时序逻辑实验报告

17 May 2014

目录

实验四 触发器设计

实验目的：

掌握基本触发器的工作原理；掌握触发器的门级和行为级设计方法。

实验原理：

触发器是最简单、最基本的时序逻辑单元，各种时序逻辑电路，如计数器，通常都是以触发器为基本单元构成的。一种常用触发器为边沿触发器型 D 触发器。

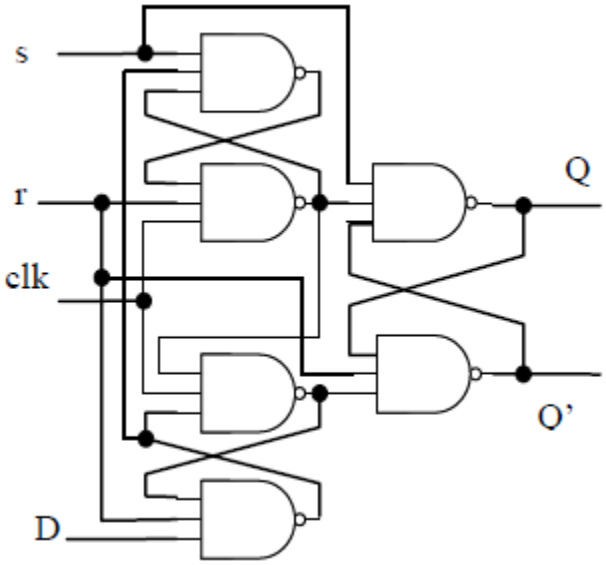
【边沿 D 触发器】

D 触发器状态转移表

D^n	现态 Q^n	次态 Q^{n+1}
0	0	0
	1	
1	0	1
	1	

D 触发器的状态转换表如上表所示。根据状态转换表，可以得到其状态方程为：$Q^{n+1}=D^n$

典型的 D 触发器逻辑图如下图所示，其中 clk 为时钟控制信号，D 触发器在时钟脉冲 clk 的上升沿发生翻转，它的状态仅取决于时钟脉冲上升沿到来之前输入端的取值。其中 s、r 分别是置位端和复位端，都是低电平有效。



关键代码 ：

D 触发器

```
module Dflipflop(s,r,clk,D,Q,Qr);

        input wire s,r,clk,D;

        output wire Q,Qr;

        nand(L1,s,L4,L2);

        nand(L2,L1,r,clk);

        nand(L3,L2,clk,L4);

        nand(L4,L3,r,D);

        nand(Q,s,L2,Qr);

        nand(Qr,Q,r,L3);
```

```
endmodule


module DflipflopDescibe(clk,D,Q,Qr);
        input wire clk,D;
        reg temp;
        output wire Q,Qr;
        always@(posedge clk)
        begin
                temp<=D;
        end
        assign Q=D;
        assign Qr=~Q;
endmodule
```

移位寄存器

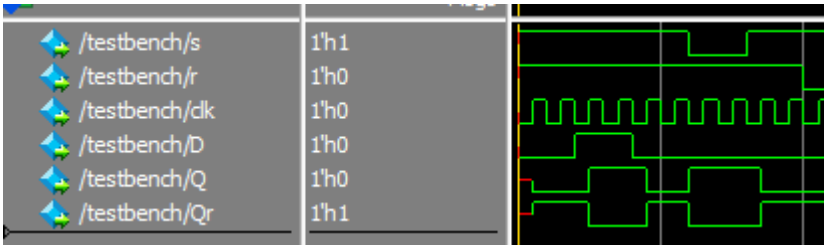
```
module ShiftRegister(clk,D,Q);
        input wire clk,D;
        output wire [3:0]Q;
        wire temp[3:0];
        Dflipflop u0(1,1,clk,D,temp[0],);
        Dflipflop u1(1,1,clk,temp[0],temp[1],);
        Dflipflop u2(1,1,clk,temp[1],temp[2],);
```

```
    Dflipflop u3(1,1,clk,temp[2],Q[3],);

    assign Q[0]=temp[0];

    assign Q[1]=temp[1];

    assign Q[2]=temp[2];

endmodule
```

