锁存器和触发器

简介

时序电路是数字电路，其输出不仅取决于当前输入（如组合电路），还取决于过去的输入序列。实际上，这些电路必须能够记住历史输入。 因此，我们引入时序概念，时钟信号保障了时序电路按照时序来运行。锁存器和触发器是时序电路中常用的存储器设备。请参阅Vivado教程，了解如何使用Vivado工具创建项目和验证数字电路。

目标

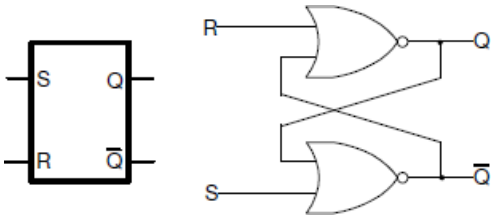
在本次实验中，你将会学到：

- 模拟各种类型的锁存器。
- 使用带有控制信号的触发器。

锁存器 Part 1

存储元件可以分为锁存器和触发器。锁存器是一种具有两种稳定状态的器件：高输出和低输出。 锁存器具有反馈路径，因此设备可以保留信息。 锁存器是易失性存储器设备，并且只要设备通电就可以存储一位数据。 顾名思义，锁存器用于“锁定”信息并保持信息。

SR锁存器（置位/复位）是一个异步器件：它不依赖于控制信号工作，仅依赖于S和R输入的状态。它的符号表示、使用异或门级电路实现和真值表如下所示：



S	R	Q	$\overline{Q}$
0	0	Latch	
0	1	0	1
1	0	1	0
1	1	Metastable	

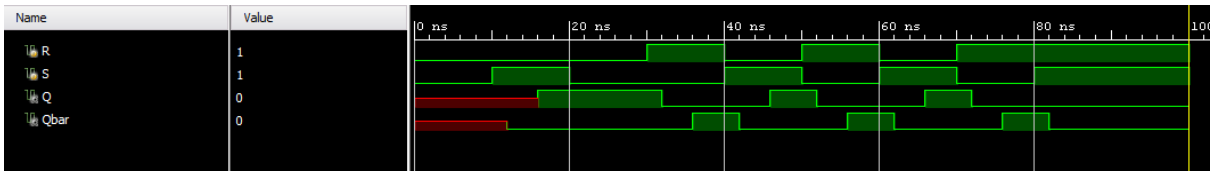
虽然Xilinx FPGA可以使用一个LUT（查找表）电路实现这种锁存器，但以下Verilog代码显示了如何使用门级电路和数据流建模对这种电路进行建模。

```
module SR_latch_gate (input R, input S, output Q, output Qbar);
nor (Q, R, Qbar);
nor (Qbar, S, Q);
endmodule


module SR_latch_dataflow (input R, input S, output Q, output Qbar);
assign #2 Q_i = Q;
assign #2 Qbar_i = Qbar;
assign #2 Q = ~ (R | Qbar);
assign #2 Qbar = ~ (S | Q);
endmodule
```

1-1. 使用上面显示的代码设计SR锁存器。 合成设计并查看合成设计的原理图。 设计一个Testbench进行测试（参见下面的波形）并验证设计。验证该设计。

该实验中将S输入分配给SW0，将R输入分配给SW1。 将Q分配给LED0，将Qbar分配给LED1。实现该设计并进行仿真验证以及下载。



1-1-1. 打开Vivado并创建一个名为lab5_1_1的空白项目。

1-1-2. 使用SR_latch_dataflow代码创建和添加Verilog模块。

1-1-3. 设计一个Testbench进行测试（参见上面的波形），执行100ns的行为仿真，并验证设计。

1-1-4. 将适当的板相关主XDC文件添加到项目中并编辑它以包括相关引脚，将S输入分配给SW0，将R输入分配给SW1，将Q分配给LED0，将Qbar分配给LED1。

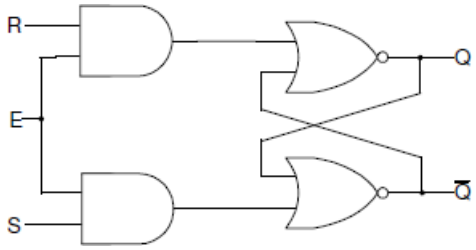
1-1-5. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

