Conception conjointe logiciel-matériel et microprocesseur embarqué, validation sur plateforme FPGA

Vincent Fristot, Sylvain Huet, Laurent Fesquet, Robin Rolland

Conception conjointe logiciel-matériel et microprocesseur embarqué, validation sur plateforme FPGA

Vincent Fristot, **Sylvain Huet**,
vincent.fristot@grenoble-inp.fr, sylvain.huet@grenoble-inp.fr
Gipsa-lab, 961 rue de la Houille Blanche – BP 46, 38402 Saint Martin d'Hères
Laurent Fesquet, laurent.fesquet@grenoble-inp.fr
Laboratoire TIMA, 46 avenue Félix Viallet, 38031 Grenoble cedex
Robin Rolland, robin.rolland@grenoble-inp.fr
CIME-Nanotech, 3 parvis Louis Néel, BP 257, 38016 Grenoble cedex 1

RESUME : Dans le cadre d'une initiation aux systèmes électroniques intégrés, nous proposons un bureau d'étude de découverte d'un processeur embarqué au cœur d'une chaîne de traitement numérique du signal. A l'aide d'une description VHDL du processeur élémentaire fournie aux étudiants, il est proposé de simuler l'exécution de programmes en assembleur et de mettre en œuvre le flot de conception d'un SOC. Cet enseignement a été apprécié car il permet d'illustrer le fonctionnement du cœur du processeur tout en validant la conception du matériel et du logiciel sur une carte FPGA.

Mots clés : bureau d'étude pour élèves ingénieurs, microprocesseur embarqué, conception conjointe logiciel/matériel, flot de conception de circuits numériques, circuit logique programmable FPGA, SOC.

1 INTRODUCTION

La création de l'école PHELMA de Grenoble-INP en 2008 a été l'occasion de mettre en place un enseignement d'initiation à la filière « Systèmes Electroniques Intégrés » qui débute en deuxième année. Il s'agit de permettre aux étudiants de première année d'école de découvrir les systèmes sur puce (SOC), la conception conjointe logiciel/matériel et de faire fonctionner un microprocesseur élémentaire.

La complexité des processeurs intégrés disponibles nous a conduit à concevoir un processeur pédagogique dédié, qui permet une mise en route rapide de petits programmes écrits en assembleur.

Après un rappel du contexte, nous présentons le sujet du bureau d'étude proposé aux étudiants ainsi que la maquette de développement FPGA support du travail à réaliser.

2 CONTEXTE

2.1 Technologie des systèmes électroniques

L'intégration croissante de dispositifs électroniques au sein d'un même circuit concerne tous les domaines courants : téléphonie, informatique, automobile, avionique, médical... Les systèmes intégrés complexes sont constitués d'une puce matérielle mais aussi de logiciels. A travers le travail proposé, nous allons donner un aperçu d'un système complet permettant le traitement numérique de signaux, ceci à l'aide d'un processeur élémentaire.

2.2 Visualiser le coeur du processeur

Les outils de conception de circuits électroniques numériques permettent de simuler tous les signaux internes. Nous avons donc décrit le comportement du processeur, de ses périphériques en VHDL et disposons d'un processeur générique (dimensionnement paramétrable des bus d'adresses, de données, de la taille des registres) et de son environnement.
Contrairement à la mise en œuvre de microcontrôleurs intégrés existants [2], il devient donc possible d'accéder à la simulation de l'intégralité des signaux internes du processeur ce qui constitue un atout pour comprendre le fonctionnement interne et les détails de l'architecture du processeur.

2.3 Conception rapide et validation sur une carte FPGA

Il est proposé de mettre en œuvre un flot de développement complet qui repose sur les outils suivants :

- **ModelSim** de Mentor Graphics [6] pour la simulation des circuits au niveau fonctionnel et après placement routage,
- **Precision Synthesis** de Mentor Graphics pour la synthèse logique à partir d'une description VHDL,
- **Quartus** d'Altera pour le placement routage, la génération des fichiers de programmation du FPGA.

