Laboratoire n°2

Conception d'un microprocesseur

Professeur:
Yvon Savaria
yvon.savaria@polymtl.ca

Inspiré du travail de:
Mickaël Fiorentino
mickael.fiorentino@polymtl.ca

Chargé de laboratoire:
Raphael Rowley
raphael.rowley@polymtl.ca

Répétiteur:
Jörg Ehmer
jorg.ehmer@polymtl.ca

Automne 2024

TABLE DES MATIÈRES

1 INTRODUCTION

La complexité d'un circuit numérique intégré à très grande échelle, tel qu'un microprocesseur, serait impossible à gérer avec les outils de conception de bas niveau (dessin des masques) que nous avons utilisé dans le laboratoire n°1. Pour mieux gérer la complexité des circuits intégrés, tout en gardant les coûts de développement dans des limites raisonnables, les outils de synthèses associés aux langages de description architecturale comme le VHDL ou le Verilog, ainsi que les outils de placement et routage automatisés permettent de générer le dessin des masques d'un circuit à partir de sa description comportementale de haut niveau.

1.1 OBJECTIFS DU LABORATOIRE

Ce deuxième laboratoire consiste à concevoir un microprocesseur. Il a pour objectif de vous familiariser avec la conception, la simulation, la synthèse, et l'implémentation physique de systèmes numériques complexes à l'aide d'outils de conception automatisés. Vous passerez ainsi à travers le flot de conception standard, tel que schématisé à la FIGURE 1. En particulier, ce laboratoire vous permettra de :

— Concevoir un modèle de processeur avec le langage VHDL.
— Réaliser des simulations comportementales (modèle VHDL), et temporelles (*netlist* post-synthèse et post-implémentation) du processeur.
— Réaliser la synthèse logique du processeur en utilisant la technologie en 45 nm du kit GPDK045 .
— Réaliser les placement & routage automatisés à partir de la *netlist* post-synthèse du processeur.

De plus, les points suivants ne sont pas exigibles, mais restent à notre sens intéressants à étudier si le temps vous le permet :

— Inclure les structures de testabilité (*Design For Test* (DFT)) au processeur et générer des vecteurs de tests.
— Évaluer la consommation d'énergie du processeur à partir de ses informations post-implémentation et de l'activité qu'il a généré en simulation temporelle.

La documentation du processeur contient toutes les informations nécessaires à sa conception. En vous aidant des fichiers initialement fournis et de la documentation, vous concevrez d'abord chacun des modules composant le processeur. Vous utiliserez alors ces modules pour concevoir le *core* pipeliné du processeur. Le langage VHDL sera présenté en classe.

Une fois le processeur modélisé au niveau comportemental, vous suivrez le tutoriel numérique, qui vous guidera dans l'apprentissage des techniques de simulation, de synthèse, de testabilité, de placement & routage, et d'évaluation de la consommation d'énergie avec les outils *Modelsim*, *Genus*, *Modus*, *Innovus* et *Voltus* à travers l'exemple d'un compteur BCD. Le modèle VHDL de ce dernier vous est également fourni si vous souhaitez vous pratiquer sur un cas simple avant de vous attaquer au processeur.