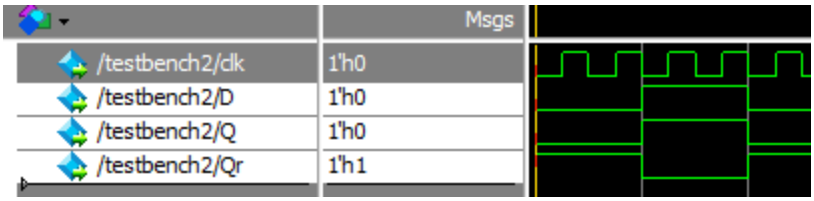
文件清单

Dflipflop.v	D 触发器
ShiftRegister.v	移位寄存器
testbench.v	testbench

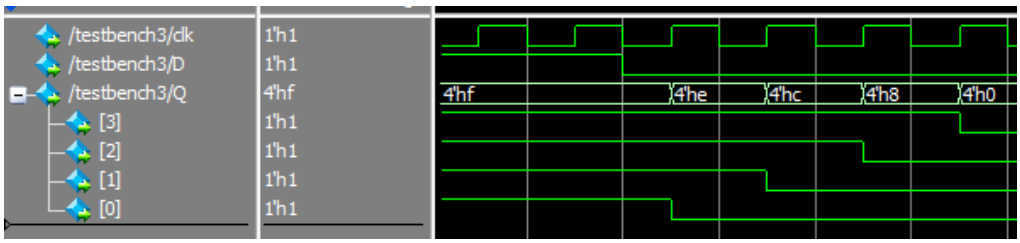
仿真结果



D 触发器门级设计可以正常工作。



D 触发器行为级设计可以正常工作。



移位寄存器可以正常工作。

综合情况

D 触发器门级设计：	Total logic elements	6
	Worst-case	11.009 ns
D 触发器行为及设计：	Total logic elements	0
	Worst-case	4.791 ns
移位寄存器：	Total logic elements	12

硬件调试情况

调试顺利，未遇到困难。

实验五 计数器设计

实验目的：

- 掌握简单时序逻辑电路的设计方法；
- 理解同步计数器和异步计数器的原理；
- 了解任意进制计数器的设计方法。

实验原理：

计数器是一种常用的时序电路，它按照规定的方式改变内部各触发器的状态，以记录输入的时钟脉冲的个数。按照规定的计数顺序的不同，计数器可以分为加法计数器、减法计数器、可逆计数器和不同进制的计数器；按照工作方式的不同，又可以分为异步计数器和同步计数器。

以二进制计数器为例，加法计数器在计数脉冲依次输入时，相应的二进制数据是依次增加的。表 1 给出了 4 位加法计数器的功能表。可以看出，每来一个计数脉冲，最低位 QA 的状态变化一次，其后各位则在低一位触发器的状态由 1 变为 0 时发生状态变化。这样，利用低一位的反相输出作为高一位的时钟输入，便可以构成加法计数器。

减法计数器的计数规律与加法计数器相反，每来一个计数脉冲，计数器的值是减一。利用 D 触发器可以方便地构成减法计数器，与加法计数器不同的是，减法计数器利用低一位的输出作为高一位的时钟输入，而加法计数器则是利用低一位的反相输出作为高一位的时钟输入。

关键代码

加法计数器

```
module postCounter(clk,r,Q);
        input wire clk,r;
        output wire [3:0]Q;
        wire [3:0]self;
        wire [3:0]temp;
        assign self[0]=temp[0];
        assign self[1]=temp[1];
        assign self[2]=temp[2];
        assign self[3]=temp[3];
        DflipflopDescibe a(clk    ,r,temp[0],Q[0],temp[0]);
        DflipflopDescibe b(temp[0],r,temp[1],Q[1],temp[1]);
        DflipflopDescibe c(temp[1],r,temp[2],Q[2],temp[2]);
        DflipflopDescibe d(temp[2],r,temp[3],Q[3],temp[3]);
endmodule
```

