ÉCOLE POLYTECHNIQUE MONTRÉAL

Le génie sans frontières

Questionnaire examen final

INF3500

Sigle du cours

Identification de l'étudiant(e)		
Nom :	**Prénom :**	
Signature :	**Matricule :**	**Groupe :**

Sigle et titre du cours	*Groupe*	*Trimestre*
INF3500 Conception et réalisation de systèmes numériques	**Tous**	**20081**
Professeur	***Local***	***Téléphone***
François-Raymond Boyer	**M-4404**	**5062/5193**

Jour	*Date*	*Durée*	*Heures*
Samedi	**26 avril 2008**	**2h30**	**9h30**

Documentation	*Calculatrice*	
☒ Toute	☐ Aucune	**Les cellulaires, agendas électroniques ou téléavertisseurs sont interdits.**
☐ Aucune	☒ Programmable	
☐ Voir directives particulières	☐ Non programmable	

Directives particulières

Ordinateurs interdits (une calculatrice exécutant du VHDL est un ordinateur), répondre à toutes les questions, la valeur de chaque question est indiquée.

Total possible de 42 sur 40.

Bonne chance à tous!

Important

Cet examen contient **4** questions sur un total de **3** pages **(excluant cette page)**

La pondération de cet examen est de **40** **%**

Vous devez répondre sur : ☐ le questionnaire ☐ le cahier ☐ les deux

Vous devez remettre le questionnaire : ☐ oui ☒ non

Tous les codes VHDL font : library ieee; use ieee.std_logic_1164.all; use ieee.numeric_std.all;

Question 1. [17 points]

L'opération d'exponentiation entière est très utile en cryptographie. Nous voulons trouver $c = a^b \bmod n$, où a, b et n sont des entiers positifs. Ces nombres sont de grande taille (>512 bits) pour avoir un système de cryptage assez sécuritaire. Voici l'algorithme d'exponentiation utilisé :

```
C = 1;
répéter pour chaque bit de B {
        C = (C*C) mod n;
        if (bit_de_poids_fort_de(B) == 1)
                C = (C*A) mod n;
        B = B << 1; // décaler B d'un bit vers la gauche (laissant tomber le bit qu'on vient de tester)
}
```

Idée de l'algorithme (pas nécessaire de le comprendre pour répondre à la question) :
Nous utilisons simplement le fait que $a^b = (a \cdot a)^{b/2}$ (l'exposant reste entier s'il était pair), et $a^b = a^{b-1} \cdot a$ (permet de changer un exposant impair en exposant pair). Appliquer directement ces règles ferait un algorithme récursif, mais en commençant par le bit de poids fort il devient itératif.

Comme première étape de la conception d'un circuit faisant cette opération, vous considérez $n = 2^W$ où W est le nombre de bits des entiers A, B et C. Ceci vous permettra de faire des tests sans que l'équipe qui s'occupe de la multiplication modulaire ait terminé leur circuit. L'opération « mod n » revient donc à prendre les bits de poids faible du résultat (RESIZE(valeur, nbits) en VHDL permet de changer la taille en bits d'un unsigned).

Le processeur dédié à l'exponentiation aura en entrée les signaux entree, chargeA, chargeB et go (en plus des signaux habituels reset et clk), et en sortie les signaux resultat et fini. Les signaux chargeA et chargeB chargent la valeur entree dans A et B, respectivement, lorsque le circuit est en attente. Le circuit commence à calculer lorsque go=1, et lorsqu'il a terminé et que la valeur du resultat (la valeur de C) est valide, il indique fini=1.

Pour des considérations de taille, un seul multiplieur doit être utilisé dans le circuit (si l'opérateur de multiplication « * » est utilisé plusieurs fois dans le code VHDL, le synthétiseur les combinera uniquement si les deux mêmes signaux sont multipliés ensemble aux différents endroits).

a. [5 pts] Donnez un diagramme montrant le chemin des données du processeur. Indiquez clairement toutes les ressources requises, leur nombre et leur largeur en bits. Indiquez clairement les signaux de contrôle.

b. [5 pts] Donnez la machine à états du processeur. Identifiez bien chaque état, les actions à prendre dans chaque état, et les conditions pour les transitions entre les états.

c. [7 pts] Donnez une description de votre processeur en VHDL, en vous basant sur le début de code suivant.

```
entity exponentiateur is
    generic ( W : integer ); -- nombre de bits des opérandes
    port(  reset, CLK, go :      in std_logic;
           entree :              in unsigned(W - 1 downto 0);
           chargeA, chargeB :    in std_logic;
           resultat :            out unsigned(W - 1 downto 0);
           fini :                out std_logic );
end exponentiateur;

architecture arch of exponentiateur is
signal A, B, C : unsigned(W - 1 downto 0);
...
```

Question 2. [8 points]

