SympA'5
Rennes, 8 juin - 11 juin 1999
5ème Symposium sur les Architectures Nouvelles de Machines

Une plate-forme pour l'enseignement et le prototypage d'architectures reconfigurables

Jacques-Olivier HAENNI[1], Christof TEUSCHER[1], Francisco J. GÓMEZ[2], Héctor Fabio RESTREPO[1] et Eduardo SANCHEZ[1]

[1]Laboratoire de Systèmes Logiques, Ecole Polytechnique Fédérale de Lausanne
CH – 1015 Lausanne, Suisse
E-mail : {name.surname}@di.epfl.ch
Web : http://lslwww.epfl.ch

[2]Escuela Técnica Superior de Informática, Universidad Autónoma de Madrid
E – 20849 Madrid, Spain
E-mail : francisco.gomez@ii.uam.es

Résumé

Labomat 3 est une plate-forme reconfigurable développée dans notre laboratoire. Elle est utilisée à la fois pour l'enseignement et pour la recherche. Cette carte est composée d'un microprocesseur connecté à deux circuits FPGA moyen de gamme et est munie d'un système d'exploitation temps réel sur lequel peut tourner une machine virtuelle Java. Labomat 3 est également équipée d'interfaces de communication évoluées (dont Ethernet 10Base-T et TCP/IP). L'association de ces caractéristiques fait de Labomat 3 un outil unique pour l'enseignement.
Cet article présente également trois domaines d'applications : conception de systèmes logiques, architecture des ordinateurs et codesign.

1 Introduction

Il est communément admis que la reconfigurabilité et la densité croissante des circuits FPGA entraînent une vraie révolution dans le domaine des circuits digitaux [7]. Après avoir été quasiment réservés à des tâches de prototypage, ils ont rapidement été impliqués dans un grand nombre d'applications.

Naturellement, l'enseignement a aussi tiré parti de ces possibilités, et, par conséquent, de nombreuses plates-formes existent. Dans certains cas, les fabriquants de circuits FPGA proposent eux-mêmes de telles plates-formes [1, 5]. Toutefois, ces systèmes de développement laissent actuellement de côté un des domaines les plus prometteurs : le codesign – le choix des parties d'une application à implémenter en matériel ou en logiciel [2].

Dans un curriculum universitaire classique subsiste encore une forte distinction entre la conception de logiciel et de matériel. Toutefois, avec le codesign, cette frontière très claire

se dissout, ce qui entraîne un changement fondamental dans la formation de l'ingénieur.

La carte présentée dans ce papier, Labomat 3, à été développée dans le but d'être utilisée par des étudiants dans toutes sortes de cours de conception de matériel, des systèmes logiques élémentaires au codesign, en passant par l'architecture des ordinateurs ou des microprocesseurs.

Labomat 3 présente quelques caractéristiques uniques :

1. Un microprocesseur de la famille 68000 de Motorola et deux circuits FPGA de Xilinx sont disponibles, dont un XC4013E qui permet aux étudiants, par exemple, d'implémenter un processeur complet avec un jeu d'instructions simple, mais avec un pipeline à 5 niveaux et la correction des aléas. La carte dispose aussi d'un système d'exploitation temps réel (RTEMS [9]) et d'un large ensemble d'outils logiciels, parmi lesquels une machine virtuelle Java et TCP/IP.

2. Une interface Ethernet ainsi que le protocole TCP/IP sont disponibles, permettant l'accès et la configuration de la carte via le réseau.

3. Labomat 3 contient un circuit FPGA XC6216 de Xilinx [11]. Un logiciel de CAO tirant parti de la reconfigurabilité partielle et dynamique de ce circuit a été développé. Il permet aux étudiants de dessiner un schéma logique de manière traditionnelle, avec l'avantage que ce schéma est une image conforme de la configuration du circuit. Il est dès lors possible de directement exécuter le schéma sur le circuit FPGA, sans devoir passer par le processus de placement et routage. Il est également possible de lire les valeurs de tous les signaux directement sur le schéma. Ce système combine donc les avantages de la simulation et de l'émulation.

Toutes ces caractéristiques font de Labomat 3 un outil d'enseignement unique.

La suite de ce papier est organisée comme suit. Dans la première partie, nous donnons une description complète du matériel de Labomat 3, ainsi que de son logiciel et de ses interfaces de communication. Dans la deuxième partie, nous présentons trois domaines d'application : conception de systèmes logiques, architecture des ordinateurs et codesign. Nous conclurons ce papier en mettant en évidence quelques considérations sur les travaux futurs.

