1. Généralités

Un processus de développement typique pourrait être celui de la *Figure 4.1.* La souplesse du développement d'applications à base de DSP est un avantage important en termes de temps, de facilité, de fiabilité, et donc de coût. Comme le suggère la *Figure4.1*, créer une application DSP, c'est mener de front deux études distinctes.

1.1. La partie matérielle :

Elle inclue la mise en oeuvre du DSP lui-même, mais aussi la création d'une chaîne d'acquisition et/ou de restitution du signal (parfois des signaux) à traiter. Les moyens de transformation du signal analogique vers le domaine numérique s'appuient eux aussi sur des circuits spécialisés (AIC, CODEC, CNA, CAN...) Le choix des performances à obtenir et des moyens pour réaliser la chaîne d'acquisition et/ou restitution du signal est primordial pour exploiter au mieux les capacités d'un DSP. L'objectif est de rendre l'application finale homogène, ergonomique, et ayant un coût de fabrication industriel approprié.

1 .2. La partie logicielle :

Elle s'appuie sur des outils classiques adaptés aux spécificités des DSP. L'approche est différente de celle utilisée pour la partie matérielle, car il est toujours possible de recommencer autant de fois que nécessaire pour arriver au résultat.

Seul un temps de développement trop limité ou une mauvaise évaluation de départ des capacités du DSP cible peut créer des problèmes de développements. La conception logicielle n'en est pas plus facile pour autant, car le programme réalise l'essentiel du traitement du signal.

Le rôle du DSP ne se limite pas forcément au seul traitement numérique du signal, un DSP peut assurer les mêmes fonctions qu'un microprocesseur « normal », et donc être le coeur du système informatique de l'application. Ainsi,

le cas échéant, un DSP peut exécuter à la fois un système d'exploitation temps réel et assurer les fonctions de traitement numérique du signal.

