DIGITAL DESIGN

LAB9 COMBINATORIAL CIRCUIT VERILOG-SUMMARY（1）+ A2P2 REFERENCE CODE

2022 FALL TERM @ CSE . SUSETCH

LAB9

- Sequential vs Parallel in verilog
  - begin - end vd fork - join
  - The parallel in the design
- Verilog summary(1)
- A2-P2 reference code

WHAT'S THE SIMULATION RUNNING TIME

```
// part A
module decoder_mux_sim( );
reg sdne;
reg [1:0] sdx;
wire [3:0] sdy;
reg [15:0] smx;
reg [3:0] smsel;
wire smy;
d74139 u1(sdne,sdx,sdy);
mux16to1 u2(smx,smsel,smy);
```

```
endmodule  // part D
```

```
initial begin     // part B
    {sdne,sdx} = 3'b0;
    repeat(7) #10 {sdne,sdx} = {sdne,sdx} + 1;
       #10 $finish();
end
```

```
initial begin     // part C
    smsel=4'b0;
    smx = 16'h0001;
    repeat(15) begin
        #10 smsel = smsel + 1;     smx = smx<<1;
    end
    #10 $finish();
end
```

Q1. What's the simulation time while running the testbench on the left hand?

Q2. If move the partC ahead of partB in the testbench, will the simlation time change?

WHAT'S THE SIMULATION RUNNING TIME

```
// part A
module decoder_mux_sim( );
reg sdne;
reg [1:0] sdx;
wire [3:0] sdy;
reg [15:0] smx;
reg [3:0] smsel;
wire smy;
d74139 u1(sdne,sdx,sdy);
mux16to1 u2(smx,smsel,smy);
```

```
endmodule  // part D
```

```
initial begin     // part B
    {sdne,sdx} = 3'b0;
    repeat(7) #10 {sdne,sdx} = {sdne,sdx} + 1;
       #10 $finish();
end
```

```
initial begin     // part C
    smsel=4'b0;
    smx = 16'h0001;
    repeat(15) begin
        #10 smsel = smsel + 1;     smx = smx<<1;
    end
    #10 $finish();
end
```

Q1. What's the simulation time while running the testbench on the left hand?

A1. 80

Q2. If move the partC ahead of partB in the testbench, will the simlation time change?

A2. NO, won't change

SEQUENTIAL VS PARALLEL

```
module block2();
    reg [1:0]x,y;
    initial
    begin
        #10 x=2'd0;
        #10 x=2'd1;
        #10 x=2'd2;
        #10 x=2'd3;
    end

    initial
    fork
        #10 y=2'd0;
        #20 y=2'd1;
        #30 y=2'd2;
        #40 y=2'd3;
    join

    initial
        #50 $finish(1);
endmodule
```

Answer the following question according to the code on the left hand

- Is “block2” a design module ?
- There are three “inital” blocks in module “block2”, does the inital on the top run firstly, the initial on the buttom run secondly?
- While running the module “block2”, what’s its simulation time?
- Guess the difference between “begin-end” and “fork-join”

SEQUENTIAL BLOCK VS PARALLEL BLOCK

```
module block2();
    reg [1:0]x,y;
    initial
    begin
        #10 x=2'd0;
        #10 x=2'd1;
        #10 x=2'd2;
        #10 x=2'd3;
    end

    initial
    fork
        #10 y=2'd0;
        #20 y=2'd1;
        #30 y=2'd2;
        #40 y=2'd3;
    join

    initial
        #50 $finish(1);
endmodule
```

- In one module **all the block executes at the same time**(**time 0**)
- **Sequential block**(**begin … end**):
  - **synthesizable(could be used in the circuit design)**
  - all the statements in one sequential block executes with the order of writing.
- **Parallel block**(**fork … join**):
  - **Not synthesizable(CAN NOT be used in the circuit design)**
  - all the statements in one parallel block executes at same time

SEQUENTIAL VS PARALLEL

```
module block2();
    reg [1:0]x,y;
    initial
    begin
        #10 x=2'd0;
        #10 x=2'd1;
        #10 x=2'd2;
        #10 x=2'd3;
    end

    initial
    fork
        #10 y=2'd0;
        #20 y=2'd1;
        #30 y=2'd2;
        #40 y=2'd3;
    join

    initial
        #50 $finish(1);
endmodule
```

Answer the following question according to the code on the left hand

- Is “block2” a design module ?
  - **NO, it’s NOT a desing module**
- There are three “inital” blocks in module “block2”, does the inital on the top run firstly, the initial on the buttom run secondly?
  - **NO, they runs at the same time**
- While running the module “block2”, what’s its simulation time? **50**

sakai-quiz - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

1. b= {a,0} , b2bit,a 1bit， a的值是1'b1,请问做完赋值后 b的数值是?

