

## TD 3 - CONTRÔLE CÂBLÉ OU MICROPROGRAMMÉ

Du contrôle câblé au contrôle microprogrammé d'un multiplieur entier et d'un multiplieur flottant.

Page web : <http://www-rocq.inria.fr/~acohen/teach/archi.html>

### Exercice 3.1 - Le multiplieur entier

Support : microcontroleur\_mask.lgf et mult4\_mask.lgf.

Le contrôle vu en cours est un circuit rapide, puisqu'à chaque étape, le jeton qui véhicule l'information de contrôle ne doit passer qu'à travers une bascule D et éventuellement un circuit combinatoire. Cependant il est également peu flexible. À chaque modification du circuit à contrôler, il faut souvent revoir entièrement le circuit de contrôle. Lorsque la performance n'est pas un facteur critique mais que le coût l'est, il peut être intéressant de trouver une méthode plus flexible qui ne nécessite pas de changements importants lors d'une modification du circuit à contrôler. C'est notamment le cas du contrôle *microprogrammé*. Le contrôle vu en cours est appelé contrôle *câblé*.

L'atout du contrôle microprogrammé est d'implémenter le contrôle sous forme d'un « programme » — baptisé *microprogramme* — stocké dans une mémoire (ROM) ; en cas de changement du comportement du composant, voire de sa structure, il suffit de charger un nouveau microprogramme. Un microprogramme est constitué de *microinstructions* qui activent les signaux de contrôle. Chaque microinstruction indique la valeur de chaque signal de contrôle du composant (un bit par signal). Il existe trois types de microinstructions : des branchements inconditionnels, des branchements conditionnels et des microinstructions "simples". Chaque type de microinstruction, que ce soit un branchement ou une microinstruction simple définit la valeur des signaux de contrôle. Chaque microinstruction comporte 4 champs : un champ indiquant le type de la microinstruction, un champ indiquant la condition (ce champ est utilisé par les branchements conditionnels), un champ indiquant la valeur des signaux de contrôle (la taille en bits de ce champ est égale au nombre de signaux de contrôle du composant), et un champ indiquant l'adresse de destination (ce champ est utilisé par les branchements).

Les différents composants d'un circuit de contrôle microprogrammé sont indiqués dans la figure 11. On va considérer que chaque microinstruction peut activer jusqu'à 4 signaux de contrôle simultanément. On suppose également qu'il y a jusqu'à 4 conditions provenant du circuit à contrôler et dont peuvent dépendre les microinstructions. On suppose enfin que l'adresse de la microinstruction est codée sur 8 bits (la mémoire

