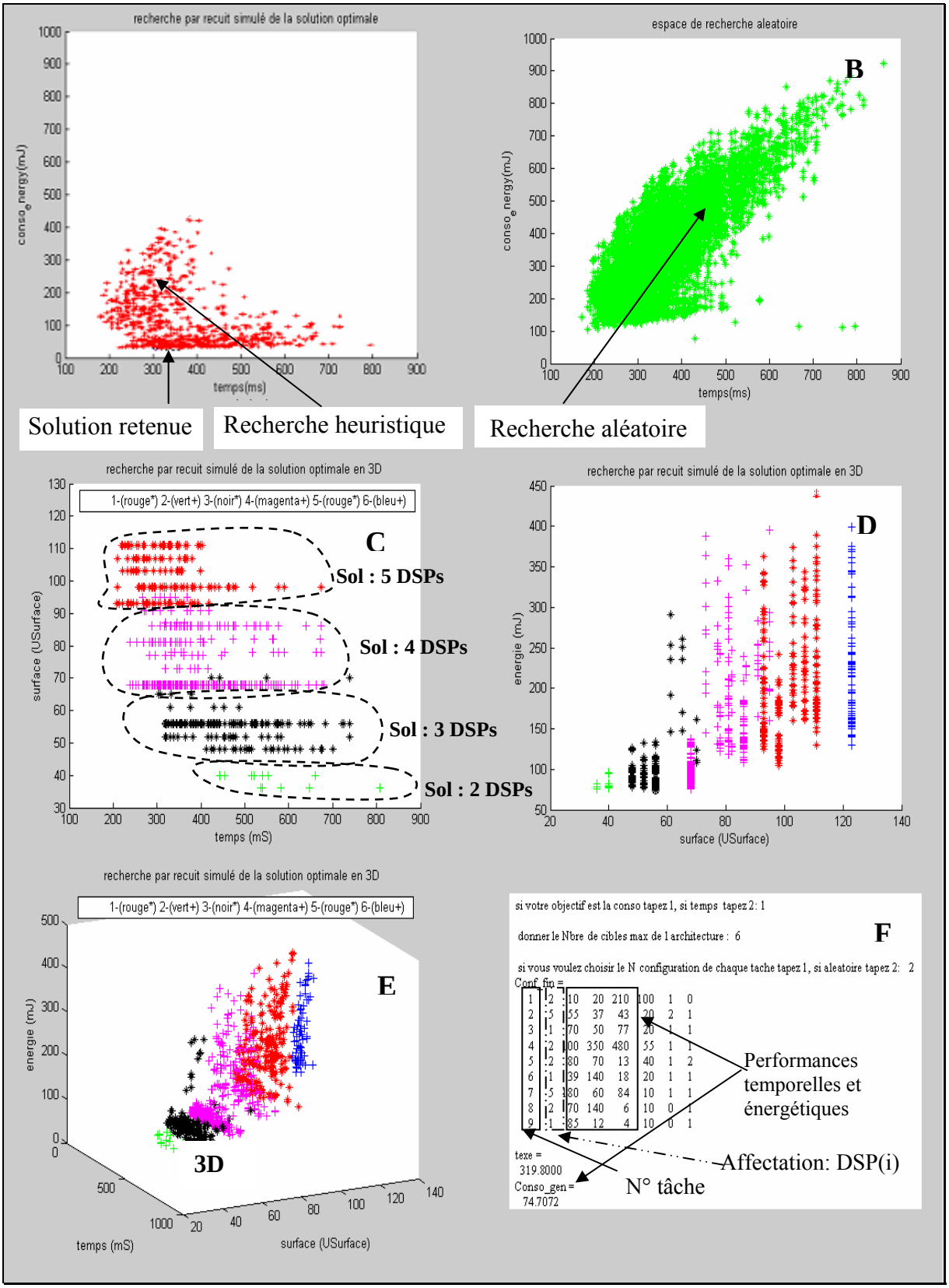


Figure 36 : *Résultats de l'exploration*

Ainsi, le concepteur a la possibilité d'extraire d'une part l'architecture cible adéquate pour son produit avec un minimum d'informations paramétrables à un niveau d'abstraction assez élevé lors de la conception. Par ailleurs, l'outil propose un *mapping* adéquat des tâches de l'application afin d'avoir un système qui répond à l'objectif et aux contraintes. Parmi les points clefs de cette exploration:

- Le paramétrage du nombre de ressources de l'architecture cible à implémenter et le paramétrage des modèles d'estimation en fonction de l'architecture et de l'application,
- La mutli granularité des modèles d'estimation : en fait, au cours de ce travail, les modèles, proposés dans le chapitre suivant, tiennent compte de la granularité de l'application. C'est le cas de l'application MPEG-2 où on a proposé des modèles d'estimation au niveau tâches, au niveau application pour l'étendre au niveau standard vidéo (Pal, Secam et NTSC).

III.4 Conclusion

Dans ce chapitre, on a traité l'aspect faible consommation dans les systèmes embarqués. Une méthodologie et un environnement d'exploration basse consommation de l'espace de solutions sont proposés et mis en place. Cet environnement exploite des modèles d'estimation et de performances temporels et énergétiques riches qui tiennent compte de nombreux paramètres algorithmiques et architecturaux. Ceci a permis de dégager les caractéristiques et les mécanismes nécessaires afin d'extraire une solution architecturale qui répond aux besoins. Les points clés de ce problème sont abordés à travers une méthode d'analyse paramétrique et une heuristique d'exploration de l'espace basée sur le recuit simulé.

Comme suite, il est intéressant d'exploiter cet environnement pour explorer l'espace de solutions d'une application plus significative comme MPEG2 afin de valider l'approche. D'ailleurs au cours de ce travail, des modèles paramétrables de MPEG2 sont établis à divers niveaux de granularité. Ils seront détaillés dans le chapitre suivant.

Chapitre

IV

Expérimentations et Etude de cas

Chapitre IV. Expérimentations et étude de cas

IV.1 Introduction

Dans les chapitres précédents, on a présenté les approches de modélisation et d'estimation de la consommation, ainsi que la nouvelle méthodologie d'exploration basse consommation de l'espace des solutions. L'objectif de ce chapitre est d'une part, partir de quelques applications de traitement de signal standards et d'extraire les paramètres existants et influants sur la consommation et d'autre part, estimer la consommation (énergie & puissance) des DSPs et FPGA afin d'explorer l'espace des solutions. Par la suite, ces estimations seront validées par mesures pratiques sur carte tout en jouant sur les paramètres dégagés. Ceci permet d'explorer l'espace des solutions possibles et d'envisager la meilleure solution dans le cadre de l'adéquation algorithme architecture. Dans ce chapitre, les lois ou modèles de consommation sont établis au début pour des applications « classiques » : la Transformé en Cosinus Discret (DCT) et la Transformé de Fourier Rapide (FFT). Ensuite, une étude énergétique détaillée sur MPEG-2 est menée afin d'explorer son espace de conception. Et afin de valider l'approche proposée, une formulation mathématique basée sur la probabilité est établie.

IV.2 Filtre à réponse impulsionnelle finie

Un filtre numérique non récursif aussi appelé filtre RIF pour "*Réponse Impulsionnelle Finie*" est un filtre numérique dont la sortie ne dépend que des échantillons d'entrées présents et passés. Un tel filtre a une fonction de transfert de type polynomiale. L'expression de celle-ci est la suivante :

$$y[n]=\sum_{k=0}^{N-1} b[k]*x[n-k] \quad (18)$$

x[n] : représente l'entrée du filtre,
b[k] : représente les coefficients du filtre,
y[n] : représente la sortie du filtre,
N : l'ordre du filtre.

La firme *Texas Instrument* fournit avec son outil Code Composer le cœur de cette application. Son étude montre qu'elle est paramétrable en fonction de l'ordre du filtre. Ce

paramétrage permet d'expertiser l'espace des solutions possibles de l'application. Et afin d'étudier l'influence de ce paramètre sur la consommation, les outils SoftExplorer et CodeComposer sont utilisés.

La figure 37 montre la tendance de la consommation sur le C6201 en fonction de l'ordre et de la fréquence. Le tableau 5 montre la variation du temps d'exécution selon les paramètres de l'application sur 3 cibles C6201, C6701 & C5510. Une fois ces valeurs sont dégagées par SoftExplorer, un modèle peut être établi en fonction de l'ordre du filtre pour chaque cible relativement aux estimations de SoftExplorer.

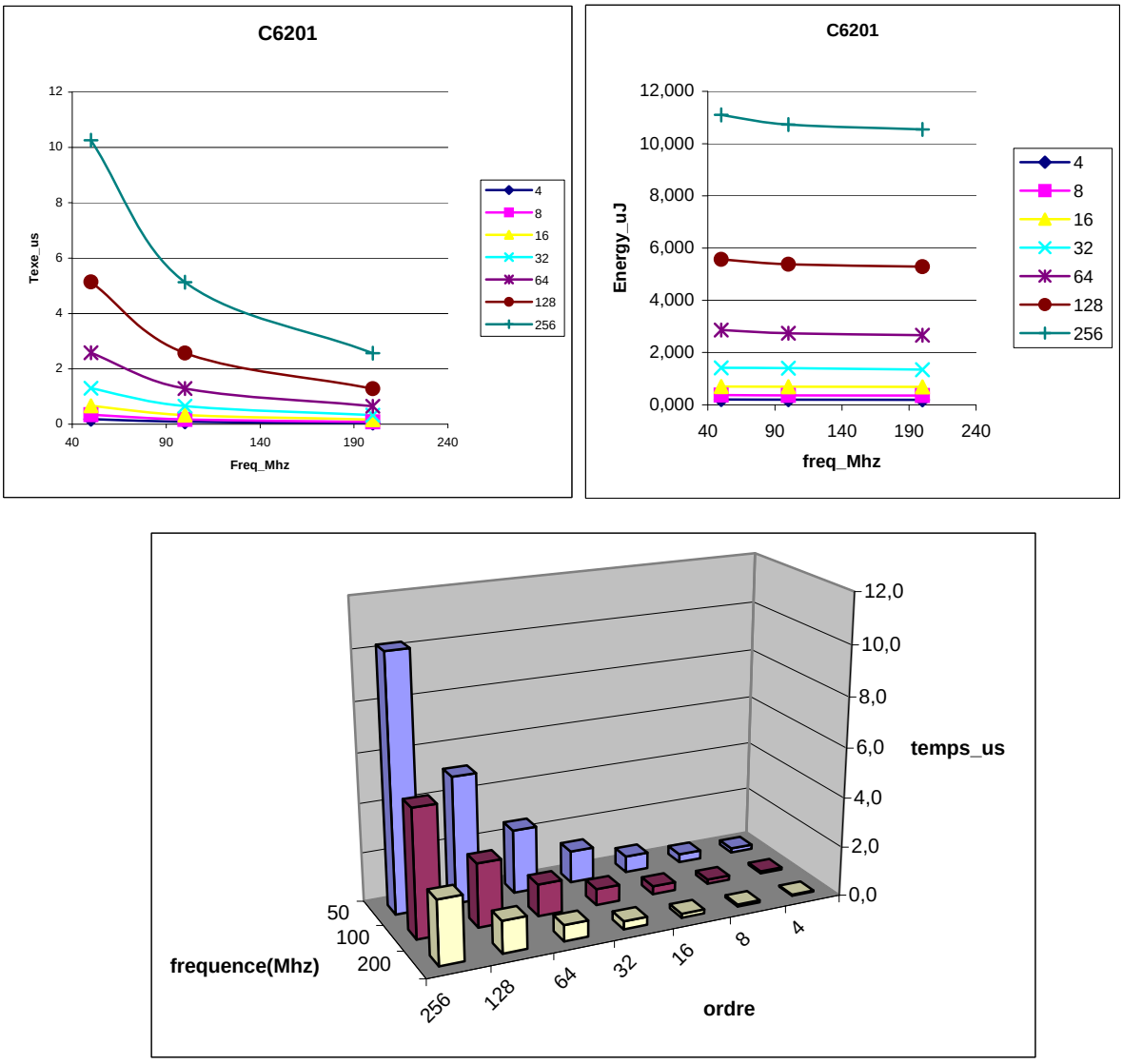


Figure 37 : *Tendance de la consommation sur un C6201*

Tableau 5 : *Temps d'exécution estimé et modélisé de l'application en fonction de l'ordre*