A. 2'b10 B.2'b01 C.2'h00 D.2'h11 → 2'b01

2. 一个4-1的多路选择器，module mux4to1(input [3:0] D, input [1:0]s,output f)

y=s[1]'s[0]'.D[0] + s[1]'s[0].D[1] + s[1]s[0]'.D[2] + s[1]s[0].D[3]

如果要用mux4to1实现一个组合逻辑电路：f(a,b,c)=ab'c+a'b'c'+ab+a'b，有以下几种使用方法

1) mux4to1({1,c,1,c'},{a,b},y);

2) mux4to1({1'b1,c,1'b1,c'},{a,b},y);

3) mux4to1({1,1,c,c'},{b,a},y);

4) mux4to1({1'b1,1'b1,c,c'},{b,a},y);

第 1 行，第 11 列 270% Windows (CRLF) UTF-8

a在高位

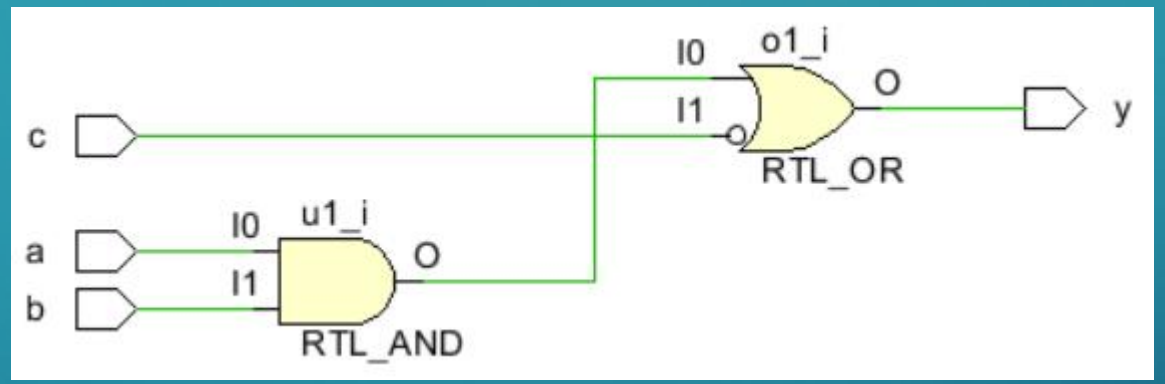
此处应加上一个位宽 (待补)

VERILOG-SUMMARY(1)

- Q1. In verilog, the constant “0” equal to “1’b0”, is it true or false?
- Q2. Could we using “123port” as the name a module, a variable or a port? Why?
- Q3. While define a variable to bind with the output port, what’s the data type of the variable?
- Q4. To define two input ports: **a** is **2**bits, **b** is **1**bit, which option(s) is(are) correct?
  - A. input a,b;
  - B. input reg a,b;
  - C. input [2:0] a,b;
  - D. input [1:0] a,b;
  - E. input [1:0]a; input b;
  - F. input a,[2:0] b;

VERILOG-SUMMARY(2)

- Q5. There are two modules at the bottom of the page, which one(s) is(are) same with the circuit bellow?

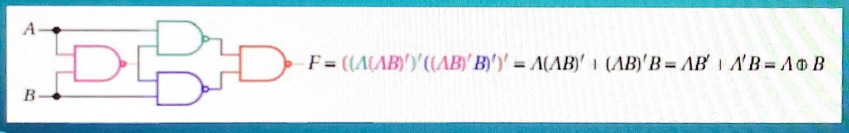



```
module demo1(input a,b,c,output y);
wire w1,nc;

not n1(nc,c);
and u1(w1,a,b);
or o1(y,w1,nc);
endmodule
```

```
module demo2(input a,b,c,output y);
wire w1,nc;

and u1(w1,a,b);
not n1(nc,c);
or o1(y,w1,nc);
endmodule
```

TASK1-USING XOR GATES- USING NAND GATES

```
module alt1_xor(input [3:0] x, output y );
    xor uxor(y, x[0], x[1], x[2], x[3]);
endmodule
```

