

实验三 移位寄存器

电子科学与工程学院 刘时宜 201180078

实验日期: 2021 年 11 月 17 日

2021 年 11 月 24 日

指导老师: 高健

点击目录、书签栏、以及行文中的图表标号的均可跳转至相应页面

目录

1 实验目的	2
2 实验仪器与主要器材	2
3 实验原理	2
4 实验过程	3
4.1 移位寄存器单源周期触发测试	3
4.1.1 实验结果	3
4.2 移位寄存器单次触发测试电路	5
4.3 数据收发器	7
4.3.1 数据发送端电路搭建	8
4.3.1.1 十分频模块	8
4.3.1.2 分频电路模块	8
4.3.1.3 数据选择器	9
4.3.1.4 十进制计数器	9
4.3.1.5 4 位 D 触发器	10
4.3.1.6 4 位 D 触发器	10
4.3.1.7 8 位并入串出移位寄存器	11
4.3.1.8 8 位串入并移位寄存器	11
4.3.1.9 静态显示模块	12
4.3.1.10 主电路模块	12

4.3.2	数据接收端电路搭建	14
4.3.2.1	分频电路模块	14
4.3.2.2	分频电路模块	15
4.3.2.3	8 位串入并出移位寄存器	16
4.3.2.4	8 位串入并出移位寄存器	17
4.3.2.5	D 锁存器	17
4.3.2.6	16 位 BCD 码移位寄存器	18
4.3.2.7	动态显示模块	18
4.3.2.8	主电路模块	19
4.3.3	综合、编译、下载至开发板进行实验	21
5	实验总结	23

1 实验目的

- a. 验证移位寄存器的功能
- b. 利用移位寄存器搭建数据收发器

2 实验仪器与主要器材

仪器:

Basys3 FPGA 开发板	1 台
KEYSIGHT DSOX1102AG 示波器	1 台
示波器高频探头	1 套
ROGOL DM3068 万用表	1 台

软件:

Multisim	14.1
Digilent Adept	2.19.2
Vivado	2015.4

耗材:

导线	若干
----	----

3 实验原理

在各种复杂的数字电路中，不但需要对二值信号进行算术运算和逻辑运算，还经常需要将这些信号和运算结果保存下来。因此，需要使用具有记忆功能的基本逻辑单元。能够储存

1 位二值信号的基本单元电路统称触发器。

寄存器用于储存一组二值代码，它被广泛运用于各类数字系统和数字计算机当中。因为一个触发器能够储存 1 位二值数据，所以用 N 个触发器组成的寄存器可以储存 N 位二进制代码。

移位寄存器除了具有储存代码的功能以外，还具有移位功能。所谓移位功能，是指寄存器里储存的代码能够在移位脉冲的作用下依次左移或者右移。因此，移位寄存器不但可以用来寄存代码，还可以用来实现数据的串行-并行转换、数值的运算以及数据处理等。

4 实验过程

4.1 移位寄存器单源周期触发测试

实验电路如图4.1.1所示。将已经编译好的 bit 文件下载到开发板上，观察实验现象。

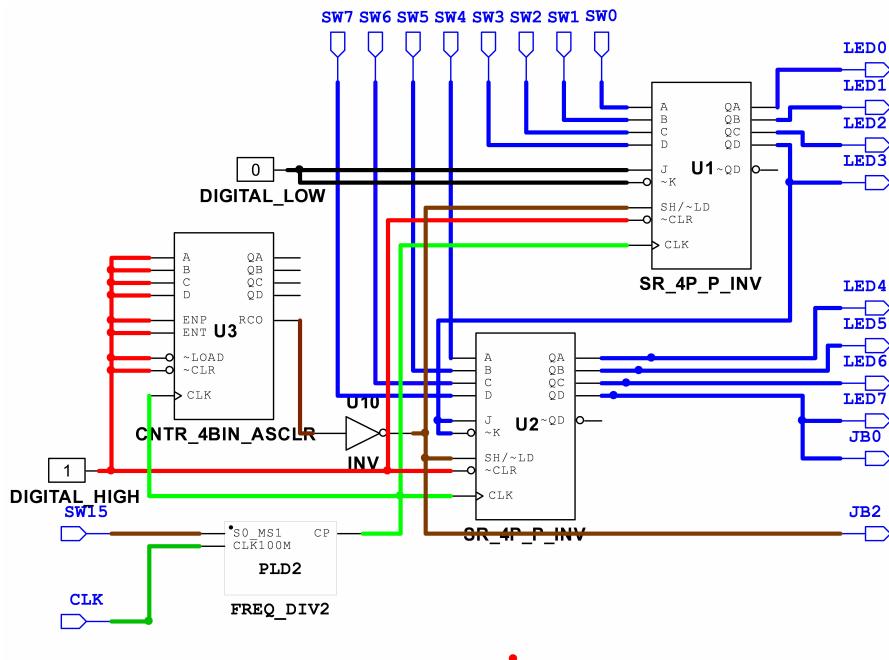


图 4.1.1: 移位寄存器单源周期触发测试电路

4.1.1 实验结果

使用示波器测量引脚 JB0 与 JB2 的波形图4.1.2所示：

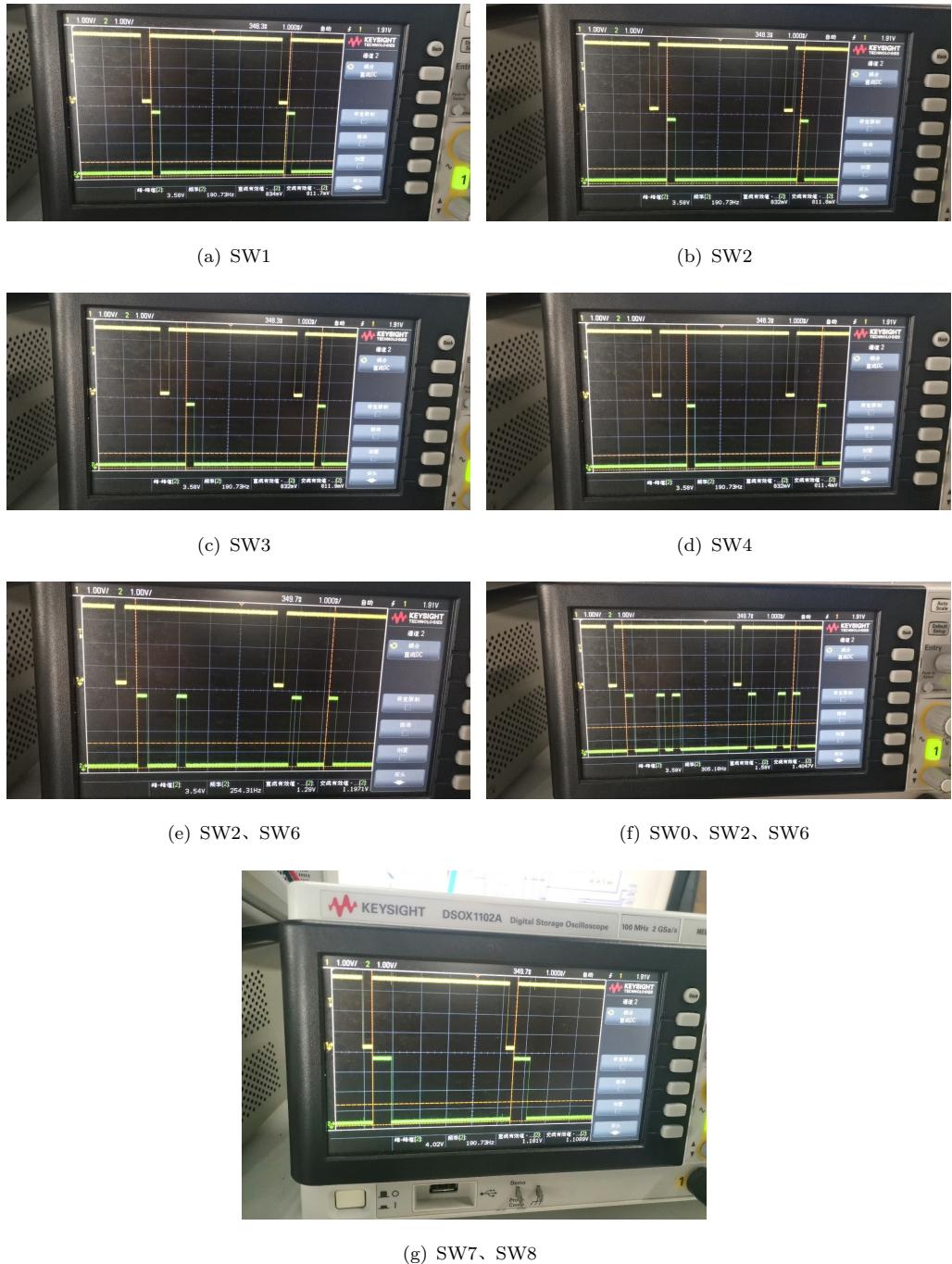


图 4.1.2: 移位寄存器单源周期触发测试-输出波形

可以看到在图 (a) – (d) 中，随着开关位置的变化，移位寄存器最高位的输出波形（通道 2，绿色）随之相较移位寄存器置数信号位置依次向后移动，反映了移位寄存器数据移位

