《数字逻辑》

（第5次实验：在Logisim和FPGA开发板上实现异步时序逻辑电路）

厦门大学信息学院软件工程系 曾文华

2024年11月4日

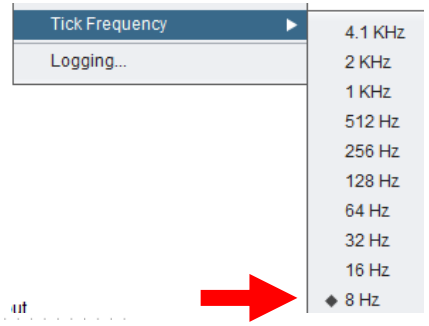
目录

第一部分：在Logisim上实现异步时序逻辑电路

请打开设计文件“第5次实验（发给学生）\第6章例题电路的实现.circ”

（一）在Logisim上实现教材第6章的例题

- **1、（验证实验）例题6.1的实现**
  - 在Logisim上实现例题6.1的**模4加1计数器**（图6.2），该电路的输入为x，状态为y_2、y_1，输出为Z。
  - **验证步骤**：
    - 设置时钟频率=8Hz。
    - 先按Ctrl+R，再按Ctrl+K，接下去**按x按钮5次**。
    - 观看是否产生类似图6.4的波形。



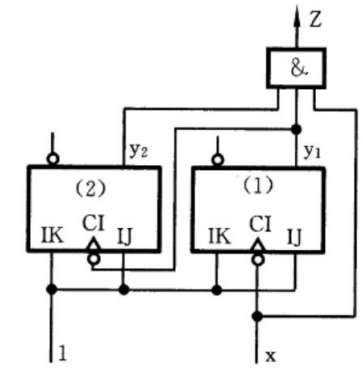


图 6.2　逻辑电路

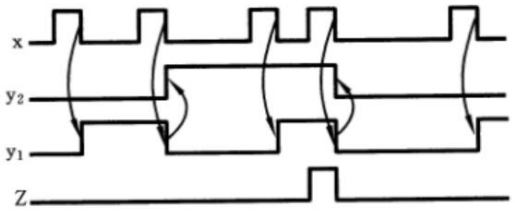
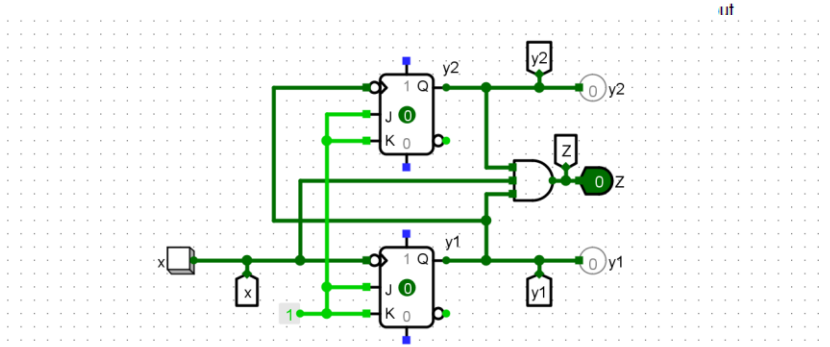
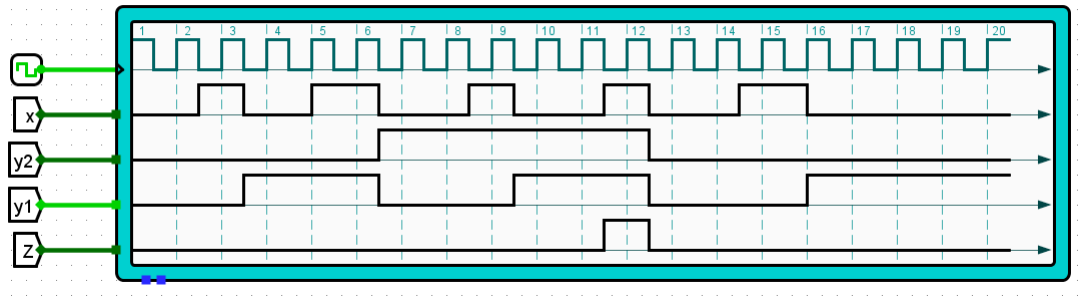


图 6.4　时间图





• 2、（验证实验）例题6.2的实现

- 在Logisim上实现例题6.2的“x_1 - x_2 - x_3”序列检测器（图6.5），该电路的输入为x_1、x_2、x_3，状态为y_2、y_1，输出为Z。

- **验证步骤：**
  - 先按Ctrl+R，再按Ctrl+K，接下去x_1、x_2、x_3按钮按照图6.7的式样进行按动（依次点击：x1、x2、x1、x3、x1、x2、x3、x1、x3、x2）。
  - 没有得到预期的波形！

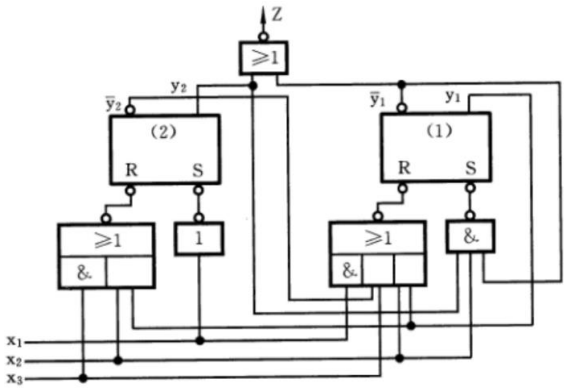


图 6.5　逻辑电路

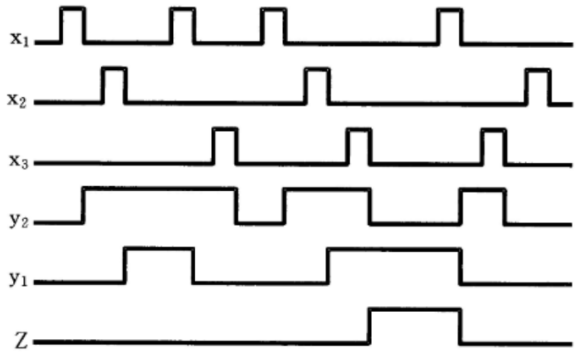
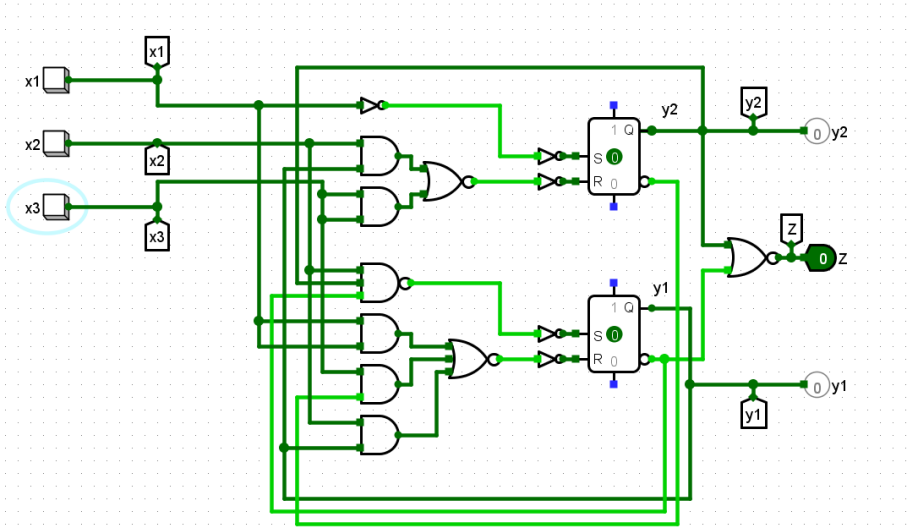
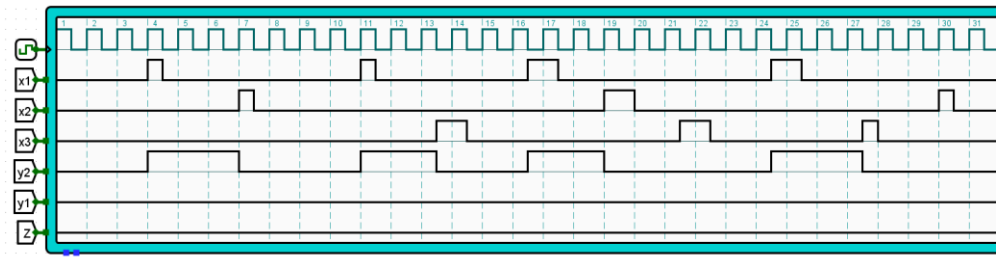
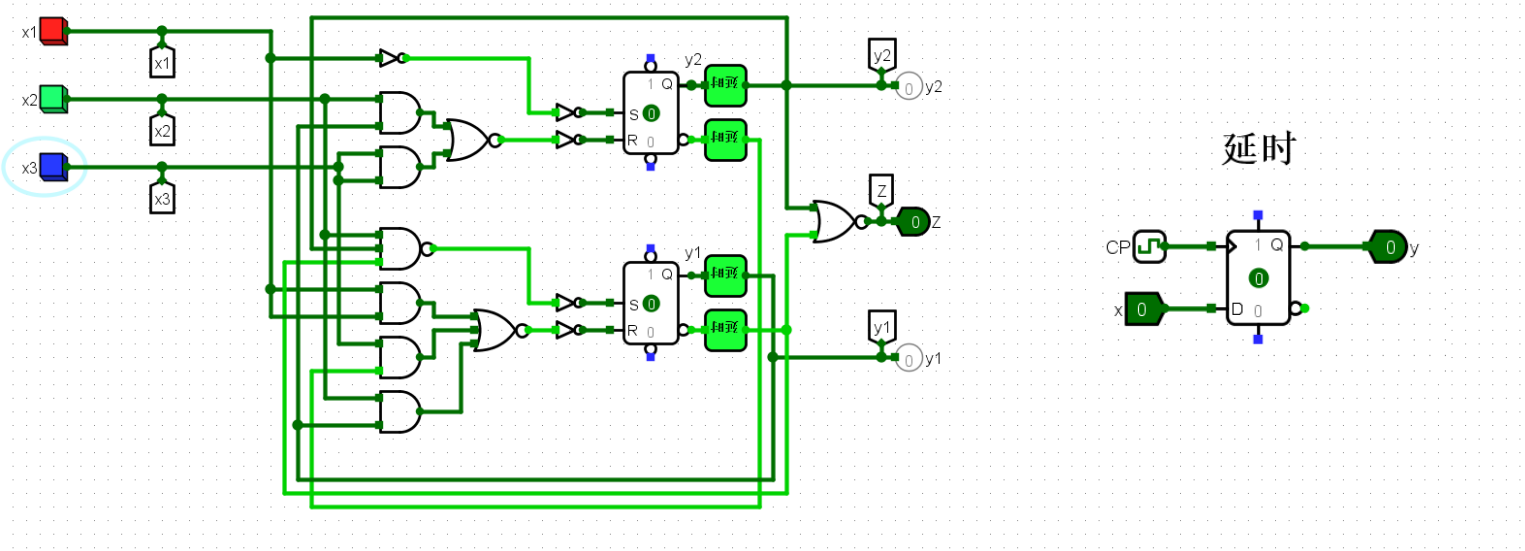


图 6.7　时间图





- 增加延迟环节后，有时会得到正确的结果，有时得不到（依次点击：x1、x2、x1、x3、x1、x2、x3、x1、x3、x2）。
- 有兴趣的同学可以分析原因！



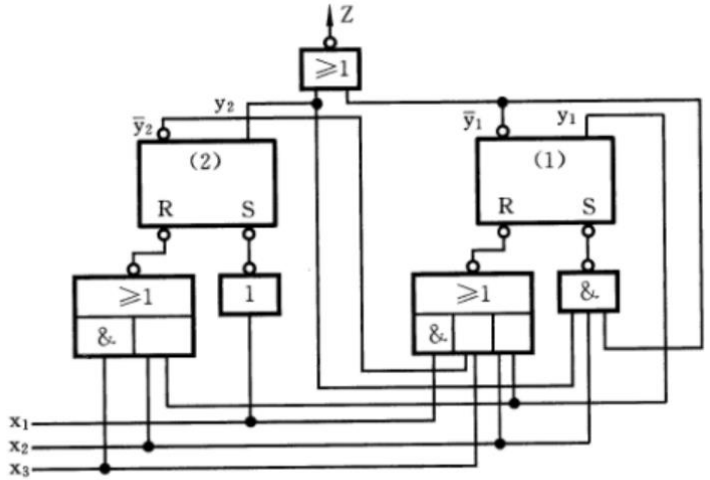
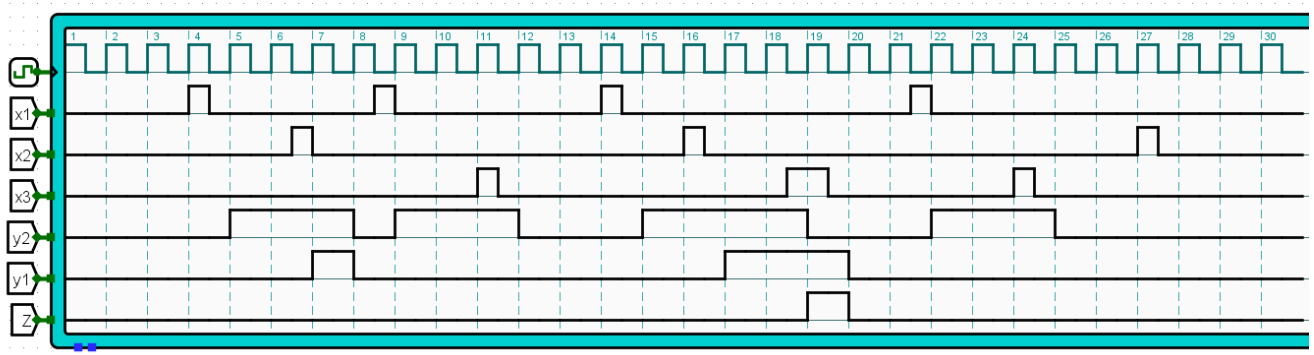


图 6.5　逻辑电路



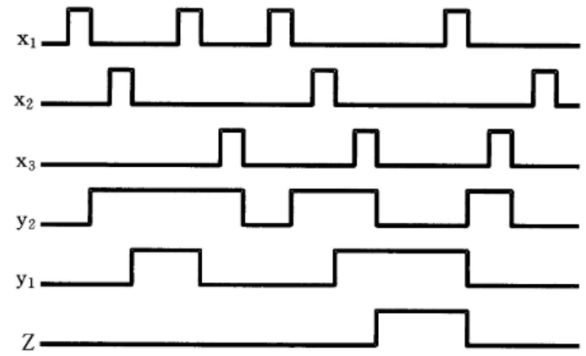


图 6.7　时间图

• 3、（验证实验）例题6.3的实现

- 在Logisim上实现例题6.3的“x_1 - x_2 - x_2”序列检测器（图6.11），该电路的输入为x_1、x_2，状态为y_2、y_1，输出为Z。

- **验证步骤**：
  - 先按Ctrl+R，再按Ctrl+K，接下去x_1、x_2按钮按照图6.8的式样进行按动（依次点击：x1、x2、x1、x2、x2、x2、x1）。
  - 可以得到正确的波形。