Ordre	4	8	16	32	64	128	256
	C6201&C6701 Modèle : Texe(uS) = 2,006*ordre/freq(Mhz)						
50Mhz	---						
Texe_SE_uS	0,180	0,340	0,665	1,300	2,580	5,140	10,260
Texe_modele_uS	0,160	0,320	0,641	1,283	2,567	5,135	10,270
Erreur %	10,844%	5,600%	3,471%	1,243%	0,478%	0,090%	-0,104%
100Mhz	---						
Texe_SE_uS	0,090	0,170	0,330	0,654	1,290	2,570	5,130
Texe_modele_uS	0,080	0,160	0,320	0,641	1,283	2,567	5,135
Erreur %	10,844%	5,600%	2,739%	1,847%	0,478%	0,090%	-0,104%
200Mhz	---						
Texe_SE_uS	0,046	0,084	0,166	0,326	0,646	1,285	2,565
Texe_modele_uS	0,040	0,080	0,160	0,320	0,641	1,283	2,567
Erreur %	12,78%	4,48%	3,33%	1,55%	0,63%	0,09%	-0,10%
	C5510 Modèle : Texe(uS) = 3,013*ordre/freq(Mhz)						
50Mhz	---						
Texe_SE_uS	0,280	0,520	1,000	1,960	3,880	7,72	15,400
Texe_modele_uS	0,240	0,481	0,963	1,926	3,852	7,705	15,411
Erreur %	14,000%	7,385%	3,680%	1,714%	0,701%	0,187%	-0,073%
100Mhz	---						
Texe_SE_uS	0,140	0,260	0,500	0,980	1,940	3,860	7,705
Texe_modele_uS	0,120	0,240	0,481	0,963	1,926	3,852	7,705
Erreur %	14,00%	7,38%	3,68%	1,71%	0,70%	0,19%	-0,01%
200Mhz	---						
Texe_SE_uS	0,070	0,130	0,250	0,490	0,970	1,930	3,85
Texe_modele_uS	0,060	0,120	0,240	0,481	0,963	1,926	3,852
Erreur %	14,00%	7,38%	3,68%	1,71%	0,70%	0,18%	-0,08%

On remarque bien que le temps nécessaire à l'exécution du code sur le DSP varie quasi-linéairement avec l'ordre. En fait, le temps double si on double l'ordre. Le rapport Texe*freq/ordre se stabilise au fur et à mesure que l'ordre monte, d'où une erreur moins importante. Pour l'ordre 8, l'erreur max de modélisation est de 7,5% alors que à partir de l'ordre 16, l'erreur max de 3,6%. Cette erreur par rapport à SoftExplorer est due à la difficulté de trouver une loi de variation qui est linéaire.

Le tableau 6 montre l'évolution de la puissance consommée par les 3 DSPs en fonction de l'ordre et la fréquence.

Tableau 6 : *Puissance estimée et modélisée de l'application en fonction de l'ordre et la fréquence*

Ordre	4	8	16	32	64	128	256
	C6201 Modèle : P(W)=0,021* freq(Mhz)						
50Mhz							
P_SE_W	1,087	1,103	1,087	1,084	1,084	1,08	1,082
P_modele_W	1,050	1,050	1,050	1,050	1,050	1,050	1,050
Erreur %	3,404%	4,805%	3,404%	3,137%	3,137%	2,778%	2,957%
100Mhz							
P_SE_W	2,101	2,132	2,101	2,100	2,100	2,095	2,090
P_modele_W	2,1	2,1	2,1	2,1	2,1	2,1	2,1
Erreur %	0,048%	1,501%	0,048%	0%	0%	-0,239%	0,478%
200Mhz							
P_SE_W	4,23	4,189	4,13	4,12	4,12	4,114	4,112
P_modele_W	4,2	4,2	4,2	4,2	4,2	4,2	4,2
Erreur %	0,709%	-0,263%	-1,695%	-1,942%	-1,942%	-2,090%	-2,140%
	C6701 Modèle : P(W) = 0,007*freq(Mhz)						
50Mhz							
P_SE_W	0,33	0,35	0,351	0,365	0,365	0,369	0,370
P_modele_W	0,35	0,35	0,35	0,35	0,35	0,35	0,35
Erreur %	-5,11%	0%	0,28%	4,11%	4,11%	5,15%	5,41%
100Mhz							
P_SE_W	0,698	0,689	0,698	0,698	0,698	0,701	0,701
P_modele_W	0,7	0,7	0,7	0,7	0,7	0,7	0,7
Erreur %	-0,29%	-1,6%	-0,29%	-0,29%	-0,29%	0,14%	0,14%
200Mhz							
P_SE_W	1,432	1,367	1,420	1,421	1,421	1,418	1,366
P_modele_W	1,4	1,4	1,4	1,4	1,4	1,4	1,4
Erreur %	1,62%	-2,41%	1,41%	1,48%	1,48%	1,27%	-2,49%
	C5510 Modèle : P(W) = 0,0027882*freq(Mhz)						
50Mhz							
P_SE_W	0,146	0,147	0,148	0,148	0,148	0,148	0,148
P_modele_W	0,144	0,144	0,144	0,144	0,144	0,144	0,144
Erreur %	1,26%	1,39%	2,59%	2,59%	2,59%	2,59%	2,59%
100Mhz							
P_SE_W	0,268	0,270	0,271	0,271	0,271	0,272	0,272
P_modele_W	0,278	0,278	0,278	0,278	0,278	0,278	0,278
Erreur %	-3,66%	-2,90%	-2,52%	-2,52%	-2,52%	-2,14%	-2,14%
200Mhz							
P_SE_W	0,513	0,514	0,517	0,518	0,518	0,518	0,518
P_modele_W	0,514	0,514	0,514	0,514	0,514	0,514	0,514
Erreur %	-0,12%	0,07%	0,65%	0,84%	0,84%	0,84%	0,84%

Le tableau 7 récapitule les modèles énergétiques estimés du FIR pour les 3 cibles en fonction de l'ordre et la fréquence. Il est bien clair selon ces modèles que la puissance varie

surtout en fonction de la fréquence et que l'énergie qui est produit de la puissance avec le temps dépend principalement de l'ordre.

Tableau 7 : *Modèles de l'application FIR en fonction de l'ordre et la fréquence*

RQ: fréq (Mhz)	Texe (uS)	Puissance (W)	Energie(uJ)
C6201	2,006*ordre/freq	0,021*freq	0,0421*ordre
C6701	2,006*ordre/freq	0,007*freq	0,014*ordre
C5510	3,013*ordre/freq	0,002788*freq	0,831 10^{-2} *ordre

IV.3 Transformé de Fourier Rapide

La transformation de Fourier Rapide (TFR), ou encore Fast Fourier Transform (FFT), a été retenue comme deuxième application. L'étude de cette application montre qu'elle est paramétrable en fonction du nombre de points d'entrée (8, 16, 32,…,4096). Ce paramétrage permet d'expertiser l'espace des solutions possibles de l'application. (Ktari et al., 2007)

Des tests sur cette application sont élaborés avec l'environnement Code Composer afin de valider le code et d'extraire le nombre de cycles nécessaires à l'exécution du programme en fonction du nombre de points. Et afin d'estimer la consommation (puissance & énergie) du code de la FFT, on a exploité l'outil SoftExplorer (Version C) permettant de fournir ces informations. La validation des estimations est faite par mesure sur carte DSP.

Tableau 8 : *Modèle temporel et énergétique de la FFT*

DSP	N_points	Texe (uS)	Puissance (W)	Energie(uJ)
C6201	8..64	1668*N_points/freq	0,0194*freq	produit
	64..512	2120* N_points /freq		
	512..2048	2880*N_points /freq		
C6701	Même que C6201		0,0049*freq	
C5510	8..64	2469* N_points /freq	0,0025*freq	produit
	64..512	3180* N_points /freq		
	512..2048	4320* N_points /freq		

Validation par mesure

Afin de valider cette méthodologie d'estimation de consommation en temps d'exécution et en puissance, des essais sur carte ont été faits. On a exploité la carte C6701 disponible afin de tester le modèle de l'application FIR.

La mesure du courant alimentant le cœur du DSP est faite à l'aide de l'oscilloscope numérique de l'analyseur logique, Tektronix TLA 704 avec une sonde de courant.

A partir du moment où une mesure est effectuée, il faut la refaire 6 fois, de cette façon on réduit le risque d'erreur et on affine la moyenne obtenue (Loi de Student). Ces tests sont faits pour chaque ordre et fréquence.

IV.4 MPEG-2

IV.4.1 Présentation et spécification

Le principe fondamental de la vidéo est que l'œil humain a la faculté de retenir pendant un certain temps (de l'ordre d'un dixième de seconde) toute image imprimée sur la rétine. Il suffit donc de faire défiler un nombre suffisant d'images par seconde, pour que l'œil ne se rende pas compte qu'il s'agit d'images distinctes. La télévision couleur balaye l'image avec trois faisceaux, un par couleur primaire : rouge, vert et bleu. Ces signaux RVB sont ensuite combinés linéairement en un signal de luminance (Y) et deux signaux de chrominance (U et V). Concernant la vidéo numérique, elle est une suite de trames formées d'une matrice rectangulaire de pixels. Pour la vidéo numérique couleur, 8 bits sont utilisés pour chaque couleur RVB, soit donc 24 bits par pixel. (Sohn et al., 2007)

IV.4.1.1 Les standards et les formats des images

Dans une image, chaque pixel est représenté par la luminance et la chrominance. Les composantes de chrominances sont souvent sous-échantillonnées de manière à avoir une seule valeur de la composante U (resp. V) pour deux ou quatre pixels. Cette première réduction de la quantité d'informations se base sur le fait que la perception humaine est plus sensible à l'intensité de la lumière qu'à la couleur. On parle alors de format d'échantillonnage. Différents formats d'échantillonnage ont été définis tels que : les formats 4 :4 :4, 4 :2 :2 et 4 :2 :0 (Figure 38).

Si nous prenons le 4 :2 :2 comme exemple, ce format indique que chaque pixel est échantillonné en luminance, tandis qu'un pixel sur deux est échantillonné en chrominances.

