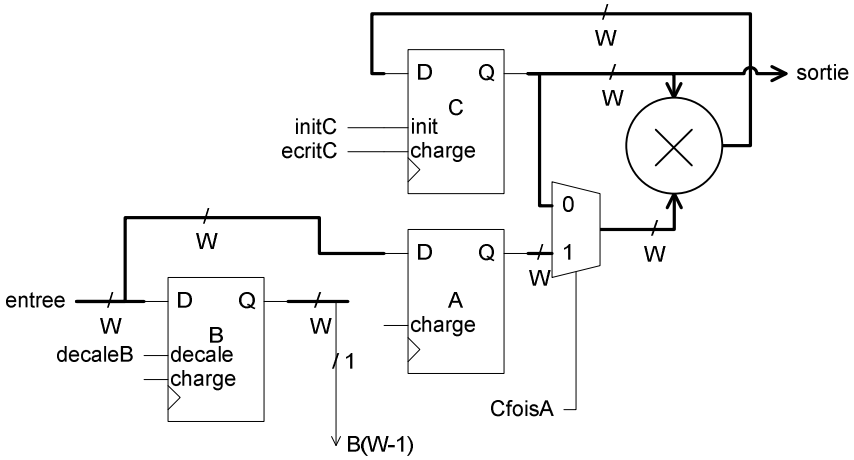
INF3500 : Conception et réalisation de systèmes numériques

Solutions du final

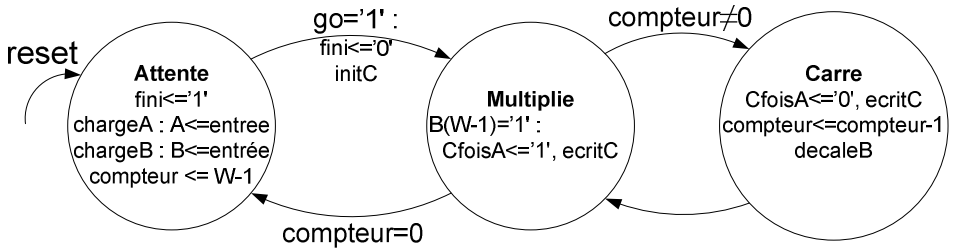
26 avril 2008, 9h30 – 12h

1.

a)



b)



c)

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

        constant compteurmax : positive := W - 1;
        signal compteur : natural range 0 to compteurmax;

        type type_etat is (attente, carre, multiplie);
        signal etat : type_etat;
begin
    process (CLK, reset)
    begin
        if rising_edge(CLK) then
            if reset = '1' then
                etat <= attente;
            else
                case etat is
                    when attente =>
```

```
                    fini <= '1';
                    if chargeA = '1' then
                        A <= entree;
                    end if;
                    if chargeB = '1' then
                        B <= entree;
                    end if;
                    compteur <= compteurmax;
                    if go = '1' then
                        fini <= '0';
                        C <= to_unsigned(1, W);
                        etat <= multiplie;
                    else
                        etat <= attente;
                    end if;
                when multiplie =>
                    if (B(W-1) = '1') then
                        C <= RESIZE(C*X, W); -- C <= RESIZE(C*A, W);
                    end if;
                    if (compteur = 0) then
                        etat <= attente;
                    else
                        etat <= carre;
                    end if;
                when carre =>
                    compteur <= compteur - 1;
                    C <= RESIZE(C*X, W); -- C <= RESIZE(C*C, W);
                    B <= shift_left(B, 1);
                    etat <= multiplie;
                when others =>
                    etat <= attente;
            end case;
        end if;
    end if;
end process;

X <= A when etat = multiplie else C;
-- signal de sortie
resultat <= C;
end arch;
```

2.

```
clk <= not clk after period / 2;

process
    variable A : natural range 0 to 2 ** W - 1 := 0;
    variable B : natural range 0 to 2 ** W - 1 := 0;
    variable AexposantB : natural range 0 to 2 ** W - 1 := 0;

    variable seed1 : positive := 1;
    variable seed2 : positive := 2;
    variable aleatoire : real;

begin
    chargeA <= '0';
    chargeB <= '0';
    go <= '0';

    reset <= '1';
    wait until rising_edge(clk);
    wait until falling_edge(clk);
    reset <= '0';

    while true loop
        uniform(seed1, seed2, aleatoire);
        A := integer(floor(aleatoire * real(2 ** (W/2)))); -- 0 <= t <= 2**(W/2)
        uniform(seed1, seed2, aleatoire);
        B := integer(floor(aleatoire * real(2 ** (W/2)))); -- 0 <= t <= 2**(W/2)
        AexposantB := (A**B) mod (2**W);

        wait until falling_edge(clk);
        chargeA <= '1'; chargeB <= '0';
        entree <= to_unsigned(A, W);

        wait until falling_edge(clk);
        chargeA <= '0'; chargeB <= '1';
        entree <= to_unsigned(B, W);
        go <= '1';

        wait until falling_edge(clk);
        go <= '0';

        wait until rising_edge(fini);
        assert to_integer(resultat) = AexposantB report "erreur!" severity failure;
    end loop;
end process;
```

3.
a) Pour JAL il faut pouvoir envoyer le PC au bloc de registres (ajouter un choixSource pour PC). Pour JR il faut que le contrôleur puisse envoyer la valeur d'un registre dans le PC (B envoyé au contrôleur). Le contrôleur doit faire une adition sur le PC, plutôt que d'y écrire la valeur venant de l'instruction. JAL et JR peuvent être comme les autres branchements, avec un numéro « condition » différent. Ici on a choisi "0101" pour JAL, il faudra faire attention qu'il soit considéré comme inconditionnel.

b)
machine à états, dans « when jump, if… then »:
PC <= PC + to_integer(signed(IR(7 downto 0)));

c)
multiplexeur :
changer le « range » de 3 à 4;
ajouter : when 4 => donnee <= PC;

signaux de contrôle :
with etat select choixSource <= 0 when opUAL, 4 when jump, 3 when others;
choixCharge <= 15 when etat = jump else to_integer(unsigned(IR(11 downto 8)));
charge <= '1' when etat = opUAL or etat = lireMemoire or (etat = jump and IR(11 downto 8) = "0101") else '0';

machine à états, dans « when jump, if… »:
ajouter: or (IR(11 downto 8) = "0101")

4.
a)
Chemin long : A/B→xor→nand→nand→D d'un délai de 2+5+2+2 = 11ns.
Chemins courts : A/B/C→nand→nand→D d'un délai de 2+2+2 = 6ns.
D→C d'un délai de 2ns.
Avec le temps de préparation de 1ns, il faut une période d'au moins 11+1 = 12ns (83MHz).

b)
Oui, par « retiming » on peut déplacer la bascule « D » et la remplacer par 4 bascules aux entrées du premier niveau de portes nand. Le chemin critique devient alors 5ns (une porte xor), donc 6ns de période (166MHz).