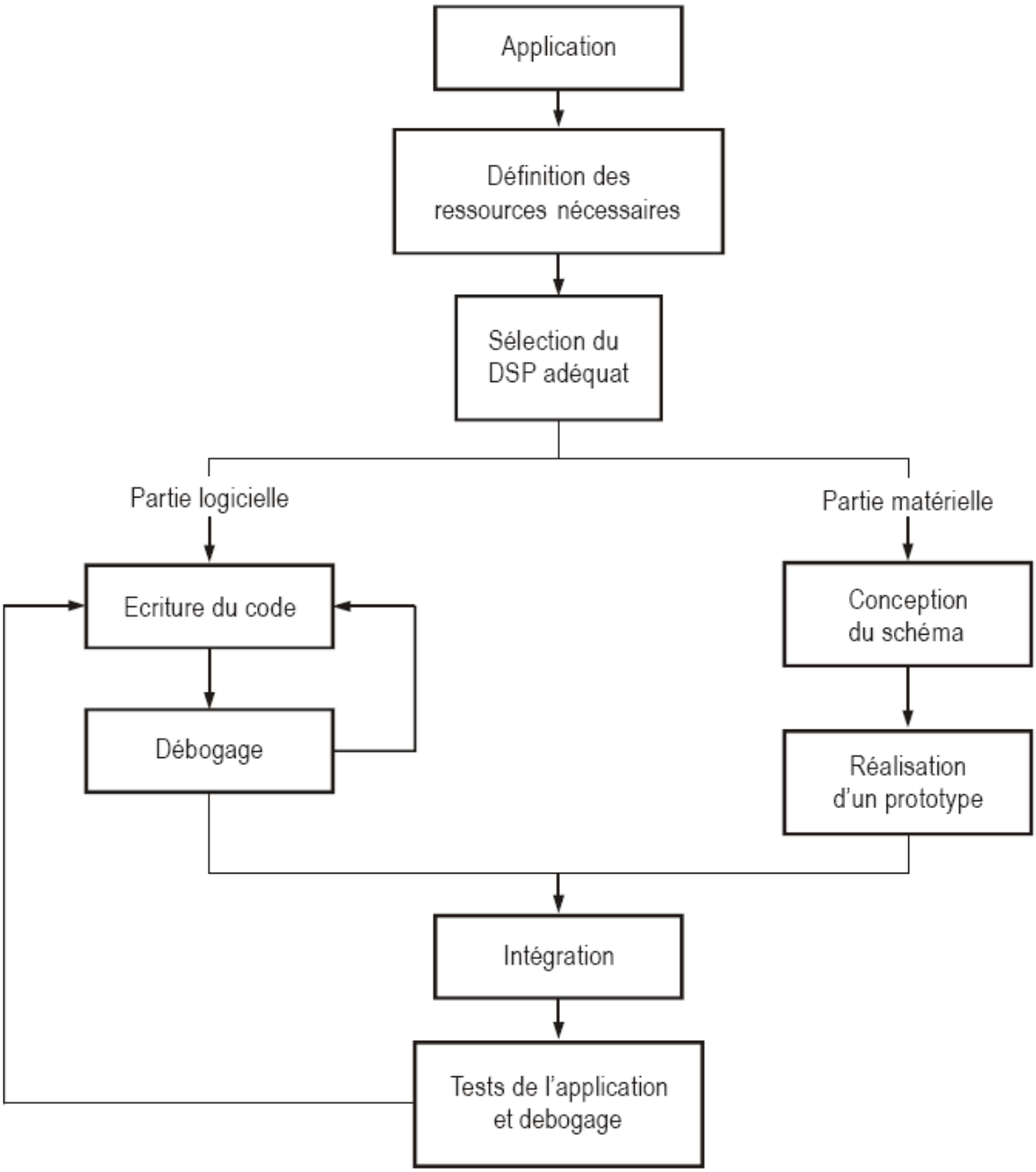

Figure 4.1 *: Exemple d'un processus de développement typique*

2. Définition des ressources nécessaires

Cette phase doit permettre d'évaluer les besoins nécessaires à la mise en oeuvre du système de traitement numérique du signal voulu. Elle consiste notamment à définir les spécifications de la chaîne d'acquisition et de restitution du signal, telles que :

- la résolution (nombre de bits nécessaires pour quantifier le signal),
- la vitesse d'échantillonnage (critère de Shannon),
- la technologie et donc le type de convertisseurs utilisé,
- les filtres anti-repliements s'ils ne sont pas intégrés dans les convertisseurs.

Elle permet plus généralement de cerner tous les besoins du système numérique, tels que par exemple la consommation de courant et l'autonomie pour une application portable, ou au contraire les bus nécessaires à l'intégration de l'application dans un système hôte.

3. La sélection du DSP le plus adapté

La sélection d'un DSP se base avant tout sur la puissance de traitement nécessaire, et sur le résultat de benchmarks réalisant des fonctions représentatives des traitements à réaliser.

Toutefois, la performance du DSP n'est pas le seul critère à prendre en compte, il faut également tenir compte des impératifs suivants :

- ✓ Le type de DSP à utiliser (virgule fixe ou flottante) en fonction du domaine d'application.
- ✓ Les ressources mémoires utilisés, car s'il faut par exemple exécuter très rapidement une FFT 1024 points, un DSP intégrant plus de 2048 mots de mémoire vive statique peut être nécessaire.
- ✓ Les besoins d'un ou de plusieurs timers internes, de ports série synchrones ou asynchrone, etc.

- ✓ La nécessité éventuelle d'exécuter un système temps réel, qui s'avérera plus facile à implanter sur certains DSP.
- ✓ Le coût du DSP, son rapport « performance/prix » en fonction du volume de production envisagé.
- ✓ La pérennité du produit, c'est-à-dire l'évolution prévue par le fabricant (roadmap).