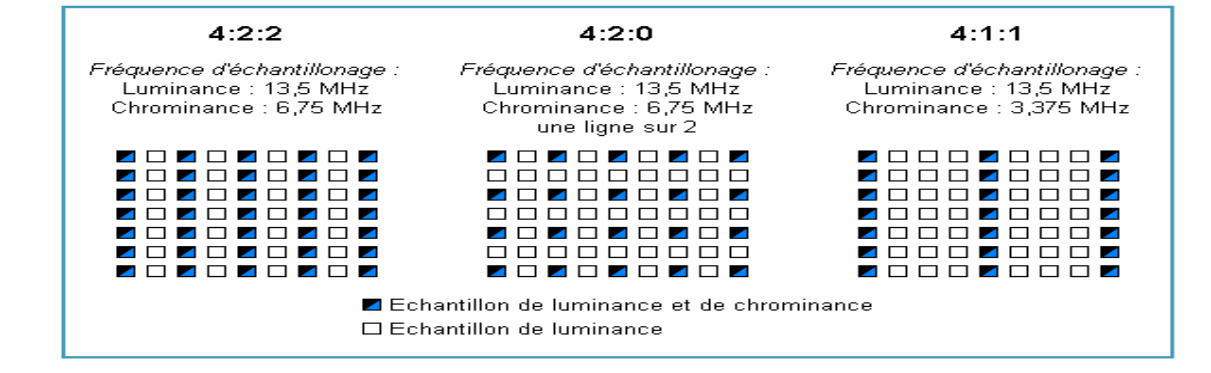


Figure 38 : *Les formats d'échantillonnage*

IV.4.1.2 L'algorithme du codage

La norme MPEG définit un ensemble d'étapes de codage qui permettent de transformer un signal vidéo (numérisé dans un format normalisé) en un flux binaire (bitstream) destiné à être stocké sur un support ou transmis dans un réseau. Le flux binaire est décrit selon une syntaxe codée d'une manière normalisée pour pouvoir être restituée par n'importe quel décodeur respectant la norme MPEG.

L'algorithme du codage définit une structure hiérarchique (figure 39). Le groupe d'images ou GOP est constitué d'une suite périodique d'images compressées. On distingue trois types d'images compressées:

-Une image de type I (ou intra) est compressée d'une manière indépendante des autres images, elles subissent donc un codage spatial ou intra,

-Une image de type P (ou prédite) est codée en utilisant une prédiction d'une image antérieure de type I ou P d'où ce qu'on appelle le codage prédictif (codage inter)

-Une image de type B (ou bidirectionnelle) codée par double prédiction (ou interpolation) qui utilise comme référence une image antérieure de type I ou P et une image future de type I ou P obtenues par un codage bidirectionnel (codage inter).

Un GOP commence par une image I, puis une suite périodique d'images P séparées par un nombre constant d'images B. La structure du GOP est alors définie par deux paramètres ; le nombre d'images du GOP (N) et la distance entre images I/P (M). A l'entrée du codage, les données vidéo sont présentées sous forme numérique où chaque pixel est codé par les trois composantes (Y, U et V).

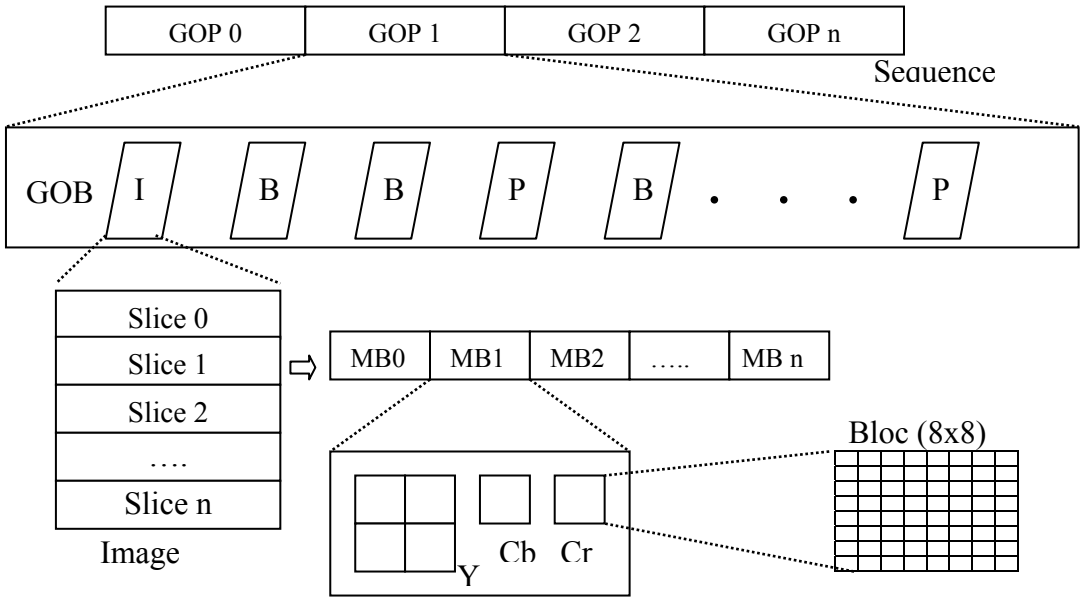


Figure 39 : *Structure hiérarchique du codage MPEG*

Le grand principe du codage vidéo MPEG est «*Ne jamais transmettre un élément d'image déjà transmis*», ce principe est réalisé par l'exploitation de deux types de redondances : la redondance spatiale, qui exprime la corrélation entre les pixels d'une même image et la redondance temporelle définissant la corrélation entre les pixels de deux images successives.

A cause de la différence des caractéristiques du signal vidéo dans les deux domaines spatial et temporel, deux techniques de codage existent pour réduire ces redondances : le codage *Intra* pour exploiter les redondances spatiales et le codage *Inter* qui vise à réduire la corrélation temporelle.

En codage Intra, la première étape consiste à effectuer une analyse de la fréquence spatiale à l'aide de la Transformée en Cosinus Discrète (DCT). Le résultat de cette transformée est une suite de coefficients décrivant l'amplitude de chaque composante fréquentielle présente dans le signal. Une transformée inverse reproduit le signal initial. La DCT n'effectue pas de compression par elle-même. Après la DCT, les coefficients subissent une quantification ce qui correspond à une première compression. Les coefficients sont ensuite scrutés (soit en zigzag soit avec un balayage alternatif) pour accroître la probabilité de commencer par les coefficients les plus significatifs (dont l'énergie est plus grande). Après le dernier coefficient non nul, un code de fin de bloc (EOB = End of Block) est généré.

En codage Inter, la réduction des redondances temporelles repose sur le principe de transmettre uniquement les différences entre les images.

A l'entrée, le codeur inter reçoit une image (c'est l'image présente), l'estimation et la compensation du mouvement s'effectuent en se référant à l'image précédente ou de référence. Une image de différence qui contient l'erreur de prédiction est ainsi produite ainsi que des vecteurs de mouvement

Cette image de différence est compressée en tant que telle par le codeur spatial (codage *intra*). Elle subit ensuite avec les vecteurs de mouvement un codage entropique. Le décodeur inverse le codage spatial et ajoute l'image de différence à l'image précédente pour obtenir l'image suivante.

IV.4.1.3 Le codeur MPEG-2

Le codeur MPEG-2 est décomposé de : la DCT, l'IDCT, la quantification, la quantification inverse, l'estimation de mouvement et la compensation de mouvement et le codage entropique. Cette décomposition est illustrée dans la figure 40 suivante :

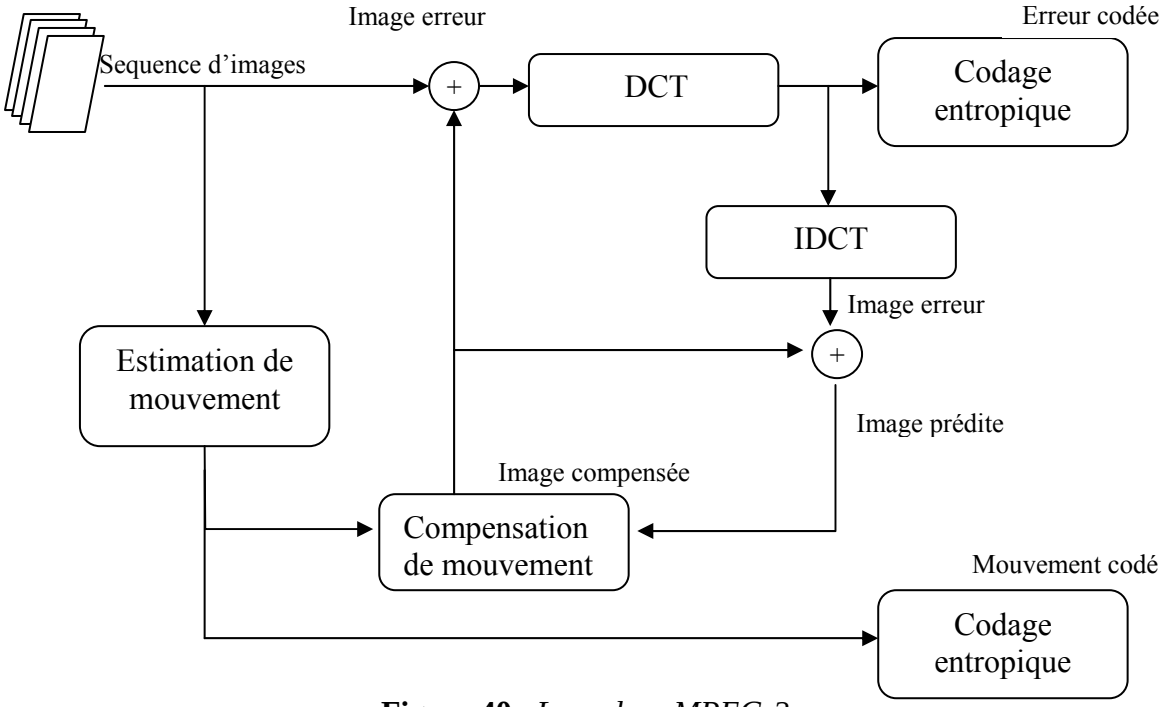


Figure 40 : *Le codeur MPEG-2*

IV.4.2 Modélisation de l'application

IV.4.2.1 Décomposition

Afin de modéliser la consommation du codeur MPEG-2, une décomposition de l'application en module est faite. Avec le graphe (Figure 40-41), les fonctions qui

correspondent à chaque bloc du codeur MPEG-2 sont distinguées. Ces fonctions seront étudiées selon leurs importances. Les fonctions principales de MPEG-2 sont les suivantes :

- Estimation et compensation de mouvement
- Prédiction
- DCT et I-DCT
- Quantification et I-Quantification
- Des traitements divers (VLC et MUX)

IV.4.2.2 Le graphe de tâches

La décomposition en tâches de l'application du décodage MPEG-2 a permis l'obtention du graphe (Figure 41). Les données échangées entre les tâches sont constituées de données, d'images et de paramètres de codage. Diverses études sur MPEG-2 ont montré que plus que 90% du traitement (temps CPU) est dans les blocs (estimation et compensation de mouvement, la DCT et la quantification). Donc ces fonctions sont les plus importantes à étudier. (Kerman et al., 2003)

IV.4.2.3 Lois de consommation

Le tableau 9 montre l'évolution de la consommation pour 3 cibles de TI des diverses tâches de MPEG.

