CPU 16Bits

Cours d'architecture des ordinateurs L3 Université de Cergy-Pontoise
Professeur : J.Lorandel, M.A.Khelif

Auteurs : Quentin Gerard, Matthieu Vilain

April 2018

Résumé

Ce projet est réalisé dans le cadre du cours d'architecture des ordinateurs, enseigné en Licence 3 par monsieur J.Lorandel à l'Université de Cergy-Pontoise. L'objectif de ce projet est d'implémenter un processeur permettant de faire des opérations sur des mots de 16bits en VHDL pour pouvoir l'exécuter sur une carte FPGA.

1 Description du processeur

Composition Le processeur sera composé de registres permettant de stocker des mots de 16bits, d'un multiplexeur permettant de re-diriger les flux de données dans les différents composants du processeur, d'un alu faisant des opérations sur les mots stockés dans les registres et d'une unité de contrôle (réalisée sous la forme d'une machine à états) décodant les instructions en entrées et commandant les opérations faites sur les autres composants. Ces composants sont reliés entre eux suivant le schéma ci-dessous :

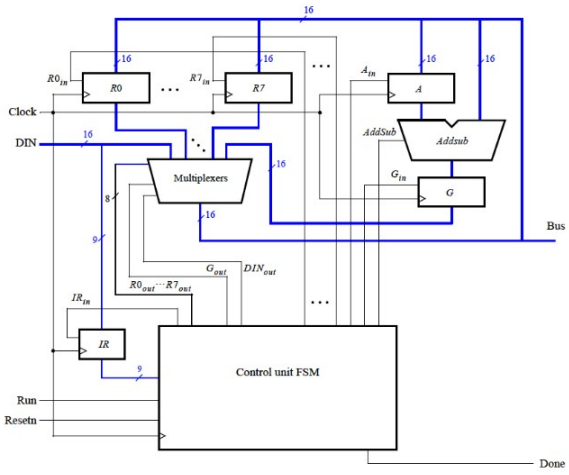

FIGURE 1 – Schéma global du processeur

Fonctionnement Le processeur reçoit un mot de 16bits sur l'entrée Din. Les 9 premiers bits de l'instruction est le Code OP. Les 3 premiers bits de ce code correspondent à l'opération qui sera effectuée. Elles sont réparti comme suit :

Opération	Code	Description
ADD	000	Addition de l'ALU
SUB	001	Soustraction de l'ALU
MULT	010	Multiplication de l'ALU
AND	011	ET logique de l'ALU
OR	100	OU logique de l'ALU
NOT	101	NON logique de l'ALU
MV	110	Fait une copie du contenu de Rx dans Ry
MVI	111	Fait une copie de Din dans Rx

Les 3 bits suivant du Code OP correspondent au premier registre à utiliser (Rx) et les 3 derniers au second registre utilisé (Ry).
Le Code OP 111 000 000 permet donc de mettre le l'entrée Din donc le registre 0 ou encore 000 000 001 permet de faire l'addition entre la valeur du premier registre et celle du second.

2 L'ALU

L'ALU permet de faire des opérations sur les données stockées dans les registres. Elle est composée des opérations :

— Addition N-bits
— Soustraction N-bits
— Multiplication N-bits (non fonctionnel au moment de l'écriture du rapport)
— ET logique
— OU logique
— NON logique

3 Le multiplexeur

Le multiplexeur permet simplement de diriger les signaux vers les différents registres.

