Ø 週五

Hardware Labe Final

1. FF/latch:
   (A) Design a positive edge FF with asynchronous reset and a Latch with synchronous reset using verilog (1~~0~~5%)
   (B) Describe what the difference between Flip-Flop and latch. (10%)
   (C) When you synthesis your design using Xlinx, sometimes there will be latches in your design. When latches are found, Xlinx will display warning messages because of these latches. Explain briefly how you eliminate latches in your design. (10%) 5 word 內

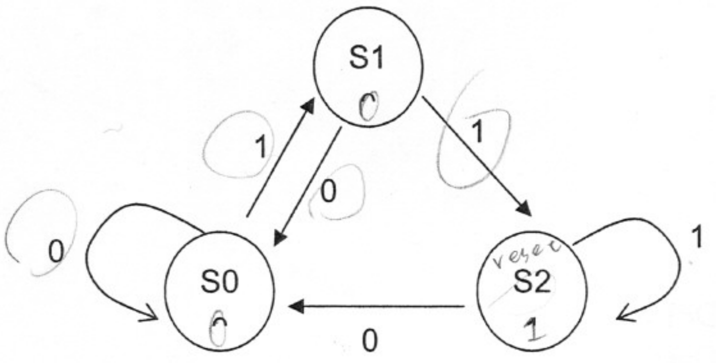
```
module AA(q, clk, reset);
output [3:0] q;
input clk, reset;
T_FF tff0 (q[0], clk, reset);
T_FF tff1 (q[1], q[0], reset);
T_FF tff2 (q[2], q[1], reset);
T_FF tff3 (q[3], q[2], reset);
Endmoudle
---------------------------------
module T_FF(q, clk, reset);
output q;   input clk, reset; wire d;
D_FF dff0 (q, d, clk, reset);
```

```
not n1(d, q);
endmodule
----------------------------------------
module D_FF(q, d, clk, reset);
output q;
input d, clk, reset;
reg    q;

always@(posedge reset or negedge clk)
if (reset)          q=1'b0;
else          q = d;
endmodule
```

Figure 1

2. Draw the schematic of Figure 1 and describe the functionality of this module ~~(10%)~~ (15%)
3. Describe why function verification is difficult and what is a testbench? (10%)
4. Describe what are the three major components in an FPGA? And how it can be programmable? (10%) switch logic gate connect
5. Suppose a machine has an asynchronous active-low reset,and the state transitions occur at positive clock edges. Let S2 be the reset state. The state transition graph of this FSM is shown as below, please complete the remaining Verilog code below to model the machine. (20%)



```
module FSM (clock, reset, in_bit, out_bit);
input clock, reset, in_bit;
output out_bit;
reg [1:0] state, next_state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10;
assign out_bit = ~r reset (1)    ? 1 : 0 ;
always @ (    (2)    or    (3)    ) negdge reset
begin      ~~posed~~ posedge clk
if(reset == 0) state = S2 (4)     ;
else state = next state (5) ;
end

              state
always @ (    (6)    or    (7)    )
case(state)
S0: begin
if(in_bit == 1) next_state =    (8) S1 ;
else next_state = S0;
end
S1: begin
if(in_bit == 1) next_state = S2;
else next_state =    (9) S0
end
S2: begin
if(in_bit == 1) next_state =    (10) S? ;
else next_state = S0;
end
endcase
endmodule
```

- 8

0

+7

111
0000
000

6. In Lab6, we have designed a 4-bit ALU with two function-select inputs S1 and S0 that performs logic and arithmetic operations on a pair of operands A and B. For arithmetic operations, A and B are interpreted as two 4-bit numbers in signed 2's complement. The ALU has an overflow status bit. Please describe the design for the overflow status bit in form of a Truth table. (15%)

7. Find out the sensitive list of the always block shown bellow: (15%)

```
always@(                )
begin
   if(sel)
      out=a+b;
   else
      out=a+c;

   case(op)
      1'b0:LED=1'b0;
      1'b1:LED=k;
      default: LED=1'b0;
```

+7 overflow

+7

-7 overflow

-8

8

1000

1001

1001

111
000
1001

110

111