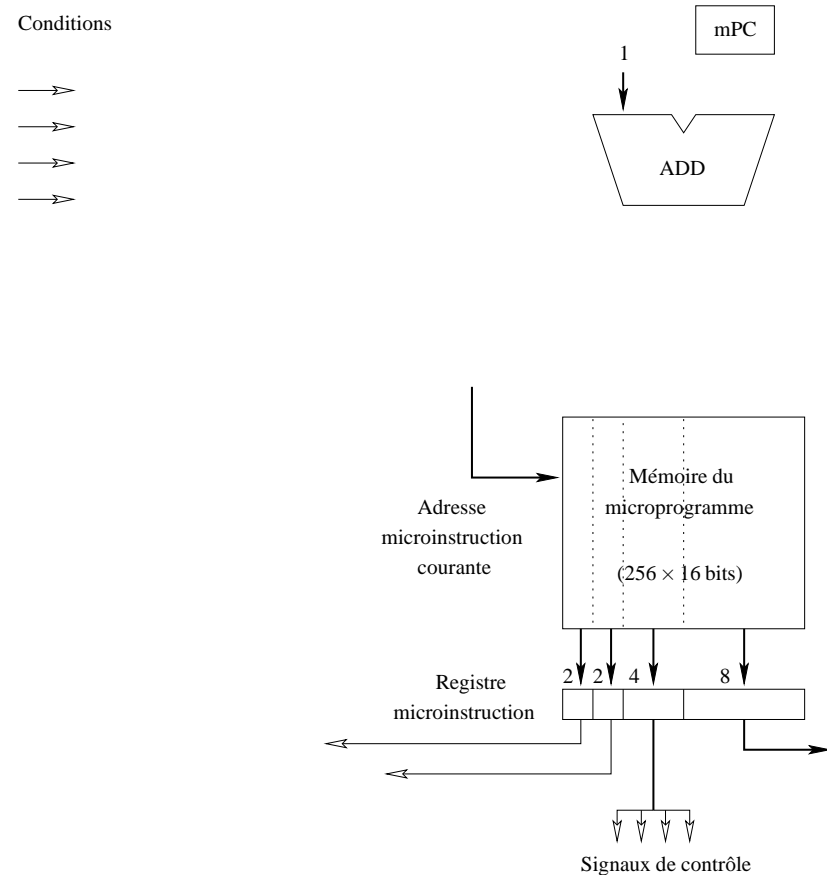


FIG. 11 – Microcontrôleur

peut contenir 256 microinstructions). Chaque microinstruction contient 16 bits. Les microinstructions sont stockées dans une ROM de largeur 16 bits (on utilisera la SRAM8K de DigLog en mode *read-only*). Les bits d'une microinstruction se subdivisent ainsi : les bits 15 et 14 indiquent le type de l'instruction. Les codes des microinstructions sont les suivants : code 01 pour une microinstruction simple, code 10 pour une microinstruction de branchement inconditionnel, et code 11 pour une microinstruction de branchement conditionnel ; le code 00 signifie « Stop » (fin du microprogramme). Les bits 13 et 12 indiquent le numéro de la condition dont dépend le branchement si la microinstruction

est un branchement conditionnel (si la condition vaut 1, le branchement est pris). Les bits 11 à 8 correspondent aux signaux de contrôle numéro 3 à 0 ; il y a un bit par signal de contrôle ; lorsque le bit est à 1 le signal sera activé, il est à 0 sinon. Les bits 7 à 0 indiquent le numéro de la microinstruction de destination (son emplacement dans la mémoire) si la microinstruction est un branchement. Enfin, un registre contient la microinstruction courante. On considérera que la première microinstruction du microprogramme est placée à l'adresse 0 de la mémoire, en supposant qu'un signal *RESET* commande le démarrage à cette adresse 0. Les microinstructions du programme sont placées à des adresses consécutives en mémoire. Quand une microinstruction simple est exécutée ou qu'un branchement conditionnel n'est pas pris, le programme passe à l'instruction suivante. Il y a un additionneur pour calculer l'adresse de l'instruction suivante.

On souhaite réaliser un circuit de contrôle microprogrammé et l'utiliser ensuite pour contrôler le multiplieur entier vu en cours, puis le multiplieur flottant du TD 2.

**Question 3.1.1**

Les composants nécessaires à la réalisation du microcontrôleursont fournis dans le fichier *microcontroleur\_mask.lgf*. Relier ces composants entre eux, en rajoutant éventuellement quelques portes logiques lorsque c'est nécessaire, pour réaliser un circuit de contrôle microprogrammé(penser à utiliser le fichier *wire.lgf* contenant diverses formes de routage pré-dessinées).

**Réponse**

Le schéma complet du microcontrôle est donné dans la figure 12. On notera que l'adresse de la microinstruction courante est conservée dans le registre mPC alors que l'adresse de la microinstruction suivante est dans le registre de microinstruction.

