Lab #7 - Register File
Isai Mercado Oliveros
Oct 29, 2014

1-bit Register Verilog code (3)

```
20  //////////////////////////////////////////////////////////////////////////////////
21  module bit1register(q,clk,clr,din,write);
22
23     input clk,clr,din,write;
24     output q;
25     wire temp;
26
27     assign temp = (write)? din:q;
28     flipFlop flipflop(q,clk,temp,clr);//module flipFlop(q, clk, d, clr);
29
30  endmodule
31
```

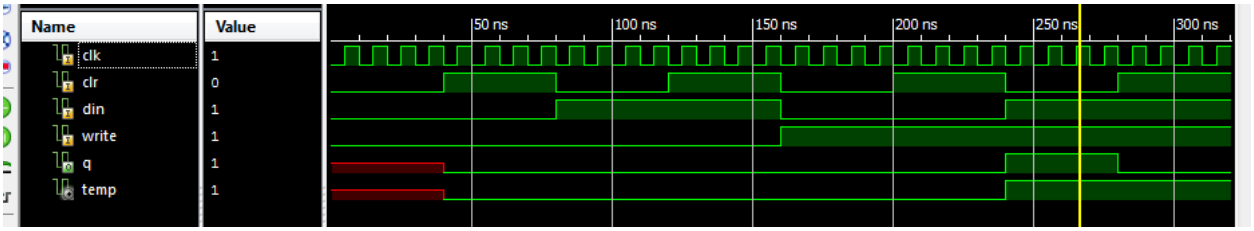
1-bit Register TCL file (1)

```
File  Edit  Format  View  Help
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write


#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 0 -time 0 -value 1 -time 40ns -repeat 80ns
isim force add din 0 -time 0 -value 1 -time 80ns -repeat 160ns
isim force add write 0 -time 0 -value 1 -time 160ns -repeat 320ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

1-bit Register simulation waveform (3)



4-bit Register Verilog code (3)

```
21  module bit4register(q, clk, clr,din,write);
22
23      input[3:0] din;
24      input clk,clr,write;
25      output[3:0] q;
26
27      bit1register reg1(q[0],clk,clr,din[0],write);
28      bit1register reg2(q[1],clk,clr,din[1],write);
29      bit1register reg3(q[2],clk,clr,din[2],write);
30      bit1register reg4(q[3],clk,clr,din[3],write);
31
32  endmodule
33
```

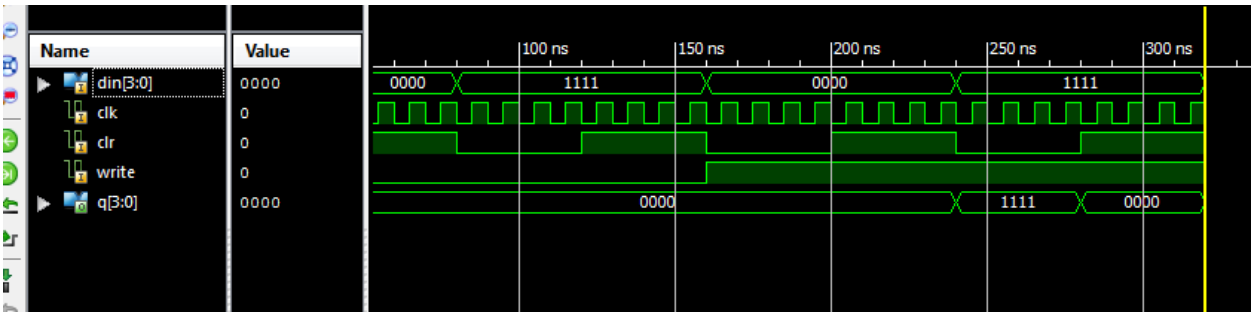
4-bit Register TCL file (1)

```
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write


#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0000 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 0 -time 0 -value 1 -time 40ns -repeat 80ns
isim force add din 0000 -time 0 -value 1111 -time 80ns -repeat 160ns
isim force add write 0 -time 0 -value 1 -time 160ns -repeat 320ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

4-bit Register simulation waveform (3)



2 to 4 Decoder Verilog code (1)

```
20  ////////////////////////////////
21  module decoder2to4(control,f);
22     input[1:0] control;
23     output[3:0] f;
24
25     assign f = 1 << control;
26
27  endmodule
28
```

2 to 4 Decoder TCL file (1)

File Edit Format View Help

```
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0000 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add control 00 -time 0 -value 01 -time 10ns -value 10 -time 20ns -value 11 -time 30ns -repeat 40ns
#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

2 to 4 Decoder simulation waveform (2)

4-bit x 4-word Registers Verilog code (3)