2 Description matérielle

La carte est séparée en deux parties principales (fig. 1) :

1. **la partie processeur** est construite autour d'un processeur MC68EN360 de Motorola [8]. Il est connecté à 512 ko d'EPROM (pour le démarrage de la carte) et à un maximum de 32 Mo de DRAM. La mémoire Flash (512 ko) peut garder des configurations de FPGA ou toute autre information

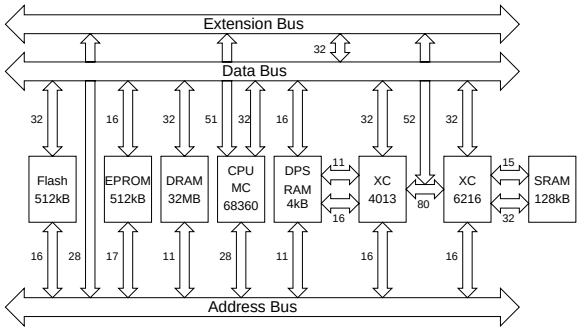

FIG. 1 - *Architecture de Labomat 3*

non volatile. Une mémoire double-accès de 4 ko facilite le partage des données entre le processeur et la partie reconfigurable. Des interfaces RS-232 et Ethernet 10Base-T sont utilisées pour communiquer avec le monde extérieur.

Le processeur MC68360 a été choisi pour ses capacités de communication. De plus, il existe beaucoup de logiciel du domaine public pour ce processeur.

2. **la partie reconfigurable** est construite autour de deux circuits FPGA de Xilinx, un XC4013E et un XC6216. Le XC6216 est connecté à 128 ko de SRAM qui n'est pas directement accessible depuis le processeur. Le XC4013E est connecté à la mémoire double-accès qui est également connectée au processeur.

Le processeur peut directement accéder aux circuits FPGA comme à des périphériques par ses bus de données et d'adresses. Un bus de 80 signaux de large connecte les deux circuits FPGA. Notre laboratoire utilise un connecteur standard à 10 pôles pour tous ses équipements d'enseignement. Un sous-ensemble de 52 signaux parmi les 80 reliant les circuits FPGA est ainsi directement accessible sur de tels connecteurs 10 pôles. Le bus d'extension possède aussi les bus de données, d'adresse et de contrôle.

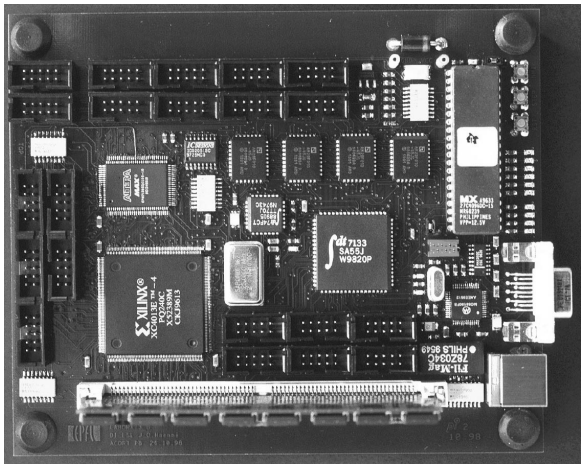

FIG. 2 - *La carte Labomat 3*

Le contrôle du bus et des interruptions est réalisé par un circuit programmable (MAX7128). Nous avons réussi à cacher les timings compliqués du bus du processeur au circuit FPGA.

Chaque circuit FPGA reçoit deux horloges, chacune pouvant provenir de différentes sources: une horloge programmable peut générer une vaste gamme de fréquences; un bouton-poussoir permet de faire du pas-à-pas; les signaux d'horloge peuvent également être générés par le processeur. La fréquence d'horloge de base du système est de 25 MHz.

Le code de démarrage est exécuté depuis l'EPROM. Ce code télécharge le système d'exploitation complet et les applications depuis l'interface Ethernet (voir section 4). Il n'est ainsi pas nécessaire de reprogrammer l'EPROM à chaque changement de système d'exploitation.

La carte Labomat 3 est implémentée sur un PCB à 6 couches (fig. 2). Le XC6216 et le processeur sont soudés sur la face inférieure de la carte.

3 Description logicielle

Plutôt que d'implémenter un simple moniteur aux possibilités limitées, nous avons préféré porter un système d'exploitation complet sur Labomat 3. Cela lui apporte, entre autres, une grande facilité d'utilisation.

Après avoir examiné plusieurs possibilités, nous avons choisi RTEMS qui est un système d'exploitation temps réel et dont les besoins en mémoire sont relativement modestes. De plus, il possède des pilotes pour Ethernet et contient une librairie TCP/IP, qui permet

l'usage de protocoles standard de haut niveau basés sur TCP/IP plutôt qu'un protocole simple ne s'appuyant que sur Ethernet. Enfin, il a déjà été porté sur le processeur 68360 et son code source est disponible gratuitement.

