第四次实验——流水灯

一、实验目的

1.使用 Verilog 编写流水灯，并在开发板上实现。

2.实现方向控制、速度控制。

二、实验过程

1.实现流水显示—— Display.v

流水显示是这个实验的基础。下面两张图是代码截图：

```
23  module Display(
24      input CLK_in,
25      input [1 : 0] Dir,
26      output reg [15 : 0] Light
27      );
28
29      reg [4 : 0] Num;
30
31      always@(posedge CLK_in) begin
32          if (Dir == 2'b00) begin
33              Num = Num;
34          end
35          if (Dir == 2'b10) begin
36              if (Num == 0) begin
37                  Num = 16;
38              end
39              Num = Num - 1;
40          end
41          if (Dir == 2'b11) begin
42              Num = Num + 1;
43              if (Num == 16) begin
44                  Num = 0;
45              end
46              end
47      end
```

```
    always@(*) begin
        case (Num)
            0: Light = 16'b1000000000000000;
            1: Light = 16'b0100000000000000;
            2: Light = 16'b0010000000000000;
            3: Light = 16'b0001000000000000;
            4: Light = 16'b0000100000000000;
            5: Light = 16'b0000010000000000;
            6: Light = 16'b0000001000000000;
            7: Light = 16'b0000000100000000;
            8: Light = 16'b0000000010000000;
            9: Light = 16'b0000000001000000;
            10: Light = 16'b0000000000100000;
            11: Light = 16'b0000000000010000;
            12: Light = 16'b0000000000001000;
            13: Light = 16'b0000000000000100;
            14: Light = 16'b0000000000000010;
            15: Light = 16'b0000000000000001;
        endcase
    end

endmodule
```

基本思路是， `Display` 模块要能根据方向选择选择流水方向。其流水显示的更新时间由输入的 `CLK_in` 决定。唯一的输出就是指定要亮哪一个灯。

由于方向有三个——即向前、向后、暂停，因此方向变量 `Dir` 需要两位。这里我这样规定：如果第一位是 `0` ，那么表示暂停；否则，若第二位为 `0` ，那么向左流动；若第二位为 `1` ，那么向右流动。

由于总共有 `16` 个灯，所以记录要亮起的灯的编号 `Num` 至少需要四位。由 `Dir` 决定 `Num` 如何变化。

每有信号改变，就更新一次灯亮起的状态。输出的十六位 `Light` 变量，每一位上若为 `1` 就代表其指示的灯要亮起。这样，要想实现流水效果，就应该由前到后依次设置一位为 `1` ，如图所示。

2.实现方向选择—— `Direct.v`

方向选择决定了流水线显示的方向。这个模块负责根据按下的方向选择键输出方向。

显然，三个方向对应三个按钮即三个一位输入，以及一个方向选择输出的两位信号 `Dir_Sel` 。代码如下截图所示：

```
23  module Direct(
24      input Button_Left,
25      input Button_Right,
26      input Button_Down,
27      output [1 : 0] Dir_Sel
28      );
29
30      reg [1 : 0] temp;
31
32      always@(*) begin
33          if (Button_Down == 1) begin
34              temp <= 2'b00;
35          end
36          if (Button_Left == 1) begin
37              temp <= 2'b10;
38          end
39          if (Button_Right == 1) begin
40              temp <= 2'b11;
41          end
42      end
43
44      assign Dir_Sel = temp;
45
46  endmodule
```

每当有一次按钮按下，就更新 `temp` 的值，并赋值给输出信号 `Dir_Sel` 。

这里，指定左按钮是向左流动，下按钮是暂停，右按钮是向右流动。

3.实现速度调控—— `CLK_Div.v`

因为 `Display.v` 是根据输入的时钟信号更新显示的，因此要想调控速度，就要从输入的时钟信号下手。这与之前做的分频器是相似的。代码截图如下：

```v
module CLK_Div #(parameter N = 12500000)(
    input CLK_in,
    input [1 : 0] Sp,
    output CLK_out
    );

    reg [31 : 0] counter;
    reg [31 : 0] MAX;
    reg out;

    always@(*) begin
        if (Sp == 2'b00) begin
            MAX <= N;
        end
        if (Sp == 2'b01) begin
            MAX <= 2 * N;
        end
        if (Sp == 2'b10) begin
            MAX <= 3 * N;
        end
        if (Sp == 2'b11) begin
            MAX <= 4 * N;
        end
    end
    always@(posedge CLK_in) begin
        if (counter == MAX) begin
            counter <= 0;
        end
        else begin
            counter <= counter + 1;
        end
    end

    always@(posedge CLK_in) begin
        if (counter == MAX) begin
                out <= !out;
        end
    end

    assign CLK_out = out;

endmodule
```