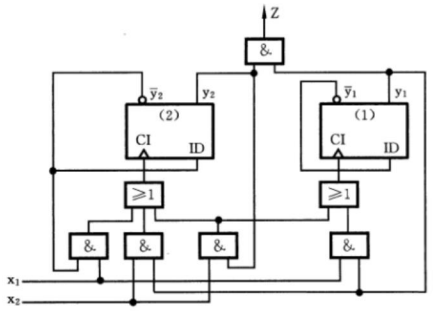


图 6.11　逻辑电路

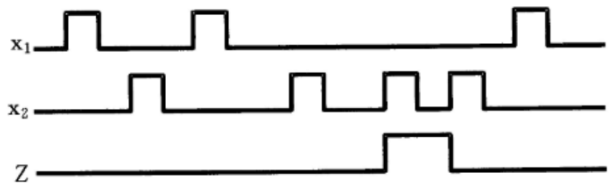
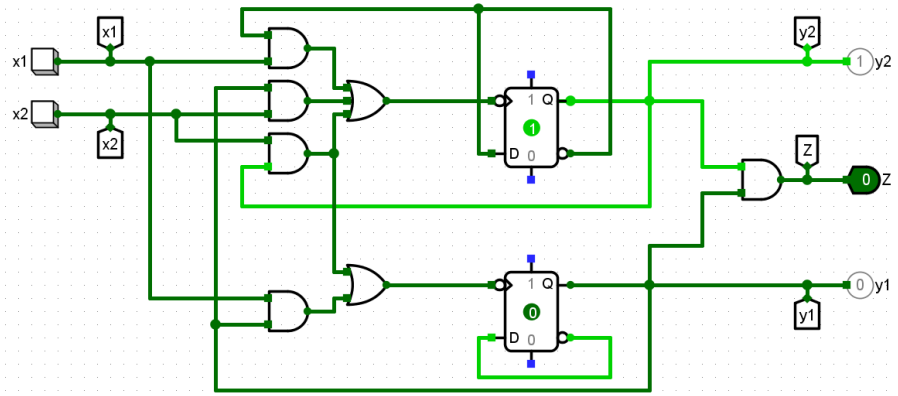
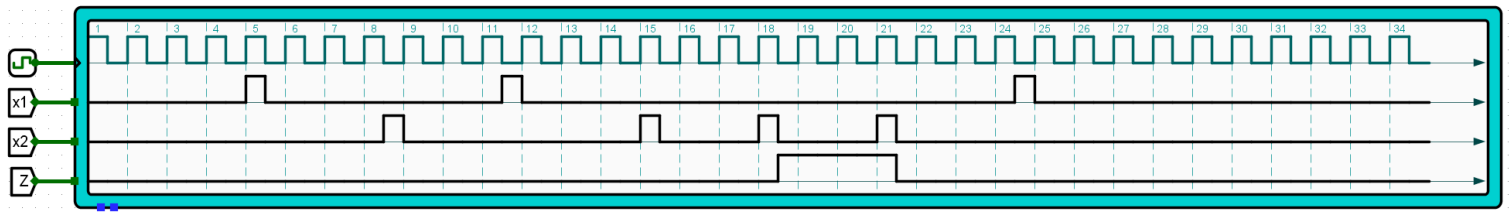


图 6.8　时间图





• 4、（验证实验）例题6.4的实现

- 在Logisim上实现例题6.4的异步模8加1计数器（图6.13），该电路的输入为x，状态为y_3、y_2、y_1，输出为Z。
- 验证步骤：
  - 先按Ctrl+R，再按Ctrl+K，接下去按x按钮8次。
  - 观看是否实现异步模8加1计数器的功能。

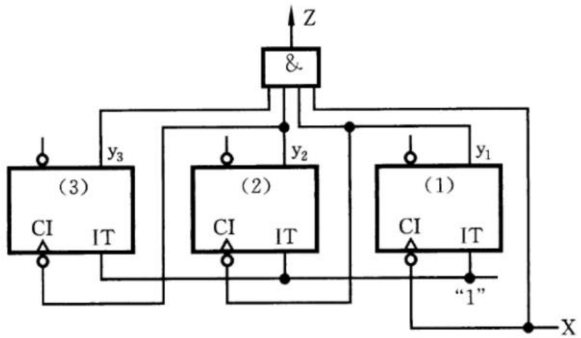
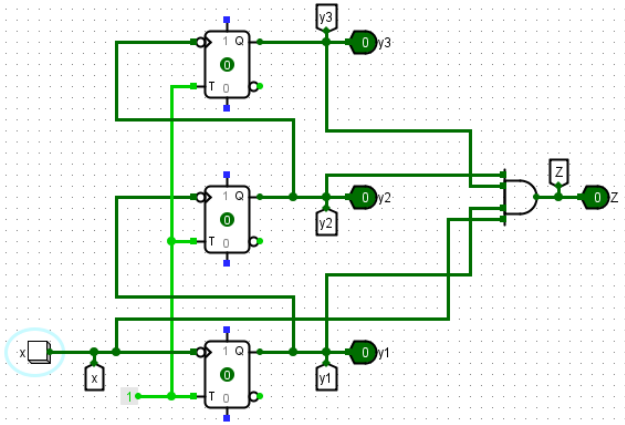


图 6.13　逻辑电路

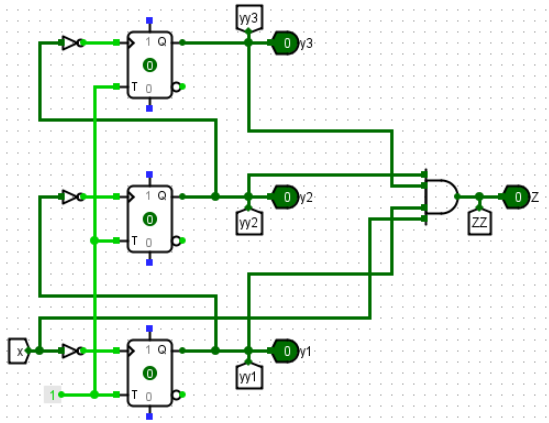
表 6.13　二进制状态表

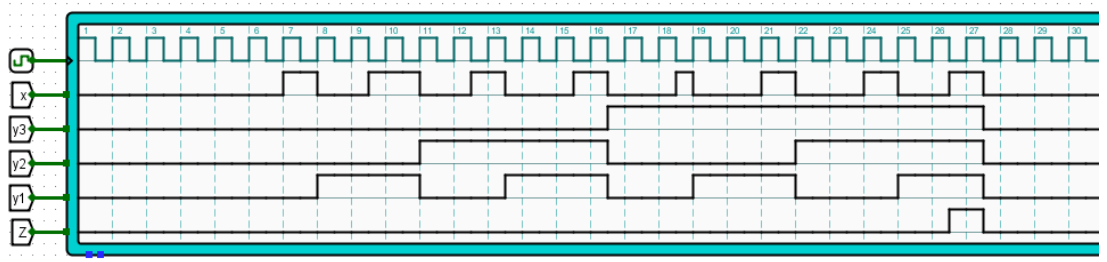
现　态	次态 $y_3^{n+1}\ y_2^{n+1}\ y_1^{n+1}$/输出 Z
$y_3\ y_2\ y_1$	x=1
0 0 0	0 0 1 / 0
0 0 1	0 1 0 / 0
0 1 0	0 1 1 / 0
0 1 1	1 0 0 / 0
1 0 0	1 0 1 / 0
1 0 1	1 1 0 / 0
1 1 0	1 1 1 / 0
1 1 1	0 0 0 / 1

下降沿触发的T触发器



上升沿触发的T触发器





• 5、（验证实验）例题6.5的实现

- 在Logisim上实现例题6.5的“00->10->11”序列检测器（图6.17），该电路的输入为（x_2、x_1），状态为（y_2、y_1），输出为Z。

- 验证步骤：
  - 先按Ctrl+R，再按Ctrl+K，接下去输入x_2x_1依次设置为：00、10、11、01、00、01、11、10（依次点击：x2、x1、x2、x1、x1、x2、x1）。
  - 观看是否产生类似图6.19的波形。

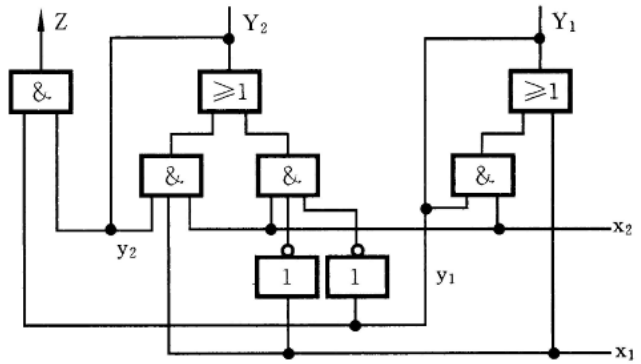
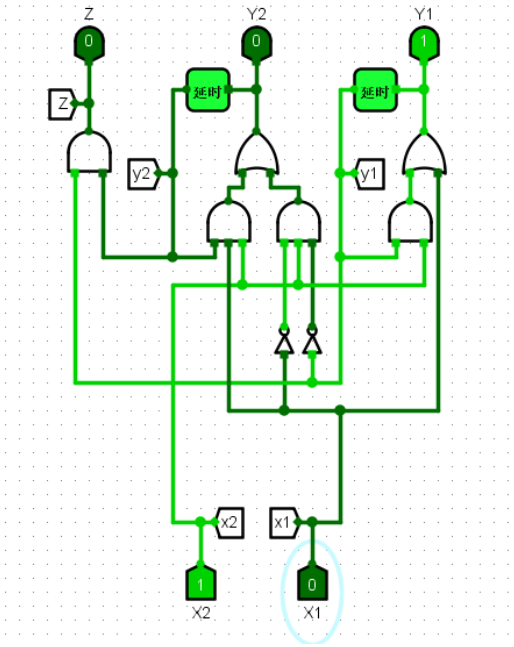
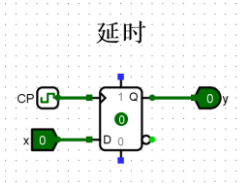


图 6.17　逻辑电路





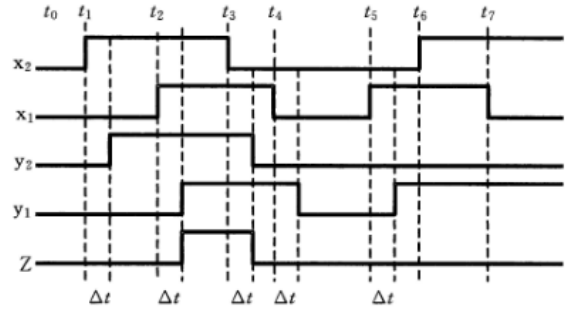
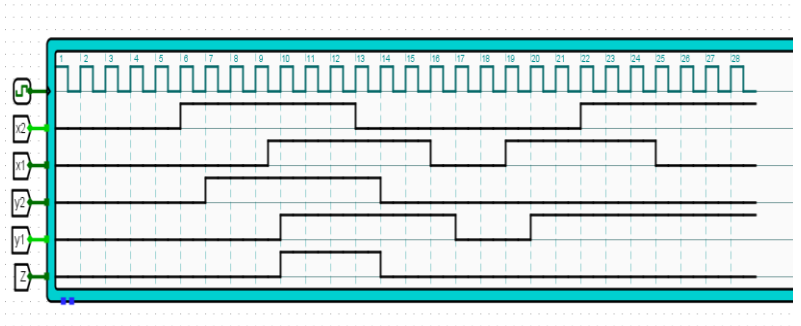


图 6.19　时间图



• 6、（验证实验）例题6.11的实现

- 在Logisim上实现例题6.11的单脉冲发生器（图6.35），该电路的输入为（x_2、x_1），状态为（y_2、y_1），输出为Z。

- **验证步骤：**
  - 先按Ctrl+R，再按Ctrl+K，接下去按x按钮1次。
  - 观看是否产生类似图6.31的波形。

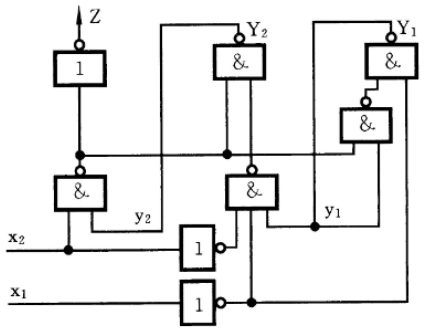
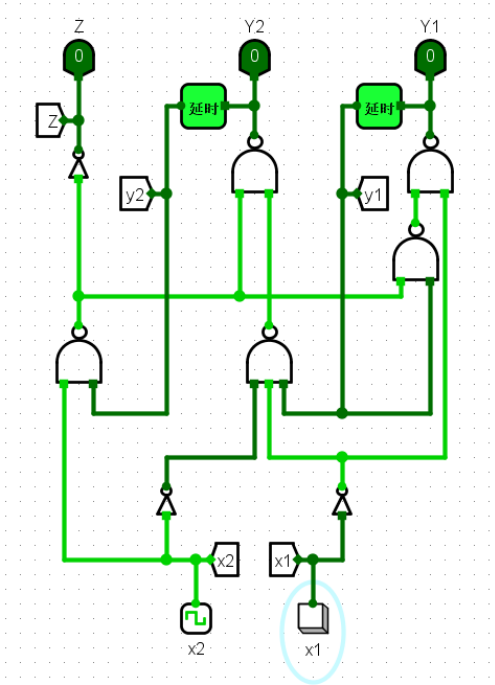
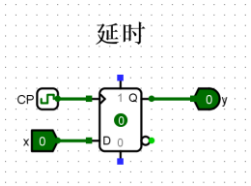


图 6.35　逻辑电路





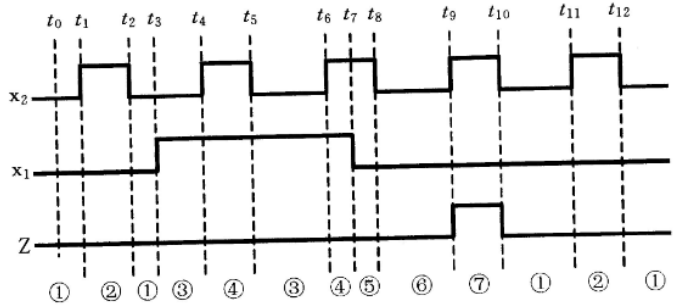
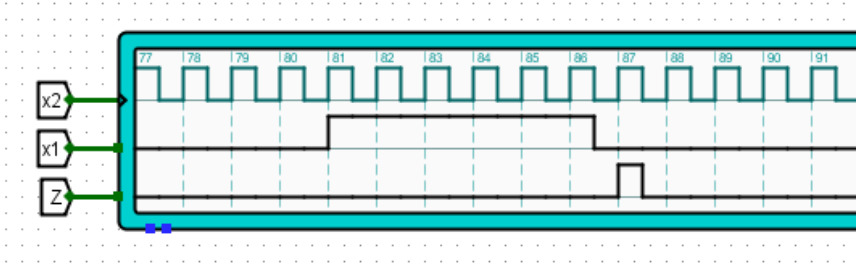