的功能特点。在多输入时，可以看到移位寄存器将并行输入转换成为了串行输出。图(g)中的输入数据为我的学号。

切换时钟频率到秒分频，观察 LED 灯光情况，测试视频已随邮件附件发送，分别为“1s-1.mp4”，“1s-2.mp4”。

4.2 移位寄存器单次触发测试电路

自己搭建实验电路图如图4.2.1所示。电路搭建完成后，生成 bit 文件并下载至开发板上，验证试验现象。

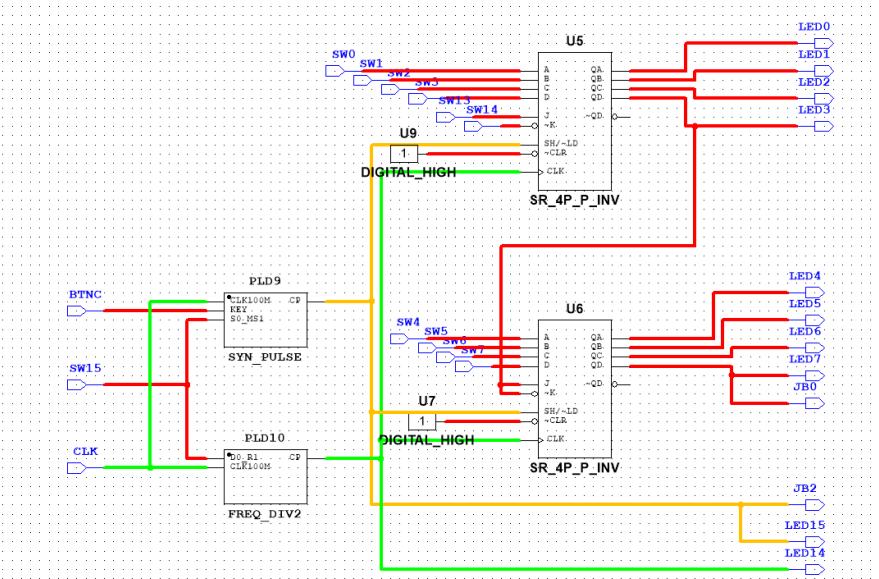


图 4.2.1: 移位寄存器单次触发测试电路

测试引脚 JB2（通道 1，绿色）、JB0（通道 1，黄色）的波形如图4.2.2所示。可以看到，无移位输入信号时，JB0 的波形混乱，怀疑为电路工作频率过高所致电路行为不正常，因此更改分频电路模块，降低电路工作频率，更改的分频电路模块如图4.2.3所示，更改的部分用浅蓝色连线标明。

