INF3500 : Conception et réalisation de systèmes numériques

Examen final

Décembre 2010

Durée: 2h30.

Pondération: 40%.

Documentation: Une feuille recto verso 8.5”×11” ou A4 permise.

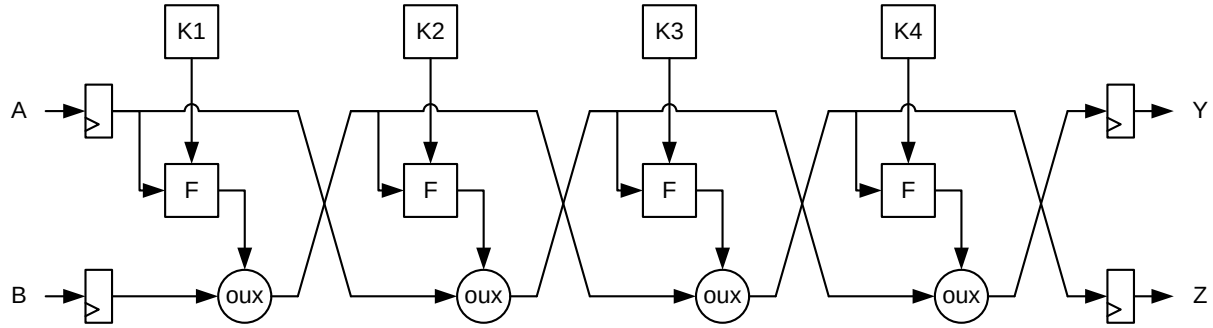
Calculatrice: Programmable permise.

Directives particulières:

- Ordinateurs interdits.
- Répondre à toutes les questions, la valeur de chaque question est indiquée.
- Répondre dans le cahier fourni.
- Ne pas remettre le questionnaire, sauf la dernière page qui doit être annotée et remise.
- Ne posez pas de questions. En cas de doute sur le sens d’une question, énoncez clairement toute supposition que vous faites.

Question 1. (10 points)

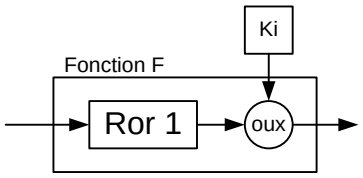
Le réseau de Feistel est utilisé dans les algorithmes de chiffrement par bloc. Plusieurs algorithmes utilisent le réseau de Feistel, dont DES, Blowfish et RC5. Le diagramme suivant illustre un réseau de Feistel simple à quatre étages. Les algorithmes cryptographiques basés sur un réseau de Feistel diffèrent principalement dans le nombre d'étages et dans la nature de la fonction F.



Le message à chiffrer est décomposé en un flux de nombres appliqués aux entrées A et B du réseau. Les quatre clés secrètes K1, K2, K3 et K4 restent constantes pour le chiffrement du message. Les sorties Y et Z sont un flux de nombres représentant le message chiffré.

On constate qu'à chaque étage le signal du haut est combiné à la clé par la fonction F. On effectue ensuite un ou-exclusif bit à bit avec le signal du bas. À la fin de l'étage, les signaux du haut et du bas sont interchangés pour le prochain étage.

Supposez ici que tous fils du diagramme représentent un signal de 16 bits. Supposez que la fonction F consiste à appliquer une rotation de l'entrée de 1 bit vers la droite puis à effectuer l'opération ou-exclusif bit à bit avec la clé K_i exprimée sur 16 bits.



a. (6 points) Considérez la déclaration d'entité suivante pour le diagramme du réseau de Feistel. Donnez une architecture en VHDL synthétisable correspondant à cette entité et au diagramme.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity feistel4 is
    generic (
        W : positive := 16
    );
    port (
        clk, reset : in std_logic;
        A, B : in unsigned(W - 1 downto 0);
        Y, Z : out unsigned(W - 1 downto 0)
    );
end feistel4;
```

b. (4 points) Donnez le code VHDL pour un banc d'essai qui doit vérifier le fonctionnement de votre architecture pour le réseau de Feistel.

Le banc d'essai doit générer des vecteurs de test aléatoire, calculer la réponse attendue et vérifier que les valeurs des sorties sont conformes. Si un vecteur de test engendre une erreur à la sortie, le banc d'essai doit s'arrêter et indiquer un message d'erreur à la console.

Solution

a.

