**北京科技大学数字逻辑实验报告**

学院： 计通 专业： 计算机 班级： 计1701

姓名： 吴思懿 学号： 41724010 实验日期： 2018 年 11 月 26 日

**实验名称：**实验一、基本练习

**实验目的：**熟悉实验环境，学习如何使用 Vivado 2018创建工程、代码编辑与仿真、综合生成网表、添加约束、下载运行等功能。

**实验内容：**

（1）学习视频，了解 Vivado 设计流程和功能：“EGo 五分钟快速上手.mp4”

和“Ego五分钟搭建你的数字积木.mp4”。

（2）按照“1、流水灯设计.pdf”完成，流水灯实验；

（3）简单Verilog模块练习：A、设计一个周期为40个时间单位的时钟信号，其占空比为25％。使用always和initial块进行设计。时钟信号在仿真0时刻的值初始化为0。B、分别使用门级描述、和逻辑表达式实现一个1位的二选一多路选择器。C、使用上面设计的二选一多路选择器模块搭建一个1位的四选一多路选择器。D、课后练习：分别使用if语句和case语句实现四选一多路选择器。E、课后练习：使用case语句实现简单ALU计算模块（包含加、减、乘、除、移位运算）

**实验步骤：**

实验内容（1）

流水灯：

实验源代码counter文件

|  |
| --- |
| `timescale 1ns / 1ps  module counter(  input clk,  input rst,  output clk\_bps  );  reg [13:0]ent\_first,ent\_second;  always @(posedge clk or posedge rst)  if(rst)  ent\_first<=14'd0;  else if(ent\_first==14'd10)  ent\_first<=14'd0;  else  ent\_first<=ent\_first+1'b1;  always@(posedge clk or posedge rst)  if(rst)  ent\_second<=14'd0;  else if(ent\_second==14'd10)  ent\_second<=14'd0;  else if(ent\_first==14'd10)  ent\_second<=ent\_second+1'b1;  assign clk\_bps = ent\_second==14'd10 ? 1'b1 : 1'b0;  endmodule |

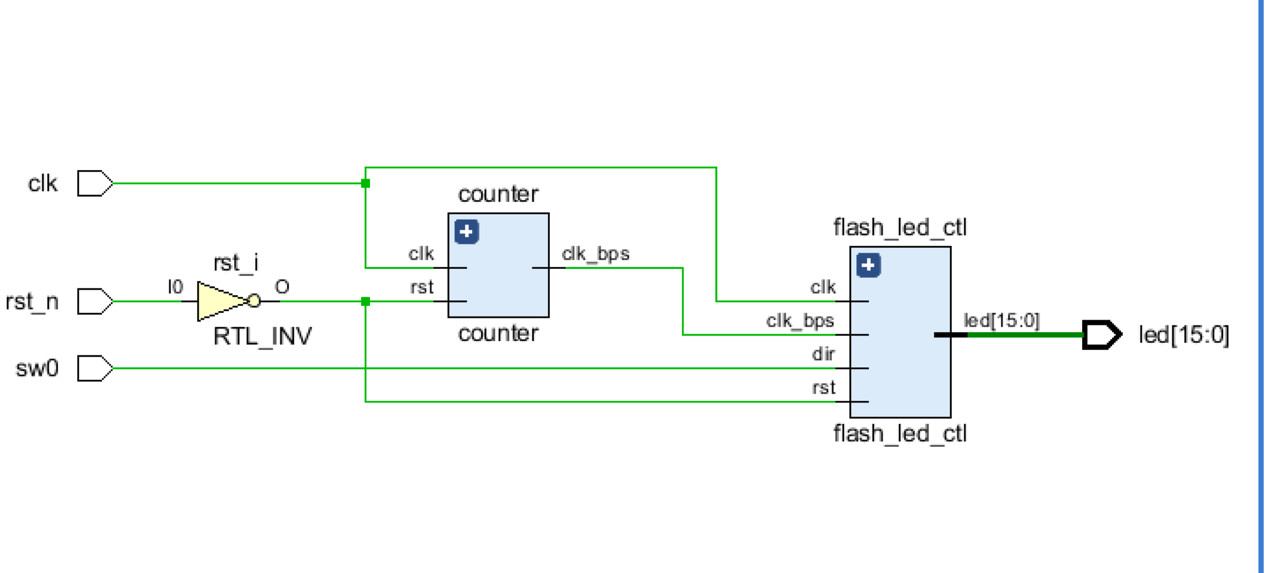
Top文件

|  |
| --- |
| `timescale 1ns / 1ps  module flash\_led\_top(  input clk,  input rst\_n,  input sw0,  output [15:0]led  );  wire clk\_bps;  wire rst;  assign rst = ~rst\_n;    counter counter(  .clk( clk ),  .rst( rst ),  .clk\_bps( clk\_bps )  );  flash\_led\_ctl flash\_led\_ctl(  .clk( clk ),  .rst( rst ),  .dir( sw0 ),  .clk\_bps( clk\_bps ),  .led( led )  );  endmodule |

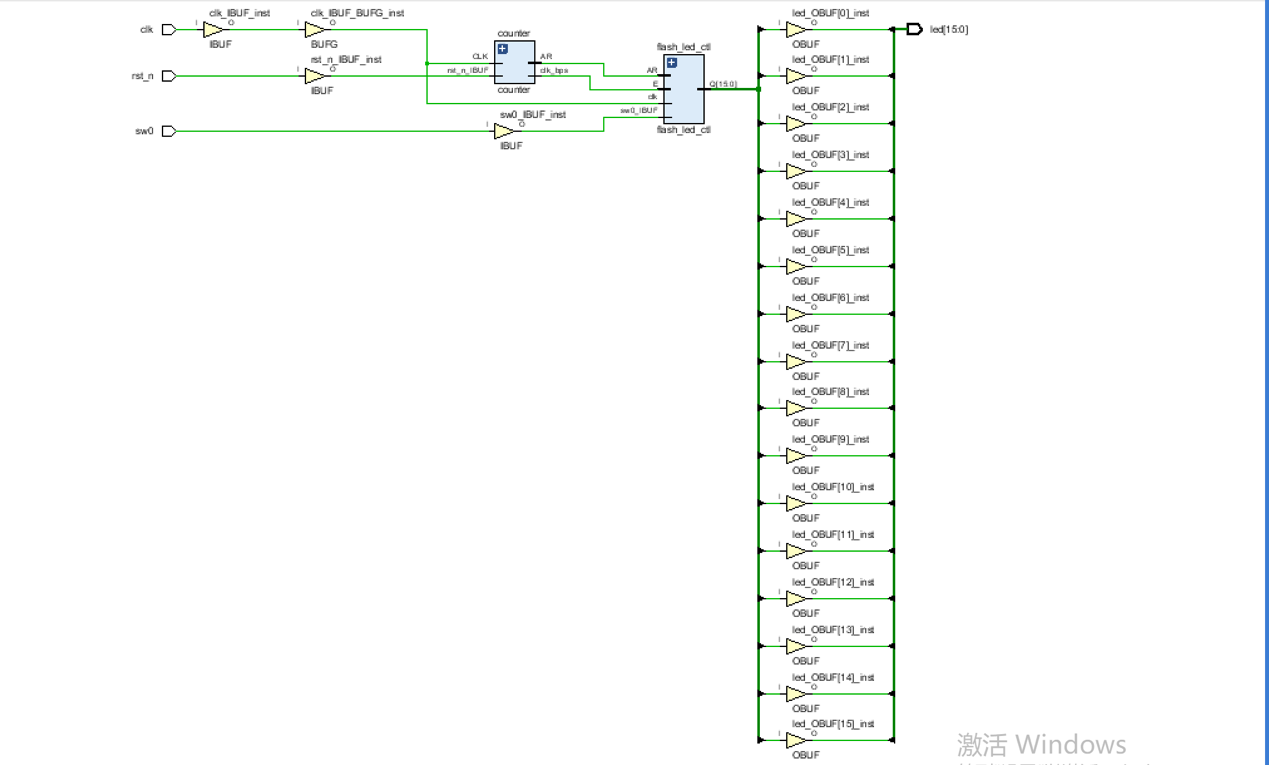
Ctl文件

|  |
| --- |
| `timescale 1ns / 1ps  module flash\_led\_ctl(  input clk,  input rst,  input dir,  input clk\_bps,  output reg[15:0]led  );  always @( posedge clk or posedge rst )  if( rst )  led <= 16'h8000;  else  case( dir )  1'b0: //从左向右  if( clk\_bps )  if( led != 16'd1 )  led <= led >> 1'b1;  else  led <= 16'h8000;  1'b1: //从右向左  if( clk\_bps )  if( led != 16'h8000 )  led <= led << 1'b1;  else  led <= 16'd1;  endcase  endmodule |

