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 只能在仿真中使用
  - 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
```

80ns

执行过

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

160ns

不执行

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

同时执行 80ns时结束

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);
```

```
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
```

↑ 移位信号

```
endmodule  // part D
```

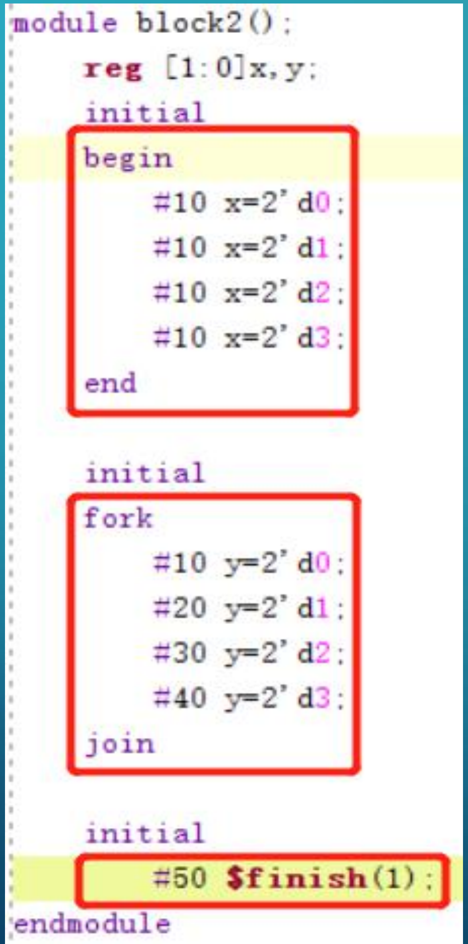
Q1. What’s the simulation time while running the testbench on the left hand?

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

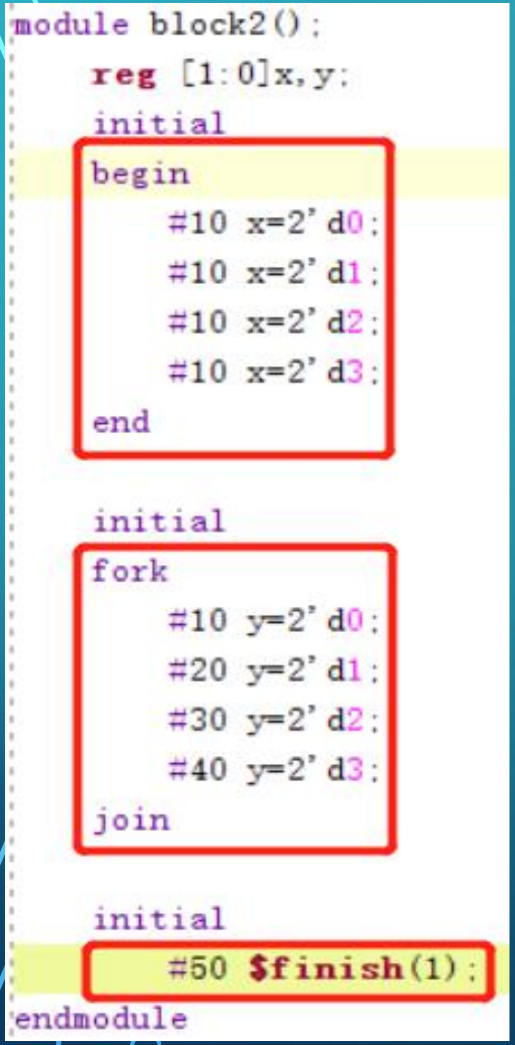


```
module block2();
    reg [1:0]x,y;
    initial
    begin
        #10 x=2'd0;
        #10 x=2'd1;
        #10 x=2'd2;
        #10 x=2'd3;
    end

    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



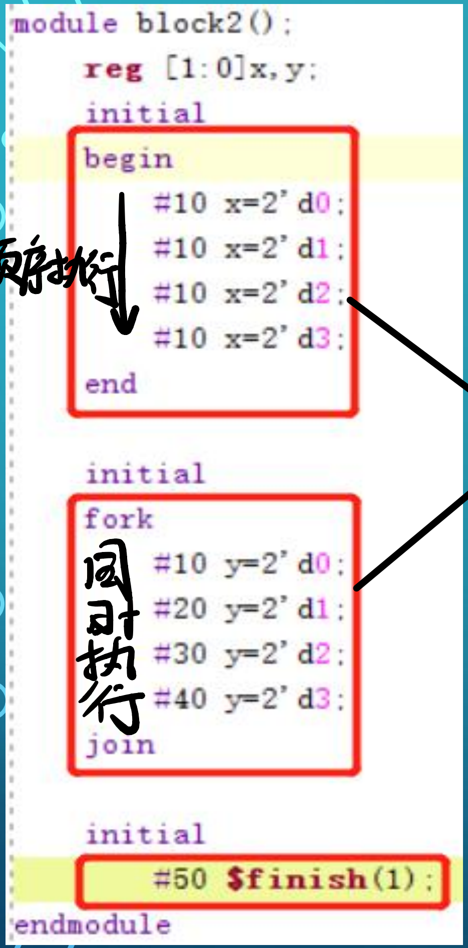
```
module block2();
    reg [1:0]x,y;
    initial
    begin
        #10 x=2'd0;
        #10 x=2'd1;
        #10 x=2'd2;
        #10 x=2'd3;
    end

