400099 - M1 - 1er Semestre
1ere Session

Architecture évoluée

AO documents autorisés **Maîtrise d'informatique** **janvier 2014**

Exercice 1 (6 points)

On considère un pipeline multifonctions. Il peut réaliser plusieurs types d'opération en même temps par exemple le pipeline de TI-ASC permettait de réaliser une ADD flottant ou un ADD fixe ou une MUL fixe ou un Dot product flottant.
La table de réservation d'un tel pipeline est obtenue par superposition des tables de chaque fonction.

1) Construisez la table de réservation du pipeline qui réalise soit A soit B avec :

Fonction A :

X			X	
	X			
		X		X

Fonction B :

	X			X
			X	
X		X		

Donnez les vecteurs de collision de A et B.

2) Par extension du modèle mono fonction, on peut construire pour ce pipeline 4 vecteurs de collision de 4 bits (Bit à 1 si déclenchement impossible, bit à 0 si possible, écriture de la gauche vers la droite t+1, t+2, t+3 ...):
V_{AA} représente les collisions de A quand A est déjà déclenché
V_{AB} représente les collisions de A quand B est déjà déclenché
V_{BA} représente les collisions de B quand A est déjà déclenché
V_{BB} représente les collisions de B quand B est déjà déclenché
Donnez les valeurs de ces 4 vecteurs.

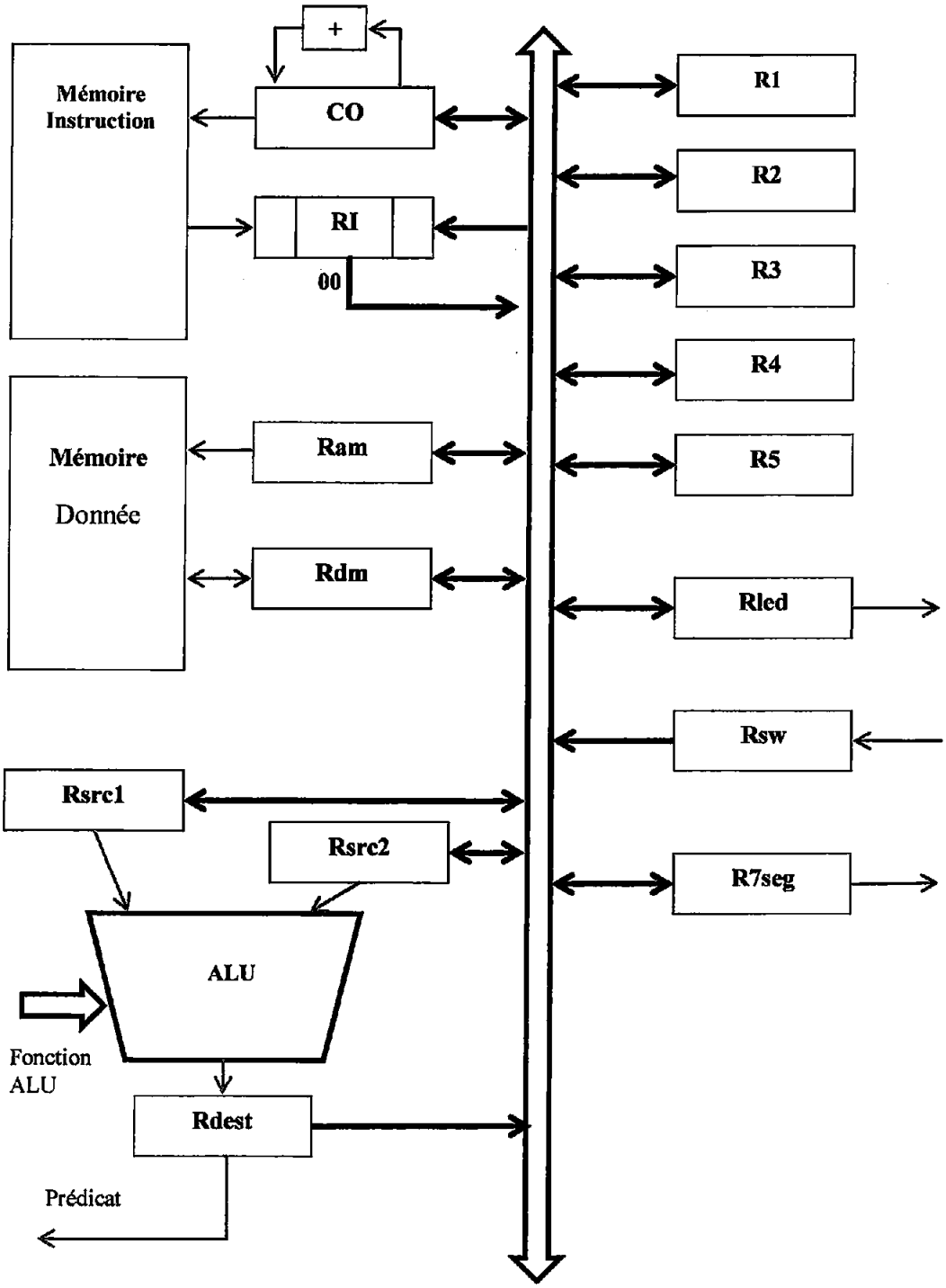
3) A partir de ces vecteurs de collision, on construit 2 matrices de collision qui regroupent les vecteurs de collision après un A $M_A\{V_{AA} / V_{BA}\}$et ceux après un B M_B $\{V_{AB} / V_{BB}\}$ (Les V_{xy} sont les lignes des matrices). Ces 2 matrices sont les valeurs initiales du graphe d'état suivant que l'on déclenche en premier un A ou un B. A partir de ces deux états initiaux construisez le graphe complet des états possibles du pipeline en fonction des déclenchements possibles de calcul de A ou de B. Le fonctionnement est semblable à celui d'un pipeline mono fonction.