减法计数器

```
module negCounter(clk,r,Q);
```

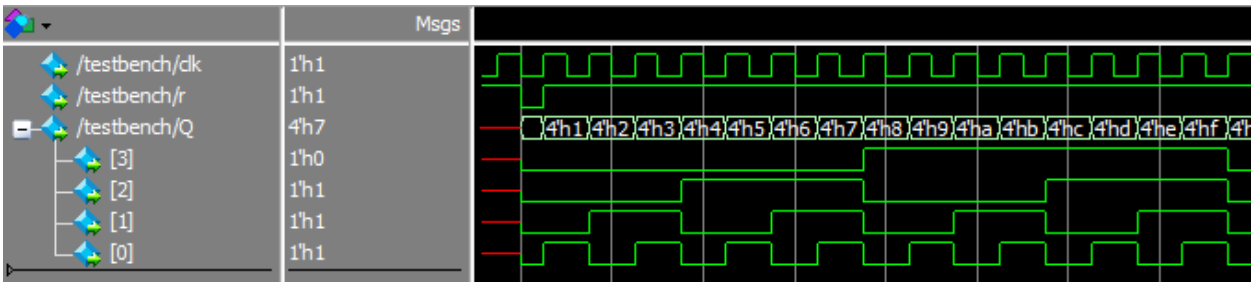
```
    input wire clk,r;
    output wire [3:0]Q;
    wire [3:0]self;
    wire [3:0]temp;
    assign self[0]=temp[0];
    assign self[1]=temp[1];
    assign self[2]=temp[2];
    assign self[3]=temp[3];
    DflipflopDescibe a(clk ,r,temp[0],Q[0],temp[0]);
    DflipflopDescibe b(Q[0],r,temp[1],Q[1],temp[1]);
    DflipflopDescibe c(Q[1],r,temp[2],Q[2],temp[2]);
    DflipflopDescibe d(Q[2],r,temp[3],Q[3],temp[3]);
endmodule
```

文件清单

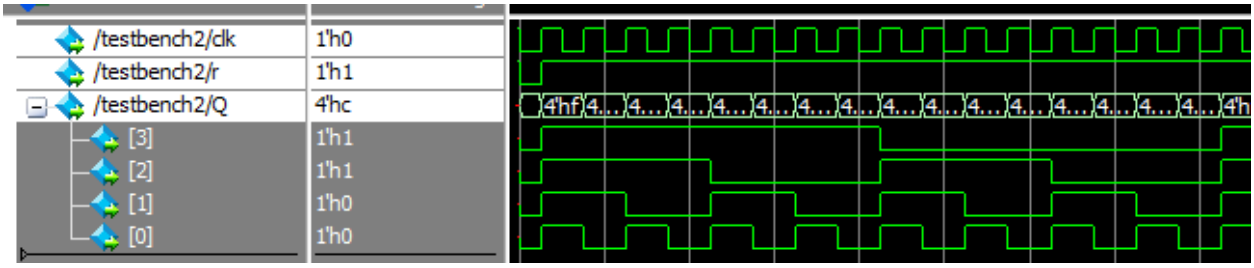
Counter.v　　计数器

testbench.v　testbench

仿真情况



加法计数器可以正常工作



减法计数器可以正常工作

综合情况

加法计数器	Total logic elements	4
	Worst-case	9.186 ns
加法计数器	Total logic elements	4
	Worst-case	9.186 ns

硬件调试情况

调试顺利，未遇到困难。

实验六 累加器设计

实验目的：

掌握时序逻辑电路的基本分析方法和设计方法；

理解累加器的工作原理，用硬件描述语言实现指定功能的累加器设计。

实验原理：

累加器可以实现多个数据的累积相加求和，图 14 给出了其结构图。在累加器工作前，将累加寄存器清零；在时钟信号的控制下，寄存器数据与输入数据相加，结果送入寄存器保存。重复这个过程，直到用户停止相加，从而实现了多个数据的累加求和。