图 6.31　典型时间图



（二）在Logisim上实现教材第6章的习题

- **1、（设计实验，课后完成）习题6.1的实现**
  - 请在Logisim上实现习题6.1的电路（图6.36），并验证该电路的功能。

6.1　分析图 6.36 所示脉冲异步时序逻辑电路。
（1）作出状态表和状态图；
（2）说明电路逻辑功能。

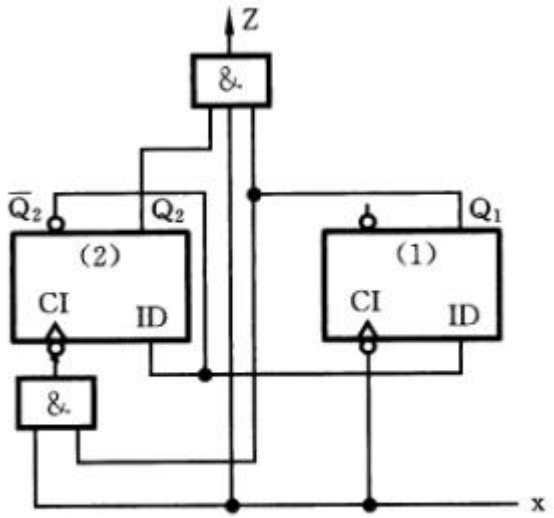


图 6.36　逻辑电路

- 2、（设计实验，课后完成）习题6.4的实现
  - 请在Logisim上实现习题6.4的电路（图6.39），并验证该电路的功能。

6.4　分析图 6.39 所示脉冲异步时序逻辑电路，作出时间图并说明该电路逻辑功能。

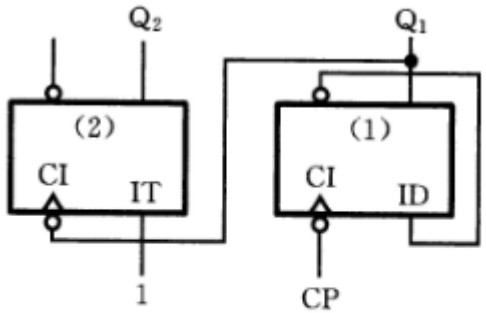


图 6.39　逻辑电路

- **3、（设计实验，课后完成）习题6.5的实现**
  - 请在Logisim上实现习题6.5的电路（3位二进制减1计数器），并验证该电路的功能。

6.5　用 D 触发器作为存储元件，设计一个脉冲异步时序逻辑电路。该电路在输入端 x 的脉冲作用下，实现 3 位二进制减 1 计数的功能，当电路状态为“000”时，在输入脉冲作用下输出端 Z 产生一个借位脉冲，平时 Z 输出 0。

- 4、（设计实验，课后完成）习题6.6的实现
  - 请在Logisim上实现习题6.6的电路（“x_1 - x_1 - x_2”序列检测器），并验证该电路的功能。

6.6　用 T 触发器作为存储元件，设计一个脉冲异步时序逻辑电路，该电路有两个输入 x_1 和 x_2，一个输出 Z，当输入序列为“x_1—x_1—x_2”时，在输出端 Z 产生一个脉冲，平时 Z 输出为 0。

- **5、（设计实验，课后完成）习题6.9的实现**
  - 请在Logisim上实现习题6.9的电路（图6.41），并验证该电路的功能。

6.9　分析图 6.41 所示电平异步时序逻辑电路，作出流程表和总态图，说明该电路的逻辑功能。

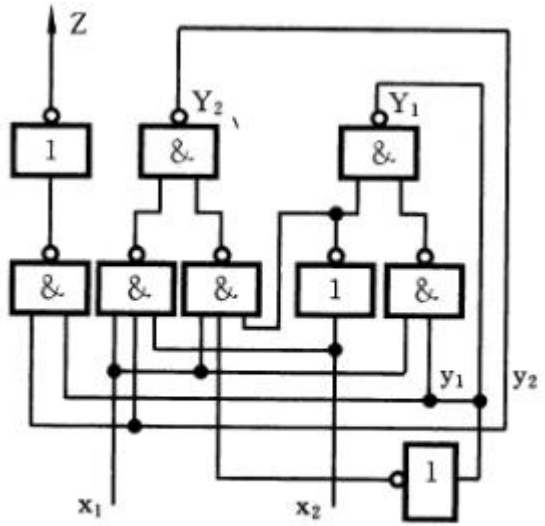


图 6.41　逻辑电路

- 6、（设计实验，课后完成）习题6.13的实现
  - 请在Logisim上实现习题6.13的电路（图6.43），并验证该电路的功能。

6.13　图 6.43 为某电平异步时序逻辑电路的结构框图。图中，

$$Y_2 = x_2 y_2 + \bar{x}_1 y_2 + x_2 \bar{x}_1 y_1$$

$$Y_1 = x_2 x_1 + \bar{x}_2\ \bar{x}_1 y_2 + x_1 y_2 \bar{y}_1$$

$$Z = y_2 y_1$$

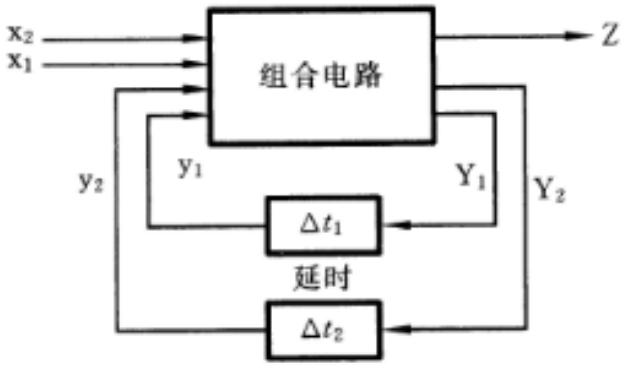


图 6.43　结构框图

- **7、（设计实验，课后完成）习题6.15的实现**
  - 请在Logisim上实现习题6.15的电路，并验证该电路的功能。

6.15　某电平异步时序逻辑电路有两个输入 x_1、x_2 和一个输出 Z。当 $x_2=1$ 时，Z 总为 0；当 $x_2=0$ 时，x_1 第一次从 0→1 的跳变使 Z 变为 1，该 1 输出信号一直保持到 x_2 由 0→1，才使 Z 为 0。试用与非门实现该电路功能。

- **8、（选做实验）习题6.2的实现**
  - 请在Logisim上实现习题6.2的电路（图6.37），并验证该电路的功能。

6.2　分析图 6.37 所示脉冲异步时序逻辑电路。

（1）作出状态表和时间图；

（2）说明电路逻辑功能。

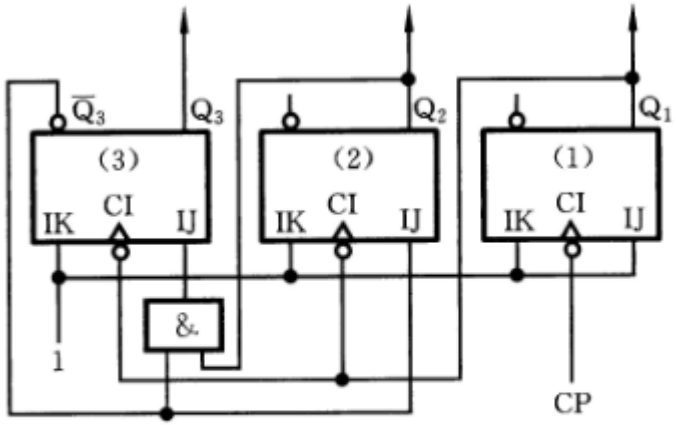


图 6.37　逻辑电路

- **9、（选做实验）习题6.3的实现**
  - 请在Logisim上实现习题6.3的电路（图6.38），并验证该电路的功能。

6.3　分析图 6.38 所示脉冲异步时序逻辑电路。
（1）作出状态表和状态图；
（2）说明电路逻辑功能。

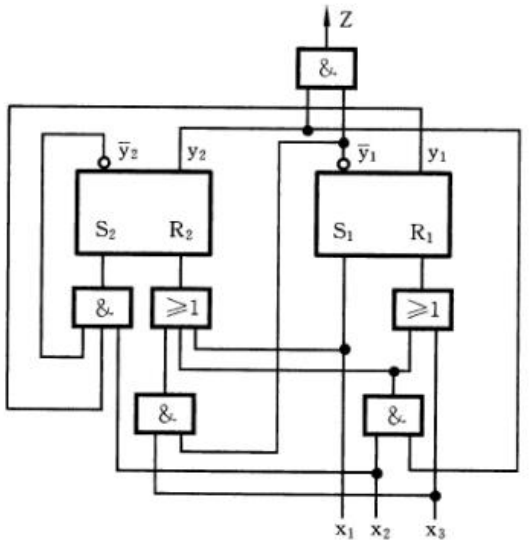


图 6.38　逻辑电路

- **10、（选做实验）习题6.7的实现**
  - 请在Logisim上实现习题6.7的电路（模4加1计数器），并验证该电路的功能。

6.7　试用与非门构成的基本 R-S 触发器设计一个模 4 加 1 计数器。

- **11、（选做实验）习题6.8的实现**
  - 请在Logisim上实现习题6.8的电路（图6.40），并验证该电路的功能。

6.8　分析图 6.40 所示电平异步时序逻辑电路，作出流程表。

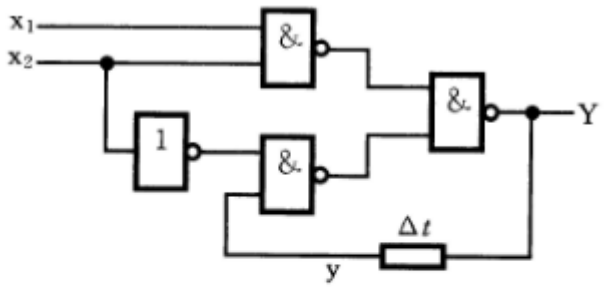


图 6.40　逻辑电路

- **12、（选做实验）习题6.14的实现**
  - **请在Logisim上实现习题6.14的电路，并验证该电路的功能。**

6.14　对表 6.36 所示的最简流程表进行无临界竞争的状态编码，并确定激励状态和输出函数表达式。

表 6.36　最简流程表

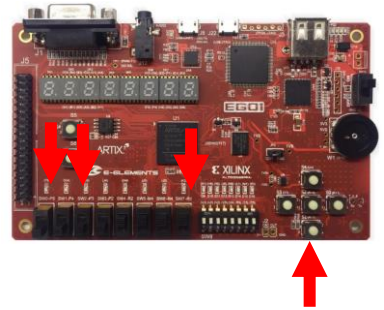
二次状态	激励状态 Y/输出 Z			
y	$x_2x_1=00$	$x_2x_1=01$	$x_2x_1=11$	$x_2x_1=10$
A	Ⓐ/0	Ⓐ/0	Ⓐ/0	C/d
B	Ⓑ/0	A/0	C/d	Ⓑ/0
C	B/d	A/d	Ⓒ/1	Ⓒ/1

第二部分：在FPGA开发板上实现异步时序逻辑电路

将“第5次实验（发给学生）\EGO1”目录中的所有文件，拷贝到\DigitalLogic\EGO1目录中

（一）在FPGA开发板上实现教材第6章的例题

- 1、（验证实验）例题6.1的实现
  - 分别用结构化描述方式和行为描述方式实现例题6.1的“模4加1计数器”，该电路的输入为x，状态为y_2、y_1，输出为Z。
  - 输入x为开发板上的S1按键，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。
  - 验证步骤：
    - 运行程序后（example_6_1_EGO1工程、example_6_1_1_EGO1工程），按S1按键（输入x）多次，观察3个LED灯（状态y_2、y_1，输出Z）的变化是不是与表6.2一致？（最左边的2个灯按照00、01、10、11、00……规律变化，11变到00时，最右边的灯闪一下/亮）



EGO1开发板

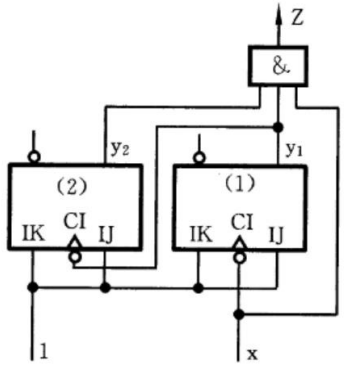


图 6.2　逻辑电路

表 6.2　状态表

现态		次态 $y_2^{n+1}y_1^{n+1}$/输出 Z
y_2	y_1	x=1
0	0	0 1 / 0
0	1	1 0 / 0
1	0	1 1 / 0
1	1	0 0 / 1

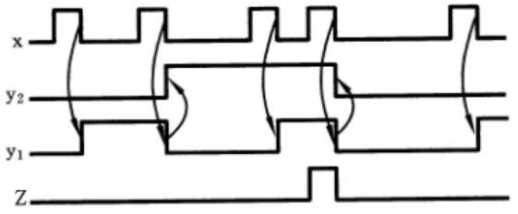


图 6.4　时间图

工程名：example_6_1_EGO1

```
module jk_flip_flop(
    input j, k,
    input cp,
    output q, qn
);
    reg y;
    always @(negedge cp)                //行为描述方式
    begin
            case({j,k})
                    1: y <= 0;                      // JK = 01     Y=0
                    2: y <= 1;                      // JK = 10     Y=1
                    3: y <= ~y;                     // JK = 11     Y=~Y
            endcase
    end
    assign  q= y;
    assign qn = ~y;
endmodule

module example_6_1(
    input x,
    input y2, y1,
    output ny2, ny1, z
);
    wire ny2n, ny1n;

    jk_flip_flop U1(.j(1),.k(1),.cp(x),.q(ny1),.qn(ny1n));          //结构化描述方式
    jk_flip_flop U2(.j(1),.k(1),.cp(y1),.q(ny2),.qn(ny2n));
    and_gate U3(.a(x),.b(y1),.c(y2),.f(z));

endmodule

module example_6_1_exe(
    input btn_1,                                //S1按钮
    output [15:0] led_pin                       //16个led灯
);
    reg y2, y1;
    example_6_1 U(.x(btn_1), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));
    initial begin
       y2=0;
       y1=0;
    end
    always @(*)                                 //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end
endmodule
```