Ces outils sont appelés à l'aide de commandes en ligne, au travers de scripts préparés par les enseignants ce qui facilite grandement le travail des étudiants. Il est demandé aux étudiants de réaliser les simulations de l'exécution de programmes en assembleur, ce qui équivaut à une exécution des instructions en pas à pas, tout en visualisant le détail du fonctionnement interne du processeur.
Une carte de prototypage FPGA permet de faire fonctionner le dispositif complet de traitement de signaux

analogiques, ce qui renforce l'attrait de l'enseignement, à l'image de la plateforme matérielle de démonstration des turbo-codes [1].

3 SUJET DU BUREAU D'ETUDE

En s'inspirant de l'architecture d'un processeur élémentaire proposée par le fabricant de circuit logiques programmables Altera (DE1_lab_exercises - lab 9 et 10 - documents téléchargeables [3]), nous avons mis au point un bureau d'étude de 4 séances - soit 16h - consacrées à l'étude du processeur et à une application de démodulation d'un signal modulé en frequency shift keying - FSK.
Notre objectif est d'introduire la notion de processeur à travers un processeur élémentaire - le *nanoprocesseur* - et de réaliser un système complet qui permet l'exécution de programmes adaptés à une chaîne de traitement de signaux.

Ce processeur est organisé autour de 8 registres de données (dont un utilisé pour le Program-Counter), une unité arithmétique et logique munie d'un registre accumulateur et d'un registre de résultat, d'un bus externe capable d'adresser 256 mots, le tout étant cadencé par un séquenceur qui permet l'exécution des instructions du programme en 3 à 5 cycles d'horloge, fixée à 50 MHz.
Le processeur constitue le cœur du projet dont l'architecture à partie opérative (alu, registres, bus) et partie de commande (séquenceur) [7] est présentée figure 1:

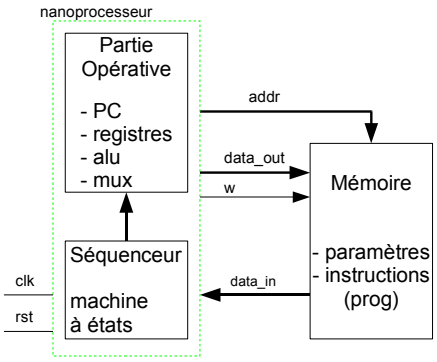

Fig 1. Le nanoprocesseur et son espace mémoire

Nous présentons une démarche structurée de conception de circuits numériques, basée sur une description du processeur et de son environnement ainsi que sur des outils de simulation et de synthèse logique.

3.1 Spécification du processeur

3.1.1 L'environnement du processeur

Les périphériques du processeur à disposition des étudiants sont les suivants :

- des mémoires permettant de stocker les instructions de programme ainsi que les données traitées ;
- différentes entrées/sorties numériques afin de connecter boutons poussoirs, afficheurs, convertisseur AN.

Une horloge (clk) et un signal de remise à zéro (rst) assurent un fonctionnement synchrone.

Ainsi, le processeur communique avec l'extérieur à l'aide d'un bus de 8 bits d'adresses et 16 bits de données; le bus de données est séparé en entrées et en sorties (data_in, data_out).

3.1.2 Jeu d'instructions

Nous choisissons d'implémenter un jeu d'instructions réduit pour réaliser de petits programmes démonstratifs. Voici la liste des 12 instructions du *nanoprocesseur* :

- `mv` : transfert entre registres
- `ldi` : chargement donnée immédiate dans registre
- `add, sub, and` : op arithmétiques entre registres
- `mvnz, mvgt` : transferts conditionnels
- `brnz, brgt, brz, brmi` : branch. conditionnels
- `bra` : branchement inconditionnel

Les instructions sont codées sur 10 bits, dont les 4 bits de poids fort codent la nature de l'instruction.

3.2 Description de l'unité arithmétique et logique (ALU)

Le fonctionnement de l'ALU est précisé à l'aide de la description en langage VHDL suivante :

```
entity ALU is
   port ( a,b : in std_logic_vector(15 downto
0);
   alu_code : in std_logic_vector(1 downto 0);
   r : out std_logic_vector(15 downto 0));
end ALU;

architecture behavior of ALU is
begin
   process (a,b,alu_code)
   begin
      case alu_code is
      when alu_add => r<=std_logic_vector(un
                  signed(b) + unsigned(a));
      when alu_sub => r<=std_logic_vector(un
                  signed(b) - unsigned(a));
      when alu_and => r<=std_logic_vector(un
                  signed(b) and unsigned(a));
      when others => r<=std_logic_vector(unsi
                  gned(b) + unsigned(a));
      end case;
   end process;
end behavior;
```

Fig 2. Description VHDL de l'ALU

Nous avons implémenté trois opérations arithmétiques et logiques de base dans cet ALU. Il est aisé de compléter les fonctionnalités en ajoutant d'autres instructions.

