实验五：触发器和寄存器

实验时间： 2017年11月27日 第十二周 星期一
实验者：16307130194 陈中钰 16级 计算机科学技术学院
座位号：30
指导老师：唐志强

1 实验目的

- 了解并掌握时序电路的基本器件触发器的设计，包括 D 触发器、T 触发器、JK 触发器
- 了解并掌握时序电路中的异步清零、异步置数功能的实现，知道异步操作的作用和必要性
- 熟练掌握各种触发器的特征方程，能实现触发器之间的转换
- 熟悉触发器的功能和实现，能将触发器应用于功能器件的设计和实现
- 了解计数器的工作原理，并设计4位二进制计数器
- 加强对时序电路的理解，以及设计时序电路的能力

2 实验原理

2.1 异步操作

- always@()语句的敏感表中，除了有 posedge clk 以外，异步控制开关也应该是敏感项，即当异步开关电平值改变时，也应该触发
- 异步操作优先级高于同步操作，可以通过 if else 来控制
- 当异步置零为1时，q<=0；当异步置数为1时，q<=1
- always@()敏感表中不能同时有上升（下降）沿触发和电平触发，因此只能把电平触发也改成上升（下降）沿触发
- 异步操作控制表

reset	myset	功能
0	0	正常运行
0	1	异步置数
1	X	异步清零

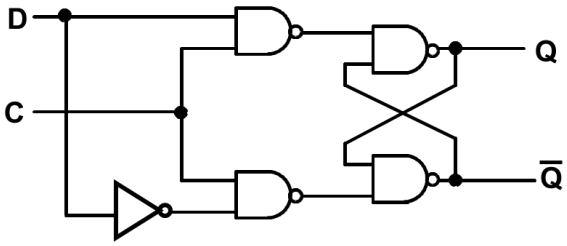
2.2 D 触发器

- 输入 D，状态 Q(n)，下一状态 Q(n+1)
- 状态表

D	Q(n)	Q(n+1)	操作
0	0	0	复位
0	1	0	
1	0	1	置位
1	1	1	

可得 Q=D

- 逻辑电路图



2.3 JK 触发器

- 输入 J，K，状态 Q(n)，下一状态 Q(n+1)
- 状态表

J	K	Q(n)	Q(n+1)	操作
0	0	0	0	无变化
0	0	1	1	
0	1	0	0	复位
0	1	1	0	
1	0	0	1	置位
1	0	1	1	
1	1	0	1	翻转
1	1	1	0	

可得，当 JK=00时保持 q<=q；为01时复位 q<=0；为10时置位 q<=1；为11时翻转 q<=~q

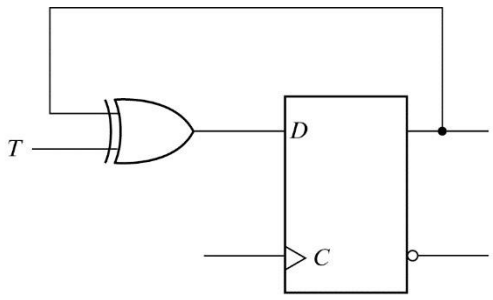
- 通过 case()语句控制，逻辑更明晰，更容易检验正误

2.4 T 触发器

- 输入 T，状态 Q(n)，下一状态 Q(n+1)
- 状态表

T	Q(n)	Q(n+1)	操作
0	0	0	无变化
0	1	1	
1	0	1	翻转
1	1	0	

- 电路图



2.5 D 触发器转换成 JK 触发器

- 综合 JK 触发器、D 触发器的状态表，列出包含 J，K，D 的状态表

J	K	Q(n)	Q(n+1)	D	操作
0	0	0	0	0	无变化
0	0	1	1	1	
0	1	0	0	0	复位

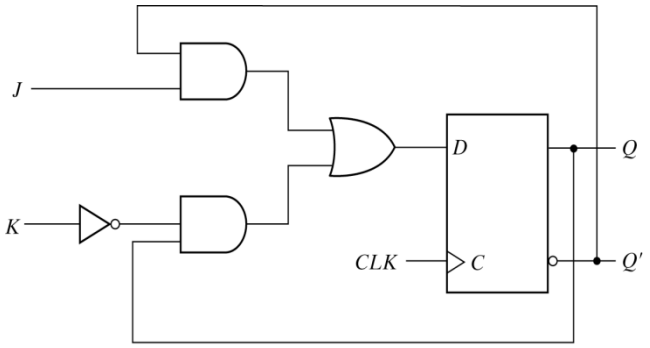
0	1	1	0	0	
1	0	0	1	1	置位
1	0	1	1	1	
1	1	0	1	1	翻转
1	1	1	0	0	