RTL图



综合实现图



实验内容（2）

A时钟代码：

|  |
| --- |
| `timescale 1ns / 1ps  module clk\_tb();  reg clk;  initial  clk<=0;  always  begin  #30 clk<=1;  #10 clk<=0;  end  endmodule |

RTL详细设计图

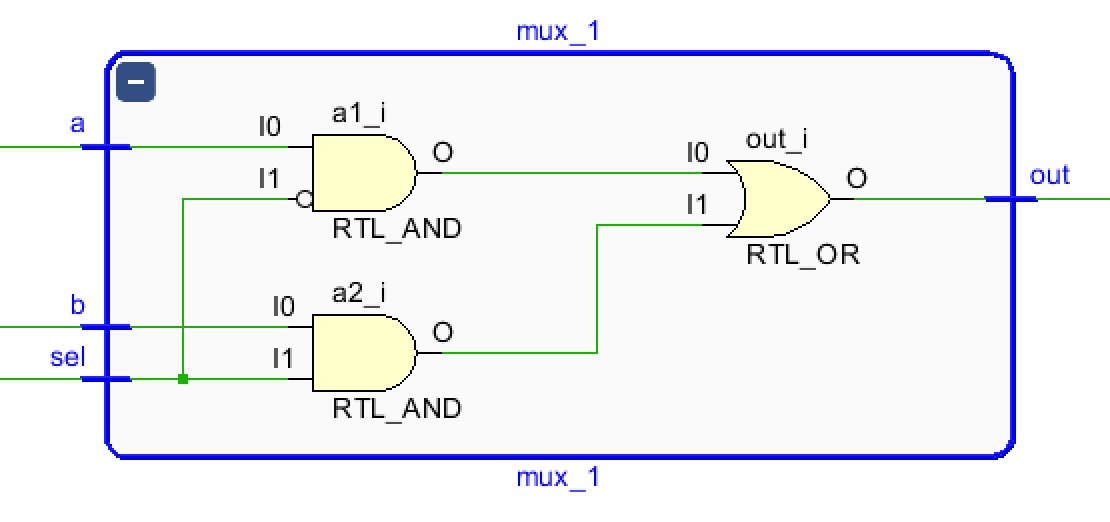
说明：因为仅仅是一个时钟脉冲信号，所以无RTL图

B门级描述二选一

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_1(out,sel,a,b);  output out;  input sel,a,b;  wire out;  wire \_sel,a1,a2;  not(\_sel,sel);  and(a1,a,\_sel);  and(a2,b,sel);  or(out,a1,a2);  endmodule |

RTL图



说明：这是将最终利用这个二选一模块制作四选一后将模块放大的RTL图。

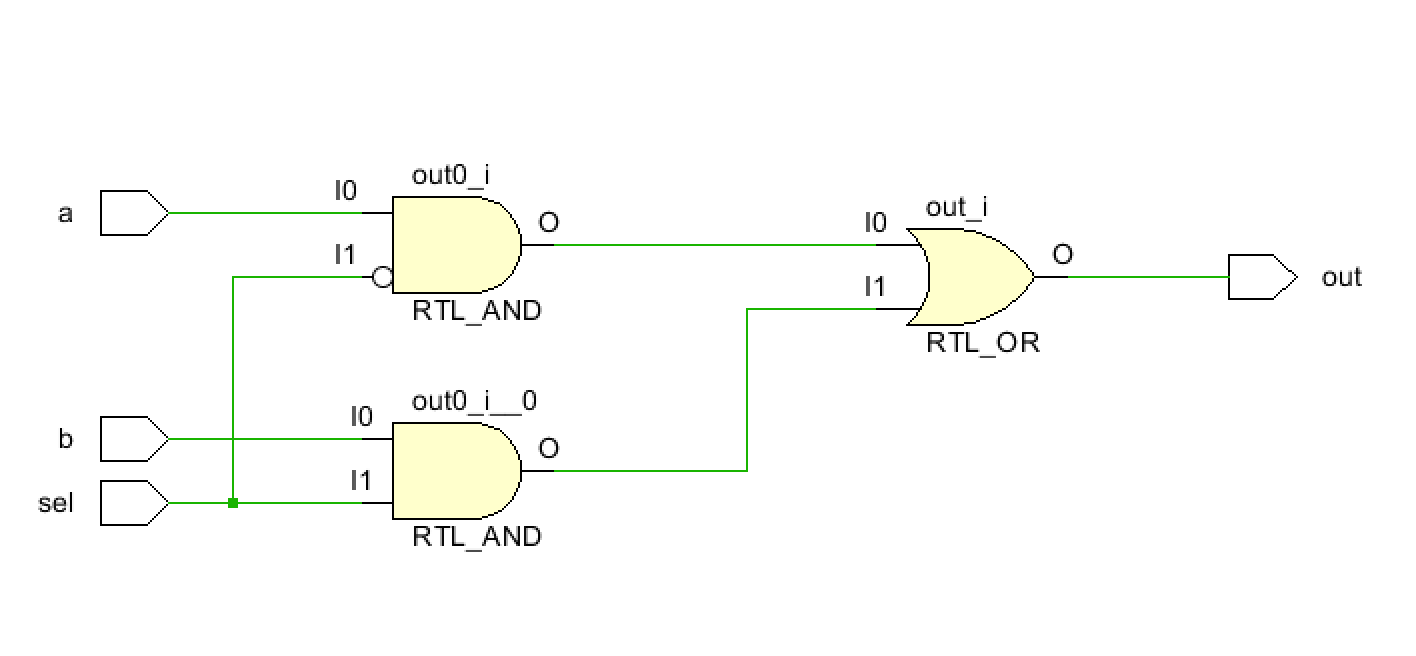
B逻辑表达式二选一

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_2(  input a,  input b,  input sel,  output out);  assign out=(a&~sel)|(b&sel);  endmodule |