1-1-6. 合成设计并在Synthesized Design过程组下查看原理图。验证它使用2个LUT和4个IO（2个IBUF和2个OBUF）。

1-1-7. 实现设计并查看项目摘要。 它应该显示2个LUT和4个IO。

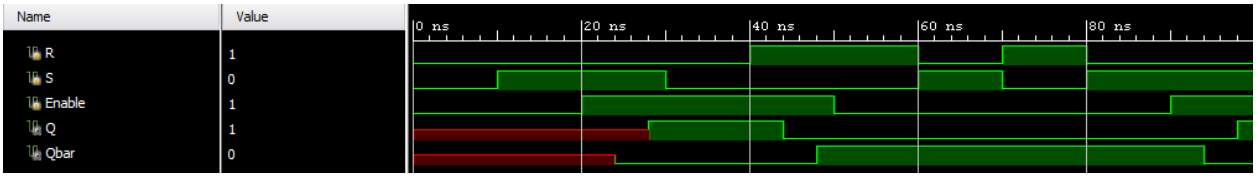
1-1-8. 生成比特流，将其下载到Basys3或Nexys4 DDR板，并验证功能。

在某些情况下，可能需要指示锁存器何时需要锁存数据和何时不能锁存数据。这就引入了门控SR锁存器， 门控SR锁存器是SR锁存器的简单扩展，它提供一条使能线，必须将其置为高电平才能够改变锁存器中锁存的数据。 但即使现在有了控制线，SR锁存器也不是同步的，因为即使在使能处于高电平时，输入可以随时改变输出。 当使能输入为低电平时，AND门的输出也为低电平，因此Q和条Q输出保持锁存到先前的数据。 只有当Enable输入为高电平时，锁存器的状态才会发生变化，如真值表所示。 当使能线被置高电平时，门控SR锁存器在操作上与SR锁存器相同。 Enable线有时是时钟信号，但通常是读或写控制线。门控SR锁存器的符号、电路和真值表如下所示。



Enable	S	R	Q	Q̄
0	0	0	Latch	
0	0	1	Latch	
0	1	0	Latch	
0	1	1	Latch	
1	0	0	Latch	
1	0	1	0	1
1	1	0	1	0
1	1	1	Metastable	

1-2. 使用数据流建模设计门控SR锁存器（如上图所示）。 合成设计并查看合成设计的原理图。 设计一个Testbench进行测试（生成如下所示的输入）并验证设计。仿真设计。 将S输入分配给SW0，将R输入分配给SW1，并将输入分配给SW2。将Q分配给LED0，将Qbar分配给LED1。 实现设计并进行仿真和下载。



1-2-1. 打开Vivado并创建一个名为lab5_1_2的空白项目。

1-2-2. 创建并添加Verilog模块，该模块将使用数据流建模对门控SR锁存器进行建模。 为模型中使用的每个赋值语句分配2个单位延迟。

1-2-3. 仿真100ns。

1-2-4. 将适当的电路板相关主XDC文件添加到项目中并对其进行编辑以包括相关引脚，将S输入分配给SW0，将R输入分配给SW1，将启用分配给SW2，将Q分配给LED0，将Qbar分配给LED1。

1-2-5. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

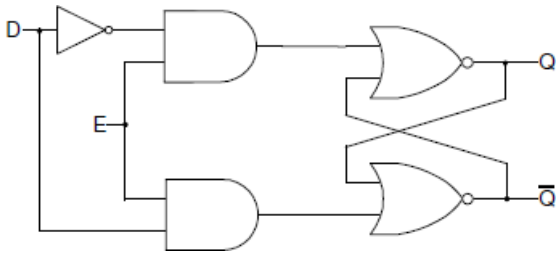
1-2-6. 合成设计并在Synthesized Design过程下查看原理图。验证它使用2个LUT和5个IO