```
architecture arch1 of feistel4 is

type unsigned2D is array(natural range <>) of unsigned(W - 1 downto 0);
constant K : unsigned2D(1 to 4) := (x"1234", x"5678", x"9ABC", x"DEF0");

signal A0, A1, A2, A3, A4, B0, B1, B2, B3, B4 : unsigned(W - 1 downto 0);

begin

   -- registres à l'entrée pour A et B et à la sortie pour Y et Z
   process (clk, reset)
   begin
      if reset = '1' then
         Y <= (others => '0');
         Z <= (others => '0');
         A0 <= (others => '0');
         B0 <= (others => '0');
      elsif rising_edge(clk) then
         A0 <= A;
         B0 <= B;
         Y <= A4;
         Z <= B4;
      end if;
   end process;

   -- description du réseau comme tel
   -- on pourrait aussi paramétrer A et B, puis utiliser une description
   -- avec une boucle (dans un process) ou bien avec énoncé generate
   B1 <= A0;
   A1 <= (A0 ror 1) xor K(1) xor B0;

   B2 <= A1;
   A2 <= (A1 ror 1) xor K(2) xor B1;

   B3 <= A2;
   A3 <= (A2 ror 1) xor K(3) xor B2;

   B4 <= A3;
   A4 <= (A3 ror 1) xor K(4) xor B3;

end arch1;
```

b.

```
library IEEE;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity feistel4_TB is
   generic (W : positive := 16);
end feistel4_TB;

architecture arch1 of feistel4_TB is

type unsigned2D is array(natural range <>) of unsigned(W - 1 downto 0);
constant K : unsigned2D(1 to 4) := (x"1234", x"5678", x"9ABC", x"DEF0");

signal clk, reset : std_logic := '0';
```

```
signal A, B, Y, Z : unsigned(W - 1 downto 0);
signal Y_att, Z_att : unsigned2D(0 to 2);

begin

clk <= not clk after 10 ns;
reset <= '1' after 0 ns, '0' after 22 ns;
UUT : entity feistel4(arch1) generic map(W)
   port map (clk, reset, A, B, Y, Z);

process(clk, reset)
variable A_stim, B_stim : unsigned(W - 1 downto 0);
variable seed1 : positive := 1;
variable seed2 : positive := 2;
variable aleatoire : real;
variable A_int, B_int : unsigned2D(1 to 5);
variable compteurReset : natural;  -- début de la vérif. après reset
begin

   if reset = '1' then
      compteurReset := 0;
   elsif rising_edge(clk) then

      -- génération des stimuli
      uniform(seed1, seed2, aleatoire);
      aleatoire := floor(aleatoire * real(2 ** W));
      A_stim := to_unsigned(integer(aleatoire), W);

      uniform(seed1, seed2, aleatoire);
      aleatoire := floor(aleatoire * real(2 ** W));
      B_stim := to_unsigned(integer(aleatoire), W);

      -- application des stimuli à l'entrée
      A <= A_stim; B <= B_stim;

      -- calcul des réponses attendues
      A_int(1) := A_stim; B_int(1) := B_stim;
      for q in 2 to 5 loop
         A_int(q) := (A_int(q - 1) ror 1) xor K(q - 1) xor B_int(q - 1);
         B_int(q) := A_int(q - 1);
      end loop;

      Y_att(0) <= A_int(5); Z_att(0) <= B_int(5);

      -- retard des réponses attendues de 2 coups d'horloge dans le temps
      -- voir le diagramme: 2 bascules de délai
      Y_att(1) <= Y_att(0); Z_att(1) <= Z_att(0);
      Y_att(2) <= Y_att(1); Z_att(2) <= Z_att(1);