说明：与门级设计想法一致

RTL详细设计图

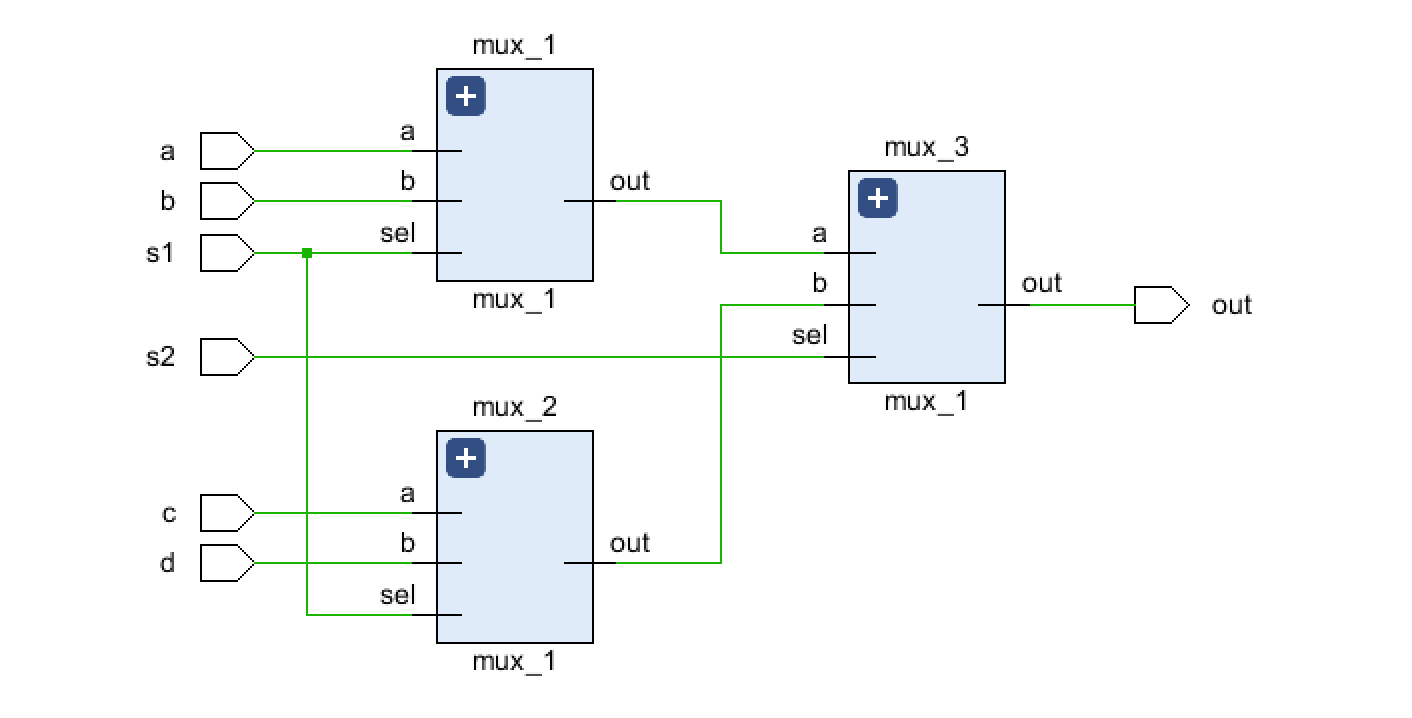


C四选一多路选择器

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_C(  input wire a,b,c,d,  input wire s1,s2,  output wire out );  wire out1,out2;  mux\_1 mux\_1(  .a(a),  .b(b),  .sel(s1),  .out(out1));  mux\_1 mux\_2(  .a(c),  .b(d),  .sel(s1),  .out(out2));  mux\_1 mux\_3(  .a(out1),  .b(out2),  .sel(s2),  .out(out));  endmodule |

RTL设计图

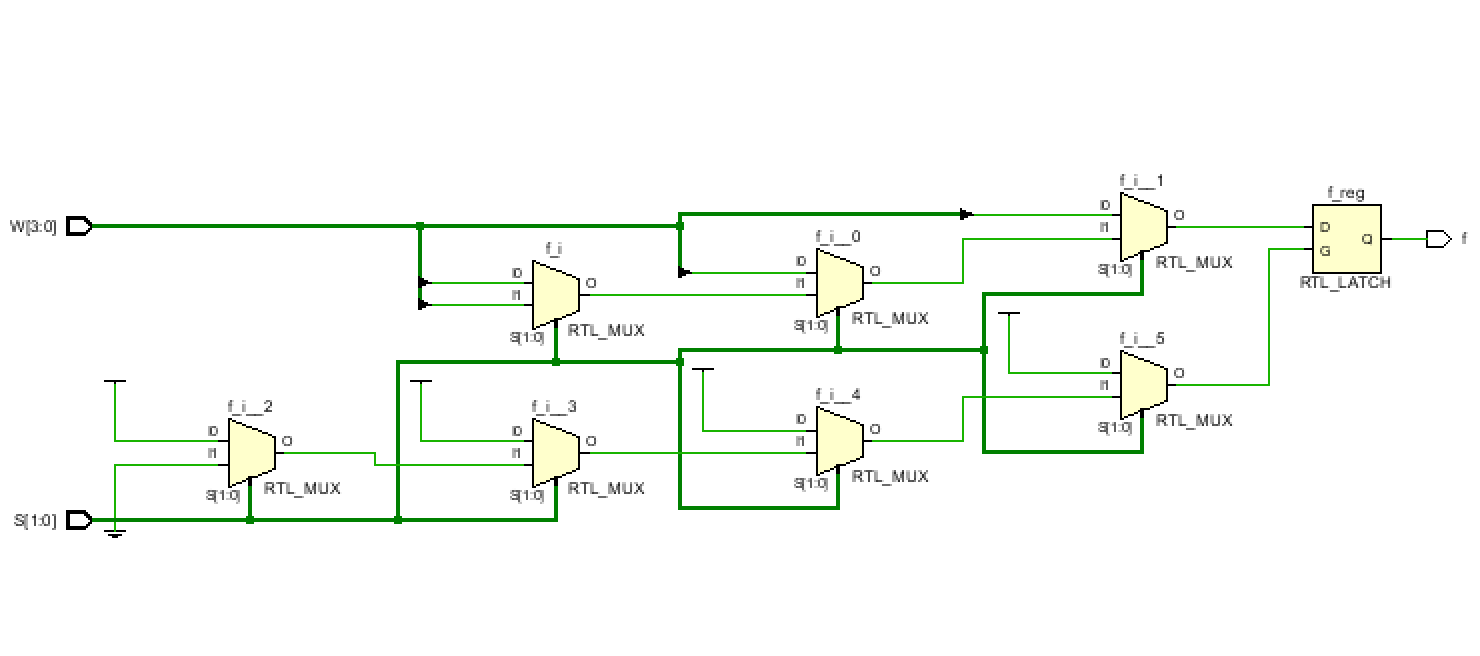


D1 if实现四选一

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_if(W,S,f );  input[3:0]W;  input[1:0]S;  output reg f;  always@(W,S)  if(S==2'b00)  f=W[0];  else if(S==2'b01)  f=W[1];  else if(S==2'b10)  f=W[2];  else if(S==2'b11)  f=W[3];  endmodule |

