Université Bordeaux 1 — U.F.R. Mathématiques et Informatique
Licence Informatique 2ème année — Année 2012 – 2013

ARCHITECTURE DES ORDINATEURS

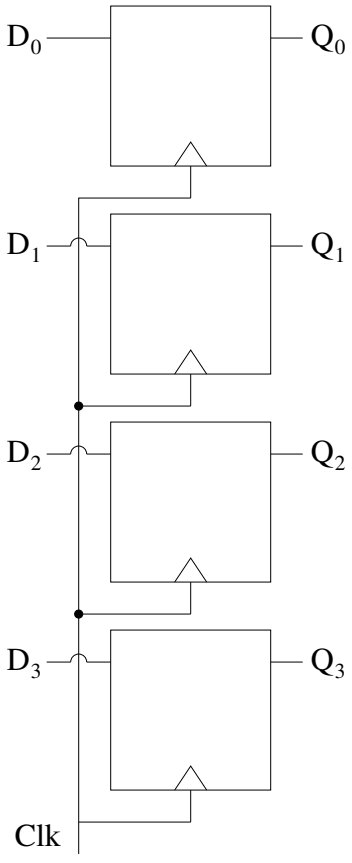
Devoir Surveillé 2

1 heure 20 minutes
sans documents

N.B. : - Travaillez calmement. Ne bâclez pas vos réponses : il vaut mieux traiter correctement moins de questions que tout faire de travers.
- Les réponses aux questions doivent être argumentées et aussi concises que possible.
- Le barème est donné à titre indicatif.

Question 1 (4 points)

Le circuit ci-contre est un registre de 4 bits conçu en juxtaposant simplement des bascules « D ». On souhaite étendre ce circuit pour pouvoir l'utiliser comme verrou (« *latch* ») dans la version pipe-linée du processeur Y86, et plus précisément pour stocker la valeur `D_icode`, qui est le code de l'instruction en cours à l'étage *Decode*.



(1.1) (2 points)

Dans sa version actuelle, le registre prend la valeur $\texttt{D}_3\texttt{D}_2\texttt{D}_1\texttt{D}_0$ au début de chaque cycle. Dans certains cas, il est toutefois nécessaire de forcer une instruction à demeurer à l'étage *Decode* au cycle suivant.

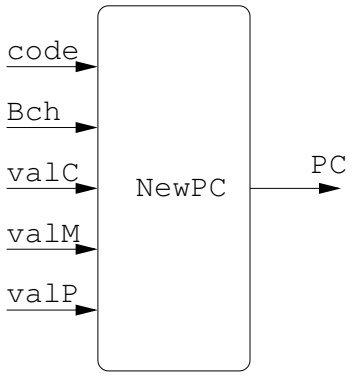
On veut donc ajouter un fil de commande `stall` permettant, lorsqu'il vaut 1, de forcer le registre à conserver sa valeur actuelle au prochain cycle. Ajoutez au circuit ci-contre les blocs logiques et/ou portes logiques et connexions nécessaires à ce fonctionnement. *N.B. : Mettez en œuvre une solution qui ne crée pas de dérive de l'horloge !*

(1.2) (2 points)

Parfois, il est également nécessaire d'injecter une « bulle » à l'étage *Decode* du pipe-line. Sachant que le code de l'instruction `NOP` est 0 (tous les bits à zéro), ajoutez au circuit les portes et/ou blocs logiques, ainsi qu'un fil de commande `bubble`, pour déclencher, lorsque ce fil vaut 1, l'injection d'une bulle au cycle suivant.

Question 2 (4 points)

La figure ci-contre montre les entrées du circuit combinatoire `NewPC` chargé de calculer l'adresse de la prochaine instruction que le processeur Y86 séquentiel devra charger (l'entrée `Bch` signifie « *Branch Flag* »). Le code HCL tronqué décrivant ce circuit est donné sous le schéma.



(2.1) (1 point)

Expliquez le code HCL fourni.

