Victor Yuan

December 1, 2016

Lab 13 Shift Registers

Introduction:

In this lab two Shift Registers where designed and implemented into Verilog. The first Register in Experiment one, is a 2-Bit serial input parallel output shift registers. The second Register in experiment two is a 2-Bit parallel input and serial output shift register. Both of these shift registers will be using a J-K negative edge triggered flip-flop.

Team Member Responsibilities:

Lap Partner: Albert

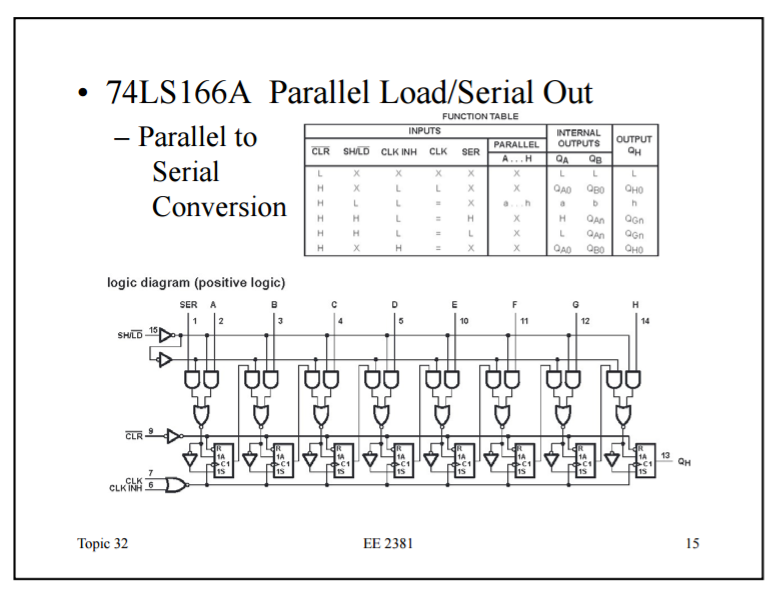
I designed the first shift register, Albert designed the Second. We both did sections of the coding

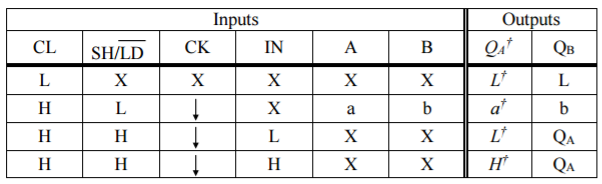
Materials:

Lab Computer, Verilog, putty.

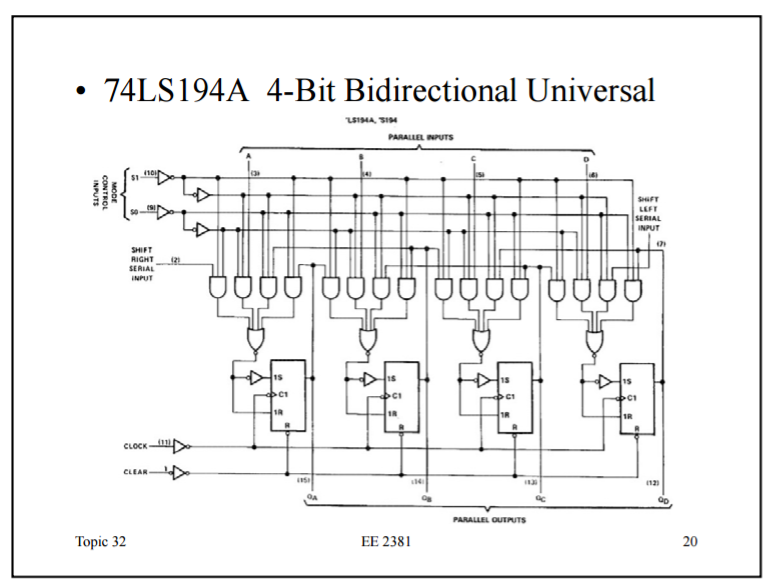
Procedure:

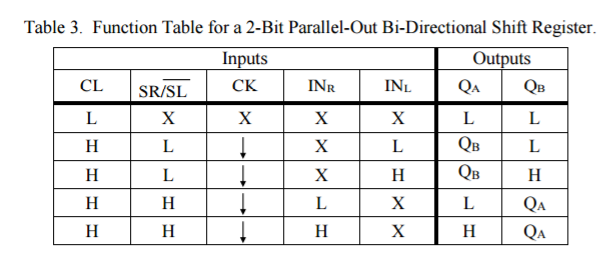
For Experiment 1, build a 2-bit parallel in, serial out shift register. The best way to do this is to look at a bigger shift register and simplify its design. The following is a slide from topic 32 showing a parallel in, serial out 74LS166A (JK Flip-Flop)