D'autres éléments non négligeables interviennent dans le choix d'un DSP, il s'agit des moyens disponibles pour mener le développement en un temps donné, comme :

- ✓ La qualité de la documentation (de préférence claire et abondante).
- ✓ La disponibilité de notes d'applications, d'un support technique.
- ✓ La qualité du système de développement utilisé.
- ✓ La possibilité d'utiliser un langage de haut niveau (Langage C).
- ✓ La présence de librairies (du constructeur ou de tierces parties).
- ✓ La possibilité de réaliser facilement des prototypes et à faible coût.

Le choix n'est pas toujours simple et certains critères peuvent être contradictoires, certaines règles de choix se dégagent quand même. Ainsi pour des applications destinées à faire un fort volume de production, le critère déterminant est sans conteste le prix du DSP. Pour des applications à faible volume de production, le prix du DSP importe peu, le critère est alors la facilité de développement.

Dans tous les cas, la présence d'un bon support technique est un facteur à ne pas négliger, car un DSP est quand même plus complexe à mettre en oeuvre qu'un microprocesseur classique.

4. Structure matérielle de développement

Un environnement (ou système) de développement pour DSP peut être scindé en deux parties principales:

- ✓ Un environnement de développement pour créer et mettre en forme le logiciel de l'application (création du source, utilisation des bibliothèques, assemblage).
- ✓ Un environnement de développement utilisant des outils spécifiques pour tester et déboguer le logiciel de l'application (simulateur, module d'évaluation, émulateur).

4.1 Le simulateur

Le simulateur est un programme particulier exécuté par un PC ou une station de travail. Son rôle consiste à simuler le plus exactement possible le fonctionnement du DSP cible. L'interface utilisateur du simulateur permet de consulter les mémoires, tous les registres internes du DSP, ses entrées/sorties, etc. Le simulateur exécute chaque instruction DSP comme le ferai le DSP lui-même, et en répercute les résultats dans les mémoires et les registres simulés.

L'avantage de ce moyen de développement est qu'il ne nécessite pas la mise en oeuvre du DSP cible, le test d'un module logiciel peut donc se faire rapidement dès sa création. Comme l'indique la *figure 4.2* , l'écriture d'un logiciel DSP est un processus très itératif, la disponibilité d'un simulateur est donc toujours appréciable eu égard au gain de temps de développement qu'il génère.

L'inconvénient est que le logiciel DSP en cours de développement n'est pas du tout exécuté en temps réel. Les opérations d'entrées/sorties sont simulées en utilisant des fichiers sur le disque dur du PC. Le simulateur devient vite limitatif lorsqu'il s'agit de tester le code en charge des opérations d'entrés/sorties.

4.2 Le module d'évaluation

Le module d'évaluation se présente sous la forme d'une carte électronique incorporant le DSP cible et le minimum des ressources

nécessaires à sa mise en oeuvre, telles que des mémoires externes, un AIC, le cas échéant une liaison série RS232, et une alimentation. La partie matérielle est figée et n'est pas (ou alors très peu) évolutive. Un module d'évaluation s'utilise donc généralement « tel quel », et est surtout utile quand ses caractéristiques recouvrent celles de l'application à développer. Le module est piloté à partir d'un logiciel adéquat exécuté par un PC.

Figure 4.2 : module d'évaluation *TMS320C6416 (1 Ghz) DSP Starter Kit (DSK)*

La communication avec le module d'évaluation s'effectue au travers d'une liaison série (USB ou RS232) s'il s'agit d'un modèle autonome, ou à via un bus du PC s'il s'agit d'une carte à enficher. Le programme à tester est téléchargé dans le module pour être exécuté par le DSP. Comme le simulateur, le module d'évaluation permet de consulter la mémoire et les registres du DSP à volonté. Il permet également de poser des points d'arrêts simples aux endroits stratégiques du code à déboguer.