3.3 Architecture retenue

Le nanoprocesseur est constitué d'une partie opérative et d'un séquenceur, le coeur de la partie opérative étant l'ALU qui traite des données de 16 bits.
La partie opérative rassemble principalement les 8 registres de données (16 bits), l'ALU, un multiplexeur qui positionne le bus interne « nanobus ».

Le séquenceur est la partie de commande du processeur. A partir du code de l'instruction à exécuter, et des bits d'état, le séquenceur positionne l'ensemble des signaux de commande de la partie opérative.

La description de la machine à états du séquenceur est fournie aux étudiants, il n'est pas possible de la présenter ici en raison du format de cet article.
L'exécution d'une instruction commence par l'accès au code instruction situé en mémoire.
Le cycle fetch1 permet de sortir la valeur du registre PC sur le bus d'adresses externe pour accéder au code de l'instruction. Le cycle fetch2 est celui de la lecture du code de l'instruction issu de la mémoire. Suivent ensuite 1,2 ou 3 cycles d'exécution selon la complexité de l'instruction.
Ainsi, les instructions sont exécutées en un total de 3 cycles (transferts et branchements), 4 cycles (transfert immédiat ou indirect), 5 cycles (opérations arithmétiques ou logiques). Pour aller plus loin, il est possible d'améliorer ces performances en adoptant une architecture optimisée de type RISC [8].

3.4 Exemple de programme

Nous donnons ci-dessous un exemple de programme d'addition du contenu des registres r0 et r3, le résultat est stocké dans r0.

adresse	code op	instructions	commentaire
0 1	0040 AAAA	`ldi r0,#AAAA`	`charge AAAA dans r0`
2 3	0058 000F	`ldi r3,#000F`	`charge 000F dans r3`
4	0083	`add r0,r3`	`r0 ← r0 + r3`
5	027F	`loop: bra loop`	`boucle infinie`

Fig 3. Premier programme d'application

L'exécution de ce programme par le processeur est simulée figure 9 (voir en fin d'article).
On visualise les signaux d'adresse et de données de la mémoire programme (ROM), l'état d'exécution des instructions, le résultat de l'opération effectuée par l'ALU ainsi que 2 bits d'état relatifs à ce résultat (Z zéro, G négatif). La valeur des 8 registres r0 à r7 est affichée.
Cette simulation agit donc comme un débugger permettant de déceler tout fonctionnement anormal du programme ou toute erreur intervenue lors du codage à la main des instructions de programme.
Le flot de développement utilisé par les étudiants consiste en l'écriture du programme assembleur en binaire (codes opérations) à inclure dans la description du contenu d'une ROM en VHDL.

4 MAQUETTE PEDAGOGIQUE

Le processeur et son environnement sont intégrés dans un circuit logique programmable (FPGA) Cyclone II 2C20 d'Altera qui comprend 18000 cellules d'éléments logiques (LE), 240 kbits de RAM et 315 ports d'entrée-sortie [4].
Nous utilisons le kit de développement DE1 d'Altera [3], qui dispose d'afficheurs 7 segments à LED, de boutons poussoirs, d'une horloge à 50 MHz, de connecteurs d'entrées et sorties logiques, présenté à la figure 4 :

