COE 405, Term 162

Design & Modeling of Digital Systems

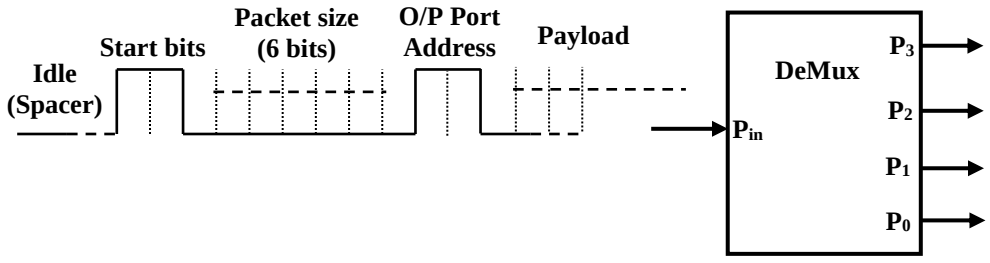
Assignment# 3 Solution

Due date: Sunday, April 9

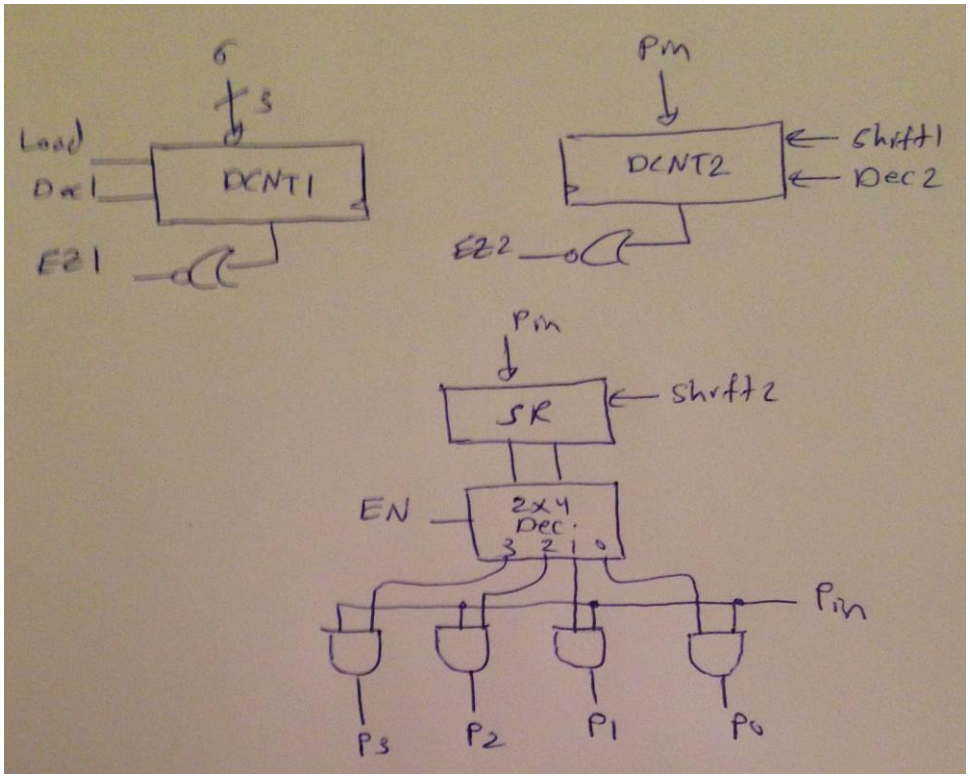
It is required to design a simple network DeMux. The circuit has one input and four outputs (as illustrated in the figure below). Data packets arrive serially on the input and are routed to one of the four outputs:

- Packets have 10-bits headers,
- When the input line is idle (i.e. no data is arriving), it is kept low,
- A packet has two start bits (11), followed by 6-bits specifying the packet size in bits (i.e. from 0 to 63 bits), followed by two bits specifying the address of the output port (0 to 3), and immediately followed by the payload (i.e., packet data),
- The DeMux strips out the header before outputting the packet's payload,
- Whenever data is not transmitted across any of the output ports, its output line will be 0,
- Assume that packet size, port address and data will be transmitted from least significant to most significant bits,
- The circuit should be fully synchronous with an external clock and have a master asynchronous reset input.