Un problème posé par les modules d'évaluations est la non disponibilité de l'ensemble des ressources du DSP. En effet, en plus du code à tester, le DSP exécute également un mini moniteur de débogage, chargé de communiquer

avec le PC et d'interpréter des commandes de bases. Une partie des interruptions et de la mémoire du DSP est donc attribué au moniteur de débogage.

Un module d'évaluation n'en reste pas moins un outil de développement approprié pour tester des parties de codes en temps réel. Il est disponible immédiatement, ce qui n'est pas toujours le cas du prototype de l'application à développer, et son faible prix en fait souvent un outil d'apprentissage apprécié.

4.3 L'émulateur temps réel

L'émulateur temps réel est l'outil privilégié pour développer des applications DSP. C'est l'outil le plus souple et le plus performant, car il ne souffre pas des limitations d'un simulateur ou d'un module d'évaluation. Son rôle consiste à émuler en temps réel le fonctionnement du DSP au sein même du prototype de l'application à développer. Toutes les ressources du DSP cible sont libres pour tester non seulement le code du programme de l'application, mais également le fonctionnement du prototype.

Figure 4.3 : Emulateur USB haute performance 1,5Mo/s

Tout comme le module d'évaluation, un émulateur est piloté par un PC, via lequel il est possible d'examiner la mémoire et les registres du DSP. Il est également possible de poser des points d'arrêts à déclenchements

sophistiqués, basés par exemple sur des conditions logiques portant sur le contenu de registres, de mémoires, voire de ports d'entrées/sorties. Un émulateur permet en outre de garder une trace des instructions exécutées dans telle ou telle partie du code à tester, ce qui facilite grandement le débogage dans certains cas complexes.

Seul moyen vraiment sûr pour tester un programme et un prototype, un émulateur reste néanmoins handicapé par son prix élevé dont il faut tenir compte dans le coût global d'un développement. Il faut noter que les DSP récents incluent directement dans leurs coeurs des fonctions d'émulation (points d'arrêts, registres spéciaux, etc.) Cette approche permet de simplifier la conception des émulateurs et tends à les rendre moins chers.

5. Structure logicielle de développement

Les deux principales méthodes pour écrire un programme DSP consistent à utiliser un assembleur dédié ou un langage de haut niveau. Un langage de haut niveau comme le langage C présente l'avantage d'être connu par la plupart des ingénieurs amenés à travailler dans le domaine du traitement du numérique du signal. Un programme DSP écrit en langage C peut donc être compris relativement facilement par un grand nombre de personnes, sans qu'elles aient besoin de connaître précisément le DSP cible. De plus, la portabilité du langage C permet de d'utiliser un programme sur des DSP fabriqués par différents constructeurs.