      -- vérification automatique des sorties
      if (compteurReset >= 3) then
         assert (Y = Y_att(2) and Z = Z_att(2))
         report "erreur" severity error;
      else
         compteurReset := compteurReset + 1;
      end if;

   end if;

end process;

end arch1;
```

Question 2. (10 points)

Considérez le diagramme du réseau de Feistel de la Question 1. Le réseau est implémenté sur un FPGA. Après implémentation, on a caractérisé les différentes parties du réseau comme suit. Les bascules ont un délai de 1 ns, un temps de préparation t_{su} de 0.25 ns, et un temps de maintien t_h de 0.1 ns. Les blocs de fonction F ont un délai de 5 ns, et les ou-exclusif ont un délai de 3 ns. Les fils d'interconnexion ont des délais de 0.1 ns chacun. Les blocs des clés n'ont pas de délais, ce sont des constantes.

a. (3 points) Identifiez le chemin critique du circuit et donnez la fréquence maximale d'opération.

b. (1 point) Donnez la latence du circuit en secondes et le débit du circuit en nombres de résultats par seconde.

c. (2 points) On désire atteindre un débit minimal de 100×10^6 résultats par seconde. Indiquez comment pipeliner le circuit pour atteindre ce niveau de performance. Annotez le diagramme de la dernière page du questionnaire et remettez-le dans votre cahier d'examen.

d. (1 point) Expliquez comment le code VHDL décrivant le circuit doit être modifié pour inclure les registres de pipeline que vous avez identifiés à la partie c.. Ne donnez pas le code au complet. Donnez un seul exemple pour un seul signal.

e. (3 points) Évaluez la latence du circuit pipeliné en secondes et le débit du circuit pipeliné en résultats par seconde. Justifiez complètement votre réponse.

Solution

a. Le chemin critique va de la bascule pour le signal A à la bascule pour le signal Y, en passant par les quatre boîtes 'F' et les quatre ou-exclusifs. Le délai total est donné par :

délai bascule + 4 × délai 'F' + 4 × délai 'oux' + 9 × délai fil + t_{su} = 1 + 4 × 5 + 4 × 3 + 9 × 0.1 + 0.25 = 34.15 ns. La fréquence maximale est donc 29.3 MHz

b. La latence est de 34.15 ns. Le débit est de 29.3×10^6 résultats/seconde.

c. Une solution élégante consiste à ajouter un registre après chaque étage du réseau. Chaque étage a deux entrées et deux sorties. On peut considérer les registres pour A et B comme les registres d'entrée du premier étage, et ajouter des registres aux deux entrées de chaque étage. Le nombre total de registres nécessaires est donc de 6 (de 16 bits chacun).

d. Il faut placer les définitions des signaux d'entrée des étages à l'intérieur de la définition du processus, à l'intérieur de la condition sur la transition sur le signal d'horloge. Par exemple,

```
B1 <= A0;
A1 <= (A0 ror 1) xor K(1) xor B0;
```

Pourraient être placés sous la ligne « `elsif rising edge(clk) then` »

e. Le chemin critique du circuit pipeliné devient

délai bascule + 1 × délai 'F' + 1 × délai 'oux' + 3 × délai fil + t_{su} = 1 + 5 + 3 + 3 × 0.1 + 0.25 = 9.55 ns. La fréquence maximale est donc 104.7 MHz.

La latence du circuit devient 4 coups d'horloge, donc 38.2 ns (elle a augmenté).

Le débit du circuit devient 104.7×10^6 résultats/seconde (en régime permanent).

Question 3. (6 points)

Considérez le code VHDL suivant pour un module combinatoire et son banc de test associé.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity module4 is
   port (
      A, B, C : in std_logic;
      F, G : out std_logic
   );
end module4;

architecture arch of module4 is

signal S1, S2: std_logic;

begin

   S1 <= A and B;
   S2 <= not(C or B);

   process(S1)
   begin
      F <= S1 or not(S2);
   end process;

   process(S2)
   begin
      G <= A xor B;
   end process;

end arch;
```

```
library IEEE;
use IEEE.std_logic_1164.all;

entity module4TB is
end module4TB;

architecture arch of module4TB is

signal A, B, C, F, G : std_logic;

begin

   UUT : entity module4(arch) port map
(A, B, C, F, G);
   A <= '1' after 0 ns;
   B <= '1' after 0 ns, '0' after 10 ns;
   C <= '0' after 0 ns;

end arch;
```

a. (3 points) Donnez la liste des événements telle qu'elle pourrait être dressée par un simulateur qui exécuterait le banc d'essai.

b. (3 points) En vous basant sur votre liste des événements, donnez la valeur de tous les ports et signaux internes du module combinatoire en fonction du temps, en tenant compte des délais deltas.

