Mini-Control (MCL)

Une machine virtuelle sur un processeur minimal

Dominique Rhême, dominique.rheme@eif.ch, EIA-FR.
Michel Bovet, michel.bovet@eif.ch, EIA-FR/ABMI SA
Louis Schmitt, louis.schmitt@eif.ch, EIA-FR
Laurent Foubes et JPB, communauté Internet.



Pourquoi faire simple alors qu'il est si facile de tout compliquer? Pourquoi réveiller les vieux démons et sans cesse tenter de (re)créer son propre processeur?

Quelques réflexions plus loin et nous voici embarqués dans le projet MCL.

Genèse

En fait, le véritable point de départ nous est venu par l'observation de la situation dans le contrôle de processus. Les développeurs ont mis durant deux décennies toute leur ingéniosité à complexifier ce qui peut l'être. A la base, il y avait l'automate séquentiel programmable. La machine à tisser Jacquard en est d'ailleurs l'archétype. Quels sont les besoins? Pour développer un automate, il faut:

- un langage de haut niveau,
- un assembleur (produit le code machine),
- un interpréteur de commandes,
- un compilateur,
- un *débogueur*,
- un exécuteur de tâches.

Certains diront *«et encore un système d'exploitation»*.

Les progrès foudroyants de la technologie aidant, peu d'efforts ont été entrepris pour rendre le concept plus efficace. A chaque génération, les gains en puissance de calcul et en place mémoire sont tels que toute optimisation est un luxe dont on ne peut payer le prix.

Conséquence: pour équiper deux ou trois systèmes d'axes d'une machine X il vous en coûtera une petite fortune. Des bâtis, des paniers, des cartes processeurs, des cartes d'interface toutes propriétaires. Et les outils informatiques de développement? Il en va de même, avec le confort que l'on sait à chaque *release* du système d'exploitation...

Mais combien d'outils ou de méthodes différentes faut-il pour tout ça?

Un concept existe, simple: la machine virtuelle.

Une machine virtuelle est basée sur l'idée de contenir un dictionnaire des commandes disponibles au départ. Le seul travail de programmation consiste à ajouter de nouvelles définitions dans le dictionnaire. Ces nouveaux programmes sont construits à l'aide des mots précédemment connus de la machine virtuelle.

Par nature, la machine virtuelle est capable d'exécuter une commande (interpréteur) ou de créer une nouvelle commande (compilateur).

Cependant, une machine virtuelle doit être implémentée sur un processeur réel. Le maillon faible! Nous voici donc à l'objectif du projet MCL: *Concevoir un processeur minimal dont le jeu d'instructions corresponde au jeu basique de la machine virtuelle*. Ainsi, la dernière transposition, le dernier intervalle entre la machine virtuelle et le processeur n'existe plus. La machine virtuelle est directement implémentée sur un processeur ad hoc.

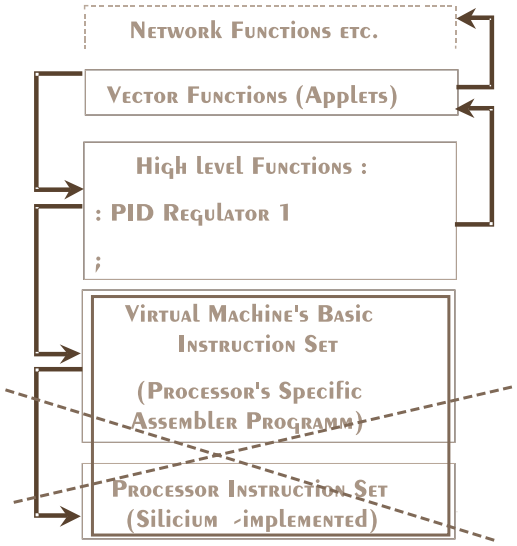

fig. 1 – La machine virtuelle, le coeur informatique entre le processeur et le monde réel

Les choix

Une ambition adaptée aux ressources constitue le début de la sagesse. Nous sommes les 10'000èmes, pour le moins, à nous lancer dans la modélisation d'un processeur. Cela n'a donc un intérêt que si le fruit de ce travail nous apporte une meilleure expertise ...

- en modélisation VHDL[1],
- dans la maîtrise des outils de conception électronique (design flow),
- en connaisance approfondie des circuits de logique programmable (FPGA[2]),

1 VHDL (*Very high speed integrated circuits Hardware Description Langage)* est un langage de description matérielle pour les circuits électroniques.