RTL设计图

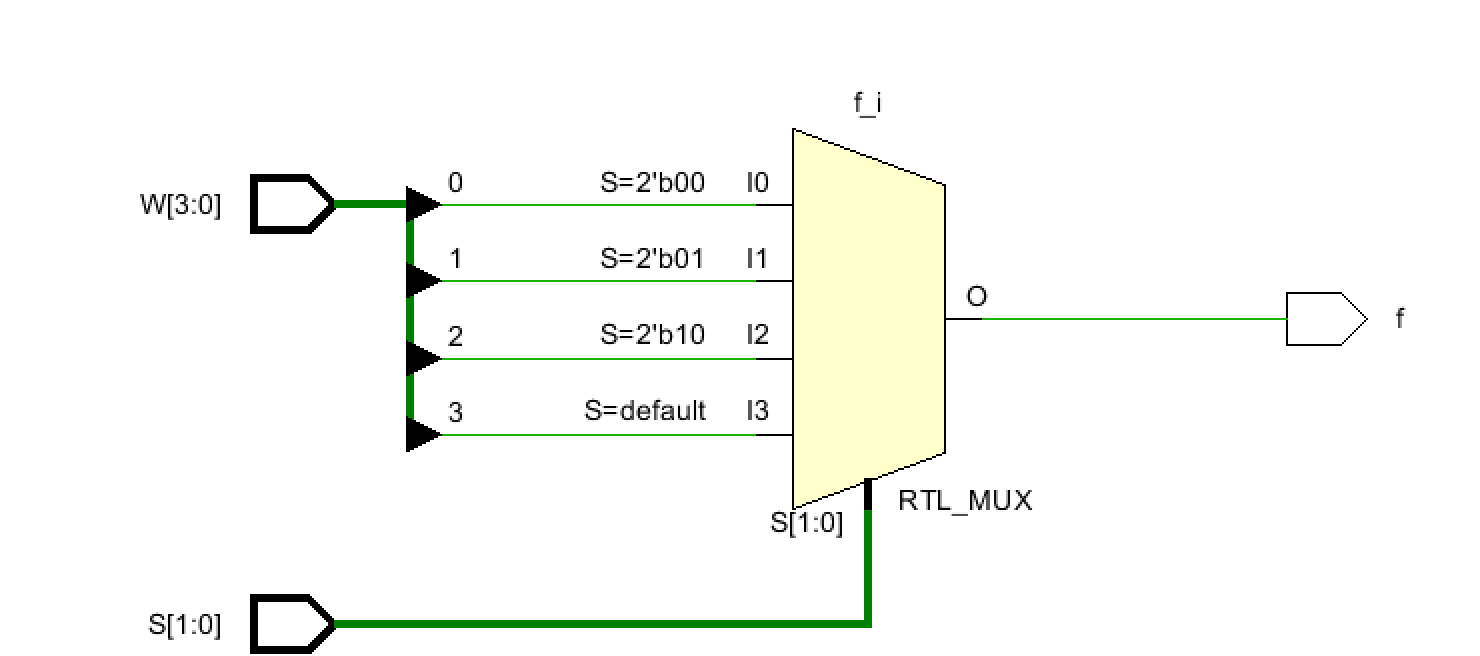


D1 case实现四选一

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_case(W,S,f );  input[3:0]W;  input[1:0]S;  output reg f;  always@(W,S)  case(S)  0:f=W[0];  1:f=W[1];  2:f=W[2];  default:f=W[3];  endcase  endmodule |

RTL设计图

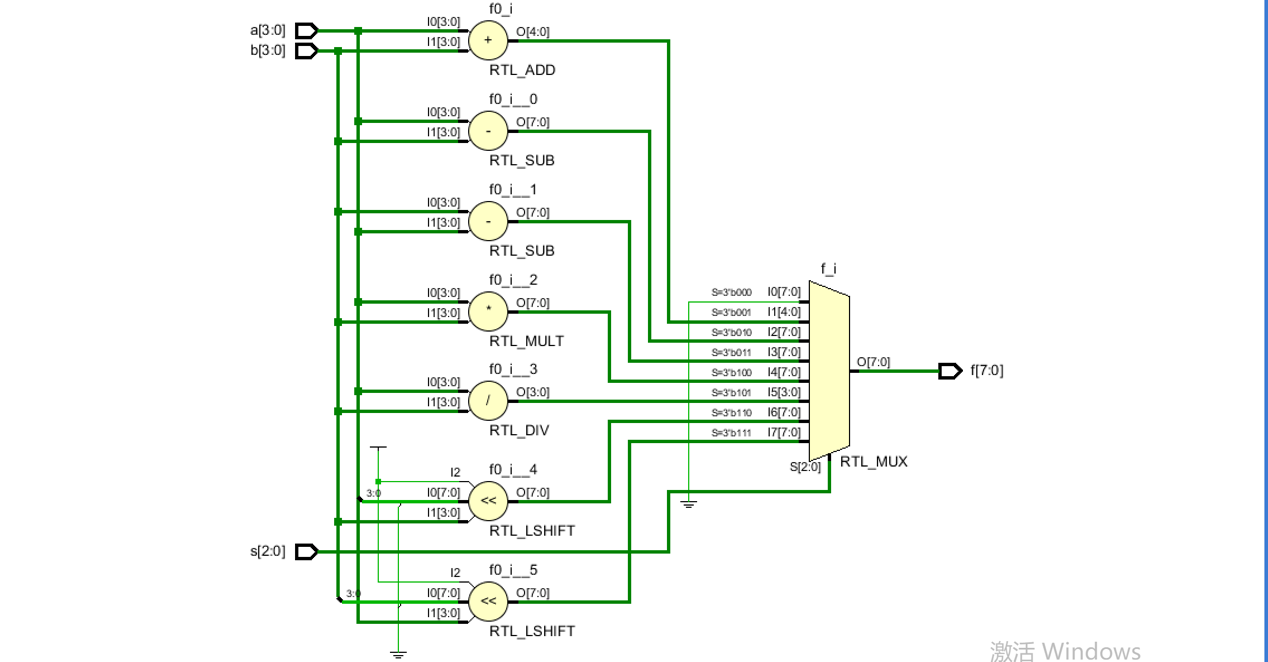


D2ALU计算

源文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module alu(s,a,b,f);  input [2:0]s;  input [3:0]a,b;  output reg [7:0]f;  always@(\*)  case(s)  0:f=4'b0000;  1:f=a+b;  2:f=a-b;  3:f=b-a;  4:f=a\*b;  5:f=a/b;  6:f=a<<b;  7:f=b<<a;  default:f=4'b1111;  endcase  endmodule |

