DS Architecture des ordinateurs JL Dekeyser Janvier 2016 DOCUMENTS AUTORISES

Exercice 1/3 HoMade 14 points

Le processeur HoMade permet d'enrichir ses fonctionnalités par l'intégration de nouveaux IPs. En TP vous avez créé un IP rdm. Celui-ci génère un nombre aléatoire suivant la méthode LSFR. Un code VHDL, partiel et simplifié, vous est donné en annexe.

Dans le calcul de PI avec la méthode Monté Carlo, vous produisez un nombre aléatoire de 32 bits sur la pile Une partie du code pour le calcul de PI est aussi donné en annexe (sans les I/O) celle cette partie sera prise en compte par la suite.

Question 1

Ce code de calcul de PI utilise le constructeur ***for next*** de l'assembleur Hasm. Cette construction de boucle est traduite en instructions ***>r*** et ***r>*** qui permettent de stocker et rappeler l'indice de boucle et les bornes supérieure et inférieure de la boucle sur ou depuis une seconde pile nommée ***datastack***. A chaque itération il y a un surcout à payer en nombre de cycle pour la gestion proprement dit de cette boucle. Il faut aussi avoir instancié l'IP ***datastack*** dans le processeur HoMade.

- Montrez que l'on peut se passer de ce constructeur ***for next*** et par la même occasion de l'IP ***datastack*** en gérant la boucle avec un ***begin until*** . Réécrire le code de calcul de PI.

```
homadeCode :    begin     // debut de boucle
        homadeCode?
        ( again | until )     //  until consomme le sommet de pile et répète le code
seulement si ce sommet est FAUX
```

Question 2

Dans le code de calcul de PI, plusieurs cycles sont nécessaires pour obtenir deux nombres aléatoires de 12 bits à partir du nombre aléatoire produit par rdm sur 32 bits. On utilise pour cela des IPs ***dup and*** et ***->***.

- A partir du code VHDL de l'IP rdm , proposez un nouvel IP rdm12x2 qui produit directement ces deux nombres de 12 bits sur deux ports de sortie au lieu d'un seul port avec les mêmes sous-chaines de bits que dans le code PI. Le second port sera nommé Nout.
- Quel doit être le code de cette instruction IP sur 16 bits si l'on produit maintenant deux valeurs sur la pile ?
- Modifiez le code Hasm du calcul de PI afin de prendre en compte ce nouvel IP rdm12x2 dans ce calcul.

Question 3

L'IP mul16 permet de faire une multiplication de 2 mots de 16 bits afin de produire un mot de 32 bits par multiplication. Il est utilisé tel que dans le code de square. Sur la nexys 3 j'ai utilisé un IPcore généré par ISE pour construire cet IP, le code est en annexe.

- Proposez un nouvel IP square en VHDL qui calcule le carré d'un seul nombre de 16 bit et produit un mot de 32 bits.
- Que gagne-t-on dans le code de PI, montrez la partie de code modifiée.

Question 4

HoMade permet de spécialiser des IPs en fonction de l'application : compromis surface/temps. Je vous propose de construire un IP doublesquare qui calcule en même temps le carré de deux nombres de 16 bits et produit en sortie deux nombres de 32 bits.

- Donnez le code VHDL de doublesquare.
- Donnez le code de l'IP correspondant.
- Modifiez le code Hasm de PI qui utilise cet IP.

Question 5

Afin de vérifier si le point obtenu aléatoirement est à l'intérieur ou non du cercle (théorème de Pythagore), le code Hasm calcule la somme des deux carrés et la compare à la valeur $FFE001 (Rayon $FFF au carré).

- Proposez un IP en VHDL qui produit directement en sortie le couple (1 0) ou (0 1) sur la pile en fonction de la valeur présente sur le sommet de pile et de cette constante $FFE001.
- Donnez le code de l'IP.
- Modifiez le code Hasm de PI en fonction.

Exercice 2/3 Traitement Out Of Order 4 points

Voici les caractéristiques d'un mini processeur pipeline et multi unités fonctionnelles.