4) Trouvez le meilleur cycle du graphe d'état.

5) On dispose de deux registres à décalage, proposez un algorithme glouton qui autorise ou non le déclenchement d'une nouvelle instruction A ou B en fonction des requêtes présentées (priorité sur la A)

Exercice 2 (14 points)

Le processeur HoMade peut être vu comme un intégrateur d'IPs. Les IPs ainsi intégrés sont connectés à la pile d'exécution. Vous allez construire un intégrateur d'IPs à partir d'un processeur RISC à architecture Load/Store. Ce processeur S3 est celui utilisé par les étudiants de L2. Voici la description du processeur S3



Le processeur S3 est un processeur 16 bits autant pour les données que pour les adresses. Le jeu d'instructions est réduit et se situe au niveau du transfert de registre. Les instructions sont décodées puis exécutées. S3 anticipe le chargement de l'instruction suivante pendant l'exécution du chargement en cours : on parle de **pipeline à deux étages**. Les flèches épaisses correspondent aux microcommandes qui permettent de déclencher des transferts de données. La flèche particulière en sortie de *RI* transfert les 8 bits de *RI(11 :4)* concaténés à gauche avec x00. Cela construit un mot de 16 bits à partir d'une partie de RI. La flèche *prédicat* est utilisée par la FSM, elle correspond à '0' si tous les bits de Rdest sont égaux à '0' et '1' sinon.

L'ALU peut exécuter 16 fonctions différentes, certaines vous seront données par la suite. Le registre Rled est toujours connecté aux leds de la carte Nexys3 en sortie, R7seg à l'afficheur 4 digits 7 segments et Rsw aux switches en entrée les 8 bits de poids forts sont alors forcés à x00. Voici la liste des instructions.

MOV Rs Rd : transfert de registres Rs : tous les registres qui ont une flèche vers le bus
Rd : tous les registres qui ont une flèche depuis le bus

ADD : Rdest = Rscr1 + Rscr2
SUB : Rdest = Rscr1 - Rscr2
ADDC : Rdest = Rscr1 + Rscr2 + 1
SUBC : Rdest = Rscr1 - Rscr2 – 1
INV : Rdest = NOT Rsrc1
AND : Rdest = Rsrc1 AND Rsrc2
OR : Rdest = Rsrc1 OR Rsrc2
INC : Rdest = Rsrc1 + 1
COMP2 : Rdest = NOT Rsrc1 + 1
CONCAT : Rdest = Rsrc2(7 :0) & Rsrc1(7 :0)
MVI xdd Rd : transfert de x00dd vers Rd
MZ Rs Rd : idem que MOV mais seulement si prédicat = '0'
MNZ Rs Rd : idem que MOV mais seulement si prédicat <> '0'
MVIZ xdd Rd: idem que MVI mais seulement si prédicat = '0'
MVINZ xdd Rd: idem que MVI mais seulement si prédicat <> '0'
PAUSE : bloque l'exécution jusque la pression du bouton Btn0 de la carte.
NOP : Opération nulle.

Les transferts de registres vers le CO ont pour effet de changer le déroulement séquentiel du programme vers une nouvelle adresse : celle qui sera transférée dans CO. C'est la seule façon de faire des branchements. Comme le processeur est un pipeline de 2 étages et qu'il n'y a pas de vidage de pipeline, quand le changement du CO est effectif, l'instruction suivante a déjà été chargée, elle sera quand même exécutée avant le branchement

1) Quelle est la sémantique du branchement lorsque le registre destination d'un Move est le CO ? Que produit le programme suivant :
   0 NOP
   1 MVI 01 R7seg
   2 PAUSE
   3 MVI 02 CO
   4 MVI 02 R7Seg
   5 MVI 03 R7seg

2) Expliquez le fonctionnement et l'usage qu'on peut faire des instructions MOV, MVI, MZ, MNZ, MVIZ, MVINZ avec le registre CO en destination.

3) Donnez le code assembleur de IF R1 = R2 then R3 = 04 else R3 = R3 + 1 ENDIF

4) Donnez le programme qui affiche le Ième nombre de Fibonacci, I étant saisi sur les switches.

La suite du sujet concerne le développement d'IPs short. On ajoute une instruction IP vvv ou vvv est le numéro de l'IP que l'on désire activé, il est codé sur 12 bits ici. Tous les Ips qui

seront placés dans le processeur S3 le seront entre les registres Rsrc1, Rsrc2 et Rdest. (à la place de l'ALU)
Pour adapter l'ALU qui permet de réaliser 16 fonctions, il faut prendre en compte les 8 bits de poids fort de vvv pour sélectionner l'ALU puis les 4 bits de poids faible pour déclencher une opération parmi les 16 possibles. Vous disposez de comparateur x bits et de circuit BUFT (IN Select, bus_in ; OUT bus_out) fonctionnant comme un tri state et qui va permettre de produire ou non le résultat de l'ALU sur un nouveau bus local connecté en sortie sur Rdest et en entrée sur les sorties des IPs.

5) Proposez un schéma qui implémente notre ALU comme un IP. Les instructions ADD deviennent IP x001, SUB : IP x002 etc... 00v étant les numéros associés à l'IP ALU

6) Venez accoler à votre processeur un second IP qui génère un nombre aléatoire sur 16 bits. On le numérote x010. Donnez le code complet VHDL de cet IP. Donnez le schéma du processeur S3 (la partie entre les Rsrc et Rdest avec les commandes nécessaires).

7) Ecrire un programme qui calcule PI par méthode Monté-Carlo.