Data Storage Units

Calvin Reese
cjreese@fortlewis.edu

2/21/22

1 Introduction

This HW, we worked on integrating memory with our code.

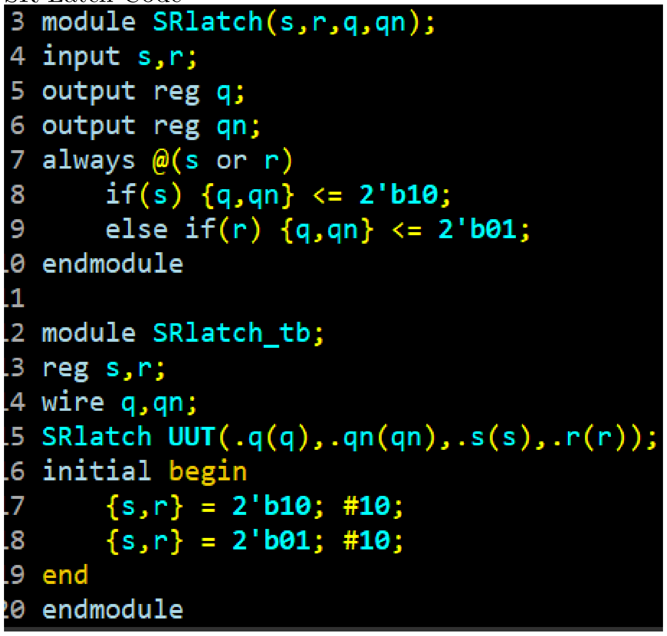
2 Materials and Methods

The tutorial for making these examples are in `http://www.yilectronics.com/Courses/CE433/s2022/lectures/week4_dataStorage/week4_DataStorage.html`

3 Results

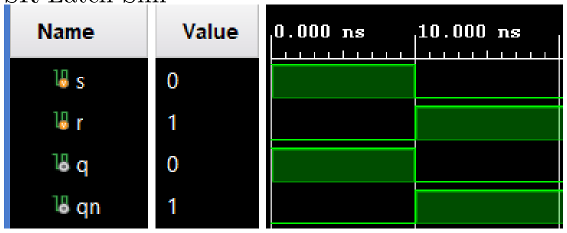
3.1 Task 1

SR Latch Code



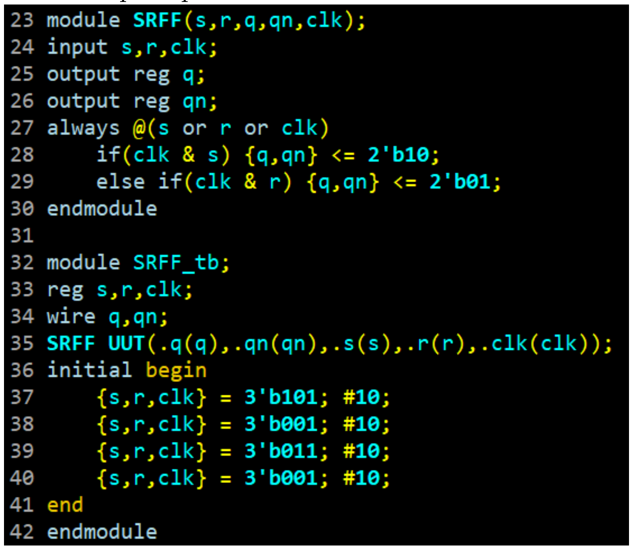
```
module SRlatch(s,r,q,qn);
input s,r;
output reg q;
output reg qn;
always @(s or r)
    if(s) {q,qn} <= 2'b10;
    else if(r) {q,qn} <= 2'b01;
endmodule

module SRlatch_tb;
reg s,r;
wire q,qn;
SRlatch UUT(.q(q),.qn(qn),.s(s),.r(r));
initial begin
    {s,r} = 2'b10; #10;
    {s,r} = 2'b01; #10;
end
endmodule
```

SR Latch Sim



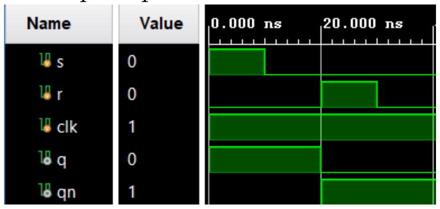
Name	Value
s	0
r	1
q	0
qn	1

SR Flip Flop Code



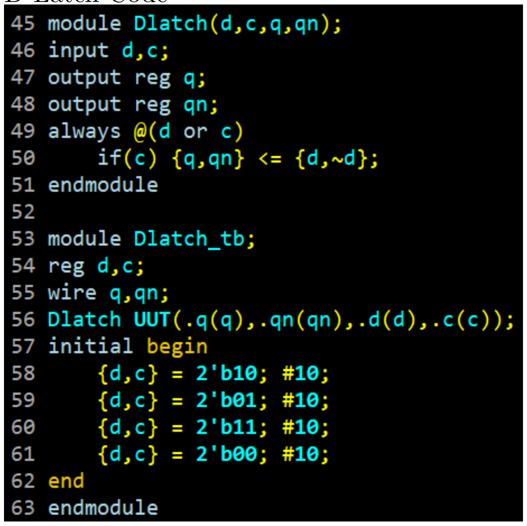
```verilog
module SRFF(s,r,q,qn,clk);
input s,r,clk;
output reg q;
output reg qn;
always @(s or r or clk)
    if(clk & s) {q,qn} <= 2'b10;
    else if(clk & r) {q,qn} <= 2'b01;
endmodule

module SRFF_tb;
reg s,r,clk;
wire q,qn;
SRFF UUT(.q(q),.qn(qn),.s(s),.r(r),.clk(clk));
initial begin
    {s,r,clk} = 3'b101; #10;
    {s,r,clk} = 3'b001; #10;
    {s,r,clk} = 3'b011; #10;
    {s,r,clk} = 3'b001; #10;
end
endmodule
```

