**实验七：总线实验**

实验时间： 2017年12月11日 第十四周 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

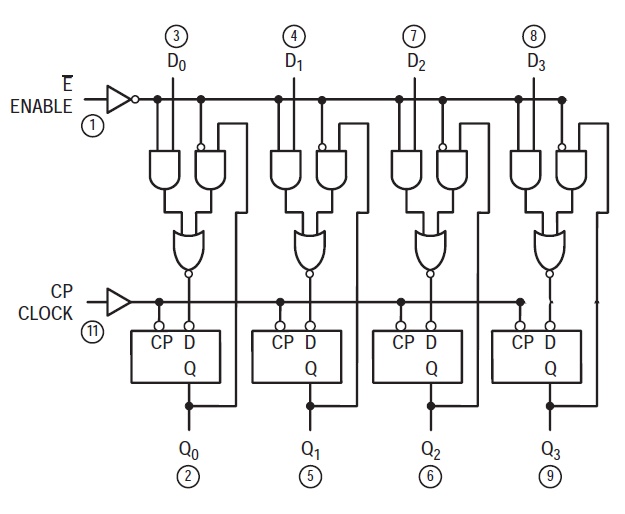
1. 实验目的

* 了解并掌握寄存器的原理和设计，加深对时序电路的理解
* 掌握RAM的储存功能的实现机制
* 能熟练设计、运用多于1位宽度的多路选择器
* 能理解实验的数据通路，加深模块化的理解，并能熟练进行电路微操作

1. 实验原理
   1. 74LS377寄存器：带使能的4位宽D触发器

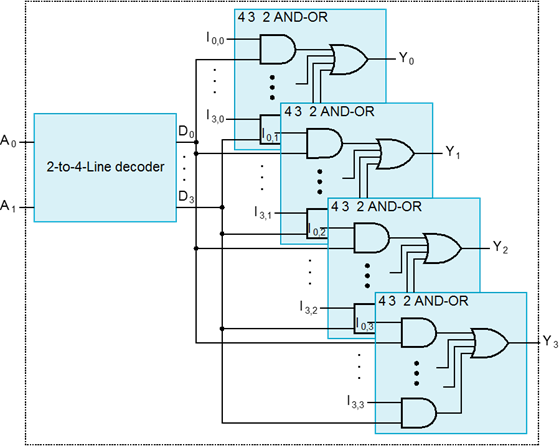
* 输入：输入信号D[3:0]，开关输入使能信号EN，时钟信号CLK；输出：输出信号Q[3:0]，并在对应的LED上显示
* 状态表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EN | D[i] | Q[i](n) | Q[i](n+1) | 操作 |
| 0 | 0 | 0 | 0 | 无变化 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 置数 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

* 具体实现：当EN=0时，Q不变；当EN=1时，Q<=D置数
* 逻辑电路图
  1. RAM（4位）
* 输入：时钟信号clk，控制写入信号wen，地址addr[3:0]，输入数据din[3:0]；输出：输出数据qout[3:0]
* 存储机制：模块中定义了reg[3:0]ram[0:15]，也就是16个reg型的ram[3:0]变量，这些变量从0~15编号，而这些编号也就是地址，每个地址对应一个ram[3:0]变量，也就是对应一个存储在RAM中的值。通过SW[4:1]输入二进制地址addr[3:0]，值的范围为0000~1111，也就是0~15，恰好与编号一一对应，故ram[addr]就是存储在addr的4位二进制数
* 写入：当wen控制信号为1，且时钟上升沿到达时，会把din中的4位二进制数值写入ram[addr]，为同步过程，可以通过always@()语句实现
* 读取：异步向qout输出ram[addr]的值，并在对应LED上显示
  1. 4位宽4-1 MUX
* 输入：选择信号s[1:0]，被选择的4个4位宽向量i0[3:0]，i1[3:0]，i2[3:0]，i3[3:0]；输出选择出的长度为4的向量y[3:0]
* s[1:0]=00，01，10，11时分别对应选择数据通路中该器件的a，b，c，d输入
* 可以通过case()语句，对s[1:0]的值进行判断，把对应输入向量赋值给输出向量
* 由于MUX是异步的，因此所有输入值都在always@()语句的敏感表中，可以用\*代替
* 选择表格

|  |  |  |
| --- | --- | --- |
| 控制 | | 输出 |
| s[1] | s[0] | y |
| 0 | 0 | i0 / a |
| 0 | 1 | i1 / b |
| 1 | 0 | i2 / c |
| 1 | 1 | i3 / d |

