SORBONNE UNIVERSIT É

M1 SESI

Mod´elisation SystemC d'un RISC-V pipelin´e

Rapport d'avancement
dans le cadre du projet PSESI

Je l'ai importé dans OpenOffice pour ajouter facilement mes commentaires

Etudiants:
M. Timoth´ee Le Berre
M. Louis Geoffroy Pitailler
M. Kevin Lastra

Encadrante :
Mm. Daniela Genious

Nous souhaitons remercier
Mme. Daniella Genius, M. Pirouz Bazargan Sabet et M. Franck Wajsb¨ urst
qui nous ont ´enorm´ement aid´e lors de la r´ealisation de ce projet.

Cf e-mail; en ce que me concerne, je n’ai pas L’impression d’avoir beaucoup aidé

Table de Mati`eres

1 Introduction

Je mettrai le suivant dans une sous-section et dirai tout d'abord en une ou deux phrases ce que vous avez prévu de faire

En 1971, Intel sort son premier microprocesseur, l'Intel 4004 bas´e sur une architecture 4 bits CISC (Complex Instruction Set Computing).

Un peu plus petit ?

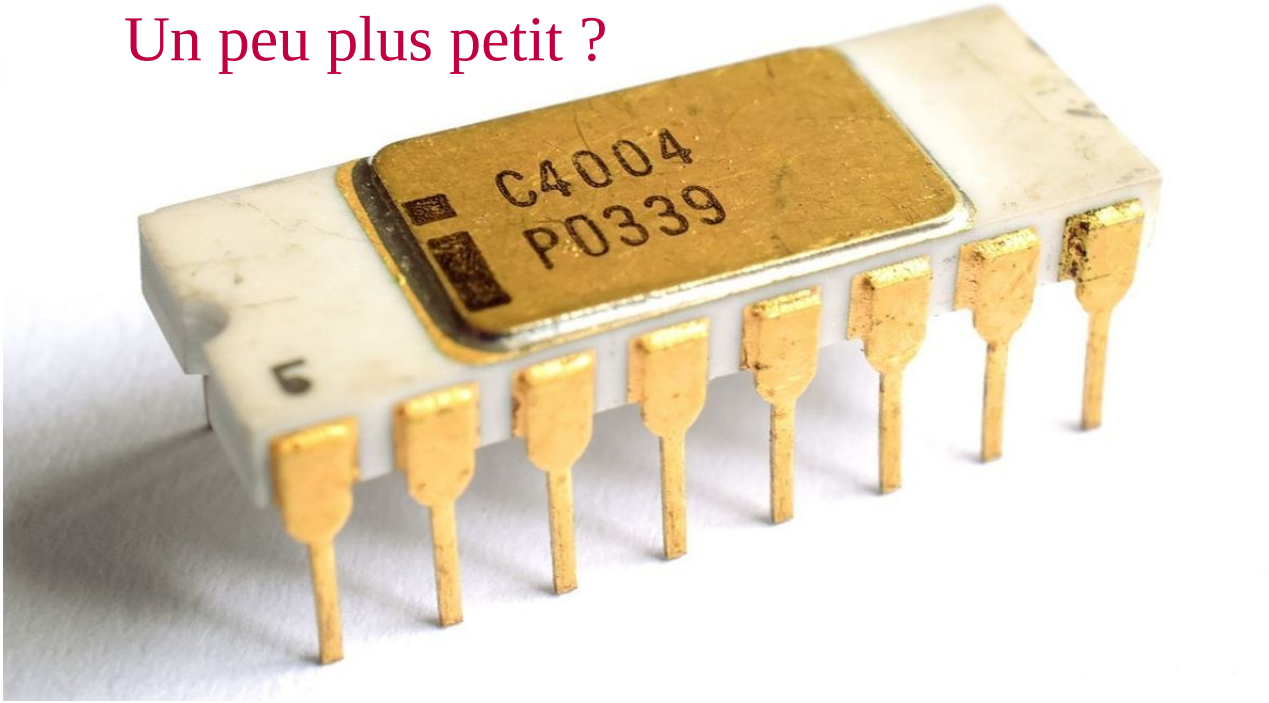

Figure 1: Intel 4004, source : https://www.lesnumeriques.com/cpu-processeur/l-intel-4004-premier-processeur-du-fondeur-fete-aujourd-hui-ses-50-ans-n171113.html

C'est du passé, non ?