这里，我实现的是四档速度控制，由两个拨板开关实现。其原理是由输入的速度档位决定 Counter 的上限值 MAX ， 每当 Counter 累计到 MAX 就将输出反相一次并将 Counter 归零。

4.模块联合——（top）FluentLight.v

以上三个是基本模块。现在 top 文件中统筹之。代码截图如下：

```
module FluentLight(
    input CLK,
    input [1 : 0] speed,
    input Button_left,
    input Button_right,
    input Button_down,

    output [15 : 0] light
    );

    wire CLK_wire;
    wire [1 : 0] Dir_wire;
    CLK_Div clk_div (
        .CLK_in( CLK ),
        .Sp( speed ),
        .CLK_out( CLK_wire )
    );

    Direct direct (
        .Button_Left( Button_left ),
        .Button_Right( Button_right ),
        .Button_Down( Button_down ),
        .Dir_Sel( Dir_wire )
    );

    Display display (
        .CLK_in( CLK_wire ),
        .Dir( Dir_wire ),
        .Light( light )
    );

endmodule
```

Direct CLK_Div 模块为 Display 提供输入，故只需要两个 wire 连线，即 CLK_wire [1 : 0] Dir_wire 。将每个变量对应即可。

5.约束文件—— FluentLight.xdc

约束文件代码如下：

```
set_property PACKAGE_PIN W5 [get_ports CLK]
set_property PACKAGE_PIN T17 [get_ports Button_right]
set_property PACKAGE_PIN W19 [get_ports Button_left]
set_property PACKAGE_PIN U17 [get_ports Button_down]
set_property PACKAGE_PIN V17 [get_ports {speed[0]}]
set_property PACKAGE_PIN V16 [get_ports {speed[1]}]
set_property PACKAGE_PIN L1 [get_ports {light[15]}]
set_property PACKAGE_PIN P1 [get_ports {light[14]}]
set_property PACKAGE_PIN N3 [get_ports {light[13]}]
set_property PACKAGE_PIN P3 [get_ports {light[12]}]
set_property PACKAGE_PIN U3 [get_ports {light[11]}]
set_property PACKAGE_PIN W3 [get_ports {light[10]}]
set_property PACKAGE_PIN V3 [get_ports {light[9]}]
set_property PACKAGE_PIN V13 [get_ports {light[8]}]
set_property PACKAGE_PIN V14 [get_ports {light[7]}]
set_property PACKAGE_PIN U14 [get_ports {light[6]}]
set_property PACKAGE_PIN U15 [get_ports {light[5]}]
set_property PACKAGE_PIN W18 [get_ports {light[4]}]
set_property PACKAGE_PIN V19 [get_ports {light[3]}]
set_property PACKAGE_PIN U19 [get_ports {light[2]}]
set_property PACKAGE_PIN E19 [get_ports {light[1]}]
set_property PACKAGE_PIN U16 [get_ports {light[0]}]
```