1-2-7. 实现设计并查看项目摘要。 它应该显示2个LUT和5个IO。

1-2-8. 设计Testbench以仿真和验证设计。 它应产生的输入如上面图中。

1-2-9. 生成比特流，将其下载到Basys3或Nexys4 DDR板，并验证功能。

D锁存器（D源于”data”），也叫作透明锁存器，是门控SR锁存器的简单扩展，其消除了无效输入状态（亚稳态）的可能性。 由于门控SR锁存器允许我们在不使用S或R输入的情况下锁存输出，我们可以通过使用互补驱动器驱动设置和复位输入来移除其中一个输入，即我们移除一个输入并自动使其成为另一个输入的互补输入。 只要Enable线为高电平，D锁存器就会输出D输入，否则输出是当Enable输入为最后一个高电平时D输入的输出。 这就是为什么它也被称为透明锁存器 - 当Enable被置位时，锁存器被称为“透明” - 它信号直接传播通过它，好像它不存在一样。



Enable	D	Q	$\overline{Q}$
0	0	Latch	
0	1	Latch	
1	0	0	1
1	1	1	0

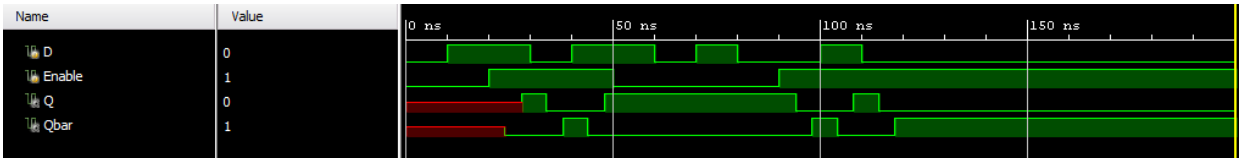
D锁存器可以在行为建模中建模，如下所示：

```
module D_latch_behavior (input D, input Enable, output Q, output Qbar);

always @ (D or Enable)
    if(Enable)
    begin
       Q <= D;
       Qbar <= ~D;
   end
endmodule
```

请注意，由于当Enable为低电平时我们没有做什么，电路会“记住”之前的状态。 当Enable为高电平，由于always块也对D敏感，Q和Qbar将随时更新D，从而使其具有“透明”行为。 另请注意，这里使用非阻塞赋值运算符（<=）代替已在数据流建模中使用的阻塞运算符（=）。 实验7（时序电路的Testbenches）介绍了阻塞和非阻塞分配之间的区别。

1-3. 使用数据流建模设计D锁存器（如上图所示）。合成设计并查看合成设计的原理图。 实现一个Testbench进行测试（生成如下所示的输入）并验证设计。 模拟设计。 将D输入分配给SW0，并将输入分配给SW1。 将Q分配给LED0，将Qbar分配给LED1。 实现设计并验证硬件中的功能。



1-3-1. 打开Vivado并创建一个名为lab5_1_3的空白项目。

1-3-2. 创建并添加将使用数据流建模对D锁存器建模的Verilog模块。 为模型中使用的每个赋值语句分配2个单位延迟。

1-3-3. 实现一个Testbench以测试和验证设计。 它应该产生输入刺激，如上图所示。

1-3-4. 将适当的板相关主XDC文件添加到项目中并编辑它以包括相关引脚，将D输入分配给SW0，将输入分配给SW1，Q分配给LED0，Qbar分配给LED1。

1-3-5. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

1-3-6. 合成设计并在Synthesized Design过程组下查看原理图。验证它使用2个LUT和4个IO。

1-3-7. 实现设计并查看项目摘要。 它应该显示2个LUT和4个IO。

1-3-8. 生成比特流文件，将其下载到Basys3或Nexys4 DDR板，并验证功能。

触发器 Part 2

触发器是时序电路，其输出可以根据其输入在时钟信号的有效边沿上改变。 与锁存器不同，锁存器是透明的，并且当门控信号在输入改变时输出会改变，触发器通常在输入改变时不改变输出，即使时钟信号被改变。 触发器广泛用于同步电路中。