1. deux instructions peuvent être lues, déclenchées et produire des résultats en même temps
2. les deux prochaines instructions sont chargées lorsque le buffer de décodage est vide
3. il existe 3 unités fonctionnelles en parallèles: * (2 clock), / (2 clock), + ou - (1 clock).
4. Les écritures sont effectives seulement après avoir traversées l'un des deux étages write

Voici le début déroulement dans l'ordre d'un programme sur ce processeur :

decode		/	*	+/-	write		CY
1	2						1
3	4		1	2			2
	4	3	1				3
	4	3			1	2	4
5	6			4	3		5
	6		5			4	6

```
1. R3=R0*R1
2. R4=R0+R2
3. R5=R0/R1
4. R6=R1+R4
5. R7=R1*R2
6. R1=R0-R2
7. R3=R3*R1
8. R1=R4+R4
```

- Compléter le tableau pour un traitement dans l'ordre.
- Proposez un renommage des registres pour éliminer les anti-dépendances.
- Remplir le tableau de réservation pour une exécution dans le désordre mais qui garantisse l'ordre des écritures dans les registres.
- Remplir le tableau pour une exécution et une écriture dans le désordre.

Exercice 3/3 les caches 2pts

Dessinez moi une mémoire cache set associative pour un processeur 32 bits de donnée et d'adresse avec 2 ensembles de 8 mots par ligne. Le cache à une taille de données de 32 Ko. Précisez le nombre de champs, la taille de chaque champ et le surcoût en taille dû aux TAGs.

ANNEXE

```
==========================================================
--IP RDM
entity IP_rdm is
                port (
                     clk , reset : in std_logic;
                     IPcode : in std_logic_vector (10 downto 0);
                     Tout : out std_logic_vector (31 downto 0)
                );

end IP_rdm;

architecture Rdm of IP_rdm is
signal EN    : std_logic := '0';
signal qbus : std_logic_vector (31 downto 0) ;
COMPONENT random
     PORT(
          clk, reset  : IN std_logic;
          enable : IN std_logic;
          random_num : out std_logic_vector (31 downto 0)
     );
END COMPONENT;
begin
     Inst_random: random
          PORT MAP(
          clk => clk,
          reset => reset ,
          enable => EN,
          random_num => qbus
     );
EN <= '1' when IPcode(10 downto 0) = Mycode else '0';
Tout <= (31 downto random_size => '0' )  & qbus  when
          IPcode(10 downto 0)= Mycode else
          (others =>'Z');
end Rdm;
==========================================================

// PI monte carlo avec TIC)
:IP rdm $8810 ;
: square
     dup mul16
;
: PI
     0  // le compteurs  inside
     $ffff   //nombre de tirages - 1
```

```
      for
            rdm  dup
            $fff and
            square
            swap
            12    ->
            $fff  and
            square
            +
            $FFE001
            <=
            if
                  ++
            endif
      next
;
================================================================

-- IP mul16
entity IP_mul16 is
    Port (
            Tin : in  STD_LOGIC_VECTOR (31 downto 0);
            Nin : in  STD_LOGIC_VECTOR (31 downto 0);
            IPcode : in  STD_LOGIC_vector(10 downto 0) ;
            Tout : out  STD_LOGIC_VECTOR (31 downto 0));
end IP_mul16;

architecture Behavioral of IP_mul16 is
component multiply
      port (
      a: in std_logic_vector(15 downto 0);
      b: in std_logic_vector(15 downto 0);
      p: out std_logic_vector(31 downto 0));
end component;
signal mult : std_logic_vector (31 downto 0) ;

-- Synplicity black box declaration
attribute syn_black_box : boolean;
attribute syn_black_box of multiply: component is true;
begin
my_mul16 : multiply
            port map (
                  a => Tin(15 downto 0),
                  b => Nin(15 downto 0),
                  p => mult);
Tout <= mult when Mycode = IPcode else (others =>'Z');
end Behavioral;
```