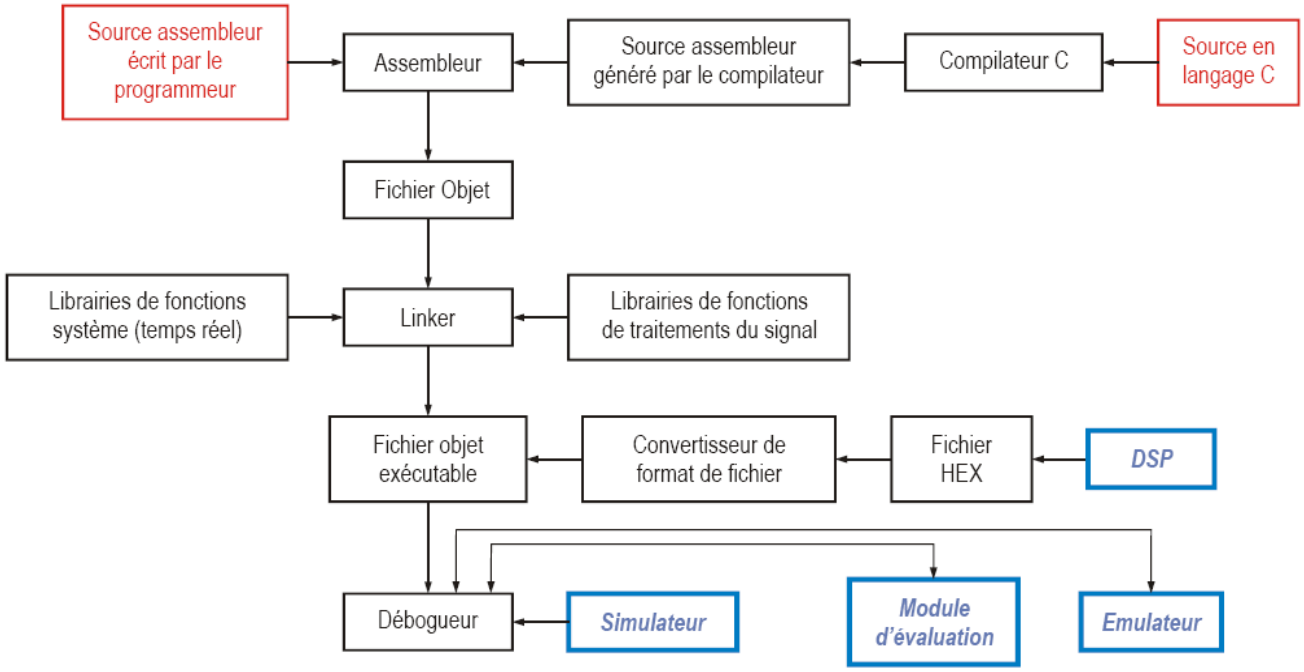

Figure 4.4 *: Organigramme d'un système de développement de logiciel pour DSP*

L'utilisation d'un assembleur pour programmer un DSP n'en demeure pas moins intéressante. Plus que pour un microprocesseur classique, les performances de traitement sont cruciales, et l'assembleur est le seul langage permettant d'utiliser totalement les possibilités spécifiques de tel ou tel DSP. Par exemple, un algorithme de codage LPC est ainsi estimé 1,5 fois plus rapide quand il est implémenté en assembleur plutôt qu'en C. Même lorsqu'ils sont spécialement optimisés pour le DSP cible, les compilateurs C ne permettent pas de générer un programme ayant les performances d'un code bien écrit en assembleur par un développeur confirmé.

La souplesse du système de développement permet d'ajouter facilement des portions de programmes assembleurs (assemblage « en ligne », ou librairie de fonctions assembleurs) à un logiciel écrit en C. Ce mélange est recommandé par les constructeurs, ainsi la plupart des exemples de traitements numériques sont donnés en assembleur, alors que les exemples de mise en oeuvre du DSP

sont donnés en C. Les outils de débogage acceptent indifféremment l'un, l'autre, ou le mélange des deux langages.

Certaines sociétés indépendantes des constructeurs fournissent des librairies de fonctions de traitements numériques du signal prêtes à l'emploi. Le langage C facilite l'intégration des librairies, voire même la mise en oeuvre de véritables systèmes d'exploitations temps réels et multitâche, par exemple le système SPOX de Spectron Microsystems.

Bien qu'étant le plus répandu, le langage C n'est pas le seul utilisable pour programmer un DSP, il existe quelques rares compilateurs ADA, FORTRAN et PASCAL pour DSP.

6. Utilisation d'un logiciel pour programmer

6.1 Intérêt d'un logiciel convivial : génération de codes pour DSP

Dans une étude de simulation en électronique, en traitement du signal, ou en automatique, il est intéressant d'utiliser des outils mathématiques appropriés pour tester ou faire fonctionner en quasi-émulation un montage, un système ou une transmission de données. La programmation du DSP se fait sur l'écran de l'ordinateur, à l'aide de schémas-blocs et de signaux de commande précisés sur la modélisation du système, ce qui est plus agréable et plus convivial pour l'utilisateur. Les algorithmes sont étudiés par simulation, puis sont transformés pour être exécutables par un DSP.