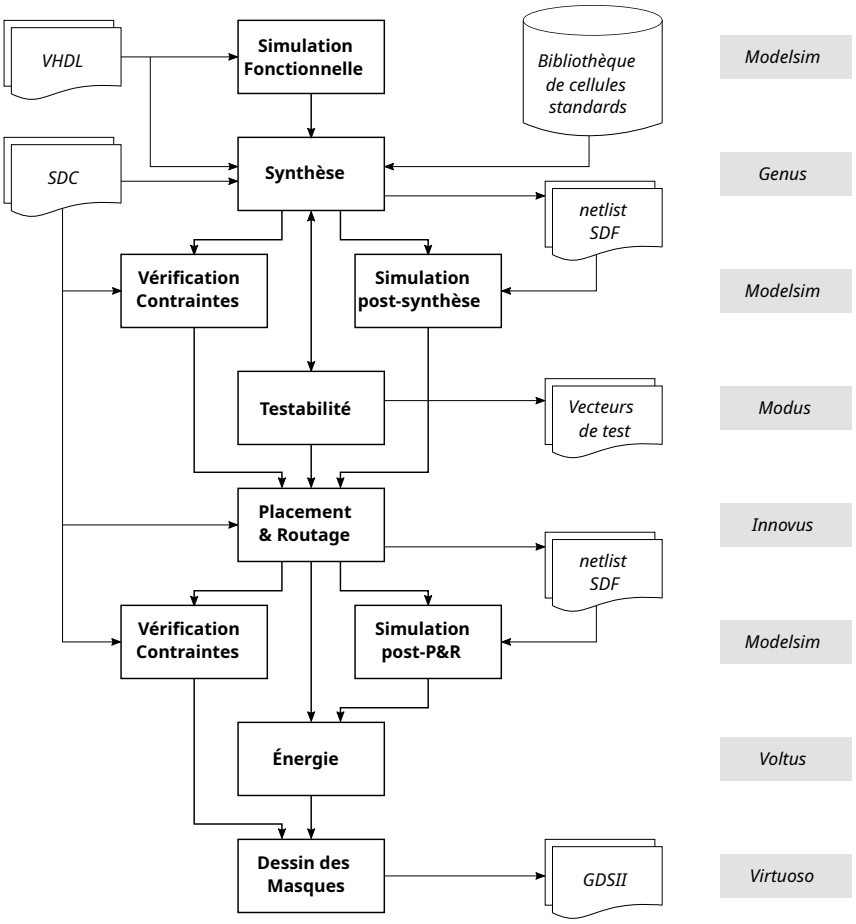

FIGURE 1 – Flot de conception

1.2 DOSSIER DE TRAVAIL

Le dossier de travail associé au laboratoire n°2 est disponible sur un répertoire *git*. Téléchargez ce répertoire sur votre station de travail avant de commencer :

```
% cd /export/tmp/user/Labs
% git clone https://git.step.polymtl.ca/ele8304/lab2.git
```

Le dossier de travail est organisé de la façon suivante :

```
asm/.......................... # Programmes assembleur
scripts/...................... # Automatisation du flot de conception
sources/...................... # Code source VHDL
constraints/.................. # Fichiers de contraintes
simulations/.................. # Simulations comportementales et temporelles
implementation/............... # Synthese logique, placement et routage
doc/.......................... # Documentation, Rapport
```

1.3 BARÈME DE NOTATION

TABLEAU 1 – Barème de notation

Modules	**/5**
Core	**/9**
Implémentation	**/4**
Qualité du rapport	**/2**
Performances (bonus)	**/2**

Le TABLEAU 1 donne la répartition des points du laboratoire. 14 points sont alloués à la conception et à la simulation des modules et du *core* en VHDL. 4 points sont alloués au flot de conception (synthèse, testabilité, placement & routage, vérifications). 2 points sont alloués à la qualité du rapport. Enfin, **2 points bonus** sont alloués à l'évaluation des performances et de la consommation d'énergie du processeur.

À noter, **1 point** de la section Modules sera évalué pendant la séance Lab 2.2 le **23 octobre 2024**. Ensuite, **2 points** de la section Core seront évalués pendant la séance Lab 2.3 le **20 novembre 2024**. Finalement, **1 point** de la section Implémentation sera évalué pendant la séance Lab 2.4 le **27 novembre 2024**. Plus de détails seront fournis pendant le Lab 2.1 quant aux évaluations en laboratoire.

1.4 PÉNALITÉS DE RETARD

La date limite de remise du laboratoire est précisée sur Moodle. Les pénalités de retard P_{retard} sont calculées en fonction du numbre h d'heures de retard tel que :

$$P_{retard} = 0.5 \times \left[1 + \left(\frac{h}{24}\right)^2\right]$$

1.5 RAPPORT

Votre rapport doit inclure tous les éléments nécessaires à la justification de vos analyses. C'est-à-dire vos résultats de simulation, vos résultats de synthèse et d'implémentation, et tout autre élément que vous jugerez pertinent pour démontrer votre bonne compréhension du travail réalisé. Les 2 points alloués à la qualité du rapport concernent la précision du vocabulaire employé, la qualité de la présentation et des figures, ainsi que la qualité de la syntaxe et de l'orthographe. Une seule remise par groupe est nécessaire. On vous demande de remettre un fichier compressé (`.zip`) sur Moodle avant l'échéance de remise du laboratoire contenant le dossier de travail complété de votre travail, en particulier :

— Votre rapport au format PDF.
— Vos codes sources VHDL (modules, *core*, et bancs d'essais).
— Vos scripts *tcl* pour la simulation, la synthèse, et l'implémentation.
— Vos résultats post-synthèse et post-implémentation (*netlist*, rapports de STA, DRC, LVS, rapports d'énergie, surface occupée etc.)