RTL设计图



**实验结果与分析：**

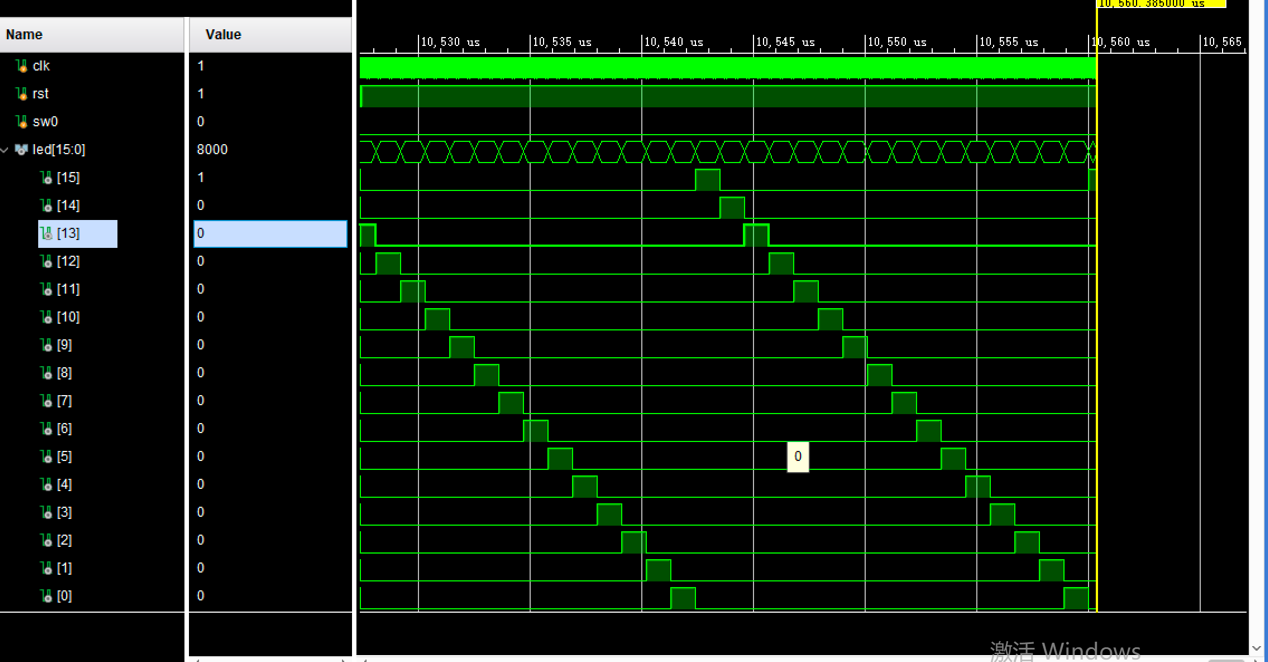
**实验内容1**

**流水灯**

**激励文件代码**

|  |
| --- |
| **`timescale 1ns / 1ps**  **module flash\_led\_top\_tb;**  **reg clk,rst,sw0;**  **wire [15:0]led;**  **initial begin**  **clk=1'b0;**  **rst=1'b1;**  **sw0=1'b0;**  **#10 rst=1'b0;**  **#10 rst=1'b1;**  **#1000000000**  **#1000000000**  **#1000000000**  **#1000000000**  **#1000000000**  **#1000000000**  **sw0=1'b1;**  **end**  **always#5 clk<=~clk;**  **flash\_led\_top flash\_led\_top(**  **.clk(clk),**  **.rst\_n(rst),**  **.sw0(sw0),**  **.led(led)**  **);**  **endmodule** |

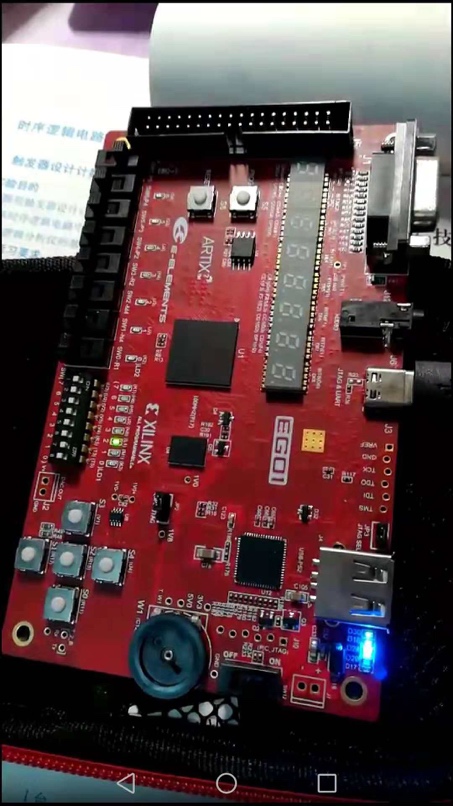
**仿真结果截图**

****

**流水灯仿真结果截图**

**说明：在counter文件中，将等待时间d1000改成了d1,于是可将时间缩短，减少内存消耗**

**现象图**

****

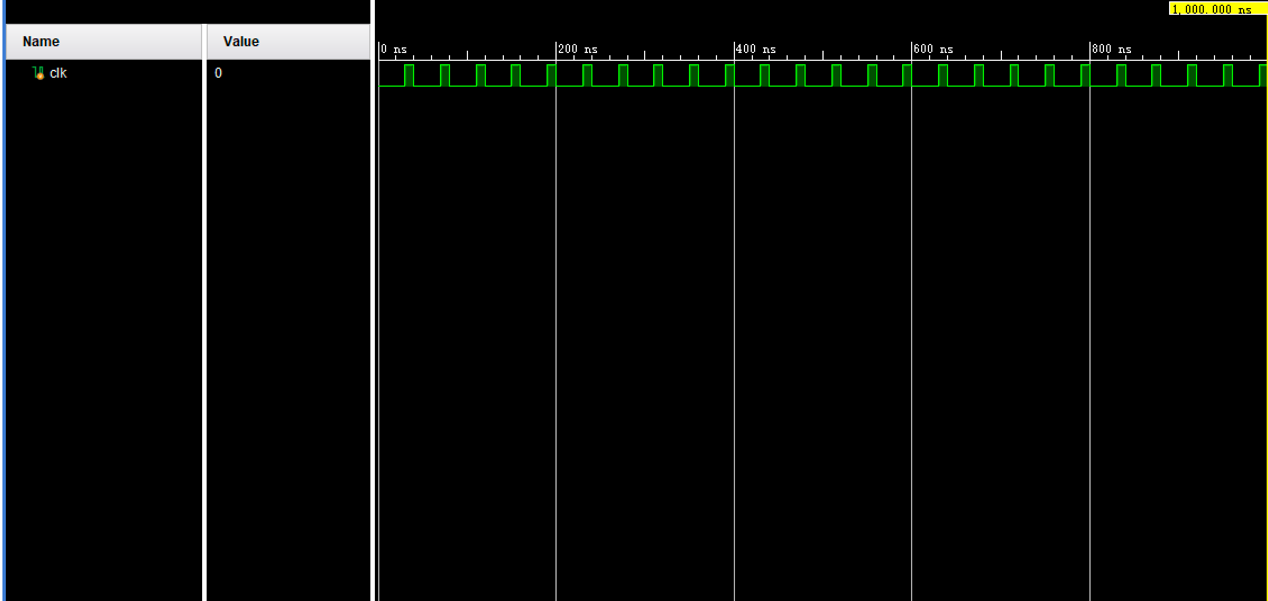
**流水灯现象图**

**说明：这是流水灯进行到一半的照片**

**实验内容2**

**A时钟**

仿真结果截图



时钟仿真截图

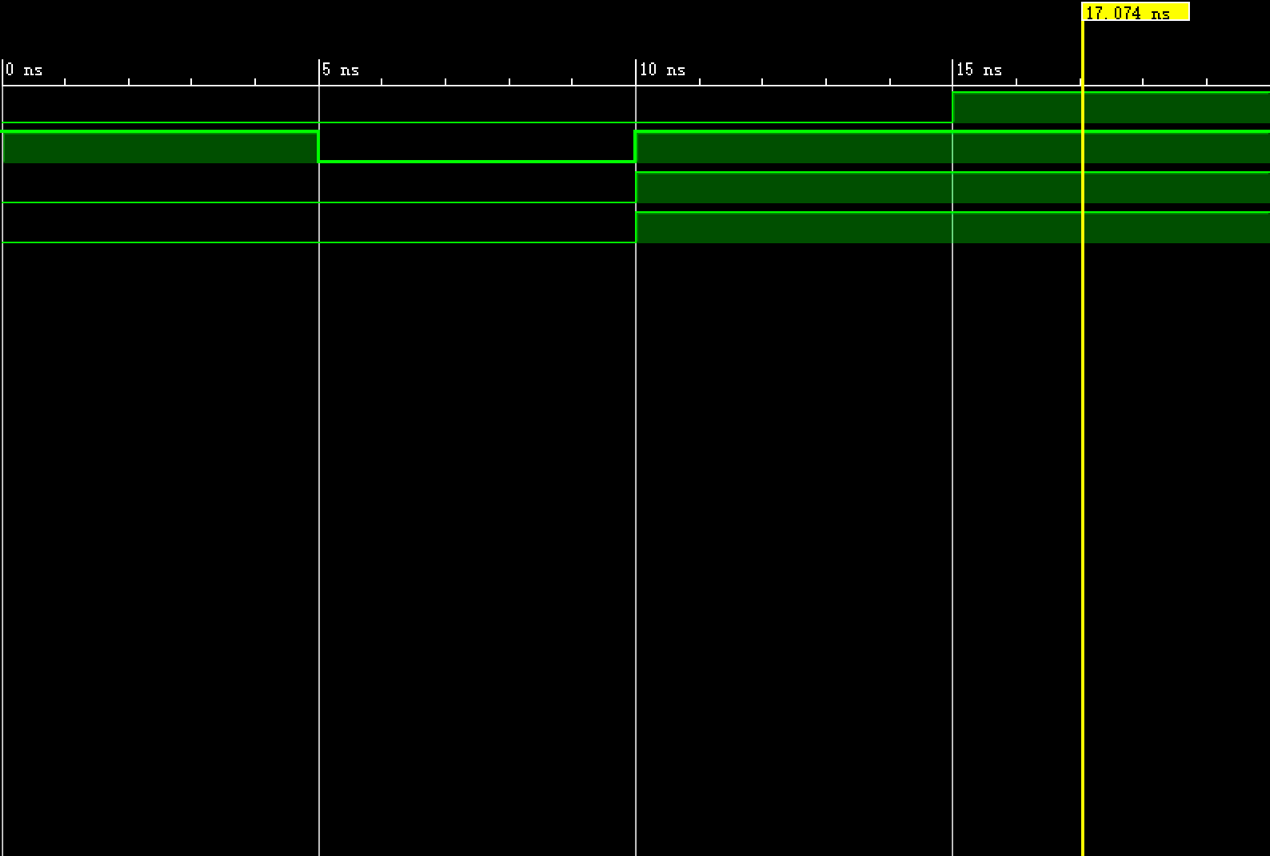
说明：以1ns为一个时间单位，此仿真图中，刚开始30ns为低电平，之后10ns为高电平，以40ns为一个周期，由此波动。