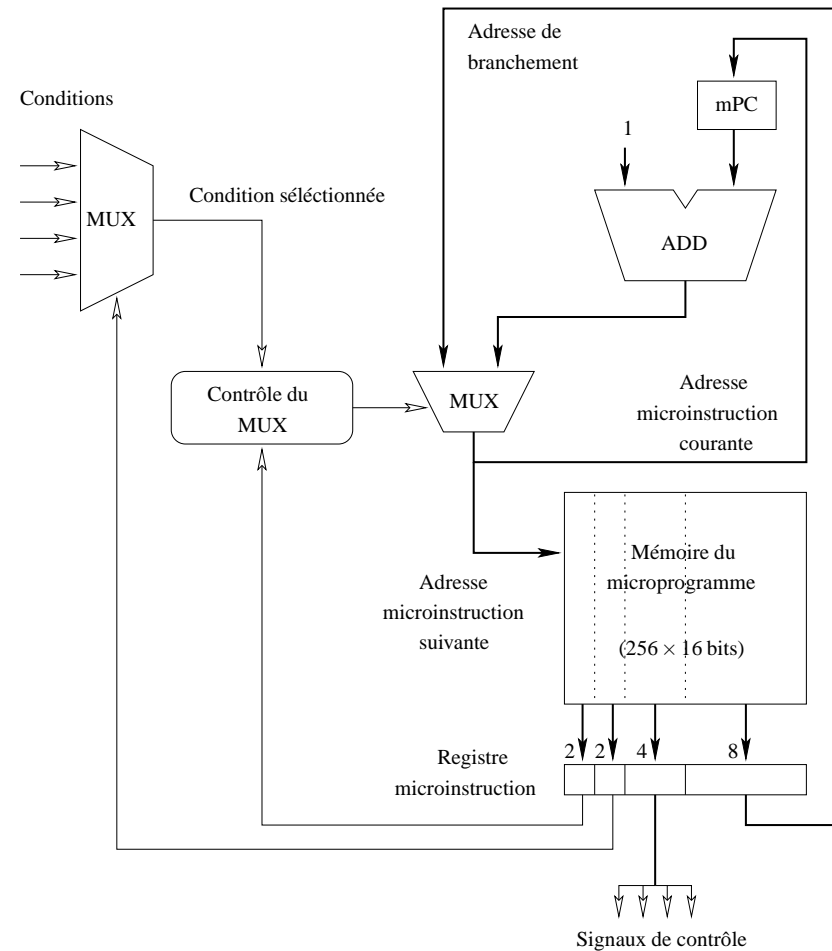


FIG. 12 – Schéma du microcontrôle

Pour le circuit, voir le fichier *microcontroleur.lgf*.

## Question 3.1.2

On veut tester ce circuit avec une boucle contenant deux microinstructions qui alternent de façon régulière entre deux signaux de contrôle quelconques. Coder ces microinstructions en binaire, puis en hexadécimal, et les placer dans la SRAM. Exécuter ce microprogramme.

## Réponse

Pour la première instruction, on choisit d'activer le premier signal de sortie — bits 15 et 14 à 01 — et de ne pas effectuer de branchement — bits 9 et 8 à 01 ; en binaire 0100000100000000, en hexadécimal  $x4100$ . Pour la deuxième instruction, on active le deuxième signal de sortie — bits 15 et 14 à 10 — et on effectue un branchement inconditionnel — bits 9 et 8 à 10 — à l'adresse 0 — bits 7 à 0 à 00000000 ; en binaire 1000001000000000, en hexadécimal  $x8200$ .

Voilà le fichier `microcontr_leur.lgf` et les deux fichiers contenant le microprogramme, `microcontr_leur_h1.ram` (octets de poids fort) et `microcontr_leur_lo.ram` (octets de poids faible). La SRAM est normalement pré-initialisée.

## Question 3.1.3

Le masque `mu1t4_mask.lgf` contient le circuit du multiplicateur vu en cours ; on identifie notamment tous les signaux de contrôle définis en cours, ainsi qu'un compteur qui fournit un signal de condition indiquant si la multiplication est terminée (ce signal s'appelle =3). Associer chacun des bits du champ des signaux de contrôle à l'un des signaux de contrôle du multiplicateur entier vu en cours, puis écrire le microprogramme de contrôle du multiplicateur. Coder puis tester ce microprogramme en vérifiant que le multiplicateur effectue correctement l'opération de multiplication.