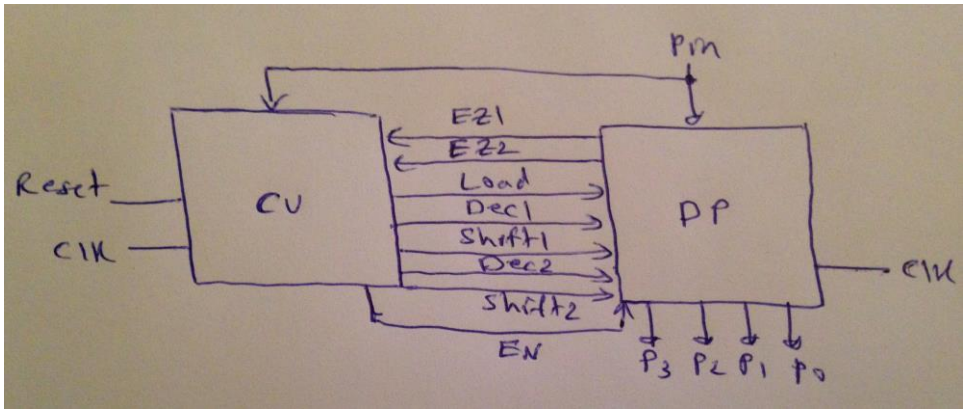
The diagram below shows the block diagram of the circuit and the data format.



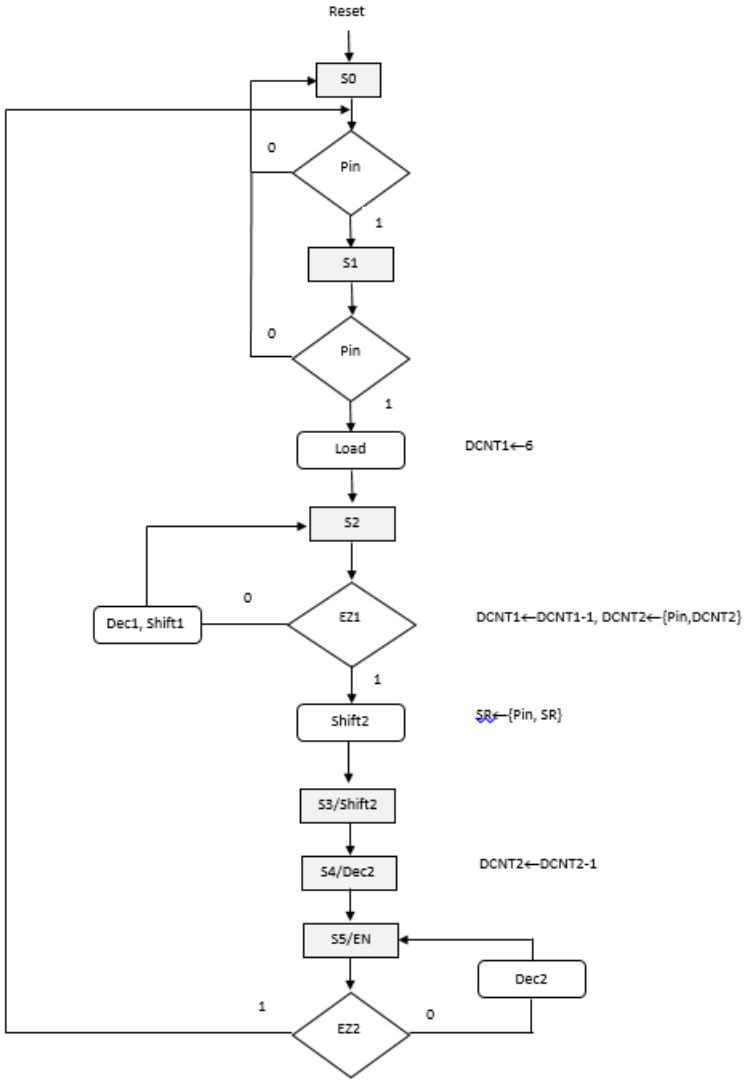
(i) Determine the required data path blocks to design this circuit along with their control signals (show the block diagram of these circuits only).