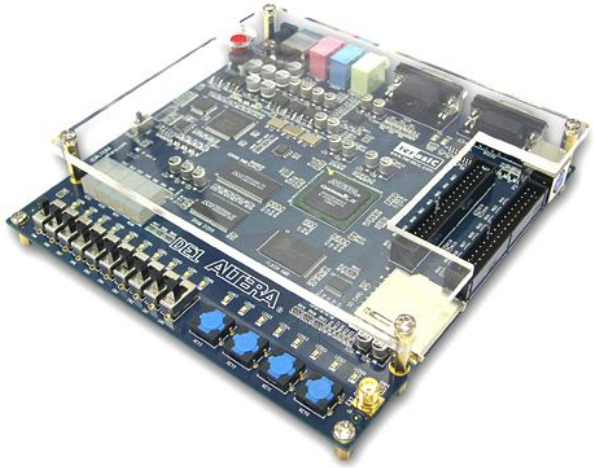

Fig 4. Le kit de développement DE1

Une carte d'extension présentée à la figure 5 a été ajoutée à ce kit afin de traiter des signaux analogiques (acquisition d'un signal modulé en entrée et signal démodulé en sortie). Cette carte permet la réalisation d'un filtre analogique du premier ou du second ordre, la numérisation d'un signal (CAN) et la restitution d'un signal analogique en sortie (CNA) après traitement numérique.

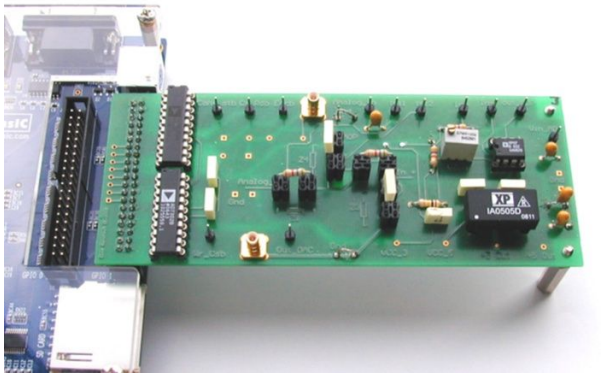

Fig 5. La carte d'extension d'entrée-sortie analogique

Les convertisseurs supportent une fréquence d'échantillonnage de 1MHz (convertisseurs AD7822 – AD7302).
Le développement de cette carte analogique a été nécessaire car les convertisseurs analogiques présents sur la carte DE1 sont dédiés à des signaux audio et ne sont pas accessibles à des mesures (composants CMS).

4.1 Validation sur carte de prototypage

Le *nanoprocesseur* est doté de mémoire ROM pour stocker le programme à exécuter, de mémoire RAM pour stocker des paramètres variables et de ports d'entrées/sorties ou d'interface avec l'extérieur.
La figure 6 représente la hiérarchie des entités intégrées au projet : le processeur central et ses périphériques sont connectés à des modules matériels de pilotage des afficheurs 7 segments et de pilotage des convertisseurs analogiques de la carte d'extension.

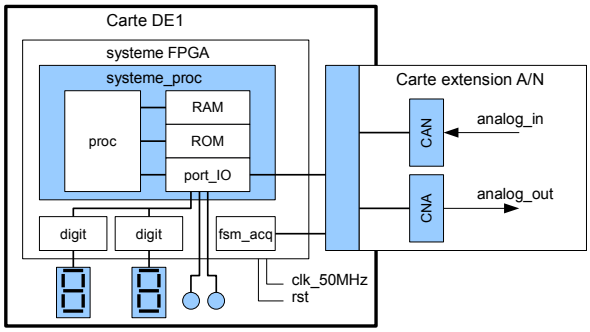

Fig 6. Système embarqué sur la carte de prototypage

4.2 Chaîne de traitement numérique

Nous disposons donc d'une chaine classique de traitement de signal, basée sur le nanoprocesseur :

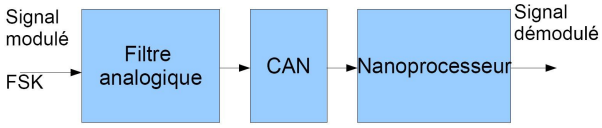

Fig 7. La chaine de traitement de signal

Nous utilisons ce dispositif pour réaliser une application de démodulation d'un signal analogique.

4.3 Application : un démodulateur FSK

Nous disposons d'un signal binaire modulé en FSK (deux fréquences distinctes correspondant aux deux états logiques) représenté à la trace supérieure (figure 8). La carte analogique ajoutée à la plateforme FPGA intègre un filtre passe bas et le convertisseur analogique-numérique permettant l'acquisition du signal au centre de la figure 8.
Le processeur exécute un programme qui compare le signal entrant à un seuil ajustable et mémorise l'état actif pendant au moins une période du signal d'entrée. Le résultat de la détection est représenté – trace inférieure figure 8, c'est le résultat de la démodulation.