Du code spécifique à la carte (Custom Hardware Library, CHL) inclut un ensemble de fonctions permettant d'accéder aux ressources de la carte.

Au dessus de ce système d'exploitation, nous avons installé une machine virtuelle Java. Les avantages de Java sont, entre autres, sa simplicité d'utilisation, son API standardisé, sa robustesse et ses capacités de communication par le réseau. Nous avons choisi Kaffe [10] car son code source est disponible et qu'il a déjà été porté pour le processeur 68000, réduisant ainsi notre travail.

En plus de l'API Java standard, l'utilisateur a à sa disposition un API spécifique à la carte. Cet API contient des classes et des méthodes permettant l'accès aux ressources de la carte depuis du code Java.

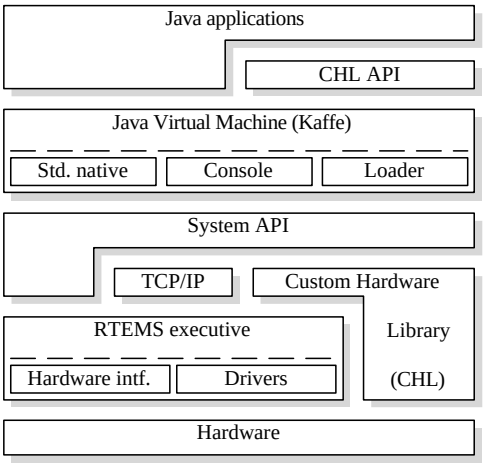

FIG. 3 - *Architecture logicielle de Labomat 3*

La figure 3 montre les différentes couches de logiciel implémentées sur Labomat 3.

4 Interfaces de communication

La plate-forme Labomat 3 offre un ensemble d'interfaces de communication permettant de configurer et contrôler la carte. Ce contrôle peut se faire par un programme C ou Java, ou de manière interactive par l'un des trois schémas suivants (fig. 4) :

1. La carte peut être connectée par son interface RS-232. Un logiciel standard d'émulation de terminal affiche alors un menu en mode texte permettant l'accès aux ressources matérielles (configuration des circuits FPGA, etc.).

2. Labomat 3 peut aussi implémenter un serveur telnet. Une fois connecté à la carte, l'utilisateur dispose du même menu en mode texte que par la ligne sérielle. Toutes les commandes passent alors par l'interface Ethernet de Labomat 3.

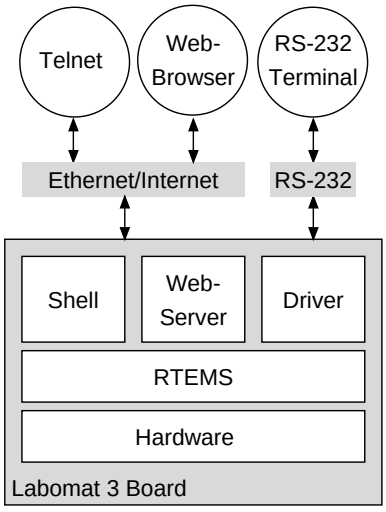

FIG. 4 - *Interfaces de communication*

3. Pour le futur, nous prévoyons d'implémenter un serveur Web sur Labomat 3 pour nous permettre de contrôler cette carte à partir d'un navigateur Web. Une carte connectée à Internet pourra alors être configurée et programmée depuis n'importe quelle machine également reliée à Internet (notons que ceci est déjà vrai avec l'utilisation d'un serveur telnet). Les étudiants pourront alors travailler à domicile et

communiquer avec une carte installée dans notre laboratoire.

D'autres applications utilisant la communication peuvent aisément être implémentées, vu que les éléments de base, tels que les sockets ou la machine virtuelle Java sont disponibles.

Les fichiers de configuration des circuits FPGA sont, pour l'instant, lus depuis un serveur TFTP (Trivial File Transfer Protocol). Le téléchargement d'un fichier de configuration et la configuration d'un circuit FPGA sont initiés par l'un des outils de communication décrits précédemment (ligne sérielle, serveur telnet ou serveur Web) ou directement depuis du code C ou Java.

Chaque carte Labomat 3 a une adresse matérielle unique (adresse MAC) à laquelle peut correspondre n'importe quelle adresse IP.