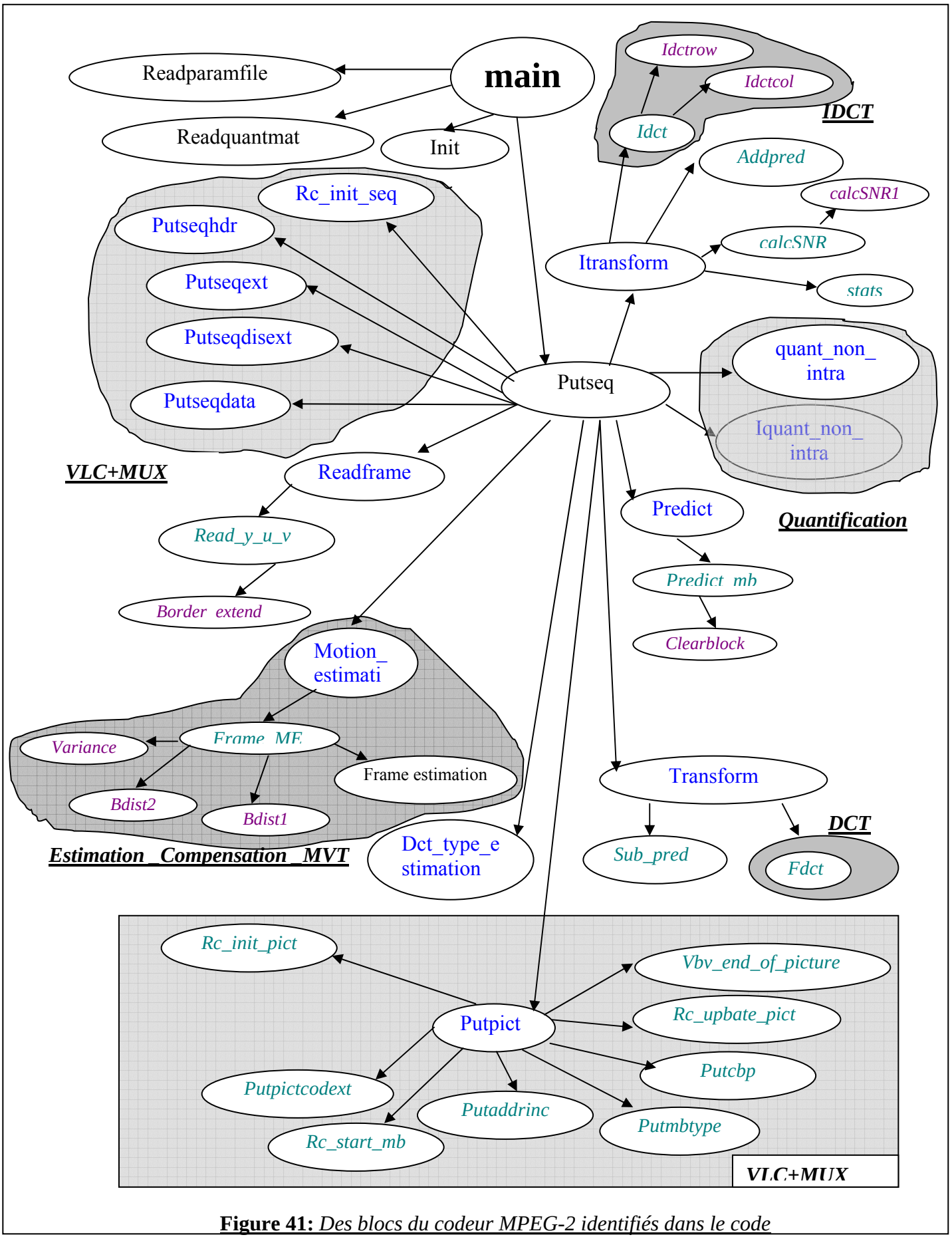


Figure 41: *Des blocs du codeur MPEG-2 identifiés dans le code*

Tableau 9 : *Tableau récapitulatif*

	C6201		
F(MHz)/ Mapped	T(mS)	P(W)	E(mJ)
DCT	1,700/F	0,0242 F	0,042
IDCT	1,107/F	0,0334 F	0,037
Quantification INTRA	6,760/F	0,0284 F	0,192
Quantification N-INTRA	4,525/F	0,0316 F	0,143
I-Quantification N-INTRA-MPEG1	2,298/F	0,0261 F	0,060
I-Quantification INTRA-MPEG1	2,420/F	0,0219 F	0,053
I-Quantification INTRA-MPEG2	4,489/F	0,0196 F	0,088
I-Quantification N-INTRA-MPEG2	4,310/F	0,0348 F	0,150
Erreur max(model % SoftExplorer)	*5,4%*	*6,4%*	*8%*

	C6701		
F(MHz) / Mapped	T(mS)	P(W)	E(mJ)
DCT	1,700/F	0,007 F-0,055	0,011
IDCT	1,107/F	0,0094 F-0,3199	0,007
Quantification INTRA	6,760/F	0,0074 F	0,050
Quantification N-INTRA	4,525/F	0,0093 F-0,365	0,042-1,651/F
I-Quantification N-INTRA-MPEG1	2,298/F	0,0079 F-0,1424	0,015
I-Quantification INTRA-MPEG1	2,420/F	0,0071 F	0,017
I-Quantification INTRA-MPEG2	4,489/F	0,0086 F- 0,220	0,038-0,987/F
I-Quantification N-INTRA-MPEG2	4,310/F	0,0096 F-0,3322	0,041-1,431/F
Erreur max(model % SoftExplorer)	*5,4%*	*4,3%*	*5%*
Erreur max(model % measure)	*13,71%*	*7,5%*	*14,94%*
	C5510		
F(MHz) / Mapped	T(mS)	P(W)	E(mJ)
DCT	3,30/F	0,00265 F	0,009
IDCT	06,25/F	0,00240 F	0,015
Quantification INTRA	14,07/F	0,00270 F	0,038
Quantification N-INTRA	9,285/F	0,00280 F	0,026
I-Quantification N-INTRA-MPEG1	5,0/F	0,00250 F	0,0125
I-Quantification INTRA-MPEG1	5.0/F	0,00250 F	0,0125
I-Quantification INTRA-MPEG2	9,230/F	0,00258 F	0,0235
I-Quantification N-INTRA-MPEG2	9,230/F	0,00260 F	0,024
Erreur max(model% SoftExplorer)	*6,1%*	*8,4%*	*10%*

Ainsi on a établi divers modèles de consommation de chaque bloc de l'application MPEG2 en fonction de la fréquence et de la cible (C55, C67 et C62). Les modèles temporels dépendent seulement de la fréquence et de la cible vu qu'on travaille sur des blocs 8*8 (taille fixe). En plus, on montre bien avec ces résultats que le C55 est un DSP faible consommation qui est très efficace pour les applications soumises à une contrainte de puissance maximale. Ces modèles d'estimation sont confrontés à des mesures sur carte DSP pour le C67. Par ailleurs, ces modèles établis pour ce DSP présentent une erreur maximale de 14,9% sur

l'énergie et 13,7% sur le temps. Pour les deux autres DSPs (C55 et C62), l'erreur présentée est par rapport à l'outil SoftExplorer.

Dans la section suivante, on présente les modèles énergétiques de l'estimation du mouvement, du codage entropique et de la prédiction.

IV.4.2.4 Estimation de mouvement

Divers types d'images sont présentes dans un GOP (Group Of Picture) : I, P, B. Pour chacun de ces types, un traitement spécifique est fait. Dans la majorité des cas, le type I occupe 8,3% du GOP, 25% pour les P et 66,6% pour les B. (Figure 42)

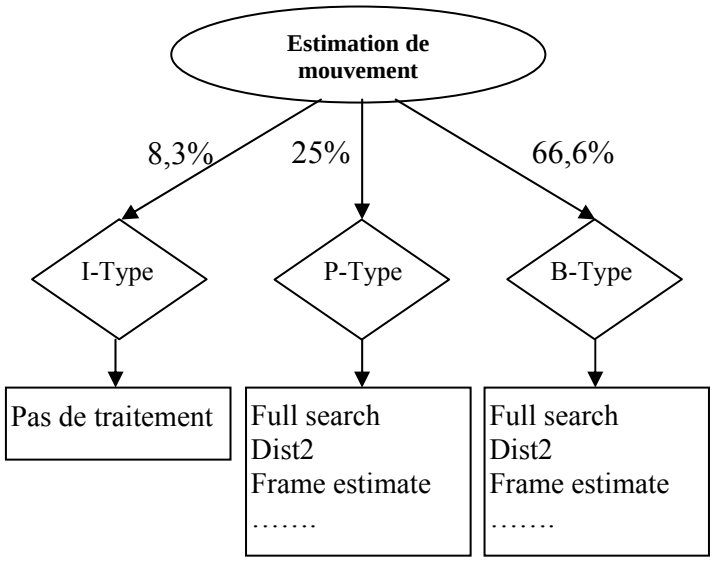
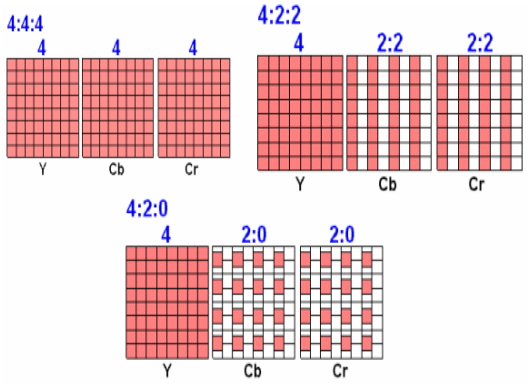


Figure 42 : *Les cas d'estimation de mouvement*

Par ailleurs le format de la chrominance joue un rôle important dans la modélisation de l'estimation de mouvement. En fait, plus on a d'information à traiter (Cb,Cr), plus ça nécessite du temps.



Chrom_format	Macroblock	Application
4:2:0 (6 blocks)	YYYYCbCr	television, divertissement.
4:2:2 (8 blocks)	YYYYCbCrCbCr	environnement de production studio, équipement d'édition professionnel
4:4:4 (12 blocks)	YYYYCbCrCbCrCb CrCbCr	Traitement et calcul graphique

Figure 43 : *Format des chrominances*

Dans le tableau suivant, on présente les divers modèles énergétiques de l'estimation de mouvement pour le C67, C55 et le C62. Les validations sur carte DSP sont faites pour le C67.

