Professor Fearing EECS150/Problem Set 3 Solution Fall 2013

Due at 10 am, Thu. Sep. 26

Reading Harris and Harris: 3.4, 4.9

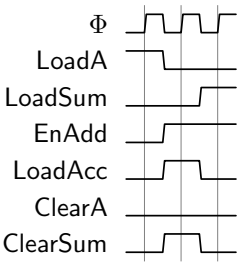
1. **(20 pts) Datapath and control I.** For the datapath below, determine a sequence of operations which will exchange the contents of `Acc` and `Sum`. (Contents of register `A` can be discarded). The data path is 8 bits wide. Φ is the clock.

   a. Describe the necessary operations using a Register Transfer Level (RTL) description. How many clock cycles are required?

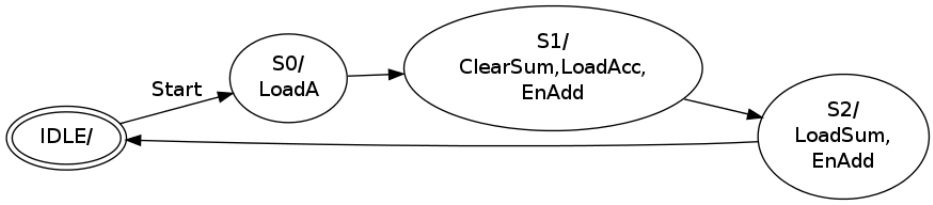
   ```
   A ← Acc;
   Sum ← 0, Acc ← Sum;
   Sum ← Sum + A;
   ```

   3 clock cycles are required.

   b. Draw a functional timing diagram for the circuit including Φ, `LoadA`, `LoadSum`, `EnAdd`, `LoadAcc`, `ClearA`, `ClearSum` which shows the RTL operation of part a.



   c. Draw a state diagram for a controller which starting in Idle, when given a Start signal, will swap `Acc` and `Sum` and then go back to the Idle state.



   Signals are only listed if asserted (i.e. set to 1).

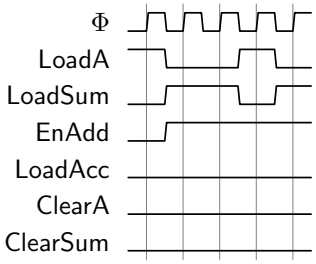
2. **(30 pts) Datapath and control II.** For the datapath below, determine a sequence of operations which will perform the operation `z = 4 * x + 2 * y`, where initial values are `Acc = x`, `Sum = y` and answer `z` is stored in Sum.

   a. Describe the necessary operations using a Register Transfer Level description. What is the minimum number of clock cycles required?

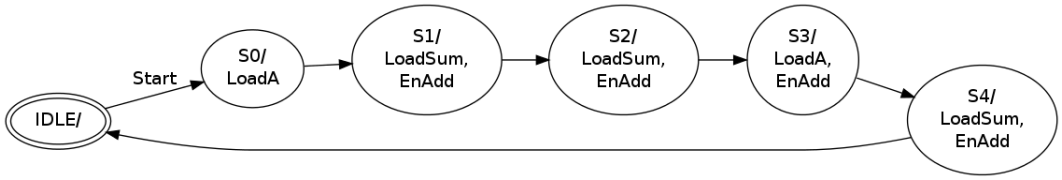
   ```
   A ← Acc;
   Sum ← Sum + A;
   Sum ← Sum + A;
   A ← Sum;
   Sum ← Sum + A;
   ```

   5 clock cycles are required.

b. Draw a functional timing diagram for the circuit including Φ, `LoadA`, `LoadSum`, `EnAdd`, `LoadAcc`, `ClearA`, `ClearSum` which shows the RTL operation of part a.



c. Draw a state diagram for a controller which given a Start signal, will perform the operation, and then go back to the Idle state.