5 Systèmes logiques : un outil d'édition et de simulation

5.1 Introduction

Le développement d'un système logique pour un circuit FPGA se divise en plusieurs étapes (fig. 5) : le circuit est dessiné en utilisant un outil d'édition, puis simulé. Ensuite, l'implémentation matérielle, qui consiste en une phase de placement et routage, peut avoir lieu. Ce processus peut être optimisé soit en vitesse, soit en taille. Il est alors possible de configurer le circuit FPGA en lui envoyant une suite de bits générée lors de la phase précédente.

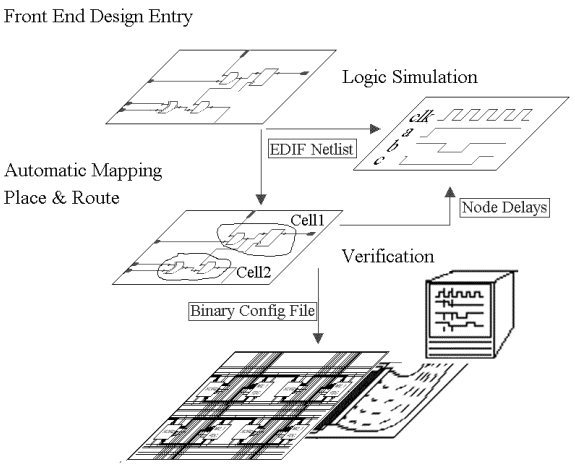

FIG. 5 - *Développement d'un système logique pour un circuit FPGA*

La famille de circuits FPGA XC6200 de Xilinx offre la possibilité de configurer indépendamment chaque cellule logique, de même que de lire et modifier l'état de chaque cellule. Ceci ouvre un nouveau domaine d'applications basées sur la reconfiguration matérielle en "temps réel".

Il est possible de concevoir des systèmes logiques pour cette famille en utilisant des outils du commerce (éditeur, synthétiseur, simulateur). Toutefois, nous avons développé un nouvel outil, RVS6200, qui permet d'éditer à l'écran un schéma logique tel qu'il sera dans le circuit FPGA, sans devoir passer par une phase de placement et de routage (fig. 6).

5.2 Description de l'outil logiciel

5.2.1 Généralités

L'éditeur de schémas logiques (fig. 7) possède une interface graphique similaire à d'autres éditeurs. Il inclut les fonctionnalités habituelles telles que couper, copier, coller, zoom avant/arrière, drag-and-drop et le déplacement d'objets dans la fenêtre. Le fond de la fenêtre contient une grille qui représente la matrice de 64x64 cellules du circuit FPGA

XC6216.

5.2.2 Edition de schémas

L'insertion d'un composant débute par le choix de ce composant dans une bibliothèque à l'aide d'une boîte de dialogue. Ce composant est ensuite placé à l'endroit désiré sur la fenêtre à l'aide de la souris.

Chaque composant possède, sur sa périphérie, des points de contact permettant de le connecter à d'autres composants ou à une patte du circuit. Les connexions que l'on peut établir sur l'éditeur sont clairement limitées par les capacités de routage du circuit. Une cellule peut ainsi être connectée à ses voisins immédiats, ou à des cellules plus lointaines en utilisant les lignes de connexion FastLane 4.

L'éditeur possède également des commandes permettant de configurer le circuit FPGA et de tester le circuit.

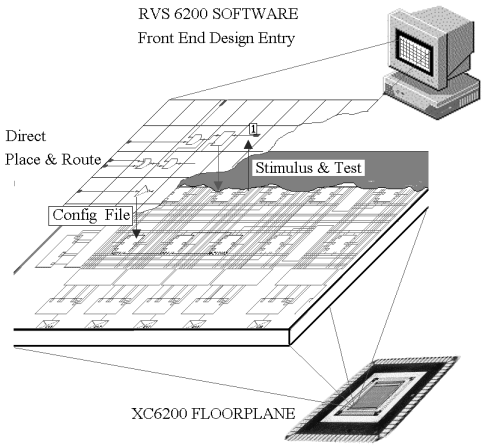

FIG. 6 - *Conception d'un système logique, avec configuration et vérification en temps réel*

5.2.3 Communication

Le logiciel RVS6200 communique avec la carte Labomat 3 en exploitant ses capacités de communication (Ethernet et TCP/IP).

L'aspect le plus innovateur de cet outil est que le schéma qui apparaît à l'écran représente exactement ce qui ce trouvera sur le circuit FPGA. De plus, il n'est plus nécessaire de disposer d'un outil de simulation, puisqu'il est possible d'exécuter directement un schéma sur le circuit FPGA, sans passer par les phases de placement et de routage, tout en offrant la possibilité de lire la valeur de chacune des bascules.