B门级描述二选一

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_1\_tb;  reg a,b,sel;  mux\_1 mux(out,sel,a,b);  initial begin  a=0;b=1;sel=0;  #5b=0;#5sel=1;b=1;  #5a=1;  #5$finish;  end  initial  $monitor($time,"out=%b ,sel=%b ,a=%b ,b=%b",out,sel,a,b);  endmodule |

仿真结果



门级描述二选一仿真截图

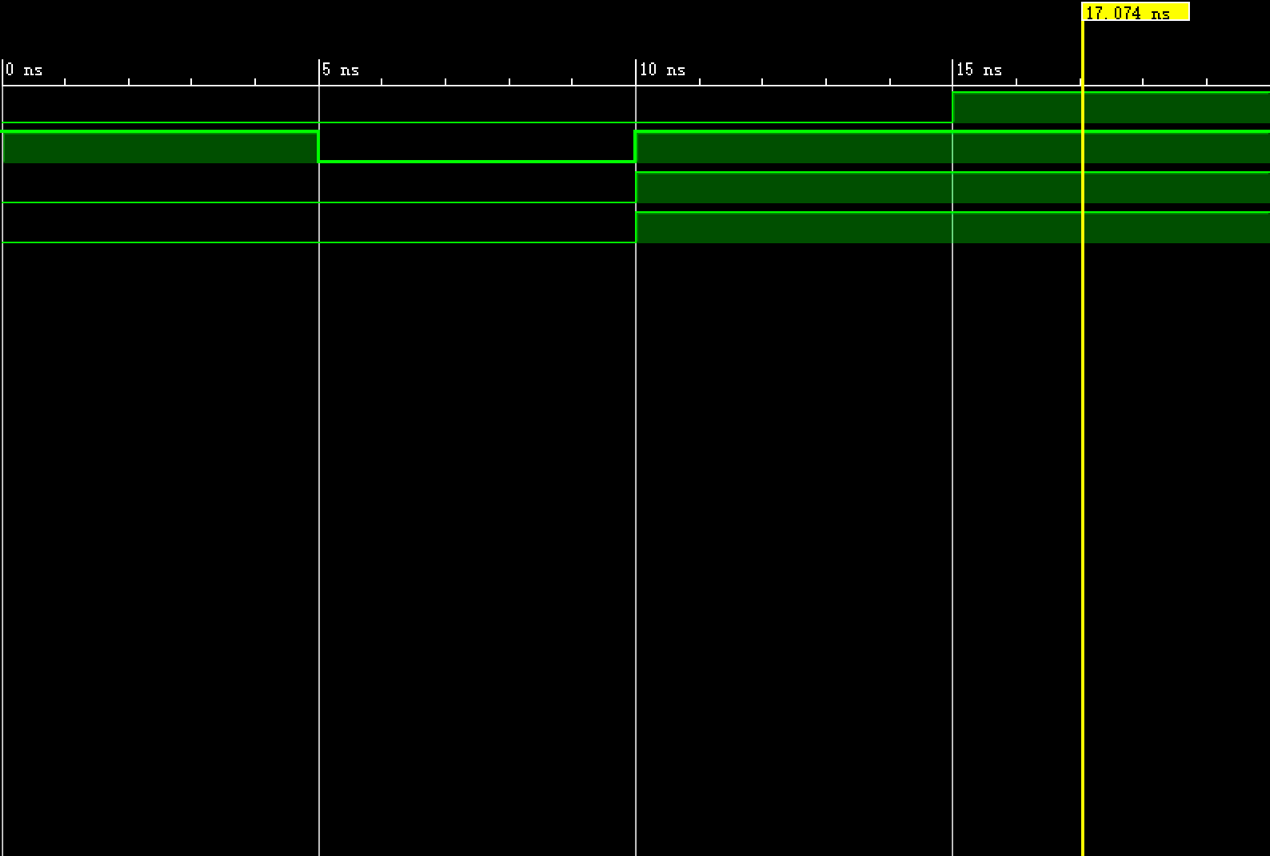
说明：我将有选择功能的sel设置为值为1时，选择输出b的值，值为0时，选择输出a的值。

B逻辑表达式二选一

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_2\_tb;  reg a,b,sel;  mux\_2 mux\_2(out,sel,a,b);  initial begin  a=0;b=1;sel=0;  #5b=0;#5sel=1;b=1;  #5a=1;  #5$finish;  end  initial  $monitor($time,"out=%b ,sel=%b ,a=%b ,b=%b",out,sel,a,b);  endmodule |

仿真结果截图



逻辑表达式二选一仿真结果截图

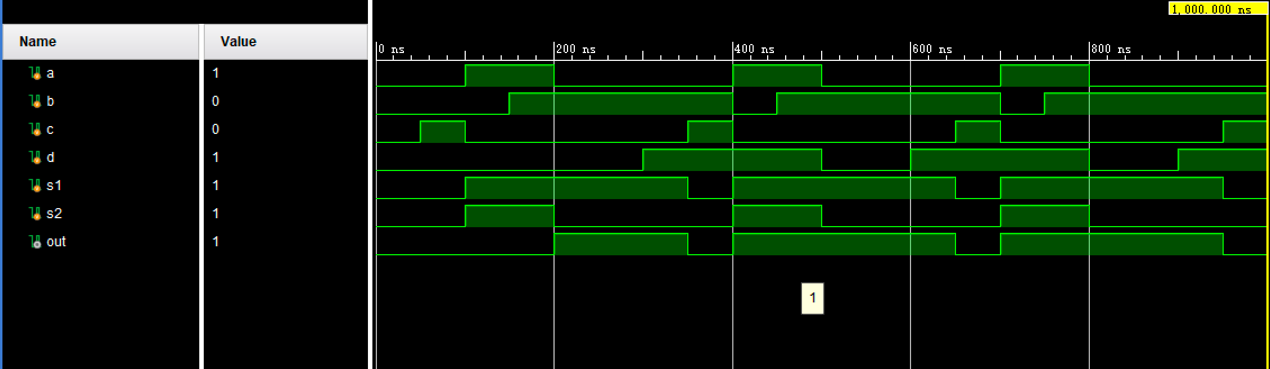
说明：与门级设计想法一致

C四选一多路选择器

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_C\_tb;  reg a,b,c,d;  reg s1,s2;  wire out;  initial begin  a=0;b=0;c=0;d=0;s1=0;s2=0;  end  always  fork  #100a=1;#200a=0;  #150b=1;#100b=0;  #50c=1;#100c=0;  #300d=1;#200d=0;  #50s1=0;#100s1=1;  #100s2=1;#200s2=0;  join  mux\_C m1(.a(a),.b(b),.c(c),.d(d),.s1(s1),.s2(s2),.out(out));  endmodule |