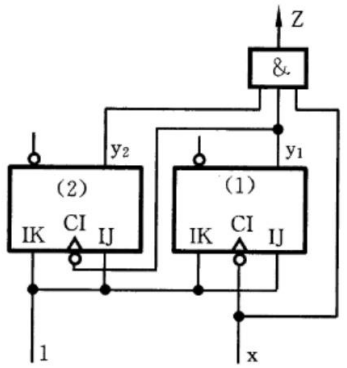


图 6.2　逻辑电路

结构化描述方式

工程名：example_6_1_1_EGO1

```
//采用行为描述方式实现例题6.1的模4加1计数器，输入（x）为开发板上的S1按键，状态（y2、y1）和

`timescale 1ns / 1ps

module example_6_1_1(
    input x,
    input y2, y1,
    output reg ny2, ny1, z
);

    always @(negedge x)                 //行为描述方式
    begin
            case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 1; z <= 0; end                    //
                    1: begin ny2 <= 1; ny1 <= 0; z <= 0; end                    //
                    2: begin ny2 <= 1; ny1 <= 1; z <= 0; end                    //
                    3: begin ny2 <= 0; ny1 <= 0; z <= 1; end                    //
            endcase
    end

endmodule

module example_6_1_1_exe(
    input btn_1,                                //S1按钮
    output [15:0] led_pin                       //16个led灯
);

    reg y2, y1;

    example_6_1_1 U(.x(btn_1), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
       y2=0;
       y1=0;
    end

    always @(*)                                 //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

行为描述方式

表 6.2　状态表

现态 y_2	现态 y_1	次态 $y_2^{n+1}y_1^{n+1}$/输出 Z (x=1)
0	0	0 1 / 0
0	1	1 0 / 0
1	0	1 1 / 0
1	1	0 0 / 1

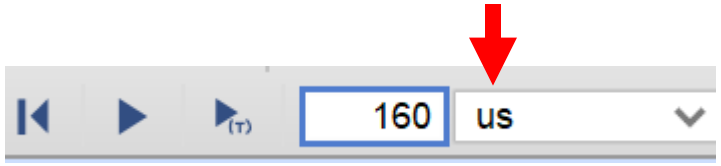
```
module and_gate(
    input a,
    input b,
    input c,
    output f
);
    reg y;
    always @(*)                 //行为描述方式
    begin
        y <= a & b & c;
    end
    assign f = y;
endmodule

module jk_flip_flop(
    input j, k,
    input cp,
    input rd,
    output q, qn
);
    reg y;
    always @(negedge cp  or negedge rd)              //行为描述方式
    begin
      if(rd==0)                                       // rd=0       y=0
            y <= 0;
      else                                            // rd=1
            case({j,k})
                  1: y <= 0;                        // JK = 01     Y=0
                  2: y <= 1;                        // JK = 10     Y=1
                  3: y <= ~y;                       // JK = 11     Y=~Y
            endcase
    end
    assign  q= y;
    assign qn = ~y;
endmodule

module example_6_1_2(
    input x,
    input y2, y1,
    input rd,
    output ny2, ny1, z
);
    wire ny2n, ny1n;
    jk_flip_flop U1(.j(1),.k(1),.cp(x),.rd(rd),.q(ny1),.qn(ny1n));          //结构化描述方式
    jk_flip_flop U2(.j(1),.k(1),.cp(y1),.rd(rd),.q(ny2),.qn(ny2n));         //结构化描述方式
    and_gate U3(.a(x),.b(y1),.c(y2),.f(z));                                 //结构化描述方式
endmodule
```

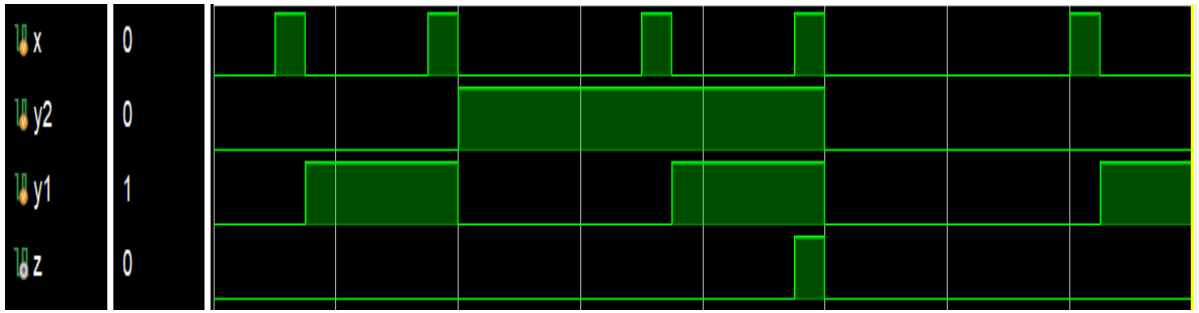
工程名：example_6_1_2_EGO1

对应结构化描述方式的仿真

仿真程序与开发板无关！

```
module example_6_1_2_sim();
    reg x;
    reg y2, y1;
    wire z;
    reg rd;
    wire ny2, ny1;
    example_6_1_2 U(.x(x), .y2(y2), .y1(y1), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;
        x=0;
        rd=0;

        #5000
        rd=1;

        #5000
        x=1;

        #5000
        x=0;

        #20000
        x=1;

        #5000
        x=0;

        #30000
        x=1;

        #5000
        x=0;

        #20000
        x=1;

        #5000
        x=0;

        #40000
        x=1;

        #5000
        x=0;
    end
```





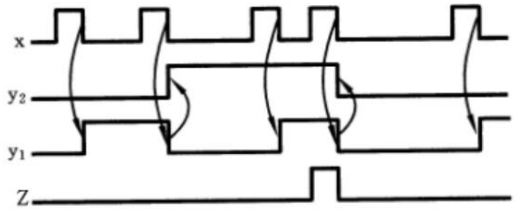


图 6.4　时间图

```
    always #1 begin y2 <= ny2; y1 <= ny1; end

endmodule
```

example_6_1_3_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//采用行为描述方式实现例题6.1的模4加1计数器，输入（x）为开发板上

`timescale 1ns / 1ps

module example_6_1_3(
    input x,
    input y2, y1,
    input rd,
    output reg ny2, ny1, z
);

    always @(negedge x or negedge rd)                    //行为描述方式

    begin
      if(rd==0)                                                  //  rd=0
            begin ny2 <= 0; ny1 <= 0; z <= 0; end
      else                                                       //  rd=1
            case({y2, y1})
                  0: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                  1: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                  2: begin ny2 <= 1; ny1 <= 1; z <= 0; end
                  3: begin ny2 <= 0; ny1 <= 0; z <= 1; end
            endcase
    end

endmodule
```

example_6_1_3_sim_EGO1.v - 记事本

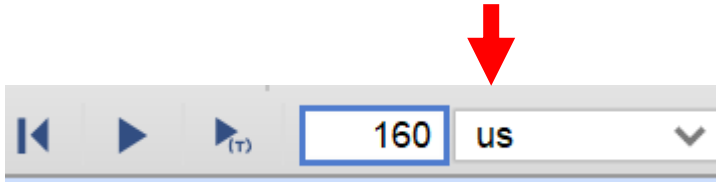
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

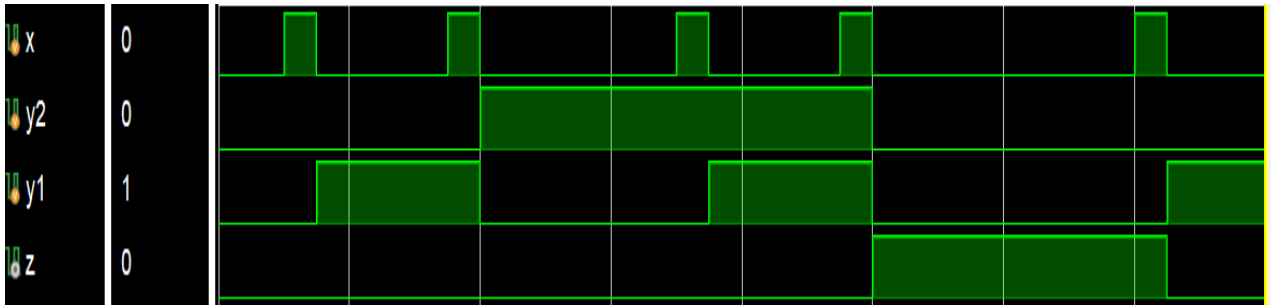
```
//例题6.1的仿真程序。

`timescale 1ns / 1ps

module example_6_1_3_sim();

    reg x;
    reg y2, y1;
    wire z;
    reg rd;
    wire ny2, ny1;

    example_6_1_3 U(.x(x), .y2(y2), .y1(y1), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;
        x=0;
        rd=0;

        #5000
        rd=1;

        #5000
        x=1;

        #5000
        x=0;

        #20000
        x=1;

        #5000
        x=0;

        #30000
        x=1;

        #5000
        x=0;

        #20000
        x=1;

        #5000
        x=0;

        #40000
        x=1;

        #5000
        x=0;
    end

    always #1 begin y2 <= ny2; y1 <= ny1; end

endmodule
```

工程名：example_6_1_3_EGO1

对应行为描述方式的仿真

仿真程序与开发板无关！





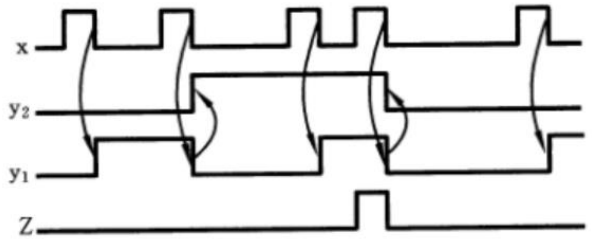


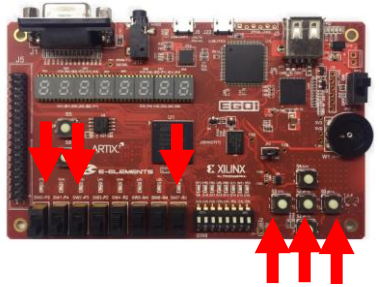
图 6.4 时间图

• 2、（验证实验）例题6.2的实现

- 分别用结构化描述方式和行为描述方式实现例题6.2的“x_1 - x_2 - x_3”序列检测器，该电路的输入为（x_1、x_2、x_3），状态为y_2、y_1，输出为Z。

- 输入（x_1、x_2、x_3）为开发板上的S3、S2、S0按键，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 验证步骤：
  - 运行程序后，依次按S3、S2、S0按键（输入x_1、x_2、x_3），观察3个LED灯（状态y_2、y_1，输出Z）的变化？（没有得到预期的结果）



EGO1开发板

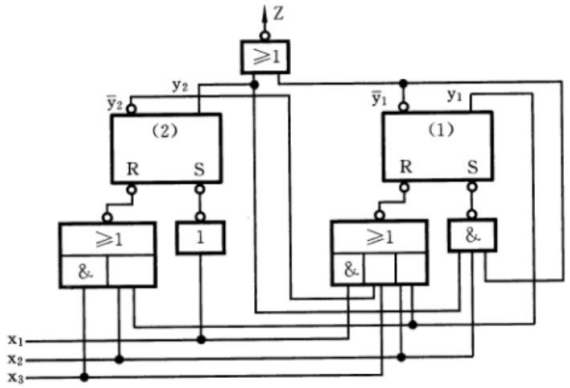


图 6.5　逻辑电路

表 6.4　状态表

现　态		次态 $y_2^{n+1}y_1^{n+1}$			输出
y_2	y_1	x_1	x_2	x_3	Z
0	0	10	00	00	0
0	1	10	00	00	1
1	0	10	11	00	0
1	1	10	00	01	0

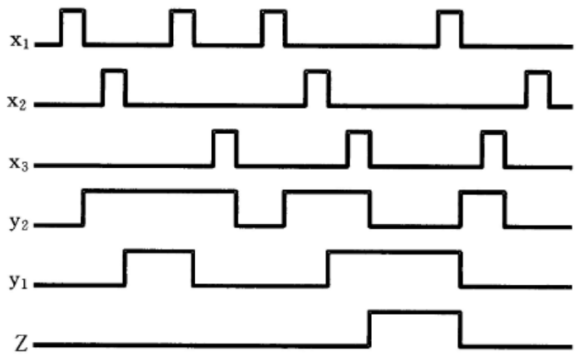


图 6.7　时间图

```
module rs_flip_flop(
    input r, s,
    output q, qn
);
    reg y;
    always @(*)                 //行为描述方式
    begin
            case({r,s})
                    0: y <= 1'bx;                   // RS = 00     Y=不定
                    1: y <= 0;                      // RS = 01     Y=0
                    2: y <= 1;                      // RS = 10     Y=1
            endcase
    end
    assign  q= y;
    assign qn = ~y;
endmodule
```

工程名：example_6_2_EGO1

没有得到预期的结果！有兴趣的同学可以分析原因！

表 3.13 与非门构成的基本 R-S 触发器功能表

R	S	Q^{n+1}	功能说明
0	0	d	不定
0	1	0	置 0
1	0	1	置 1
1	1	Q	不变

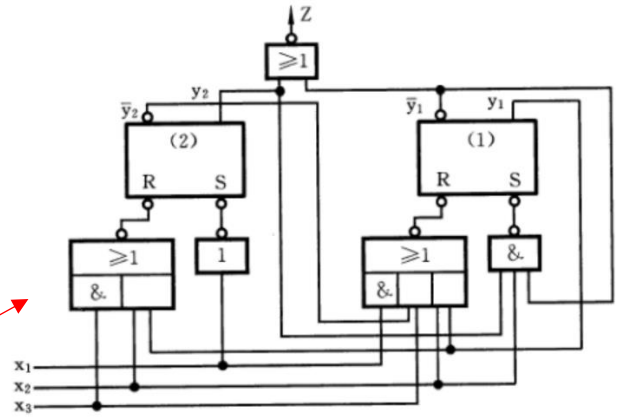


图 6.5 逻辑电路

结构化描述方式

```
module example_6_2(
    input x1, x2, x3,
    input y2, y2n, y1, y1n,
    output ny2, ny1, z
);

    wire s2, r2, s1, r1, t1, t2, t3, t4;
    wire ny2n, ny1n;

    not_gate U1(.a(x1),.f(s2));                                 //结构化描述方式
    and_gate2 U2(.a(x3),.b(x3),.f(t1));                         //结构化描述方式
    and_gate2 U3(.a(x2),.b(y1),.f(t2));                         //结构化描述方式
    nor_gate2 U4(.a(t1),.b(t2),.f(r2));                         //结构化描述方式
    nand_gate3 U5(.a(x2),.b(y2),.c(y1n),.f(s1));                //结构化描述方式
    and_gate2 U6(.a(x1),.b(x1),.f(t3));                         //结构化描述方式
    and_gate2 U7(.a(x3),.b(y2n),.f(t4));                        //结构化描述方式
    and_gate2 U8(.a(y1),.b(y1),.f(t5));                         //结构化描述方式
    nor_gate3 U9(.a(t3),.b(t4),.c(t5),.f(r1));                  //结构化描述方式
    rs_flip_flop U10(.r(r2),.s(s2),.q(ny2),.qn(ny2n));          //结构化描述方式
    rs_flip_flop U11(.r(r1),.s(s1),.q(ny1),.qn(ny1n));          //结构化描述方式
    nor_gate2 U12(.a(y2),.b(y1n),.f(z));                        //结构化描述方式

endmodule
```

```
module example_6_2_exe(
    input btn_3, btn_2, btn_0,                                  //3个按钮
    output [15:0] led_pin                                       //16个led灯
);

    reg y2, y2n, y1, y1n;

    example_6_2 U(.x1(btn_3), .x2(btn_2), .x3(btn_0), .y2(y2), .y2n(y2n), .y1(y1), .y1n(y1n), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y2n=1;
        y1=0;
        y1n=1;
    end

    always @(*)                                 //行为描述方式
    begin
            y2 <= led_pin[0];
            y2n <= ~led_pin[0];
            y1 <= led_pin[1];
            y1n <= ~led_pin[1];
    end

endmodule
```

example_6_2_1_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
//行为描述方式实现例题6.2的模4加1计数器，输入（x1、x2、x3）为