Solution :

a. Liste des événements

À t = 0 + 0 Δ, initialisation de la simulation (tout à ‘U’)

À t = 0 + 1 Δ, assignation de 110 à (A, B, C)

À t = 0 + 2 Δ, évaluation des signaux S1 et S2 (provoquées par les changements sur A, B, C)

À t = 0 + 3 Δ, évaluation de F et G (provoquées par les changements sur S1 et S2)

Aucun nouvel événement n’est ajouté à la liste d’événements.

À t = 10 ns + 0 Δ, assignation de 0 à B

À t = 10 ns + 1 Δ, évaluation de S1 et S2 (provoquées par le changement sur B)

À t = 10 ns + 2 Δ, évaluation de F et G (provoquées par le changement sur S1 et S2)

Aucun nouvel événement n’est ajouté à la liste d’événements.

b.

Time	Delta	UUT/S1	UUT/S2	UUT/A	UUT/B	UUT/C	UUT/F	UUT/G
0 ps	0	U	U	U	U	U	U	U
0 ps	1	U	U	1	1	0	U	U
0 ps	2	1	0	1	1	0	U	U
0 ps	3	1	0	1	1	0	1	0
10000 ps	0	1	0	1	0	0	1	0
10000 ps	1	0	1	1	0	0	1	0
10000 ps	2	0	1	1	0	0	0	1

Question 4. (8 points)

Faites la conception d'un processeur spécialisé pour calculer la moyenne arithmétique d'une plage de cellules dans une mémoire RAM de 1024 cellules de 8 bits chacune. Les données en mémoire sont exprimées en complément à deux. La moyenne arithmétique est donnée par la somme des valeurs des cellules, divisée par le nombre de cellules.

Avant d'effectuer le calcul, le processeur doit d'abord recevoir, sur deux ports distincts et simultanément, l'adresse de la première cellule de la plage ainsi que le logarithme en base 2 du nombre de cellules entrant dans le calcul de la moyenne. Le nombre de cellules est donc toujours une puissance positive de 2. Le chargement est effectué quand un signal de contrôle spécial 'go' est activé, puis le processeur débute ses opérations.

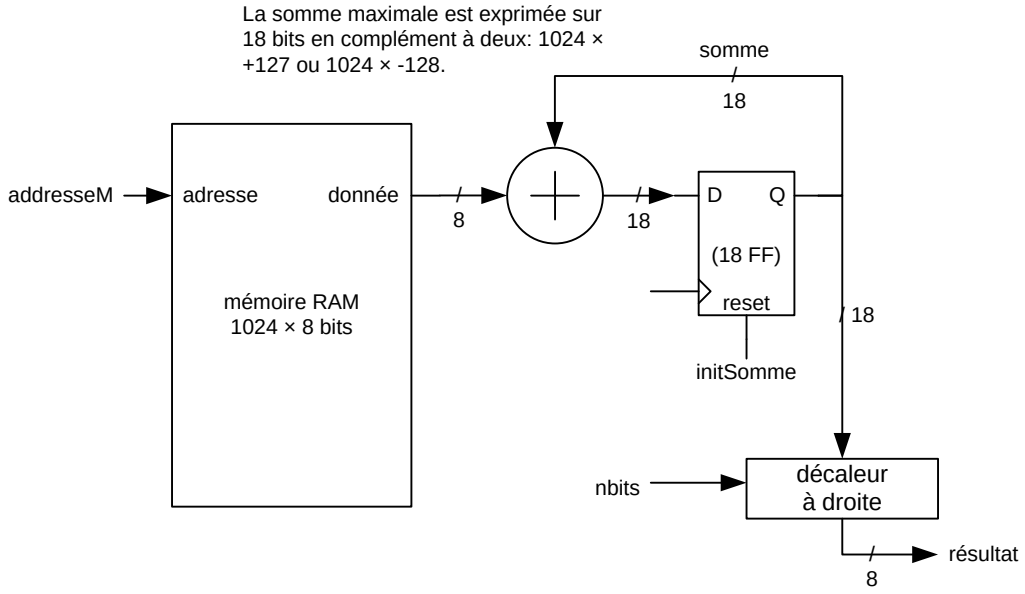
Le processeur doit effectuer le calcul, puis activer un signal spécial lorsque le résultat est appliqué à son port de sortie. Il doit aussi activer un signal d'erreur spécial à la sortie si la cellule de départ et le nombre de cellules spécifiés sont tels que la plage spécifiée dépasse la dernière cellule de la mémoire. La moyenne doit être exprimée sur 8 bits.