Selection	Registre
0000	R0
0001	R1
0010	R2
0011	R3
0100	R4
0101	R5
0110	R6
0111	R7
1000	G
1001	Din

4 Unité de Contrôle (FSM)

L'Unité de contrôle est le composant qui va permettre de contrôler l'ALU, le Banc de Registre ainsi que le Multiplexeur. Il va activer ou désactiver ses signaux en sorties en fonction de l'instruction reçu du Registre d'Instruction. Comme nous pouvons le voir sur le schéma du processeur, l'Unité de Contrôle est interfacé de la manière suivante :

— `IR_s` : Signal de sortie sur 1 bit activant l'écriture dans le registre d'instruction
— `A_s` : Signal de sortie sur 1 bit activant l'écriture dans le registre A, premier termes de l'ALU
— `G_s` : Signal de sortie sur 1 bit activant l'écriture dans le registre G, résultat de l'ALU

— `Ri_s` : Signal de sortie sur 8 bits activant l'écriture dans un des registre du banc de registre
— `Bus_sel` : Signal de sortie sur 4 bits activant l'entrée correspondante dans le Multiplexeur
— `ALU_sel` : Signal de sortie sur 3 bits sélectionnant l'opération sur l'ALU
— Done : Signal de sortie sur 1 bit signalant la fin de l'exécution de l'instruction en cours
— Ir : Signal entrant sur 9 bits, contenant l'instruction à exécuter
— Run : Signal entrant sur 1 bit, signalant la présence de nouvelle données dans le bus d'entrée (Din)
— Reset : Signal entrant sur 1 bit permettant de réinitialiser l'état de l'Unité de Contrôle

L'Unité de Contôle est une machine à état fini (FSM), elle est donc composé de :
— Un Registre d'état, stockant l'état actuelle de l'Unité de Contrôle
— Une Fonction de Génération, générant les sorties en fonction de l'état actuel, et des entrées (Cas Mealy) dans le cas de notre Unité de Contrôle
— Une Fonction de Transition, calculant l'état futur de l'Unité de Contrôle en fonction de l'état actuelle et des entrées

4.1 Les États de L'UC

Notre Unité de Contrôle possède 3 états fondamentales :
— Lecture dans le Registre d'instruction (IR)
— Exécution de l'instruction (cf. Liste des Instructions)
— Instruction exécutée (DONE)

Hors certaines instructions prennent 3 cycles pour s'exécuter et d'autres seulement 1 cycles. C'est le cas des instructions utilisant l'ALU qui on besoin de 3 étapes pour s'exécuter, et seulement 1 pour les instructions MV et MVI. Nous allons donc différencier ces différents types d'instructions dans notre UC. Nous appellerons l'état :
— MV : L'état où l'UC exécute une instruction MV
— MVI : L'état où l'UC exécute une instruction MVI
— ALU0 : L'état où l'UC exécute le premier cycle d'une instruction sur l'ALU
— ALU1 : L'état où l'UC exécute le deuxième cycle d'une instruction sur l'ALU
— ALU2 : L'état où l'UC exécute le troisième et dernier cycle d'une instruction sur l'ALU

Nous avons donc en tout 7 états possible pour l'UC. L'état actuel de l'UC est inscrit dans un registre que l'on appellera Registre d'état. Le Registre d'état implémenté est un registre classique à reset asynchrone, actualisant sa valeur à chaque mouvement d'horloge par la valeur en entrée.

4.2 Fonction de Transition

La fonction de transition est la fonction de l'UC qui va permettre de déterminer en fonction de l'état actuel de l'UC et des entrées, l'état futur de l'UC. Voici le diagramme d'états de l'UC :
Le tableau des transition avec $CODEOP = C_2\ C_1\ C_0$:

L'implémentation de la fonction de transition dans l'UC est réalisé par description comportementale.

4.3 Fonction de Génération

La fonction de génération de l'UC va déterminer l'état des signaux de sorties en fonction de l'état et de l'entrée. Voici le tableau de la fonction de génération. Nous rappelons qu'une instruction est composé de la manière suivante : $C_2\ C_1\ C_0 + RX_2\ RX_1\ RX_0 + RY_2\ RY_1\ RY_0$, avec C_i pour le CODEOP, RX_i pour le Registre d'écriture et RY_i pour le Registre de lecture. Également voici un rappelle des sorties de l'UC :
— `IR_s`, `A_s`, `G_s` et `Ri_s` : Pour activer l'écriture dans les registres correspondants