`timescale 1ns / 1ps

module example_6_2_1(
    input x1, x2, x3,
    input y2, y1,
    output reg ny2, ny1, z
);

    always @(*)
    begin
            if(x1==1 & x2==0 & x3==0) begin
                case({y2, y1})
                    0: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 1; ny1 <= 0; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    3: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                endcase
            end

            if(x1==0 & x2==1 & x3==0) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 0; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                endcase
            end

            if(x1==0 & x2==0 & x3==1) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 0; z <= 1; end
                    2: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                endcase
            end
    end

endmodule
```

工程名：example_6_2_1_EGO1

行为描述方式

没有得到预期的结果！有兴趣的同学可以分析原因！

表 6.4 状态表

现态		次态 $y_2^{n+1}y_1^{n+1}$			输出
y_2	y_1	x_1	x_2	x_3	Z
0	0	10	00	00	0
0	1	10	00	00	1
1	0	10	11	00	0
1	1	10	00	01	0

```
module example_6_2_1_exe(
    input btn_0, btn_2, btn_3,                          //3个按钮
    output [15:0] led_pin                               //16个led灯
);

    reg y2, y1;

    example_6_2_1 U(.x1(btn_3), .x2(btn_2), .x3(btn_0), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y1=0;
    end

    always @(*)                                         //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

example_6_2_2_sim_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

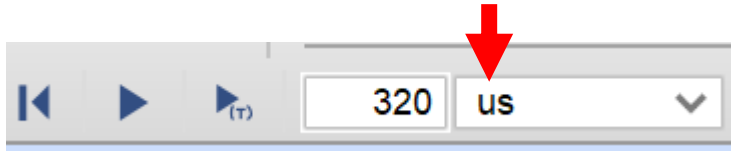
```
//例题6.2的仿真程序。

`timescale 1ns / 1ps

module example_6_2_2_sim();

    reg x1, x2, x3;
    reg y2, y1;
    wire z;
    reg rd;
    wire ny2, ny1;
    reg y2n, y1n;

    example_6_2_2 U(.x1(x1), .x2(x2), .x3(x3), .y2(y2), .y2n(y2n), .y1(y1), .y1n(y1n), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y2n=1;
        y1=0;
        y1n=1;
        x1=0;
        x2=0;
        x3=0;
        rd=0;

        #50000
        rd=1;

        #15000
        x1=1;

        #10000
        x1=0;

        #20000
        x2=1;

        #10000
        x2=0;

        #20000
        x1=1;

                #10000
                x1=0;

                #20000
                x3=1;

                #10000
                x3=0;

                #20000
                x2=1;

                #10000
                x2=0;
            end

            always #1 begin y2 <= ny2; y2n <= ~ny2; y1 <= ny1; y1n <= ~ny1; end

        endmodule
```

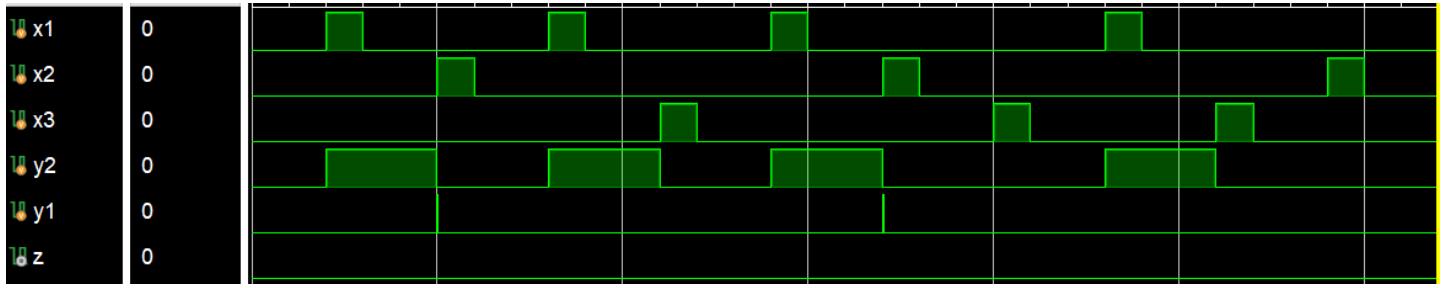
工程名：example_6_2_2_EGO1

对应结构化描述方式的仿真

仿真程序与开发板无关！

没有得到预期的结果！有兴趣的同学可以分析原因！





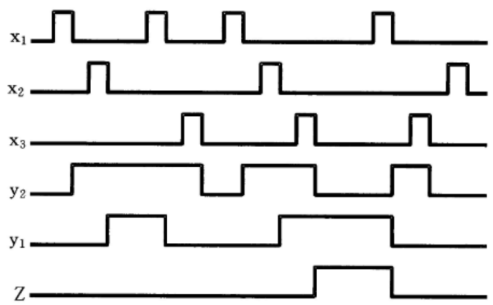


图 6.7 时间图

example_6_2_3_sim_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

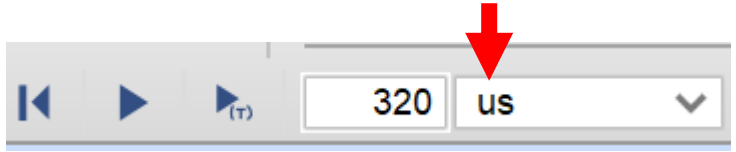
工程名：example_6_2_3_EGO1

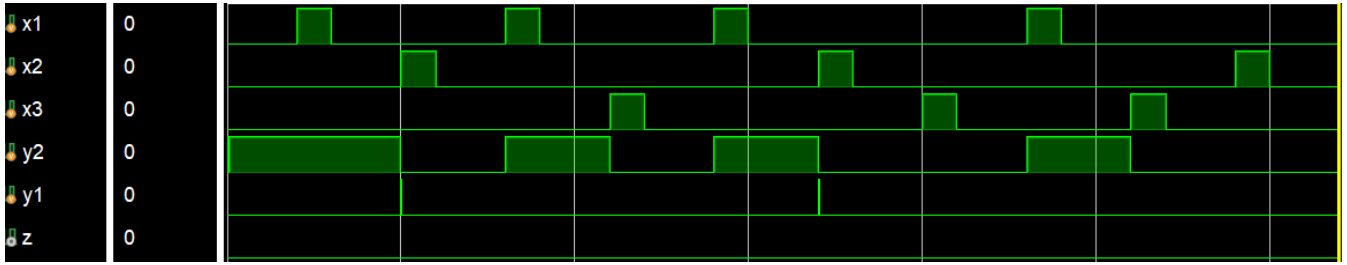
对应行为描述方式的仿真

仿真程序与开发板无关！

没有得到预期的结果！有兴趣的同学可以分析原因！

```
//例题6.2的仿真程序。

`timescale 1ns / 1ps

module example_6_2_3_sim();

    reg x1, x2, x3;
    reg y2, y1;
    wire z;
    reg rd;
    wire ny2, ny1;
    reg y2n, y1n;

    example_6_2_3 U(.x1(x1), .x2(x2), .x3(x3), .y2(y2), .y1(y1), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y2n=1;
        y1=0;
        y1n=1;
        x1=0;
        x2=0;
        x3=0;
        rd=0;

        #50000
        rd=1;

        #15000
        x1=1;

        #10000
        x1=0;
            #10000
            x2=0;
        end

        always #1 begin y2 <= ny2; y2n <= ~ny2; y1 <= ny1; y1n <= ~ny1; end

    endmodule
```





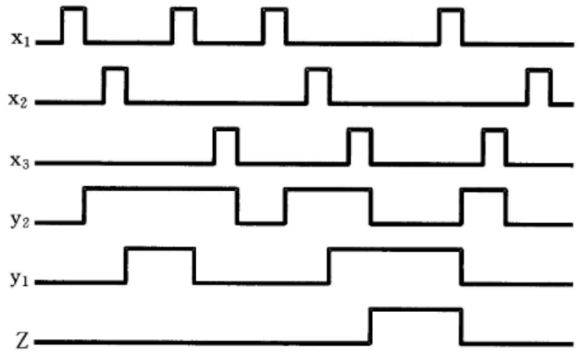


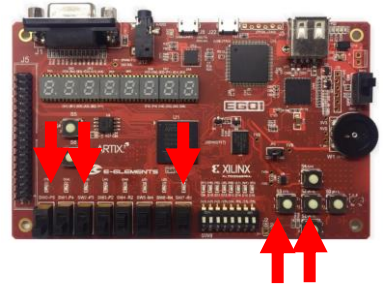
图 6.7 时间图

• 3、（验证实验）例题6.3的实现

- 请分别用结构化描述方式和行为描述方式实现例题6.3的“x_1 - x_2 - x_2”序列检测器，该电路的输入为x_1、x_2，状态为y_2、y_1，输出为Z。

- 输入x_1、x_2为开发板上的S3、S2按键，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 验证步骤：
  - 运行程序后，依次按S3、S2、S2按键（x_1 - x_2 - x_2），观察最右边LED灯（输出Z）的变化是不是与图6.8的波形相同？



EGO1开发板

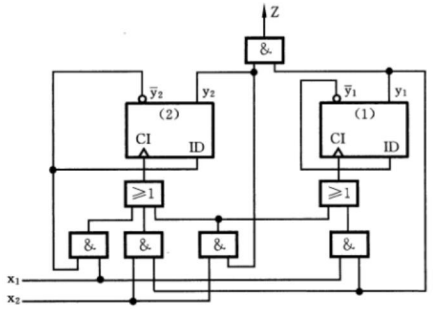


图 6.11　逻辑电路

表 6.11　二进制状态表

现态		次态 $y_2^{n+1}y_1^{n+1}$		输出
y_2	y_1	$x_1=1$	$x_2=1$	Z
0	0	10	00	0
0	1	10	11	0
1	0	10	01	0
1	1	10	00	1

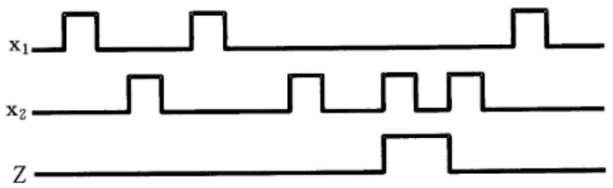


图 6.8　时间图

工程名：example_6_3_EGO1

```
module d_flip_flop(
    input cp,
    input d,
    output q, qn
);
    reg y;
    always @(posedge cp)                //行为描述方式
    begin
            case(d)
                    0: y <= 0;                      // D = 0     Y=0
                    1: y <= 1;                      // D = 1     Y=1
            endcase
    end
    assign  q= y;
    assign qn = ~y;
endmodule

module example_6_3(
    input x1, x2,
    input y2, y1,
    output ny2, ny1, z
);

    wire cp2, cp1;
    wire t1, t2, t3, t4;
    wire ny2n, ny1n;

    and_gate2 U1(.a(x1),.b(~y2),.f(t1));                    //结构化描述方式
    and_gate2 U2(.a(x2),.b(y1),.f(t2));                     //结构化描述方式
    and_gate2 U3(.a(x2),.b(y2),.f(t3));                     //结构化描述方式
    nor_gate3 U4(.a(t1),.b(t2),.c(t3),.f(cp2));             //结构化描述方式

    and_gate2 U5(.a(x1),.b(y1),.f(t4));                     //结构化描述方式
    nor_gate2 U6(.a(t3),.b(t4),.f(cp1));                    //结构化描述方式

    d_flip_flop U7(.cp(cp2),.d(~y2),.q(ny2),.qn(ny2n));     //结构化描述方式
    d_flip_flop U8(.cp(cp1),.d(~y1),.q(ny1),.qn(ny1n));     //结构化描述方式

    and_gate2 U9(.a(y1),.b(y2),.f(z));                      //结构化描述方式

endmodule
```

结构化描述方式

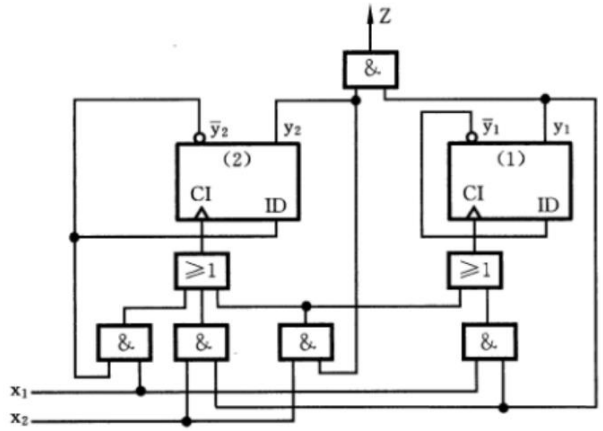


图 6.11　逻辑电路

```
module example_6_3_exe(
    input btn_3, btn_2,                                 //S3、S2按钮
    output [15:0] led_pin                               //16个led灯
);

    reg y2, y1;

    example_6_3 U(.x1(btn_3), .x2(btn_2), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y1=0;
    end

    always @(*)                                 //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

```
`timescale 1ns / 1ps

module example_6_3_1(
    input x1, x2,
    input y2, y1,
    output reg ny2, ny1, z
);
    always @(*)                 //行为描述方式
    begin
            if(x1 == 1 & x2 == 0)
                case({y2, y1})
                    0: begin ny2 <= 1; ny1 <= 0; z <= 0; end        // y2 y1 = 00    y2n+1 y1n+1 = 10    z=0
                    1: begin ny2 <= 1; ny1 <= 0; z <= 0; end        // y2 y1 = 01    y2n+1 y1n+1 = 10    z=0
                    2: begin ny2 <= 1; ny1 <= 0; z <= 0; end        // y2 y1 = 10    y2n+1 y1n+1 = 10    z=0
                    3: begin ny2 <= 1; ny1 <= 0; z <= 1; end        // y2 y1 = 11    y2n+1 y1n+1 = 10    z=1
                endcase
            if(x1 == 0 & x2 == 1)
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end        // y2 y1 = 00    y2n+1 y1n+1 = 00    z=0
                    1: begin ny2 <= 1; ny1 <= 1; z <= 0; end        // y2 y1 = 01    y2n+1 y1n+1 = 11    z=0
                    2: begin ny2 <= 0; ny1 <= 1; z <= 0; end        // y2 y1 = 10    y2n+1 y1n+1 = 01    z=0
                    3: begin ny2 <= 0; ny1 <= 0; z <= 1; end        // y2 y1 = 11    y2n+1 y1n+1 = 00    z=1
                endcase
    end
endmodule
```

工程名：example_6_3_1_EGO1

没有得到预期的结果！有兴趣的同学可以分析原因！

表 6.11　二进制状态表

现态		次态 $y_2^{n+1}y_1^{n+1}$		输出
y_2	y_1	$x_1=1$	$x_2=1$	Z
0	0	10	00	0
0	1	10	11	0
1	0	10	01	0
1	1	10	00	1

行为描述方式