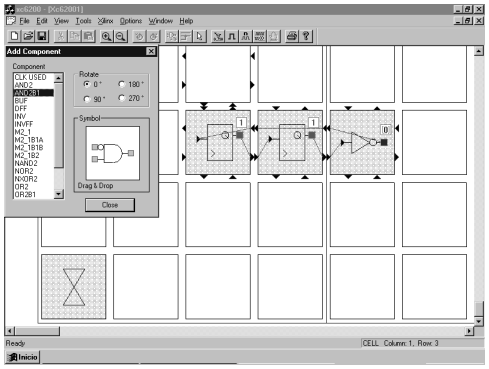

FIG. 7 - *Interface graphique de l'éditeur de schémas*

5.3 Exemple : un multiplicateur 4 bits

En guise d'exemple, nous montrons la conception d'un multiplicateur 4 bits. Ce dernier est composé d'une unité de multiplication qui est répliquée selon une structure régulière. Le schéma contient également des registres qui forment un pipeline, permettant ainsi l'introduction de nouvelles valeurs à chaque coup d'horloge.

Le premier pas est la définition de l'unité de multiplication 1 bit de base comme un nouvel élément de la bibliothèque. Cette unité contient une porte logique AND et un full-adder. Après le placement et la connexion des composants de base de l'unité de multiplication, un nouveau composant englobant cette unité peut être créé et inséré dans la bibliothèque à l'aide de la commande *Create Component*. Ses signaux d'entrée et de sortie sont alors automatiquement sélectionnés parmi les signaux laissés non connectés sur le schéma. Un symbole graphique représentant le composant peut alors être dessiné avec des lignes, des

rectangles, des cercles et des courbes de Bézier.

Pour obtenir un multiplicateur 4 bits, il suffit de connecter plusieurs instances du multiplicateur 1 bit comme indiqué sur la figure 8.

Une fois que le schéma est terminé, une connexion avec Labomat 3 est établie pour configurer le circuit FPGA et tester le système.

Une fois que le circuit FPGA est configuré, il est possible de lui envoyer des coups d'horloge. La valeur des différents signaux est alors mise à jour en temps réel sur le schéma, permettant ainsi un dépannage aisé du système.

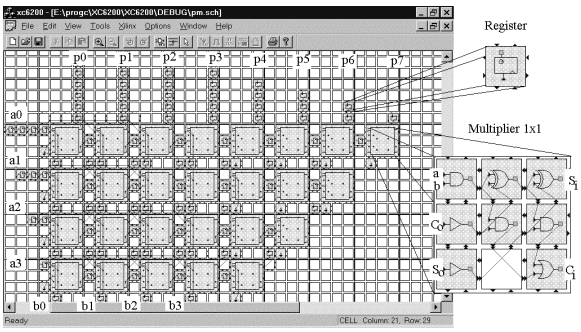

FIG. 8 - *Multiplicateur 4 bits avec pipeline*

6 Architecture des ordinateurs : un processeur à pipeline

6.1 Introduction

Ce chapitre présente l'implémentation d'un processeur à pipeline sur le circuit FPGA XC4013E de Xilinx. Les étudiants peuvent non seulement simuler leur processeur, mais aller un pas plus loin en exécutant leur système sur du matériel réel. Le but n'est pas de réaliser un processeur à haute performance, mais plutôt d'examiner certaines difficultés telles que les aléas et les exceptions qui apparaissent naturellement dans un processeur à pipeline. Il s'agit donc clairement d'une application purement académique.

6.2 Spécifications

Les spécifications de ce processeur sont en partie inspirées de l'architecture DLX [4]: il s'agit d'une architecture load/store 8 bits avec quatre registres internes à 8 bits (R0, R1, R2 et R3). Contrairement à l'architecture DLX, le registre R0 ne contient pas la constante 0.

Le jeu d'instructions contient les huit instructions suivantes:

LOAD	Rd $\leftarrow$ M[*addr*]
STORE	M[*addr*] $\leftarrow$ Rs
MOVE	Rd $\leftarrow$ Rs
SUB	Rd $\leftarrow$ Rs1 - Rs2
ADD	Rd $\leftarrow$ Rs1 + Rs2
CMP	flag $\leftarrow$ Rs1 - Rs2
BRANCH	if flag then PC $\leftarrow$ PC + *offset*
JMP	PC $\leftarrow$ PC + *offset*

Rd, Rs, Rs1 et Rs2 représentent chacun un registre parmi les quatre registres disponibles et M[*addr*] un accès en mémoire à l'adresse *addr*. Aucune instruction ne génère plus d'un résultat.