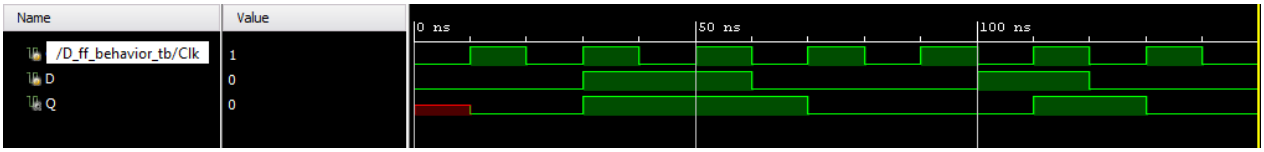
D触发器是一种广泛使用的触发器类型。 它也被称为数据或延迟触发器。 D触发器在时钟周期的某一部分（例如时钟的上升沿）捕获D输入的值。 捕获的值成为Q输出。 在其他时候，输出Q不会改变。 D触发器可以被视为存储器单元或延迟线。 触发器中的有效边沿可以为上升，也可为下降。 下图显示了上升（也称为正）边沿触发D触发器和下降（负边沿）触发D触发器。

可以使用行为建模对正边沿触发的D触发器建模，如下所示：

```
module D_ff_behavior (input D, input Clk, output reg Q);
always @ (posedge Clk)
    if(Clk)
    begin
       Q <= D;
   end
endmodule
```

请注意，always块对Clk信号的上升沿很敏感。 当敏感信号发生更改（事件）时，将执行if块中的语句。posedge灵敏度使触发器行为成为可能。 对于下降沿灵敏度使用属性negedge。