```
module example_6_3_1_exe(
    input btn_3, btn_2,                         //S3、S2按钮
    output [15:0] led_pin                       //16个led灯
);

    reg y2, y1;
    example_6_3_1 U(.x1(btn_3), .x2(btn_2), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y1=0;
    end

    always @(*)                         //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

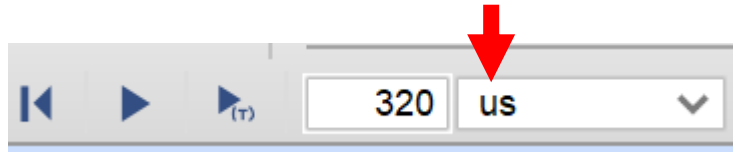
```
module d_flip_flop(
    input cp,
    input d,
    input rd,
    output q, qn
);
    reg y;
    always @(posedge cp  or negedge rd)                 //行为描述方式
    begin
        if(rd==0)
            y <= 0;
        else
            case(d)
                    0: y <= 0;                          // D = 0     Y=0
                    1: y <= 1;                          // D = 1     Y=1
            endcase
    end
    assign  q= y;
    assign qn = ~y;
endmodule

module example_6_3_2(
    input x1, x2,
    input y2, y1,
    input rd,
    output ny2, ny1, z
);

    wire cp2, cp1;
    wire t1, t2, t3, t4;
    wire ny2n, ny1n;

    and_gate2 U1(.a(x1),.b(~y2),.f(t1));                //结构化描述方式
    and_gate2 U2(.a(x2),.b(y1),.f(t2));                 //结构化描述方式
    and_gate2 U3(.a(x2),.b(y2),.f(t3));                 //结构化描述方式
    nor_gate3 U4(.a(t1),.b(t2),.c(t3),.f(cp2));         //结构化描述方式

    and_gate2 U5(.a(x1),.b(y1),.f(t4));                 //结构化描述方式
    nor_gate2 U6(.a(t3),.b(t4),.f(cp1));                //结构化描述方式

    d_flip_flop U7(.cp(cp2),.d(~y2),.rd(rd),.q(ny2),.qn(ny2n));   //结构化描述方式
    d_flip_flop U8(.cp(cp1),.d(~y1),.rd(rd),.q(ny1),.qn(ny1n));   //结构化描述方式

    and_gate2 U9(.a(y1),.b(y2),.f(z));                  //结构化描述方式

endmodule
```

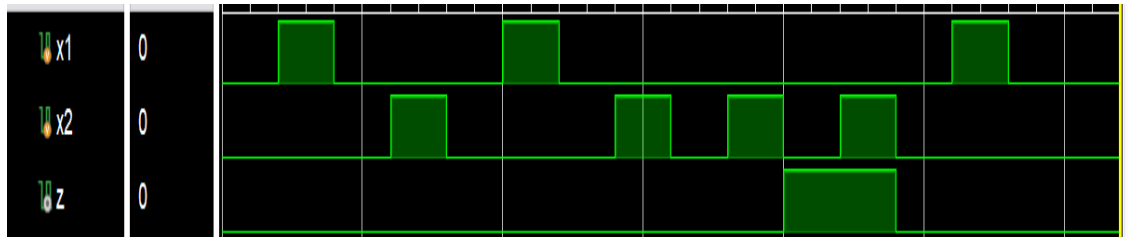
工程名：example_6_3_2_EGO1

对应结构化描述方式的仿真

仿真程序与开发板无关！

```
`timescale 1ns / 1ps

module example_6_3_2_sim();

    reg x1, x2;
    wire z;
    reg rd;
    reg y2, y1;
    wire ny2, ny1;

    example_6_3_2 U(.x1(x1), .x2(x2), .y2(y2), .y1(y1), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;
        x1=0;
        x2=0;
        rd=0;

        #5000
        rd=1;

        #15000
        x1=1;
        x2=0;

        #20000
        x1=0;
        x2=0;

        #20000
        x1=0;
        x2=1;

        #20000
        x1=0;
        x2=0;

        #20000
        x1=1;
        x2=0;

        #20000
        x1=0;
        x2=0;
        #20000
        x1=0;
        x2=0;
    end

    always #1 begin y2 <= ny2; y1 <= ny1; end

endmodule
```





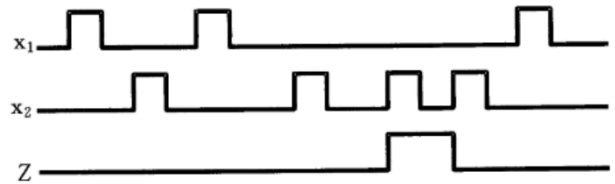


图 6.8　时间图

```
`timescale 1ns / 1ps

module example_6_3_3(
    input x1, x2,
    input y2, y1,
    input rd,
    output reg ny2, ny1, z
);

    always @(*)                    //行为描述方式
    begin
        if(rd==0)
            begin ny2 <= 0; ny1 <= 0; z <= 0; end
        else
            if(x1 == 1 & x2 == 0)
                case({y2, y1})
                    0: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    2: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    3: begin ny2 <= 1; ny1 <= 0; z <= 1; end
                endcase
            if(x1 == 0 & x2 == 1)
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 1; ny1 <= 1; z <= 0; end
                    2: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 0; z <= 1; end
                endcase
    end

endmodule
```

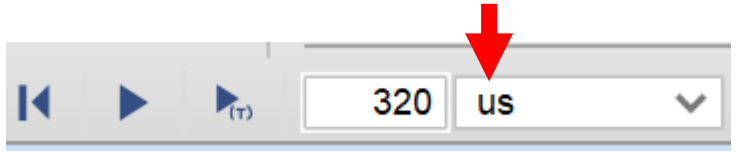
工程名：example_6_3_3_EGO1

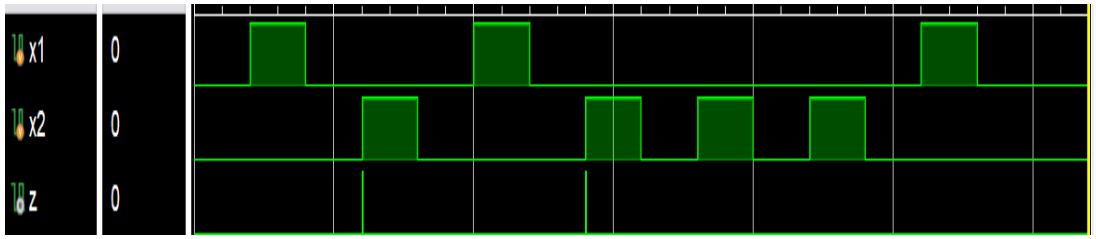
对应行为描述方式的仿真

仿真程序与开发板无关！

```
`timescale 1ns / 1ps

module example_6_3_3_sim();

    reg x1, x2;
    wire z;
    reg rd;
    reg y2, y1;
    wire ny2, ny1;

    example_6_3_3 U(.x1(x1), .x2(x2), .y2(y2), .y1(y1), .rd(rd), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;
        x1=0;
        x2=0;
        rd=0;

        #5000
        rd=1;

        #15000
        x1=1;
        x2=0;

        #20000
        x1=0;
        x2=0;

        #20000
        x1=0;
        x2=1;

        #20000
        x1=0;
        x2=0;

        #20000
        x1=1;
        x2=0;

        #20000
        x1=0;
        x2=0;
        #20000
        x1=0;
        x2=0;
    end

    always #1 begin y2 <= ny2; y1 <= ny1; end

endmodule
```

没有得到预期的仿真结果！有兴趣的同学可以分析原因！





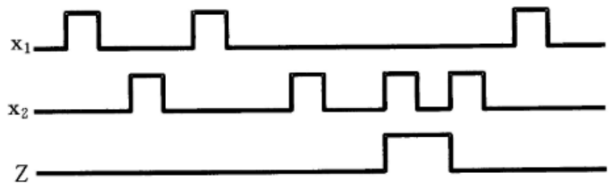


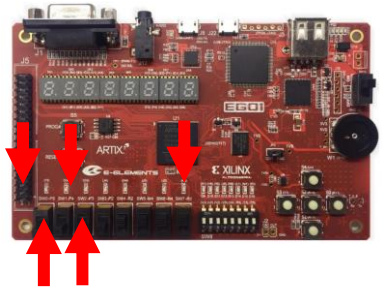
图 6.8　时间图

• 4、（验证实验）例题6.5的实现

- 请分别用结构化描述方式和行为描述方式实现例题6.5的“00->10->11”序列检测器，该电路的输入为（x_2、x_1），状态为（y_2、y_1），输出为Z。

- 输入（x_2、x_1）为开发板上的最左边的2个拨动开关，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 验证步骤（仿真）：
  - 观看仿真结果的波形是否与图6.19的波形相同（类似）？



EGO1开发板

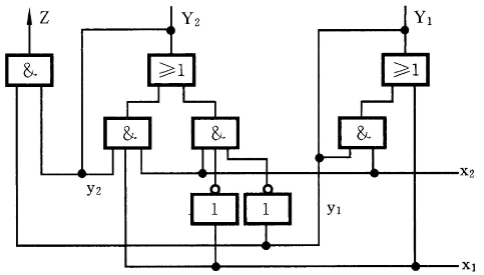


图 6.17　逻辑电路

表 6.18　流程表

二次状态		激励状态 Y_2Y_1				输出
y_2	y_1	$x_2x_1=00$	$x_2x_1=01$	$x_2x_1=11$	$x_2x_1=10$	Z
0	0	⓪⓪	01	01	10	0
0	1	00	⓪①	⓪①	⓪①	0
1	1	00	01	① ①	01	1
1	0	00	01	11	① ⓪	0

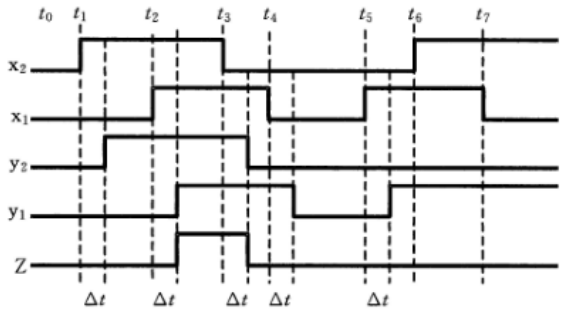


图 6.19　时间图

example_6_5_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
    end
    assign f = y;
endmodule

module example_6_5(
    input x2, x1,
    input y2, y1,
    output ny2, ny1, z
);

    wire t1, t2, t3, t4, t5;

    not_gate U1(.a(x1),.f(t1));                          //结构化描述方式
    not_gate U2(.a(y1),.f(t2));                          //结构化描述方式

    and_gate2 U3(.a(x2),.b(y1),.f(t3));                  //结构化描述方式

    and_gate3 U4(.a(t1),.b(t2),.c(x2),.f(t4));           //结构化描述方式
    and_gate3 U5(.a(x1),.b(x2),.c(y2),.f(t5));           //结构化描述方式

    or_gate2 U6(.a(x1),.b(t3),.f(ny1));                  //结构化描述方式
    or_gate2 U7(.a(t4),.b(t5),.f(ny2));                  //结构化描述方式

    and_gate2 U8(.a(y1),.b(y2),.f(z));                   //结构化描述方式

endmodule

module example_6_5_exe(
    input sw_pin[7:0],                                   //8个拨动开关
    output [15:0] led_pin                                //16个led灯
);

    reg y2, y1;

    example_6_5 U(.x2(sw_pin[7]), .x1(sw_pin[6]), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y1=0;
    end

    always @(*)                                          //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

工程名：example_6_5_EGO1

结构化描述方式

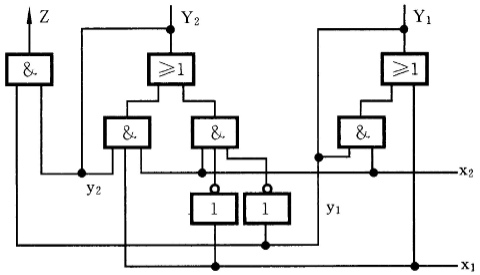


图 6.17 逻辑电路

可以综合、实现，无法生成bit文件！有兴趣的同学可以分析原因！

example_6_5_1_EGO1.v - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
module example_6_5_1(
    input x2, x1,
    input y2, y1,
    output reg ny2, ny1, z
);

    always @(*)                    //行为描述方式
    begin
            if(x2==0 & x1==0) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 0; z <= 1; end
                    2: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                endcase
            end

            if(x2==0 & x1==1) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                endcase
            end

            if(x2==1 & x1==1) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 1; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 1; z <= 0; end
                endcase
            end

            if(x2==1 & x1==0) begin
                case({y2, y1})
                    0: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                endcase
            end
    end

endmodule
```

工程名：example_6_5_1_EGO1

行为描述方式

表 6.18 流程表

二次状态 y_2 y_1	激励状态 Y_2Y_1 $x_2x_1=00$	$x_2x_1=01$	$x_2x_1=11$	$x_2x_1=10$	输出 Z
0 0	(00)	01	01	10	0
0 1	00	(01)	(01)	(01)	0
1 1	00	01	(11)	01	1
1 0	00	01	11	(10)	0

可以综合、实现，无法生成bit文件！有兴趣的同学可以分析原因！