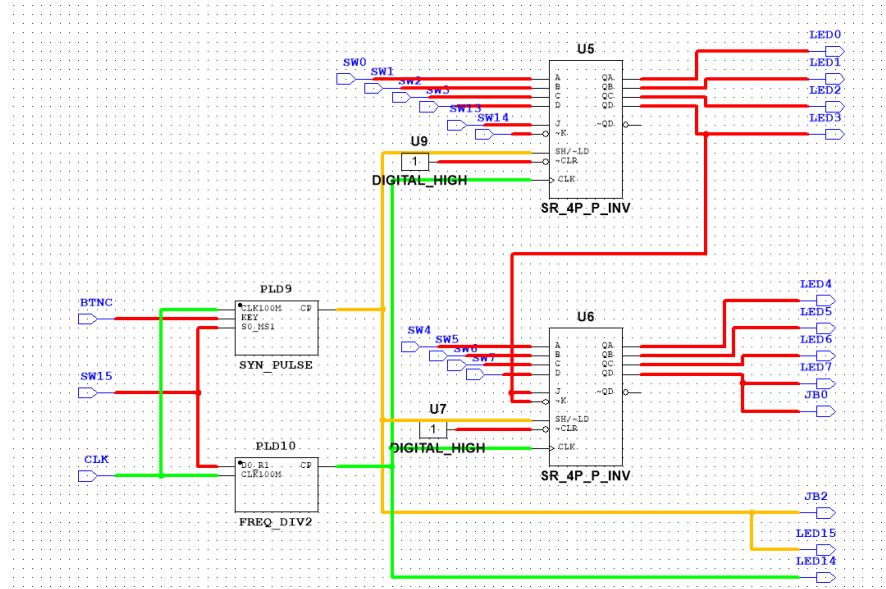


图 4.2.2: 原电路实验波形

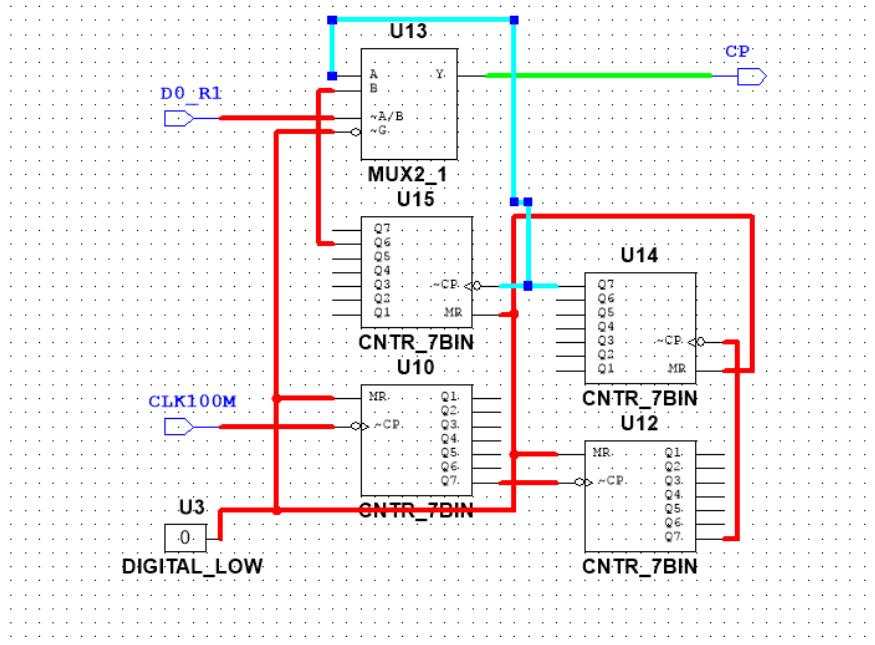


图 4.2.3: 分频电路更改

更改后电路波形正常，为验证 JK 触发器在不同 J、K 输入条件下的置数情况，调整 J、

K 的输入值，测试 JB2（通道 1，绿色）、JB0（通道 1，黄色）的波形如图4.2.4所示。

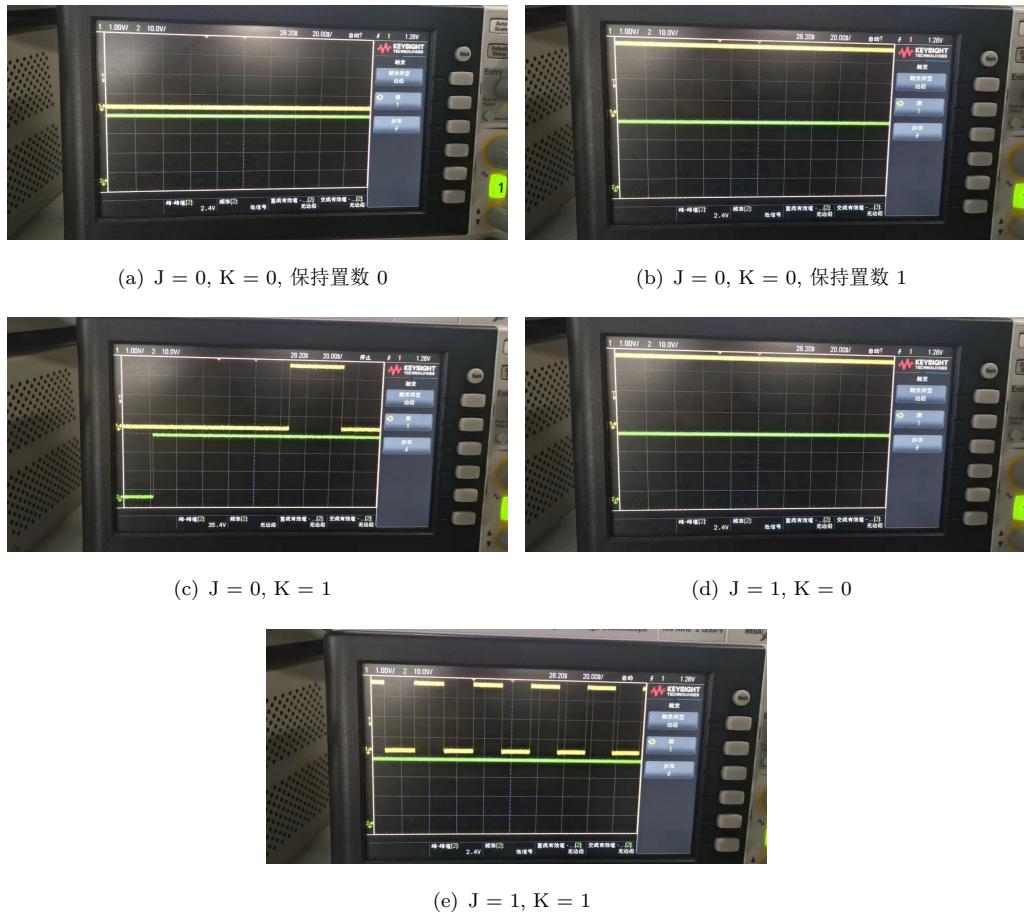


图 4.2.4: 移位寄存器单次触发测试电路-输出波形

可以看到， $J = 0, K = 0$ 时，触发器次态与现态相同，JK 触发器工作在保持状态； $J = 0, K = 1$ 时，触发器次态为 0，为置零工作模式； $J = 1, K = 0$ 时，触发器次态为 1，为置一工作模式； $J = 1, K = 1$ 时，触发器次态与现态相反，触发器在 0、1 之间不停翻转。总而言之，通过实验验证了 JK 触发器在不同输入状态下的工作情况，与理论相符。

切换时钟频率到秒分频，观察 LED 灯光情况，测试视频已随邮件附件发送，为“s-1.mp4”。

4.3 数据收发器

利用移位寄存器构成数据收发器，电路原型如图4.3.1所示。

其中发送端利用计数器结合按键时间，产生随机数，随后送到触发器中进行显示，并连接到并入串出移位寄存器，待发送信号后将数据发出。发送数据的第一位恒为高电平，用于触发接收端读取报文数据并进行处理显示。

接收端接受 5 位数据，其中第一位为数据报头，用于触发数据接收过程。当串入并出移位寄存器第 5 位为位高电平时，D 锁存器被使能，读取低 4 位数据信息，并在移位寄存器清零后保存数据信息，送至下级电路进行数据显示。

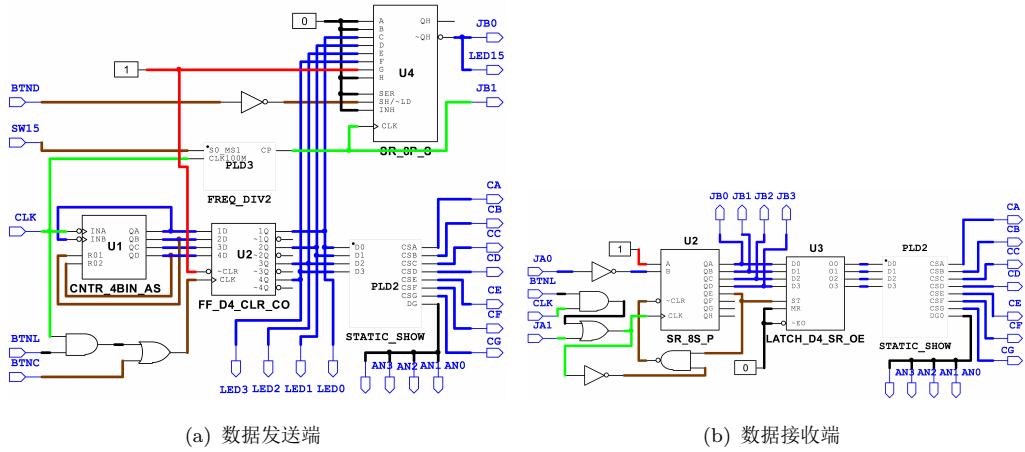


图 4.3.1: 数据收发器-原型电路

4.3.1 数据发送端电路搭建

以下所有电路搭建过程均由 verilog + Vivado 进行实现。

首先，电路所需的各个模块：

4.3.1.1 十分频模块

```

1 module ten(
2     input clock,
3     output ten
4 );
5 reg [3:0] Q;
6 always @(posedge clock) begin
7     if (Q <= 8) begin
8         Q = Q + 1;
9     end
10    else
11        Q = 0;
12    end
13    assign ten = Q[3];
14 endmodule

```

4.3.1.2 分频电路模块

利用刚才搭建的十分频模块进行级联，即可得到不同频率的时钟信号，将所有得到的信号送至输出端口 CP，以备后级电路使用。

```

1 module freq_div(
2     input clock,
3     output [10:1] CP
4 );
5
6     ten div1(clock, CP[1]);
7     ten div2(CP[1], CP[2]);
8     ten div3(CP[2], CP[3]);
9     ten div4(CP[3], CP[4]);
10    ten div5(CP[4], CP[5]);
11    ten div6(CP[5], CP[6]);
12    ten div7(CP[6], CP[7]);
13    ten div8(CP[7], CP[8]);
14    ten div9(CP[8], CP[9]);
15    ten div10(CP[9], CP[10]);
16
17 endmodule

```

4.3.1.3 数据选择器

用于选择时钟频率，使电路工作在不同频率下。

```

1 module mux2to1(
2     input s,
3     input [1:0] data,
4     output reg out
5 );
6     always @(data or s) begin
7         out = s ? data[1]:data[0];
8     end
9 endmodule

```

4.3.1.4 十进制计数器

用于配合时钟信号，产生随机数。

```

1 module counter10(
2     input clk,
3     input reset,
4     output [3:0] Q
5 );

```

```

6      reg [3:0]number;
7      always @ (posedge clk or posedge reset) begin
8          if (reset) begin
9              number = 0;
10         end
11         else if (number < 9) begin
12             number = number+1;
13         end
14         else
15             number = 0;
16     end
17     assign Q = number;
18 endmodule

```

4.3.1.5 4 位 D 触发器

用于配合十进制计数器，保存计数器产生的随机数准备发送。

```

1 module FF_D4(
2     input clk,
3     input [3:0] D,
4     output [3:0] Q
5 );
6     reg [3:0]W;
7     always @ (posedge clk) begin
8         W = D;
9     end
10    assign Q = W;
11 endmodule

```

4.3.1.6 4 位 D 触发器

用于配合十进制计数器，保存计数器产生的随机数准备发送。

```

1 module FF_D4(
2     input clk,
3     input [3:0] D,
4     output [3:0] Q
5 );
6     reg [3:0]W;
7     always @ (posedge clk) begin
8         W = D;
9     end
10    assign Q = W;

```

```
11 endmodule
```

4.3.1.7 8 位并入串出移位寄存器

用于 D 触发器，在数据发送信号到来时进行数据发送。

```
1 module SR_8P(
2     input [7:0] DataIn,
3     input clk,
4     inout load,
5     output reg Q
6 );
7
8     reg [7:0] ShiftData;
9     always @(posedge clk) begin
10         if (load) begin
11             ShiftData <= DataIn;
12             Q <= ShiftData[7];
13         end
14         else begin
15             ShiftData <= {ShiftData[6:0], 1'b0};
16             Q <= ShiftData[7];
17         end
18     end
19 endmodule
```

4.3.1.8 8 位串入并移位寄存器

连接 8 位并入串出移位寄存器输出端，并将并行输出连接至 LED 进行显示，便于观察电路工作状态。

```
1 module SR_NS_C(Data, clk, clr, Q);
2     parameter n = 8;
3     input Data;
4     input clk, clr;
5     output reg [n-1:0] Q;
6
7     always @(posedge clk) begin
8         if(clr)
9             Q = 0;
10        else
11            Q = {Q[n-2:0], Data};
12    end
13 endmodule
```