Les Processeurs CISC vont largement dominer le march´e jusque dans les ann´ees 80 o`u une nouvelle architecture va faire son apparition : l'architecture RISC.
Les architectures CISC sont beaucoup plus complexes que les risques, en effet ces derniers impl´emente des fonctions tr`es complexes en mat´eriel. On peut par exemple citer l'Intel 8086 qui impl´emente mat´eriellement des instructions permettant de faire des comparaisons entre des chaines de caract`eres.
Les architectures RISC au contraire impl´emente uniquement des fonctions basiques et simple de mani`ere mat´eriel, la philosophie du RISC ´etant en effet de laisser les taches complexes au compilateur.
RISC ´etait `a l'origine un projet men´e par David Patterson `a l'Universit´e de Berkely en Californie entre 1980 et 1984.
Cette architecture va vite montrer de gros avantages par rapport `a l'architecture CISC et de nombreux projets vont se d´evelopper en se basant dessus.
En 1981, le MIPS (Microprocessor without Interlocked Pipeline Stages) -qui se base sur une architecture de type RISC- va faire son apparition `a l'Universit´e de Standford.
La technologie MIPS va ˆetre commercialis´ee `a partir de 1984 et sa premi`ere impl´ementation, le R2000, va devenir l'un des processeurs les plus utilis´es pour l'embarqu´e.