```
module example_6_5_1_exe(
    input sw_pin[7:0],                          //8个拨动开关
    output [15:0] led_pin                       //16个led灯
);

    reg y2, y1;

    example_6_5_1 U(.x2(sw_pin[7]), .x1(sw_pin[6]), .y2(y2), .y1(y1), .ny2(led_pin[0]), .ny1(led_pin[1]), .z(led_pin[7]));

    initial begin
        y2=0;
        y1=0;
    end

    always @(*)                         //行为描述方式
    begin
            y2 <= led_pin[0];
            y1 <= led_pin[1];
    end

endmodule
```

```
module or_gate2(
    input a,
    input b,
    output f
);
    reg y;
    always @(*)                    //行为描述方式
    begin
        y <= a | b;
    end
    assign f = y;
endmodule

module example_6_5_2(
    input x2, x1,
    input y2, y1,
    output ny2, ny1, z
);

    wire t1, t2, t3, t4, t5;

    not_gate U1(.a(x1),.f(t1));
    not_gate U2(.a(y1),.f(t2));

    and_gate2 U3(.a(x2),.b(y1),.f(t3));

    and_gate3 U4(.a(t1),.b(t2),.c(x2),.f(t4));
    and_gate3 U5(.a(x1),.b(x2),.c(y2),.f(t5));

    or_gate2 U6(.a(x1),.b(t3),.f(ny1));
    or_gate2 U7(.a(t4),.b(t5),.f(ny2));

    and_gate2 U8(.a(y1),.b(y2),.f(z));

endmodule
```

example_6_5_2_sim_EGO1.v - 记事本

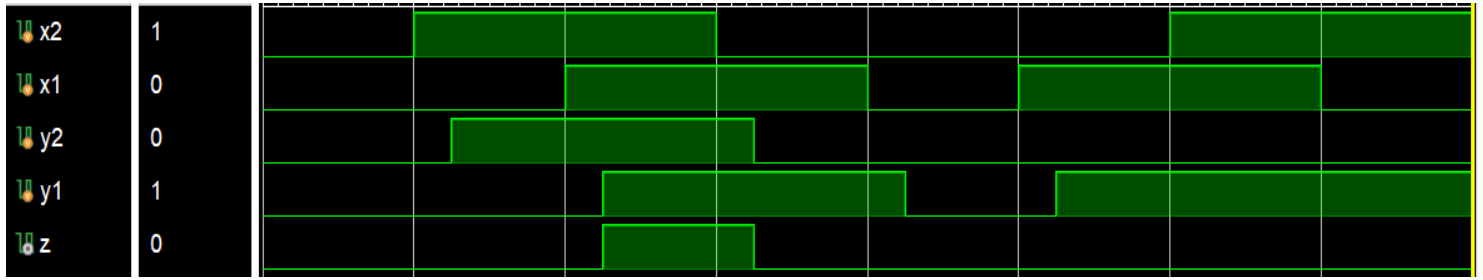
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
module example_6_5_2_sim();

    reg x2, x1;
    reg y2, y1;
    wire z, ny2, ny1;

    example_6_5_2 U(.x2(x2), .x1(x1), .y2(y2), .y1(y1), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;
        x2=0;
        x1=0;

        #20
        x2=1;
        x1=0;

        #20
        x2=1;
        x1=1;

        #20
        x2=0;
        x1=1;

        #20
        x2=0;
        x1=0;

        #20
        x2=0;
        x1=1;

        #20
        x2=1;
        x1=1;

        #20
        x2=1;
        x1=0;
    end

    always #5 begin y2 <= ny2; y1 <= ny1; end

endmodule
```

工程名：example_6_5_2_EGO1

对应结构化描述方式的仿真

仿真程序与开发板无关！



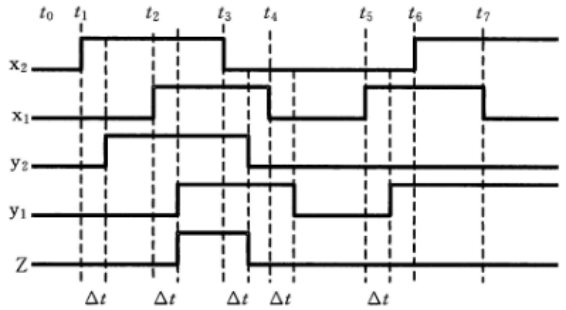


图 6.19 时间图

```v
module example_6_5_3(
    input x2, x1,
    input y2, y1,
    output reg ny2, ny1, z
);

    always @(*)                    //行为描述方式
    begin
            if(x2==0 & x1==0) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 0; z <= 1; end
                    2: begin ny2 <= 0; ny1 <= 0; z <= 0; end
                endcase
            end

            if(x2==0 & x1==1) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                endcase
            end

            if(x2==1 & x1==1) begin
                case({y2, y1})
                    0: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 1; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 1; z <= 0; end
                endcase
            end

            if(x2==1 & x1==0) begin
                case({y2, y1})
                    0: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                    1: begin ny2 <= 0; ny1 <= 1; z <= 0; end
                    3: begin ny2 <= 0; ny1 <= 1; z <= 1; end
                    2: begin ny2 <= 1; ny1 <= 0; z <= 0; end
                endcase
            end
    end

endmodule
```

example_6_5_3_sim_EGO1.v - 记事本

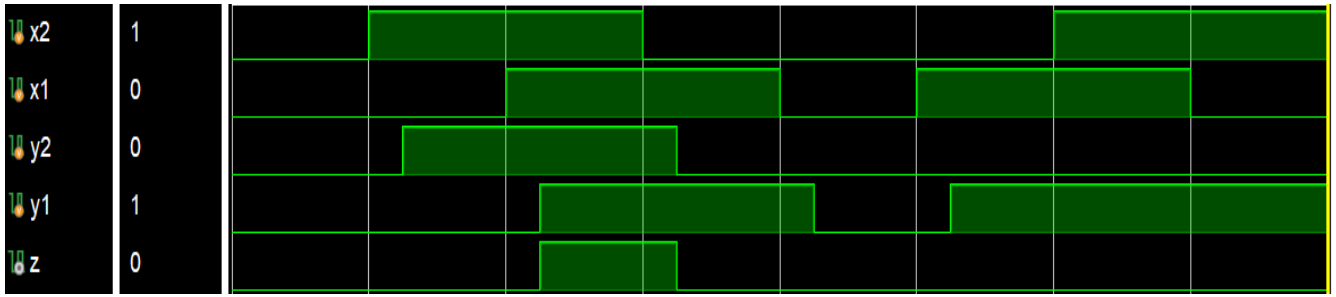
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```v
module example_6_5_3_sim();

    reg x2, x1;
    reg y2, y1;
    wire z, ny2, ny1;

    example_6_5_3 U(.x2(x2), .x1(x1), .y2(y2), .y1(y1), .ny2(ny2), .ny1(ny1), .z(z));

    initial begin
        #0
        y2=0;
        y1=0;

        x2=0;
        x1=0;

        #20
        x2=1;
        x1=0;

        #20
        x2=1;
        x1=1;

        #20
        x2=0;
        x1=1;

        #20
        x2=(
        x1=(

        #20
        x2=0;
        x1=1;

        #20
        x2=1;
        x1=1;