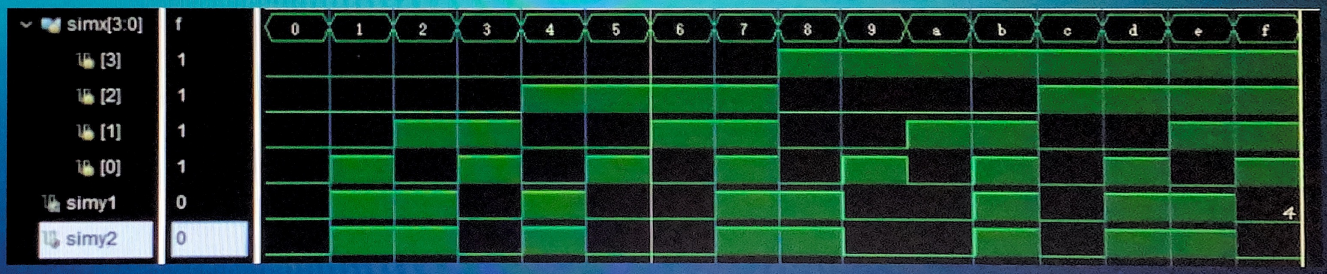



```
module alt1_nand(input [3:0] x, output y );
        wire x01;
        nand unand01(x01, x[0], x[1]);
        wire x010, x011;
        nand unand20(x010, x[0], x01);
        nand unand21(x011, x[1], x01);
        wire xorx0x1;
        nand unand301(xorx0x1, x010, x011);

        wire x23;
        nand unand23(x23, x[2], x[3]);
        wire x232, x233;
        nand unand222(x232, x[2], x23);
        nand unand223(x233, x[3], x23);
        wire xorx2x3;
        nand unand323(xorx2x3, x232, x233);

        wire x01x23;
        nand uand0123(x01x23, xorx0x1, xorx2x3);
        wire x0123x01, x0123x23;
        nand uand2010123(x0123x01, xorx0x1, x01x23);
        nand uand2230123(x0123x23, xorx2x3, x01x23);
        nand unand30123(y, x0123x01, x0123x23);
endmodule
```

TASK1--TESTBENCH-SIMULATION RESULT

```
module a2t1_sim( );
reg [3:0] simx;
wire simy1, simy2;
//module alt1_nand(input [3:0] x, output y );
//module alt1_xor(input [3:0] x, output y );
alt1_xor u1(simx, simy1);
alt1_nand u2(simx, simy2);

initial begin
    simx = 4'b0;
    repeat(15) #10 simx = simx+1;
    #10 $finish();
end
endmodule
```

A parity generator circuit has one input “x” which is 4 bit-width, and one output “y” which is 1 bit-width. The output is 1 if the number of “1” in “x” is odd, else the output is 0.



TASK2(SUM-OF-MINTERMS , PRODUCT-OF-MAXTEMS)