```
20  //////////////////////////////////////////////////////////////
21  module registerFile(sel,clk,clr,w,write,q0,q1,q2,q3);
22
23     input[1:0] sel;
24     input[3:0] w;
25     input clk,clr,write;
26     output[3:0] q0,q1,q2,q3;
27     wire[3:0] we,f;
28     //decoder2to4(control,f);
29     //bit4register(q, clk, clr,din,write);
30     decoder2to4 dec (sel,f);
31
32     assign we[0] = (f[0] == 1 && write)? 1 : 0;
33     assign we[1] = (f[1] == 1 && write)? 1 : 0;
34     assign we[2] = (f[2] == 1 && write)? 1 : 0;
35     assign we[3] = (f[3] == 1 && write)? 1 : 0;
36
37     bit4register reg1 (q0, clk, clr,w,we[0]);
38     bit4register reg2 (q1, clk, clr,w,we[1]);
39     bit4register reg3 (q2, clk, clr,w,we[2]);
40     bit4register reg4 (q3, clk, clr,w,we[3]);
41
42  endmodule
43
```

4-bit x 4-word Registers TCL file (1)

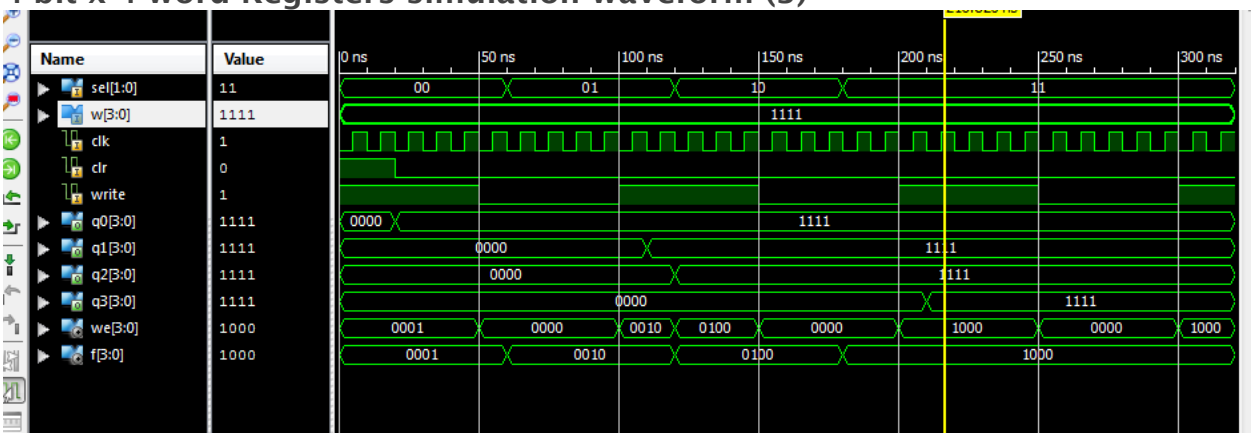
File Edit Format View Help

```
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0000 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add sel 00 -time 0 -value 01 -time 60Ons -value 10 -time 120ns -value 11 -time 180ns -repeat 320ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add w 1111 -time 0 -value 0000 -time 150ns -repeat 480ns
isim force add write 1 -time 0 -value 0 -time 50ns -repeat 100ns

#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

4-bit x 4-word Registers simulation waveform (3)



4:1 4-bit MUX Verilog code (3)

```
////////////////////////////////////////////////////////////////////////////////
module mux4to1_4bits(input0, input1, input2, input3, output0, control);
    input[3:0] input0,input1,input2,input3;
    output[3:0] output0;
    input[1:0] control;

    assign output0 = (control == 0)? input0 :
                     (control == 1)? input1 :
                     (control == 2)? input2 :
                     (control == 3)? input3 : 0;

endmodule
```

4:1 4-bit MUX TCL file (1)