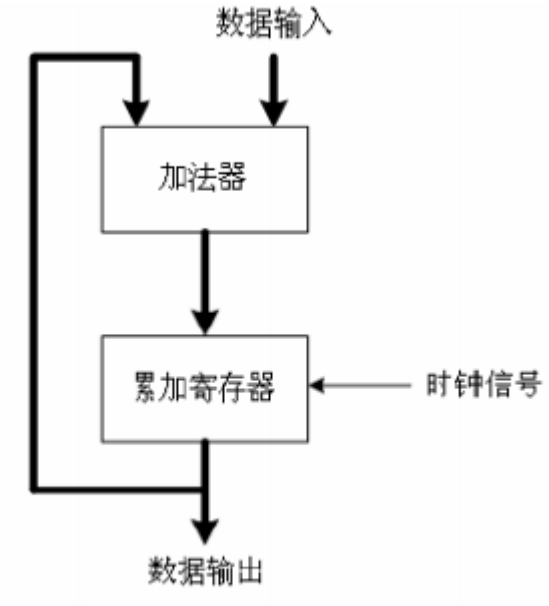


图 1 4 位二进制加法计数器

关键代码

累加器

```
module accumulator(A, clk,S,rst,ena);

        input wire [3:0]A;

        output wire [3:0]S;

        input wire clk,rst,ena;
```

```
reg [3:0]r=0;
wire [3:0]inner;
SuperAdder(A,r,0,inner,);
always@(posedge clk or negedge rst)
begin
    if(rst==1'b0)
    begin
        r[0]<=0;
        r[1]<=0;
        r[2]<=0;
        r[3]<=0;
    end
    else if(clk==1'b1 && ena==1'b1)
    begin
        r[0]<=inner[0];
        r[1]<=inner[1];
        r[2]<=inner[2];
        r[3]<=inner[3];
    end
end
```

```
        assign S[0]=r[0];

        assign S[1]=r[1];

        assign S[2]=r[2];

        assign S[3]=r[3];


endmodule
```

乘法器

```
module Multiplier(A,B,S,clk,rst,enable);

        input wire [1:0]A;

        input wire [1:0]B;

        output wire [3:0]S;

        input wire clk,rst,enable;

        reg ena;

        wire [3:0]Q;

        accumulator (B, clk,S,rst,ena||enable);

        postCounter (clk,rst,Q);

        always@(clk or Q or rst)

        begin

                if(rst==1'b0) ena<=1;

                else if(ena==1'b1) ena<=((A[0]^Q[0]) || (A[1]^Q[1]));

        end
```

```
endmodule
```

