Jae Lee
ECEN 220
Lab #7
10/22/2013

1-bit Register Verilog code

```
module reg_one(dout, din, write, clr, clk);
                output dout;
                input din, write, clr, clk;
                wire temp1;

                feff f1(dout, clk, temp1, clr);
                mux21 m1(temp1, write, dout, din);
endmodule
```

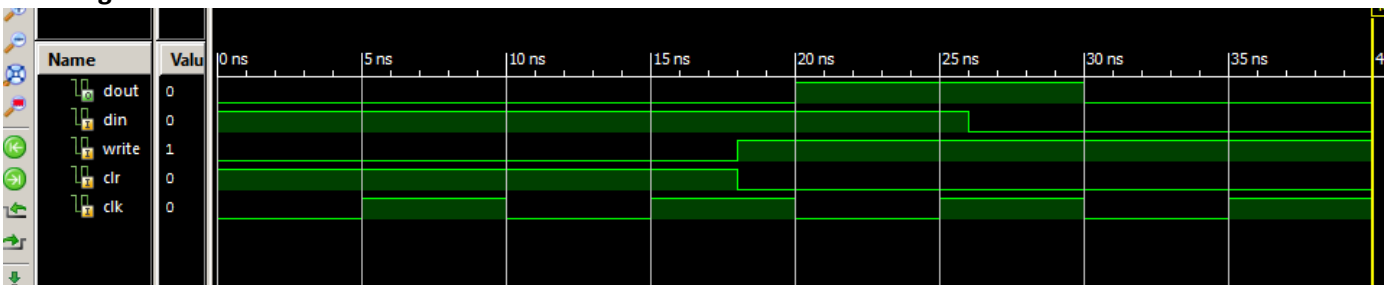
1-bit Register TCL file

```
wave add / -radix hex

isim force add din 1 -time 0 -value 0 -time 26ns
isim force add write 0 -time 0 -value 1 -time 18ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 1 -time 0 -value 0 -time 18ns


run 40ns
```

1-bit Register simulation waveform



4-bit Register Verilog code

```
module reg_four(dout, din, write, clr, clk);
                output[3:0] dout;
                input[3:0] din;
                input write, clr, clk;
                reg_one r0(dout[0], din[0], write, clr, clk);
                reg_one r1(dout[1], din[1], write, clr, clk);
                reg_one r2(dout[2], din[2], write, clr, clk);
                reg_one r3(dout[3], din[3], write, clr, clk);
endmodule
```

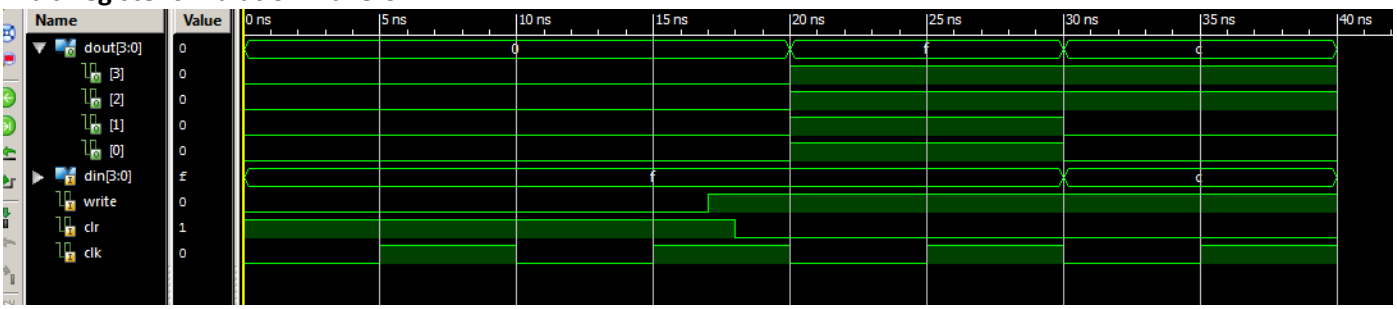
4-bit Register TCL file

```
wave add / -radix hex
```

```
isim force add write 0 -time 0 -value 1 -time 17ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 1 -time 0 -value 0 -time 18ns
isim force add din 1111 -time 0 -value 1100 -time 30ns

run 40ns
```