idem

La figure n’est pas référencée dans le texte

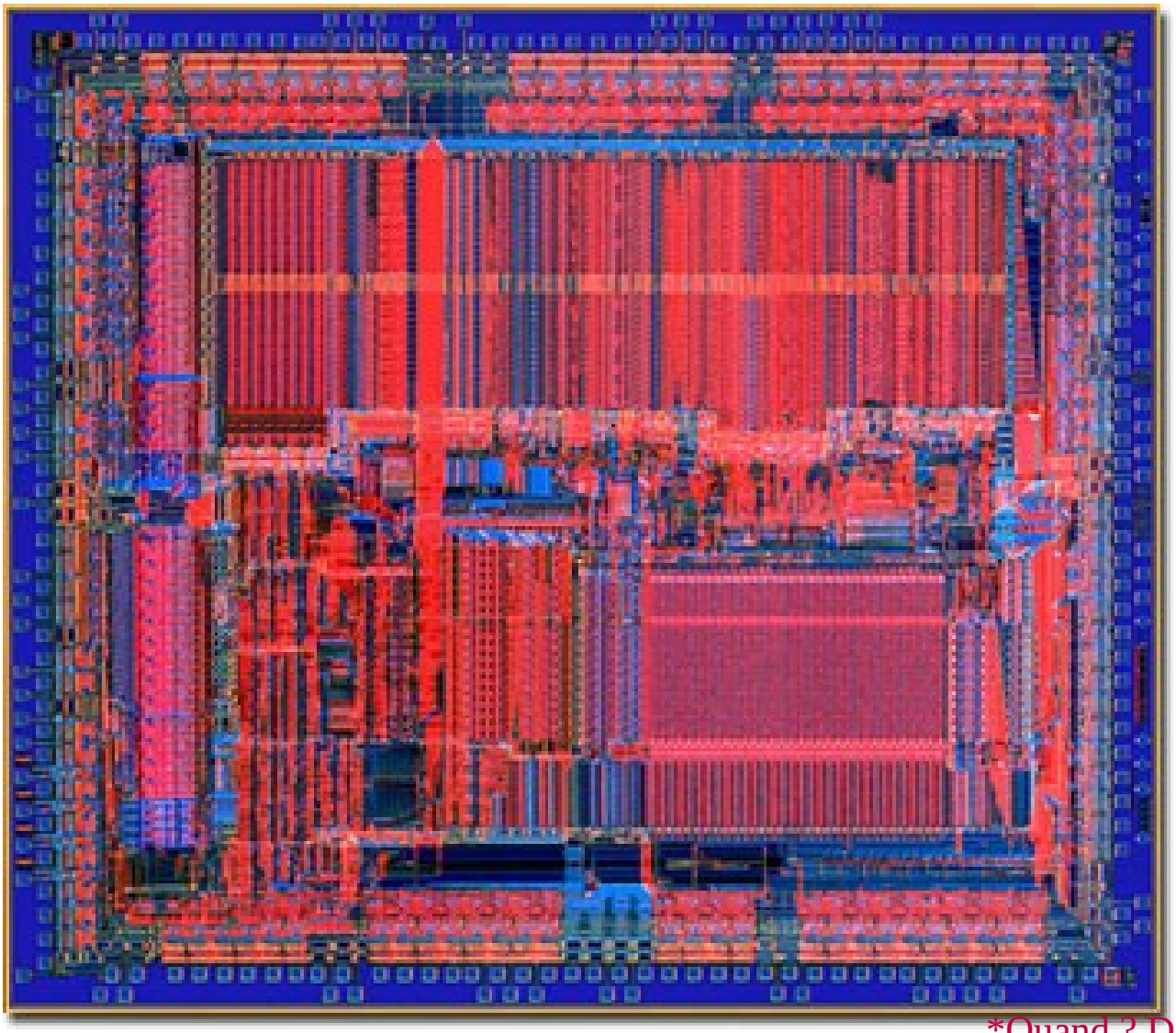

Figure 2: Intel 4004, source : https://www.cpushack.com/MIPSCPU.html

*Quand ? Dans les années 90 je crois. Voir avec Pirouz.

Grˆace `a sa simplicit´e de compr´ehension et son efficacit´e, le MIPS s’impose comme un standard dans l’enseignement de l’architecture processeur, c’est pourquoi le Lip6 l’a choisi comme base de son enseignement*.
N´eanmoins, cette architecture a fait son temps et commence peu `a peu ` a se montrer trop vieille** C’est pourquoi il a ´et´e d´ecid´e de changer la base des cours de Sorbonne Universit´e utilisant en utilisant l’architecture RISC-V, une architecture libre, facile `a impl´ementer, proche du MIPS et surtout utilis´e `a l’heure actuelle.

Au cours de notre premier semestre de M1 SESI, nous avons eu l’occasion d’impl´ementer une architecture ARMv2a en VHDL. Cela nous a ´enorm´ement plu et lorsque Mme.Genius a propos´e une impl´ementation d’une architecture RISC-V en systemC nous avons tout de suite postul´e afin de pouvoir participer `a ce projet.

Ce projet nous a permis de nous familiariser avec le jeu d’instruction RISC-V et avec le langage SystemC.
RISC-V ´etant l’une des impl´ementations de RISC les plus importantes - aux cˆot´es de l’architecture ARM -, il est tr`es int´eressant d’en ´etudier son fonctionnement.
Ne voulant pas simplement r´ealiser une architecture scalaire et voulant aller plus loin que ce que nous avions

**Trop veille dans quel sens?
C’est juste qu’enfin qqc de comparable, largement accepté
Et surtout libre est arrivé

d´ej`a eu l'occasion de faire en VLSI avec l'architecture ARM, nous avons d´ecid´e d'avancer rapidement sur le projet dans le but de finir d´ebut mars l'impl´ementation scalaire et d'ensuite pouvoir nous concentrer sur une impl´ementation SS2*.

N´eanmoins, apr`es discussion avec notre encadrante, il a ´et´e d´ecid´e de commencer d'abord par l'ajout d'une partie Kernel `a notre design et d'ensuite passer `a l'impl´ementation SS2 s'il nous restait du temps.

*Ici, certains lecteurs ne savent pas qu'est-ce que c'est un SS2, et que vous aviez prévu de l'implémenter. P.e. le mettre dans la toute première phrase de l'introduction : RISC-V Pipeliné à 5 étages, SS2, kernel instr.