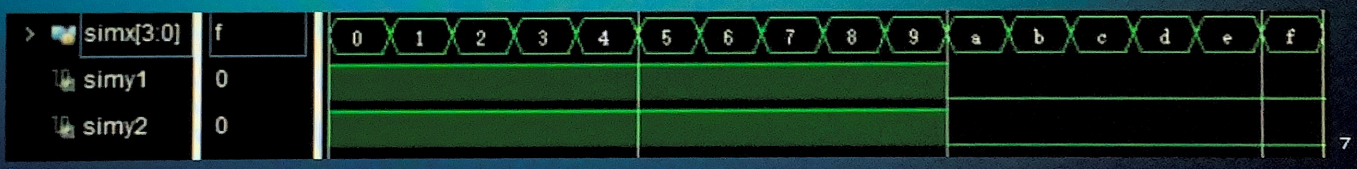
```
module a2t2_somin(input [3:0]x, output y);
//y = m0+m1+m2+m3+m4+m5+m6+m7+m8+m9.
wire m0, m1, m2, m3, m4, m5, m6, m7, m8, m9;
wire nx3, nx2, nx1, nx0;
not un3(nx3, x[3]); not un2(nx2, x[2]); not un1(nx1, x[1]); not un0(nx0, x[0]);
and um0 (m0, nx3, nx2, nx1, nx0);  //0000
and um1 (m1, nx3, nx2, nx1, x[0]); //0001
and um2 (m2, nx3, nx2, x[1], nx0); //0010
and um3 (m3, nx3, nx2, x[1], x[0]); //0011
and um4 (m4, nx3, x[2], nx1, nx0); //0100
and um5 (m5, nx3, x[2], nx1, x[0]);//0101
and um6 (m6, nx3, x[2], x[1], nx0); //0110
and um7 (m7, nx3, x[2], x[1], x[0]);//0111
and um8 (m8, x[3], nx2, nx1, nx0);  //1000
and um9 (m9, x[3], nx2, nx1, x[0]); //1001
or uor1(y, m0, m1, m2, m3, m4, m5, m6, m7, m8, m9);
endmodule
```

```
module a2t2_pomax(input [3:0]x, output y);
//y = m0+m1+m2+m3+m4+m5+m6+m7+m8+m9.
//y = M10.M11.M12.M13.M14.M15
wire nx3, nx2, nx1, nx0;
not un3(nx3, x[3]); not un2(nx2, x[2]); not un1(nx1, x[1]); not un0(nx0, x[0]);
wire M10, M11, M12, M13, M14, M15;
or uM10(M10, nx3, x[2], nx1, x[0]);   //1010 ->0101
or uM11(M11, nx3, x[2], nx1, nx0);    //1011 ->0100
or uM12(M12, nx3, nx2, x[1], x[0]);   //1100 ->0011
or uM13(M13, nx3, nx2, x[1], nx0);    //1101 ->0010
or uM14(M14, nx3, nx2, nx1, x[0]);    //1110 ->0001
or uM15(M15, nx3, nx2, nx1, nx0);     //1111 ->0000
and uand1(y, M10, M11, M12, M13, M14, M15);
endmodule
```

TASK2--TESTBENCH-SIMULATION RESULT

```
module a2t2_sim( );
reg [3:0] simx;
wire simy1, simy2;
//module a2t2_somin(input [3:0]x, output y);
//module a2t2_pomax(input [3:0]x, output y);
a2t2_somin ua2t2_somin(simx, simy1);
a2t2_pomax ua2t2_pomax(simx, simy2);
initial begin
    simx = 4'b0;
    repeat(15) #10 simx = simx+1;
    #10 $finish();
end
endmodule
```

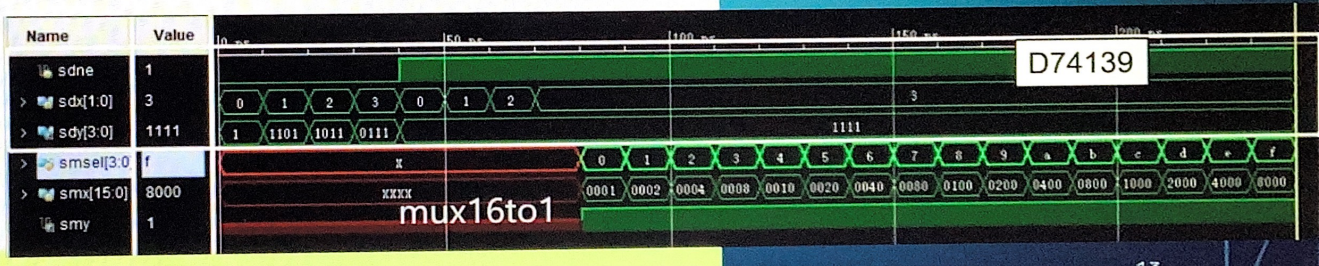
A combinational circuit has one input “x” which is 4 bit-width , and one output “y” which is 1 bit-width. The output is 1 if the input is a valid BCD coded decimal digit. Otherwise, the output is 0.



TASK4-TESTBENCH-TO-VERIFY-DECODER-MUX

```
module decoder_mux_sim( );
reg sdne;
reg [1:0] sdx;
wire [3:0] sdy;
reg [15:0] smx;
reg [3:0] smsel;
wire smy;
d74139 ud74139(sdne, sdx, sdy);        //module d74139(input ne,input [1:0]x,  output reg [3:0] y);
mux16to1 umux16to1(smx, smsel, smy);   //module mux16to1(input [15:0] x,input[3:0]sel, output reg y );
initial begin
    {sdne, sdx} = 3'b0;
    repeat(7) #10 {sdne, sdx} = {sdne, sdx} + 1;
    #10
    smsel=4'b0;
    smx = 16'h0001;
    repeat(15) begin
        #10
        smsel = smsel + 1;
        smx = smx<<1;
    end
    #10 $finish();
end
endmodule
```

While using more than one $finish in the testbench, only one of them would work.