(ii) Show the block diagram of the data path and control unit and all the interface signals.



(iii) Obtain the ASMD chart of the control unit that will control the operation of the circuit.



(iv) Design the control unit using D-FFs and any other components of your choice.

State Table:

C.S.	Input			N.S.	Output					
	Pin	EZ1	EZ2		Load	Dec1	Shift1	Shift2	Dec2	EN
S0	0	X	X	S0	0	0	0	0	0	0
S0	1	X	X	S1	0	0	0	0	0	0
S1	0	X	X	S0	0	0	0	0	0	0
S1	1	X	X	S2	1	0	0	0	0	0
S2	X	0	X	S2	0	1	1	0	0	0
S2	X	1	X	S3	0	0	0	1	0	0
S3	X	X	X	S4	0	0	0	1	0	0
S4	X	X	X	S5	0	0	0	0	1	0
S5	X	X	0	S5	0	0	0	0	1	1
S5	0	X	1	S0	0	0	0	0	0	1
S5	1	X	1	S1	0	0	0	0	0	1

We use the binary state assignment S0=000, S1=001, S2=010, S3=011, S4=100, S5=101 and the three Flip Flops, F2 F1 F0, to encode states.

Transition Table:

C.S.	Input			N.S.	Output					
F2 F1 F0	Pin	EZ1	EZ2	F2 F1 F0	Load	Dec1	Shift1	Shift2	Dec2	EN
0 0 0	0	X	X	0 0 0	0	0	0	0	0	0
0 0 0	1	X	X	0 0 1	0	0	0	0	0	0
0 0 1	0	X	X	0 0 0	0	0	0	0	0	0
0 0 1	1	X	X	0 1 0	1	0	0	0	0	0
0 1 0	X	0	X	0 1 0	0	1	1	0	0	0
0 1 0	X	1	X	0 1 1	0	0	0	1	0	0
0 1 1	X	X	X	1 0 0	0	0	0	1	0	0
1 0 0	X	X	X	1 0 1	0	0	0	0	1	0
1 0 1	X	X	0	1 0 1	0	0	0	0	1	1
1 0 1	0	X	1	0 0 0	0	0	0	0	0	1
1 0 1	1	X	1	0 0 1	0	0	0	0	0	1

The output signals equations are as follows:

Load = F2' F1' F0 Pin

Dec1 = Shif1 = F2' F1 F0' EZ1'

Shif2 = F2' F1 F0' EZ1 + F2' F1 F0

Dec2 = F2 F1' F0' + F2 F1' F0 EZ2'

EN = F2 F1' F0

The Flip-Flop Equations are as follows:

F2+ = F2' F1 F0 + F2 F1' F0' + F2 F1' F0 EZ2'

F1+ = F2' F1' F0 Pin + F2' F1 F0'

