Taylor Cowley

EE 220

May 24 2016

Lab 07: Register File

**1 bit Register Verilog Code**

module register(dout, din, clk, clr, write);

input din;

input clk;

input clr;

input write;

output dout;

wire intermediate;

FF\_DC flip\_flop(dout, clk, clr, intermediate);

mux\_2 mux(intermediate, write, dout, din);

endmodule

**1-bit Register tcl file**

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns

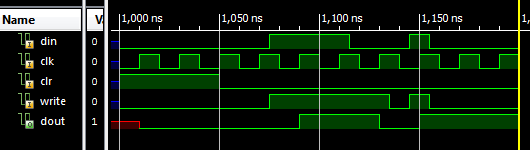
isim force add clr 1 -time 0 -value 0 -time 50ns

isim force add write 0 -time 0 -value 1 -time 75ns -value 0 -time 135ns -value 1 -time 145ns -value 0 -time 155ns

isim force add din 0 -time 0 -value 1 -time 75ns -value 0 -time 115ns -value 1 -time 145ns -value 0 -time 155ns

run 200ns

**1-bit Register simulation waveform**



**4-bit Register Verilog Code**

module register\_4(dout, din, clk, clr, write);

input[3:0] din;

input clk;

input clr;

input write;

output[3:0] dout;

register r0(dout[0],din[0],clk,clr,write);

register r1(dout[1],din[1],clk,clr,write);

register r2(dout[2],din[2],clk,clr,write);

register r3(dout[3],din[3],clk,clr,write);

endmodule

**4-bit Register tcl file**

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns

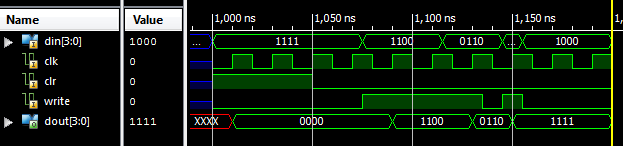
isim force add clr 1 -time 0 -value 0 -time 50ns

isim force add write 0 -time 0 -value 1 -time 75ns -value 0 -time 135ns -value 1 -time 145ns -value 0 -time 155ns

isim force add din 1111 -time 0 -value 1100 -time 75ns -value 0110 -time 115ns -value 1111 -time 145ns -value 1000 -time 155ns

run 200ns

**4-bit Register simulation waveform**



**2 to 4 Decoder Verilog code**

module decoder\_2\_4(in, out);

input[1:0] in;

output[3:0] out;

assign out[3] = in[1] & in[0];

assign out[2] = in[1] & ~in[0];

assign out[1] = ~in[1] & in[0];

assign out[0] = ~in[1] & ~in[0];

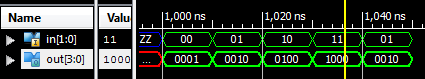
endmodule

**2 to 4 decoder tcl file**

isim force add in 00 -time 0ns -value 01 -time 10ns -value 10 -time 20ns -value 11 -time 30ns -value 01 -time 40ns

run 50ns

**2 to 4 decoder simulation waveform**



**4x4 Registers Verilog**

module register\_4\_4(dout, din, clk, clr, write, address);

input[3:0] din;

input clk;

input clr;

input write;

input [1:0] address;

output[15:0] dout;

wire [3:0] wr\_en, write\_enable;

register\_4 r0(dout[3:0], din, clk, clr, write\_enable[0]);

register\_4 r1(dout[7:4], din, clk, clr, write\_enable[1]);

register\_4 r2(dout[11:8], din, clk, clr, write\_enable[2]);

register\_4 r3(dout[15:12], din, clk, clr, write\_enable[3]);

decoder\_2\_4 decode(address, wr\_en);

// We concat the write signal 4 times to be the same length as wr\_en

assign write\_enable = wr\_en & {write,write,write,write};

endmodule

**4x4 Registers tcl file**

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns

isim force add clr 1 -time 0 -value 0 -time 50ns

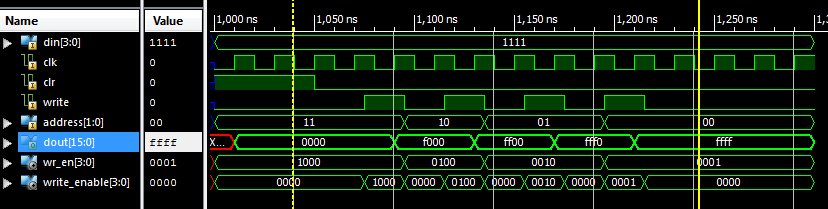
isim force add write 0 -time 0 -value 1 -time 75ns -value 0 -time 95ns -value 1 -time 115ns -value 0 -time 135ns -value 1 -time 155ns -value 0 -time 175ns -value 1 -time 195ns -value 0 -time 215ns