4.3.1.9 静态显示模块

由 BCD-数码管译码器以及相应数码管连接电路构成，用于将输入数据显示到数码管上。

```
1 module BCD_dec(
2     input [3:0]D,
3     output reg [6:0]Q
4 );
5
6     always @ (D) begin
7         case(D)
8             0: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b0};
9             1: Q = {1'b0, 1'b1, 1'b1, 1'b0, 1'b0, 1'b0, 1'b0};
10            2: Q = {1'b1, 1'b1, 1'b0, 1'b1, 1'b1, 1'b0, 1'b1};
11            3: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b0, 1'b0, 1'b1};
12            4: Q = {1'b0, 1'b1, 1'b1, 1'b0, 1'b0, 1'b1, 1'b1};
13            5: Q = {1'b1, 1'b0, 1'b1, 1'b0, 1'b1, 1'b1, 1'b1};
14            6: Q = {1'b1, 1'b0, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1};
15            7: Q = {1'b1, 1'b1, 1'b1, 1'b0, 1'b0, 1'b0, 1'b0};
16            8: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1};
17            9: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b0, 1'b1, 1'b1};
18            default : Q = 0;
19        endcase
20    end
21 endmodule
22
23 module STATIC_SHOW (
24     input [3:0]D,
25     output [6:0]LED_digit,
26     output [3:0]enable
27 );
28
29     wire [6:0]BCDcode;
30     assign enable = 0;
31
32     BCD_dec BCDdec(D, BCDcode);
33     assign LED_digit = ~BCDcode;
34
35 endmodule
```

4.3.1.10 主电路模块

将刚才构建的各个电路模块进行级联，构成完整的数据发送端电路。

```
1 module main(
2     input BTND, BTNL, BTNC,
3     input CLK,
4     input SW15,
5     output LED0, LED1, LED2, LED3, LED15,
6     output LED14, //clkUsed
7     output LED4, LED5, LED6, LED7, LED8, LED9, LED10, LED11, //for testing
8     output JB0, JB1,
9     output CA, CB, CC, CD, CE, CF, CG, AN3, AN2, AN1, AN0
10 );
11
12 wire [3:0]CTN10out;
13 wire triggerData;
14 wire [3:0]data; // def MSB on the left hand side!!!
15
16 // getting data to be transmitted
17 assign triggerData = (BTNL && CLK) || BTNC;
18 counter10 CTN10(CLK, 1'b0, CTN10out);
19 FF_D4 D4_1(triggerData, CTN10out, data);
20
21 //showing data
22 assign {LED3,LED2,LED1,LED0} = data;
23 STATIC_SHOW sta_show1(data,{CA, CB, CC, CD, CE, CF, CG}, {AN3, AN2, AN1, AN0});
24
25 //getting low freq clk signal
26 wire [10:1]clkAll;
27 wire [1:0]clkOpt;
28 wire clkUsed;
29 freq_div f1(CLK, clkAll);
30 assign clkOpt = {clkAll[5], clkAll[8]};
31 mux2to1 M1(SW15, clkOpt,clkUsed);
32 assign JB1 = clkUsed;
33 assign LED14 = clkUsed;
34
35 //transmitting data
36 wire sendOut;
37 SR_8P SR1({1'b0, 1'b1, data, 1'b0, 1'b0}, clkUsed, BTND, sendOut);
38 assign JB0 = sendOut;
39 assign LED15 = sendOut;
40
41 // for testing
42 SR_NS_C SR2(sendOut, clkUsed, 1'b0, {LED11, LED10, LED9, LED8, LED7, LED6, LED5, LED4});
```

43

44 endmodule

4.3.2 数据接收端电路搭建

数据接收端的电路原型存在问题，详细叙述如下：

原型电路中，接收端电路 D 锁存器使能端与移位寄存器输出第 5 位相连，为异步工作模式。当移位寄存器清零时，第 5 位与第 4 位“同时”清零，“同时”，D 锁存器使能端归零，D 锁存器不再接受数据。

然而，实际电路中存在信号传播时间问题，移位寄存器输出置零顺序存在竞争。由 Vivadate 综合出的电路移位寄存器输出第 5 位晚于低 4 位置零，使得 D 锁存器在移位寄存器归零后仍然读取并保存数据，使得正常数据无法保存，稳定时 D 锁存器存储数字为'0000'。

为解决信号传播时间问题，将 D 锁存器的使能端由异步工作模式更换为同步工作模式，引入时钟信号，使得锁存器在数据清零前停止读取数据，解决这个问题。

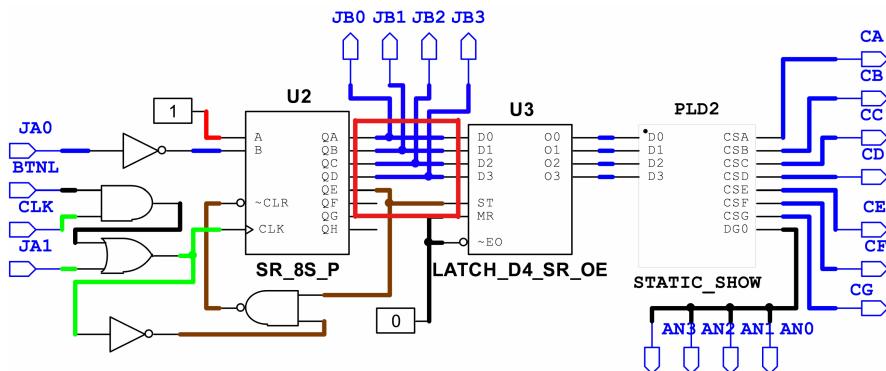


图 4.3.2: 接收端电路问题

此外，更改了原型电路，将数据显示模块改为动态显示模块，使得数码管 4 位可以显示 4 个不同的数字。同时搭建 16 位移位寄存器，存储 4 位 10 进制数所需的 16 位 BCD 码。在有数据到来时，以 4 位为整体进行移位，达到显示移位的效果。

4.3.2.1 分频电路模块

数据接收端本身不需要此模块，引入此模块用于单板测试之用。

```

1 module ten(
2     input clock,
3     output ten
4 );
5     reg [3:0]Q;

```

```
6      always @ (posedge clock) begin
7          if (Q <= 8) begin
8              Q = Q + 1;
9          end
10         else
11             Q = 0;
12     end
13     assign ten = Q[3];
14 endmodule
15
16 module freq_div(
17     input  clock,
18     output [10:1]CP
19 );
20
21     ten div1(clock, CP[1]);
22     ten div2(CP[1], CP[2]);
23     ten div3(CP[2], CP[3]);
24     ten div4(CP[3], CP[4]);
25     ten div5(CP[4], CP[5]);
26     ten div6(CP[5], CP[6]);
27     ten div7(CP[6], CP[7]);
28     ten div8(CP[7], CP[8]);
29     ten div9(CP[8], CP[9]);
30     ten div10(CP[9], CP[10]);
31
32 endmodule
```

4.3.2.2 分频电路模块

数据接收端本身不需要此模块，引入此模块用于单板测试之用。

```
1 module ten(
2     input  clock,
3     output ten
4 );
5     reg [3:0]Q;
6     always @ (posedge clock) begin
7         if (Q <= 8) begin
8             Q = Q + 1;
9         end
10        else
11            Q = 0;
```

```

12      end
13      assign ten = Q[3];
14 endmodule
15
16 module freq_div(
17     input clock,
18     output [10:1] CP
19 );
20
21     ten div1(clock, CP[1]);
22     ten div2(CP[1], CP[2]);
23     ten div3(CP[2], CP[3]);
24     ten div4(CP[3], CP[4]);
25     ten div5(CP[4], CP[5]);
26     ten div6(CP[5], CP[6]);
27     ten div7(CP[6], CP[7]);
28     ten div8(CP[7], CP[8]);
29     ten div9(CP[8], CP[9]);
30     ten div10(CP[9], CP[10]);
31
32 endmodule
33
34 module mux2to1( // 用于选择时钟信号来源
35     input s,
36     input [1:0] data,
37     output reg out
38 );
39     always @ (data or s) begin
40         out = s ? data[1]:data[0];
41     end
42 endmodule

```

4.3.2.3 8 位串入并出移位寄存器

用于接收数据发送端送来数据

```

1 module SR_NS_C(Data, clk, clr, Q);
2     parameter n = 8;
3     input Data;
4     input clk, clr;
5     output reg [n-1:0] Q;
6
7     always @ (posedge clk or posedge clr) begin

```

```

8      if (clr) begin
9          Q <= 0;
10         end
11     else
12         Q <= {Q[n-2:0], Data};
13     end
14
15 endmodule