Tableau 10 : *Estimation de consommation de l'estimation de mouvement (Mode Mapped)*

	C6701 (mesures)			
F(MHz)		T(mS)	P(W)	E(uJ)
Estimation de mouvement	4:4:4	2,452*height *width /F	0,0079F	19,370*height*width
	4:2:2	1,635*height *width /F		12,916*height*width
	4:2:0	1,226*height *width /F		9,685*height*width
Erreur max (modele % mesure)		3%	4,3%	5,8%
	C6201			
F(MHz)		T(mS)	P(W)	E(uJ)
Estimation de mouvement	4:4:4	2,452*height *width /F	0,0256 F	62,77*height*width
	4:2:2	1,635*height *width /F		41,85*height*width
	4:2:0	1,226*height *width /F		31,38*height*width
Erreur max (modele % mesure)		-	-	-

	C5510			
F(MHz)		T(mS) (Code Composer)	P(W)	E(uJ)
Estimation de mouvement	4:4:4	6,924*height *width /F	0,00249 F	17,24*height*width
	4:2:2	4,616*height *width /F		11,49*height*width
	4:2:0	3,462*height *width /F		8,62*height*width
Erreur max (modele % mesure)		-	-	-

IV.4.2.5 Codage entropique

Après la quantification, la matrice de coefficients de la DCT comporte donc des termes nuls. Le codage permet de gérer plus efficacement ces coefficients. Quand une suite de valeurs identiques se présente, comme des zéros, le codage émet simplement le nombre de zéros plutôt que toute la suite de bits nuls. Le tableau 11 montre à titre d'exemple la consommation de ce codage.

Tableau 11 : *Estimation de consommation du codage entropique (8*8)*

	C6201		
F(MHz)	T(mS)	P(W)	E(mJ)
VLC	5,660/F	0,0283 F	0,160
	C6701		
VLC	5,660/F	0,0074 F	0,041
	C5510		
VLC	10,890/F	0,00259 F	0,028

IV.4.2.6 Prédiction et compensation

De même que l'estimation de mouvement, le modèle de la prédiction et la compensation dépend de la fréquence, la dimension de l'image et de la chrominance. (Tableau 12)

Tableau 12 : *Estimation de consommation de la prédiction et compensation*

	C6701			
F(MHz)		T(mS)	P(W)	E(uJ)
Predi-Comp	4:4:4	0,201*height *width /F	0,0074F	1,487*height*width
	4:2:2	0,134*height *width /F		0,991*height*width
	4:2:0	0,101*height *width /F		0,747*height*width
Erreur max (modele % mesure)		3,21%	5%	7,4%

	C6201			
F(MHz)		T(mS)	P(W)	E(uJ)
Predi-Comp	4:4:4	0,201*height *width /F	0,0295 F	5,929*height*width
	4:2:2	0,134*height *width /F		3,953*height*width
	4:2:0	0,101*height *width /F		2,979*height*width
Erreur max (modele % mesure)		-	-	-

	C5510			
F(MHz)		T(mS)	P(W)	E(uJ)
Predi-Comp	4:4:4	0,529*height *width /F	0,0029 F	1,534*height*width
	4:2:2	0,352*height *width /F		1,022*height*width
	4:2:0	0,264*height *width /F		0,767*height*width
Erreur max (modele % mesure)		-	-	-

IV.4.3 Répartition du temps CPU et de la puissance

En se basant sur les divers modèles proposés en temps et en puissance, une répartition du temps CPU et de la puissance des diverses tâches sont établies. Le bloc « estimation de mouvement » occupe la plus grande part du temps CPU. Pour une image de taille 128*128 4 :2 :2 s'exécutant sur un C6701 à 100 MHz, l'estimation de mouvement occupe 70,6% du temps. (Figure 44)

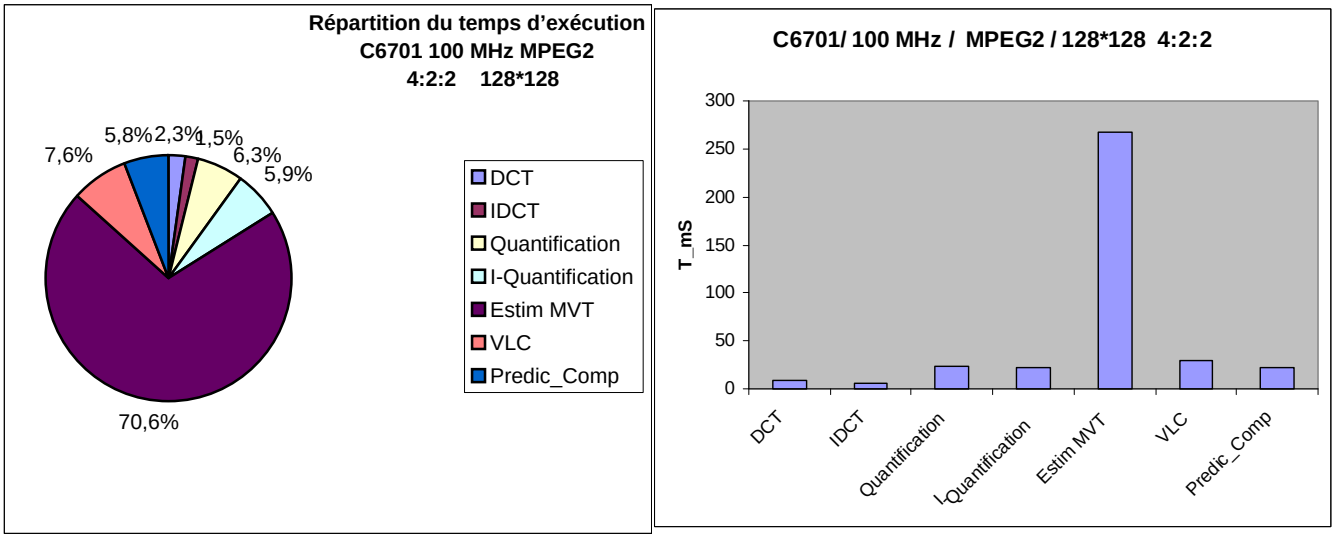


Figure 44 : *Répartition du temps CPU*

Pour la puissance, la différence de consommation entre les diverses taches varie de 10 à 20% à une fréquence donnée. (Figure 45)

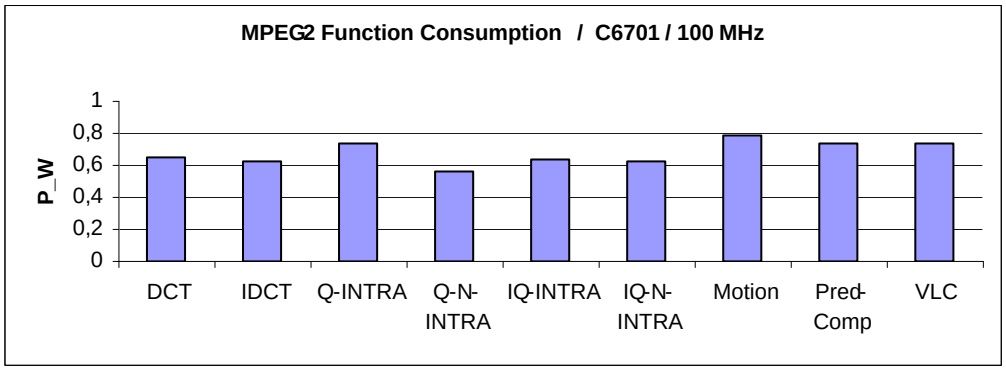


Figure 45 : *Répartition de la puissance*

Pour la consommation en énergie, l'estimation de mouvement est la tâche la plus importante. En effet, l'énergie suit la même répartition temporelle de la figure 44 puisque la puissance varie légèrement à une fréquence fixe. L'énergie consommée par cette tâche est de l'ordre 74%.

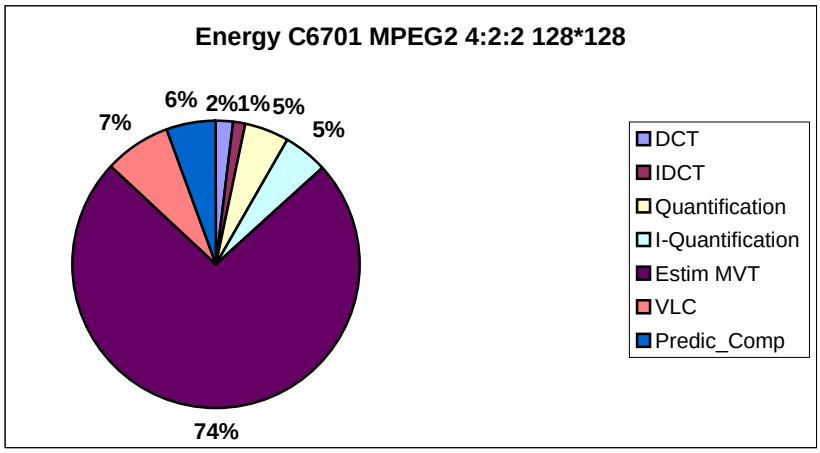


Figure 46 : *Répartition de dissipation énergétique*

IV.4.4 Modèle haut niveau MPEG2 (du pixel à l'image)

Une fois que toutes les applications nécessaires sont modélisées, le modèle de l'application MPEG2 peut être obtenu en cumulant tous les modèles (Tableau 13). Prenons le cas du traitement d'un GOP (12 I/S) avec des images 4 :2 :2 de taille 128*128. Dans chaque image, il y a 256 blocs de taille 8*8, et puisque la chrominance est généralement de type 4 :2 :2, une matrice supplémentaire de taille 128*128 pour (Cr et Cb) va être ajoutée au traitement.

Tableau 13 : *Modèle général de MPEG2*

Mapped		**C6701**		
F(MHz)		T(mS)	P(W)	E(mJ)
MPEG2	4:4:4	41,713*height* width *N_GOP /F	0,0076 F	0,317*height*width*N_GOP
	4:2:2	27,809*height* width *N_GOP /F		0,211*height*width*N_GOP
	4:2:0	20,856*height* width *N_GOP /F		0,158*height*width*N_GOP
		C6201		
F(MHz)		T(mS)	P(W)	E(mJ)
MPEG2	4:4:4	41,713*height* width *N_GOP /F	0,0286 F	1,192*height*width*N_GOP
	4:2:2	27,809*height* width *N_GOP /F		0,795*height*width*N_GOP
	4:2:0	20,856*height* width *N_GOP /F		0,596*height*width*N_GOP

	C5510			
F(MHz)		T(mS)	P(W)	E(mJ)
MPEG2	4:4:4	111,669*height* width *N_GOP /F	0,00265 F	0,295*height*width*N_GOP
	4:2:2	74,446*height* width *N_GOP /F		0,197*height*width*N_GOP
	4:2:0	55,834*height* width *N_GOP /F		0,147*height*width*N_GOP

Les modèles temporels obtenus pour l'application MPEG2 sont fonction de la dimension de l'image, du nombre de GOP, de la cible et de la fréquence. Alors que les modèles de puissance sont fonction de la fréquence du DSP.