Un pipeline à 5 niveaux, dont chaque niveau est exécuté en un coup d'horloge, est mis en œuvre:

IF	Instruction fetch cycle
ID	Instruction decode cycle
EX	Execution cycle
MEM	Memory access cycle
WB	Write back cycle

6.3 Format d'instruction

Instr.	OPC 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOAD	0	0	0	Rd		address										
STORE	0	0	1	Rs		address										
MOVE	0	1	0	Rs		Rd		0	0	0	0	0	0	0	0	0
SUB	0	1	1	Rd		Rs1		Rs2		0	0	0	0	0	0	0
ADD	1	0	0	Rd		Rs1		Rs2		0	0	0	0	0	0	0
CMP	1	0	1	0	0	Rs1		Rs2		0	0	0	0	0	0	0
BRANCH	1	1	0	0	0	offset										
JMP	1	1	1	0	0	offset										

FIG. 9 - *Format d'instruction*

De même que dans le cas des processeurs RISC actuels, les instructions ont un format de longueur fixe (16 bits). Le seul mode d'adressage disponible est l'adressage absolu pour les instructions load/store et l'adressage relatif au PC pour les instructions de branchement et de saut (fig. 9).

Chacun des quatre registres peut être lu ou écrit sans restriction. Un registre est spécifié par deux bits dans une instruction, laissant ainsi 11 bits disponibles pour l'adresse ou l'offset, le cas échéant.

6.4 Architecture du processeur

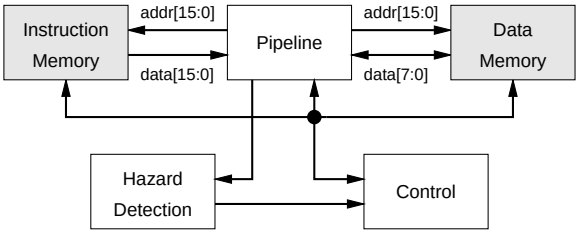

FIG. 10 - *Architecture du processeur*

Afin d'éviter des blocages du pipeline, les mémoires d'instructions et de données sont séparées (architecture de Harvard). En effet, il est alors possible de simultanément lire une instruction (niveau IF) et accéder à une donnée en mémoire (niveau MEM).

L'unité de détection des aléas collabore très étroitement avec l'unité de contrôle. Les cinq blocs fonctionnels de la figure 10 correspondent à des entités VHDL différentes.

6.5 Pipeline

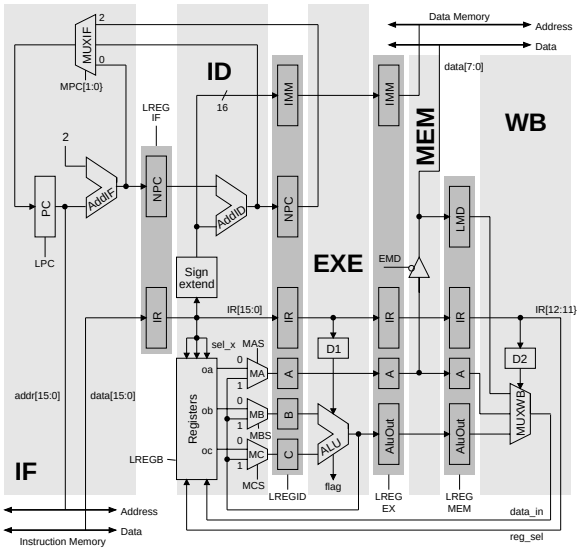

FIG. 11 - *Pipeline*

Afin de pouvoir soutenir un rythme d'exécution d'un coup d'horloge par instruction (CPI), le pipeline doit pouvoir exécuter n'importe quelle combinaison d'instructions. Afin d'éviter des aléas structurels, le pipeline de la figure 11 n'a pas de conflit de ressource : n'importe quelle instruction peut s'exécuter simultanément avec n'importe quelle autre instruction. L'accès aux registres est réalisé aux deux flancs d'horloge. Un des quatre registres (R0, R1, R2 et R3) est sélectionné par un multiplexeur pour chaque sortie (oa, ob et oc). Le contenu des registres est modifié au flanc montant de l'horloge durant le niveau WB, alors qu'il est lu durant

le flanc descendant.

Puisque le CPI vaut 1, le PC doit être incrémenté dans le niveau de chargement de l'instruction (niveau IF). Trois multiplexeurs (MA, MB et MC) permettent le bypassing. D1 et D2 sont des décodeurs qui contrôlent les opérations de l'ALU et de MUXWB. Ces opérations dépendent des instructions dans les registres IR. Le bloc d'extension de signe du niveau ID étend l'offset ou l'adresse absolue de 11 à 16 bits.