```

4.3.2.4 8 位串入并出移位寄存器

用于接收数据发送端送来的数据。

```

1 module SR_NS_C(Data, clk, clr, Q);
2     parameter n = 8;
3     input Data;
4     input clk, clr;
5     output reg [n-1:0] Q;
6
7     always @ (posedge clk or posedge clr) begin
8         if (clr) begin
9             Q <= 0;
10        end
11     else
12         Q <= {Q[n-2:0], Data};
13     end
14
15 endmodule

```

4.3.2.5 D 锁存器

用于检测数据到达情况并读取、保存到达的数据。

注意，为了解决上文所述的信号竞争问题，引入了时钟信号，做同步使能。

```

1 module DLatchN(Data, set, clk, Q);
2     parameter n = 4;
3     input [n-1:0] Data;
4     input set,clk;
5     output reg [n-1:0] Q;
6
7     always @ (negedge clk) begin
8         if (set) begin
9             Q = Data;

```

```

10      end
11    end
12 endmodule

```

4.3.2.6 16 位 BCD 码移位寄存器

用于保存 4 位十进制数，送至动态显示模块进行输出。

```

1 module SR_16_4BIT_BLOCK(
2     input wire [3:0] data,
3     input wire en,
4     input clk,
5     output reg [15:0] Q
6 );
7
8     always @ (negedge clk) begin
9         if(en)
10             Q = {Q[11:0], data};
11     end
12 endmodule

```

4.3.2.7 动态显示模块

用于显示 4 位不相同的十进制数。

```

1 module dyn_show(
2     input wire [15:0] Data,
3     input wire clk,
4     output wire [6:0] LED,
5     output reg [3:0] LED_en
6 );
7
8     reg [1:0] state;
9     reg [3:0] selected;
10
11     BCD_dec BCD(selected, LED);
12
13     always @ (posedge clk) begin
14         case (state)
15             0: {selected, LED_en} <= {Data[3:0], 4'b1110};
16             1: {selected, LED_en} <= {Data[7:4], 4'b1101};
17             2: {selected, LED_en} <= {Data[11:8], 4'b1011};
18             3: {selected, LED_en} <= {Data[15:12], 4'b0111};
19         endcase

```

```

20         state <= state+1;
21     end
22 endmodule
23
24 module BCD_dec(
25     input [3:0]D,
26     output reg [6:0]Q
27 );
28
29     always @ (D) begin
30         case (D)
31             0: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b0};
32             1: Q = {1'b0, 1'b1, 1'b1, 1'b0, 1'b0, 1'b0, 1'b0};
33             2: Q = {1'b1, 1'b1, 1'b0, 1'b1, 1'b1, 1'b0, 1'b1};
34             3: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b0, 1'b0, 1'b1};
35             4: Q = {1'b0, 1'b1, 1'b1, 1'b0, 1'b0, 1'b1, 1'b1};
36             5: Q = {1'b1, 1'b0, 1'b1, 1'b1, 1'b0, 1'b1, 1'b1};
37             6: Q = {1'b1, 1'b0, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1};
38             7: Q = {1'b1, 1'b1, 1'b1, 1'b0, 1'b0, 1'b0, 1'b0};
39             8: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1, 1'b1};
40             9: Q = {1'b1, 1'b1, 1'b1, 1'b1, 1'b0, 1'b1, 1'b1};
41             default : Q = 0;
42         endcase
43     end
44 endmodule

```

4.3.2.8 主电路模块

连接以上各模块，形成完整的接收端电路。

```

1 module main(
2     input JA0, // sender data input
3     input BTNL,BTNU,//BTNU for testing
4     input CLK,
5     input JA1,
6     input SW15, // for selecting clk signal for testing
7
8     output CA, CB, CC, CD, CE, CF, CG, AN0, AN1, AN2, AN3,
9     output JB0, JB1, JB2, JB3,
10    output LED4, LED3, LED2, LED1, LED0, // for SR1out[3:0]
11    output LED8, LED7, LED6, LED5, //for DL1out
12    output LED15 // for watching clk signal
13 );

```

```
14
15     parameter nbit = 8;
16
17     //getiing clk signal
18     wire psuedoCLK;
19     assign psuedoCLK = (BTNL && CLK) || JA1;
20     wire [10:1]clkAll;
21     wire clk0pt;
22     wire clkUsed;
23     freq_div f1(CLK, clkAll);
24     assign clk0pt = clkAll[8];
25     mux2to1 M1(SW15, {clk0pt, psuedoCLK} ,clkUsed);
26     assign LED15 = clkUsed;
27
28
29     // get signal and trigger storage
30     // SR1out[4] is for triggering data storage and reset, SR1out[3:0] is valid data signal.
31     wire SR1clr;
32     wire [nbit-1:0]SR1out;
33     wire dataIn;
34     assign dataIn = JA0 || BTNU;
35     SR_NS_C #(.n(nbit))R1(dataIn, psuedoCLK, SR1clr ,SR1out);
36
37     assign SR1clr = SR1out[4] && (~clkUsed);
38     assign {JB3, JB2, JB1, JB0} = SR1out[3:0];
39     assign {LED4, LED3, LED2, LED1, LED0} = SR1out[4:0];
40
41     //storing data
42     wire [3:0]DL1out;
43     DLatchN #(.n(4))DL1(SR1out[3:0], SR1out[4], clkUsed, DL1out);
44
45     // dyn show data
46
47     wire [15:0] dynLED;
48     SR_16_4BIT_BLOCK SR_dyn_out(DL1out, SR1out[4], clkUsed, dynLED);
49     dyn_show dyn(dynLED, clkAll[3], {CA,CB,CC,CD,CE,CF,CG}, {AN3, AN2, AN1, AN0});
50
51
52     //showing data
53     //STATIC_SHOW staticShow(DL1out, {CA,CB,CC,CD,CE,CF,CG}, {AN0,AN1,AN2,AN3});
54     //assign {LED8, LED7, LED6, LED5} = DL1out;
```