    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

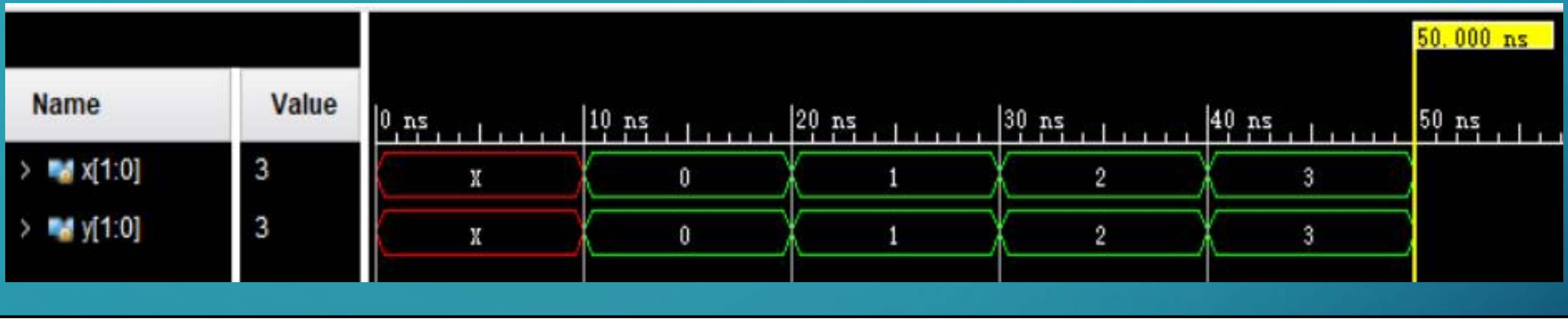
SEQUENTIAL VS PARALLEL



```
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
```

顺序执行

同时执行



并行

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

- Is “block2” a design module ?
  - **NO, it’s NOT a desing module** ：没有输入输出
- 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**

VERILOG-SUMMARY(1)

- Q1. In verilog, the constant “0” equal to “1’b0”, is it true or false?

  False

  b={a,0}, b 2bit, a 1bit

  b = 2'h00

  0未改位宽表示, 32位 即32'b0

- Q2. Could we using “123port” as the name a module, a variable or a port? Why?

  False 不能以数字开头

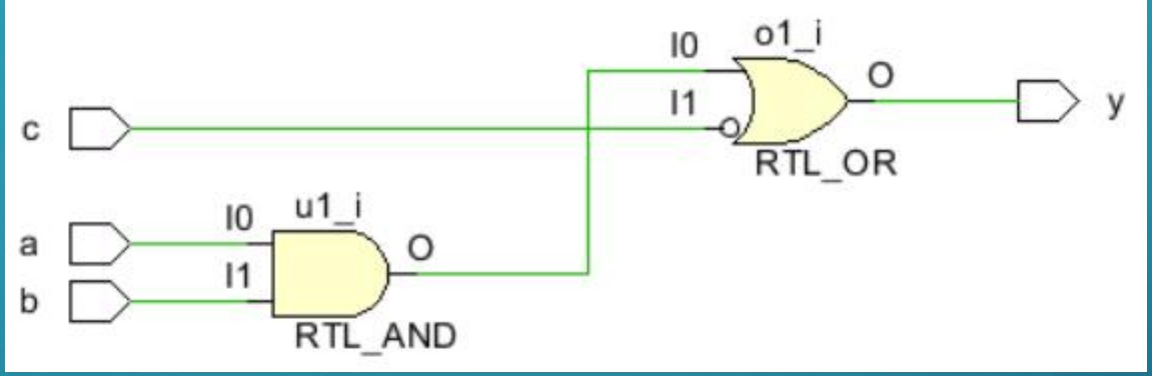
- Q3. While define a variable to bind with the output port, what’s the data type of the variable?

  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?



```
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
```

True

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
```

拼接

```
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
```

"mutiple driver ~"

重复对y赋值（因为同时驱

```
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
```

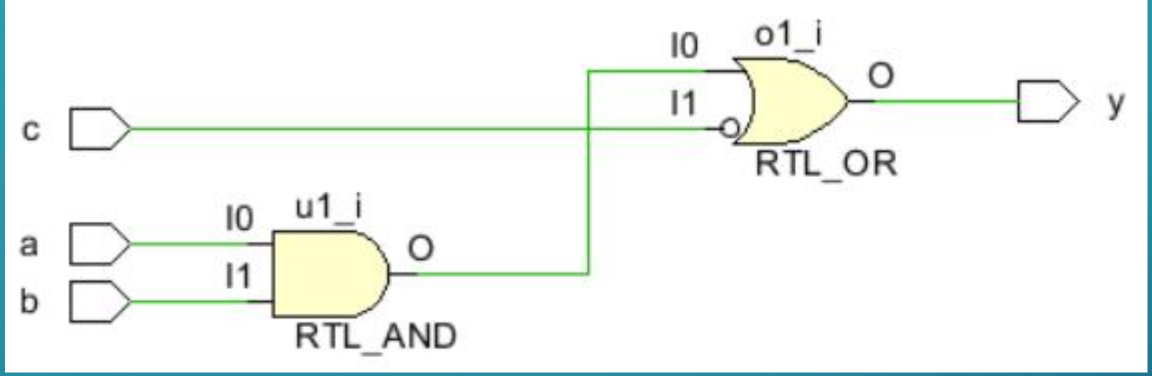
↑ 32位

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
```