4-bit Register simulation waveform



- 2 to 4 Decoder Verilog code (1)
- 2 to 4 Decoder TCL file (1)
- 2 to 4 Decoder simulation waveform (2)

2 to 4 Decoder Verilog code

```
module decorder24(q,a);
        input[1:0] a;
        output[3:0] q;

        assign q = (4'b0001 << a);
endmodule
```

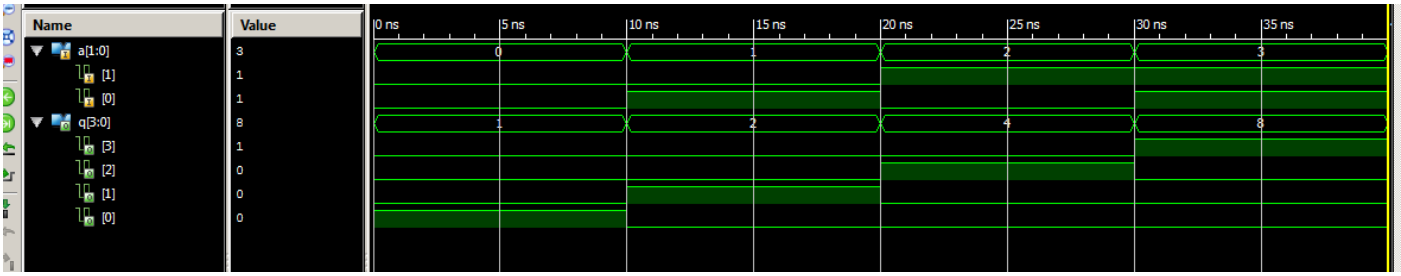
2 to 4 Decoder TCL file

```
wave add / -radix hex

isim force add a 00 -time 0 -value 01 -time 10ns -value 10 -time 20ns
-value 11 -time 30ns

run 40ns
```

2 to 4 Decoder simulation waveform



- 4-bit x 4-word Registers Verilog code (3)
- 4-bit x 4-word Registers TCL file (1)
- 4-bit x 4-word Registers simulation waveform (3)

4-bit x 4-word Registers Verilog code

```
module word_register(dout, addr, regWE, din, clk, clr);

        output[15:0] dout;
        input[1:0] addr;
        input regWE, clk, clr;
        input[3:0] din;
        wire[3:0] sel;
        wire temp0, temp1, temp2, temp3;

        and(temp0, sel[0], regWE);
        and(temp1, sel[1], regWE);
        and(temp2, sel[2], regWE);
        and(temp3, sel[3], regWE);
        decorder24 d1(sel, addr);
        reg_four r0(dout[3:0], din, temp0, clr, clk);
        reg_four r1(dout[7:4], din, temp1, clr, clk);
        reg_four r2(dout[11:8], din, temp2, clr, clk);
        reg_four r3(dout[15:12], din, temp3, clr, clk);
endmodule
```