2 Objectifs

L'objectif premier de notre projet ´etait d'impl´ementer une architecture RISC-V 32 bits pipelin´ee ` a 5 ´etages sans extension, dans le but de remplacer l'architecture MIPS du Lip6. Les ´etages de notre impl´ementation sont les mˆemes qu'une MIPS R3000, `a savoir :

- IFETCH
- DECODE
- EXECUTE
- MEMORY
- WRITE-BACK

Un dessin aiderait

Le langage impos´e pour cette r´ealisation est systemC*.Notre impl´ementation devait donc ˆetre proche de l'architecture MIPS que nous avons ´etudi´e en cours, d'**ø**`l'impl´ementation sous forme d'un pipeline 5 ´etages. Une fois l'impl´ementation termin´ee, nous devions ´egalement mettre en place une plateforme de TP**.

Finalement il a ´et´e d´ecid´e que la mise en place de la plateforme de TP n''etait pas prioritaire et que l'impl´ementation de la partie Kernel primait.
Notre objectif est donc de mettre en place une architecture RISC-V scalaire avec partie Kernel et s'il nous reste du temps d'impl´ementer un SS2 et enfin de mettre en place une plateforme de TP.

Un dessin aiderait pour illustrer les étapes/dépendences entre elles

*mettre une référence exa www.accellera.org

**plus prudent : dire que c'est un volet si le temps Reste, mais qu'il faut que le tout tourne sur Les machines des salles de TP. OK je vois vous les dites dans la suite

3 Situation actuelle du projet

À l'heure où nous ´ecrivons ce rapport, nous avons fini la partie scalaire et le nettoyage du code en accord avec les conventions que nous avons pris pour le nom des signaux.*
Notre processeur ne poss`ede pas encore de bypass qui sont en cours de cr´eation, mais il est capable de compiler du C ou de l'assembleur.
L'impl´ementation RISC-V que nous avons surnomm´e v1.0 est `a ce jour totalement finalis´e. Nous avons en effet r´ealis´e multiple tests C tel que la suite de Fibonacci r´ecursive ou encore un algorithme de calcul de PGCD.

*p.e. dire un peu plus si le temps reste,
Aussi pourquoi les reste des noms a été changé
(pour être plus parlant etc.)

3.1 R´esum´e graphique

Tous les membres de notre projet ont particip´e à l'ensemble de la conception, mais certains membres se sont plus concentr´e sur certaines taches. Ainsi, le tableau ci-dessous indique qui a majoritairement contribu´e à la r´ealisation de la tache.

	Janvier	F´evrier	Mars	Avril	Mai	Juin
Documentation RiscV	X					
Conception CORE RiscV v1.0						
Etage IFETCH		X				
Etage DECODE		X				
Etage EXEC		X				
Etage MEMORY		X				
Etage WRITEBACK		X				
D´ebogage CORE		X				
Nettoyage du programme			X			
Conceptions Caches						
Mis `a jour Mips3000R		X				
Documentations Mips3000R		X				

X : Repr´esente le mois où la tache a ´et´e finalis´ee.

Je n'avais pas toute de suite compris de diagramme, avant de voir que les 3 colonnes de la fin sont en fait redondants; supprimer le mois de juin; mettre ici (ou mieux dans un Deuxième diagramme du même style les dates butoirs des taches encore à accomplir.

Timoth´ee Le Berre
Louis Geoffroy Pitailler
Kevin Lastra

4 Probl`emes rencontr´es lors de la conception

Plusieurs probl´ematiques ont surgi depuis le d´ebut du projet mais gr^ace au travail d''equipe et a la bonne r´epartition des taches, nous avons pu passer outre. Passer outre = éviter, non? Vous voulez dire maitriser?