IV.4.5 Du pixel au Standard

Le modèle de l'application MPEG2 est ainsi établi en fonction de la taille de l'image, du nombre de GOP, de la cible et de la fréquence de fonctionnement. Un modèle général de la consommation de MPEG2 sera établi en changeant la granularité de modélisation (du block (8*8) à la norme (PAL, SECAM, NTSC…)). Ces modèles seront basés sur le modèle de consommation de MPEG2. (Tableau 14)

Tableau 14 : *Modèle général de la consommation pour les standards*

Standard	Caractéristiques	T(mS)	P(W)	E(mJ)
ITU-R BT.601 NTSC	720*484, 30fps, 4:2:2	9,69 10^6 N_GOP /F	0,0076 F	7,36 10^4 N_GOP
ITU-R BT.601 PAL/SECAM	720*575, 25fps, 4 :2 :2	11,51 10^6 N_GOP /F		8,74 10^4 N_GOP
SIF NTSC	352*240, 30fps, 4 :2 :0	1,76 10^6 N_GOP /F		1,3310^4 N_GOP
SIF PAL/SECAM	352*288, 25fps, 4 :2 :0	2,11 10^6 N_GOP /F		1,59 10^4 N_GOP
				C6701 Mapped

Standard	T(mS)	P(W)	E(mJ)
ITU-R BT.601 NTSC	9,69 10^6 N_GOP /F	0,00265 F	2,5610^4 N_GOP
ITU-R BT.601 PAL/SECAM	11,51 10^6 N_GOP /F		3,04 10^4 N_GOP
SIF NTSC	1,76 10^6 N_GOP /F		0,46 10^4 N_GOP
SIF PAL/SECAM	2,11 10^6 N_GOP /F		0,55 10^4 N_GOP
			C6201 Mapped

Standard	T(mS)	P(W)	E(mJ)
ITU-R BT.601 NTSC	25,9 10^6 N_GOP /F	0,00265 F	6,89 10^4 N_GOP
ITU-R BT.601 PAL/SECAM	30,8 10^6 N_GOP /F		8,18 10^4 N_GOP
SIF NTSC	4,71 10^6 N_GOP /F		1,24 10^4 N_GOP
SIF PAL/SECAM	5,65 10^6 N_GOP /F		1,48 10^4 N_GOP
			C5510 Mapped

Les modèles ainsi établis montrent bien que chaque norme vidéo a ses propres caractéristiques, en terme de performance, de consommation. Par ailleurs, plus la taille de l'image est importante, plus son traitement est long et nécessite plus de temps et d'énergie. Le standard PAL/SECAM est le plus gourmand en calcul vu que la taille de l'image est plus grande. Par contre, la puissance dépend essentiellement de la fréquence de fonctionnement du DSP. La solution SIF PAL /SECAM paraît la moins coûteuse en terme d'énergie et temps de calcul vu la petite dimension d'images à traiter. Ce qui implique une image moins nette. Ici, le critère QoS(Qualité de Service) intervient afin de gérer le compromis qualité d'images/ performance et énergie.

IV.4.6 Conclusion

Le modèle de l'application MPEG2 est ainsi établi en fonction de la taille de l'image, du nombre de GOP, de la cible et de la fréquence de fonctionnement. Et ceci bien entendu en se basant sur des mesures et des simulations pour les différents blocs de l'application. Et par la suite extraire un modèle général de la consommation de MPEG2 en changeant la granularité de modélisation (du block (8*8) à la norme (PAL, SECAM…)). Une fois ces modèles sont établis, il est intéressant de les exploiter lors de l'exploration de l'espace des solutions logicielles. En se basant sur les résultats trouvés, et en supposant que le DSP permet de stocker l'ensemble des données en mémoire interne, on remarque bien que les temps d'exécution sur une architecture monoprocesseur seront importants. En effet, avec une telle architecture, les contraintes temporelles ne seront pas respectées pour des images de grandes tailles. D'où l'intérêt d'exploiter et explorer des architectures multiprocesseurs.

IV.5 MPEG2-Exploration de l'espace des solutions

Afin de concrétiser l'approche d'exploration à travers l'environnement développé au cours de ce travail, une deuxième étude sur MPEG 2 est menée. En fait, une telle application nécessite généralement une architecture multiprocesseur voir avec un ASIC comme accélérateur. D'ailleurs, une telle application est généralement implémentée avec d'autres fonctionnalités dans les applications mobiles actuelles (exemple : GSM). Pour cela, un espace de conception multiprocesseur est exploité, cet espace se compose d'une bibliothèque six DSPs (2*C5510, 2*C6701 et 2*C6201) communicant via un bus PCI partagé. L'objectif de cette exploration est d'extraire une solution basse consommation avec une contrainte stricte

sur le temps. C'est à l'environnement d'extraire le meilleur *mapping* architectural ainsi que le nombre de ressources (DSPs) adéquat (de deux à six ressources). Le tableau suivant montre les résultats de l'exploration basse consommation où chaque tâche est affectée à un DSP parmi les trois choisis par l'outil.

Tableau 15: *Les meilleures solutions choisies par l'environnement*

MPEG2 - 1 GOP (Group of Picture)			
Architecture	**2*C5510 & C6701**	**1*C5510 &2*C6701**	**2*C5510 & C6701**
Temps (mS)	70.61	53.36	65.67
Puissance moyenne (W)	1.13	1.61	1.22
Energie (mJ)	86.57	85.91	80.12
Mapping /(DSP) *1er C5510 : (A)* *2eme C5510 : (B)* *1er C6701 : (C)* *2eme C6701: (D)*	MPC / (C) DCT / (A) IDCT / (B) Quant / (A) IQuant / B) VLC / (B)	MPC / (C) DCT / (D) IDCT / (D) Quant / (A) IQuant / (A) VLC / (D)	MPC / (C) DCT / (B) IDCT / (B) Quant / (A) IQuant / (C) VLC / (A)

Ce tableau montre bien que l'architecture adéquate pour cette application est de préférence composée de trois DSPs. Le C62 n'est pas à priori adéquat pour cette application, en fait il n'est pas orienté pour les applications basse consommation. Par ailleurs, cette exploration montre bien que la consommation de l'application peut changer en variant le *mapping* des tâches. (Ktari et al., 2008a)

Une fois l'exploration basse consommation est faite, la question qui se pose est : Est ce que les résultats sont précis ou pas tellement. Afin de répondre à cette question, on présente dans la section suivante la technique de validation de l'approche basée sur la probabilité et les statistiques.

IV.6 Fiabilité de l'approche

Lorsqu'on parle de mesures ou de résultats des instruments de mesure, il y a plusieurs concepts qui sont souvent confondus avec d'autres comme la distinction entre la justesse et la précision. En fait, la justesse se réfère à la différence entre la mesure et la valeur réelle. Elle ne peut être discutée de façon significative que si la valeur réelle soit connue. Mais la précision se réfère à la distribution de la mesure. Par ailleurs, dans tout type d'estimation, des erreurs se présentent que ce soit à cause du processus d'estimation ou bien à cause des moyens de mesures. (Figure 47)

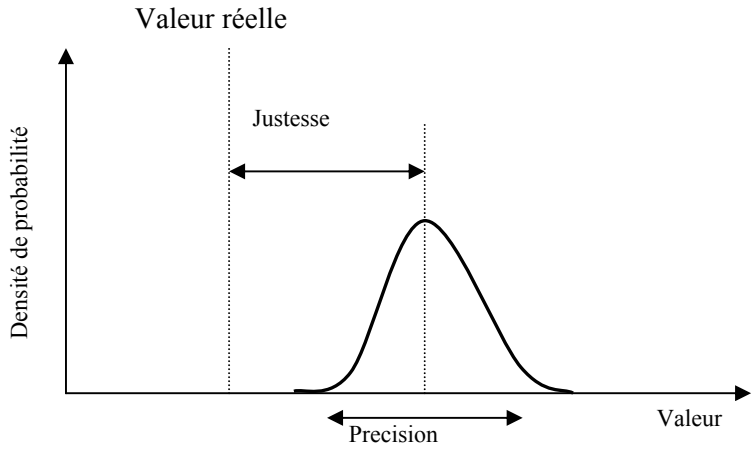


Figure 47 : *Différence entre la précision et la justesse*

Mathématiquement, lorsque plusieurs mesures d'une distribution normale sont faites, la justesse peut être estimée en calculant la moyenne m et l'écart moyen σ. (Figure 48)

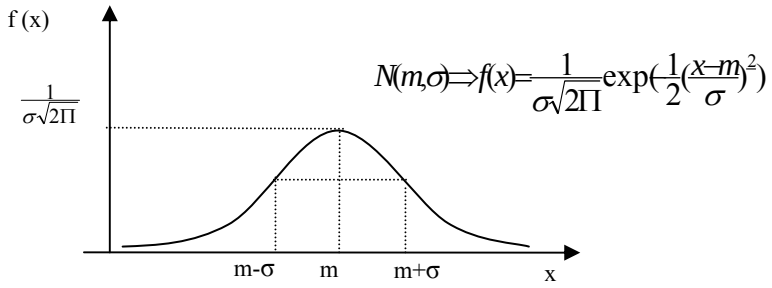


Figure 48 : *Distribution normale des mesures*

Il est parfois possible d'identifier un intervalle de telle sorte qu'on peut affirmer que cet intervalle "couvre" la valeur réelle de la mesure avec une certaine probabilité donnée P. Cet intervalle est alors appelé un intervalle de confiance. Il indique la précision d'une estimation car pour un risque α donné, l'intervalle est d'autant plus grand que la précision est faible. Par ailleurs, dans la construction d'intervalle de confiance, 3 éléments interviennent: la taille de l'échantillon, la fiabilité du résultat représentée par le coefficient de confiance et la précision du résultat représentée par l'amplitude de l'intervalle de confiance. Ceci mène à faire une étude faisant appel à la probabilité et les statistiques.

Rappelons que l'intervalle de confiance de l'espérance μ pour un coefficient de risque α est :

$$\overline{X} - t_\alpha \frac{\hat{\sigma}}{\sqrt{n}} \leq \mu \leq \overline{X} + t_\alpha \frac{\hat{\sigma}}{\sqrt{n}} \qquad (19)$$

Quelque soit la valeur de n si $X \rightarrow N(\mu , \sigma)$ et σ^2 est inconnue, avec t_α la variable de *student* avec N-1 degrés de liberté.

Il est à noter aussi que la somme de deux variables gaussiennes indépendantes est elle-même une variable gaussienne. Plus explicitement : Soient X1, X2 deux variables aléatoires indépendantes suivant respectivement les lois $N(m1,\sigma1^2)$ et $N(m2,\sigma2^2)$. Alors, la variable aléatoire X1+X2 suit la loi normale $N(m1+m2, \sigma1^2+\sigma2^2)$. Cette propriété se démontre directement (par convolution), ou indirectement (au moyen des fonctions caractéristiques). De cette façon, on a un moyen pour estimer la consommation de 2 DSPs fonctionnant ensemble.