F0+ = F2' F1' F0' Pin + F2' F1 F0' EZ1 + F2 F1' F0' + F2 F1' F0 EZ2' + F2 F1' F0 Pin

(v) Model the data path in Verilog.

```
module DMUX_DP (output P3, P2, P1, P0, EZ1, EZ2, input Pin, Load,
Dec1, Shift1, Dec2, Shift2, EN, CLK);


// Down Counter 1
```

```
reg [2:0] DCNT1;

assign EZ1 = ~|DCNT1;

always @(posedge CLK)

if (Load) DCNT1 = 6;
else if (Dec1) DCNT1 = DCNT1 - 1;


// Down Counter 2

reg [5:0] DCNT2;

assign EZ2 = ~|DCNT2;

always @(posedge CLK)

if (Shift1) DCNT2 = {Pin, DCNT2[5:1]};
else if (Dec2) DCNT2 = DCNT2 - 1;


// 2-bit Shift Register

reg [1:0] SR;

always @(posedge CLK)
if (Shift2) SR = {Pin, SR[1]};


// 2x4 Decoder with enable

reg O0, O1, O2, O3;

always @(SR, EN)
if (!EN) begin O0=0; O1=0; O2=0; O3=0; end
else begin
O0=0; O1=0; O2=0; O3=0;
case (SR)
2'b00: O0=1;
2'b01: O1=1;
2'b10: O2=1;
2'b11: O3=1;
endcase
end

// DMUX Output Signals

assign P0 = Pin & O0;
assign P1 = Pin & O1;
assign P2 = Pin & O2;
assign P3 = Pin & O3;



endmodule
```

(vi) Model the control unit in Verilog based on your derived implementation in (iv).

Although, it was not asked for in the assignment, a behavioral model of the control unit was developed first to ensure correctness. The behavioral model is given below:

```
module DMUX_CU (output reg Load, Dec1, Shift1, Dec2, Shift2, EN,
input Pin, EZ1, EZ2, CLK, Reset);

parameter S0 = 3'b000, S1=3'b001, S2=3'b010, S3=3'b011,
S4=3'b100, S5=3'b101;

reg [2:0] state, next_state;

always @(posedge CLK)
   if (Reset) state <= S0;
   else state <= next_state;

always @(state, Pin, EZ1, EZ2) begin
   Load=0; Dec1=0; Shift1=0; Dec2=0; Shift2=0; EN=0;
   case (state)
      S0:
      if (Pin)
         next_state=S1;
      else next_state=S1;
      S1:
      if (Pin)
       begin  next_state=S2; Load=1; end
      else next_state=S0;
      S2:
      if (EZ1)
       begin Shift2=1; next_state=S3; end
      else begin Shift1=1; Dec1=1; next_state=S2; end
      S3:
      begin Shift2=1; next_state=S4;  end
      S4:
      begin
       Dec2=1; next_state=S5;
      end
      S5:
      begin EN=1;
      if (EZ2)
         if (Pin) next_state=S1; else next_state=S0;
      else begin Dec2=1; next_state=S5; end
      end
endcase

end


endmodule
```

Structural Control Unit Model:

```
module DFF (output reg Q, input D, CLK, Reset);

always @(posedge CLK, posedge Reset)
if (Reset) Q = 1'b0;
else if (CLK) Q = D;

endmodule

module DMUX_CUS (output  Load, Dec1, Shift1, Dec2, Shift2, EN,
input Pin, EZ1, EZ2, CLK, Reset);

DFF FF0 (F0, D0, CLK, Reset);
DFF FF1 (F1, D1, CLK, Reset);
DFF FF2 (F2, D2, CLK, Reset);

assign D2 = ~F2 & F1 & F0 | F2 & ~F1 & ~F0 | F2 & ~F1 & F0 &
~EZ2;

assign D1 = ~F2 & ~F1 & F0 & Pin | ~F2 & F1 & ~F0;

assign D0 = ~F2 & ~F1 & ~F0 & Pin | ~F2 & F1 & ~F0 & EZ1 | F2 &
~F1 & ~F0 | F2 & ~F1 & F0 & ~EZ2 | F2 & ~F1 & F0 & Pin;

assign Load = ~F2 & ~F1 & F0 & Pin;

assign Dec1 = ~F2 & F1 & ~F0 & ~EZ1;

assign Shift1 = ~F2 & F1 & ~F0 & ~EZ1;

assign Shift2 = ~F2 & F1 & ~F0 & EZ1 | ~F2 & F1 & F0;

assign Dec2 = F2 & ~F1 & ~F0 | F2 & ~F1 & F0 & ~EZ2;

assign EN = F2 & ~F1 & F0;


endmodule
```

(vii) Write a Verilog test bench to test the correct functionality of your implementation of the circuit. You test bench should send two packets one of size 4 bits addressed to port 0 transmitting the Hex. Value B and the second packet is of size 8 bits sent to port 2 transmitting the decimal value 9A.

