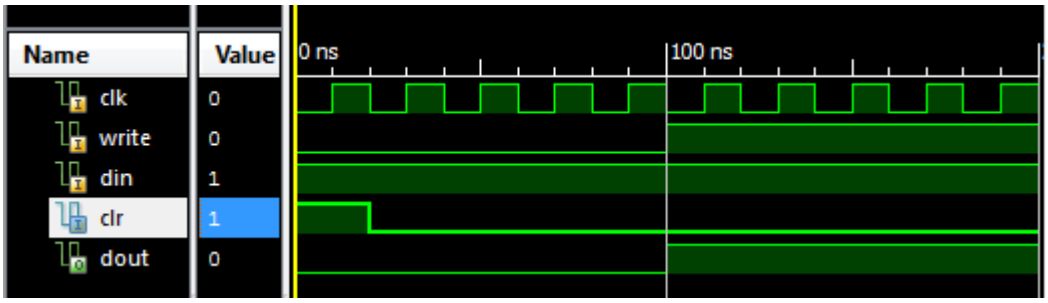
David DeGraw
3/3/2015
Lab 7: Register FIle

1 Bit Register:

```
module fd_c(q, clk, d, clr);
   input clk, clr, d;
   output reg q;
   always @(negedge clk)
   if (clr) q <= 0;
   else q <= d;
endmodule

module mux21(q, i0, i1, sel);
   output q;
   input i0, i1, sel;

   assign q = (~sel & i0) | (sel & i1);

endmodule

module register1bit(clk, clr, write, din, dout);
   output dout;
   input clk, write, din, clr;
   wire mout, dout;

   mux21 m1 (mout, dout, din, write);

   fd_c register (dout, clk, mout, clr);

endmodule
```

```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add din 1
isim force add write 0 -time 0 -value 1 -time 100ns

run 200ns
```

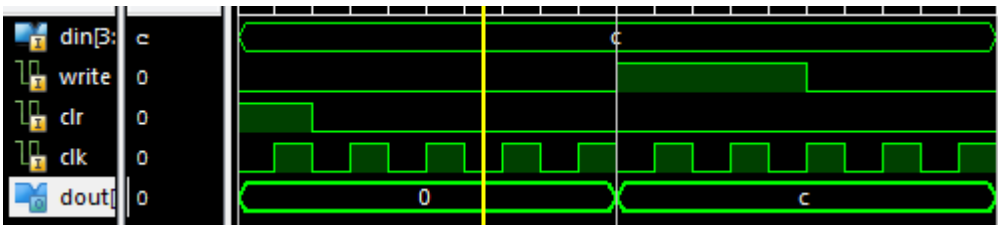


David DeGraw
3/3/2015
Lab 7: Register FIle

4 Bit Register:

```
module register4bit(dout, din, write, clr, clk);
   output [3:0] dout;
   input [3:0] din;
   input write, clr, clk;
   wire [3:0] dout;

   register1bit r1 (clk, clr, write, din[3], dout[3]);
   register1bit r2 (clk, clr, write, din[2], dout[2]);
   register1bit r3 (clk, clr, write, din[1], dout[1]);
   register1bit r4 (clk, clr, write, din[0], dout[0]);

endmodule
```

```
wave add / -radix hex

isim force add clk 0 -time 0 -value 1 -time 10ns -repeat 20ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add din 1100
isim force add write 0 -time 0 -value 1 -time 100ns -value 0 -time 150ns

run 200ns
```

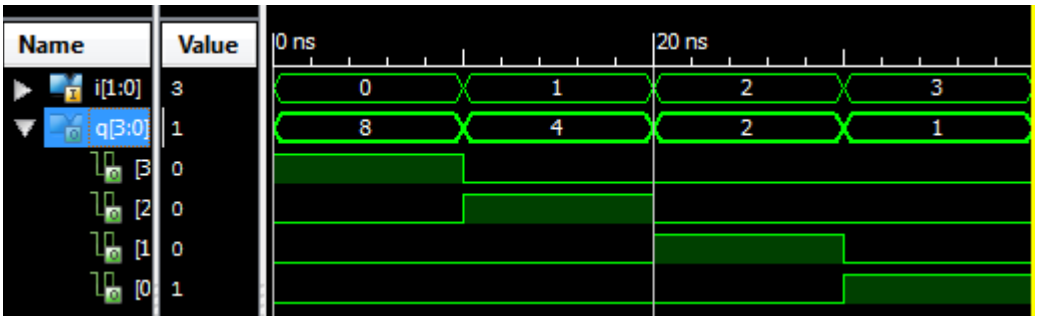


David DeGraw
3/3/2015
Lab 7: Register FIle

2:4 Decoder:

```
module decoder24(q, i);
   output [3:0] q;
   input [1:0] i;

   assign q = 4'b1000 >> i;

endmodule
```

```
wave add / -radix hex

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

run 40ns
```

