**Midterm Example Questions**

13/04/2024

1. Simplify the following Boolean expression to a minimum expression (by using Boolean Algera)

F= A'B(D' + C'D) + B(A + A'CD)

2. Using Karnaugh map to find the simplest sum-of-products form of the following functions F then implement this circuit in Verilog

F(A, B, C, D) =Σ (1,5,9, 10,11,13,14)

3.Write the algebraic function of F shown in given the truth table, then use algebraic manipulation to simplify the equation obtained. Write the Verilog code to implement the simplified Boolean expression function of F using Verilog Gate level modeling

A number of numbers on a white background

Description automatically generated with medium confidence

4. Using Karnaugh map to find the simplest sum-of-products form of the following functions F then implement the simplified circuit in Verilog

F(A, B, C, D) =Σ (1,5,9,10,11,13,14)

5. Given the Digital Logic Circuit:

A diagram of a circuit

Description automatically generated

a) Draw the truth table step-by-Step the for this circuit

b) Develop the Verilog module to implement this circuit using structural model

c) Write the testbench for verification this cmodule

6. Given the Digital Logic Circuit:

A diagram of a circuit

Description automatically generated

a) Draw the truth table step-by-Step the for this circuit

b) Develop the Verilog module to implement this circuit using structural model

c) Write the testbench for verification this cmodule

6. Given digital circuit diagram of full adder and 4-bit adder/subtractor circuits

A diagram of a circuit

Description automatically generated

Full adder circuit

A diagram of a diagram

Description automatically generated

4-bit adder/subtractor circuit

a) Write the Verilog Module for full adder using structural model.

b) Write the Verilog Module for 4-bit adder/subtractor module using structural model 4 full adder and exclusive OR gates.

7. Write the Verilog code to implement the 4 bit ALU module with the given truth table

A diagram of a rectangle

Description automatically generated

8. Draw the FSM state diagram for the Sequence Recognizer to Identify the sequence 0001, regardless of where it occurs in a longer sequence. Write the Verilog Module to Implement this circuit

9. Draw a schematic of the circuit defined in each Verilog module below (5 Points)

module Combo (a,b,c,d,e, z);

input a, b, c, d, e ;

output z ;

assign z = ( (a & b) | (c ^ d ) & ~ e) ;

endmodule

12. Write the truth table and VHDL explicit structural description of the following logic

A picture containing diagram, sketch, line, plan

Description automatically generated

10. Write the truth table for the following Verilog code, write testbench to test this module

module Example(en,a,y);

input en;

input [1:0]a;

output [3:0]y;

reg [7:0]y;

always @(en,a)

begin

if (en==0)

y=0;

else

case(a)

2'b00:y=4'b0001;

2'b01:y=4'b0010;

2'b10:y=4'b0100;

2'b11:y=4'b1000;

default: begin end

endcase

end

endmodule

11. Write the truth table for the following Verilog code:

module Foo (clk, load, si, d, so);

input clk, si, load;

input [7:0] d;

output so;

reg [7:0] tmp;

always @(posedge clk or posedge load)

begin

if (load)

tmp <= d;

else

tmp <= {tmp[6:0], si};

end

assign so = tmp[7];

endmodule

12. Convert the following Verilog code into VHDL code : (5 Points)

module Example ( a,b,c,d,s0,s1,y)

input a, b, c, d ;

input s0, s1;

output y;

assign out = s1 ? (s0 ? d : c) : (s0 ? b : a);

endmodule

16. . Write The truth table of following VHDL module

module example(clk, sIn, sOut);

input sIn, clk;

output reg sOut;

reg [7:0]state;

always @(posedge clk)

begin

sOut <= state[7];

state <= {state[6:0], sIn};

end

endmodule

13. Write The truth table of following VHDL module is: (5 Points)

library ieee;

use ieee.std\_logic\_1164.all;

entity Foo is

port( I3: in std\_logic\_vector(2 downto 0);

I2: in std\_logic\_vector(2 downto 0);

I1: in std\_logic\_vector(2 downto 0);

I0: in std\_logic\_vector(2 downto 0);

S: in std\_logic\_vector(1 downto 0);

O: out std\_logic\_vector(2 downto 0)

);

end Foo;

architecture behv of Foo is

begin

process(I3,I2,I1,I0,S)

begin

case S is

when "00" => O <= I0;

when "01" => O <= I1;

when "10" => O <= I2;

when "11" => O <= I3;

when others =>O <= "ZZZ";

end case;

end process;

end behv;