1.5.1 RECOMMANDATIONS SUR LE CONTENU DU RAPPORT

Il n'est pas obligatoire de respecter les recommandations, mais on vous le conseille grandement. Avoir un rapport clair contribuera grandement à une bonne note [1].

Pour chaque module à tester (adder + modules dont le code est donné), on vous conseille d'avoir le plan suivant :

— Explication du fonctionnement haut niveau du module (à quoi il sert).

— Explication du rôle des signaux d'entrée et sortie (quelles valeurs ils doivent prendre dans quelle situation etc...)

— Simulation avec captures d'écran : Chaque simulation / capture doit être accompagnée d'une petite explication ou au moins description de ce qu'il se passe / de ce qui est intéressant.

En règle générale, il vaut mieux nous donner trop d'explications que pas assez, pour qu'on soit sûr que vous aillez bien compris.

Pour les étage du pipeline, on ne demande pas de simulation pour chaque étage, mais chaque étage doit être brièvement expliqué (fonctionnement haut niveau et signaux d'entrée et sortie).

Pour le test du processeur complet, le mieux est de suivre différentes instructions dans le pipeline en expliquant comment elle est interprétée par chaque étage (quels signaux de sortie sont set/ pour quelle raison) en faisant les captures d'écran correspondantes. Pour cette consigne, il faut au moins une instruction de chaque type pour illustrer les différents cas possibles, on vous demande donc au moins : une instruction arithmétique (ADD), une instruction de branchement (BEQ), une instruction SW et une instruction LW (on vérifiera que la valeur lue est bien la valeur écrite). Cela représente 4 instructions en tout et elles sont toutes inclues dans le programme riscv_basic.s. Bien sûr, il faut aussi montrer le déroulement de l'instruction spécialisée.

Pour la synthèse, vous pouvez montrer votre script et expliquer ce qu'il effectue. Puis, il pourrait être bien de montrer comment vous vous êtes assurer que le processeur fonctionne post-synthèse.

Pour le Place & Route, montrez le script et commentez le. Donnez un résumé des résultats de Place & Route. Comment vous vous êtes assurer du fonctionnement du processeur post-implémentation?

Remarque—*Nous souhaitons avoir vos commentaires sur les difficultés que vous avez rencontrées ainsi que le temps d'apprentissage que vous avez passé sur les outils durant la réalisation de ce laboratoire. Nous sommes particulièrement intéressés aux lacunes pouvant subsister dans la documentation.*

1. Ces recommandations sont inspirées des anciens chargés de laboratoire Justin Pabot et Timothée Matéo Trembly.

2 DIRECTIVES

2.1 MODULES

Dans cette première partie, on vous demande de concevoir les modules composant le processeur. Les explications relatives au fonctionnement des modules, leurs schémas de principe, et leurs interfaces VHDL, sont détaillés dans la documentation du processeur. En particulier, on vous demande de :

— Comprendre et concevoir le modèle VHDL de tous les modules décrit dans la documentation du processeur, en respectant avec *exactitude* leurs interfaces VHDL, et de valider leur fonctionnement à l'aide d'une simulation comportementale et d'un banc d'essai. (**5 pts**)

Afin d'accélérer votre avancée, certains modules vous sont déja fournis, en particulier :

— Le banc de registres
— Le *Program Counter*
— L'Unité Arithmétique et Logique (ALU)

Il vous reste donc à créer le modèle VHDL de l'adder 32 bits, ainsi que les bancs d'essais des 4 modules.

Pendant la séance Lab 2.2, votre compréhension des modules et la qualité de vos bancs d'essai seront évalués (**1 point sur 5**).

2.2 CORE

Dans cette partie, nous vous demandons de concevoir le *core* du processeur en utilisant les modules réalisés précédemment. Le fonctionnement du *pipeline* est détaillé dans la documentation. En particulier, on vous demande de :

— Concevoir le modèle VHDL du *core* tel que décrit dans la documentation, en respectant avec *exactitude* son interface VHDL. (**4 pts**)
— Concevoir le module d'instruction spécialisée et l'intégrer au processeur (Voir 2.2.1). (**2 pts**)
— Valider son fonctionnement à l'aide d'une simulation comportementale et d'un banc d'essai montrant l'exécution du *benchmark* **riscv_basic.S** fourni dans le dossier de travail. (**3 pts**)

Pour pouvoir exécuter un programme avec votre processeur, nous vous fournissons également un banc d'essai. Notez que celui-ci instancie la mémoire double port (`dpm.vhd`) fournie dans le dossier de travail. Il est également possible d'effectuer des tests complémentaires avec vos propres programmes assembleurs en utilisant le Makefile pour la compilation. Assurez-vous que le fichier d'initialisation de la mémoire d'instruction (`*.hex`) reflète les modifications que vous avez apportées à votre programme après la compilation.

