Projet Microprocesseur (2018-19)

Cours *Systèmes numériques : de l'algorithme aux circuits*

Timothy Bourke <Timothy.Bourke@ens.fr>

9 octobre 2018

Le but de ce projet est de créer en partant de zéro un microprocesseur capable d'exécuter le programme d'une montre digitale telle que celle présentée dans le cours. Pour cela, vous devrez créer un simulateur de circuit, capable d'exécuter tout circuit digital synchrone (correct). Ce simulateur sera ensuite utilisé pour simuler l'exécution de votre microprocesseur, qui sera suffisamment complexe ou spécialisé pour pouvoir exécuter le programme de la montre digitale.

Au cours du projet, vous serez donc amenés à créer (dans cet ordre) :
— Un simulateur capable d'exécuter tout circuit digital synchrone
— Une architecture de microprocesseur (avec son jeu d'instructions)
— Le programme de la montre digitale, dans le jeu d'instructions de votre microprocesseur

1 Le projet

Le simulateur de *net-list*

Votre simulateur devra pouvoir simuler tout circuit digital synchrone (correct), à partir de sa description dans le langage de net-list et d'une liste d'entrées (lues sur l'entrée standard ou dans un fichier). Votre simulateur devra rejeter les circuits qui possèdent une boucle combinatoire. Il devra gérer toutes les fonctionnalités du langage de *net-list*, notamment les primitives mémoires (ROM et RAM).

Le simulateur prendra en entrée :
— la *net-list* du circuit
— le nombre n de cycles de simulation
— les valeurs des entrées sur n cycles

et retournera :
— la valeur des sorties calculées sur n cycles

Afin de simplifier votre travail, votre simulateur prendra en entrée une *net-list* dans un langage très simple, représentée comme une liste non ordonnée de portes logiques et d'opérations sur des nappes de fils. Cette *net-list* sera générée par un compilateur, appelé MiniJazz et disponible à l'adresse https://github.com/inria-parkas/minijazz/releases/tag/v0.3.0, à partir d'une description de plus haute-niveau hiérarchique. La description de ces langages est disponible sur le site web.

Le microprocesseur

Il s'agit de concevoir un microprocesseur capable d'exécuter un programme faisant fonctionner la montre digitale du cours. Vous pouvez choisir de réaliser un microprocesseur généraliste utilisant un jeu d'instruction générique en vous inspirant de jeu d'instructions existantes ou de réaliser un microprocesseur plus adapté au problème, en ajoutant par exemple des instructions pour faciliter le comptage modulo 24.

La première phase consistera à spécifier l'architecture du microprocesseur ainsi que son jeu d'instructions. Outre les choix classiques sur l'architecture de votre microprocesseur (mémoire, taille et nombre des registres, etc.), vous devez aussi penser aux problèmes concrets d'entrée-sortie (pour afficher l'heure par exemple).

Vous pouvez également choisir d'implémenter un processeur compatible MIPS ou RISC-V. Dans ce cas, il devra supporter soit les instructions de base du MIPS R300 [1], soit les instructions RV32I du RISC-V [2]. Vous devrez aussi respecter les différentes conventions des langages (p. ex., pour MIPS, le registre R0 doit toujours contenir 0).

Dans un second temps, vous devrez exécuter votre processeur sur votre simulateur. Vous devrez également programmer un assembleur pour faciliter l'écriture de programmes dans le jeu d'instructions du microprocesseur. On supposera que le programme exécuté par le microprocesseur est stocké dans une ROM au début de la simulation pour simplifier les choses.

Application basique : La montre digitale

Votre montre digitale doit pouvoir gérer les secondes, les minutes, les heures, les jours, les mois et les années. Le programme de démonstration devra en plus avoir deux modes de fonctionnement :

- Un mode horloge dans lequel l'heure du système est utilisée pour initialiser la montre et les secondes défilent en temps réel.
- Un mode de défilement rapide dans lequel la montre avance au rythme le plus rapide permis par votre simulateur.

Applications avancées

Si vous maitrisez bien les fondements du sujet, n'hésitez pas à aller plus loin en implémentant les applications plus intéressantes, comme pour la cryptographie avec instructions dédiée (p. ex., la « carryless multiplication » ou l'implémentation « bitslice » du chiffrement par bloc AES), ou en ajoutant les optimisations d'architecture (p. ex., un pipeline ou une mémoire cache).

2 Détails pratiques

Organisation

Vous commencerez le projet en travaillant individuellement sur le simulateur de net-list. Ensuite, nous vous rassemblerons par groupes de 3 ou 4 pour travailler ensemble sur un seul système.

1. https://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2010/ue/LI221-2010oct/td_tp/memento2010.doc.pdf
2. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-54.pdf

Vous êtes libres de choisir les outils et les langages que vous désirez pour l'implémentation des différents programmes, dans la limite où le résultat est facilement compilable et utilisable sous plusieurs systèmes d'exploitation (Linux, Mac OS X, Windows). Un analyseur syntaxique du langage de *net-list* est fourni en OCaml, mais il peut facilement etre adapté dans un autre langage si besoin est.

Plusieurs séances du cours seront en partie dédiées au projet. Vous êtes encouragé à discuter de vos choix techniques par mail ou en passant à mon bureau (passage saumon, bureau S12).

Échéances et travaux à rendre

Tous les documents à rendre doivent être typographiés. Les sources des programmes demandés devront être envoyées dans une archive, contenant un fichier `README.md` avec les instructions pour compiler le programme, l'exécuter (par exemple les arguments du programme) et toute autre information que vous jugerez utile. Les rapports et archives devront être envoyés par mail à `Timothy.Bourke@ens.fr`, au plus tard à minuit de la date limite, pour que je puisse avoir le temps de juger votre travail et de vous aider.

Les échéances du projet sont les suivantes :

13 novembre Rendu du simulateur (individuel)
Vous devrez fournir un simulateur capable de simuler tous les exemples fournis avec le TP 1, et un (court) rapport qui devra contenir un bref compte-rendu sur le fonctionnement de votre simulateur et les difficultés rencontrées.

18 décembre Rapport intermédiaire (court) sur le microprocesseur (groupe)
Ce rapport comprendra la spécification de l'architecture de votre microprocesseur et de son jeu d'instructions.

20 janvier Rapport final (groupe)
Ce rapport contiendra les éléments (éventuellement mis à jour si des changements ont été faits) présents dans les deux premiers rapports intermédiaires, auxquels vous ajouterez ce qui concerne l'exécution du microprocesseur sur le simulateur (avec par exemple le programme de la montre dans le jeu d'instructions du processeur). Vous devrez fournir avec ce rapport :
- La version finale de votre simulateur
- Un assembleur pour votre processeur
- Un script permettant de lancer la simulation de votre microprocesseur à partir de sa description et du code source du programme de la montre.

22 janvier Soutenance (groupe)
Vous présenterez l'ensemble de votre projet. Vous devrez faire une démonstration de votre montre digitale. Les détails pratiques vous seront communiqués ultérieurement.