Considérez le problème de la vérification d'un circuit combinatoire qui doit rencontrer les spécifications de la question 1.
Complétez le squelette de banc d'essai suivant en VHDL en incluant une génération algorithmique de vecteurs de tests ainsi qu'une évaluation automatisée des réponses du circuit. Indiquez clairement dans quelle partie du code existant votre code doit être placé. Rappel : les opérateurs ** et mod sont définis sur les natural, et wait until et wait for peuvent être utilisés en série sans contraintes si le process n'a pas de liste de sensibilité.

```
use ieee.math_real.all;

entity exponentiateurTB is
        generic (W : INTEGER := 8); -- un petit nombre, pour faire les tests initiaux
end exponentiateurTB;

architecture TB_ARCHITECTURE of exponentiateurTB is

    component exponentiateur
        generic (W : INTEGER);
        port ( reset, CLK, go :     in std_logic;
               entree :             in unsigned(W - 1 downto 0);
               chargeA, chargeB :   in std_logic;
               resultat :           out unsigned(W - 1 downto 0);
               fini :               out std_logic );
    end component;

    signal reset, CLK, go :     std_logic := '0';
    signal entree :             UNSIGNED(W-1 downto 0);
    signal chargeA, chargeB :   std_logic := '0';
    signal resultat :           UNSIGNED(W-1 downto 0);
    signal fini :               std_logic;
    constant period :           time := 10 ns;
    ...
begin
    UUT : exponentiateur generic map (W => W)
        port map (reset, CLK, go, entree, chargeA, chargeB, resultat, fini);
    ...
    process
        variable A, B, AexposantB :  natural range 0 to 2 ** W - 1 := 0;
        ...
    begin
        ...
    end process;

end TB_ARCHITECTURE;
...
```

Question 3. [9 points]

Considérez le processeur à usage général décrit dans les notes de cours, sections 8.6 à 8.8 inclusivement. Nous voulons supporter une plus grande mémoire d'instructions ainsi que la possibilité de faire des sous-routines. Pour ce faire, les branchements seront relatifs (les 8 bits d'adresse spécifiés dans l'instruction seront considérés comme une valeur signée disant la position où aller relativement à l'instruction présentement pointée), et deux nouveaux types de branchements seront ajoutés : le JAL (jump and link) qui copie le pointeur d'instruction dans R15 avant d'effectuer le saut, et le JR (jump register) qui branche à l'adresse contenue dans un registre.

a. [3 pts] Expliquez brièvement les modifications à faire au processeur actuel. Indiquez ce qu'il faudra modifier au chemin des données (Fig. 8-12, p.8-16), et à l'unité de contrôle (Fig. 8-15, p.8-24). Comment les deux nouvelles instructions pourrait être codées (Tab. 8-5 et 8-6, p.8-25) pour que leur implémentation dans le processeur actuel soit simple.

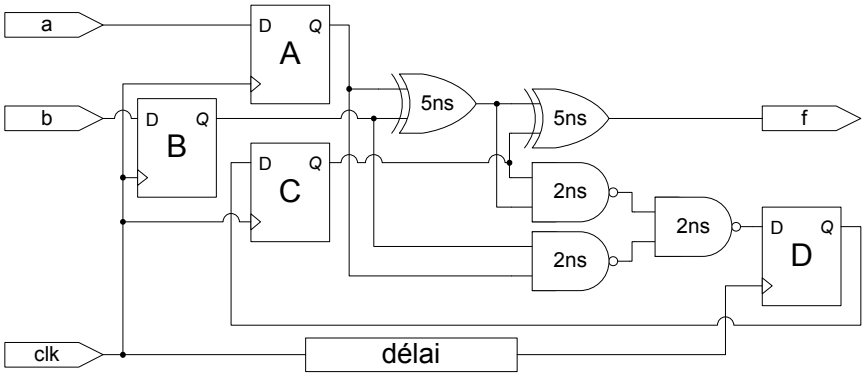
Pour les sous-questions suivantes, indiquez les lignes VHDL à ajouter/modifier, dans le contrôle (Exemples 8-13 et 8-14, p.8-28 et 8-29) et si nécessaire dans le chemin de données (Exemples 8-8 à 8-11, p.8-19 à 8-22).

b. [2 pts] Modifiez les branchements pour qu'ils soient relatifs.

c. [4 pts] Ajoutez l'instruction JAL.

Question 4. [8 points]

Considérez le circuit numérique suivant. Supposez que les bascules ont un temps de préparation de 1 ns, un temps de maintien de 0,5 ns et un délai de propagation de 2 ns. Les délais des autres composantes sont indiqués. Négligez les délais des interconnexions. Négligez toute contrainte de synchronisation avec les entrées a et b et la sortie f.



a. [4 pts] Identifiez clairement le(s) chemin(s) à surveiller pour savoir si les bascules seront stables. Déterminez la fréquence maximale d'horloge dans le cas où le délai sur l'horloge est nul (aucun déphasage).

b. [4 pts] En gardant le délai sur l'horloge nul, et sans avoir à changer le comportement sur les entrées (même valeurs envoyées aux mêmes cycles), expliquez comment il est possible d'accélérer ce circuit en utilisant des bascules placées différemment et dites quelle nouvelle période d'horloge ceci donnerait.