Pendant la séance Lab 2.3, votre avancement sur la conception du Core et la qualité de vos bancs d'essai seront évalués (**2 point sur 9**).

2.2.1 INSTRUCTION SPÉCIALISÉE

Le endianness du standard RISC-V est le little-endian. Le processeur conçu dans ce laboratoire demeure isolé du monde externe mais il serait souhaitable qu'il puisse un jour communiquer sur Internet. Cependant, la suite TCP/IP utilisée par Internet est le big-endian. Il faut donc une instruction spécialisée pour convertir des données d'un format à l'autre.

L'instruction spécialisée se nomme **eswp** et est une instruction I-TYPE (voir la documentation du processeur pour plus de détails). Le format de l'instruction est comme montré à la figure 2.

Bits	31-20	19-15	14-12	11-7	6-0
Rôle	I-imm[11:0]	rs1	funct3	rd	Opcode

FIGURE 2 – Format de l'instruction **eswp**

L'instruction **eswp** utilise le format I-TYPE. Lorsque `funct3` est un nombre pair, la conversion se fait de little-endian vers big-endian. Lorsque `funct3` est un nombre impair, la conversion se fait de big-endian vers little-endian. Si `funct3` est positif (en complément à 2), l'instruction convertit le contenu du registre `rs1`. Puis, si `3` est négatif, l'instruction convertit la valeur immédiatte de format I-IMM. Le résultat de la conversion est envoyé au registre `rd`.

À noter, le `opcode` de **eswp** est toujours : 1010101.

Le module responsable de l'exécution du **eswp** doit se retrouver à l'étage **EX**. Aussi, le module doit être intégré de sorte à ce que votre processeur soit toujours à l'épreuve des différents conflits.

2.3 IMPLÉMENTATION

Dans cette partie on vous demande de réaliser l'implémentation physique du processeur en utilisant la technologie 45 nm du kit GPDK045 de Cadence. En particulier, on vous demande de :

— Réaliser la synthèse logique du processeur, **sans inclure la testabilité**. Celle-ci peut également être réalisée si le temps vous le permet, et sera un motif de bonification. Vérifiez que les contraintes temporelles sont respectées avec l'outil STA, et validez le fonctionnement du processeur à l'aide d'une simulation temporelle post-synthèse. Utilisez le même banc d'essai que pour la simulation comportementale, et mettez en évidence les délais dans le circuit. (**2 pts**)

— Réaliser le placement & routage du processeur. Vérifiez que les contraintes temporelles sont respectées avec l'outil STA, vérifiez l'intégrité de son dessin des masques avec les vérifications DRC et LVS, et validez le fonctionnement du processeur à l'aide d'une simulation temporelle post-placement-routage (même banc d'essai). (**2 pts**)

Pendant la séance Lab 2.4, votre avancement sur l'implémentation et la qualité de vos scripts seront évalués (**1 point sur 4**).

2.4 PERFORMANCES - BONUS

Dans cette partie on vous demande d'évaluer les performances de votre processeur post-implémentation. C'est à dire, de mettre en parallèle le temps d'exécution du *benchmark* **riscv_basic.S** avec l'énergie consommée pendant son exécution. En particulier, on vous demande de :

— Réaliser la simulation post-implémentation du processeur en exécutant le *benchmark* **riscv_basic.S**. Utilisez le même banc d'essai que pour la simulation comportementale. Enregistrez l'activité générée par la simulation temporelle dans un fichier `vcd`.

— Déterminer la performance de votre processeur en Million d'Instructions Par Secondes (*MIPS*), en utilisant les informations fournies par votre compteur de performance.

— Évaluer la puissance moyenne P (en mW) consommée par votre processeur pendant l'exécution du programme à partir de l'activité enregistrée en simulation post-implémentation. Le ratio $MIPS/P$ sera la métrique de référence.

N.B. : Notez qu'il est possible d'effectuer des modifications de dimensionnement (incluant des modifications du circuit et du flot de conception) de façon à obtenir les meilleures performances possibles. Si vous faites ce choix, vérifiez le respect des contraintes temporelles avec l'outil STA.

Le nombre de points accordés pour une partie/la totalité des démarches présentées ci-dessus sera laissée à l'appréciation des chargés de laboratoire, selon le soin qui leur est apporté et l'avancement moyen de chaque équipe.