14. Convert the following VHDL code into Verilog Code;

library IEEE; use IEEE.STD\_LOGIC\_1164.all;

entity gates is

port(a, b: in STD\_LOGIC\_VECTOR(3 downto 0);

y1, y2, y3, y4, y5: out STD\_LOGIC\_VECTOR(3 downto 0));

end;

architecture synth of gates is

begin

y1 <= a and b;

y2 <= a or b;

y3 <= a xor b;

y4 <= a nand b;

y5 <= a nor b;

end;

15. Given the Ex8 module, and testbench, what is the output result of the testbench

module Ex8 (input1, input2, out);

input [31:0] input1, input2;

output [31:0] out;

reg [31:0] out;

always@( input1 or input2)

begin

out <= input1 \* input2;

end

endmodule

module test;

reg [31:0] input1, input2;

wire [31:0] out;

Ex8 mx\_DUT (input1, input2, out);

initial begin

$monitor ("in1 = %d, in2 = %d, out = %d", input1, input2, out);

input1 = 32'd6; input2 = 32'd3;

#5 input1 = 32'd8; input2 = 32'd12;

#5 input1 = 32'd5; input2 = 32'd9;

#10 $finish;

end

endmodule

16. . Write The truth table of following VHDL module:

library IEEE; use IEEE.STD\_LOGIC\_1164.ALL;

entity Ex\_11 is

port(clk, reset: in STD\_LOGIC;

load, sin: in STD\_LOGIC;

d: in STD\_LOGIC\_VECTOR(7 downto 0);

q: out STD\_LOGIC\_VECTOR(7 downto 0);

sout: out STD\_LOGIC);

end;

architecture synth of Ex\_11 is

begin

process(clk, reset) begin

if reset = '1' then q <= (OTHERS => '0');

elsif rising\_edge(clk) then

if load then q <= d;

else q <= q(6 downto 0) & sin;

end if;

end if;

end process;

sout <= q(7);

end;

17. . Write The truth table of following Verilog module:

module Foo\_ex (clk, clr, up\_down, q);

input clk, clr, up\_down;

output [3:0] q;

reg [3:0] tmp;

always @(posedge clk or posedge clr)

begin

if (clr)

tmp <= 4’b0000;

else if (up\_down)

tmp <= tmp + 1’b1;

else

tmp <= tmp - 1’b1;

end

18. Write the Verilog Module to implement following FSM with given State Diagram

Shape, arrow

Description automatically generated

19. Write the Verilog Module to implement following FSM with given State Diagram

A diagram of a mathematical equation

Description automatically generated

20. Draw the FSM state diagram for the following Verilog FSM Module.

module example\_code( input x,clk,reset,

output reg z);

parameter S0 = 0 , S1 = 1 , S2 = 2 , S3 = 3 ;

reg [1:0] PS,NS ;

always@(posedge clk or posedge reset)

begin

if(reset)

PS <= S0;

else

PS <= NS ;

end

always@(PS or x)

begin

case(PS)

S0 : begin

z = 0 ;

NS = x ? S1 : S0 ;

end

S1 : begin

z = 0 ;

NS = x ? S1 : S2 ;

end

S2 : begin

z = 0 ;

NS = x ? S3 : S0 ;

end

S3 : begin

z = x ? 1 : 0 ;

NS = x ? S1 : S2 ;

end

endcase

end

endmodule

21. Draw the State diagram for the following Verilog FSM code

module fsm (clk, reset, x1, outp);

input clk, reset, x1;

output outp;

reg outp;

reg [1:0] state;

parameter s1 = 2’b00; parameter s2 = 2’b01;

parameter s3 = 2’b10; parameter s4 = 2’b11;

always @(posedge clk or posedge reset)

begin

if (reset)

state <= s1;

else begin

case (state)

s1: if (x1 == 1’b1)

state <= s2;

else

state <= s3;

s2: state <= s4;

s3: state <= s4;

s4: state <= s1;

endcase

end

end

always @(state) begin

case (state)

s1: outp = 1’b1;

s2: outp = 1’b1;

s3: outp = 1’b0;

s4: outp = 1’b0;

endcase

end

endmodule

22. Draw the FSM state diagram for the Sequence Recognizer to Identify the sequence 1000, regardless of where it occurs in a longer sequence. Write the Verilog Module to Implement this circuit.