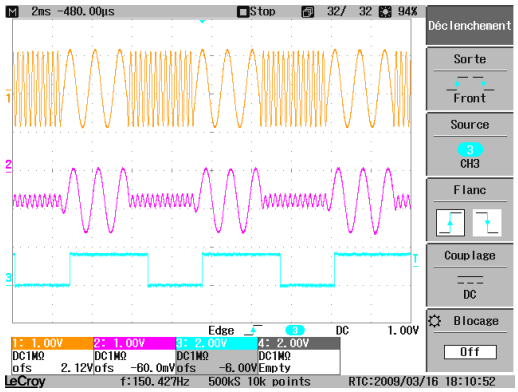

Fig 8. Démodulation à l'aide du nanoprocesseur

5 TRAVAIL DEMANDE

Nous détaillons le déroulement des 4 séances de 4h proposées aux étudiants :

Séance 1 : *L'unité arithmétique et logique*
Le code VHDL décrivant l'ALU est donné, il est demandé de modifier ce code VHDL pour ajouter l'instruction 'not' , de simuler ce nouvel ALU, de comprendre l'architecture générée par l'outil de synthèse « precision » en faisant le lien entre la description comportementale en VHDL et le résultat de la synthèse.

Séance 2 : *Le séquenceur - la machine à états*
Il s'agit de détailler l'exécution d'un programme sur le *nanoprocesseur* afin de calculer la somme des j premiers entiers :

$$S = \sum_{i=0}^{j} i$$

Ecrire ce programme à l'aide des instructions disponibles, l'assembler et le simuler sous modelsim (simulation fonctionnelle).

Séance 3 : *Programme de démodulation FSK*
La démodulation proposée consiste à détecter la présence de la composante basse fréquence obtenue après filtrage analogique passe bas. A partir d'un seuil programmable à l'aide des boutons poussoirs, la valeur du seuil est affichée en hexadécimal sur 2 afficheurs 7 segments. Ecrire le programme réalisant cette fonction.
Comparer le signal entrant avec le seuil, introduire un timer logiciel afin de mémoriser l'état actif pendant au moins une période du signal de la composante BF.

Séance 4 : *Intégration et prototypage du nanoprocesseur*
Le but est de réaliser le prototype complet sur la carte DE1 d'Altera, avec simulation, synthèse et placement routage du système FPGA, afin de générer le fichier de programmation du circuit logique programmable.

6 CONCLUSION

Le bureau d'étude proposé constitue une initiation aux systèmes électroniques intégrés. Plusieurs notions liées à la conception de circuits numériques sont abordées et illustrées.

Sur les 11 groupes de deux étudiants concernés par cet enseignement, tous ont fait tourner de petits programmes en simulation. 7 groupes ont conçu le programme final permettant d'ajuster le seuil à l'aide de boutons poussoirs et l'ont validé sur la maquette DE1.
Deux groupes ont fait fonctionner et ont validé le programme complet de démodulation FSK ; ils ont donc répondu à la totalité du travail demandé.
Les étudiants ont apprécié le travail proposé au cours de ces séances. Plusieurs ont conclu à un bilan positif même s'ils n'ont pas pu terminer faute de temps. Ils ont souligné la diversité des aspects abordés dans ce BE.

A partir de ce sujet, il est possible de faire évoluer le processeur en ajoutant d'autres instructions. L'étude de l'architecture du séquenceur doit aboutir à une optimisation et à une accélération de l'exécution des instructions. Enfin, l'automatisation du processus d'assemblage par le développement d'un assembleur permet de faire le lien entre un développement logiciel et le matériel sous-jacent.

L'ensemble des sources est disponible sur le site du CETSIS : description en VDHL comportemental du processeur, bancs de test (bench) en VHDL, sujet proposé aux étudiants.