VERILOG-SUMMARY(3)

- Q6. Does the following pieces of verilog code relate to the same circuit?

```
module demo1(input a,b,c,output [5:0] y);
      assign y = {1'b0,a,1'b0,b,1'b0,c};
endmodule
```

```
module demo2(input a,b,c,output [5:0] y);
      assign y = 6'b0;
      assign y[0] = c;
      assign y[2] = b;
      assign y[4] = a;
endmodule
```

multiple driver

```
module demo3(input a,b,c,output [5:0] y);
      assign y = 6'b0;
      assign y[4] = a;
      assign y[2] = b;
      assign y[0] = c;
endmodule
```

```
module demo4(input a,b,c,output [5:0] y);
      assign y = {0,a,0,b,0,c};
endmodule
```

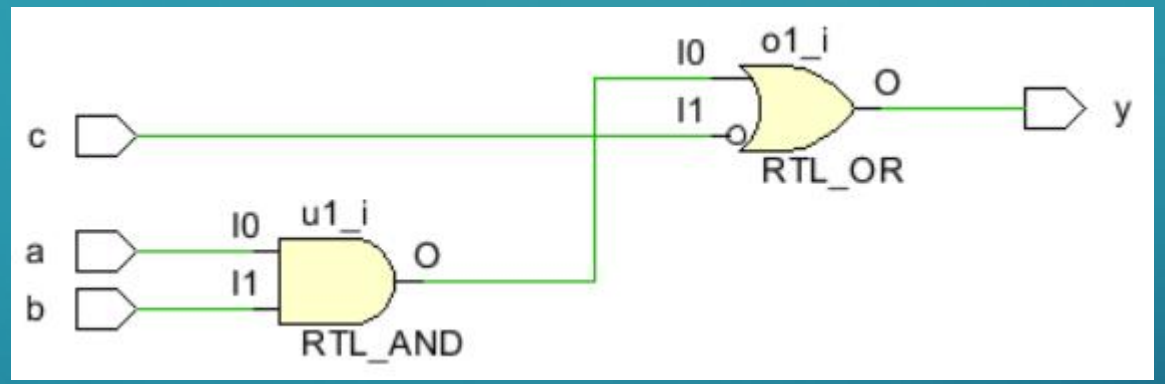
和其中一部分

VERILOG-SUMMARY(1)

- Q1. In verilog, the constant “0” equal to “1’b0”, is it true or false?
  - False
- Q2. Could we using “123port” as the name a module, a variable or a port? Why?
  - No, the name of module, variable and port CAN NOT start with number.
- Q3. While define a variable to bind with the output port, what’s the data type of the variable?
  - wire
- Q4. To define two input ports: **a** is **2**bits, **b** is **1**bit, which option(s) is(are) correct?
  - A. input a,b;
  - B. input reg a,b;
  - C. input [2:0] a,b;
  - D. input [1:0] a,b;
  - E. input [1:0]a; input b;
  - F. input a,[2:0] b;

VERILOG-SUMMARY(2)

- Q5. There are two modules at the bottom of the page, which one(s) is(are) same with the circuit bellow? demo1 and demo2 are same




```
module demo1(input a,b,c,output y);
wire w1,nc;

not n1(nc,c);
and u1(w1,a,b);
or o1(y,w1,nc);
endmodule
```

```
module demo2(input a,b,c,output y);
wire w1,nc;

and u1(w1,a,b);
not n1(nc,c);
or o1(y,w1,nc);
endmodule
```

VERILOG-SUMMARY(3)

- Q6. Does the following pieces of verilog code relate to the same circuit?
  - No, only demo2 and demo3 are same, but they are illegal.

```
module demo1(input a,b,c,output [5:0] y);
      assign y = {1'b0,a,1'b0,b,1'b0,c};
endmodule
```

```
module demo2(input a,b,c,output [5:0] y);
      assign y = 6'b0;
      assign y[0] = c;
      assign y[2] = b;
      assign y[4] = a;
endmodule
```

```
module demo3(input a,b,c,output [5:0] y);
      assign y = 6'b0;
      assign y[4] = a;
      assign y[2] = b;
      assign y[0] = c;
endmodule
```

```
module demo4(input a,b,c,output [5:0] y);
      assign y = {0,a,0,b,0,c};
endmodule
```