仿真结果图



四选一仿真结果截图

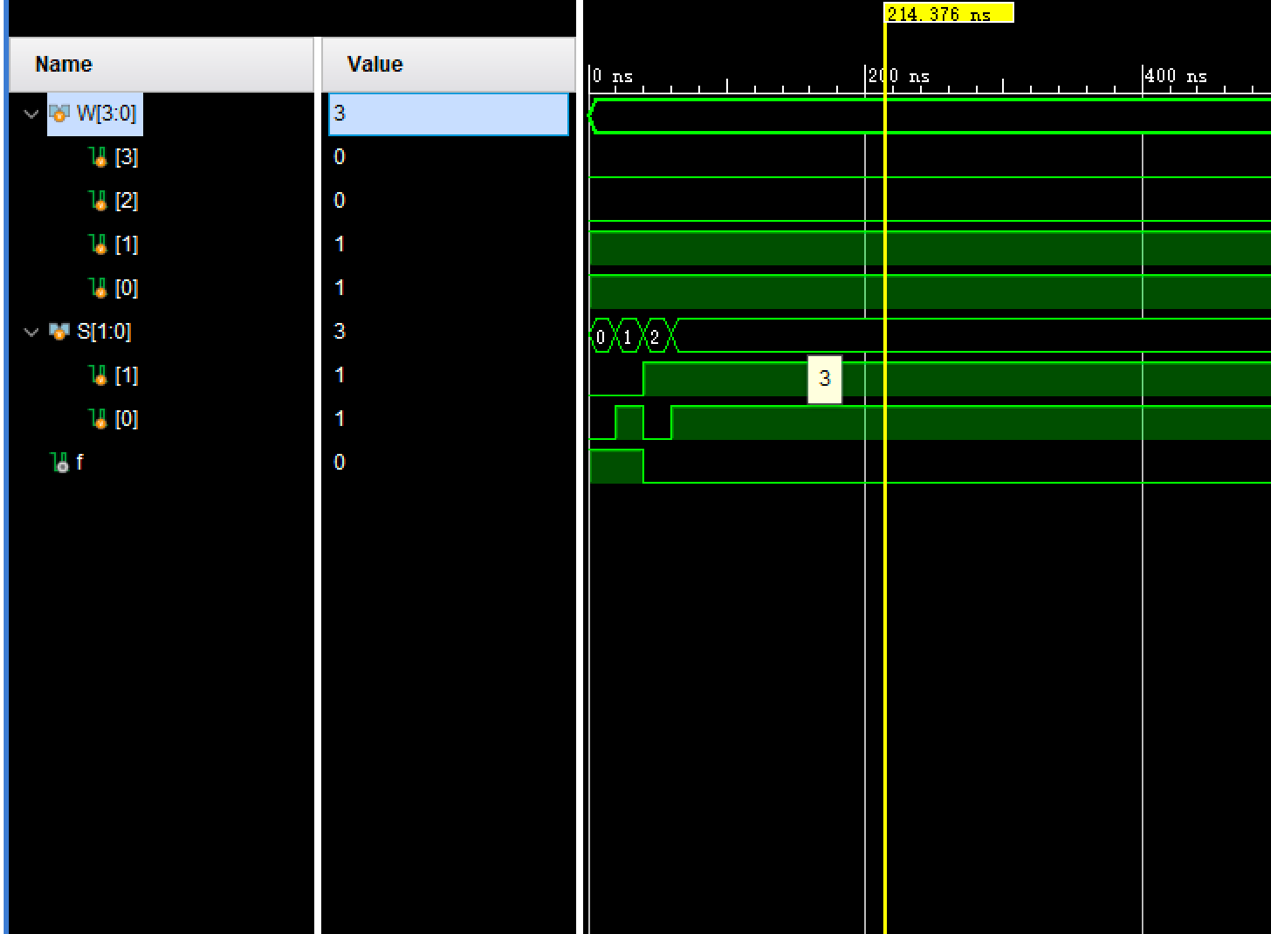
说明：这个四选一选用三个以门电路为设计思想的二选一构成，由s1和s2构成两位二进制选择，当00时，输出a值，当01时，输出b值，当10时，输出c值，当11时，输出d值。

D1 if实现四选一

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_if\_tb;  reg [3:0]W;  reg[1:0]S;  wire f;  initial begin  W=4'b0011;  S=2'b00;  #20S=2'b01;  #20S=2'b10;  #20S=2'b11;  #20W=4'b1010;S=2'b00;  #20S=2'b01;  #20S=2'b10;  #20S=2'b11;  end;  mux\_if mux\_if(.W(W),.S(S),.f(f));  endmodule |

仿真结果图



四选一if仿真截图

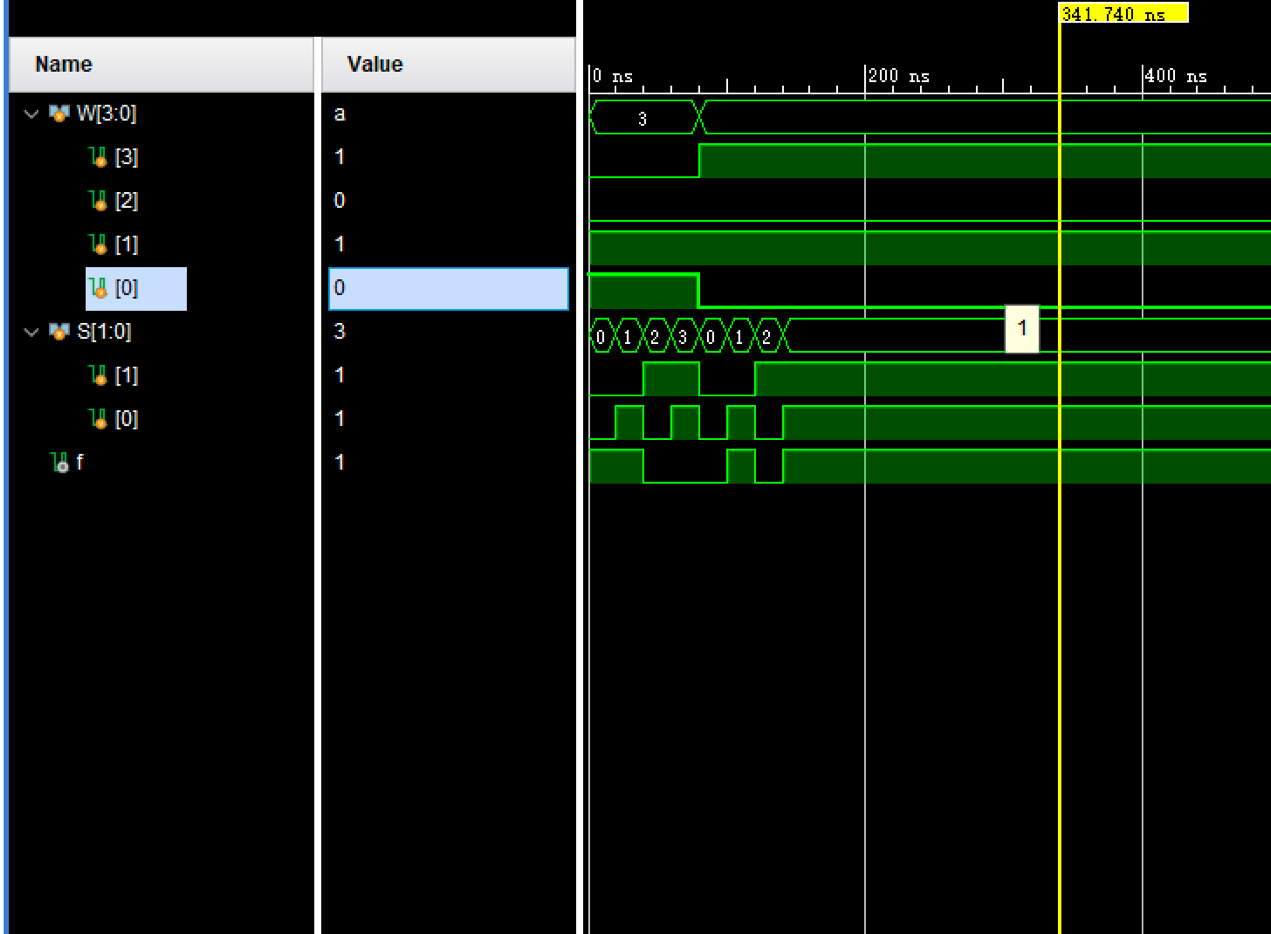
说明：在激励文件中将被选择数据W数组设为0011，而二位选择器S数组随时间变化，则输出f也会随着变化，相应的，当S【0】【1】分别为00，01，10，11时，会输出W【0】，W【1】，W【2】，W【3】的值。