a. (4 points) Donnez un diagramme montrant le chemin des données de ce processeur. Identifiez bien chaque bloc et indiquez la largeur en bits de tous les ports et de tous les signaux.

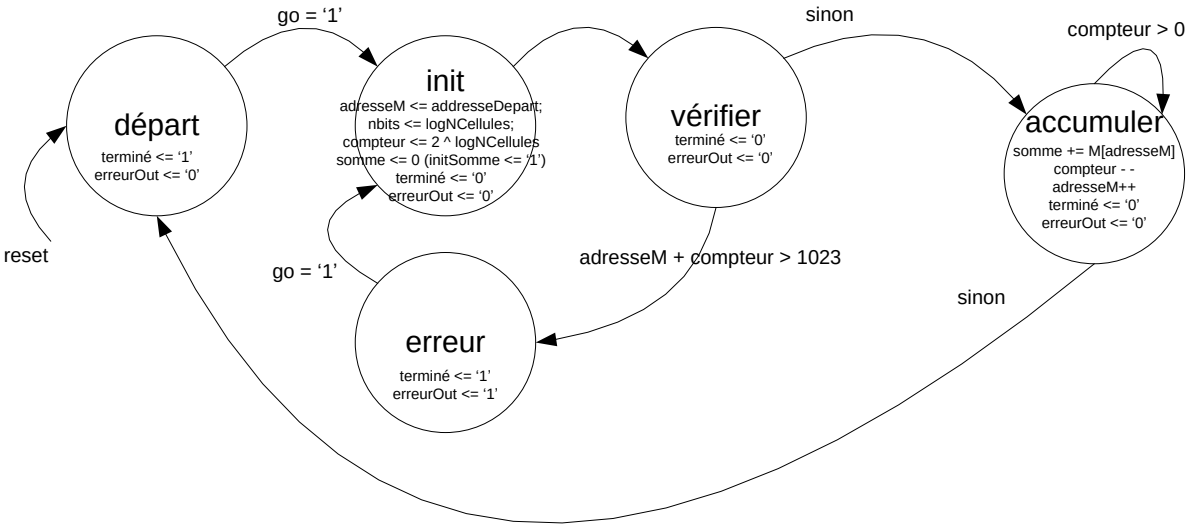
b. (4 points) Spécifiez l'unité de contrôle de ce processeur par un diagramme d'états. Identifiez bien les états et les conditions pour les transitions. Identifiez les signaux de contrôle appliqués au chemin des données.

Solution :

a.


La somme maximale est exprimée sur 18 bits en complément à deux: 1024 × +127 ou 1024 × -128.
somme
18
addresseM
adresse
donnée
8
18
D
Q
(18 FF)
reset
initSomme
mémoire RAM
1024 × 8 bits
18
nbits
décaleur
à droite
résultat
8

b.


go = '1'
sinon
compteur > 0
départ
terminé <= '1'
erreurOut <= '0'
init
adresseM <= addresseDepart;
nbits <= logNCellules;
compteur <= 2 ^ logNCellules
somme <= 0 (initSomme <= '1')
terminé <= '0'
erreurOut <= '0'
vérifier
terminé <= '0'
erreurOut <= '0'
accumuler
somme += M[adresseM]
compteur - -
adresseM++
terminé <= '0'
erreurOut <= '0'
reset
go = '1'
adresseM + compteur > 1023
sinon
erreur
terminé <= '1'
erreurOut <= '1'