- 画 J、K、Q 的 K-Map，求出 D

Q \ JK	00	01	11	10
0			1	1
1	1			1

可得 D=JQ'+K'Q

- 那么将 D 触发器的输入 D 改为 JQ'+K'Q，就能构造出 JK 触发器
- 逻辑电路图

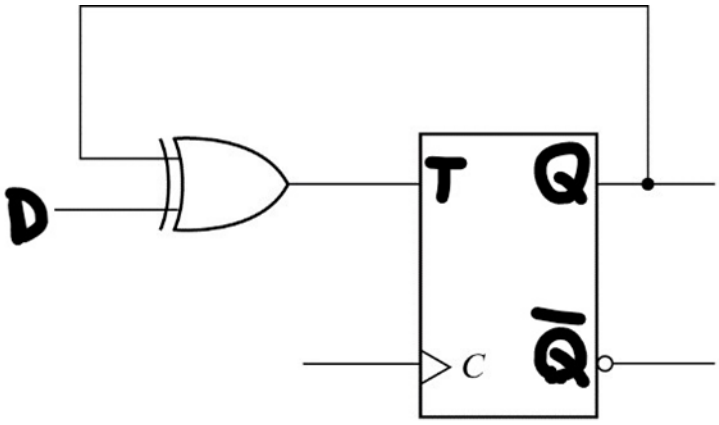


2.6 T 触发器转换成 D 触发器

- 综合 T 触发器、D 触发器的状态表，列出包含 T，D 的状态表

D	Q(n)	Q(n+1)	T	操作
0	0	0	0	复位
0	1	0	1	
1	0	1	1	置位
1	1	1	0	

- 观察可得 T= D^Q
- 那么将 T 触发器的输入 T 改为 D^Q，就能构造出 D 触发器
- 逻辑电路图



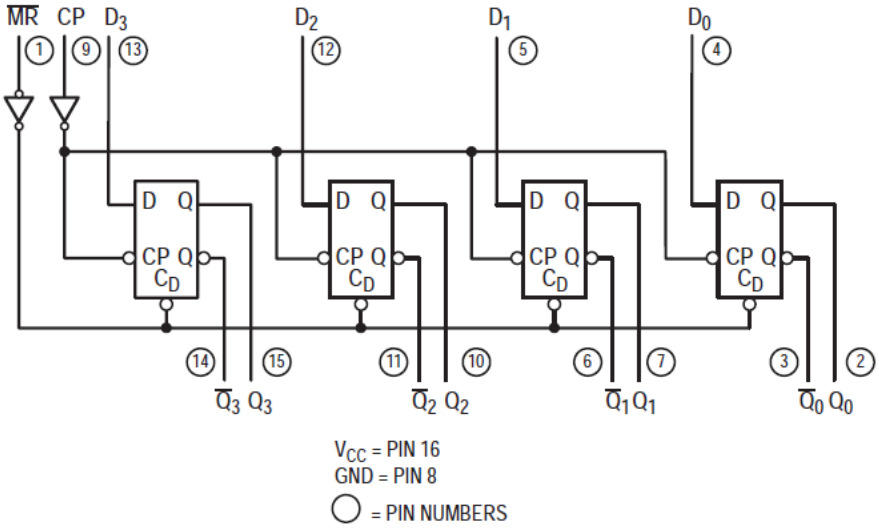
2.7 用 D 触发器构成74LS175

- 性质：4位宽的 D 触发器

- 输入 D[3:0]，当 reset 无效，而且时钟上升沿到达的时候，输出 Q[3:0]被输入更新为 D[3:0]的值
- 通过本实验的 D 触发器模块作为底层模块，各模块共用 CLK 和 reset 信号，可以构成该4位宽 D 触发器
- 其中 reset 信号为低有效（active low），故要在应用 D 触发器模块的时候，输入的 reset 信号要取反
- 状态表