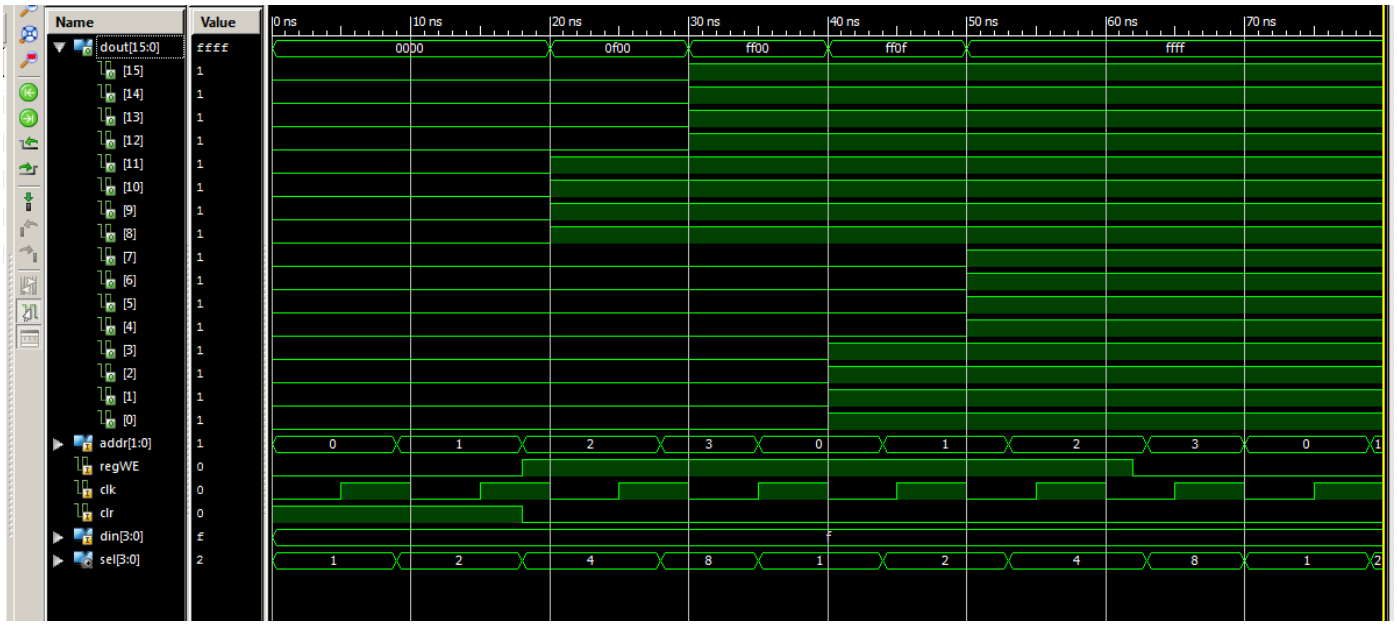
4-bit x 4-word Registers TCL file

```
wave add / -radix hex


isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 1 -time 0 -value 0 -time 18ns
isim force add din 1111 -time 0
isim force add addr 00 -time 0 -value 01 -time 9ns -value 10 -time
18ns -value 11 -time 28ns -repeat 35ns
isim force add regWE 0 -time 0 -value 1 -time 18ns -value 0 -time 62ns


run 80ns
```

4-bit x 4-word Registers simulation waveform



4:1 4-bit MUX Verilog code

```
module mux164(result, sel, din3, din2, din1, din0);
                input[1:0] sel;
                input[3:0] din3, din2, din1, din0;
                output[3:0] result;

                mux41 m0(result[0], sel[1:0],{din3[0],din2[0],din1[0],din0[0]});
                mux41 m1(result[1], sel[1:0],{din3[1],din2[1],din1[1],din0[1]});
                mux41 m2(result[2], sel[1:0],{din3[2],din2[2],din1[2],din0[2]});
                mux41 m3(result[3], sel[1:0],{din3[3],din2[3],din1[3],din0[3]});
endmodule
```

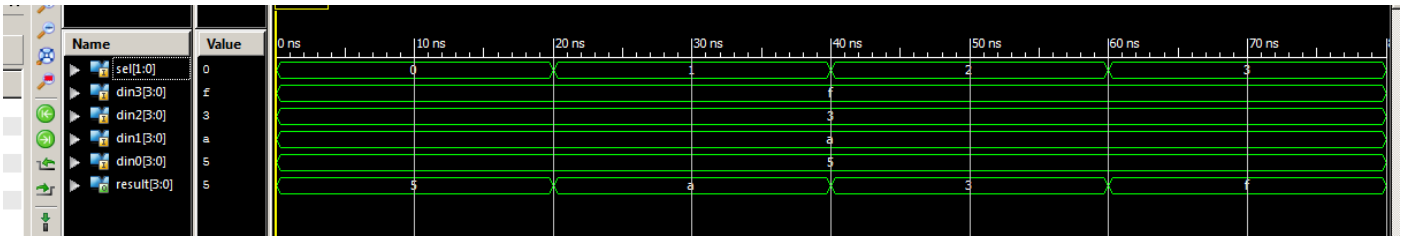
4:1 4-bit MUX TCL file

```
wave add / -radix hex


isim force add din3 1111 -time 0
isim force add din2 0011 -time 0
isim force add din1 1010 -time 0
isim force add din0 0101 -time 0
isim force add sel 00 -time 0 -value 01 -time 20ns -value 10 -time
40ns -value 11 -time 60ns

run 80ns
```