56

57

58 endmodule

4.3.3 综合、编译、下载至开发板进行实验

综合出的发送端、接收端实验电路如图4.3.3、图4.3.4所示。

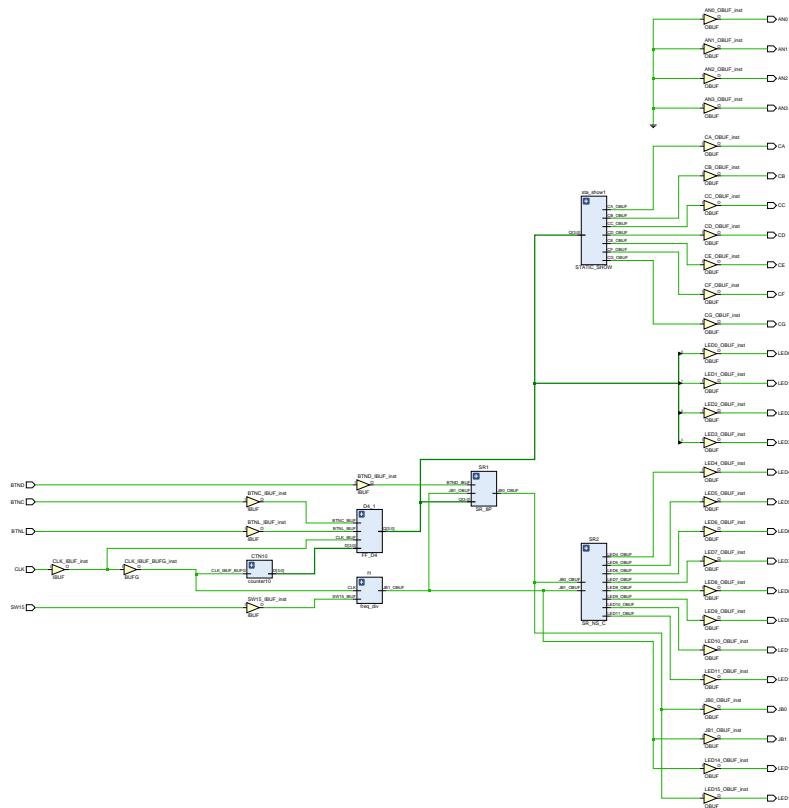


图 4.3.3: 发送端 Vivado 综合电路

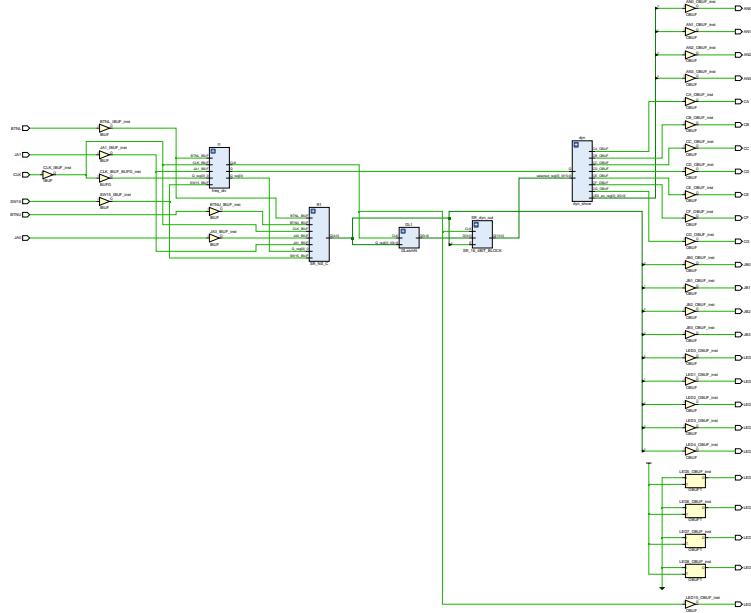
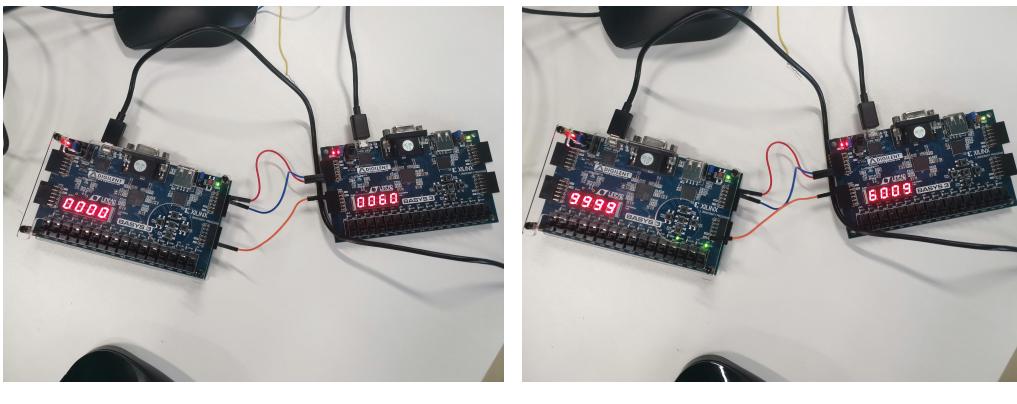


图 4.3.4: 接收端 Vivado 综合电路

下载到开发板，观察实验现象，可以看到从发送端发送的数据可以被接收端正接收，如图4.3.5所示。可以看到，先接收到的数据再后续数据到达后被移位至 BCD 高位显示，实现了预定功能。



(a) 输入 6, 0

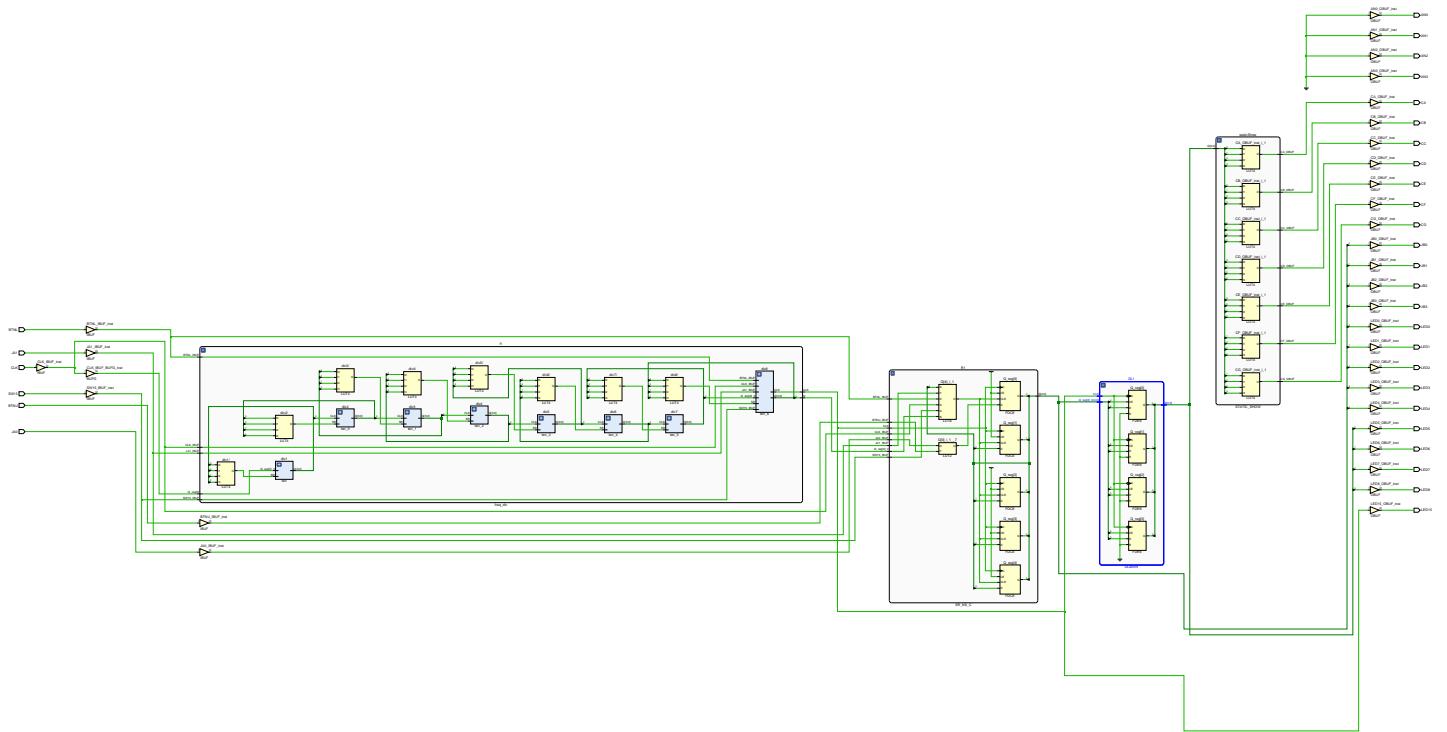
(b) 继续输入 0、9

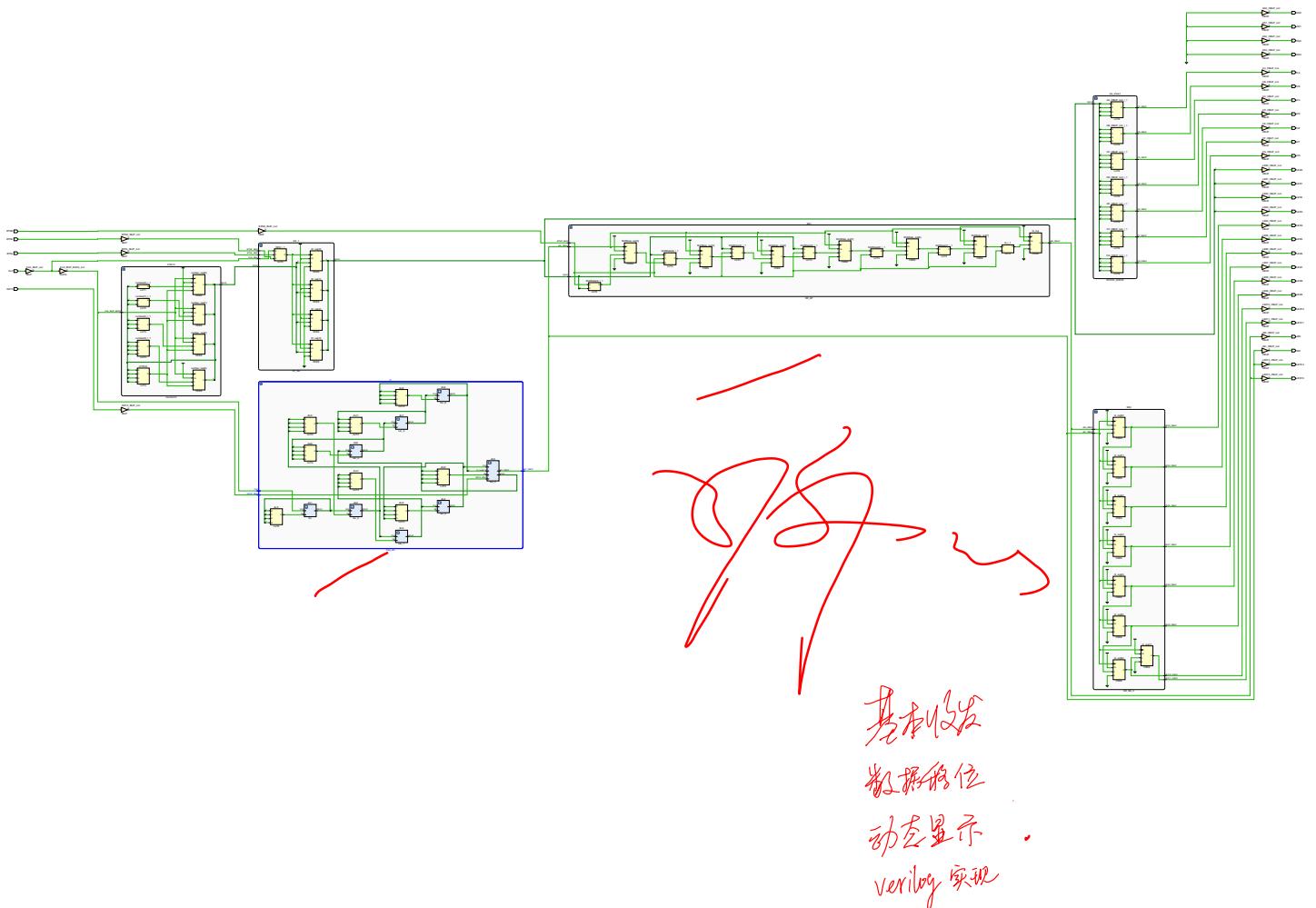
图 4.3.5: 数据收发器-实际测试

5 实验总结

- a. 了解了移位寄存器工作原理，验证了移位寄存器功能
- b. 验证了 JK 触发器的工作状态以及相应功能
- c. 利用移位寄存器，结合 verilog 以及 vivado 实现了数据收发器

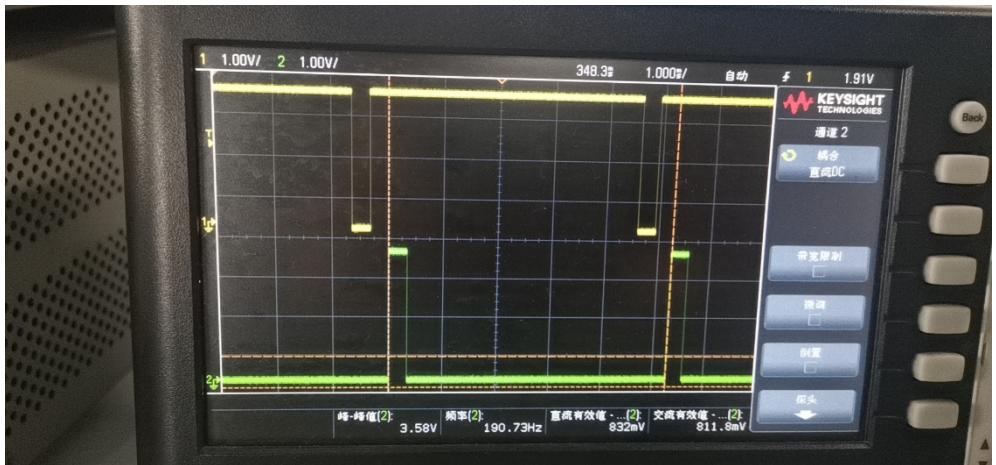
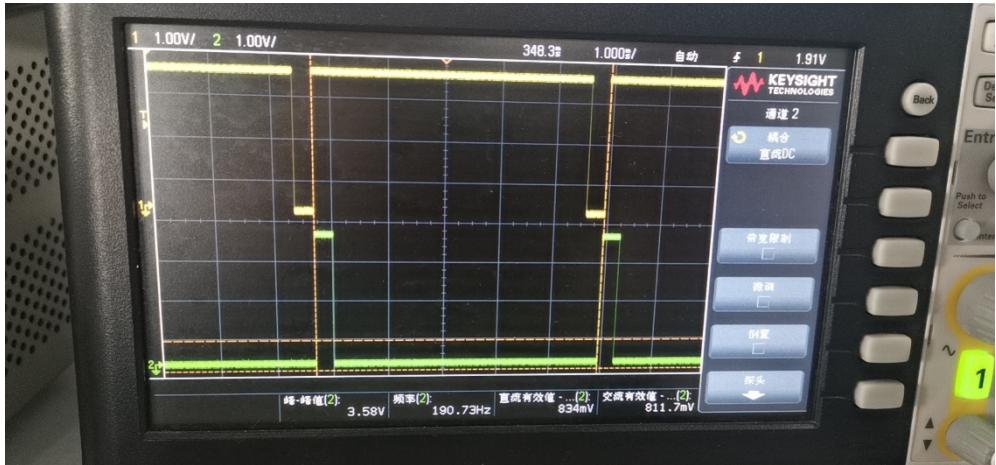
原始数据