reset	D[3:0]	Q[3:0]	Q[3:0](n+1)	操作
0	XXXX	XXXX	0000	异步清零
1	XXXX	XXXX	D[3:0]	同步置数

- 逻辑电路图

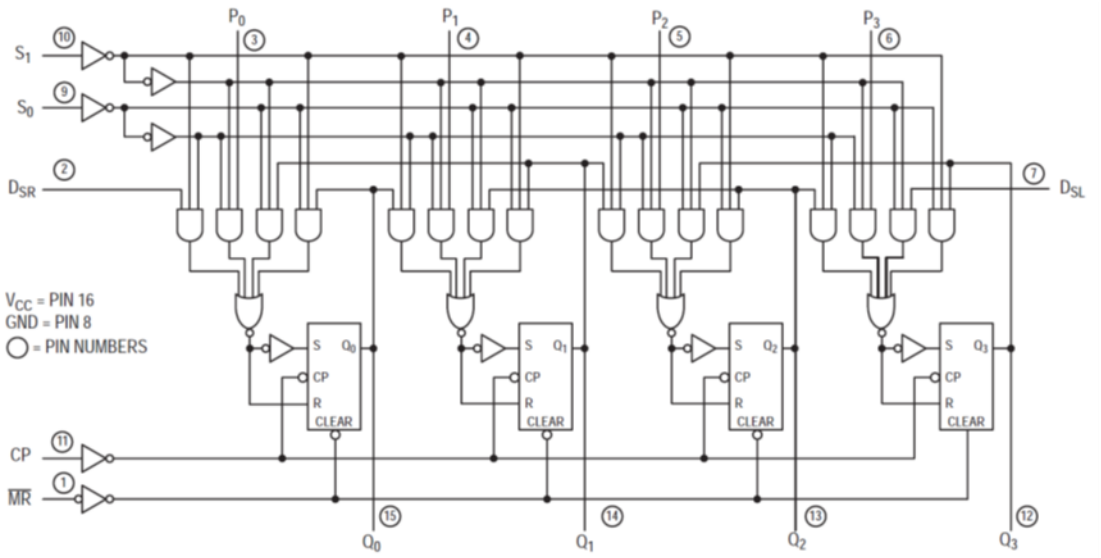


2.8 74LS194双向移位寄存器

- 具有异步清零、数据保持、左移右移、并行置数的功能
- 异步清零优先级最高，当 reset 有效时，输出设置为0000
- 左移时把右三位赋值给左三位，并在最右边补上 Dsl 的值；右移时把左三位赋值给右三位，并在最左边补上 Dsr 的值
- 并行置数时，同步置数，把 Q[3:0]置为输入的 P[3:0]
- 状态表

reset	S1	S2	工作状态
0	X	X	异步清零
1	0	0	数据保持
1	0	1	右移，补 Dsr 的值
1	1	0	左移，补 Dsl 的值
1	1	1	并行置数

- 逻辑电路图



2.9 74LS163 4位二进制计数器

- 状态表

clear	load	P	T	clk	Q	RCO	操作
0	X	X	X	↑	0000	0	清0
1	0	X	X	↑	DCBA	*	并行预设
1	1	1	1	↑	计数	*	加1计数
1	1	0	X	X	Q	*	保持
1	1	X	0	X	Q	0	保持

其中，RCO 保持为0，当且仅当 DCBA=4'b1110，且要加1、将产生进位时，RCO 为1

3 实验内容

3.1 时钟分频

- Vivado 预设的时钟品率为100MHz，太快了，需要分频以减慢变化的速度，减慢到约4Hz
- 分频机制：定义了一个28位长的计数器 q，每次都输出 q[24]的值，而从0开始，每次时钟上升沿时 q 增加1，当2^24次时钟周期后，此时 q=1000…（24个0），q[24]从0变为1，此时输出有效，而且 q[24]=1会保持接下来的2^24次时钟周期，直到 q=1000…（25个0），q[24]从1变回0，无效。也就是说输出的时钟信号的周期是输入信号的周期的2^24倍，实现了降低频率的分频效果
- 取 q[24]约为4Hz，增加值的大小会降低频率，反之提高
- 以下每个触发器都使用了此时钟分频的模块，故之后不再展示代码
- Verilog 代码

```
module clkdiv(input mclk, output clk1_4hz);
    reg [27:0]q;
always@(posedge mclk)
    q<=q+1;
assign clk1_4hz=q[24];
endmodule
```