6.6 Détection des aléas

L'unité de détection des aléas n'a pour seule entrée l'instruction couramment chargée (fetch). Puisque le pipeline ne contient aucun aléa structurel, seuls les aléas de données et de contrôle doivent être détectés. La détection des aléas de contrôle est très simple : une instruction jump bloque toujours le pipeline pour un coup d'horloge, et une instruction branch pour deux coups d'horloge, indépendamment du fait que le saut soit pris ou non.

Un moyen simple de détection des aléas de données est mis en œuvre : le numéro du registre de destination d'une instruction nouvellement chargée (fetch) est entré dans un FIFO à quatre niveaux. Considérons, par exemple, l'instruction move suivante :

MOVE R2 ← R3

Cette instruction modifiera le registre R2 lors du niveau WB. Ainsi, le numéro 2 est entré dans le FIFO, qui est décalé d'une position à chaque coup d'horloge. Chaque fois qu'une nouvelle instruction est chargée, le système teste si le numéro du (ou des) registre(s) source(s) (3 dans l'exemple ci-dessus) est présent dans le FIFO. Si tel n'est pas le cas, le chargement des instructions peut continuer, et le pipeline n'est pas bloqué. Dans le pire des cas, un aléa de données génère un blocage de quatre coups d'horloge, comme illustré dans l'exemple suivant :

MOVE R2 ← R3
MOVE R0 ← R2

La seconde instruction ne peut être exécutée avant que la première ne soit complètement terminée. Le numéro 2 (registre modifié par la première instruction) restera dans le FIFO durant quatre coups d'horloge. Clairement, un système plus subtile de bypassing augmenterait les performances. Le bypass de la figure 11 est implémenté, mais n'est pour l'instant pas utilisé par l'unité de contrôle (fig. 10).

6.7 Implémentation

Le processeur a été entièrement implémenté en VHDL à l'aide du synthétiseur FPGA-Express. Une simulation fonctionnelle triviale de la mémoire d'instructions est utilisée, bien qu'un modèle plus réaliste respectant les timings d'accès réels pourrait être utilisé. Dans notre cas, le processeur peut écrire et modifier le code exécuté par le processeur RISC, car ce code se trouve dans la mémoire à double-accès. La mémoire de données est câblée sur 8 connecteurs d'extension connectés à des interrupteurs et à des LED. Grâce au bouton pas-à-pas, le programme VHDL peut facilement être vérifié et débogué.

L'implémentation sur un XC4013E utilise 296 CLB sur 576 : il reste donc encore suffisamment de matériel disponible pour de futures extensions et améliorations. La fréquence d'horloge maximale du système est de 25 MHz.

7 Codesign : un coprocesseur à virgule flottante

Cette application consiste en la conception d'un coprocesseur à virgule flottante (FPU, floating point unit) simple. Les étudiants doivent réaliser un multiplicateur à virgule flottante à l'aide de portes logiques, le simuler, puis le tester sur la carte Labomat 3 (fig. 12). Ceci implique la conception d'une configuration de circuit FPGA implémentant le multiplicateur et son interface avec le processeur, ainsi que l'écriture d'un programme pour le processeur afin de commander et tester le coprocesseur.

Le processeur peut accéder au circuit FPGA comme à un périphérique par ses bus de données et d'adresses. Le circuit FPGA n'a pas à générer de signaux d'acquittement complexes. En fait, il est séparé du 68360 par de la logique responsable d'adapter les fréquences des horloges (l'horloge du circuit FPGA peut être différente de 25 MHz). De cette manière, le circuit FPGA peut tourner à une fréquence d'horloge quelconque, tout en recevant un signal de sélection (chip select) synchronisé sur sa propre horloge et tout en générant des signaux d'acquittement également synchronisés sur son horloge.