d. Implement the control FSM and data path as separate Verilog modules.

```
module Q2_DP(
  input Phi, LoadA, LoadSum, EnAdd,
  LoadAcc, ClearA, ClearSum);

localparam x=2, y=3; //Initialization values optional
reg [7:0] A, Sum=y, Acc=x;
wire [7:0] DataBus;

assign DataBus = EnAdd ? Sum : Acc;

always @(posedge Phi) begin
  if(ClearA) A <= 0;
  else if(LoadA) A <= DataBus;

  if(ClearSum) Sum <= 0;
  else if(LoadSum) Sum <= A + DataBus;

  if(LoadAcc) Acc <= DataBus;
end
endmodule

module Q2_FSM(
  input Phi,Start,
  output LoadA, LoadSum, EnAdd,
  LoadAcc, ClearA, ClearSum);

// Should either initialize state, or include a Reset
localparam IDLE=3'd7;
reg [2:0] State = IDLE, NextState;

assign LoadA    = (State==0) | (State==3);
```

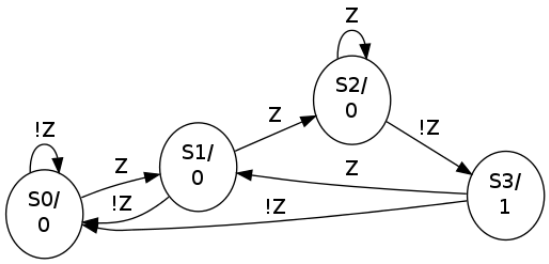
```
assign LoadSum  = (State==1) | (State==2) | (State==4);
assign EnAdd    = (State==1) | (State==2) | (State==3) | (State==4);
assign LoadAcc  = 0;
assign ClearA   = 0;
assign ClearSum = 0;

always @(posedge Phi) State <= NextState;

always @(*)begin
  if(State==IDLE) NextState = Start ? 0 : IDLE;
  else if(State==4) NextState = IDLE;
  else NextState = State + 1;
end
endmodule
```

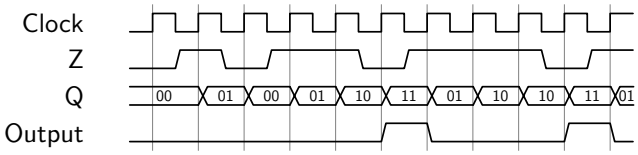
3. **(25 Points) FSM Design.** Design a Moore type FSM for a sequence recognizer. The FSM should assert Output for every sequence "110" detected in the synchronized serial input stream `Z`.

   a. Show state diagram for sequence recognizer.



   b. Show timing diagram (`Clock`, `Z`, `Q[2:0]`, `Output`) for the input sequence "01011011101", with left most bit arriving first.

   For the following, the state encoding matches the state index (S0 = 2'b00, ...,S3=2'b11).



   c. Implement the FSM in Verilog.

```
module Q3_FSM(
  input Clock, Z, Reset,
  output Output,
  output reg [1:0] State);

reg [1:0] NextState;

always@(posedge Clock) begin
  if(Reset) State <= 0;
  else State <= NextState;
end

assign Output = State==3;
```

```
  always@(*) begin
    case(State)
      0: NextState = Z ? 1 : 0;
      1: NextState = Z ? 2 : 0;
      2: NextState = Z ? 2 : 3;
      3: NextState = Z ? 1 : 0;
    endcase
  end
  endmodule
```

4. **(25 pts) Verilog Testbench** Write a Verilog test bench for the FSM in question 3, making sure that all state transitions are covered.

```
‘timescale 1ns/1ps
module Q4_Test();

reg Clock=0, Reset=1;
reg [11:0] Z          = 12'b010011011100;
reg [11:0] ExpOutAll  = 12'b000100100000;
reg [23:0] ExpStateAll = 24'b00_01_00_11_10_01_11_10_10_01_00_00;
wire Output,ExpOut;
wire [1:0] State,ExpState;
integer idx;

Q3_FSM dut(
  .Clock(Clock),
  .Reset(Reset),
  .Z(Z[idx]),
  .State(State),
  .Output(Output));

always #5 Clock = ~Clock;

// Select the current expected values from list of all expected values
assign ExpOut = ExpOutAll[idx];
assign ExpState = {ExpStateAll[2*idx+1],ExpStateAll[2*idx]};

initial #10 Reset = 0;

initial begin
  idx = 0;
  repeat(12) begin
    #6 $display("idx:%d␣O:%b␣EO:%b␣S:%b␣ES:%b",
      idx,Output,ExpOut,State,ExpState);
    if(Output !== ExpOut | State !== ExpState)
      $display("FAIL:␣Output␣or␣state␣mismatch␣at␣idx:$d",idx);
    #4 idx = idx + 1;
  end
  $stop();
end
endmodule
```