* 逻辑电路图

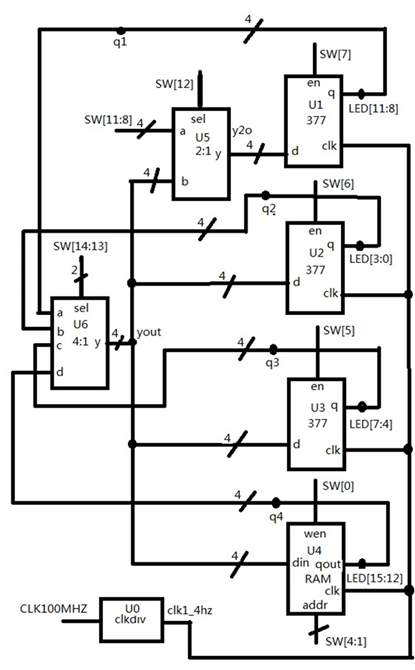


* 1. 4位宽2-1 MUX
* 输入：选择信号s，被选择的2个4位宽向量i0[3:0]，i1[3:0]；输出被选择的长度为4的向量y[3:0]
* s=0，1时分别对应选择数据通路中的该器件的a，b输入
* 可以通过case()语句，对s的值进行判断，把对应输入向量赋值给输出向量
* 由于MUX是异步的，因此所有输入值都在always@()语句的敏感表中，可以用\*代替
* 选择表格

|  |  |
| --- | --- |
| 控制s | 输出y |
| 0 | i0 / a |
| 1 | i1 / b |

* 1. 时钟分频模块
* 取的是q[24]的值，输出时钟信号约为4Hz

1. 实验分析

* U0为时钟分频模块；U1，U2，U3为3个4位宽寄存器；U4为RAM；U5为4位宽的2-1MUX；U6为4位宽的4-1MUX
* U0把时钟频率从100MHz降低到约4Hz，并控制U1~U4
* 2-1多路复用器U5的SW[11:8]是唯一的数据输入处，通过调SW[11:8]输入4位二进制数，通过调节SW[12]控制MUX选择a或b输入，并传送给U1寄存器
* 通过SW[7]使能控制U1，当SW[7]=1时，U1的值被输入值更新，并输出到LED[11:8]以及4-1MUX的输入端a
* 通过调节U6的SW[14:13]可控制MUX选择a，b，c，d输入中的一个，并传送给2-1MUX的b输入、U2、U3、U4的输入端
* 当U2的SW[6]=1时U2被触发，U2储存的值被4-1MUX传来的值更新，并输出到LED[3:0]以及4-1MUX的输入端b
* 当U3的SW[5]=1时U3被触发，U3储存的值被4-1MUX传来的值更新，并输出到LED[7:4]以及4-1MUX的输入端c
* 当U4的SW[0]=1时U4的写入被触发，U4中储存在地址SW[4:1]处的值被4-1MUX输出值更新，并输出到LED[15:12]以及4-1MUX的d输入端；其他时候，不断输出地址SW[4:1]处储存的值到4-1MUX的d输入端，并在LED[15:12]显示
* 逻辑电路图

1. 实验内容
   1. 设计思想

* 独立实现各个模块
* 把各个模块的输入、输出连接起来
  1. Verilog代码

module bus(input clk, input [14:0] SW, output [15:0] LED);

wire myclk;

wire [3:0] d1;

wire [3:0] y;

wire [15:0] light;

clkdiv U0(clk,myclk);

register U1(myclk,SW[7],d1[3:0],light[11:8]);

assign LED[11:8]=light[11:8];

register U2(myclk,SW[6],y[3:0],light[3:0]);

assign LED[3:0]=light[3:0];

register U3(myclk,SW[5],y[3:0],light[7:4]);

assign LED[7:4]=light[7:4];

ram2 U4(myclk,SW[0],SW[4:1],y[3:0],light[15:12]);

assign LED[15:12]=light[15:12];

MUX\_2\_1 U5(SW[12],SW[11:8],y[3:0],d1[3:0]);