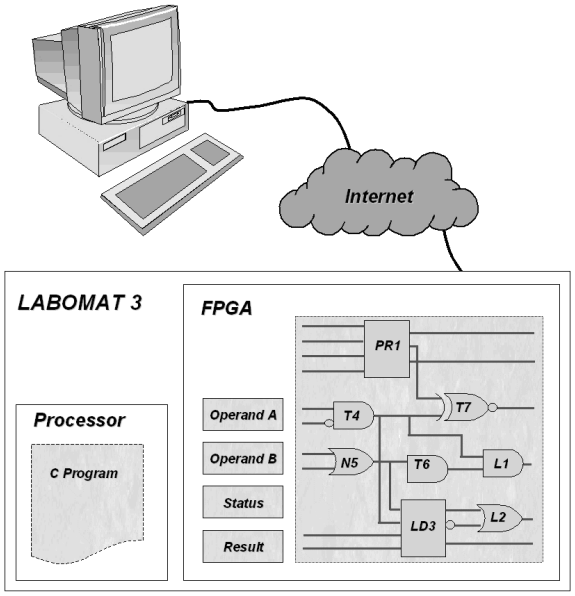

FIG. 12 - *Labomat 3 comme outil de codesign*

Le processeur et le coprocesseur communiquent par l'accès périphérique du processeur. Le processeur pourra ainsi écrire les deux opérandes à des adresses données, attendre la fin du calcul (soit en lisant continuellement un registre de statut du circuit FPGA indiquant l'état du calcul, soit en utilisant des interruptions) et lire le résultat à une adresse donnée.

La partie matérielle de ce coprocesseur est développée sous l'environnement WorkView Office sur des PC tournant Windows NT. Il est principalement composé de logique combinatoire commandée par des machines à états responsables du séquencement des micro-opérations.

Le logiciel, écrit en C, comprend un appel à une routine prédéfinie pour la configuration du circuit FPGA, suivi d'une boucle lisant les opérandes depuis le terminal et les envoyant au coprocesseur. Une fois que le calcul est terminé, le résultat est lu et affiché sur le terminal. De part la quantité de logiciel disponible sur Labomat 3, l'utilisateur n'a pas à se préoccuper de l'initialisation du processeur ou d'autres problèmes de bas niveau.

8 Conclusion

La combinaison de ressources programmables, d'outils logiciels de haut niveau et d'un accès par le réseau fait de Labomat 3 une plate-forme idéale pour l'expérimentation de la conception de systèmes comprenant à la fois du matériel et du logiciel, de même que l'exploration de différentes interfaces entre ces deux parties.

Labomat 3 peut être utilisée pour développer et valider des outils automatiques de codesign, et, puisque son architecture est simple, le concepteur peut concentrer son effort sur le partitionnement et la synthèse sans avoir affaire à la complexité de la plate-forme.

Notre laboratoire d'étudiants est actuellement équipé de 50 cartes Labomat 3, lesquelles ont été utilisées avec succès pour des exercices d'étudiants. Les étudiants ont ainsi la possibilité d'aller un pas plus loin que la simulation en testant leurs systèmes sur du matériel, ce qui leur permet d'être confrontés à des problèmes qui n'apparaissent pas en simulation.

Nous sommes actuellement en train de travailler sur des outils permettant de réunir et d'utiliser les 50 cartes simultanément, afin d'obtenir un grand ensemble reconfigurable (100 circuits FPGA) pour des expériences de calcul haute performance. La communication entre les cartes est possible par Ethernet, ou par des lignes de connexion directes reliant les cartes. Des expériences dans les domaines des réseaux de neurones artificiels, des algorithmes génétiques ou de la vie artificielle sont prévues.

Remerciements

Nous remercions André Badertscher pour la photographie de Labomat 3 et pour le montage des cartes. Ce travail est partiellement financé par un fond de la fondation Werner Steiger.

Bibliographie

1. D. van den Bout. *The practical Xilinx designer lab book*. Prentice Hall. 1998.
2. L. Garber and D. Sims. *In pursuit of hardware-software codesign*. IEEE Computer, 31(6):12-14, June 1998.
3. J.O. Haenni, J.L. Beuchat, E. Sanchez. *RENCO: A Reconfigurable Network Computer*. Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines. FCCM '98. Napa USA. April 15-17, 1998. pp. 288-289.
4. J.L. Hennessy, D.A. Patterson. *Computer Architecture - A Quantitative Approach*. Morgan Kaufmann. Second Edition, 1990.
5. Z. Salcic and A. Smailagic. *Digital systems design and prototyping using field programmable logic*. Kluwer. 1997.
6. E. Sanchez, M. Sipper, J.O. Haenni, J.L. Beuchat, A. Stauffer, and Andres Perez-Uribe. *Static and Dynamic Configurable Systems*. IEEE Transactions on Computers, Special Issue on Configurable Computing, to be published in June 1999.
7. J. Villasenor and W. H. Mangione-Smith. *Configurable computing*. Scientific American, 276(6):54-59, June 1997.
8. Motorola MC68360. *Quad Integrated Communication Controller*. User's Manual. 1993.
9. RTEMS Home Page. http://www.oarcorp.com.
10. Kaffe, A free virtual machine to run Java code. http://www.kaffe.org.
11. Xilinx. *Inc XC6200 Advanced product specification* V1.10 4/97. The Programmable Logic Data Book 1997. http://www.xilinx.com.