The DMUX module is first modeled as shown below combining the data path and control unit:

```
module DMUX (output P3, P2, P1, P0,input Pin, CLK, Reset);
```

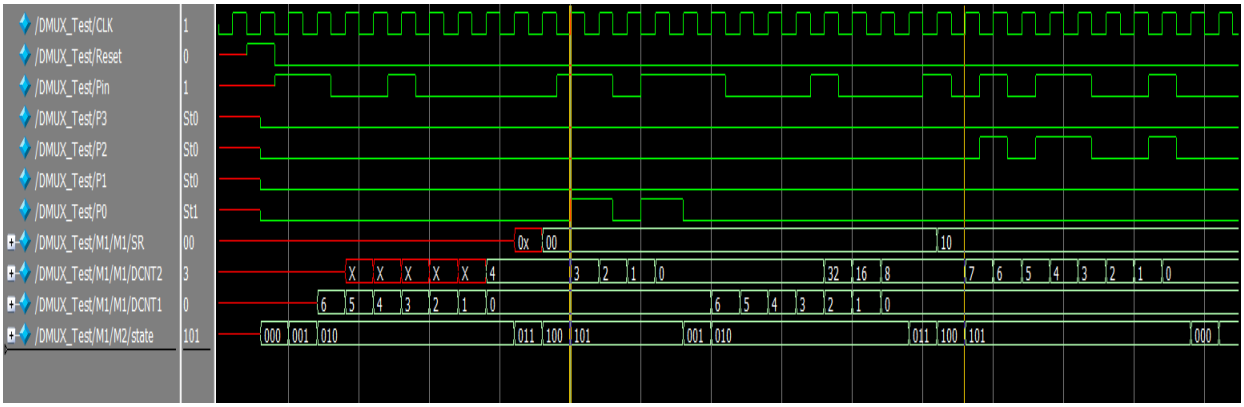
```
DMUX_DP M1 (P3, P2, P1, P0, EZ1, EZ2, Pin, Load, Dec1, Shift1,
Dec2, Shift2, EN, CLK);

DMUX_CU M2 (Load, Dec1, Shift1, Dec2, Shift2, EN, Pin, EZ1, EZ2,
CLK, Reset);

endmodule
```

Then, a test bench was developed to test the DMUX module as required. The test bench is given below:

```
module DMUX_Test();


wire P3, P2, P1, P0;
reg Pin, CLK, Reset;

DMUX M1 (P3, P2, P1, P0, Pin, CLK, Reset);

initial begin
CLK=0;
forever
#100 CLK = ~ CLK;
end

initial begin

// Resetting the machine

@(negedge CLK) Reset=1;

// Sending 1st Packet

@(negedge CLK) Reset=0; Pin=1; // sending packet header
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0; // sending packet size
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0; // sending port address
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1; // sending packet data
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1;

// Sending 2nd Packet

@(negedge CLK) Pin=1; // sending packet header
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0; // sending packet size
@(negedge CLK) Pin=0;
```

```
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0; // sending port address
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0; // sending packet data
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=1;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=0;
@(negedge CLK) Pin=1;

// Making the line idle

@(negedge CLK) Pin=0;

end

endmodule
```

As can be seen, from the simulation waveform given below, the DMUX is functioning correctly as per requirements: Behavioral Control Unit Model:



Structural Control Unit Model: