浙江大学

本科实验报告

课程名称：	数字逻辑电路设计
姓　　名：	
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
邮　　箱：	
QQ　号：	
电　　话：	
指导教师：	洪奇军
报告日期：	2020 年 12 月 15 日

浙江大学实验报告

课程名称：数字逻辑设计 实验类型：综合

实验项目名称：实验十二——移位寄存器设计与应用

学生姓名：_____ 学号： 同组学生姓名：_____

实验地点：紫金港东四 509 室 实验日期：2020 年 12 月 15 日

一、实验目的

1.1 掌握移位寄存器的工作原理及设计方法
1.2 掌握串、并数据转换的概念与方法
1.3 掌握同步串行传输方法

二、实验内容和原理

2.1 实验内容

- 结构化描述方法设计四位带并行输入的双向移位寄存器；
- 设计 32 位双向移位寄存器加入 DSTE 测试环境测试；
- 设计通用位宽的双向移位寄存器，并实现同步串行(并转串)数据传输电路 SST；

2.2 实验原理

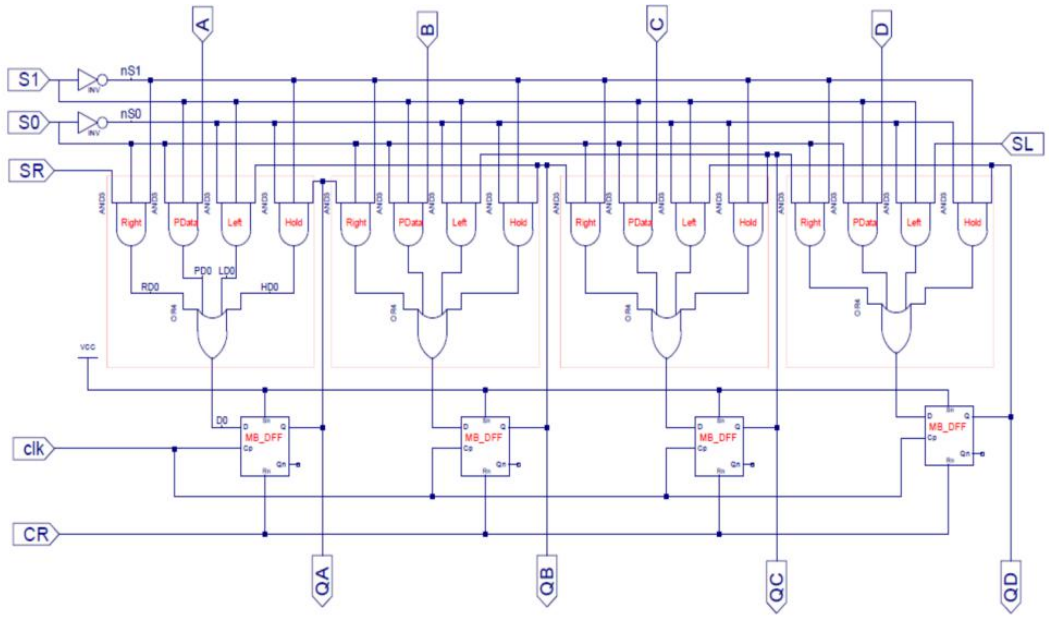
2.2.1 移位寄存器

移位寄存器在时钟脉冲到来时，其中数据按左移或右移的方式移动一位。由于这种移位按周期进行，移位寄存器必须采用主从触发器或边沿触发器。

移位寄存器的输入输出有串行与并行两种方式。

2.2.2 双向通用移位寄存器

DM/SN74LS194 是一种双向通用移位寄存器，可以用于并串转换。可以根据原理图采用 Verilog 描述双向通用移位寄存器。

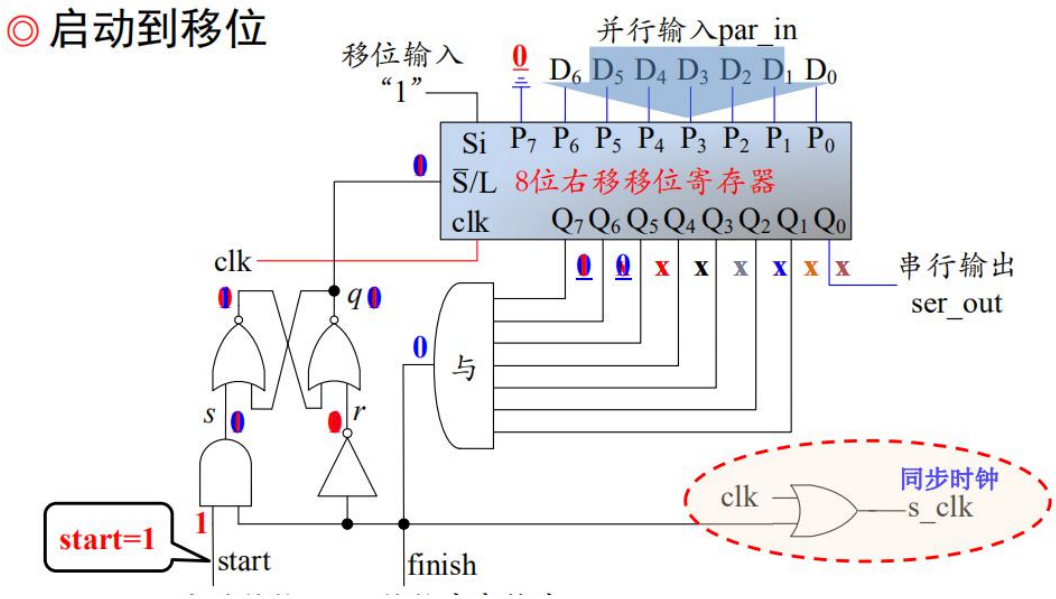


图表 2.2.2-1 双向通用移位寄存器原理图

2.2.3 同步串行传输：并-串转换器

当没有启动命令（低电平）时，电路上电后经过若干个时钟脉冲后将会稳定在 RS 触发器输出 q＝0，移位寄存器 Q7-Q0＝11111111 的状态。

当启动命令（高电平）加至启动输入端时，RS 触发器的输出端 q 被置 1，7 位并行数据及标志码“0”在第一个 clk 的作用下同时置入移位寄存器。在移位结束之前，由于标志码仍在 Q7-Q1 中，finish 信号一直为 0，保证了移位的进行。而当其到达 Q0，Q7-Q1 全为 1，finish 信号为 1，移位结束。



图表 2.2.3-1 并-串转换器原理图

三、操作方法与实验步骤

3.1 实验设备与材料

1. 装有 ISE 14.7 的计算机　　1 台
2. SWORD 开发板　　1 套

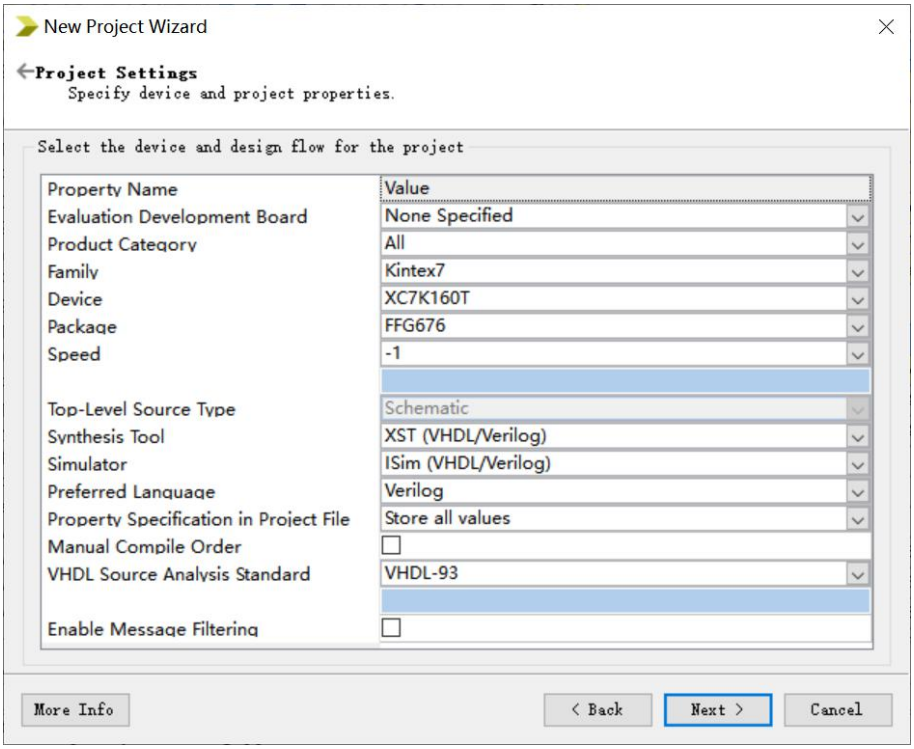
3.2 实验步骤

3.2.1 设计实现 4 位通用移位寄存器 DM74LS194

1. 建立工程

打开 ISE Design Suite 14.7，在左上角点击 File，再点击 New Project 命名为 Exp12-Shift。将工程命名，选择 Top-level source type 为 HDL 并选择项目保存位置。

点击 Next 后，出现如图表 3.2.1-1 所示的对话框，如图表所示选择对应的 Family、Device、Package、Speed 等属性，确认无误后一直点击 Next 直至创建工程结束。



图表 3.2.1-1 项目配置

2. 设计 8 位寄存器

在左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Verilog Module，输入文件名 SN74LS194，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。

此后，双击打开左侧 Sources 窗口中的 SN74LS194.v 文件，调用之前实验的 MB_DFF，输入代码：

```
`timescale 1ns / 1ps

module SN74LS194(input S1, S0, SR, SL, clk, CR, A, B, C, D,
                 output QA,
                 output QB,
                 output QC,
                 output QD
    );

    INV INV0(.I(S0), .O(nS0)),
        INV1(.I(S1), .O(nS1));

    MB_DFF ShiftA(.Cp(clk), .D(DA), .Rn(CR), .Sn(1'b1), .Q(QA), .Qn()),
           ShiftB(.Cp(clk), .D(DB), .Rn(CR), .Sn(1'b1), .Q(QB), .Qn()),
           ShiftC(.Cp(clk), .D(DC), .Rn(CR), .Sn(1'b1), .Q(QC), .Qn()),
           ShiftD(.Cp(clk), .D(DD), .Rn(CR), .Sn(1'b1), .Q(QD), .Qn());

    OR4 ORA(.I0(HDA), .I1(RDA), .I2(LDA), .I3(PDA), .O(DA)),
        ORB(.I0(HDB), .I1(RDB), .I2(LDB), .I3(PDB), .O(DB)),
        ORC(.I0(HDC), .I1(RDC), .I2(LDCC), .I3(PDC), .O(DC)),
        ORD(.I0(HDD), .I1(RDD), .I2(LDD), .I3(PDD), .O(DD));

    AND3 HoldA(.I0(nS1), .I1(nS0), .I2(QA), .O(HDA)),
         HoldB(.I0(nS1), .I1(nS0), .I2(QB), .O(HDB)),
         HoldC(.I0(nS1), .I1(nS0), .I2(QC), .O(HDC)),
         HoldD(.I0(nS1), .I1(nS0), .I2(QD), .O(HDD));

    AND3 RightA(.I0(nS1), .I1(S0), .I2(SR), .O(RDA)),
         RightB(.I0(nS1), .I1(S0), .I2(QA), .O(RDB)),
         RightC(.I0(nS1), .I1(S0), .I2(QB), .O(RDC)),
         RightD(.I0(nS1), .I1(S0), .I2(QC), .O(RDD));

    AND3 LeftA(.I0(S1), .I1(nS0), .I2(QB), .O(LDA)),
         LeftB(.I0(S1), .I1(nS0), .I2(QC), .O(LDB)),
         LeftC(.I0(S1), .I1(nS0), .I2(QD), .O(LDCC)),
         LeftD(.I0(S1), .I1(nS0), .I2(SL), .O(LDD));

    AND3 PaA(.I0(S1), .I1(S0), .I2(A), .O(PDA)),
         PaB(.I0(S1), .I1(S0), .I2(B), .O(PDB)),
         PaC(.I0(S1), .I1(S0), .I2(C), .O(PDC)),
         PaD(.I0(S1), .I1(S0), .I2(D), .O(PDD));

endmodule
```

完成后，在 Source 窗口单击选中 SN74LS194.v 文件，然后双击下方 Process 窗口的 Synthesize 下的 Check Syntax 检查语法。

3. 仿真验证

完成后，进行仿真验证。仿真代码如下：

```
`timescale 1ns / 1ps

module SN_test;

    // Inputs
    reg S1;
    reg S0;
    reg SR;
```

```
    reg SL;
    reg clk;
    reg CR;
    reg A;
    reg B;
    reg C;
    reg D;

    // Outputs
    wire QA;
    wire QB;
    wire QC;
    wire QD;

    // Instantiate the Unit Under Test (UUT)
    SN74LS194 uut (
        .S1(S1),
        .S0(S0),
        .SR(SR),
        .SL(SL),
        .clk(clk),
        .CR(CR),
        .A(A),
        .B(B),
        .C(C),
        .D(D),
        .QA(QA),
        .QB(QB),
        .QC(QC),
        .QD(QD)
    );

    initial begin
        CR = 0;
        clk = 0;
        SL = 0;
        SR = 0;
        A = 0;
        B = 0;
        C = 0;
        D = 0;
        S1 = 0;
        S0 = 0;
        fork
            forever #20 clk<=~clk;
            #10 CR=1;
            begin
                // right
                {S1,S0}=2'b01;
                SR=1;
                #200; CR=0;
                #40;  CR=1;
                // left
                {S1,S0}=2'b10;
                SL=1;
                #170;
                // parallel
                {S1,S0}=2'b11;
                {A,B,C,D}=4'b1000;
                SL=0;
                SR=0;
```

```
            #50;
            // hold
            {S1,S0}=2'b00;
            #50;
            // right
            {S1,S0}=2'b01;
            #170;
            // parallel
            {S1,S0}=2'b11;
            {A,B,C,D}=4'b0001;
            #40;
            // left
            {S1,S0}=2'b10;
            #170;

        end
      join
  end
endmodule
```

3.2.2 设计实现 32 位通用移位寄存器

在 Exp12-Shift 工程下，左侧 Sources 窗口空白处右键菜单中选择 New Source。

在对话框中，选择源类型为 Verilog Module，输入文件名 shift_32，并且勾选下方 Add to Project。此后，一直点击 Next 直至结束。

此后，双击打开左侧 Sources 窗口中的 shift_32.v 文件，调用上一步完成的 SN74LS194，输入描述代码如下：

```
`timescale 1ns / 1ps

module shift_32(input clk,
                input clear,
                input S1,S0,SL,SR,
                input [31:0]PData,
                output [31:0]Q
    );

    wire CR=~clear;
    SN74LS194
SH0(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(SL), .SR(Q[4]),

      .D(PData[0]), .C(PData[1]), .B(PData[2]), .A(PData[3]),
                    .QD(Q[0]), .QC(Q[1]), .QB(Q[2]), .QA(Q[3]));

    SN74LS194
SH1(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[3]), .SR(Q[8]),

      .D(PData[4]), .C(PData[5]), .B(PData[6]), .A(PData[7]),
                    .QD(Q[4]), .QC(Q[5]), .QB(Q[6]), .QA(Q[7]));

    SN74LS194
SH2(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[7]), .SR(Q[12]),

      .D(PData[8]), .C(PData[9]), .B(PData[10]), .A(PData[11]),
                    .QD(Q[8]), .QC(Q[9]), .QB(Q[10]), .QA(Q[11]));

    SN74LS194
SH3(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[11]), .SR(Q[16]),
```

```
     .D(PData[12]), .C(PData[13]), .B(PData[14]), .A(PData[15]),
                    .QD(Q[12]), .QC(Q[13]), .QB(Q[14]), .QA(Q[15]));

    SN74LS194
SH4(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[15]), .SR(Q[20]),

     .D(PData[16]), .C(PData[17]), .B(PData[18]), .A(PData[19]),
                    .QD(Q[16]), .QC(Q[17]), .QB(Q[18]), .QA(Q[19]));

    SN74LS194
SH5(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[19]), .SR(Q[24]),

     .D(PData[20]), .C(PData[21]), .B(PData[22]), .A(PData[23]),
                    .QD(Q[20]), .QC(Q[21]), .QB(Q[22]), .QA(Q[23]));

    SN74LS194
SH6(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[23]), .SR(Q[28]),

     .D(PData[24]), .C(PData[25]), .B(PData[26]), .A(PData[27]),
                    .QD(Q[24]), .QC(Q[25]), .QB(Q[26]), .QA(Q[27]));

    SN74LS194
SH7(.clk(clk), .CR(CR), .S1(S1), .S0(S0), .SL(Q[27]), .SR(SR),

     .D(PData[28]), .C(PData[29]), .B(PData[30]), .A(PData[31]),
                    .QD(Q[28]), .QC(Q[29]), .QB(Q[30]), .QA(Q[31]));


endmodule
```

完成后，进行仿真验证。仿真代码如下：

```
`timescale 1ns / 1ps

module shift_32_test;

    // Inputs
    reg clk;
    reg clear;
    reg S1;
    reg S0;
    reg SL;
    reg SR;
    reg [31:0] PData;

    // Outputs
    wire [31:0] Q;

    // Instantiate the Unit Under Test (UUT)
    shift_32 uut (
        .clk(clk),
        .clear(clear),
        .S1(S1),
        .S0(S0),
        .SL(SL),
        .SR(SR),
        .PData(PData),
        .Q(Q)
    );

    initial begin
```

```
        clear = 1;
        clk = 0;
        SL = 0;
        SR = 0;
        S1 = 1;
        S0 = 1;
        PData = 32'h80000000;
        fork
            forever #20 clk<=~clk;
            #5; clear=0;
            begin
                {S1,S0}=2'b11;
                #10;
                // right
                {S1,S0}=2'b01; SR=1;
                #1320;  SR=0;
                #1320;
                // parallel
                PData = 32'hAAAAAAAA;
                {S1,S0}=2'b11;
                #50;
                PData = 32'h55555555;
                #50;
                PData = 32'h00000000;
                #50;
                SL=1;
                #50;
                // left
                {S1,S0}=2'b10;
                #1320;
                // hold
                {S1,S0}=2'b00;
                #50;
                // left
                SL=0;
                {S1,S0}=2'b10;
                #1320;

            end
        join
    end

endmodule
```

仿真结果确认无误后，创建逻辑符号，方便加入实验十一的框架进行物理验证。

3.2.3 集成移位寄存器到实验十一的“混合计算器”

1. 建立工程

打开 ISE Design Suite 14.7，在左上角点击 File，再点击 New Project 命名为 Top_Shift。

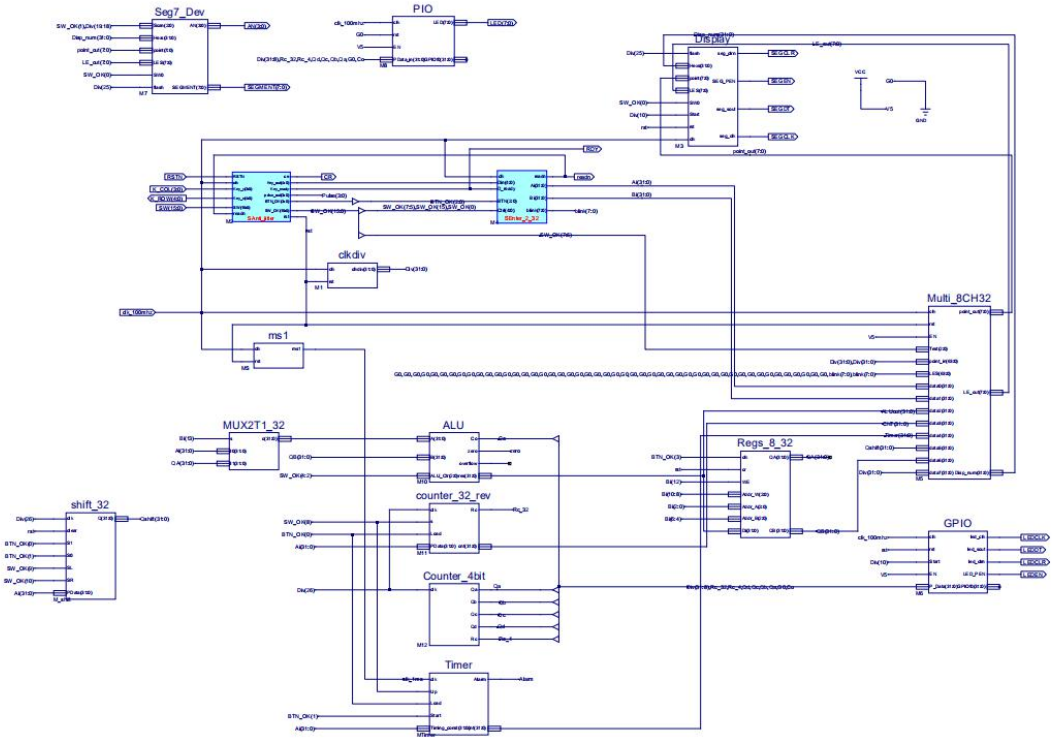
将工程命名，选择 Top-level source type 为 Schematic 并选择项目保存位置。

点击 Next 后，选择对应的 Family、Device、Package、Speed 等属性，确认无误后一直点击 Next 直至创建工程结束，同 3.2.1。

2. 集成到实验环境

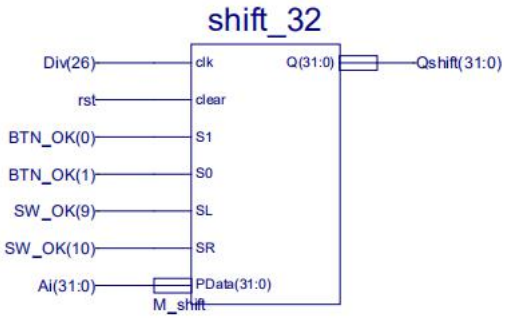
复制实验十一中含有 ALU、Counter_4bit、counter_32_rev 以及 Regs_8_32 的 Framework。加入本次实验完成的 shift_32。

加入后的原理图如图表 3.2.3-1。

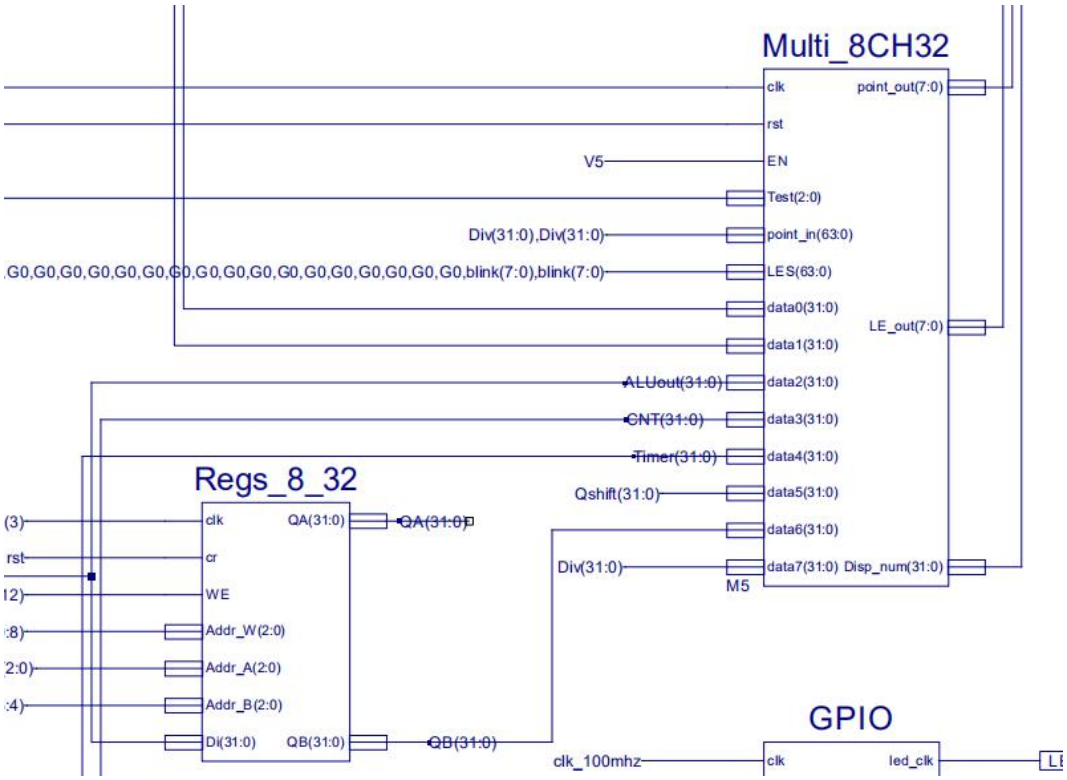


图表 3.2.3-1 Top_Shift

其中相比原框架需要修改的部分如图表 3.2.3-2 和 3.2.3-3。



图表 3.2.3-2 修改部分 1



图表 3.2.3-3 修改部分 2

后续步骤则为检查文件，生成比特流文件，并将文件下载到实验板，在实验板物理运行，不再赘述。

3.2.4 设计通用并-串转换器 P2S

采用 Verilog 代码描述 P2S(Parallel to Serial)通用并-串转换器，首先我们需要设计 64 位双向移位器 SHIFT64。新建 Verilog Module 文件 SHIFT64.v，输入代码如下：

```
`timescale 1ns / 1ps

module SHIFT64(input clk,
                  input SR,SL,
                  input S1,S0,
                  input [DATA_BITS:0]D,
                  output reg [DATA_BITS:0]Q
    );

    parameter DATA_BITS=16, DATA_COUNT_BITS=4;

    always@(posedge clk)begin
        case({S1,S0})
        2'b00:Q<=Q;
        2'b01:Q<={SR, Q[DATA_BITS:1]};
        2'b10:Q<={Q[DATA_BITS-1:0], SL};
        2'b11:Q<=D;
        endcase
    end

endmodule
```

完成后，设计转化器 P2S。新建 Verilog Module 文件 P2Snew.v。由于时间原因，将 P2Snew 加入实验框架后并未来得及进行物理验证。作为补充，我设计了仿真验证。为了方便仿真验

证，我新增了 finish 输出信号与输出 Q，这两个输出在实际物理验证时可以删去。输入代码如下：

```
`timescale 1ns / 1ps

module P2Snew(input wire clk,
              input wire rst,
              input wire Start,
              input wire [DATA_BITS-1:0]PData,
              output wire s_clk,
              output wire s_clrn,
              output wire sout,
              output wire finish,
              output wire [DATA_BITS:0]Q,
              output reg EN
    );

    parameter DATA_BITS=64, DATA_COUNT_BITS=6, DIR=1;

    wire S1, S0, SL, SR;
    wire [DATA_BITS:0]D;
    reg [1:0]Go=00;
    reg [1:0]S=00;

    always @(posedge clk)
        Go <= {Go[0], Start};
    assign shift=(Go==2'b01) ? 1:0;

    assign {SR, SL}=2'b11;
    assign {S1, S0}=DIR ? {S[0], S[1]}:S;
    assign D = DIR ? {1'b0, PData} : {PData, 1'b0};
    assign finish = DIR ? &Q[DATA_BITS:1] : &Q[DATA_BITS-1:0];
    assign sout = DIR ? Q[0] : Q[DATA_BITS];

    SHIFT64 #(.DATA_BITS(DATA_BITS)) // pass parameter

PTOS(.clk(clk), .SR(SR), .SL(SL), .S1(S1), .S0(S0), .D(D), .Q(Q));

    always @(posedge clk or posedge rst)begin
        if(rst)begin
            EN<=1;
            S<=2'b11;
        end
        else if(shift)begin
            EN<=0;
            S<=2'b11;
        end
        else if(!finish)begin
            EN<=0;
            S<=2'b10;
        end
        else begin
            EN<=1;
            S<=2'b00;
        end
    end

    assign s_clk=finish | clk;
    assign s_clrn=1;

endmodule
```

新建仿真文件 p2snew_test.v，输入仿真代码如下：

```
`timescale 1ns / 1ps

module p2snew_test;

    // Inputs
    reg clk;
    reg rst;
    reg Start;
    reg [63:0] PData;

    // Outputs
    wire s_clk;
    wire s_clrn;
    wire sout;
    wire finish;
    wire [64:0]Q;
    wire EN;

    // Instantiate the Unit Under Test (UUT)
    P2Snew uut (
        .clk(clk),
        .rst(rst),
        .Start(Start),
        .PData(PData),
        .s_clk(s_clk),
        .s_clrn(s_clrn),
        .sout(sout),
        .finish(finish),
        .Q(Q),
        .EN(EN)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        rst = 1;
        Start = 0;
        PData = 0;

        #50;
        rst=0;
        PData=64'hAAAAAAAAAAAAAAAA;
        fork
            forever #20 clk<=~clk;
            begin
                #50; Start=1; // begin to transit
                #100; Start=0;
                #3000; Start=1; PData=64'h5555555555555555;
                #100; Start=0;
            end
        join
    end

endmodule
```

四、实验结果与分析

4.1 4 位通用移位寄存器 DM74LS194 仿真验证

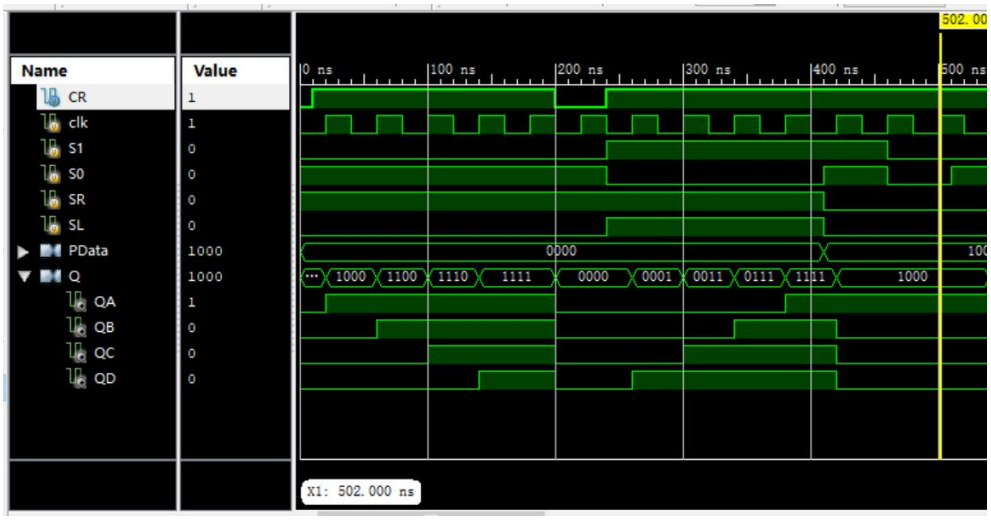
图表 4.1-1 可见，根据代码，20ns 前输出 QA QB QC QD（组合为 Q）被置零。20ns 时 CR 被置 1，此后第一个 clk 上升沿可见开始右移，将 SR=1 移入：Q 依次变化为 1000、1100、1110、1111。

240-420ns，与上述类似，可见开始左移后 SL=1 被移入 Q：Q 依次变化为 0001、0011、0111、1111。

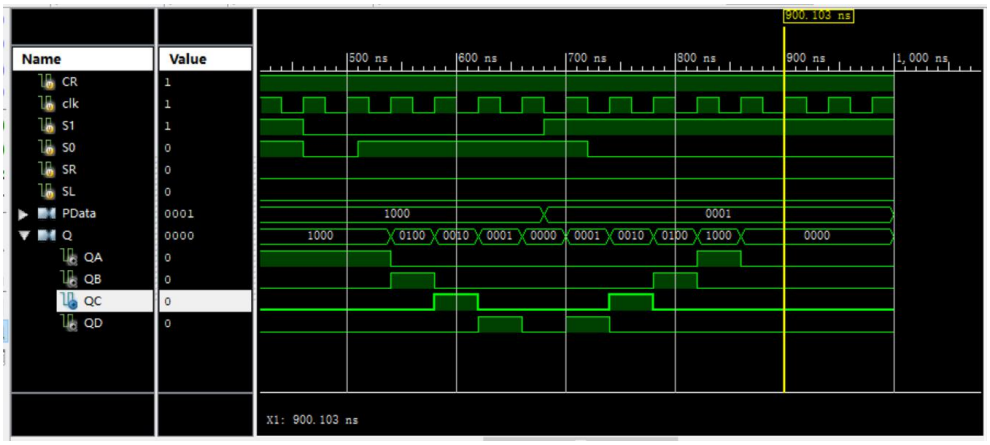
420ns 时，移位器处于并入状态。Q 为输入 PData 的值 1000。

如图表 4.1-2，540-700ns，移位器处于右移状态。可见 SR=0 被移入 Q：Q 依次变化为 0100、0010、0001、0000。此后的左移状态与此类似，不再赘述。

由仿真结果可知，设计正确。



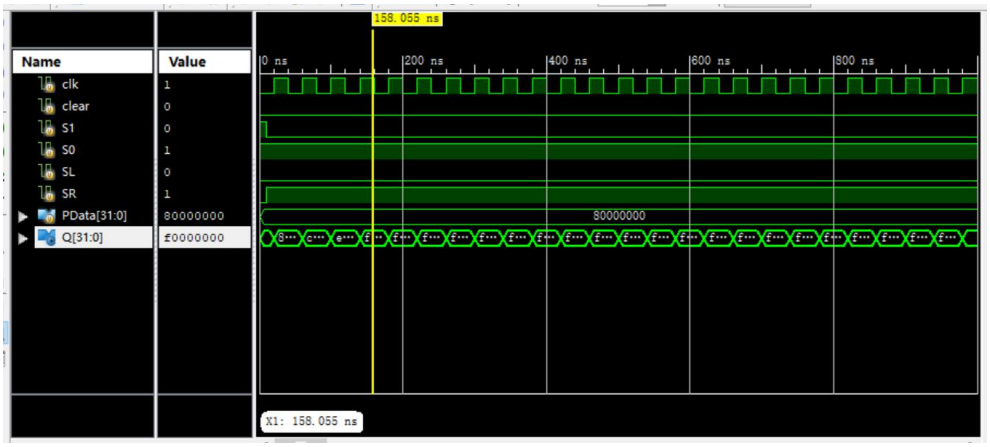
图表 4.1-1 DM74LS194 仿真模拟实验结果 1



图表 4.1-2 DM74LS194 仿真模拟实验结果 2

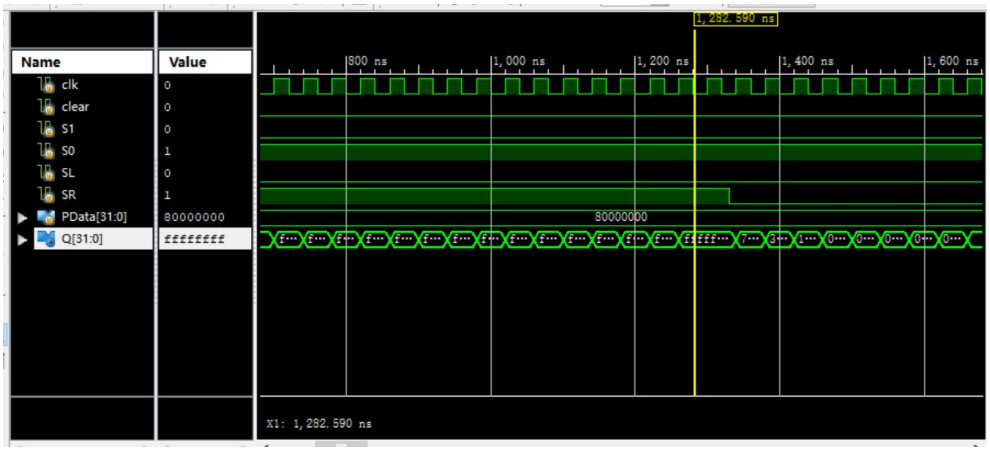
4.2 32 位通用移位寄存器仿真验证

如图表 4.2-1 可见，根据代码，20ns 时出现第一个 clk 上升沿，此时{S1,S0}=01，寄存器开始右移，将 SR=1 送入寄存器。可见 140ns 时，Q 的值为 0x00000000 四次右移后的 0xF0000000。



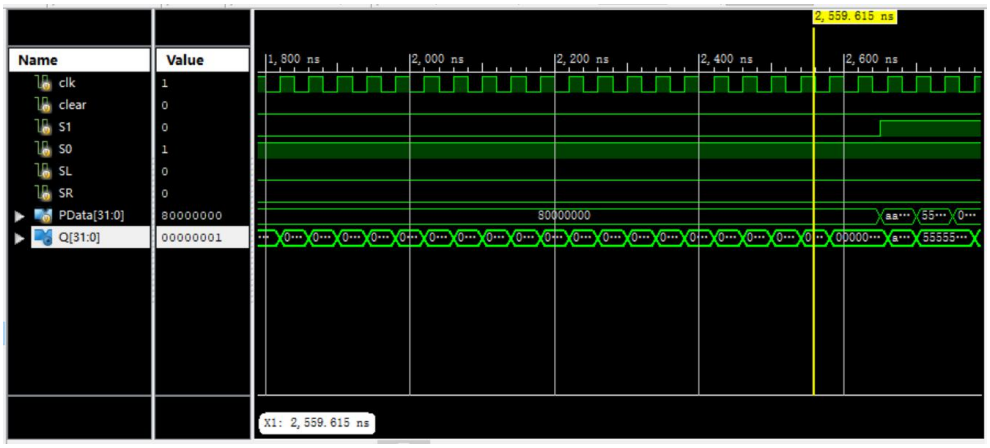
图表 4.2-1 32 位通用移位寄存器仿真模拟实验结果 1

如图表 4.2-2,1260ns 时右移了 32 次，此时 Q 的值为 0xFFFFFFFF。此后，SR 被置 0。寄存器右移时将 SR=0 送入寄存器，第一次右移后 Q 为 0x7FFFFFFF。



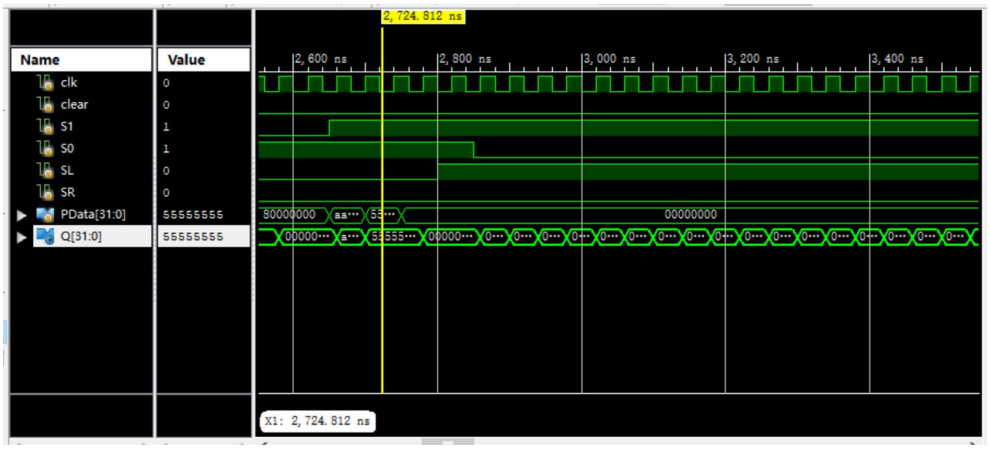
图表 4.2-2 32 位通用移位寄存器仿真模拟实验结果 2

如图表 4.2-3，2540ns 时寄存器右移 31 次，Q 为 0x00000001。20ns 后，寄存器再右移，Q 为 0x00000000。



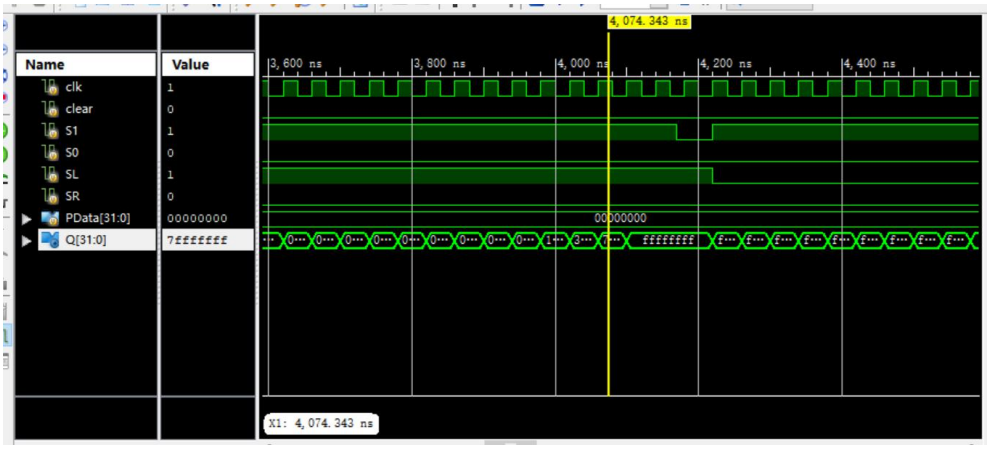
图表 4.2-3 32 位通用移位寄存器仿真模拟实验结果 3

如图表 4.2-4，2660-2860ns 为并入操作测试。可见三次并入 Q 的值都相应变为 PData 的值，正常。2860ns 以后{S1,S0}=10，开始左移，将 SL=1 送入寄存器。



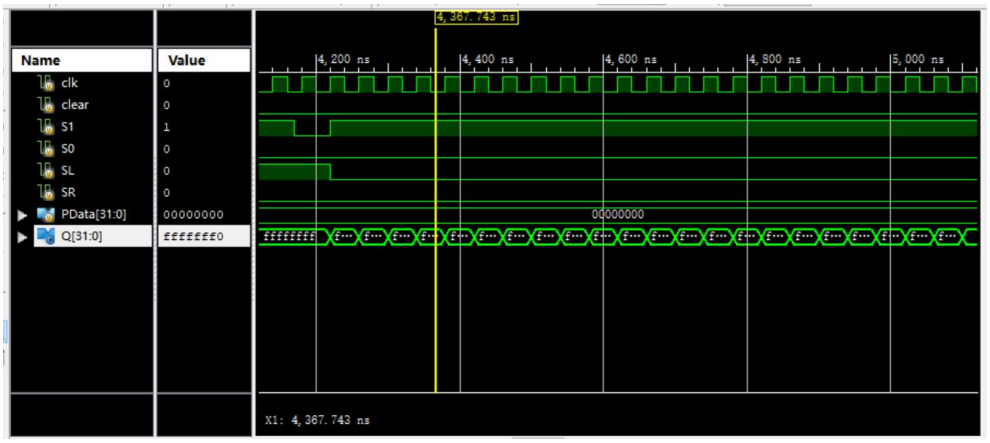
图表 4.2-4 32 位通用移位寄存器仿真模拟实验结果 4

如图表 4.2-5，4100ns 时寄存器完成了 32 次左移，此时 Q 的值为 0xFFFFFFFF。4170-4220ns 为保持状态，Q 的值不变。4220ns 后，{S1,S0}=10，寄存器处于左移状态，将 SL=0 送入寄存器。



图表 4.2-5 32 位通用移位寄存器仿真模拟实验结果 5

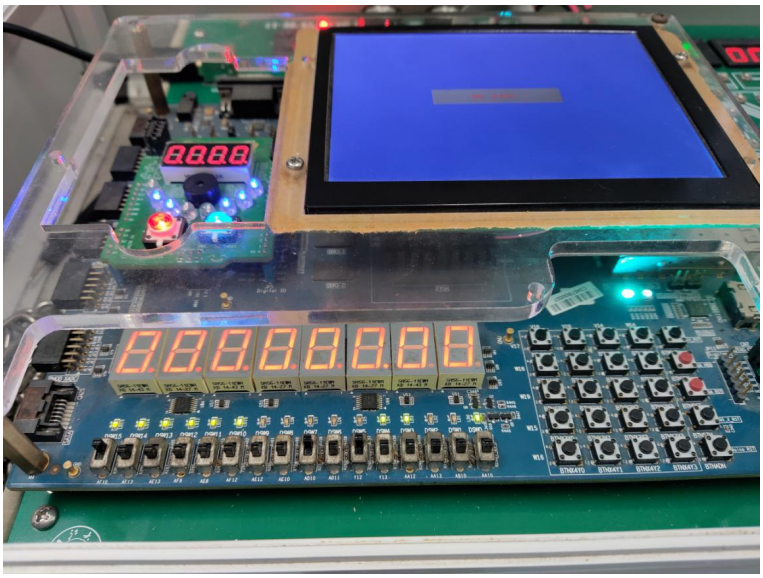
如图表 4.2-6，可见完成四次 SL=0 的左移操作后，Q 的值变为 0xFFFFFFF0。结果无误。由上述仿真测试可知，设计正确。



图表 4.2-6 32 位通用移位寄存器仿真模拟实验结果 6

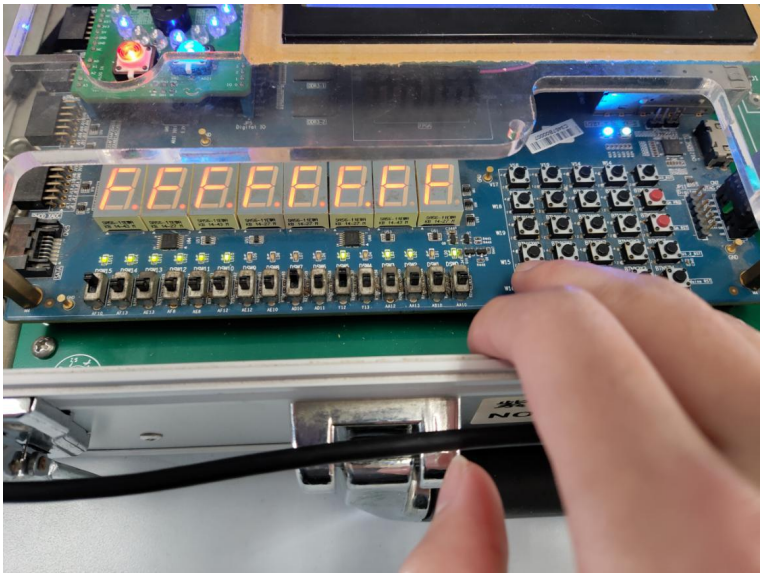
4.3 集成移位寄存器到实验十一的“混合计算器”

先设置 Ai 为 0xFFFFFFFF。如图表 4.3-1，在同时按下 BTN[0]和 BTN[1]（{S1,S0}=11，并入信号）之前，可见移位寄存器输出仍为初始的 0，因为此时{S1,S0}=00，为保持信号。



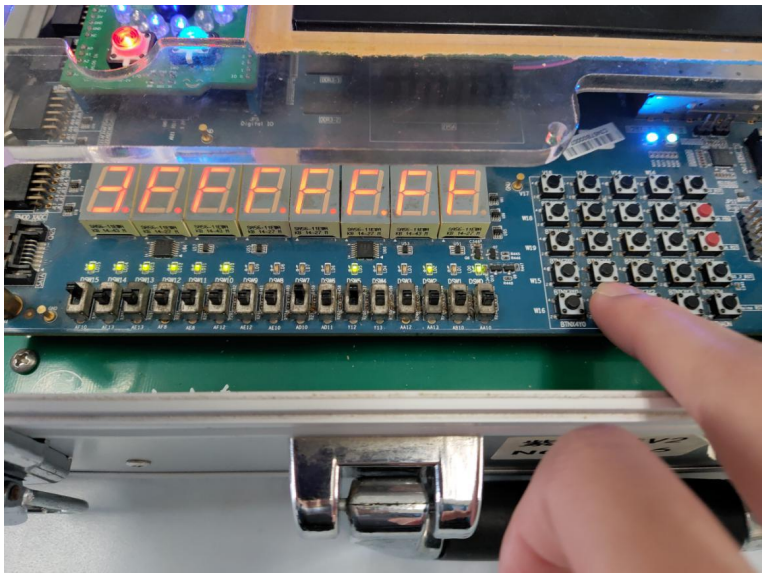
图表 4.3-1 移位寄存器初始值

而如图表 4.3-2，可见当输入并入信号后，移位寄存器输出值为输入值 0xFFFFFFFF。



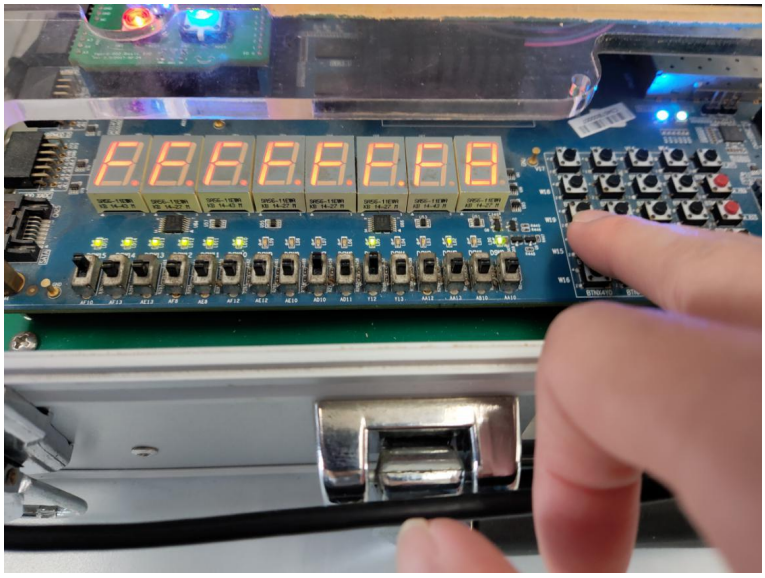
图表 4.3-2 并入

如图表 4.3-3，只按下 BTN[1]，使得{S1,S0}=01，产生右移信号。当时钟信号上升沿到来，可见移位寄存器输出发生变化。由于此时 SW[10]=0(SR=0)，右移空出的最高位补充 0，右移两次后，输出为 0x3FFFFFFF。



图表 4.3-3 右移两次

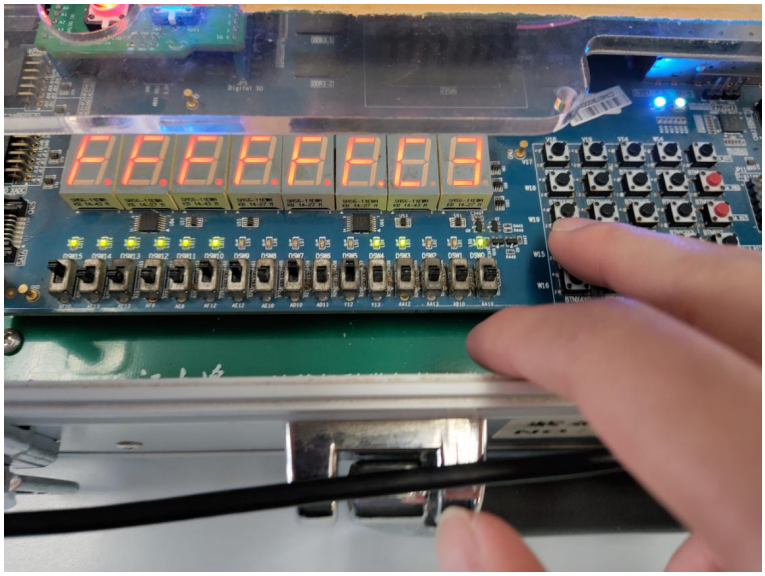
如图表 4.3-4，只按下 BTN[0]，使得{S1,S0}=10，产生左移信号。当时钟信号上升沿到来，可见移位寄存器输出发生变化。由于此时 SW[9]=0(SL=0)，左移空出的最低位补充 0，左移五次后，输出由 0x3FFFFFFF 变为 0xFFFFFFF8。



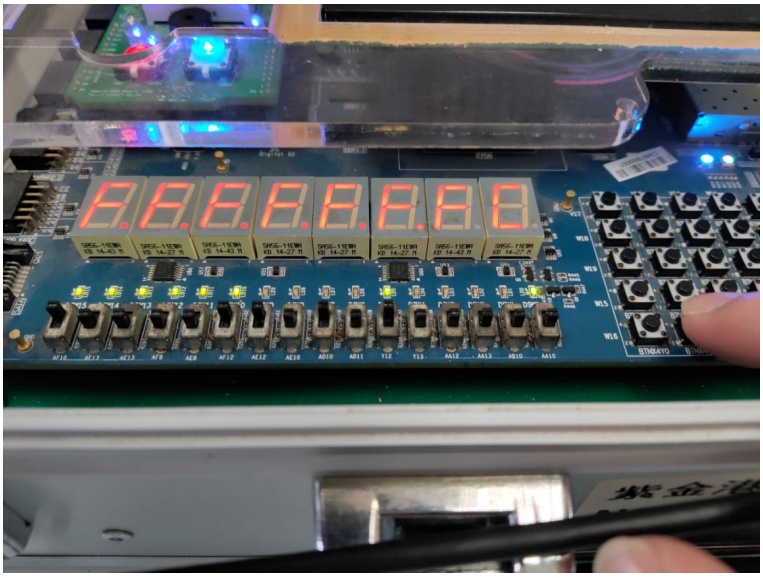
图表 4.3-4 左移五次

在图表 4.3-4 的基础上再左移一次得到 0xFFFFFFF0。如图表 4.3-5 和图表 4.3-6，将 SW[9]和 SW[10]都置 1，SL=SR=1。此时再左移三次，得到 0xFFFFFFC3；再右移四次得到 0xFFFFFFFC。

综上，可知物理验证结果无误。



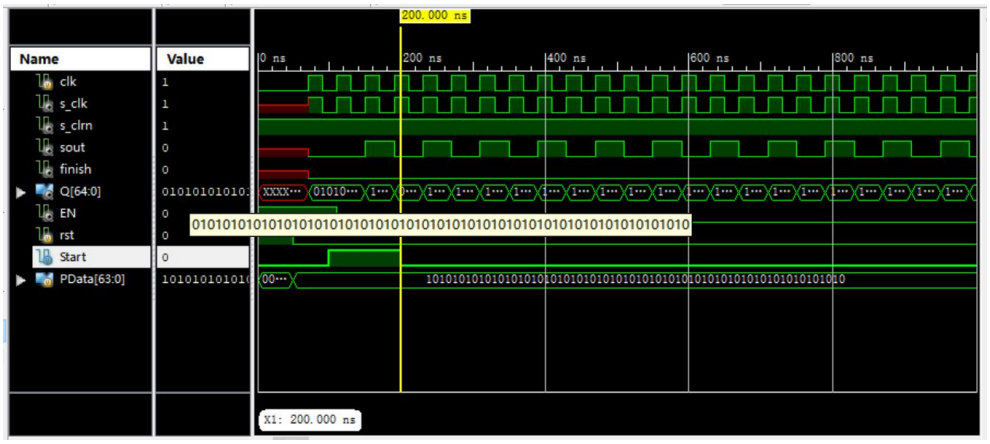
图表 4.3-5 左移补充 1



图表 4.3-6 右移补充 1

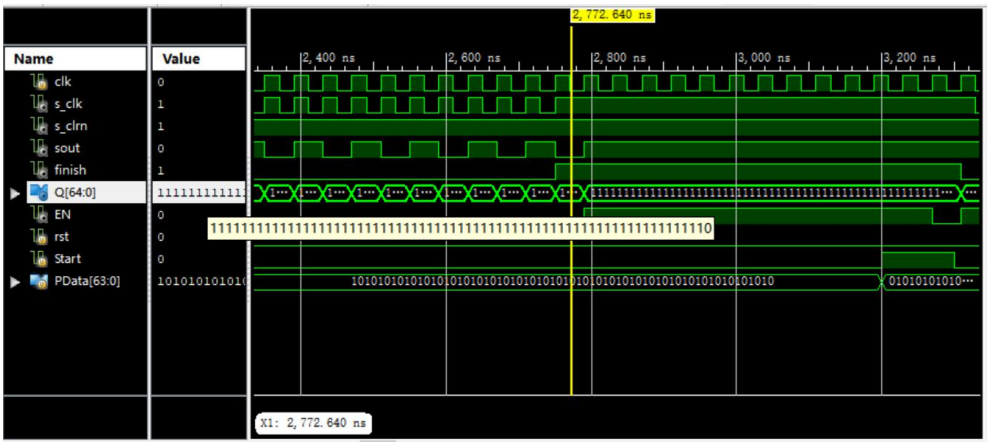
4.4 P2S 仿真验证

如图表 4.4-1 可见，根据代码，Start 为 1 时 Q 的最高位为 0，剩余 64 位为 PData 的值。此后开始右移，由于转换未完成，此时的 finish 信号为 0。同时可见 sout 信号的频率是 clk 信号的一半，这是由于一个时钟周期触发一次移位，而 PData 的值恰好是 0、1 间隔的。



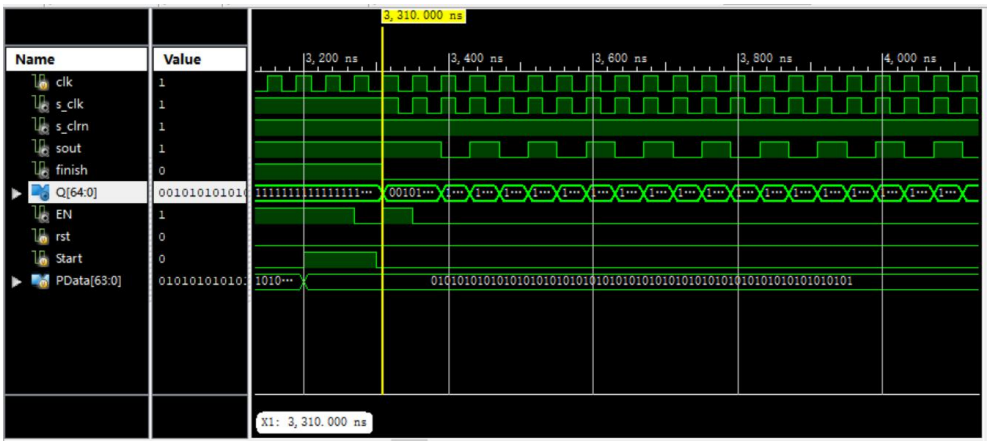
图表 4.4-1 P2S 仿真模拟实验结果 1

如图表 4.4-2 可见，2750ns 时，P2S 已右移 64 次，此时 Q 除最低位外全为 1，相应地 finish 信号为 1。最后一个 0 右移后，sout 信号也保持为最低位的 1。

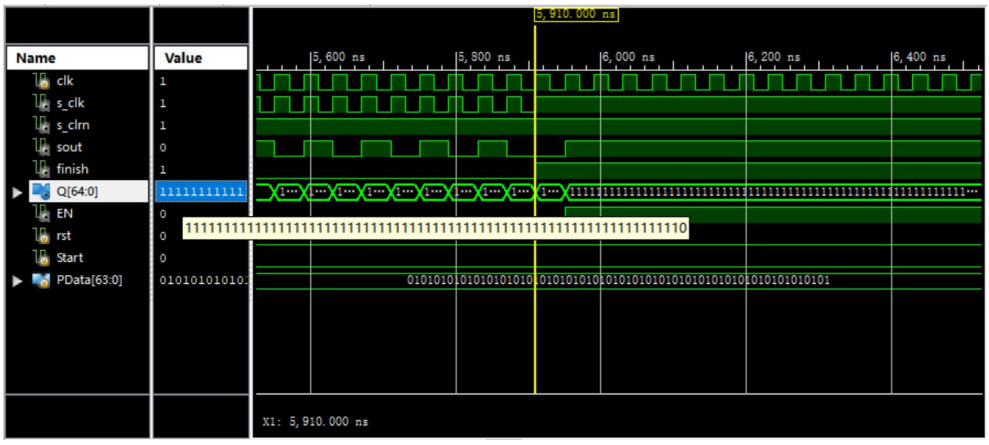


图表 4.4-2 P2S 仿真模拟实验结果 2

如图表 4.4-3 与图表 4.4-4 可见，分析与上述类似，不再赘述。



图表 4.4-3 P2S 仿真模拟实验结果 3



图表 4.4-4 P2S 仿真模拟实验结果 4

五、讨论、心得

本次实验让我接触到了移位寄存器，认识到这是一种比普通寄存器更为高级的存在，可以同时实现移位以及寄存的功能。在实验过程中，也对并行、串行的特点有了更加深刻的认识。