2-1. 使用行为建模为D触发器建模。 开发一个testbench来验证模型（参见下图）。 模拟设计。

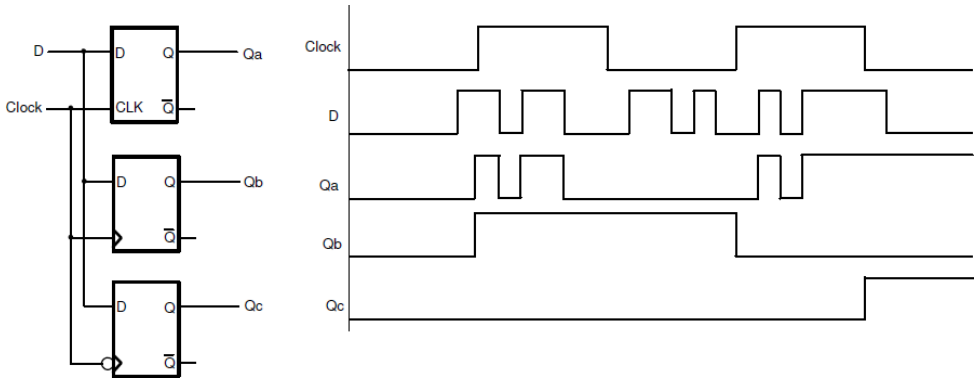


2-1-1. 打开Vivado并创建一个名为lab5_2_1的空白项目。

2-1-2. 创建并添加将模拟简单D触发器的Verilog模块。

2-1-3. 开发testbench以验证设计行为。 它应该产生输入信号，如上面的时序图所示。

以下电路和时序图说明了D锁存器、上升沿触发D触发器和下降沿触发D触发器之间的差异：



2-2. 使用行为建模对电路进行建模，如上所示。 您将使用三个always程序块。 开发一个生成输入的testbench，如上所示。 模拟和验证设计。

2-2-1. 打开Vivado并创建一个名为lab5_2_2的空白项目。

2-2-2. 创建并添加将为给定电路建模的Verilog模块。

2-2-3. 设计testbench以测试和分析设计行为。 它应该产生输入信号，如时序图所示。

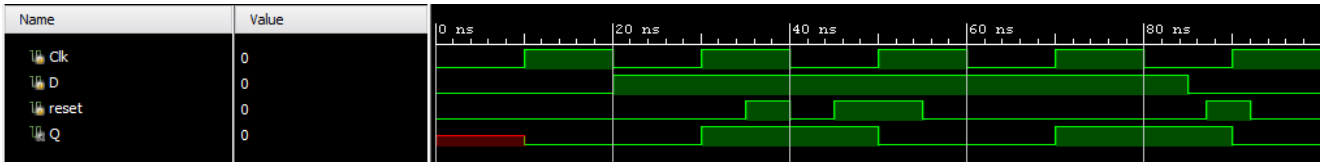
通常需要让同步元素以定义的输出开始。 在某些电路中，还需要强制同步元素到一个已知的输出，而忽略了D信号的输入。 可以修改上面讨论的D触发器以具有这样的功能。 如果在时钟的有效边沿上获得所需的输出，则这种D触发器被称为具有同步置位和复位功能的D触发器，否则它被视为具有异步预置和清除的D触发器。 各种型号如下所示：

```
module D_ff_with_synch_reset_behavior(input D, input Clk, input reset, output
reg Q);       // 同步置位和复位功能的D触发器
   always @(posedge Clk)
      if (reset)
      begin
         Q <= 1'b0;
      end else
      begin
         Q <= D;
```

```
      end
endmodule

module D_ff_with_asynch_reset_behavior(input D, input Clk, input clear,
output reg Q);      //异步预置和清除的D触发器
   always @(posedge Clk or posedge clear)
      if (clear)
      begin
         Q <= 1'b0;
      end else
      begin
         Q <= D;
      end
endmodule
```

2-3. 使用行为建模对具有同步复位的D触发器进行建模。开发一个testbench进行测试（如图所示生成输入）并验证设计。 仿真设计。 将D输入分配给SW0，复位到SW1，将Clk分配给SW15，并将Q输出到LED0。 验证硬件设计。



2-3-1. 打开Vivado并创建一个名为lab5_2_3的空白项目。

2-3-2. 创建并添加Verilog模块，该模块将使用同步复位对D触发器进行建模。

2-3-3. 设计testbench以测试和分析设计行为。 它应该产生输入激励，如时序图所示。

2-3-4. 将适当的板相关主XDC文件添加到项目中并编辑它以包括相关引脚，将D输入分配给SW0，将输入重置为SW1，将Clk重置为SW15，将Q重置为LED0。

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets { clk }];
```

需要将上面的代码行添加到XDC文件中，以允许将SW15用作时钟。

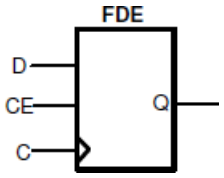
2-3-5. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

2-3-6. 合成并实现设计。

查看项目摘要并注意使用1个BUFG和4个IO。 使用BUFG是因为在设计中使用了时钟信号。

2-3-7. 生成比特流，将其下载到Basys3或Nexys4 DDR板，并验证功能。

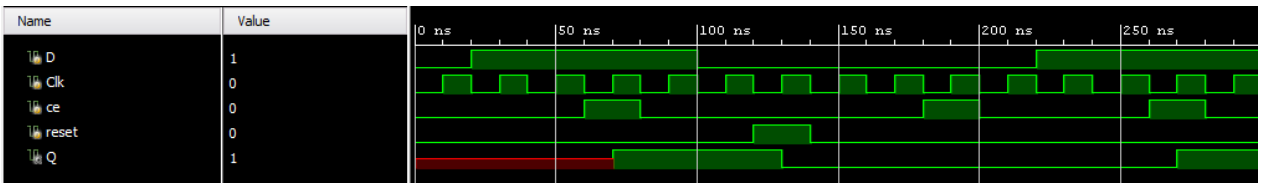
在FPGA中，位于不同可配置逻辑块（CLB）中的LUT和FF使用路由资源连接。在实施过程中，工具将使用这些资源，具体取决于电路的建模方式，所需资源的类型和数量，以及电路的驱动速度。通常用于交换信息的资源是彼此靠近的;但是，可能存在可能无法实现的情况。当相关的触发器（信息交换之间）彼此远离时，到达源触发器和目标触发器的时钟可能不会同时产生所谓的时钟偏移。时钟偏移可以改变电路的行为。在某些其他情况下，某些触发器可能不需要在每个断言的时钟边沿更新其输出。为了控制行为，FPGA中的触发器有一个额外的控制信号，称为时钟使能（CE）。在ASIC技术中，门控时钟用于控制行为。带有CE的触发器的符号如下所示：