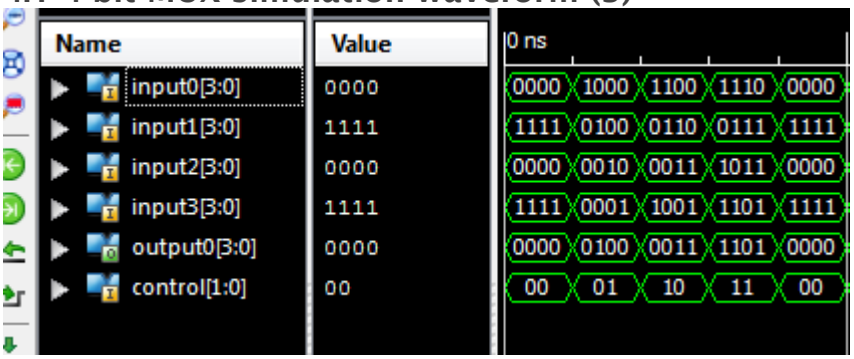
mux.tlc.txt - Notepad

File Edit Format View Help

```
#add all signals to the waveform viewer
wave add / -radix hex
#q, clk, d, clr,din,write

#define how the data input signals will behave when you run the simulation
#the command "isim force add (signal) 0 -time 0 -value 1 -time 20ns -repeat 40ns" means that (signal)
#will have a value of 0000 from time 0, then change to 1 at time 20ns, and then repeat that cycle every 40ns
isim force add control 00 -time 0 -value 01 -time 10ns -value 10 -time 20ns -value 11 -time 30ns -repeat 40ns
isim force add input0 0000 -time 0 -value 1000 -time 10ns -value 1100 -time 20ns -value 1110 -time 30ns -repeat 40ns
isim force add input1 1111 -time 0 -value 0100 -time 10ns -value 0110 -time 20ns -value 0111 -time 30ns -repeat 40ns
isim force add input2 0000 -time 0 -value 0010 -time 10ns -value 0011 -time 20ns -value 1011 -time 30ns -repeat 40ns
isim force add input3 1111 -time 0 -value 0001 -time 10ns -value 1001 -time 20ns -value 1101 -time 30ns -repeat 40ns
#Nothing will change in the waveform viewer until you run the simulation for some period of time.
run 320ns
```

4:1 4-bit MUX simulation waveform (3)



Complete Register File Verilog code (3)

```
19  //
20  //////////////////////////////////////////////////////////////////////////////////
21  module fourWordFourBitFile(in, out, writeAddress, readAddress, clk, clr, write);
22     input[3:0] in;
23     input[3:0] out;
24     input[1:0] writeAddress;
25     input[1:0] readAddress;
26     input clk, clr, write;
27     wire[3:0] input0, input1, input2, input3;
28
29        registerFile file(writeAddress, clk,clr,in,write,input0,input1,input2,input3);
30        mux4to1_4bits(input0, input1, input2, input3, out, readAddress); //mux4to1_4bi
31
32  endmodule
33
```

TestBench Verilog code (3)

```
20  //////////////////////////////////////////////////////////////////////////////////
21  //fourWordFourBitFile(in, out, writeAddress, readAddress, clk, clr, write);
22  module testBench(in, out, writeAddress, readAddress, clk, clr, write,led);
23     input[3:0] in;
24     output[3:0] out, led;
25     input[1:0] writeAddress;
26     input[1:0] readAddress;
27     input clk, clr, write;
28
29     assign led[0] = in[0];
30     assign led[1] = in[1];
31     assign led[2] = in[2];
32     assign led[3] = in[3];
33
34     fourWordFourBitFile file(in, out, writeAddress, readAddress, clk, clr, write);
35
36  endmodule
37
```

TestBench UCF file (2)

```
## clock pin for Nexys 2 Board
NET clk   LOC = "B8"; # Bank = 0, Pin name = IP_L13P_0/GCLK8, Type = GCLK, Sch name = GCLK0

## Leds
NET out[0]  LOC = "J14"; # Bank = 1, Pin name = IO_L14N_1/A3/RHCLK7, Type = RHCLK/DUAL, Sch name = JD10/LD0
NET out[1]  LOC = "J15"; # Bank = 1, Pin name = IO_L14P_1/A4/RHCLK6, Type = RHCLK/DUAL, Sch name = JD9/LD1
NET out[2]  LOC = "K15"; # Bank = 1, Pin name = IO_L12P_1/A8/RHCLK2, Type = RHCLK/DUAL, Sch name = JD8/LD2
NET out[3]  LOC = "K14"; # Bank = 1, Pin name = IO_L12N_1/A7/RHCLK3/TRDY1, Type = RHCLK/DUAL, Sch name = JD7/LD3

NET led[0]  LOC = "E17"; # Bank = 1, Pin name = IO, Type = I/O, Sch name = LD4
NET led[1]  LOC = "P15"; # Bank = 1, Pin name = IO, Type = I/O, Sch name = LD5
NET led[2]  LOC = "F4";  # Bank = 3, Pin name = IO, Type = I/O, Sch name = LD6
NET led[3]  LOC = "R4";  # Bank = 3, Pin name = IO/VREF_3, Type = VREF, Sch name = LD7



## Switches
NET readAddress[0] LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW0
NET readAddress[1] LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = SW1

NET writeAddress[0] LOC = "K18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW2
NET writeAddress[1] LOC = "K17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW3

NET in[0] LOC = "L14"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW4
NET in[1] LOC = "L13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW5
NET in[2] LOC = "N17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW6
NET in[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 write LOC = "H13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN3
```

Anomalies

All when well. This lab is ok if you understand what you are doing and also it is good to see how the board has memory. It just makes me think of what is happening every time I am typing. Also the TA's where supper helpful.