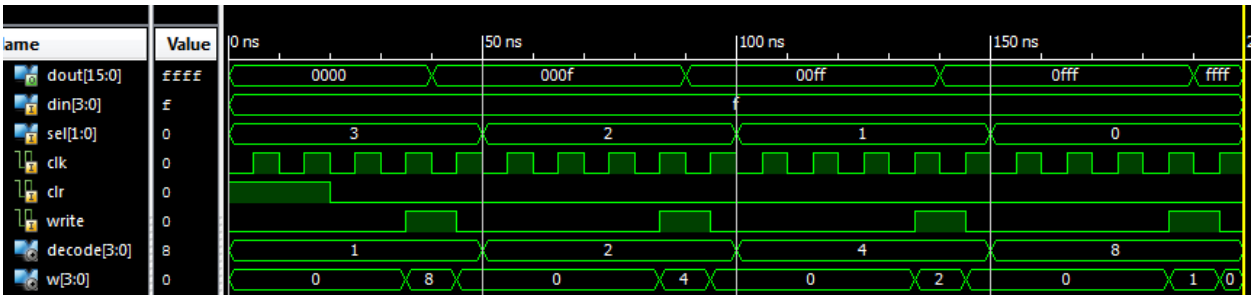


David DeGraw
3/3/2015
Lab 7: Register FIle

4 Bit x 4 Word Register:

```
module register4word(dout, din, write, clk, clr, sel);
   output [15:0] dout;
   input [3:0] din;
   input [1:0] sel;
   input clk, clr, write;
   wire [3:0] decode;
   wire w[3:0];

   decoder24 d1(decode, sel);

   assign w[0] = decode[3] & write;
   assign w[1] = decode[2] & write;
   assign w[2] = decode[1] & write;
   assign w[3] = decode[0] & write;

   register4bit r1 (dout[15:12], din, w[0], clr, clk);
   register4bit r2 (dout[11:8], din, w[1], clr, clk);
   register4bit r3 (dout[7:4], din, w[2], clr, clk);
   register4bit r4 (dout[3:0], din, w[3], clr, clk);

endmodule
```

```
wave add / -radix hex

isim force add din 1111
isim force add clk 0 -time 0 -value 1 -time 5ns -repeat 10ns
isim force add clr 1 -time 0 -value 0 -time 20ns
isim force add sel 11 -time 0 -value 10 -time 50ns -value 01 -
time 100ns -value 00 -time 150ns
isim force add write 0 -time 0 -value 1 -time 35ns -value 0 -
time 45ns -repeat 50ns

run 200ns
```