La premi`ere difficult´e majeure a ´et´e celle de rendre fonctionnelle le code SystemC du MIPS qui nous a ´et´e fourni, en effet les conventions utilis´ees ´etaient difficiles `a comprendre et le code ´etait peu document´e. Une fois le code rendu compilable, il a fallu le d´ecortiquer compl`etement ce qui a pris beaucoup de temps.
La deuxi`eme difficult´e rencontr´ee aura ´et´e la synth`ese de la sp´ecification RISCV, il a en effet fallu faire le trie entre ce que nous comptions impl´ementer ou pas et il nous a fallu bien comprendre le jeu d'instruction afin de l'impl´ement´e correctement dans DECOD. Tout à fait; soit dans le rapport soit lors de la soutenance, pouvez-vous détailler un peu ce tri ?
Enfin le d´ebogage du Core complet pour compiler des programmes C nous aura pris du temps, en effet nos tests assembleur se sont vite mont´e insuffisant et il nous a fallu trouver ce que ne fonctionnait pas lors de la compilation de nos programmes C.
Pour finir une difficult´e que nous rencontrons actuellement, mais qui devrait bient^ot ^etre r´esolu est la mise en place des bypass, en effet nous avions mis en place tout un syst`eme d'invalidation dans le banc de registre

dans le but de geler le pipeline en cas de d´ependance de donn´ees, mais nous nous sommes aper¸cu que cette invalidation n´´etait plus n´ecessaire une fois les bypass correctement impl´ement´e.

5 Objectifs restants

Le tableau ci-dessous fait office de Roadmap, il s'agit d'un r´ecapitulatif graphique des objectifs que nous nous sommes fix´es et les deadlines correspondantes.

	Fin F´evrier	Fin Mars	Fin Avril	Fin Mai	Juin
D´ebogage & impl´ementation des bypass		X			
Impl´ementation de la partie Kernel				X	
Impl´ementation SS2 & d´ebogage complet					X

Les Bypass devraient donc ˆetre bient ˆot termin´e, il nous faudra ensuite d´ecortiquer tout le cours de Mr Franck Wajsburst qui nous a ´et´e fourni, afin d'impl´ementer la partie Kernel.
Cela risque de nous prendre beaucoup de temps ´etant donn´e que nous n'avons jamais impl´ement´e de partie Kernel, nous allons donc tˆatonner jusqu'a atteindre un r´esultat satisfaisant.
Enfin s'il nous reste du temps nous passerons le processeur en Super-scalaire, mais il nous est assez difficile de nous projeter jusque-l`a ´etant donn´e que nous n'avons pas de r´eelle id´ee de quels vont ˆetre les difficult´es d'impl´ementation de la partie Kernel.

D'accord c'est le diagramme que j'ai proposé en regardant la page 6; mais il faut mettre des noms /qui fait quoi ?

P.e. dire que l'implémentation du superscalaire s'orientera à Celle vu en cours de ARCHI 1; vous pouvez donner Un exemple (e.g. diagramme généré avec l'outil de Quentin) Pour illustrer; pour ceux qui ne connaissent pas : c'est quoi Un buffer d'instruction, un dispatch etc.

Il manque la fameuse "procédure de récette", Parler des codes test que vous avez écrits, Pourquoi en C etc., sans pourtant les lister

6 Bibliographie :

https://en.wikipedia.org/wiki/IBM System/370

https://en.wikipedia.org/wiki/IBM document processorsIBM 801

Waterman, A., Lee, Y., Patterson, D., Asanovic, K., level Isa, V. I. U. (2014). The RISC-V instruction set manual. Volume I: User-Level ISA', version, 2.

Waterman, A., Lee, Y., Avizienis, R., Patterson, D. A., Asanovic, K. (2015). The risc-v instruction set manual volume 2: Privileged architecture version 1.7. University of California at Berkeley Berkeley United States.

Asanovi´c, K., Patterson, D. A. (2014). Instruction sets should be free: The case for risc-v. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146.

Utting, M., Kearney, P. (1992). Pipeline specification of a MIPS R3000 CPU. Technical Report 92-6, Software Verification Research Centre, Department of Computer Science, University of Queensland.

David A. Patterson John L. Hennessy (2021). Computer organization and design RISC-V edition, second edition.

Jurij Šilc, Jurij Silc, Borut Robic, Theo Ungerer (1999). Processor architecture : From dataflow to superscalar and beyond.