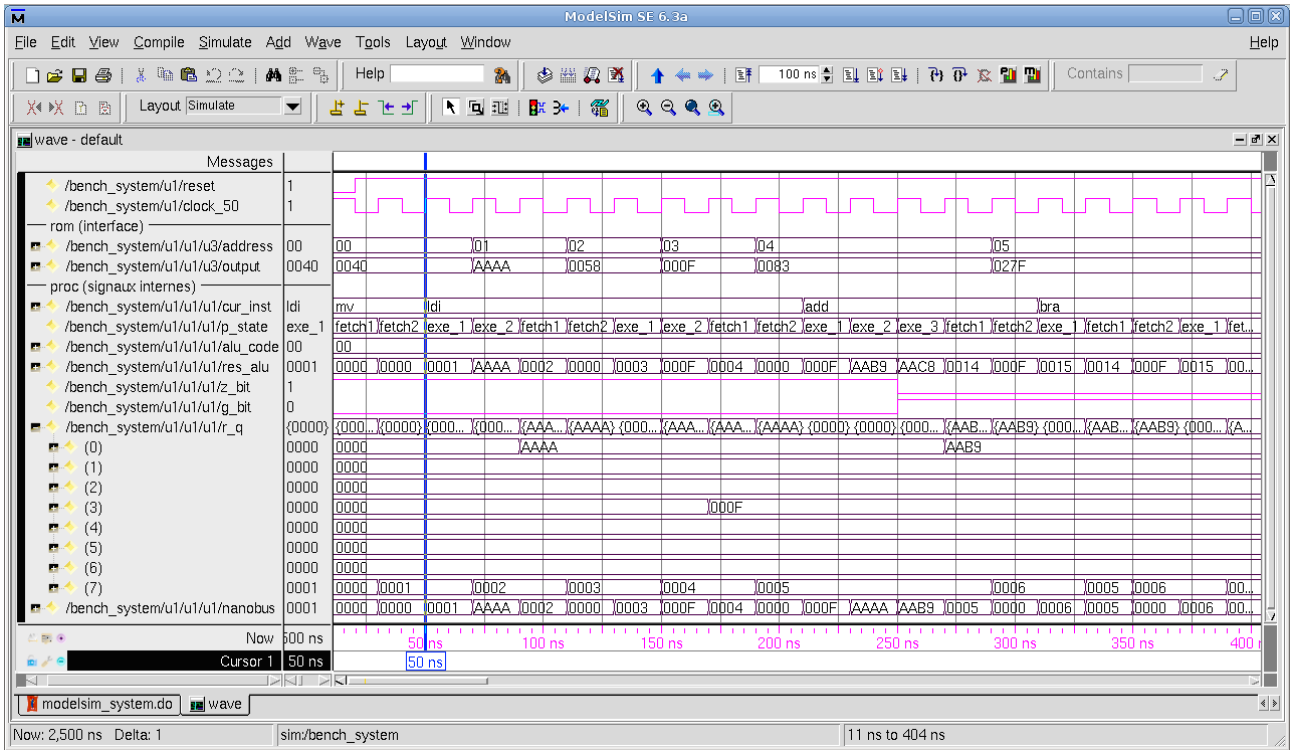

Fig 9. Simulation de l'exécution du programme donné en figure 3

Bibliographie

[1] Ch. Jégo, A. Picart et J. Tousch, "Développement d'une plateforme matérielle de démonstration dédiée aux turbo codes", *Journal de l'Enseignement des Sciences et Technologies de l'Information et des Systèmes (J3EA), Vol.2, 10 (2003).*

[2] H. Sauer, Th. Avignon, M. T. Plantegenest "Une initiation aux microprocesseurs pour les élèves ingénieurs de SupOptique", CETSIS-EEA, Clermont Ferrand, 29-30 Octobre 2001.

[3] Développement sur carte DE1 d'Altera : http://www.terasic.com/downloads/cd-rom/de1/

[4] Data Sheet des circuits de la famille Cyclone II d'Altera : http://www.altera.com/literature/lit-cyc2.jsp

[5] S. Huet, V. Fristot, "Cours de logique tronc commun PET Phelma" : http://huet.phelma.grenoble-inp.fr

[6] www.model.com

[7] Architectures Logicielles et Matérielles, P. Amblard, Dunod, collection Sciences Sup.,2000

[8] Des machines à états aux processeurs, cours d'Electronique numérique intégrée, chap 8, J.L. Danger, http://comelec.enst.fr/tpsp/eni/poly/enich8.html