4:1 4-bit MUX simulation waveform



Complete Register File Verilog code

```
module complete(Reg_out, din, regWE, clr, clk, waddr, raddr);
                output[3:0] Reg_out;
                input[3:0] din;
                input clr, clk, regWE;
                input[1:0] waddr, raddr;
                wire[15:0] dout;

                word_register wr1(dout, waddr, regWE, din, clk, clr);
                mux164 m1(Reg_out, raddr, dout[15:12], dout[11:8], dout[7:4], dout[3:0]);
endmodule
```

TestBench Verilog code

```
module testbench(Reg_out, Din_led3, Din_led2, Din_led1, Din_led0, din, regWE, clr, clk, waddr, raddr);
                output[3:0] Reg_out;
                output Din_led3, Din_led2, Din_led1, Din_led0;
                input[3:0] din;
                input clr, clk, regWE;
                input[1:0] waddr, raddr;
                wire[15:0] dout;

                complete c1(Reg_out, din, regWE, clr, clk, waddr, raddr);
                buf(Din_led3,din[3]);
                buf(Din_led2,din[2]);
                buf(Din_led1,din[1]);
                buf(Din_led0,din[0]);

endmodule
```

TestBench UCF file

```
## clock pin for Nexys 2 Board
NET clk   LOC = "B8"; # Bank = 0, Pin name = IP_L13P_0/GCLK8, Type = GCLK, Sch name = GCLK0
##NET "clk1" LOC = "U9"; # Bank = 2, Pin name = IO_L13P_2/D4/GCLK14, Type = DUAL/GCLK, Sch name
= GCLK1

## Leds
NET      Reg_out<0>  LOC = "J14"; # Bank = 1, Pin name = IO_L14N_1/A3/RHCLK7, Type = RHCLK/DUAL,
Sch name = JD10/LD0
NET Reg_out<1>  LOC = "J15"; # Bank = 1, Pin name = IO_L14P_1/A4/RHCLK6, Type = RHCLK/DUAL, Sch
name = JD9/LD1
```

```
NET Reg_out<2>  LOC = "K15"; # Bank = 1, Pin name = IO_L12P_1/A8/RHCLK2, Type = RHCLK/DUAL, Sch name = JD8/LD2
NET Reg_out<3>  LOC = "K14"; # Bank = 1, Pin name = IO_L12N_1/A7/RHCLK3/TRDY1, Type = RHCLK/DUAL, Sch name = JD7/LD3
NET Din_led0  LOC = "E17"; # Bank = 1, Pin name = IO, Type = I/O, Sch name = LD4
NET Din_led1  LOC = "P15"; # Bank = 1, Pin name = IO, Type = I/O, Sch name = LD5
NET Din_led2  LOC = "F4";  # Bank = 3, Pin name = IO, Type = I/O, Sch name = LD6
NET Din_led3  LOC = "R4";  # Bank = 3, Pin name = IO/VREF_3, Type = VREF, Sch name = LD7

## Switches
NET raddr<0> LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW0
NET raddr<1> LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = SW1
NET waddr<0> LOC = "K18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW2
NET waddr<1> LOC = "K17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW3
NET din<0> LOC = "L14"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW4
NET din<1> LOC = "L13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW5
NET din<2> LOC = "N17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW6
NET din<3> LOC = "R17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW7

## Buttons
NET clr LOC = "B18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN0
#NET "btn<1>" LOC = "D18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = BTN1
#NET C0<0> LOC = "E18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN2
NET regWE LOC = "H13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN3
```

Anomalies

This took pretty long to finish. I got confused because of many inheriting Verilog files. Input and output digits were pretty confusing to make straight forward. Writing Testbench is something that I still need to improve.