June 6, 2007

COMPUTER ENGINEERING DEPARTMENT

COE 405

DESIGN & MODELING OF DIGITAL SYSTEMS

Final Exam

Second Semester (062)

Time: 7:30-10:30 AM

(Open Book Exam)

Student Name : _____

Student ID. : _____

Question	Max Points	Score
Q1	24	
Q2	15	
Q3	25	
Q4	36	
Total	100	

Dr. Aiman El-Maleh

[24 Points]

(Q1)

(i) Given the Entity description of a **D-Latch** below:

```
Entity Dlatch IS
Port  (D, CLK: IN BIT; Q, QB : Buffer  BIT);
End;
```

Consider each of the following VHDL Architecture models and determine if it correctly models a D-latch or not. If a model does not model a D-Latch correctly, **Correct it**.

a.

```
Architecture Model1 of Dlatch is
Begin
   Q <= D when CLK = '1' else Q;
   QB <= Not Q;
End Model1;
```

b.

```
Architecture Model2 of Dlatch is
Begin
   t1 <= D Nand clk ;
   t2 <= (Not D) Nand clk ;
   Q <= t1 Nand QB;
   QB <= t2 Nand Q;
End Model2;
```

c.

```
Architecture Model3 of Dlatch is
Begin
Process(D, CLK)
Begin
   Q <= D when CLK = '1' else Q;
   QB <= Not D when CLK = '1' else QB;
End Process;
End Model3;
```

d.

```
Architecture Model4 of Dlatch is
Begin
Process(D)
Begin
   If CLK = '1' then
         Q <= D ;
         QB <= Not Q;
   End If;
End Process;
End Model4;
```

(ii) Given the Entity description of a **D-FF** below:

```
Entity DFF IS
Port  (D, Reset, CLK: IN BIT; Q, QB : Buffer  BIT);
End;
```

Consider each of the following VHDL Architecture models and determine if it correctly models a D-FF or not. If a model does not model the specifiied D-FF correctly, Correct it.

a. Rising-edge Triggered D-FF with synchronous Reset.

```
Architecture Model1 of DFF is
Begin
DF: Block  (clk = '1' and CLK'Event)
Begin
        Q <= Guarded '0' when reset='1' else D ;
        QB <= Not Q;
End Block;
End Model1;
```

b. Rising-edge Triggered D-FF with Asynchronous Reset.

```
Architecture Model2 of DFF is
Begin
DF: Block  (clk = '1' and Not CLK'Stable)
Begin
        Q <= Guarded '0' when reset='1' else D ;
        QB <= Not Q;
End Block;
End Model2;
```

c. Rising-edge Triggered D-FF with Synchronous Reset.

```
Architecture Model3 of DFF is
Begin

        Q <= '0' after 2 ns when (reset='1' and clk='1' and not
             clk'stable)  else D after 2 ns  when ( clk='1' and not
             clk'stable)  ;

      QB <= Not Q;
End Model3;
```

d. Falling-edge Triggered D-FF with Synchronous Reset.

```
Architecture Model4 of DFF is
Begin
Process
Begin
        Wait on CLK;
        If Reset = '1' then
                Q <= '0' after 2 ns;
        Else
                Q <= D after 2 ns;
        End If;
End Process;
QB <= Not Q;
End Model4;
```

[15 Points]

(Q2) Determine the circuit that will be produced when synthesizing each of the following VHDL codes:

(i)

```
Entity FinalQ3i is
        PORT(A, B, C, D: IN Bit;
                E: OUT Bit);
End;
Architecture Test of FinalQ3i is
Begin
        Process(A, B, C, D)
        Begin
                if (A'Event and A='0') Then
                        if (B=C) Then
                                E <= D;
                        Else
                                E <= Not D;
                        End if;
                End if;
        End Process;
End;
```

(ii)

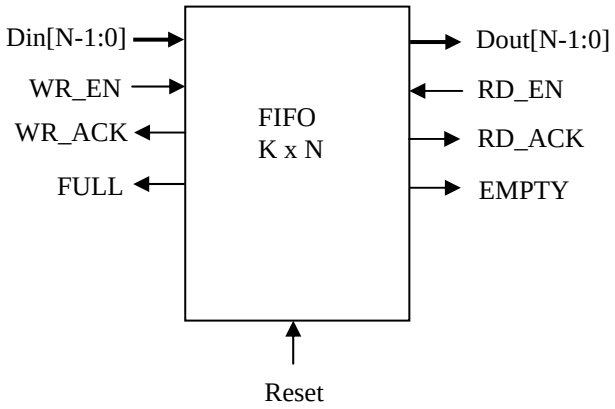
```
Entity FinalQ3ii is
        PORT(A, B, C, D: IN Bit;
                E: OUT Bit);
End;
Architecture Test of FinalQ3ii is
Begin
Process(A, B, C, D)
variable t: Bit_Vector(1 downto 0);
Begin
        t := A&B;
        Case t is
        when "00" => E <= C AND D;
        when "01" => E <= C OR D;
        when "10" => E <= C XOR D;
        when "11" => Null;
        End Case;
End Process;
End;
```

(iii)

```
Library IEEE;
Use ieee.std_logic_1164.all;
Entity FinalQ3iii is
        PORT(A, B, C, D: IN std_logic;
                E: OUT std_logic);
End;
Architecture Test of  FinalQ3iii is
        Signal t: std_logic;
Begin
        Process
        Begin
                wait until A='1';
                t <= B;
        End Process;
        E <= C when D='1' Else 'Z';
        E <= t  when D='0' Else 'Z';
End;
```