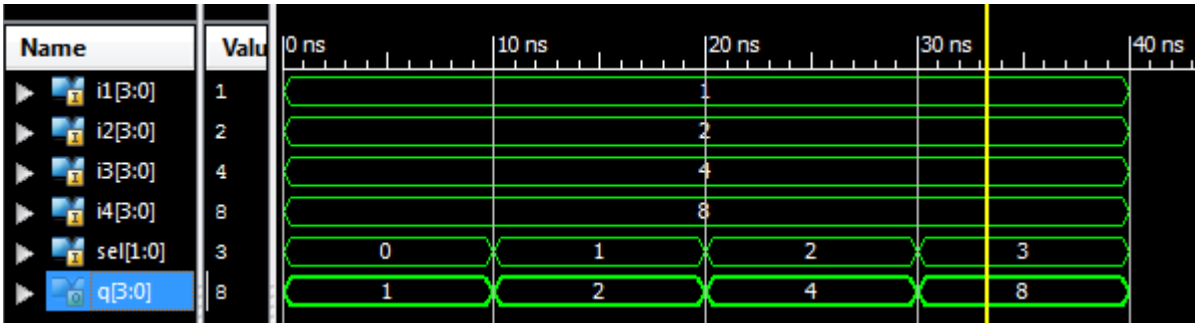


David DeGraw
3/3/2015
Lab 7: Register FIle

16:4 MUX:

```
module mux164(q, i1, i2, i3, i4, sel);
   output [3:0] q;
   input [3:0] i1, i2, i3, i4;
   input [1:0] sel;

   assign q = (sel == 2'b00) ? i1 :
              (sel == 2'b01) ? i2 :
              (sel == 2'b10) ? i3 :
              i4;
endmodule
```

```
wave add / -radix hex

isim force add i1 0001
isim force add i2 0010
isim force add i3 0100
isim force add i4 1000
isim force add sel 00 -time 0 -value 01 -time 10ns -value 10 -
time 20ns -value 11 -time 30ns

run 40ns
```



David DeGraw
3/3/2015
Lab 7: Register FIle

Complete Four Word by Four Bit Register:

```
module registerComplete (q, din, clk, clr, write, dsel, msel);
	output [3:0] q;
	input clk, clr, write;
	input [1:0] dsel, msel;
	input [3:0] din;
	wire [3:0] w1, w2, w3, w4;

	register4word r4w({w1, w2, w3, w4}, din, write, clk, clr
, dsel);

	mux164 m1 (q, w1, w2, w3, w4, msel);

endmodule
```

```
module testBench (q, leds, din, clk, clr, write, dsel, msel);
	output [3:0] q, leds;
	input clk, clr, write;
	input [1:0] dsel, msel;
	input [3:0] din;

	registerComplete r1 (q, din, clk, clr, write, dsel, msel);

	assign leds = din;

endmodule
```

David DeGraw
3/3/2015
Lab 7: Register FIle

```
## clock pin for Nexys 2 Board
NET clk   LOC = "B8"; # Bank = 0, Pin name = IP_L13P_0/GCLK8, Ty
##NET "clk1" LOC = "U9"; # Bank = 2, Pin name = IO_L13P_2/D4/GCI
#
## Leds
NET q[0]  LOC = "J14"; # Bank = 1, Pin name = IO_L14N_1/A3/RHCLI
NET q[1]  LOC = "J15"; # Bank = 1, Pin name = IO_L14P_1/A4/RHCLI
NET q[2]  LOC = "K15"; # Bank = 1, Pin name = IO_L12P_1/A8/RHCLI
NET q[3]  LOC = "K14"; # Bank = 1, Pin name = IO_L12N_1/A7/RHCLI
NET leds[0]  LOC = "E17"; # Bank = 1, Pin name = IO, Type = I/O,
NET leds[1]  LOC = "P15"; # Bank = 1, Pin name = IO, Type = I/O,
NET leds[2]  LOC = "F4";  # Bank = 3, Pin name = IO, Type = I/O,
NET leds[3]  LOC = "R4";  # Bank = 3, Pin name = IO/VREF_3, Type

## Switches
NET msel[0] LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT
NET msel[1] LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type
NET dsel[0] LOC = "K18"; # Bank = 1, Pin name = IP, Type = INPUT
NET dsel[1] LOC = "K17"; # Bank = 1, Pin name = IP, Type = INPUT
NET din[0] LOC = "L14"; # Bank = 1, Pin name = IP, Type = INPUT,
NET din[1] LOC = "L13"; # Bank = 1, Pin name = IP, Type = INPUT,
NET din[2] LOC = "N17"; # Bank = 1, Pin name = IP, Type = INPUT,
NET din[3] LOC = "R17"; # Bank = 1, Pin name = IP, Type = INPUT,

## Buttons
NET clr LOC = "B18"; # Bank = 1, Pin name = IP, Type = INPUT, Sc
#NET "btn<1>" LOC = "D18"; # Bank = 1, Pin name = IP/VREF_1, Typ
#NET "btn<2>" LOC = "E18"; # Bank = 1, Pin name = IP, Type = INP
NET write LOC = "H13"; # Bank = 1, Pin name = IP, Type = INPUT,
```

Anomalies:

None really. I was generating a bit file without using the ucf (even though I had it written, it just wasn't added to the project). It was confusing but not that hard to fix.