To simplify the desgin from a 8 bit to a 2 bit is unplug the output from the 2nd jk flip-flop from te left. Next, translate the desgin into verilog code. For the stimulus patern set CK to switch every 100 Time intervals, have CL begin at 0 then switch to 1 for the rest of the simulation then have SH,IN A, B have a different input every 200 Time intervals. Run the simulation and compare the results with table 2 as shown below: 

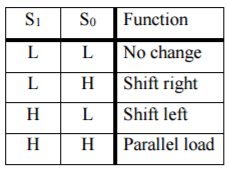
For Experiment 2, build a 2-bit serial in, parallel out shift register. Using the following 7419 4-bit Serial in/Parallel out Shift register. Disconect the output from the 2nd jk flip-flop from the group of 4-AND gates of the 3rd JK flipflop.Disconnect the output from the 3rd jk Flipflop from the group of 4-AND gates of the 2nd JKFlipflop. This should make Qa,Qb be the parallel outputs of a 2-bit Shift register.



Next, translate the desgin into verilog code. For the stimulus patern set CK to switch every 100 Time intervals, have CL begin at 0 then switch to 1, have SR,INr,INl jave a different input every 200 time interval. Run the simulation and compare the results with table 3 as shown below: 

Questions:

1. How would you extend your design of a 2-bit parallel-out shift register to a 4-bit parallel-out shift register?
   1. Reconnect the output from the 2nd JK flip flop to the group of 4-AND gates of the 3rd jk Flip-flop, and reconnect the output of the 3rd jk flip-flop back to back to the group of 4-AND gates. The finale product should have 4 jk flip-flops 2 more than the original 2.
2. How would you implement a 2-bit parallel-in, parallel-out, bi-directional (universal) shift register? Hint: You will need two mode control signals, S1 and S0. The table below describes how the mode controls are intended to function. What happens to the multiplexers?
   1. I’m going to modify the serial input on the register built in experiment 2. By having two AND gates and 1 NOR gate, this is the parallel input method from experiment one. And having each and gate take input A and B and Pairing A and B with S0 and S1. Then NOR the out puts from the two AND gates and sending that value into the inputs of the JK flip-flop. The multiplexers acts as the gate keepers.



Conclusions:

The goal of designing, modeling, and simulating of two shift registers was successful.

Code and data:

// experiment 1

module jkff\_74ALS107A(j,k,cl,ck,q,qbar);

input j,k,cl,ck; // inputs

output q,qbar; // outputs

parameter delay\_rise=0; // rise time delay

parameter delay\_fall=0; // fall time delay

supply1 pr; // constant logic 1 value

and #(0) (and1,qqbar,nand2,pr),

(and2,qqbar,ck,pr),

(and3,qq,nand1,cl),

(and4,qq,ck,cl);

nand #(1) (nand1,pr,qq,k,ck),

(nand2,cl,qqbar,j,ck);

nor #(0) (qq,and1,and2),

(qqbar,and3,and4);

buf #(delay\_rise,delay\_fall)

(q,qq),

(qbar,qqbar);

endmodule

module Main;

reg CL, SH,CK,IN, A, B;

wire qa,qb;

not U1a(v1,SH);

not U2a(a1,v1);

not U3a(v2,v1);

and U4a(v3,a1,IN);

and U5a(a2,v1,A);

and U6a(v4,v2,qa);

and U7a(a3,v1,B);

or U8a(v5,v3,a2);

or U9a(a4,v4,a3);

not U10a(a5,v5);

not U11a(v6,a4);

jkff\_74ALS107A U1B(v5,a5,CL,CK,qa,qabar);

jkff\_74ALS107A U2B(a4,v6,CL,CK,qb,qbbar);

initial CK=1;

always #100 CK=~CK;

initial begin

$monitor($time, "CL=%b SH=%b IN=%b A=%b B=%b qa=%b qb=%b ",CL,SH,IN,A,B,qa,qb);

#200 CL = 0; SH=0; IN=0; A=0; B=0;

#200 CL = 1; SH=0; IN=0; A=0; B=0;

#200 CL = 1; SH=0; IN=0; A=0; B=1;

#200 CL = 1; SH=0; IN=0; A=1; B=0;

#200 CL = 1; SH=0; IN=0; A=1; B=1;

#200 CL = 1; SH=0; IN=1; A=0; B=0;