[25 Points]

(Q3) It is required to model an **Asynchronous FIFO (First-In-First-Out) memory queue**. The FIFO has a parametrizable memory depth of upto **K** loctions with a parametrizable data width of **N** bits. The FIFO interface is given below:



The Reset signal is an **Asynchronous reset** and when set to 1 it will consider the content of FIFO empty and set the EMPTY flag to one and all other flags to 0. A handshaking mechanism is used for both writing and reading from the FIFO using the signal WR_EN, WR_ACK, RD_EN and RD_ACK. When WR_EN=1, the data in DIN input will be written to the available location in the FIFO as long as the FULL signal is not set to 1. If the FIFO is FULL the request is ignored and not acknolwdged. It is assumed that the WR_EN signal will remain 1 until a WR_ACK is set to 1 by the FIFO. After that, the WR_EN signal will go to 0. A similar handshaking mechanism is used for reading from the FIFO. When RD_EN=1, the data in the proper location will be output to DOUT as long as the EMPTY signal is not set to 1. Then, the RD_ACK signal is set to 1. If the FIFO is EMPTY the request is ignored and not acknolwdged. It is very important to note that the **FIFO should be able to read and write simultaneously** if needed.

HINT: Use a read pointer to point at the location to be read from and a write pointer to point at the location to be written to. Use also a counter to keep track of when the FIFO is full or empty.

(i) Write an entity description of the FIFO using generic parametrs for the FIFO depth **K**, and data width **N**. Also, use a generic parameter **Delay** for the time needed to read or write data from the queue. Use std_logic and std_logic_vector for all signals in the FIFO.

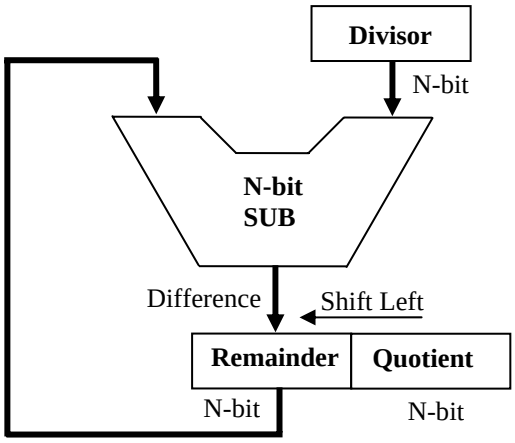
(ii) Write an architecture modeling the correct functionality of the FIFO using **behaviorl** description.

[36 Points]

(Q4) It is required to model an entity to perform **unsigned division** of an N-bit dividend number, A, by an N-bit divisor number, B. The divider produces an N-bit quotient and an N-bit remainder. Assume that the divider will be a **sequential divider** and it will set the signal Ready to 1 when the quotient and remainder results are ready. The divider has an **asynchronous reset** after which in the next clock cycle it starts the division process. The quotient and remainder willl maintain their values unless the divider is reset again. The entity of the divider is given below:

```
Entity UDIV is
        Generic (N: natural := 4);
        Port (Reset, Clk: IN std_logic;
                A, B : IN std_logic_vector(N-1 downto 0);
                Q, R : OUT std_logic_vector(N-1 downto 0);
                Ready: OUT std_logic);
End;
```

Part of the Datapath of the divider is given below:



The algorithm for performing sequential division is as follows:

1. Set Quotient=Dividened, Set Remainder=0.
2. Shift(Remainder,Quotient) Left by 1 bit
3. Difference=Remainder-Divisor
4. If (Difference≥ 0) Then
   Remainder=Difference
   Set Least Significant bit of Quotient to 1.
   End If;
5. If (#iterations<N) Then Goto Step 2.

An example of applying the algorithm for a 4-bit divider with dividened=1110 and divisor=0011 is given below. Note that the Quotient=0100 and the Remainder=0010.

Iteration		Remainder	Quotient	Divisor	Difference
0	Initialize	0 0 0 0	1 1 1 0	0 0 1 1	
1	1: SLL, Difference	0 0 0 1 ←	1 1 0 0	0 0 1 1	1 1 1 0
	2: Diff < 0 => Do Nothing				
2	1: SLL, Difference	0 0 1 1 ←	1 0 0 0	0 0 1 1	0 0 0 0
	2: Rem = Diff, set lsb Quotient	0 0 0 0	1 0 0 1		
3	1: SLL, Difference	0 0 0 1 ←	0 0 1 0	0 0 1 1	1 1 1 0
	2: Diff < 0 => Do Nothing				
4	1: SLL, Difference	0 0 1 0 ←	0 1 0 0	0 0 1 1	1 1 1 1
	2: Diff < 0 => Do Nothing				

(i) Write an entity description of the **datapath** of the divider. Then, write an architecture model for the datapath of the divider using **dataflow** modeling (i.e. using Block statements and signal assignments).

(ii) Write an entity description of the **control unit** of the divider. Describe the control unit of the divider as a **finite state machine** and draw its state diagram. Then, write an architecture model for the control unit of the divider using **behavioral** modeling.

(iii) Write an architecture model for the **divider unit**. Use configuration statements to configure the used components.