## Réponse

Le microprogramme correspondant à l'automate fini vu en cours est le suivant :

- Adresse 0. microinstruction simple, activation de RAZ et Woperande ;  
 Adresse 1. branchement conditionnel à 4 si OPERANDE0 vaut 1, aucun signal activé ;  
 Adresse 2. branchement conditionnel à 5 si le signal =3 vaut 1, aucun signal activé ;  
 Adresse 3. branchement inconditionnel à 1, activation de Décalage et Incrémentation ;  
 Adresse 4. branchement inconditionnel à 2, activation de Wresultat ;  
 Adresse 5. branchement inconditionnel à 5, activation de Fin.

initialisation	itération	addition	fin
4100	C004	D005	8401
		8202	8805

Voilà les fichiers `mu1t4_lgf`, `mu1t4_h1.ram` et `mu1t4_lo.ram` (la SRAM est normalement pré-initialisée).

## Question 3.1.4

Le compteur 2-bits peut être vu comme un élément du contrôle du multiplicateur ; à ce titre, on souhaite le supprimer et implémenter directement le décompte dans le microprogramme. Réaliser le nouveau contrôle du multiplicateur entier.

## Réponse

Il suffit de dérouler les 4 itérations du microprogramme ; de ce fait, le signal =3 devient caduc.

Le microprogramme correspondant à l'automate fini déroulé est le suivant :

- Adresse 0. microinstruction simple, activation de RAZ et Woperande ;  
 Adresse 1. branchement conditionnel à 3 si OPERANDE0 vaut 1, aucun signal activé ;  
 Adresse 2. branchement inconditionnel à 4, activation de Décalage et Incrémentation ;  
 Adresse 3. branchement inconditionnel à 2, activation de Wresultat ;  
 Adresse 4. branchement conditionnel à 6 si OPERANDE0 vaut 1, aucun signal activé ;  
 Adresse 5. branchement inconditionnel à 7, activation de Décalage et Incrémentation ;  
 Adresse 6. branchement inconditionnel à 5, activation de Wresultat ;  
 Adresse 7. branchement conditionnel à 9 si OPERANDE0 vaut 1, aucun signal activé ;  
 Adresse 8. branchement inconditionnel à 10, activation de Décalage et Incrémentation ;  
 Adresse 9. branchement inconditionnel à 8, activation de Wresultat ;  
 Adresse 10. branchement conditionnel à 12 si OPERANDE0 vaut 1, aucun signal activé ;  
 Adresse 11. branchement inconditionnel à 1, activation de fin ;  
 Adresse 12. branchement inconditionnel à 11, activation de Wresultat ;

4100	C003	8404	8202	C006	8407	8205
C009	840A	8208	C00C	880B	820B	

Charger `mu1t4_unroll_h1.lgf`, `mu1t4_unroll_lo.ram` dans le microprogramme de `mu1t4.lgf`.

## Exercice 3.2 (facultatif) - Le multiplicateur flottant

En reprenant les résultats de l'analyse de l'exercice 1.5, réaliser le contrôle microprogramme du multiplicateur flottant.

## Réponse

La multiplication des mantisses — 24 bits par 24 bits — suit un schéma plus rapide que celui du multiplicateur précédent. Le multiplicateur ne comporte qu'un seul registre 48 bits jouant simultanément le rôle d'opérande et de somme partielle des produits bit-à-bit. Le microcontrôleur est également simplifié : il ne dispose pas d'instructions conditionnelles et exécute successivement les 24 étapes de la multiplication. Voir les fichiers `mu1t.lgf` et `mu1t.ram` (la SRAM est normalement pré-initialisée).