Le logiciel dispose de « boîtes à outils » de problèmes déjà résolus, sortes de sousprogrammes qui facilitent une étude plus globale d'un système. Il est ainsi possible de créer des fichiers en langage C qui seront compilés pour être exécutés par un DSP. On parle alors de générateurs de codes pour DSP.

6.2 Les générateurs de codes vers dSPACE

La société Scientific Software propose deux logiciels: MATLAB et SIMULINK et une carte d'application nommée dSPACE.

L'analyse mathématique s'effectue grâce au logiciel MATLAB. Pour une étude plus proche des problèmes de l'électronicien ou de l'automaticien, il est possible de lui associer le logiciel SIMULINK. L'étude d'un système va alors aboutir à une commande optimisée par DSP, écrite éventuellement en langage C. Un compilateur convertit le programme écrit par le logiciel en un fichier exécutable par un DSP en virgule flottante, situé sur une carte d'implantation dSPACE.

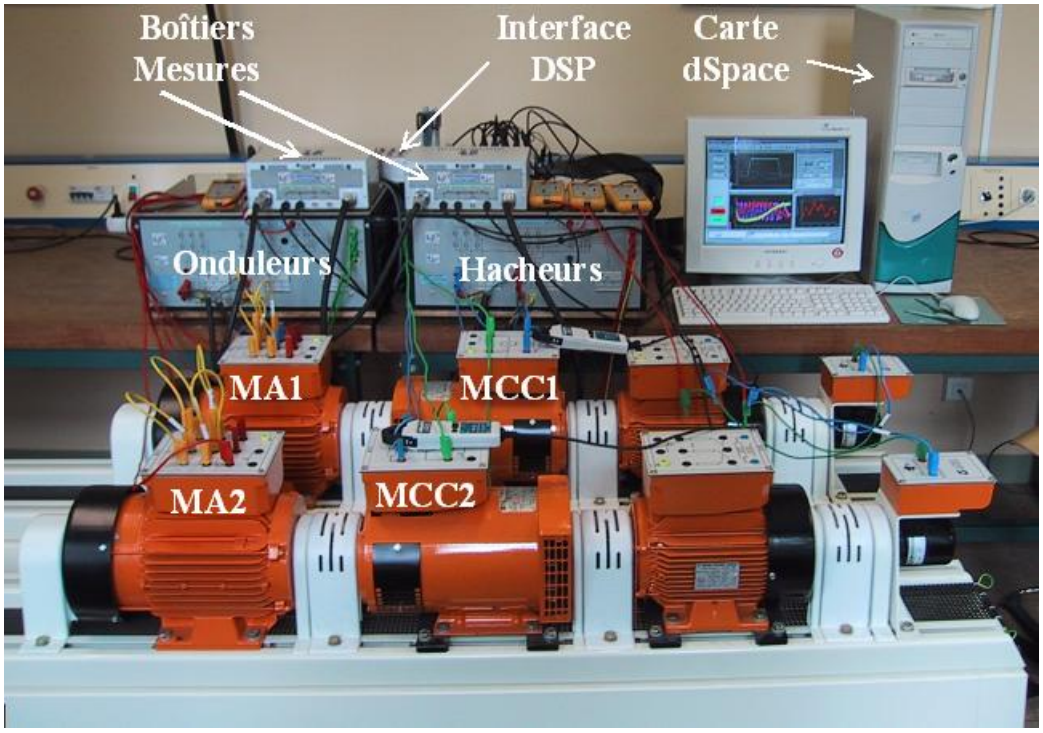

Figure 4.5 *:* Plate-forme expérimentale utilisant dSPACE

Ce DSP va ensuite fonctionner de manière à tester la commande avec des signaux réels issus d'un système complexe, comme le contrôle d'un robot.