        #20
        x2=1;
        x1=0;
    end

    always #5 begin y2 <= ny2; y1 <= ny1; end

endmodule
```

工程名：example_6_5_3_EGO1

对应行为描述方式的仿真

仿真程序与开发板无关！



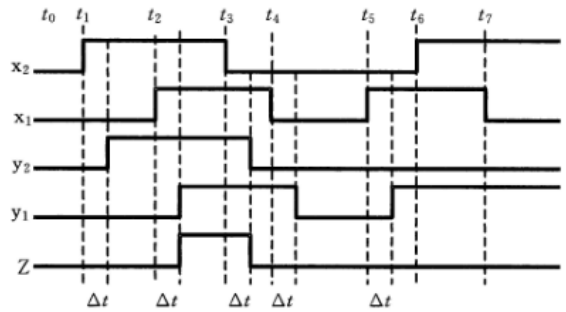


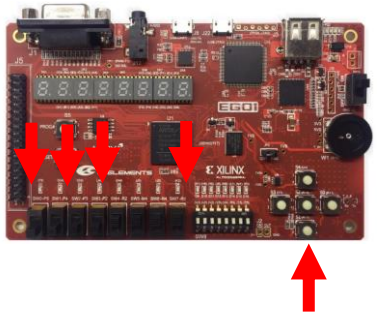
图 6.19 时间图

• 5、（设计实验，课后完成）例题6.4的实现

- 请分别用结构化描述方式和行为描述方式实现例题6.4的异步模8加1计数器，该电路的输入为x，状态为（y_3、y_2、y_1），输出为Z。

- 输入x为开发板上的S1按键，状态（y_3、y_2、y_1）为开发板上最左边的3个LED灯，输出Z为开发板上最右边的LED灯。

- 验证步骤（在开发板上）：
  - 运行程序后，按按S1按键多次，观察4个LED灯（状态y_3、y_2、y_1，输出Z）的变化情况，是不是实现了模8加1计数器的功能？



EGO1开发板

- EGO1开发板（结构化描述方式）：工程命名为example_6_4_EGO1，设计文件命名为example_6_4_EGO1.v，约束文件为EGO1.xdc。
- EGO1开发板（行为描述方式）：工程命名为example_6_4_1_EGO1，设计文件命名为example_6_4_1_EGO1.v，约束文件为EGO1.xdc。

- EGO1开发板（仿真——结构化描述方式）：工程命名为example_6_4_2_EGO1，设计文件命名为example_6_4_2_EGO1.v，仿真文件命名为example_6_4_2_sim_EGO1.v，约束文件为EGO1.xdc。
- EGO1开发板（仿真——行为描述方式）：工程命名为example_6_4_3_EGO1，设计文件命名为example_6_4_3_EGO1.v，仿真文件命名为example_6_4_3_sim_EGO1.v，约束文件为EGO1.xdc。

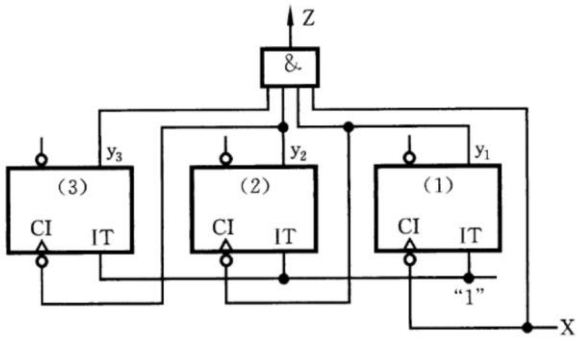


图 6.13　逻辑电路

表 6.13　二进制状态表

现　态	次态 y_3^{n+1} y_2^{n+1} y_1^{n+1}/输出 Z
y_3 y_2 y_1	x=1
0 0 0	0 0 1 / 0
0 0 1	0 1 0 / 0
0 1 0	0 1 1 / 0
0 1 1	1 0 0 / 0
1 0 0	1 0 1 / 0
1 0 1	1 1 0 / 0
1 1 0	1 1 1 / 0
1 1 1	0 0 0 / 1

• 6、（设计实验，课后完成）例题6.11的实现

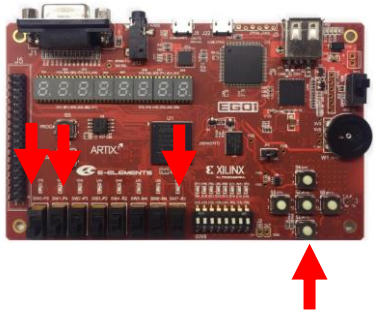
- 请分别用结构化描述方式和行为描述方式实现例题6.11的单脉冲发生器，该电路的输入为（x_2、x_1），状态为（y_2、y_1），输出为Z。

- 输入x_2接开发板的时钟脉冲，输入x_1接开发板的S1按键，状态（y_2、y_1）和输出Z分别开发板上最左边的2个LED灯以及最右边的LED灯。

- 验证步骤（在开发板上）：
  - 运行程序，按S1按，观察最右边LED灯（输出Z）的变化情况是不是与图6.31的波形相同？

- EGO1开发板（结构化描述方式）：工程命名为example_6_11_EGO1，设计文件命名为example_6_11_EGO1.v，约束文件为EGO1.xdc。
- EGO1开发板（行为描述方式）：工程命名为example_6_11_1_EGO1，设计文件命名为example_6_11_1_EGO1.v，约束文件为EGO1.xdc。

- EGO1开发板（仿真——结构化描述方式）：工程命名为example_6_11_2_EGO1，设计文件命名为example_6_11_2_EGO1.v，仿真文件命名为example_6_11_2_sim_EGO1.v，约束文件为EGO1.xdc。
- EGO1开发板（仿真——行为描述方式）：工程命名为example_6_11_3_EGO1，设计文件命名为example_6_11_3_EGO1.v，仿真文件命名为example_6_11_3_sim_EGO1.v，约束文件为EGO1.xdc。



EGO1开发板

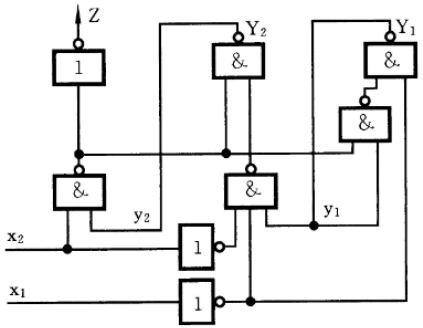


图 6.35　逻辑电路

表 6.33　二进制流程表

二次状态		激励状态 Y_2Y_1/输出 Z			
y_2	y_1	$x_2x_1=00$	$x_2x_1=01$	$x_2x_1=11$	$x_2x_1=10$
0	0	(00)/0	01/0	01/0	(00)/0
0	1	11/0	(01)/0	(01)/0	(01)/0
1	1	(11)/0	d/d	d/d	10/d
1	0	00/d	d/d	d/d	(10)/1

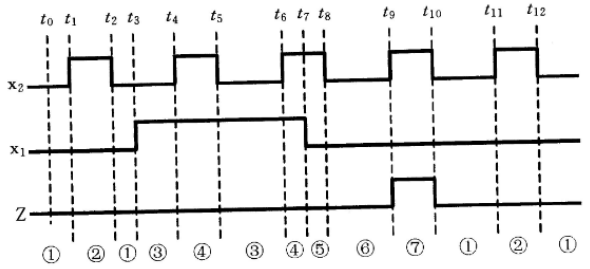
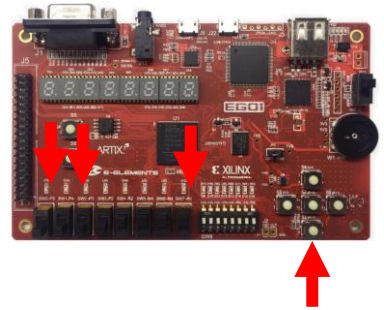


图 6.31　典型时间图

（二）在FPGA开发板上实现教材第6章的习题

- **1、（选做实验）习题6.1的实现**
  - 请分别用结构化描述方式或行为描述方式实现习题6.1的电路（图6.36），输入x为开发板上的S1按键，状态（Q_2、Q_1）和输出Z为开发板上最左边的2个LED灯以及最右边的LED灯。
  - 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.1　分析图 6.36 所示脉冲异步时序逻辑电路。

（1）作出状态表和状态图；

（2）说明电路逻辑功能。

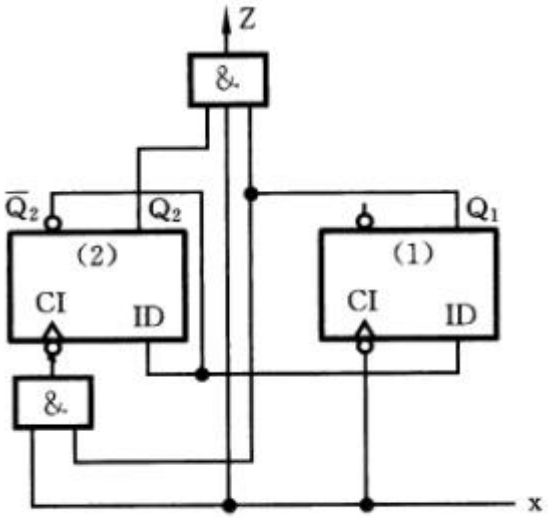
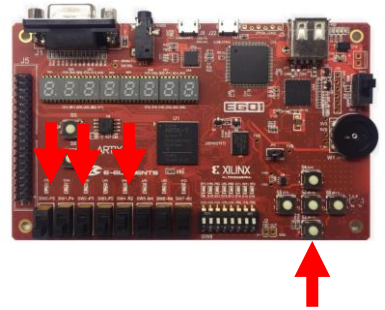


图 6.36　逻辑电路

• 2、（选做实验）习题6.2的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.2的电路（图6.37），输入CP为开发板上的S1按键，状态（Q_3、Q_2、Q_1）为开发板上最左边的3个LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.2　分析图 6.37 所示脉冲异步时序逻辑电路。

(1) 作出状态表和时间图；

(2) 说明电路逻辑功能。

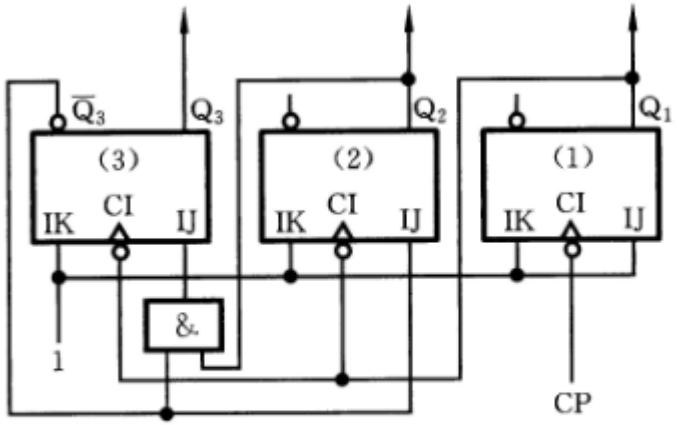
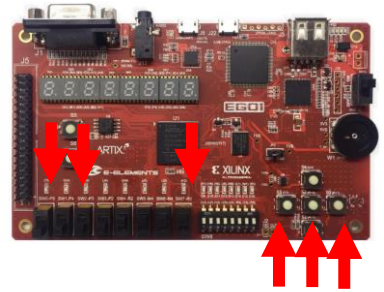


图 6.37　逻辑电路

• 3、（选做实验）习题6.3的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.3的电路（图6.38），输入（x_1、x_2、x_3）为开发板上的S3、S2、S0按键，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.3 分析图 6.38 所示脉冲异步时序逻辑电路。

(1) 作出状态表和状态图；

(2) 说明电路逻辑功能。

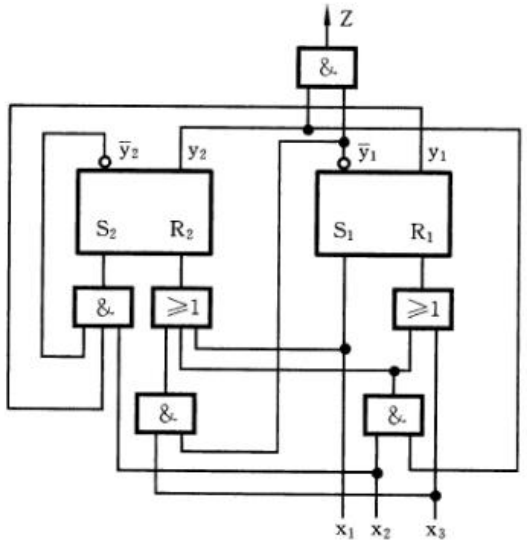
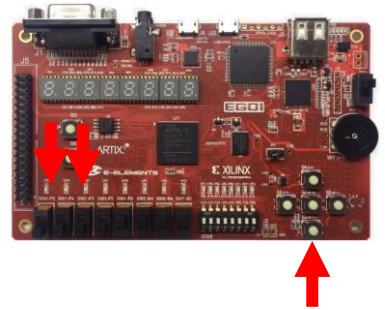


图 6.38 逻辑电路

• 4、（选做实验）习题6.4的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.4的电路（图6.39），输入CP为开发板上的S1按键，状态（Q_2、Q_1）为开发板上最左边的2个LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.4　分析图 6.39 所示脉冲异步时序逻辑电路，作出时间图并说明该电路逻辑功能。

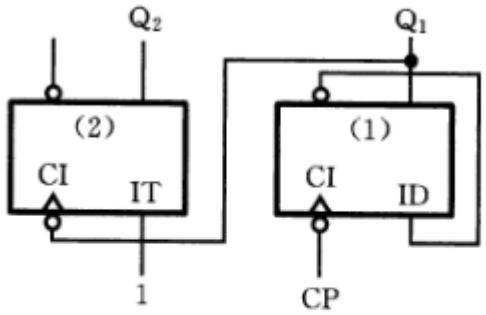
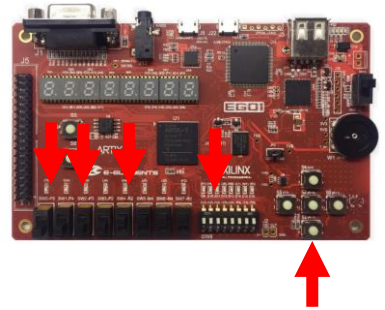


图 6.39　逻辑电路

• 5、（选做实验）习题6.5的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.5的电路（3位二进制减1计数器），输入x为开发板上的S1按键，状态（y_3、y_2、y_1）为开发板上最左边的3个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



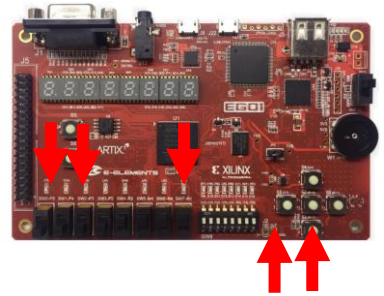
EGO1开发板

6.5　用 D 触发器作为存储元件，设计一个脉冲异步时序逻辑电路。该电路在输入端 x 的脉冲作用下，实现 3 位二进制减 1 计数的功能，当电路状态为“000”时，在输入脉冲作用下输出端 Z 产生一个借位脉冲，平时 Z 输出 0。

• 6、（选做实验）习题6.6的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.6的电路（“x_1 - x_1 - x_2”序列检测器），输入（x_1、x_2）为开发板上的S3、S2按键，状态（n个）为开发板上最左边的n个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



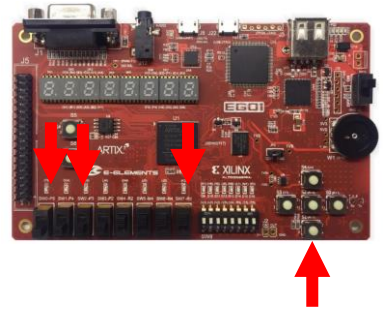
EGO1开发板

6.6 用 T 触发器作为存储元件，设计一个脉冲异步时序逻辑电路，该电路有两个输入 x_1 和 x_2，一个输出 Z，当输入序列为“x_1—x_1—x_2”时，在输出端 Z 产生一个脉冲，平时 Z 输出为 0。

• 7、（选做实验）习题6.7的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.7的电路（模4加1计数器），输入x为开发板上的S1按键（ACE1开发板为KEY1按键），状态（n个）为开发板上最左边的n个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



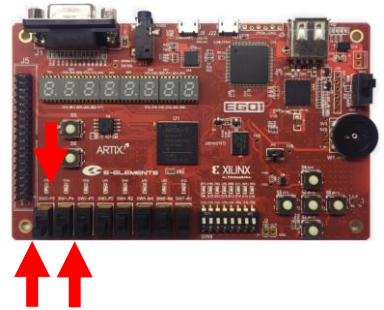
EGO1开发板

6.7　试用与非门构成的基本 R-S 触发器设计一个模 4 加 1 计数器。

• 8、（选做实验）习题6.8的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.8的电路（图6.40），输入（x_1、x_2）为开发板上最左边的2个拨动开关，状态（y）为开发板上最左边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.8　分析图 6.40 所示电平异步时序逻辑电路，作出流程表。

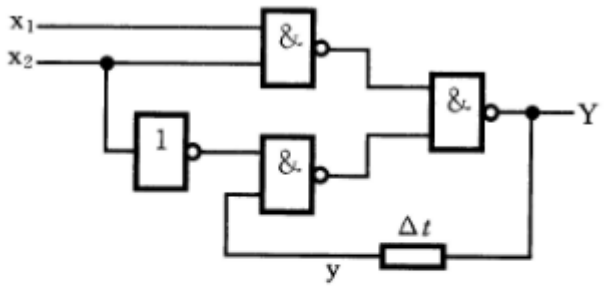
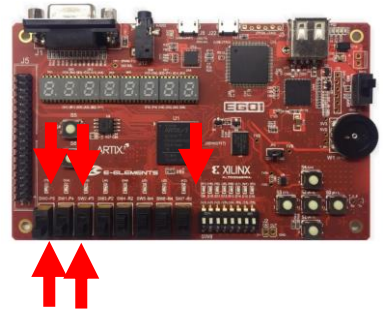


图 6.40　逻辑电路

• 9、（选做实验）习题6.9的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.9的电路（图6.41），输入（x_1、x_2）为开发板上最左边的2个拨动开关，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.9　分析图 6.41 所示电平异步时序逻辑电路，作出流程表和总态图，说明该电路的逻辑功能。

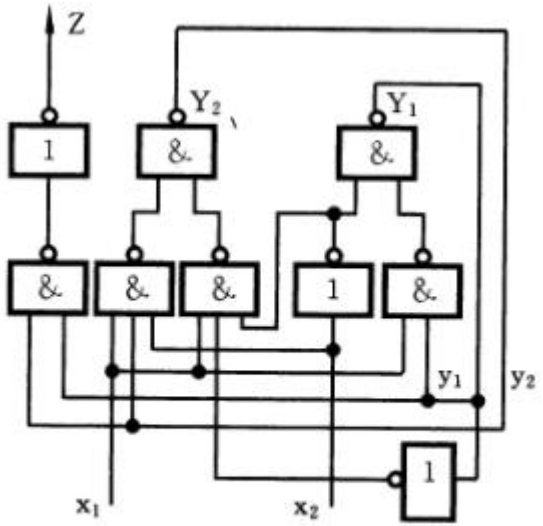
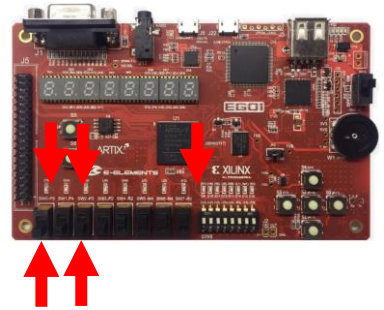


图 6.41　逻辑电路

• 10、（选做实验）习题6.13的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.13的电路（图6.43），输入（x_1、x_2）为开发板上最左边的2个拨动开关，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.13　图 6.43 为某电平异步时序逻辑电路的结构框图。图中，

$$Y_2 = x_2y_2 + \bar{x}_1y_2 + x_2\bar{x}_1y_1$$

$$Y_1 = x_2x_1 + \bar{x}_2\,\bar{x}_1y_2 + x_1y_2\bar{y}_1$$

$$Z = y_2y_1$$

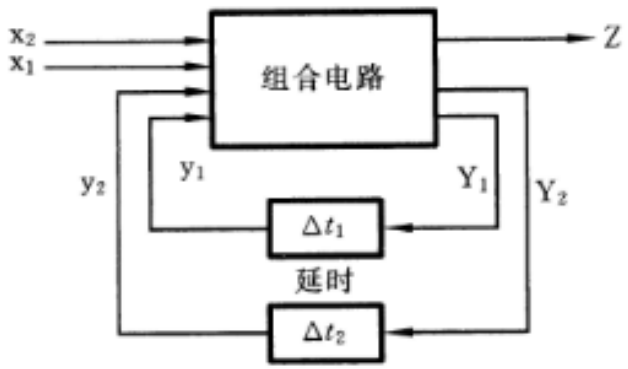
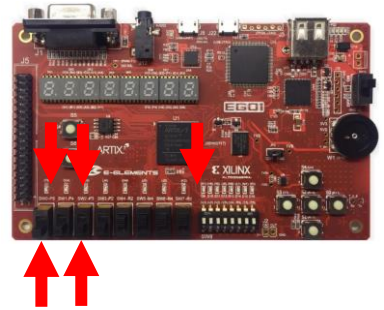


图 6.43　结构框图

• 11、（选做实验）习题6.14的实现

- 请分别用结构化描述方式或行为描述方式实现习题6.14的电路，输入（x_1、x_2）为开发板上最左边的2个拨动开关，状态（y_2、y_1）为开发板上最左边的2个LED灯，输出Z为开发板上最右边的LED灯。

- 请在开发板上（或编写仿真程序）验证该电路的功能。



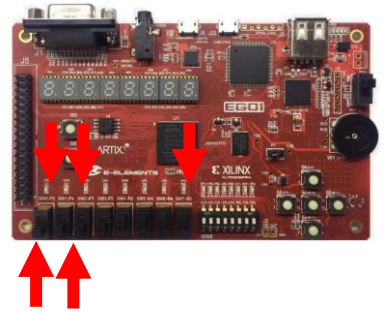
EGO1开发板

6.14　对表 6.36 所示的最简流程表进行无临界竞争的状态编码，并确定激励状态和输出函数表达式。

表 6.36　最简流程表

二次状态	激励状态 Y/输出 Z			
y	$x_2x_1=00$	$x_2x_1=01$	$x_2x_1=11$	$x_2x_1=10$
A	Ⓐ/0	Ⓐ/0	Ⓐ/0	C/d
B	Ⓑ/0	A/0	C/d	Ⓑ/0
C	B/d	A/d	Ⓒ/1	Ⓒ/1

- **12、（选做实验）习题6.15的实现**
  - 请分别用结构化描述方式或行为描述方式实现习题6.15的电路，输入（x_1、x_2）为开发板上最左边的2个拨动开关，状态（n个）为开发板上最左边的n个LED灯，输出Z为开发板上最右边的LED灯。
  - 请在开发板上（或编写仿真程序）验证该电路的功能。



EGO1开发板

6.15 某电平异步时序逻辑电路有两个输入 x_1、x_2 和一个输出 Z。当 $x_2=1$ 时，Z 总为 0；当 $x_2=0$ 时，x_1 第一次从 0→1 的跳变使 Z 变为 1，该 1 输出信号一直保持到 x_2 由 0→1，才使 Z 为 0。试用与非门实现该电路功能。

实验要求

- 1、在Logisim上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 2、在FPGA开发板上完成验证实验，将实验过程中的主要结果通过截屏（或拍照）的方式，拷贝到实验报告中，并给予适当的文字说明。
- 3、在Logisim上完成设计实验，设计文件命名为：第6章习题电路的实现.circ。
- 4、在FPGA开发板上完成设计实验，工程文件、设计文件、约束文件请严格按照规定的要求命名。
- 5、实验报告命名为：学号+姓名+第5次实验报告.docx。
- 6、将设计文件、实验报告打包压缩成1个压缩文件，命名为：学号+姓名+第5次实验.zip（或.rar），并上传到FTP上，上传截止日期：2024年11月17日晚上24点。
- 7、鼓励有兴趣的同学完成选做实验。

Thanks