3.2 带异步操作的 D 触发器

- 通过开关输入 reset，myset，d 信号；通过 LED 输出 q 信号
- reset 操作的优先级最高，其次是 myset，最后才是同步置数
- Verilog 代码

```
module D_flip_flop(input clk, input d, input myset, input reset, output
reg q);
    wire myclk;
    clkdiv c1(clk, myclk);
    always@(posedge myclk,posedge myset,posedge reset)
    begin
        if(reset) q<=0;
        else if(myset) q<=1; else q<=d;
    end
endmodule
```

- 操作：当 reset 有效时，q 被异步置零；当 reset 无效而 myset 有效时，q 被异步置1；均无效时，且时钟上升沿到达时，q 被同步置数为 d

3.3 带异步操作的 JK 触发器

- 通过开关输入 reset，myset，J，K 信号；通过 LED 输出 q 信号
- 优先级：reset > myset > JK
- Verilog 代码

```
module JK_flip_flop(input clk, input myset, input reset, input j, input
k, output reg q);
    wire myclk;
    clkdiv c1(clk, myclk);
    always@(posedge myclk, posedge myset, posedge reset)
    begin
        if(reset) q<=0;
        else if(myset) q<=1;
        else if(j&~k) q<=1;
        else if(k&~j) q<=0;
        else if(k&j) q<=~q;
        else q<=q;
    end
endmodule
```

- 操作：当 reset 有效时，q 被异步置零；当 reset 无效而 myset 有效时，q 被异步置1；均无效时，且时钟上升沿到达时，若 JK=00，则保持，若为01则置零，若为10时则置1，若为11时则取反

3.4 带异步操作的 T 触发器

- 通过开关输入 reset，myset，T 信号；通过 LED 输出 q 信号
- 优先级：reset > myset > T
- Verilog 代码

```
module T_flip_flop(input clk, input t, input myset, input reset, output
reg q);
    wire myclk;
    clkdiv c1(clk, myclk);
```

```
    always@(posedge myclk,posedge myset,posedge reset)
    begin
        if(reset) q<=0;
        else if(myset) q<=1;
        else if(t) q<=~q;
        else q<=q;
    end
endmodule
```

- 操作：当 reset 有效时，q 被异步置零；当 reset 无效而 myset 有效时，q 被异步置1；均无效时，且时钟上升沿到达时，若 T 为0，则 q 保持，若 T 为1，则 q 取反

3.5 D 触发器转换成 JK 触发器

- 通过开关输入 reset，myset，J，K 信号；通过 LED 输出 q 信号
- 按照特征方程 D=JQ'+K'Q，复用 D 触发器即可
- Verilog 代码

```
module JK_flip_flop(input clk, input myset, input reset, input j, input
k, output q);
    D_flip_flop d1(clk,j&~q|~k&q,myset,reset,q);
endmodule
```

- 操作与 JK 触发器一致

3.6 T 触发器转换成 D 触发器

- 通过开关输入 reset，myset，D 信号；通过 LED 输出 q 信号
- 按照特征方程 T=D^Q，复用 T 触发器即可
- Verilog 代码

```
module D_flip_flop(input clk, input d, input myset, input reset, output
q);
    T_flip_flop t1(clk,d&~q|~d&q,myset,reset,q);
endmodule
```

- 操作与 D 触发器一致

3.7 用 D 触发器构成74LS175

- 通过开关输入 reset，myset，d[3:0]；通过 LED 输出 q[3:0]，qn[3:0]
- 复用4个 D 触发器，并共用 CLK，reset 信号（低有效）
- Verilog 代码

```
module Quad_D_flip_flop(input clk, input [3:0] d, input reset, output
[3:0] q, output [3:0] qn);
    D_flip_flop d0(clk,d[0],0,~reset,q[0]);
    D_flip_flop d1(clk,d[1],0,~reset,q[1]);
    D_flip_flop d2(clk,d[2],0,~reset,q[2]);
    D_flip_flop d3(clk,d[3],0,~reset,q[3]);
    assign qn[0]=~q[0]; assign qn[1]=~q[1];
    assign qn[2]=~q[2]; assign qn[3]=~q[3];
endmodule
```