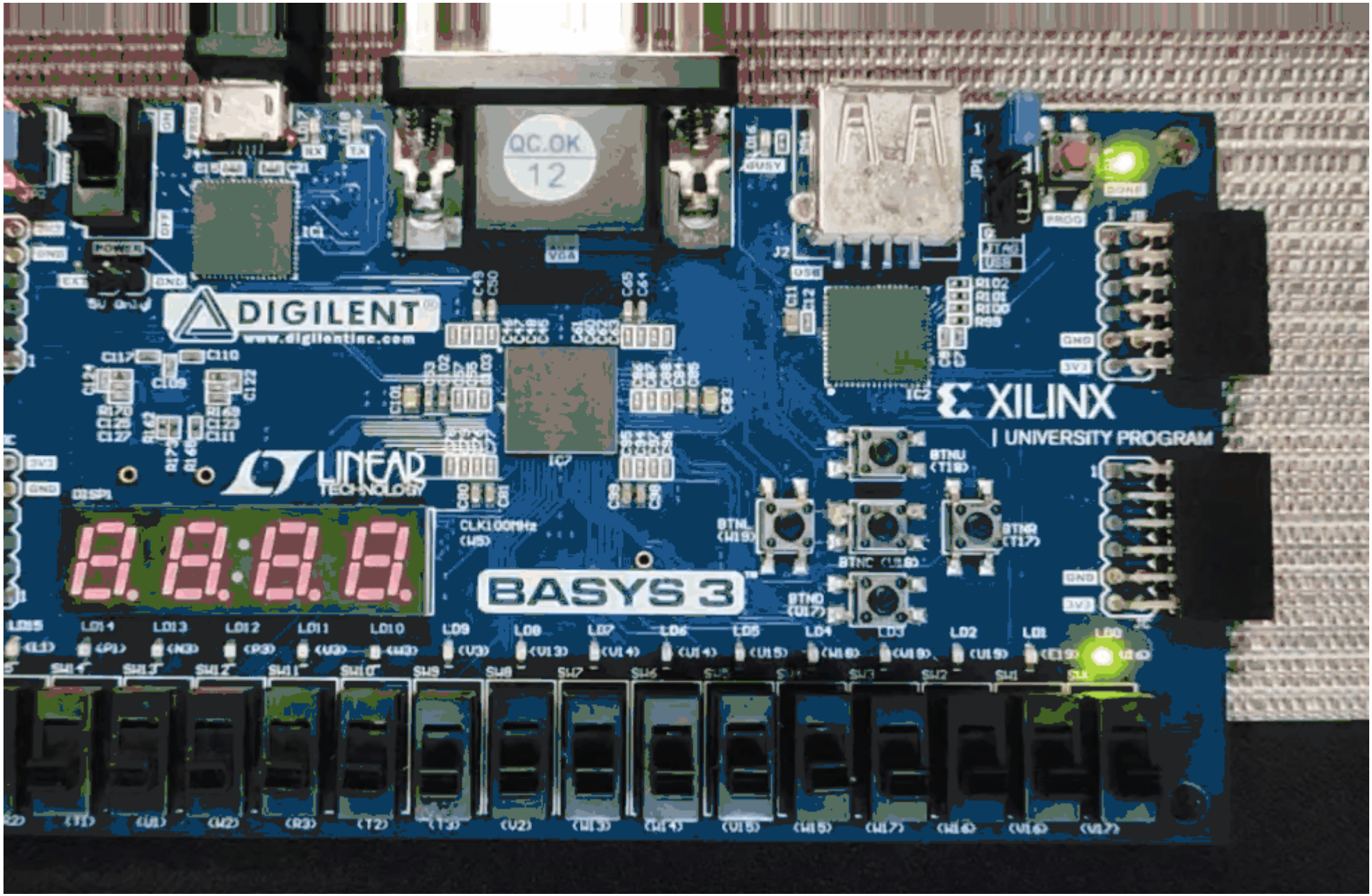
将 `[15 : 0] light` 直接每一位对应开发板上一个对应编号的 `LED` 。