Le logiciel MATLAB sert à l'analyse mathématique des données, telles que la résolution d'équations différentielles ou l'analyse numérique matricielle. Par la suite, on peut grâce à des « outils », développer l'analyse du signal : convolution, corrélation, transformée de Fourier, traitement d'images... MATLAB permet en outre l'identification du domaine de fréquences des

systèmes utilisés en électronique et en traitement du signal. Enfin, il est possible, en vue de l'application au DSP, de tester des algorithmes, par exemple ceux qui sont utilisés en codage correcteur d'erreur pour la transmission numérique des données.

Pour étudier de manière plus approfondie un système complexe, en particulier en simulation, on passe de MATLAB à SIMULINK. À partir de là, on utilise SIMULINK en générateur de codes de commande pour DSP.

L'avantage ici est que les codes envoyés sur dSPACE sont bien adaptés au DSP utilisé, quel que soit le fabricant et que les performances du composant sont optimales.

Dans ces conditions, on peut faire fonctionner le DSP en temps réel, c'est-à-dire en réaction complète et rapide avec le fonctionnement et le comportement du système commandé.

6.3 Les logiciels de développement

La société Hewlett Packard propose des logiciels permettant, à partir d'un concept, de développer complètement une programmation de DSP, en utilisant soit le langage C, soit les outils MATLAB. Bien entendu, des outils, des bibliothèques de données et de sousprogrammes sont disponibles et il est possible d'effectuer toutes les simulations nécessaires pour vérifier le fonctionnement.

7. Spécialisation

Pour un usage donné du DSP (commande numérique d'un moteur électrique, comme un moteur asynchrone triphasé, par exemple) une classification est possible selon la facilité de mise en oeuvre du programme du composant et selon les performances obtenues.

7.1 Commande de moteurs

Les TMS320F240x sont des DSP-contrôleurs 16 bits, à virgule fixe, qui sont très bien adaptés à ce rôle. Ils comportent les sorties nécessaires à la commande par modulation de largeur d'impulsions vectorielle de l'onduleur alimentant le moteur asynchrone. Il fonctionne avec deux fréquences d'horloge : l'une pour les opérations à 20MHz, l'autre pour le «chien de garde » (*Watch Dog*) de la fonction contrôleur, à 16384Hz. Les bus internes fonctionnent en 16bits. D'autre part, il est possible d'envisager plusieurs modes de commande, de types d'asservissement, à partir de données obtenues concernant le moteur : courant et position angulaire. Il effectue en temps réel les calculs de transformation nécessaires (transformation de Park) pour optimiser les commandes d'asservissement.

La société Analog Devices proposait un système concurrent nécessitant un ensemble DSP et ADSP2115, 16bits, à virgule fixe, associé à un processeur vectoriel ADMC200 pour effectuer les transformations de Park. Depuis, cette société présente un DSP-microcontrôleur ADMC330, ADMC401, qui réalisent sensiblement les mêmes fonctions que le TMS320C240 : fréquence d'horloge 13MHz, virgule fixe, 16bits. En 2002 est apparu une nouvelle famille ADSP21299x possédant toutes les caractéristiques de l'ADMC401 avec en plus un bus CAN et un temps de cycle de l'ordre de 6ns.

7.2 Traitement de signaux simultanés

Un autre type de DSP spécialisé est celui qui gère de manière souple des signaux simultanés, tel l'ADSP21csp0l de Analog Devices. C'est un DSP 16bits à virgule fixe, de performance 50MIPS, avec un temps de cycle de 20ns. L'appellation csp (*concurrent signal processing*) signifie que ce DSP peut gérer à la fois quatre entrées/sorties de mémoires externes sans difficulté, grâce à des mémoires « tampon » d'attente.

7.3 DSP pour téléphones cellulaires

UADSP21msp5l de Analog Devices dispose dans son circuit intégré d'un système de traitement de la voix. Il comporte un convertisseur analogique/numérique de type sigmadelta, avec des filtres anti-repliement de spectre. Il est spécialement conçu pour les téléphones cellulaires. La société Motorola fabrique également des DSP pour téléphones cellulaires.