Question 5. (6 points)

Réponses brèves.

a. (1 point) Énumérez quatre (4) éléments fondamentaux qu'on peut retrouver sur un FPGA présentement en vente sur le marché.

a. Solution

Pour 0.25 point chacun: blocs de logique programmable (CLB), blocs d'entrées-sorties (IOB), réseau d'interconnexions, circuits de routage rapide des retenues, blocs de mémoire intégrée, blocs de fonctions arithmétiques avancées (multiplicateurs, multiplicateurs-accumulateurs), microprocesseurs fixes, circuits de génération et de distribution d'horloge

Pour 0.125 point chacun : LUT, flip-flop, multiplexeur

b. (1 point) Lequel des énoncés suivants est vrai concernant des entités et architectures en VHDL?

i. un module peu avoir plusieurs déclarations d'entité mais une seule architecture;

ii. un module peut avoir seulement une déclaration d'entité mais plusieurs architectures;

iii. un module peut avoir seulement une déclaration d'entité et une seule architecture;

iv. un module peut avoir plusieurs déclarations d'entité et plusieurs architectures;

b. Solution : ii. Une entité, plusieurs architectures

c. (1 point) Considérez l'énoncé suivant : « Pour un circuit numérique implémenté sur FPGA, il n'y a pas de désavantage à utiliser le principe du pipeline au maximum, parce que les bascules sont à toutes fins pratiques gratuites. » Dites si vous êtes en accord ou en désaccord avec cet énoncé, et expliquez en quelques lignes pourquoi.

c. Solution

Plutôt en accord. Une fois qu'on a choisi un FPGA en particulier, il n'y a pas de coût associé à l'utilisation de chacune des ressources de la puce. Si le débit d'information doit être maximisé, alors le fait de pipeliner autant que possible devrait être encouragé, en utilisant toutes les bascules disponibles. Cependant, ajouter un grand nombre d'étages de pipeline augmente la latence du circuit, ce qui pourrait aller à l'encontre des spécifications.

d. (1 point) Lequel/lesquels des énoncés suivants est/sont vrai(s) concernant VHDL?

i. Les processus d'une architecture s'exécutent de façon concurrente.

ii. À l'intérieur d'un processus, les assignations à des signaux sont faites de façon concurrente.

iii. À l'intérieur d'un processus, les assignations à des variables sont faites de façon séquentielle.

iv. Dans une boucle ne comportant pas d'énoncé `wait`, chaque itération prend effectivement un temps nul.

v. Dans une architecture, une assignation concurrente à un signal est une façon compacte d'écrire un processus.

d. Solution : i, iii, iv, v.

e. (1 point) Énoncez deux types d'avertissements qu'un analyseur statique de code VHDL pourrait donner.

e. Solutions possibles : Opérandes de largeurs différentes dans une expression; Énoncés conditionnels dont tous les cas ne sont pas couverts et qui créent des états implicites; Énoncés conditionnels qui se recoupent; Nom d'entité différent du nom du fichier; Insertion de signaux de contrôle dans le chemin d'une horloge; Signaux ou variables qui ne sont pas initialisés avant d'être utilisés; Signaux ou variables auxquels on n'assigne jamais de valeur ou qui ne sont jamais utilisés; Expression constante dans une structure de condition, etc.

f. (1 point) Expliquez deux problèmes qui peuvent survenir si on utilise un commutateur d'une planchette d'expansion (contrôlé par un humain) comme entrée directe pour une machine à états implémentée dans un FPGA sur une carte de développement.

f. Solutions possibles: 1. Le commutateur rebondit, ce qui crée plusieurs entrées de suite qui changent de valeur. 2. Le commutateur peut être activé pendant une période interdite d'une bascule, la plaçant dans un état métastable. 3. Si plusieurs bascules dépendent de la valeur du commutateur, il est possible que certaines d'entre elles voient une valeur différente des autres si le commutateur est activé juste au mauvais moment, plaçant la machine à états dans un état interdit.

Nom : _____ Matricule : _____

Question 2

Détacher cette page, annotez-la et remettez-la avec votre cahier d'examen.