D1 case实现四选一

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module mux\_case\_tb;  reg [3:0]W;  reg[1:0]S;  wire f;  initial begin  W=4'b0011;  S=2'b00;  #20S=2'b01;  #20S=2'b10;  #20S=2'b11;  end;  mux\_case mux\_case(.W(W),.S(S),.f(f));  endmodule |

仿真结果图



四选一case仿真截图

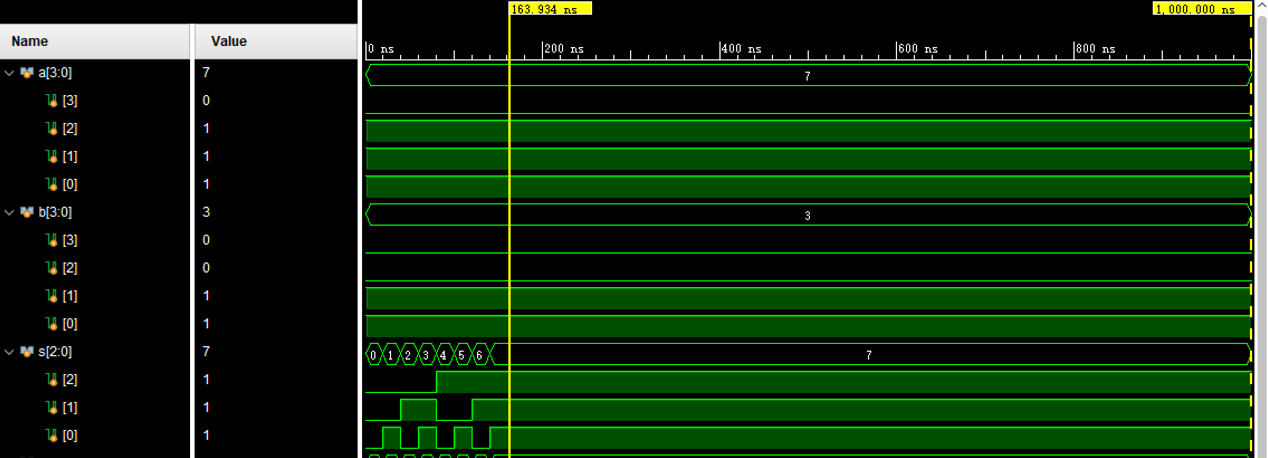
说明：与if 不同的是，我将case中的激励文件中的W数组改为0101，S数组也相应的多做了些改变，则波形图变化如此，整体思维不变。

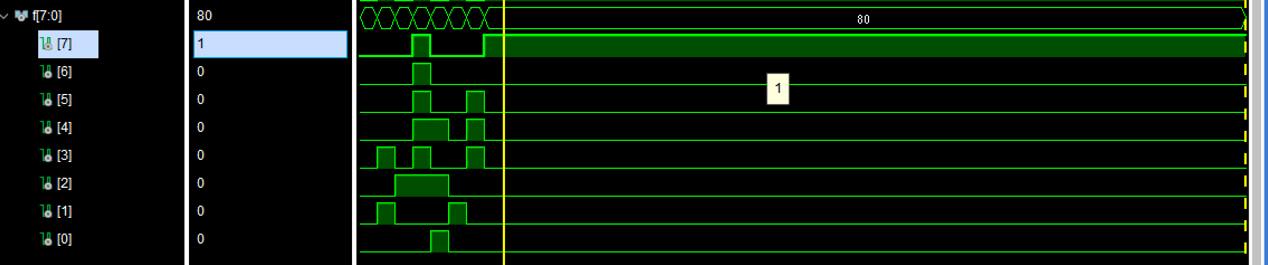
D2ALU计算

激励文件代码

|  |
| --- |
| `timescale 1ns / 1ps  module alu\_tb;  reg [3:0]a;  reg [3:0]b;  reg [2:0]s;  wire[7:0]f;  initial begin  a=4'b0111;  b=4'b0011;  s=3'b000;  #20s=3'b001;  #20s=3'b010;  #20s=3'b011;  #20s=3'b100;  #20s=3'b101;  #20s=3'b110;  #20s=3'b111;  end  alu alu(s,a,b,f);  endmodule |

仿真结果图

****

****

alu运算器仿真结果图

说明：在激励文件中，将a b分别赋值为0111和0011，在改变s的过程中，分别实现运算器的加减乘除以及移位功能。

实验结论：

a）Vivado的设计流程

1、创建新项目或打开已有项目；

2、在创建新项目的向导中创建有HDL资源综合设计的项目（定义项目名称和位置、选中RTL项目）；

3、创建源文件；

4、创建仿真源文件，进行仿真，检查结果是否符合逻辑设计；

5、添加约束，对输入输出信号添加管脚约束，以便下入板中；

6、进行综合（“Run Synthesis”）；

7、综合完成后进行实现（“Run Implementation”）；

8、实现完成后，生成比特流文件用于下载到 FPGA（“Generate Bitstream”）；

9、连接板卡，在板卡上上实现项目的展示。

b）网表

网络表是对自己设计的一种描述，包括单元、引脚、端口和网络。单元是设计目标，包括用户模块/设计内容的实例表达、库单元（BEL）的实例（LUT, FF, RAM, DSP单元等… ）、硬件功能的通用技术表达、黑盒；引脚是单元的连结点；端口是设计的顶层端口；网络形成引脚之间及引脚到端口的连结。

c）什么是约束文件？通过 IO planning 完成的是什么方面的约束？

约束文件将verilog中定义的端口号与FPGA板子上的IO口建立起联系，也同样是告诉软件该如何分配定义的端口号以生成对应的bit文件。约束分类有三种，管脚约束、区域约束和时序约束。

管脚约束：将模块的端口和FPGA的管脚对应；

区域约束：将模块放置在FPGA的特点位置；

时序约束：对数据建立、保持时间进行约束，保证设计在高速时钟下的工作可靠性等。

IO planning完成的是管脚方面的约束。

d） Vivado 设计流程中，Synthesis 的作用是什么？

把源代码综合为对应的门级网表。

e） Vivado 设计流程中， Implementation 的作用是什么？

把门级网表布局布线到芯片上最终实现。