isim force add din 1111

isim force add address 11 -time 0 -value 10 -time 95ns -value 01 -time 135ns -value 00 -time 195ns

run 300ns

**4x4 registers simulation waveform**



**16-4 mux Verilog Code**

module mux\_4(out, sel, in);

input[3:0] in;

input[1:0] sel;

output out;

wire a\_or\_b, c\_or\_d;

mux\_2 M0(a\_or\_b, sel[0], in[0], in[1]);

mux\_2 M1(c\_or\_d, sel[0], in[2], in[3]);

mux\_2 M2(out, sel[1], a\_or\_b, c\_or\_d);

endmodule

module mux\_16\_4(out, sel, in3, in2, in1, in0);

input[3:0] in3, in2, in1, in0;

input[1:0] sel;

output[3:0] out;

mux\_4 bit3(out[3], sel, {in3[3], in2[3], in1[3], in0[3]});

mux\_4 bit2(out[2], sel, {in3[2], in2[2], in1[2], in0[2]});

mux\_4 bit1(out[1], sel, {in3[1], in2[1], in1[1], in0[1]});

mux\_4 bit0(out[0], sel, {in3[0], in2[0], in1[0], in0[0]});

endmodule

**16-4 mux tcl file**

isim force add in3 1111

isim force add in2 1100

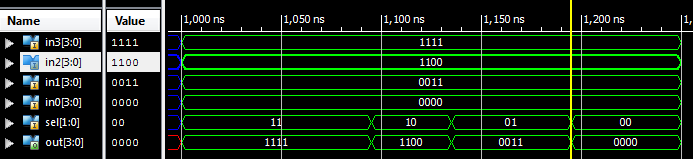
isim force add in1 0011

isim force add in0 0000

isim force add sel 11 -time 0 -value 10 -time 95ns -value 01 -time 135ns -value 00 -time 195ns

run 250ns

**16-4 mux simulation waveform**



**Complete Register File Verilog Code**

module register\_file\_4x4(out, in, write, clr, clk, wr\_address, r\_address);

input write, clr, clk;

input[3:0] in;

input[1:0] r\_address, wr\_address;

output[3:0] out;

wire[15:0] regout;

register\_4\_4 the\_register(regout, in, clk, clr, write, wr\_address);

mux\_16\_4 a\_mux(out, r\_address, regout[15:12], regout[11:8], regout[7:4], regout[3:0]);

endmodule

**Testbench Verilog code**

module register\_file\_testbench(Led, sw, btn, clk);

output[7:0] Led;

input [7:0] sw;

input [3:0] btn;

input clk;

//module register\_file\_4x4(out, in, write, clr, clk, wr\_address, r\_address);

register\_file\_4x4 booya(Led[3:0],sw[7:4],btn[3],btn[0],clk,sw[3:2],sw[1:0]);

buf(Led[7], sw[7]); // Now we have to buffer the output leds to the inputs

buf(Led[6], sw[6]);

buf(Led[5], sw[5]);

buf(Led[4], sw[4]);

endmodule

**Testbench UCF File**

## clock pin for Nexys 2 Board

NET "clk" LOC = "B8";

## Leds

NET "Led<0>" LOC = "J14";

NET "Led<1>" LOC = "J15";

NET "Led<2>" LOC = "K15";

NET "Led<3>" LOC = "K14";

NET "Led<4>" LOC = "E17";

NET "Led<5>" LOC = "P15";

NET "Led<6>" LOC = "F4";

NET "Led<7>" LOC = "R4";

## Switches

NET "sw<0>" LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW0

NET "sw<1>" LOC = "H18"; # Bank = 1, Pin name = IP/VREF\_1, Type = VREF, Sch name = SW1

NET "sw<2>" LOC = "K18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW2

NET "sw<3>" LOC = "K17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW3

NET "sw<4>" LOC = "L14"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW4

NET "sw<5>" LOC = "L13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW5

NET "sw<6>" LOC = "N17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW6

NET "sw<7>" LOC = "R17"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW7

## Buttons

NET "btn<0>" 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 "btn<2>" LOC = "E18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN2

NET "btn<3>" LOC = "H13"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = BTN3

**Anomalies**

I got very confused when adding my UCF file. For one, it added my testbench as simulation only and I had to add it to the implementation as well. And the first time I added the UCF, I accidentally added it to the wrong module. So it disappeared. I had to remove all the modules and add them again one by one to delete the bad UCF and add the right one.