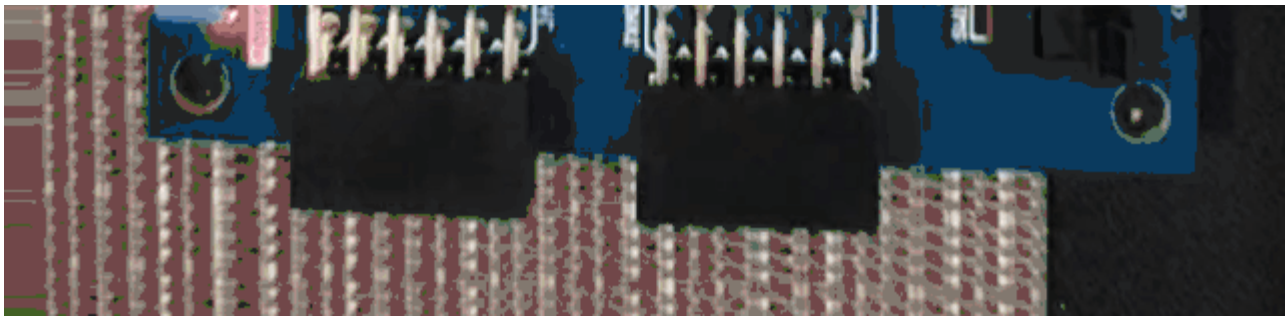
`[1 : 0] speed` 速度选择由最右边两个拨板开关约束。

`Button_Left  Button_Down  Button_Right` 即是开发板上方向键的左、下、右。

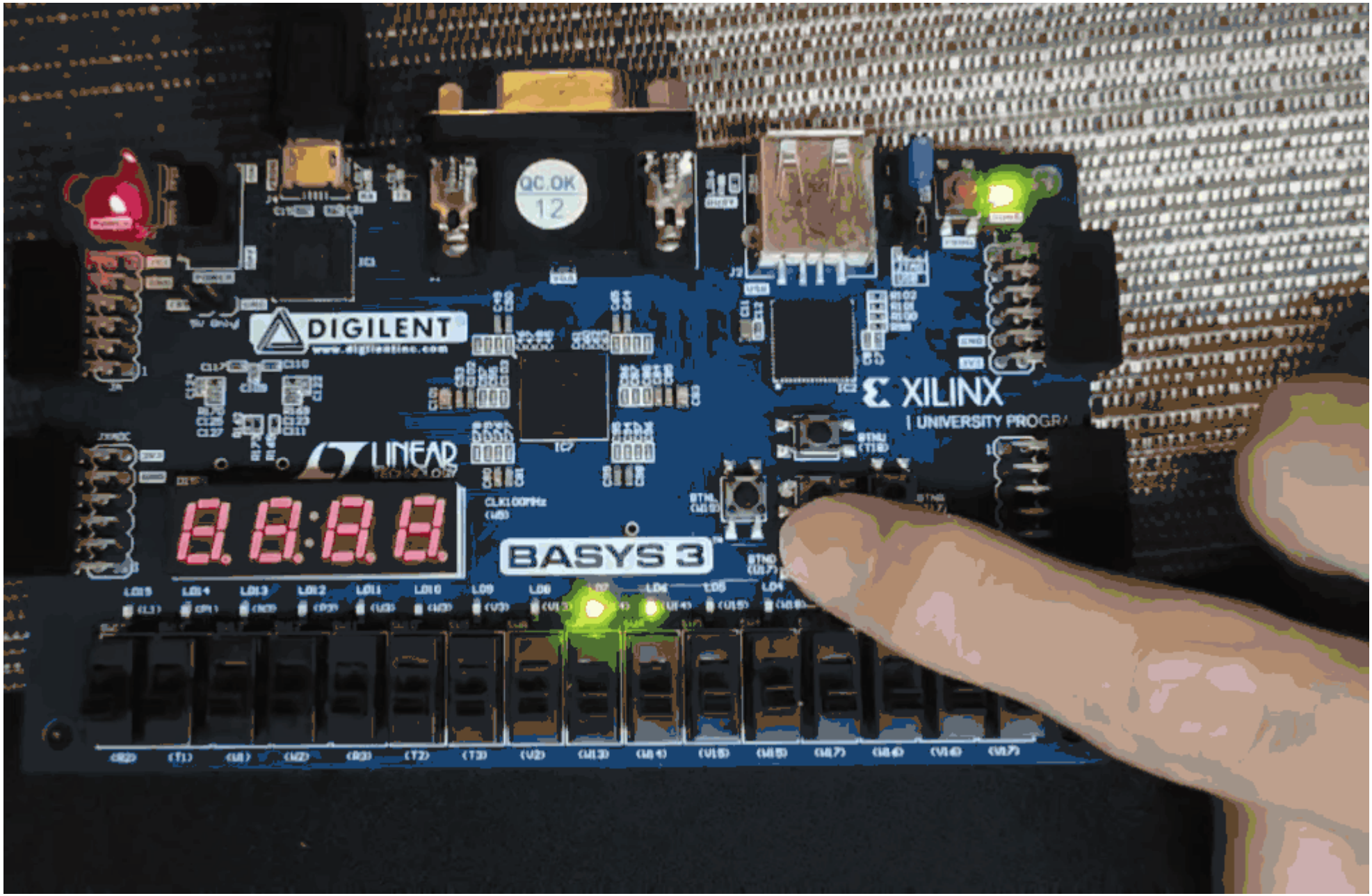
三、实验效果

在默认状态下，流水灯从右往左，以最快速度流动。


DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
QC.OK
12



通过按下方向键，可以改变流动方向。


QC.OK
12
DIGILENT
www.digilentinc.com
XILINX
UNIVERSITY PROGR
LINEAR
BASYS 3