MUX\_4\_1 U6(SW[14:13],light[11:8],light[3:0],light[7:4],light[15:12],y[3:0]);

endmodule

module register(input myclk, input en, input [3:0] d, output reg [3:0] q);

always@(posedge myclk,posedge en)

begin

if(en)

q<=d;

end

endmodule

module ram2(input clk, input wen, input [3:0] addr, input [3:0] din, output [3:0] qout);

reg[3:0]ram[0:15];

always@(posedge clk)

if(wen)

ram[addr]<=din;

assign qout=ram[addr];

endmodule

module clkdiv(input mclk, output clk1\_4hz);

reg [27:0]q;

always@(posedge mclk)

q<=q+1;

assign clk1\_4hz=q[24];

endmodule

module MUX\_4\_1(input [1:0] s, input [3:0] i0, input [3:0] i1, input [3:0] i2, input [3:0] i3, output reg [3:0] y);

always@(\*)

begin

case(s)

2'b00:y<=i0;

2'b01:y<=i1;

2'b10:y<=i2;

2'b11:y<=i3;

endcase

end

endmodule

module MUX\_2\_1(input s, input [3:0] i0, input [3:0] i1, output reg [3:0] y);

always@(\*)

begin

case(s)

1'b0:y<=i0;

1'b1:y<=i1;

endcase

end

endmodule

* 1. 改进
* 不必使用light[15:0]来作为LED[15:0]的中间量，在模块中，LED[15:0]信号可以直接作为输入、输出信号，代码会更简洁
  1. 实验操作（输入1111，0000两个值，分别存在U2，U3寄存器，并利用RAM为中介进行值的交换）
* SW[14:0]=15’b000000000000000置零
* SW[11:8]=4‘b1111，而2-1MUX的选择控制SW[12]=0，选择输出的是a，也就是开关SW[11:8]输入的值1111，输出到U1的输入口，使SW[7]=1触发U1，LED[11:8]全亮，表示1111存入U1，SW[7]=0关闭U1
* 此时SW[14:13]==2’b00，选择输出的是a，也就是U1的输出值1111，并输出到U2、U3、U4的输入口，使SW[6]=1触发U2，LED[3:0]全亮，表示1111存入U2，关闭SW[6]
* SW[11:8]=4’b0000，而2-1MUX的选择控制SW[12]=0，选择输出的是a，也就是开关SW[11:8]输入的值0000，输出到U1的输入口，SW[7]=1触发U1，LED[11:8]全暗，表示0000存入U1，SW[7]=0关闭U1
* 此时SW[14:13]==2’b00，选择输出的是a，也就是U1的输出值1111，并输出到U2、U3、U4的输入口，使SW[5]=1触发U3，LED[7:4]仍然是暗的，表示0000存入了U3，关闭SW[5]
* SW[14:13]=2’b01，使4-1MUX选择来自U2输出的b输入1111，输出到y，也就是输出到U2、U3、U4的输入口
* SW[0]=1触发U4，而SW[4:1]为0000，此时把输入值1111写入地址0000处，并输出到LED[15:12]，全亮，并输出到4-1MUX的d输入端
* SW[14:13]=2’b10，选择来自U3输出的c输入0000，并输出到y，也就是输出到U2、U3、U4的输入口，使SW[6]=1，触发U2，把0000写入U2，LED[3:0]全灭，表示0000存入U2，SW[6]=0关闭U2
* SW[14:13]=2’b11，选择来自U4在地址0000储存的值的输出1111，并输出到y，也就是输出到U2、U3、U4的输入口，使SW[5]=1触发U3，把1111写入U3，LED[7:4]全亮，表示1111存入U3，SW[5]=0关闭U3
* 操作完成

1. 实验结论

* 实现了把1111写入U2，0000写入U3，并利用RAM为中介，交换U2、U3中的值
* 寄存器具有储存、转移的功能
* RAM具有储存多个数据的功能

1. 实验感想

* 最初疏忽没有把4-1MUX的y输出和U2、U3的d输入连接起来，导致SW改变时U2、U3完全没有反应，后来通过仔细检查才发现并更正
* 虽然4位宽的D触发器可以使用4个1位宽D触发器以及若干与门、或门，通过分层设计、模块化的思想来实现，但是这样做反而会复杂化，更不直观，且使得debug更困难；4-1MUX、2-1MUX也是如此。因此需要利用Verilog语言的便捷之处，以简化问题以及代码设计，以避免粗暴的硬件翻译，也更直观，便于debug
* 要能熟练进行微操作，则必须要十分清楚各个器件的作用，了解各个输入、输出，以及整个数据通路的关系，并明晰操作流程和效果
* 在较大电路的设计中，模块化具有很强的优势，能使得整个电路虽然庞大但是不乱