Supposons qu'on a la densité de probabilité du temps d'exécution *Ft* d'une tâche I ainsi que la densité de probabilité de la puissance *Fp*, dans ce cas la densité de probabilité de l'énergie (Ktari et al, 2008b) sera:

$$f_e(e)=\int_{-\infty}^{+\infty}\frac{1}{|u|}f_t(u)*f_p(\frac{e}{u})du=\frac{1}{2\Pi\sigma_t\sigma_p}\int_{-\infty}^{+\infty}\frac{1}{|u|}\exp^{1/2[(\frac{u-m_t}{\sigma_t})^2+(\frac{\frac{e}{u}-m_p}{\sigma_p})^2]}du \qquad (20)$$

Avec m_t et m_p la moyenne de la distribution du temps et de la puissance respectivement. σ_t et σ_p représentent l'écart de la distribution du temps et de la puissance respectivement

Dans ce contexte, on se propose de déterminer un intervalle [m1, m2] qui a une probabilité 1-α de contenir la moyenne m des estimations de la consommation globale. Pour certaines tâches, la modélisation de la consommation de chaque processeur est faite par six mesures sur carte. Pour d'autres taches, elle est faite avec l'outil SoftExplorer dont l'erreur est ±7% avec une confiance de 95%. (Figure 49)

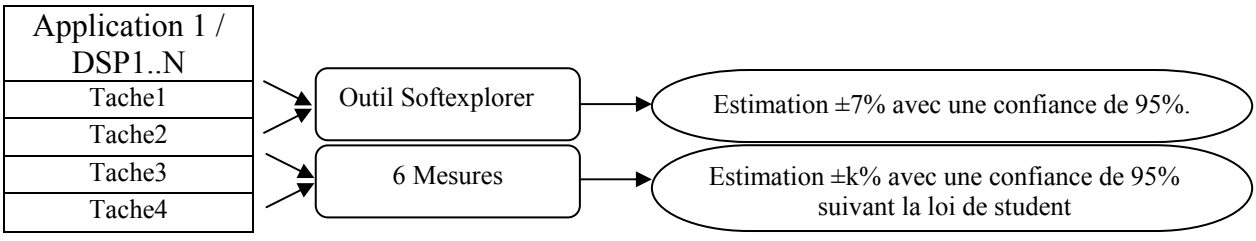


Figure 49 : *Méthodologie de validation des estimations globales*

Des résultats (Ktari et al, 2008b) dans ce sens sont dégagés sur MPEG2. Les estimations obtenues présentent une erreur de 2.3% pour un intervalle de confiance de 90%. Ainsi, à partir des estimations de performance de chaque tâche, une validation de l'approche globale d'estimation est montrée.

En fait, dans le tableau 16 on présente l'intervalle de confiance des diverses tâches importantes de MPEG2 s'exécutant sur les DSPs (le 1er : C5510, le 2eme : C6201 et le 3eme C6701).

Tableau 16: *Précision de l'estimation globale de l'énergie consommée par MPEG2*

Tâche / DSP	Estimation avec:	Intervalle de confiance (Joule)	Confiance	$\overline{x}$ et σ estimés
Motion estimation/ 1	SoftExplorer	4J±7% [4-0.28 4+0.28]	95%	$\overline{x}$ =4 σ =0.27
Prédiction / 1	SoftExplorer	2J±7% [2-0.14 2+0.14]	95%	$\overline{x}$ =2 σ =0.13
DCT/ 3	6 Mesures	[2.3-0.12 2.3+0.12] 2.3J±5.6%	95%	$\overline{x}$=2.30 σ=0.128
Quantif/ 2	6 Mesures	[4.27-0.16 4.27+0.16] 4.27J±3.8%	95%	$\overline{x}$ =4.27 σ=0.164
MPEG2				
Estimation probabiliste		[12.57-0.297 12.57+0.297] 12.57J±2.3%	90%	$\overline{x}$=12.57 σ=0.712^0.5

Avec cette méthode, on a plus besoin d'acheter et d'exploiter des cartes multiprocesseur pour estimer la consommation globale de l'application. Ceci permet de valider l'application en terme de performance et de consommation tout en réduisant le *time to market* et le coût du produit final. Ainsi, le concepteur sera guidé lors du choix de la plateforme adéquate pour son application sans avoir recours à faire des manipulations sur carte pour connaître l'avantage d'une telle solution par rapport à une autre. Ce qui simplifie le problème surtout pour les nouvelles cartes dont l'alimentation des cœurs des DSPs n'est pas toujours accessible pour faire des mesures.

IV.7 Conclusion

Dans ce chapitre, une étude sur la consommation de diverses applications est menée afin de proposer des lois de performance de haut niveau caractérisant l'influence des paramètres algorithmiques et architecturales sur l'énergie et sur la performance. Ces modèles permettent d'élever le niveau de prise en compte de la consommation afin de permettre au concepteur de pouvoir estimer la consommation du système au début du flot de conception. Le concepteur peut ainsi paramétrer et dimensionner son système de façon à respecter les diverses contraintes. Ce qui évite ainsi les retours en arrière intempestifs durant la conception et permet de réduire le temps et le coût de développement.

Et afin de réduire la complexité de l'espace des solutions, il est primordial de développer dans les premières étapes du flot de conception des méthodes de partitionnement et d'exploration pour compenser la croissance de la complexité des applications tout en intégrant la consommation comme critère lors de la conception de ces systèmes autonomes. Pour cela, une approche d'exploration basse consommation est présentée à travers un environnement de conception. Cette approche est expérimentée sur l'application MPEG2, puis validée à travers une étude probabiliste.

Conclusions et perspectives

Le problème de la consommation d'énergie est devenu prédominant lors de la conception des systèmes embarqués actuels. Dans ce contexte, ce travail étudie les méthodes de conception basse consommation des systèmes. Il est à noter que l'un des problèmes de la conception système est le partitionnement d'applications qui requiert l'utilisation de méthodes complexes. En effet, le partitionnement sous contrainte de temps, basé sur un algorithme d'ordonnancement avec un objectif de minimisation de la consommation est un problème NP-difficile.

On a étudié dans ce rapport les défis imposés par les architectures soumises à diverses contraintes en particulier la consommation. Une méthodologie d'exploration basse consommation est proposée ainsi qu'un environnement de conception. L'approche proposée dans ce travail a permis de spécifier l'application et les contraintes à plusieurs niveaux de granularités. Cette approche a permis aussi d'établir des modèles paramétriques de l'énergie qui englobent la consommation de tout le système logiciel /matériel/communication. En fait, les paramètres de l'application ainsi ceux de l'architecture seront considérés dans les modèles de performance afin d'avoir des modèles assez riches. Et finalement, l'approche a permis d'explorer efficacement l'espace des solutions basse consommation afin d'être capable de choisir d'une façon efficace la solution architecturale adéquate où le nombre de ressources à exploiter n'est pas connu à priori. En effet, le concepteur peut être confronté au problème du choix du nombre de ressources à exploiter lors de la conception de son produit. Par ailleurs à travers cette approche, le concepteur pourra paramétrer et dimensionner son système de façon à respecter les diverses contraintes au début du flot de conception.

Perspectives

Malgré les gains obtenus avec les techniques développées, certaines améliorations peuvent être apportées à ce travail, telle que la prise en compte dans l'estimation et l'optimisation de la consommation des systèmes sur puce aussi bien les réseaux sur puce. Cette prise en compte complexe nécessite une étude préalable.

Comme perspectives ouvertes par ce travail, citons l'intérêt d'étudier la gestion de la puissance possible par les systèmes d'exploitation qui sont amenés à prendre de plus en plus d'importance dans le domaine de la gestion de l'énergie. Ainsi, ils se situent à un niveau stratégique de l'architecture pour gérer finement la puissance durant les ordonnancements des

tâches. De plus, les architectures hétérogènes nécessitent parfois de répartir des systèmes d'exploitation temps réel sur plusieurs unités. Là encore, il est nécessaire d'étudier des approches de gestion/réduction de la consommation dans une telle architecture logicielle/matérielle. Par ailleurs, la contrainte de puissance maximale supporté par l'architecture cible n'est pas encore considérée lors de l'exploration de l'espace, elle sera intégrée dans la nouvelle version de l'outil. Un raffinement des paramètres de l'heuristique est aussi envisageable, ceci permettra d'éviter efficacement les minimums locaux et de converger vers une solution assez performante.

Références

(Abdennour, 2004) A. Abdennour, « Outil d'analyse et de partitionnement/ordonnancement pour les systèmes temps réel embarqués », Thèse de Doctorat, Juin 2004, UBS, France.

(Abdennour et al., 2002) A. Azzedine, J-P Diguet, J.L Pillippe, "Large exploration for HW/SW partitioning of multirate and aperiodicreal-time systems", CODES 2002. Proceedings of the Tenth International Symposium on on Hardware/Software Codesign, 2002 Page(s):85 – 90, Estes Park, Colorado, USA,

(Auguin *et al*., 2001) M. Auguin, L. Capella, F. Cuesta, E. Gresset. «Codef : a system level design space exploration tool». In International Conference on Acoustics, Speech and Signal Processing (ICASSP), Salt Lake City, USA, May 2001.

(Baghdadi et al, 2002) A.Baghdadi, N. Zergainoh, WO. Cesario, A. A. Jerraya, «Combining a Performance Estimation Methodology with a Hardware/Software Codesign Flow Supporting Multiprocessor Systems», IEEE transaction on software engineering, vol. 28, no. 9 septembre 2002

(Bandyopadhyay et al., 2004) T. Bandyopadhyay, B. Susnata, B. Swapan, « Multi Processor Scheduling Algorithm for tasks with Precedence Relation », TENCON 2004, proceedings analog and digital techniques in electrical engineering, November 2004, Thailand.

(Beak et al., 2004) W. Baek, Y. Kim, J. Kim, «ePRO: A Tool for Energy and Performance Profiling for Embedded Applications», in Proc. of International SoC Design Conference (ISOCC'04), pp. 372-375, Seoul,Korea, October 25-26, 2004

(Benoit et al., 2004) P.Benoit, G.Cambon, M.Robert, G.Sassatelli «Architecture reconfigurables, les processeurs du futur», La douzième session des journées des doctorants DOCTISS'2004, Mars 2004, Montpellier.

(Bharat et al. 1999) P. Bharat Dave, G. Lakshminarayana, K. Jha, « COSYN: Hardware–Software Co-Synthesis of Heterogeneous Distributed Embedded Systems ». IEEE Transaction (VLSI) Systems, VOL. 7, NO. 1, MARCH 1999

(Bianco *et al.,* 1998) L. Bianco, M. Auguin, G. Gogniat, A. Pegatoquet. «A path based partionning algorithm for time constrained embedded systems design», In International Symposium on Hardware/Software Codesign (CODES), Seattle,USA, March 1998.

(Brandolese et al., 2000) C. Brandolese, W. Fornacieri, F. Salice, D. Sciuto « An Instruction Level Functionnality-Based Energy Estimation Model for 32 bits Microprocessor», In Proceeding of DAC 2000, p346.

(Bossuet *et al*., 2003) L. Bossuet, W. Burleson, G. Gogniat, V. Anand, A. Laffely, J.L. Philippe. «Targeting tiled architectures in design exploration», In 10th Reconfigurable Architectures Workshop (RAW), Nice, France, April 2003.

(Dick et al., 1998) R. P. Dick, N. K. Jha « MOGAC: A Multiobjective Genetic Algorithm for Hardware-Software Co-Synthesis of Distributed Embedded Systems »IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (1998)

(Elleouet et al., 2006) D. Elleouet, Y. Savary, N. Julien,D. Houzet «A FPGA Power Aware Design Flow» International Workshop on Power and Timing Modeling, Optimization and Simulation, Patmos06, France

(Emmanuel et al., 2001) G. Emmanuel, A. Choquet-Geniet, «Ordonnancement de tâches temps réel en environnement multiprocesseur à l'aide de réseaux de Petri» Real-Time Systems, RTS'2001, March 6-8 2001, Paris.

(Fei et al., 2003) Y.Fei, S. Ravi, A.Raghunathan, N.Jha, «Energy estimation for extensible processors», In Proceeding of the Design Automation and Test in Europe (DATE 03),Germany.

(Ghali et al., 2004) K. Ghali, O. Hammami, I. Hermann, « Multiobjective Design of Embedded Processors on FPGA Platforms », 24th International Conference on Distributed Computing Systems Workshops ICDCS, Mars 2004, Japan.