(2.2) (1 point)

Quelles sont les instructions nécessitant un traitement spécifique dans `NewPC` ?

(2.3) (2 points)

Complétez ce code pour traiter les quelques cas non gérés actuellement. Pour chaque ligne ajoutée, expliquez précisément le choix de la valeur résultat.

```
PC = [
  code == Jxx && Bch : valC ;
  ...
  1 : valP ;
]
```

Question 3 (4 points)

Donnez les critères nécessaires pour qu'il soit économiquement intéressant de « pipe-liner » une fonction logique. Expliquez rapidement chacun d'eux. *(15 lignes maximum)*

Question 4 (8 points)

On désire augmenter la fiabilité de transmission d'un système de transmission de données. Pour cela, on souhaite être capable de détecter et de corriger *une* erreur par mot de 4 bits envoyé. Pour cela, on ajoutera, à chaque mot de 4 bits, 3 bits supplémentaires, calculés selon le Code de Hamming.

Pour transmettre les 4 éléments binaires (m_1, m_2, m_3, m_4) correspondant à un chiffre hexadécimal, on ajoute trois éléments binaires (k_1, k_2, k_3) pour assurer des contrôles de parité. On émet alors un mot binaire de 7 bits $(k_1, k_2, m_1, k_3, m_2, m_3, m_4)$, satisfaisant les tests de parité suivants :

$$\begin{array}{rcccl} t_1 & = & \text{parité}(k_1, m_1, m_2, m_4) & = & 0\ , \\ t_2 & = & \text{parité}(k_2, m_1, m_3, m_4) & = & 0\ , \\ t_3 & = & \text{parité}(k_3, m_2, m_3, m_4) & = & 0\ . \end{array}$$

Par exemple, $t_1 = 0$ signifie que le mot binaire (k_1, m_1, m_2, m_4) contient un nombre *pair* de bits à 1 (on peut également le voir comme le résultat, modulo 2, de la somme des bits du mot).

À la réception, on effectue les mêmes tests de parité sur le mot reçu $(K_1, K_2, M_1, K_3, M_2, M_3, M_4)$ *(les majuscules indiquent que ces valeurs reçues peuvent être différentes des valeurs émises, en cas d'erreur de transmission)* :

$$\begin{array}{rcccl} T_1 & = & \text{parité}(K_1, M_1, M_2, M_4) & = & 0\ , \\ T_2 & = & \text{parité}(K_2, M_1, M_3, M_4) & = & 0\ , \\ T_3 & = & \text{parité}(K_3, M_2, M_3, M_4) & = & 0\ . \end{array}$$

Le résultat des tests permet de former le mot binaire (T_1, T_2, T_3). Si ce mot est à zéro, c'est qu'il n'y a pas d'erreur de transmission. Sinon, le mot binaire (T_1, T_2, T_3) code la position du bit erroné, qu'il est alors facile de corriger.

Par exemple :

Mot à transmettre	Mot émis		Mot reçu	Résultat des tests
1101	101**0101**	$\Longrightarrow$	1000101	$T_3T_2T_1 = 011$

Le bit erroné est donc le $3^{ème}$ bit du mot reçu (en partant de la gauche).

(4.1) (2 points)
À l'aide de tables de Karnaugh, simplifiez les fonctions logiques correspondant aux fonctions k_1, k_2 et k_3.

(4.2) (2 points)
À l'aide de tables de Karnaugh, simplifiez les fonctions logiques correspondant aux fonctions T_1, T_2 et T_3.

(4.3) (2 point)
À l'aide de blocs logiques et/ou de portes logiques, dessinez le schéma du dispositif du récepteur permettant le calcul de T_1, T_2 et T_3.

(4.4) (2 points)
À l'aide de blocs logiques et/ou de portes logiques, proposez un dispositif simple réalisant la correction du mot reçu, pour obtenir le mot (k_1, m_1, m_2, m_4) après cette correction éventuelle.