7. Exemple de la famille ADSP21xx d'Analog Devices

La famille ADSP 2lxx est à virgule fixe utilisant des données en 16 bits. Cette famille a évolué très vite en fonction de la diversification des besoins des DSP Ces besoins ont fait naître des opérations particulières à certains DSP, comme le port hôte (*HIP : Host Interface Port*) des ADSP2111, 2171 et 2173, qui permet de communiquer des données avec un autre processeur, ou l'interface analogique de l'ADSP21msp58 (Tableau 4.1). Pour tous les DSP de la famille 2lxx, le constructeur a cherché à ce que l'architecture interne du composant permette d'exécuter en un seul cycle d'horloge les tâches ci-dessous :

- ✓ créer l'adresse suivante du programme ;
- ✓ aller chercher l'instruction suivante ;
- ✓ réaliser un ou deux déplacements de données ;
- ✓ exécuter l'instruction.

Dans un cycle d'horloge le DSP peut aussi

- ✓ recevoir ou transmettre les données d'un port série ;
- ✓ recevoir ou transmettre les données d'un port DMA ou port hôte.

Le coeur des DSP de la famille ADSP21xx dédiée spécifiquement aux calculs contient les ensembles principaux suivant :

- ! un **ALU** (*Arithmetic and Logic Unit*) permettant les calculs simples du processeur : addition, soustraction, opérations logiques entre données de 16 bits,

- un **MAC** *(Multiplier and ACcumulator)* permettant les calculs de multiplication et d'addition (ou de soustraction) entre données de 16 bits,
- un **BS** *(Barrel Shifter)* permettant le décalage de bits sur 16 ou 32 bits en mode logique et arithmétique,
- un **DAG** *(Data Address Generator),* générateur d'adresses de données, un pointeur, indispensable pour la plupart des opérations.

Le processeur ADSP-2101, le plus ancien, comporte deux ports série pour la transmission des données, 1k (1024 mots) de mémoire de données de 16 bits en mémoire vive (RAM) et 2k de mémoire de programme de 24 bits en RAM. Le bus d'adressage est en 14 bits. L'horloge interne fonctionne à 25 MHz. La technologie est de type C-MOS, où la longueur du canal est de 0.5µm. La consommation du composant est inférieure à 1W.

Le processeurADSP-2181, l'un des plus récents, comporte deux ports série pour; la transmission des données, 16k de mémoire de données de 16 bits en mémoire vive (RAM) et 16k de mémoire de programme de 24 bits en RAM. Le bus d'adressage est en 14 bits.

L'horloge interne fonctionne à 33MHz. La technologie est du type C-MOS, avec une longueur du canal de 0.5µm. Son architecture est conçue pour mener en parallèle de multiples opérations afin de favoriser le « pipe-lining ».

Caractéristiques	2101	2103	2105	2115	2111	2171	2173	2181	2183	21msp58
ALU	!	!	!	!	!	!	!	!	!	!
MAC	!	!	!	!	!	!	!	!	!	!
Décaleur	!	!	!	!	!	!	!	!	!	!
DAG	!	!	!	!	!	!	!	!	!	!
Séquenceur	!	!	!	!	!	!	!	!	!	!
Mémoire	1k	1k	512	512	1k	2k	2k	16k	16k	2k

données										
Mémoire programme	2k	2k	1k	1k	2k	2k	2k	16k	16k	2k
Timer	!	!	!	!	!	!	!	!	!	!
Port série 0	!	!		!	!	!	!	!	!	!
Port série 1	!	!	!	!	!	!	!	!	!	!
Port Hôte					!	!	!			
Port DMA								!	!	
Interface analogique										!
Alimentation	5V	3.3V	5V	5V	5V	5V	3.3V	5V	3.3V	5V
Vitesse (MIPS)	20	10	13.8	20	20	33	20	33	33	26

Tableau 4.1 : Famille ADSP21xx