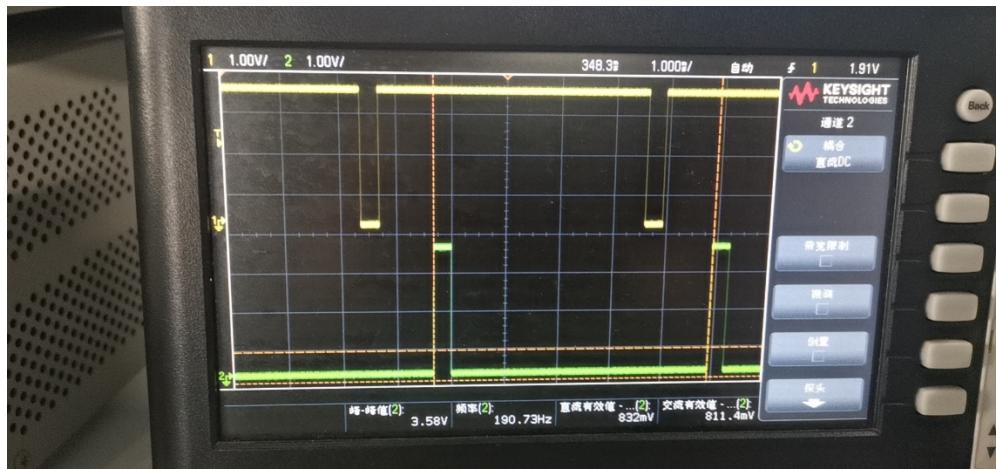




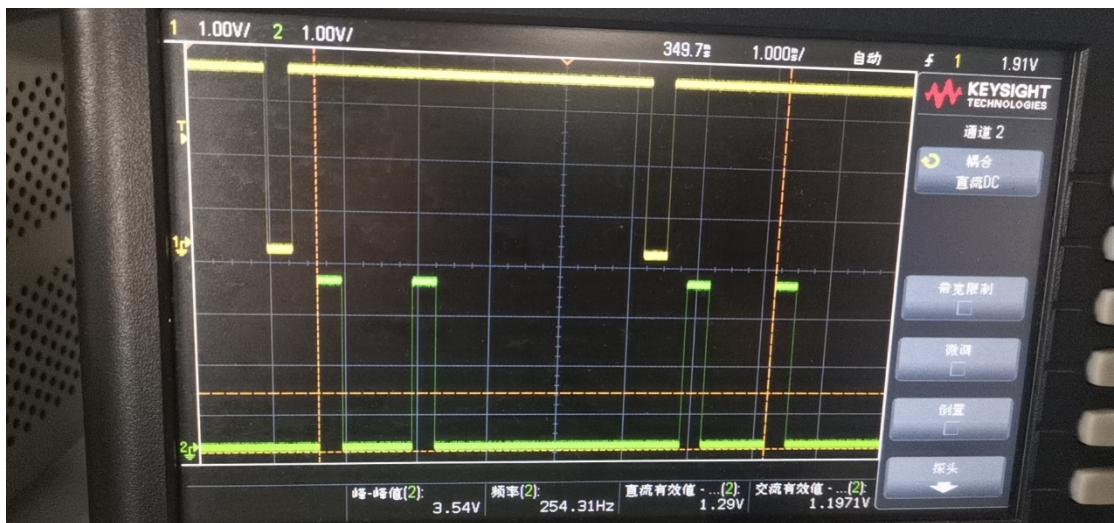
- 1S-V2

- CH1 (绿): JB2 CH2 (黄): JB0

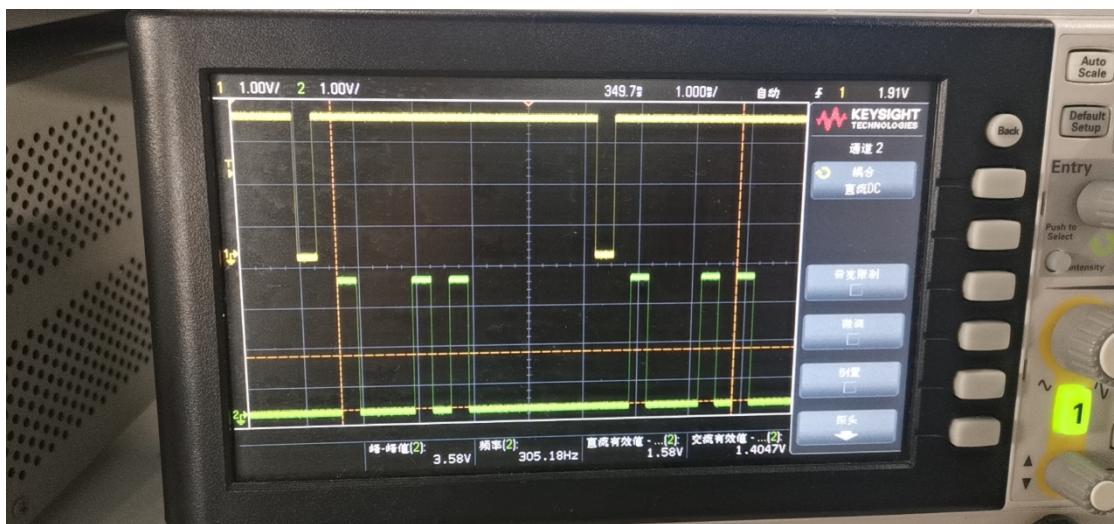




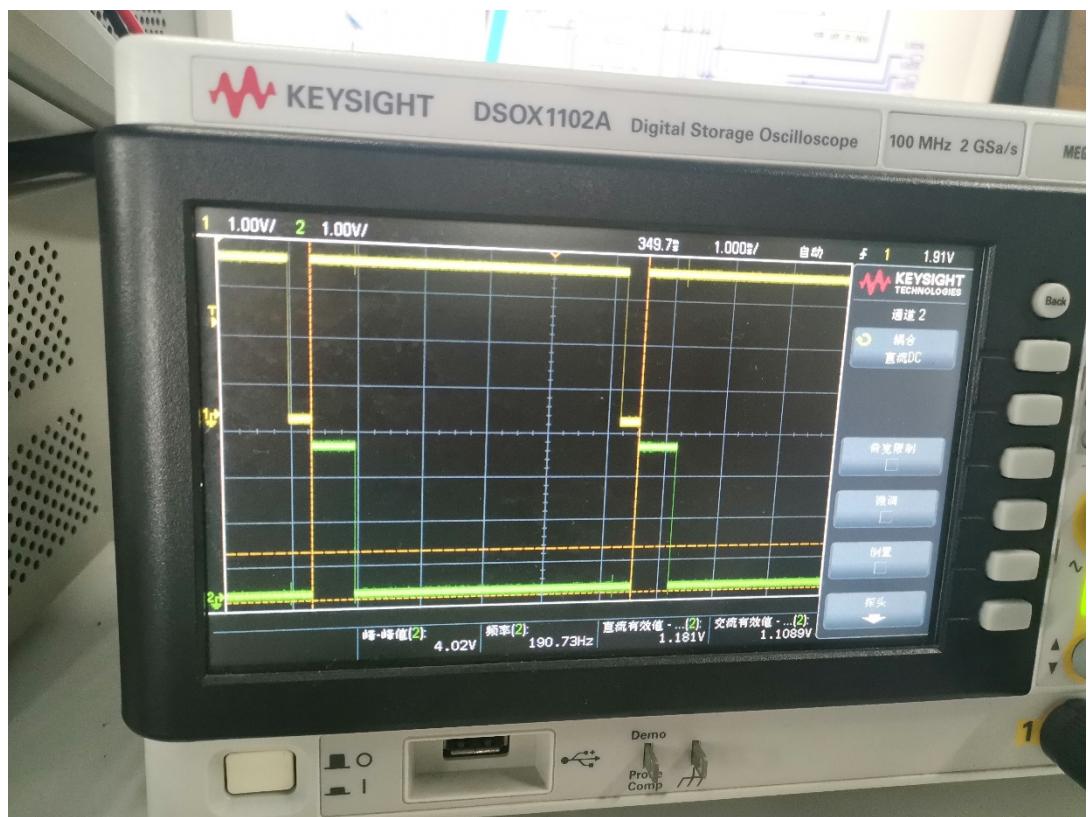
■ 2、6



■ 0、2、6



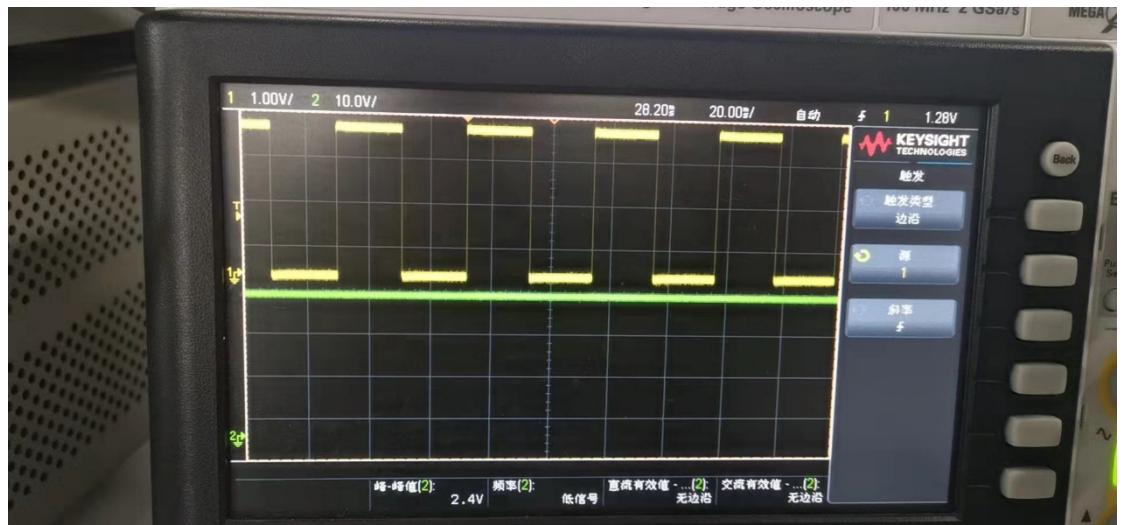
■ 学号 78, 使用 SW6, SW7



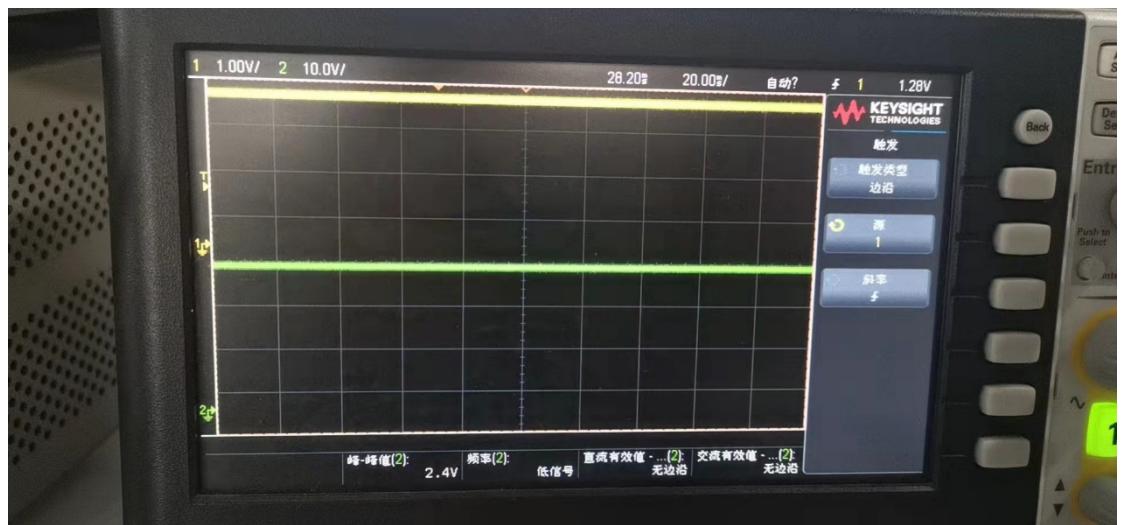
- S-V2
 - CH1 (绿): JB2 CH2 (黄): JB0
 - J=~-k=0 置零, 置数后移位



- ◆ ■ J=1, ~k = 0 : 不停翻转



■ $j = -k = 1$ 置 1



■ $j=0, -k=1$ 保持 可以 0 可以 1