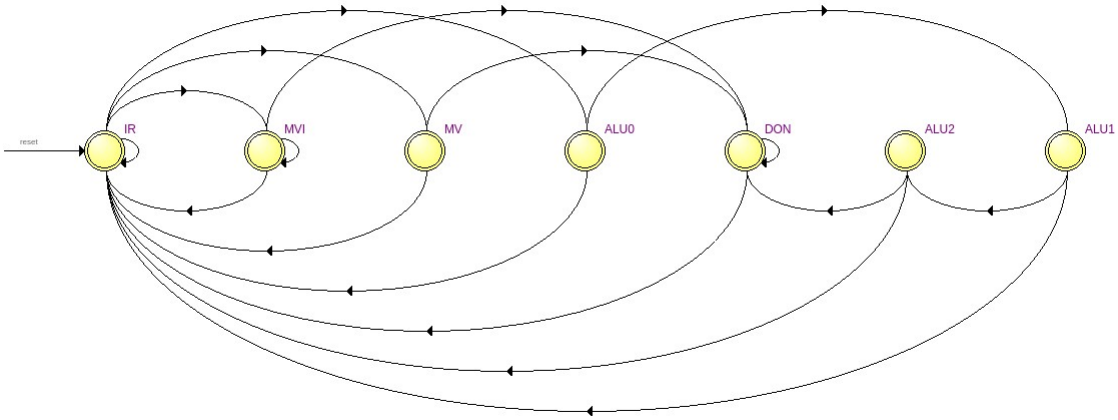

FIGURE 2 – Diagramme des états de l'UC

État Actuel	État Futur	Condition
any	IR	reset
IR	MV	$C_2.C_1.!C_0.(!\text{reset})$
IR	MVI	$C_2.C_1.C_0.(!\text{reset})$
IR	ALU0	$!C_1.(!\text{reset}) + !C_2.C_1.(!\text{reset})$
MV	DONE	$(!\text{reset})$
MVI	DONE	$(!\text{reset})$
MVI	DONE	$(!\text{reset})$
ALU0	ALU1	$(!\text{reset})$
ALU1	ALU2	$(!\text{reset})$
ALU2	DONE	$(!\text{reset})$
DONE	IR	run

TABLE 1 – Fonction de Transition

— `Bus_sel` : Pour sélectionner la sortie du Multiplexeur
— `ALU_sel` : Pour sélectionner l'opération sur l'ALU
— Done : Activer lorsque l'UC a terminé d'exécuter l'instruction

Indication : Dans le tableau qui suit nous considérons `Ri_s` *en une seule colonne tel que* `Ri` $= R_7\ R_6\ R_5\ R_4\ R_3\ R_2\ R_1\ R_0$

Voici les table de correspondance pour *decode()* et *decodeBus()* :

La fonction de génération est implémenté par description comportementale au sein de l'Unité de Contrôle.

Etat	IR_s	A_s	G_s	Ri_s	Bus_sel	ALU_sel	Done
IR	1	0	0	00000000	/	/	0
MV	0	0	0	decode(RX)	decodeBus(RY)	/	0
MVI	0	0	0	decode(RX)	1111 (Din dans le multiplexeur)	/	0
ALU0	0	1	0	00000000	decodeBus(RX)	/	0
ALU1	0	0	1	00000000	decodeBus(RY)	CODEOP	0
ALU2	0	0	1	decode(RX)	1000 (G dans le Multiplexeur)	CODEOP	0
DONE	0	0	0	00000000	/	/	1

TABLE 2 – Fonction de Génération

RX ou RY	Ri_s
000	00000001
001	00000010
010	00000100
011	00001000
100	00010000
101	00100000
110	01000000
111	10000000

TABLE 3 – Table de correspondance decode()

RX ou RY	Bus_sel
000	0000
001	0001
010	0010
011	0011
100	0100
101	0101
110	0110
111	0111

TABLE 4 – Table de correspondance decodeBus()