```
module D_ff_with_ce_behavior(input D, input Clk, input ce, output reg Q);
   always @(posedge Clk)
      if (ce)
         Q <= D;
endmodule
```

```
module D_ff_with_ce_and_synch_reset_behavior(input D, input Clk, input reset,
input ce, output reg Q);
   always @(posedge Clk)
      if (reset)
      begin
         Q <= 1'b0;
      end else if (ce)
      begin
         Q <= D;
      end
endmodule
```

2-4. 使用行为建模对具有同步复位和时钟使能的D触发器进行建模。 开发一个testbench进行测试（如图所示生成输入）并验证设计。仿真设计。 将D输入分配给SW0，复位到SW1，ce到SW2，Clk到SW15，并将Q输出到LED0。 验证硬件设计。



2-4-1. 打开Vivado并创建一个名为lab5_2_4的空白项目。

2-4-2. 创建并添加Verilog模块，该模块将使用同步复位和时钟使能对D触发器进行建模。

2-4-3. 开发testbench以测试和分析设计行为。 它应该产生输入激励，如上面的时序图所示。

2-4-4. 将适当的板相关主XDC文件添加到项目中并编辑它以包括相关引脚，将D分配给SW0，reset为SW1，ce分配给SW2，Clk分配给SW15，Q分配给LED0。

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets { clk }];
```

需要将上面的代码行添加到XDC文件中，以允许将SW15用作时钟。

2-4-5. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

2-4-6. 合成并实现设计。

查看项目摘要并注意使用了1个BUFG和5个IO。 使用BUFG是因为在设计中使用了时钟信号。

2-4-7. 生成比特流，将其下载到Basys3或Nexys4 DDR板，并验证功能。

在数字电路中，另一种称为T或Toggle的触发器用于实现时钟分频器电路。它可用于将输入信号频率变为原来的1/2.如果级联n个T触发器，则时钟分频为原来的(1/2)^n。 T触发器具有T输入（数据），时钟输入，以及可选的复位和使能控制信号。

```
module clock_divider_behavior(input Clk, output reg Q);
   always @(negedge Clk)
         Q <= ~Q;
endmodule
```

T触发器还可以具有称为CE（时钟使能）的控制信号，其将允许时钟分频仅在其时钟边沿时发生。以下代码模拟了T触发器的功能，该触发器对时钟的下降沿敏感，并具有低电平有效复位和高电平有效T控制信号。

```
module T_ff_enable_behavior(input Clk, input reset_n, input T, output reg Q);
   always @(negedge Clk)
      if (!reset_n)
         Q <= 1'b0;
      else if (T)
        Q <= ~Q;
endmodule
```

2-5. 使用上述代码建模具有同步负逻辑复位和时钟使能的T触发器。 将T输入分配给SW0，将reset_n分配给SW1，将Clk分配给SW15，并将Q输出到LED0。 验证硬件设计。

2-5-1. 打开Vivado并创建一个名为lab5_2_5的空白项目。

2-5-2. 创建并添加Verilog模块，该模块将使用低电平同步复位（reset_n）对T触发器进行建模。

2-5-3. 将适当的板相关主XDC文件添加到项目中并编辑它以包括相关引脚，将T输入分配给SW0，将reset_n输入分配给SW1，将Clk分配给SW15，将Q分配给LED0。

```
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets { clk }];
```

需要将上面的代码行添加到XDC文件中，以允许将SW15用作时钟。

2-5-4. 在“Bitstream Settings”中设置tcl.pre选项以指向提供的lab5_prehook.tcl文件。 此文件允许将组合循环上载到电路板。

2-5-5. 合成并实现设计。

2-5-6. 生成比特流，将其下载到Basys3或Nexys4 DDR板，并验证功能。

总结

在本实验中，您学习了各种锁存器和触发器的功能。 您建模并验证了这些组件的功能。 Xilinx还提供了一些基本的锁存器和触发器库组件，设计人员可以实例化和使用它们而不是编写模型。 编写模型可以提供跨供应商和技术的可移植性，而实例化库组件可以快速使用组件而无需重新发明轮子。