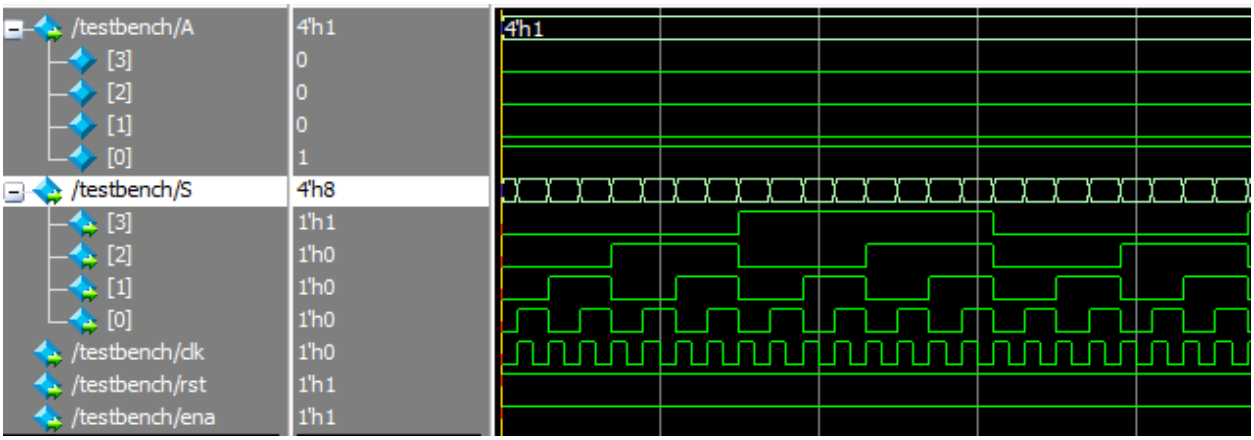
文件清单

accumulator.v　　累加器

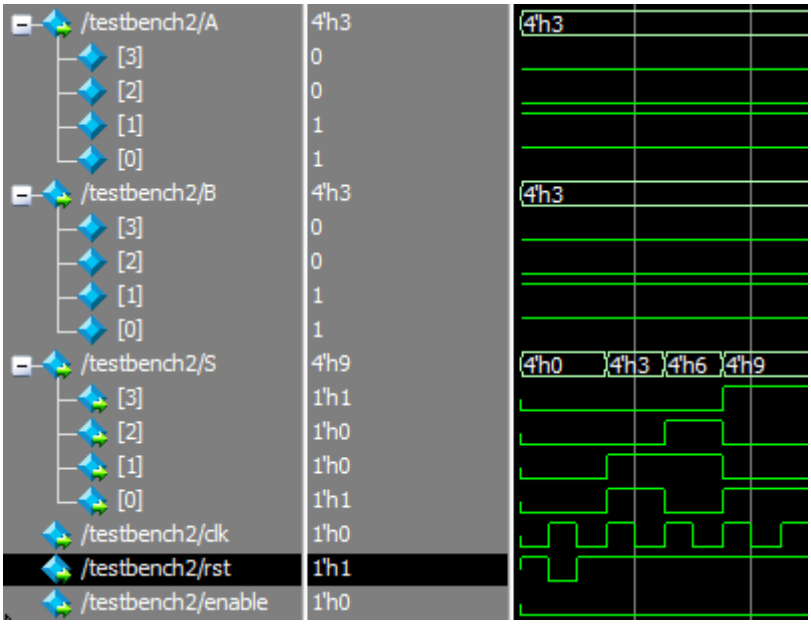
Multiplier.v　　乘法器

testbench.v　　testbench

仿真结果



累加器能正常工作。



乘法器能正常工作。

综合情况

累加器	Total logic elements	7
	Worst-case	6.528 ns
乘法器	Total logic elements	12
	Worst-case	6.292ns

硬件调试情况

调试顺利，未遇到困难。

实验七 序列检测器设计

实验目的：

掌握有限状态机的实现原理和方法；掌握序列检测的方法。

实验原理：

有限状态机（Finite State Machine, FSM）是逻辑电路设计中经常要遇到的，在数字电路中，通过建立有限状态机来进行时序数字逻辑的设计。在复杂数字系统设计中，有限状态机主要通过硬件描述语言实现，硬件描述语言能够清晰的描述状态转移过程和输入输出变量关系，使得时序逻辑设计大大简化，进而极大降低系统设计复杂度，提高系统模块化程度。

有限状态机从本质上讲是由寄存器和组合逻辑构成的时序电路，各个状态之间的转移总是在时钟的触发下进行的。可以通过建立原始状态表和状态化简来设计电路。

关键代码：

```
module FSM(A,r,clk,S,result);
    input wire A,r,clk;
    output reg [2:0]S;
    output reg result;
    always@(negedge r or posedge clk)
    begin
        if(r==0) S<=3'b000;
        else
        begin
            case(S)
            3'b000:     S<=(A==1)?3'b001:3'b000;    //start A     start
            3'b001:     S<=(A==1)?3'b001:3'b010;    //A           A       B
            3'b010:     S<=(A==1)?3'b011:3'b000;    //B           C       start
```

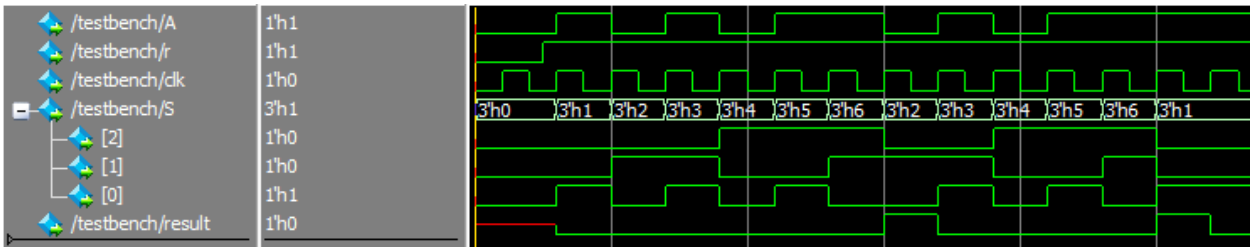
```
                3'b011: S<=(A==1)?3'b001:3'b100;    //C          A       D
                3'b100: S<=(A==1)?3'b101:3'b000;    //D          E       start
                3'b101: S<=(A==1)?3'b110:3'b010;    //E          F       B
                3'b110:     S<=(A==1)?3'b001:3'b010;    //F          A       B
                endcase
                result<=(S==3'b110)?1:0;
            end
        end
endmodule
```

文件清单：

FSM.v　　有限状态机

testbench.v　testbench

仿真结果：



序列检测器工作正常。

综合情况

累加器　　Total logic elements　5

　　Worst-case　　6.589 ns

硬件调试情况

调试顺利，未遇到困难。