2 FPGA (Field Programmable Gate Array) est un circuit numérique programmable par l'utilisateur, équivalent à plusieurs dizaines de milliers de portes logiques.

- dans l'usage de la machine virtuelle FORTH par ailleurs abondamment utilisée au laboratoire d'automatique du département de mécanique de l'EIA-FR.

En conséquence, le choix de la structure du processeur se porte sur la variante de machine dite **à pile** (stack processor). Ces processeurs simples on été largement décrits et même parfois réalisés sur silicium.

Celui qui sait utiliser une calculatrice fonctionnant en notation dite **polonaise inverse** a déjà compris les avantages de l'usage d'une pile de donnée pour enchaîner les opérations. Ce concept simplifie les flux de données (tous les opérandes passent par la pile) et permet de réaliser un processeur avec un jeu réduit d'instructions, toutes exécutables en un seul cycle-machine. C'est donc un RISC.

Enfin, cette structure convient précisément aux opérations de base de la machine virtuelle forth. MCL a ainsi un jeu de 31 instructions qui correspondent aux commandes basiques de la machine virtuelle.

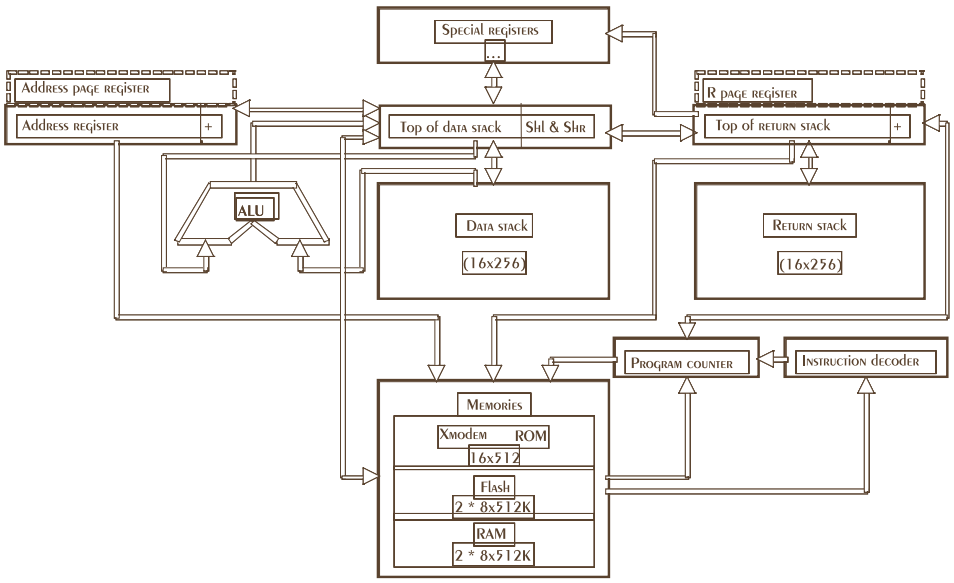

fig. 2 – L'architecture du *stack-processor*. Les deux piles (Data stack et Return stack) et la mémoire sont les principaux blocs

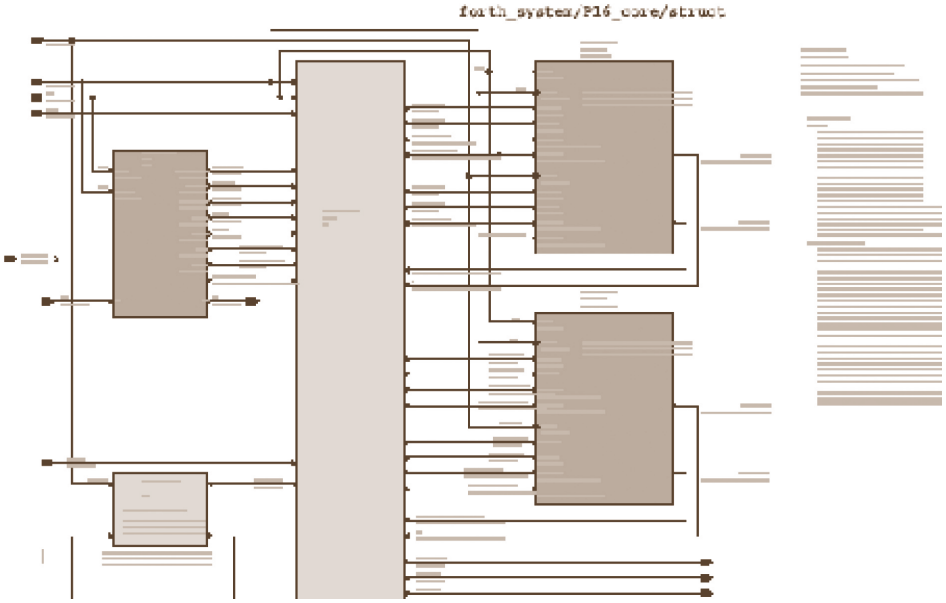

fig. 3 – La représentation *top-level* du processeur dans l'environnement HDL- Designer de Mentor Graphics