- 操作：当 reset 有效时，q 被异步置零；当 reset 无效而 myset 有效时，q 被异步置1；均无效时，且时钟上升沿到达时，q[3:0]被同步置数为 d[3:0]，并显示在

LED[3:0]上

3.8 74LS194 双向移位寄存器

- 输入 reset 异步清零信号，输入 a[1:0]控制信号，输入异步置数值 data[3:0]，输入左右移补上的值 d[1:0]，输出 q[3:0]，并在 LED[3:0]上显示
- Verilog 代码

```
module Shift_register(input clk, input reset, input [1:0] d, input
[1:0] a, input [3:0] data, output reg [3:0] q);
    wire myclk;
    clkdiv c1(clk,myclk);
    always@(posedge myclk,posedge reset)
    begin
        if(~reset) q[3:0]=4'b0000;
        else
            case(a)
            2'b00:q[3:0]=q[3:0];
            2'b01:q[3:0]={d[1],q[3:1]};
            2'b10:q[3:0]={q[2:0],d[0]};
            2'b11:q[3:0]=data[3:0];
            endcase
    end
endmodule
```

- 操作：当 reset 有效时，异步置零；当 reset 无效且时钟上升沿到达时，若 a 为00时，q 保持，a 为01时右移，且补上左输入数据 d[1]，a 为10时左移，且补上右输入数据 d[0]，a 为11时，q 同步置数为输入数据 data

3.9 74LS163 4位二进制计数器

- 时钟分频函数：为了结果更明显，特意把时钟分频调的更慢，把 assign clk1_4hz=q[24]；改为 assign clk1_4hz=q[26]；那么新频率约为原来的四分之一，即1Hz
- Verilog 代码

```
module binary_counter(input A, input B, input C, input D, input clear,
input load, input P, input T, input clk, output reg [3:0] Q, output reg
RCO);
    wire myclk;
    clkdiv a(clk,myclk);
    always@(posedge myclk)
    begin
        if(!clear) begin Q<=4'b0000; end
        else
            if(!load) begin Q<={D,C,B,A}; end
            else
            case({P,T})
                2'b11:
                    begin
                    if(Q==4'b1110) begin Q<=Q+1; RCO<=1; end
```

```
                else begin Q<=Q+1; RCO<=0; end
                end
            default:Q<=Q;
        endcase
    end
endmodule
```

- 操作：当 reset 为0时，Q 被同步清0；当 reset 为1时，若 PT 不为11，则 Q 保持，若 PT 为11，4位宽二进制计数器 Q 则进行加1计数，并在 Q 为1110并要进行加1计数，要产生进位时，RCO 为1，其他时候 RCO 为0

4 实验结论

- 完成了 D 触发器，JK 触发器，T 触发器的设计，并证明了触发器具有储存的功能
- D 触发器，JK 触发器，T 触发器之间可以通过特征方程进行转换，也就是通过加入若干门对触发器输入信号进行处理，可以获得另一个触发器
- 由多个 D 触发器可以构成一个多位宽的 D 触发器，也就是一个4位的寄存器
- 实现了74LS194双向移位寄存器，实现了数据的储存、左右移位、清0
- 实现了74LS63四位二进制计数器，实现了计数功能

5 实验感想

- 通过利用一些 Verilog 语言的优点，可以大大简化设计（如双向移位寄存器中，可以通过错位赋值直接实现移位操作，而不需死板按照器件手册中的逻辑电路图，一个逻辑门不差地实现，这样会十分复杂、繁琐、难以 debug 和理解）
- D 触发器是很多时序电路的基础，如寄存器、移位器等，需要深入学习理解这一个基本器件
- D 触发器，JK 触发器，T 触发器之间可以实现转换，说明它们的本质其实是一致的，只是由于微小的设计差异而导致有不同的功能