#200 CL = 1; SH=0; IN=1; A=0; B=1;

#200 CL = 1; SH=0; IN=1; A=1; B=0;

#200 CL = 1; SH=0; IN=1; A=1; B=1;

#200 CL = 1; SH=1; IN=0; A=0; B=0;

#200 CL = 1; SH=1; IN=0; A=0; B=1;

#200 CL = 1; SH=1; IN=0; A=1; B=0;

#200 CL = 1; SH=1; IN=0; A=1; B=1;

#200 CL = 1; SH=1; IN=1; A=0; B=0;

#200 CL = 1; SH=1; IN=1; A=0; B=1;

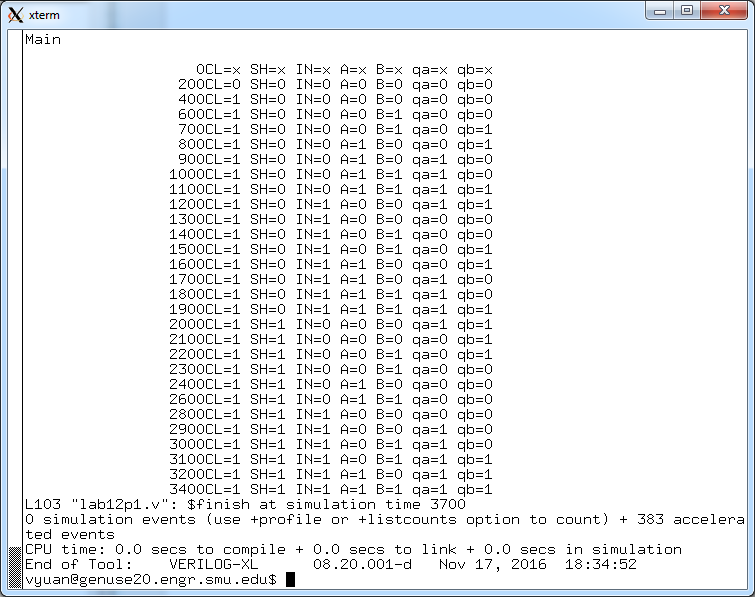
#200 CL = 1; SH=1; IN=1; A=1; B=0;

#200 CL = 1; SH=1; IN=1; A=1; B=1;

#300 $finish;

end

endmodule



//Experiment part 2

module jkff\_74ALS107A(j,k,cl,ck,q,qbar);

input j,k,cl,ck; // inputs

output q,qbar; // outputs

parameter delay\_rise=0; // rise time delay

parameter delay\_fall=0; // fall time delay

supply1 pr; // constant logic 1 value

and #(0) (and1,qqbar,nand2,pr),

(and2,qqbar,ck,pr),

(and3,qq,nand1,cl),

(and4,qq,ck,cl);

nand #(1) (nand1,pr,qq,k,ck),

(nand2,cl,qqbar,j,ck);

nor #(0) (qq,and1,and2),

(qqbar,and3,and4);

buf #(delay\_rise,delay\_fall)

(q,qq),

(qbar,qqbar);

endmodule

module Main;

reg SR, CL, CK, INr, INl;

wire qa, qb;

not u1a (a1,SR);

not u2a (a2,a1);

and u3a(a3,INr,a2);

and u4a(a4,a1,qb);

and u5a(a5,qa,a2);

and u6a(a6,a1,INl);

or u7a(a7,a3,a4);

or u8a(a8,a5,a6);

//or u9a(a9,CK,a1);

not u10a(a10,a7);

not u11a(a11,a8);

jkff\_74ALS107A U1B(a7,a10,CL,CK,qa,qabar);

jkff\_74ALS107A U2B(a8,a11,CL,CK,qb,qbbar);

initial CK=1;

always #100 CK=~CK;

initial begin

$monitor($time, "CL=%b SR=%b INr=%b INl=%b qa=%b qb=%b ",CL,SR,INr,INl,qa,qb);

#200 CL = 0; SR=0; INr=0;INl=0;

#200 CL = 1; SR=0; INr=0;INl=0;

#200 CL = 1; SR=0; INr=0;INl=1;

#200 CL = 1; SR=0; INr=1;INl=0;

#200 CL = 1; SR=0; INr=1;INl=1;

#200 CL = 1; SR=1; INr=0;INl=0;

#200 CL = 1; SR=1; INr=0;INl=1;

#200 CL = 1; SR=1; INr=1;INl=0;

#200 CL = 1; SR=1; INr=1;INl=1;

#300 $finish;

end

endmodule