Modélisation VHDL et implémentation

La modélisation du processeur à pile a été menée à bien grâce à l'environnement de développement HDL-Designer de Mentor Graphics (encore appelé **Renoir** au moment des faits). Malgré les apparences, fort peu d'éléments ont été modélisés *graphiquement*, tout au plus les assemblages des blocs principaux.

L'essentiel du modèle a été écrit en VHDL, et c'est sous cette forme qu'il est maintenu depuis deux ans.

L'implémentation du processeur n'a pas fait l'objet de trop d'hésitations. Le démarrage du projet a coincidé avec la sortie de la famille SPARTAN II de Xilinx[1]. Nous n'avions utilisé ce produit qu'une seule fois auparavant, il était temps d'y découvrir les nouvelles fonctions promises: blocs de gestion d'horloge (DLL), blocs de mémoire configurable à double port, etc. Les premières estimations de surface nous font pencher pour le modèle XC2s50, c'est à dire une FPGA de complexité de 50'000 portes environ.

La mémoire fait aussi l'objet d'une attention soutenue. Par compromis nous fixons la largeur des données à 16 bits. Finalement, le système minimal sera doté de 512 mots RAM et de 512 mots de Boot-ROM internes à la FPGA, de 512 kMots de SRAM et de 512 kMots de Flash externes. Le contenu de la mémoire ROM est inclus dans le fichier de configuration de la FPGA et, par un artifice, est aussi utilisable lors de la simulation fonctionnelle.

Un processeur tout seul ne suffit généralement pas. Outre la mémoire, une interface pour une liaison série (UART) et un circuit temporisateur (timer) on été ajoutés. La méthodologie adoptée est un exemple, certes modeste, de développement *system on chip* (SoC).

1 A la date de parution de cet article, nous nous trouvons pratiquement dans la même situation avec la nouvelle famille Spartan III, technologie 90 nm, introduite au printemps 2003

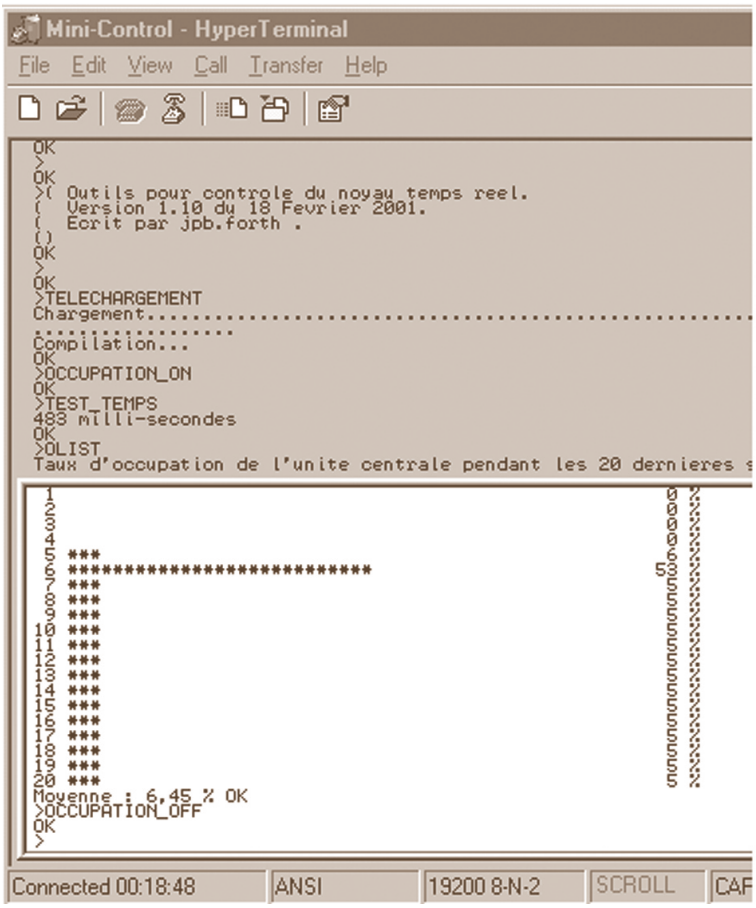

fig. 4 – La machine virtuelle Mini-Control se pilote au travers d'un hyperterminal