(Garcia *et al*., 2005) A. Garcia, L. Gonzales, R. Felix, «Power consumption management on FPGAs» 15th International Conference on Electronics, Communication and Computers, 2005, Mexique.

(Guitton et al., 2003) P. Guitton-Ouhamou, C. Belleudu, M. Auguin, « Energy Optimization in Hw/Sw Tool : Design of Low Power Architecture System ». IEEE International Workshop on System on Ship for Real-Time Systems - IWSOC'2003, Calgary, Canada, Juin 2003.

(Havinga et al., 2000) J.M. Havinga, J.M. Smit «Design techniques for low power systems » Journal of Systems Architectures, Vol 46:1, 2000.

(Heikkinen et al., 2002) J. Heikkinen, J. Sertamo, T. Rautiainen, and J. Takala, "Design of Transport Triggered Architecture Processor for Discrete Cosine Transform," 15ème Annual IEEE International ASIC/SOC Conference, Sept. 2002.

(ITRS, 2004) ITRS, «International Technology Roadmap for Semiconductors», Edition 2004, http://public.itrs.net.

(Julien et al., 2004) N. Julien « Une approche logicielle de la consommation dans les systèmes embarqués », Premier Congrès International de Signaux, Circuits & Systèmes Mars 2004, Tunisie.

(Kappagantula et al., 2003) V. Kappagantula, N. Mahapatra, « PAP: PowerAware Partitioning of Reconfigurable Systems». HPCA9/SSRS '03 Anaheim, California USA

(Ktari et al., 2005) J. Ktari, J. Laurent, M. Abid, N. Julien, «Estimation de la consommation logicielle dans un système embarqué : Etude de cas », in Proc. FTFC'2005, pp 55-59, 18-19 Mai 2005, France.

(Ktari et al, 2007) J. Ktari, M. Abid, «System Level Power and Energy Modeling for Signal Processing Applications», 2 nd IEEE International Design and Test Workshop IDT, Egypt, December 16-18, 2007.

(Ktari et al, 2008a) J. Ktari, M. Abid, «A Low Power Design Methodology Based on High Level Models», ESA'08 - The 2008 International Conference on Embedded Systems and Applications, Las Vegas, Nevada, USA (July 2008)

(Ktari et al, 2008b) J. Ktari, M. Abid,« Accuracy of Low Power Estimation for Embedded Application», FTFC'08, 7ème journées d'études Faible Tension Faible Consommation, Mai 2008, Belgique.

(Kerman et al., 2003) Y. Kerman, J. Lu, l. Shipeng, «Practical real time video codec for mobile device» IEEE-International Conference on Multimedia and Expo ICME03, Vol.I, USA,

(Lacomme et al., 2003) P. Lacomme, C. Prins, M. Sevaux, « Algorithmes de graphes» Edition Eyrolles ISBN 2-212-11385-4, 2003.

(Laurent, 2002) J. Laurent, «Estimation de la consommation dans la conception système des applications embarquées temps réel», Doctorat de l'UBS, Décembre 2002, France.

(Laurent et al., 2007) J. Laurent, E.Senn, N Julien « Méthodes et outils d'estimation de la consommation de code embarqué sur processeur », Revue Traitement et Systèmes Informatique, volume 26 n°5, Mai 2007

(Li et al., 2000) E. A. Lee. «What's ahead for embedded software ? » IEEE Computer Magazine, pages 18_26, September 2000.

(Li et al., 2003) T. Li, L K. John, «Run-time Modeling and Estimation of Operating System Power Consumption », Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), 2003, USA.

(Marteil et al., 2006) F. Marteil. «High Level Memory Hierarchy Optimisation». In EDAA Ph.D. Forum at DATE, Munich, April 2006.

(Maalej, 2007) I. Maalej, «Exploration de haut niveau des architectures multiprocesseurs : analyse et métriques», Doctorat de l'UBS, Octobre 2007, France.

(Minh et al., 2003) D Q. Minh, L. bengtsson, PL. Edefors « DSP-PP: A simulator /estimator of power consumption and performance for parallel DSP architectures », Proc. 21st IASTED International Conference Applied Informatics 2003, Austria.

(Nikolaidis et al., 2005) S.Nikolaidis, A.Chatzigeorgiou, T.Laopoulos, «Developing an environment for embedded software energy estimation», Elsevier Computer Standards & Interfaces, 2005.

(Noseda, 2002) Noseda A, «Etude du partionnement et de l'ordonnancement de tâches cycliques sur une architecture distribuée composée d'un terminal mobile 3G et d'un serveur de calcul» Mastère à Ecole Supérieure d'Ingénieurs de Marseille, 2002.

(Rabaey et al.,1996) J.M. Rabaey, M. Pedram « Low Power design Methodologies »Edition Kluwer Academic Publishers London 1996.

(Russell et al.,1998) J. Russell, M. Jacome, «Software power estimation and optimization for high-performance 32-bit embedded processors» in Proc. Int. Conf. Computer Design, pages 328-333, October 1998.

(Sequence, 2005) www.sequencedesign.com

(Shin et al., 2002) D. Shin, H. Shim, Y. Joo, «Energy-Monitoring Tool for Low-Power Embedded Programs», IEEE Design and Test off Computers, Juillet 2002, pp. 7 – 17.

(Stanley et al., 2004) J. Stanley, W. Nebel, Lkabous, «low power analysis using ORINOCO», Electronic Design Processes 2004, Avril 2004, Monterey, USA.

(Sciuto et al., 2002) D.Sciuto, F.Salice, L.Pomante, W.Fornaciari. «Metrics for design space exploration of heterogeneous multiprocessor embedded systems», In International Symposium on Hardware/Software Codesign (CODES), Estes Park, USA, May 2002.

(Sohn et al., 2007) J. Sohn, H. Kim, J. Jeong, E. Jeong, S. Lee, «A low power multimedia SoC with fully programmable 3D graphics and MPEG4/H.264/JPEG for mobile devices», ISLPED 2007, 238 – 243, USA

(Tiwari et al.,1996) V. Tiwari, S. Malik, A. Wolfe « Instruction Level Power Analysis and Optimization of Software », Journal of VLSI Signal Processing 1996 Kluwer Academic Publishers, Bouston.

(Tmar et al., 2007) H. Tmar, J-P. Diguet, A. Azzedine, J-L. Philippe, M. Abid «RTDT : a Static QoS Manager, RT Scheduling, HW/SW Partitioning CAD Tool» Microelectronics Journal (2007)

(Xilinx, 2007) www.xilinx.com

Publications scientifiques personnelles

Articles:

J. Ktari, M. Abid,« A Low Power Design Space Exploration Methodology Based on High Level Models and Confidence Intervals», accepted for publication in JOLPE - ASP Journal of Low Power Electronics, 2009.

J. Ktari, M. Abid, N. Julien, J. Laurent, « Power Consumption and Performance's Library on DSPs: Case Study MPEG2», Journal of Computer Science 3(3): 168-173, 2007, USA.

Communications:

J. Ktari, M. Abid, «A Low Power Design Methodology Based on High Level Models», ESA'08 - The 2008 International Conference on Embedded Systems and Applications, July 2008, Las Vegas, Nevada, USA

J. Ktari, M. Abid,« Accuracy of Low Power Estimation for Embedded Application», FTFC'08, 7ème journées d'études Faible Tension Faible Consommation, Mai 2008, Belgique

J. Ktari, M. Abid, «System Level Power and Energy Modeling for Signal Processing Applications» , 2 nd IEEE International Design and Test Workshop IDT, Egypt, December 16-18, 2007

F.B. Arfia, **J. Ktari**, M. Bousselmi, M. Abid, «Estimation de la consommation au niveau algorithmique : application de la DCT et l'intra prédiction dans la norme de compression vidéo H.264», ISIVC'06, CD-ROM 3rd International Symposium on Image/Video Communications over fixed and mobile networks, Septembre 2006, Tunisie.

J. Ktari, J.Laurent, M.Abid, N.Julien, « Estimation de la consommation logicielle dans un système embarqué : Etude de cas », FTFC'2005, 5ème journées d'études Faible Tension Faible Consommation, Mai 2005, France.

République Tunisienne
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique
et de la Technologie

Université de Sfax
École Nationale d'Ingénieurs de Sfax

Cycle de Formation Doctorale
dans la Discipline Génie
Electrique
*Ingénierie des Systèmes
Informatiques*

Thèse de *DOCTORAT*
N° d'ordre: 2009– 96

APPROCHE ET ENVIRONNEMENT D'EXPLORATION ARCHITECTURALE BASSE CONSOMMATION

Jalel KTARI

الخلاصة: التحسين من الاستهلاك الطاقي أصبح موضوع بحث يعالج على مستويات عدة. نحن مهتمون في هذا العمل لاستكشاف الإلكترونيات المنخفضة الطاقة. نعتبر هذه المشكلة في مجملها على النظام برمته. نقدم نماذج عالية الأداء ونقترح تقنية تسمى منخفضة الإستهلاك. ويسمح هذا النهج للنظر في عدد من العوامل الخوارزمية و التقنية على الاستهلاك. ونقترح وضع نموذج للاستدلال على الأداء الكلي للنظام لاستخدامها عند البحث. ويسمح هذا الكشف عن الحلول على عدة مستويات بإختيار المستوى الذي يضمن دقة وسرعة الحل.

Résumé : L'optimisation de la consommation est devenue un sujet de recherche traité à plusieurs niveaux. Nous nous intéressons dans ce travail à l'exploration d'architecture basse consommation afin de déduire celle qui répond au mieux aux besoins. Pour cela divers travaux focalisent sur un aspect particulier à savoir l'estimation de la consommation, modèle de performance, technique d'exploration. En plus plusieurs travaux traitent l'exploration d'architecture au niveau composant. Nous considérons ce problème dans sa globalité au niveau de tout le système. Nous proposons des modèles de performances riches et nous proposons une technique d'exploration dite basse consommation. Cette approche permet de considérer un certain nombre de paramètres algorithmiques et architecturaux sur la consommation. Un modèle complet est proposé afin de déduire les performances globales du système qui seront utilisées lors de l'exploration à travers une technique basée sur le recuit simulé. Et afin de valider l'approche, une étude probabiliste est faite afin de montrer la fiabilité des résultats trouvés.

Abstract: Power consumption is nowadays a critical design constraint for circuits and systems. To guide efficiently early choices in the design flow, high-level estimations must be available. In order to address the different abstraction levels and the various targets, a global methodology is proposed here to elaborate suitable models. In this work we are interested in exploring low consumption architectures in order to deduce that which meet(s) the constraints most. For this aim, several low power methodologies were established. They treat the energy consumption optimization problem at several levels especially on specific components like the hardware, or on the software, or on the communication or on the memory separately, and seldom on the whole system. However, as target architectures become complex, a global methodology that offers more efficient low power exploration become necessary. In fact, we propose a low power methodology based on rich performances models as well as a low power exploration technique. A complete model is proposed in order to deduce the total performances of the system according to the architecture and the application parameters.

المفاتيح: الاستهلاك الطاقي،تصميم، معالج،بحث مجال إلكتروني

Mots clés: Faible consommation, Exploration de haut niveau, Co-design, FLPA, heuristique.

Key-words : Low power design, Space exploration co-design, High level models, Accuracy