Machine virtuelle FORTH avec noyau temps réel

La machine virtuelle Forth à implémenter dans le système fut pratiquement livrée clé en main par un passionné recontré au hasard de la navigation Internet. JPB avait par le passé fait ce travail pour équiper un système basé sur un processeur 68000 (voir lien Web). Il en a donc fourni une adaptation pour Mini-Control.

De façon basique, la machine virtuelle se pilote au travers d'un hyperterminal via le lien série évoqué plus haut. De sa fenêtre, il est possible de donner des commandes (interpréteur), d'en fabriquer de nouvelles (compilateur) directement ou par téléchagement d'un fichier source. La machine virtuelle comprend notamment un gestionnaire de fichier qui utilise une partie de la mémoire flash (comme un flash-disk) et surtout un gestionnaire de tâches temps réel.

Cette dernière caractéristique est particulièrement importante dans le contrôle de processus. En effet, toutes les régulations par calculateur font appel à des algorithmes échantillonnés. Merci la transformée en z!

Etendre le dictionnaire de la machine virtuelle vers le haut n'est plus une tâche mais un jeu! C'est en tout cas ce que démontrent les habitués de ce concept. A peine l'interface pour la souris et pour l'écran VGA implémentés, voilà en retour un environneemnt multi-fenêtrage, des démos 3D, Tetris et d'autres encore.

Mais soyons sérieux, MCL cadence docilement à 25 MHz au coeur du module d'évaluation qui a marqué le point final du projet proprement dit. Ce module d'évaluation est une cible qui a déjà rendu de très bon services dans nos laboratoires. Par exemple pour la modélisation de nouveaux periphériques: CAN-bus, USB, pwm, touch-screen...

Vers le contrôle de processus

A ce stade, la boucle est presque bouclée. Un processeur minimal, modélisé et implémenté dans une FPGA modeste, peut supporter une machine virtuelle simple et efficace. Le fonctionnement irréprochable du module d'évaluation nous a confirmé cette hypothèse. Il reste cependant à démontrer la validité du concept dans une application industrielle en grandeur nature.

Dans les prochains mois, en effet, MCL et sa machine virtuelle vont se réincarner dans une platine qui sera elle aussi minimale (ce qui n'est pas le cas du module d'évaluation existant). De dimensions réduites, elle prendra place dans la glissière d'un mécanisme dont il faudra contrôler la position. Les consignes y seront transmises par liaison USB, ainsi que toutes les modifications de paramètres ou changement d'algorithme.

La machine virtuelle a ceci de caractéristique: son principe d'invariance. Tous les objets sont traités de la même façon, du plus simple au plus complexe, de l'instruction au programme jusqu'à la librairie toute entière.

Trop beau pour être vrai? Il est probable que nous allons tout de même y trouver quelques défauts mais jusqu'à preuve du contraire, le rapport coût-efficacité de MCL dans une application de contrôle de processus reste très favorable.

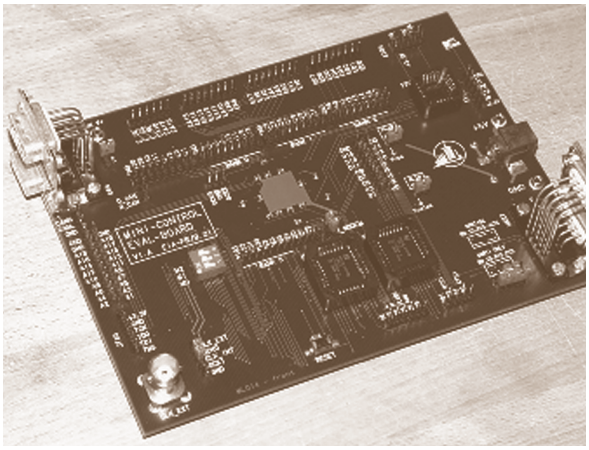

fig. 5 – Le module d'évaluation MiniControl.

Références

Projet MiniControl: HES-SO / Centre de compétences en systèmes intégrés No 00-3-1

Machine virtuelle forth: ***http:// jpb.forth.free.fr*** ■