SR Flip Flop Sim



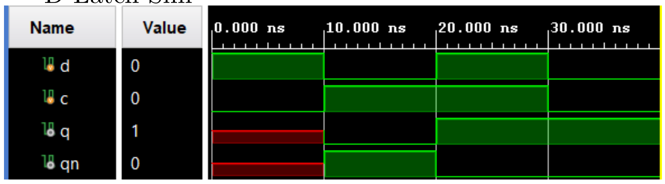
Name	Value
s	0
r	0
clk	1
q	0
qn	1

D Latch Code

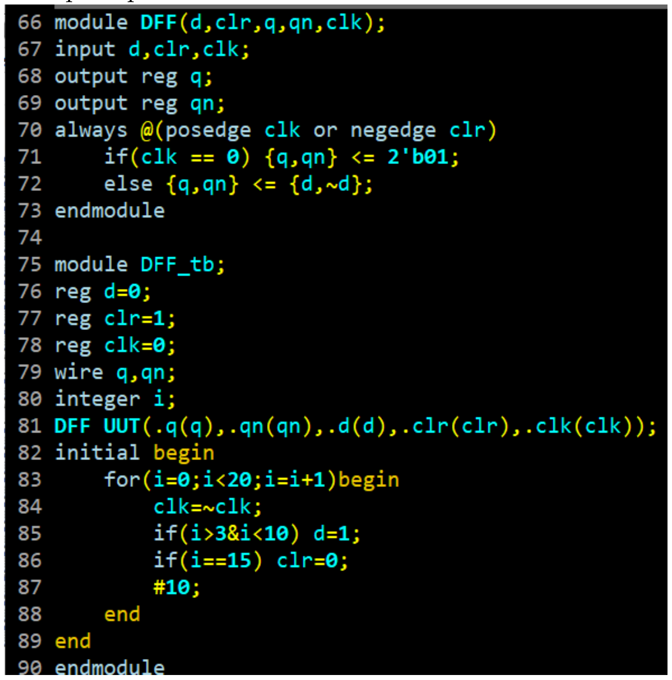


```verilog
module Dlatch(d,c,q,qn);
input d,c;
output reg q;
output reg qn;
always @(d or c)
    if(c) {q,qn} <= {d,~d};
endmodule

module Dlatch_tb;
reg d,c;
wire q,qn;
Dlatch UUT(.q(q),.qn(qn),.d(d),.c(c));
initial begin
    {d,c} = 2'b10; #10;
    {d,c} = 2'b01; #10;
    {d,c} = 2'b11; #10;
    {d,c} = 2'b00; #10;
end
endmodule
```

D Latch Sim

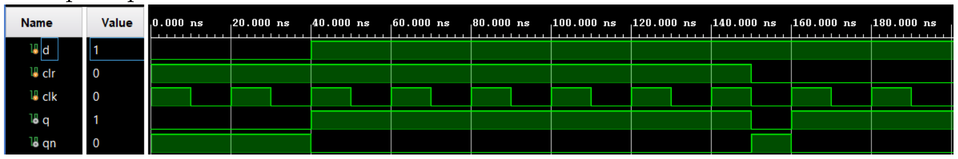


D Flip Flop Code



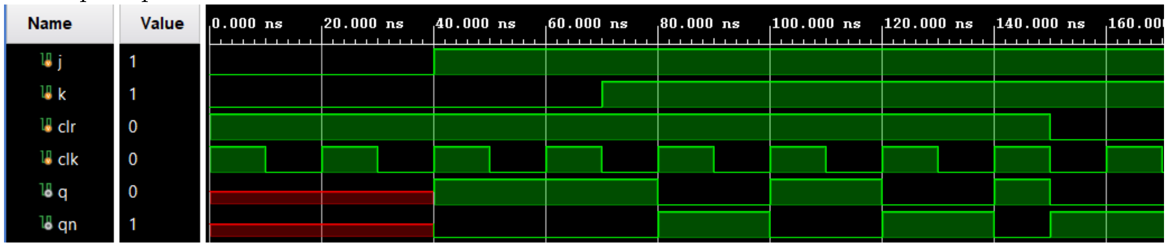
```
module DFF(d,clr,q,qn,clk);
input d,clr,clk;
output reg q;
output reg qn;
always @(posedge clk or negedge clr)
    if(clk == 0) {q,qn} <= 2'b01;
    else {q,qn} <= {d,~d};
endmodule

module DFF_tb;
reg d=0;
reg clr=1;
reg clk=0;
wire q,qn;
integer i;
DFF UUT(.q(q),.qn(qn),.d(d),.clr(clr),.clk(clk));
initial begin
    for(i=0;i<20;i=i+1)begin
        clk=~clk;
        if(i>3&i<10) d=1;
        if(i==15) clr=0;
        #10;
    end
end
endmodule
```

D Flip Flop Sim

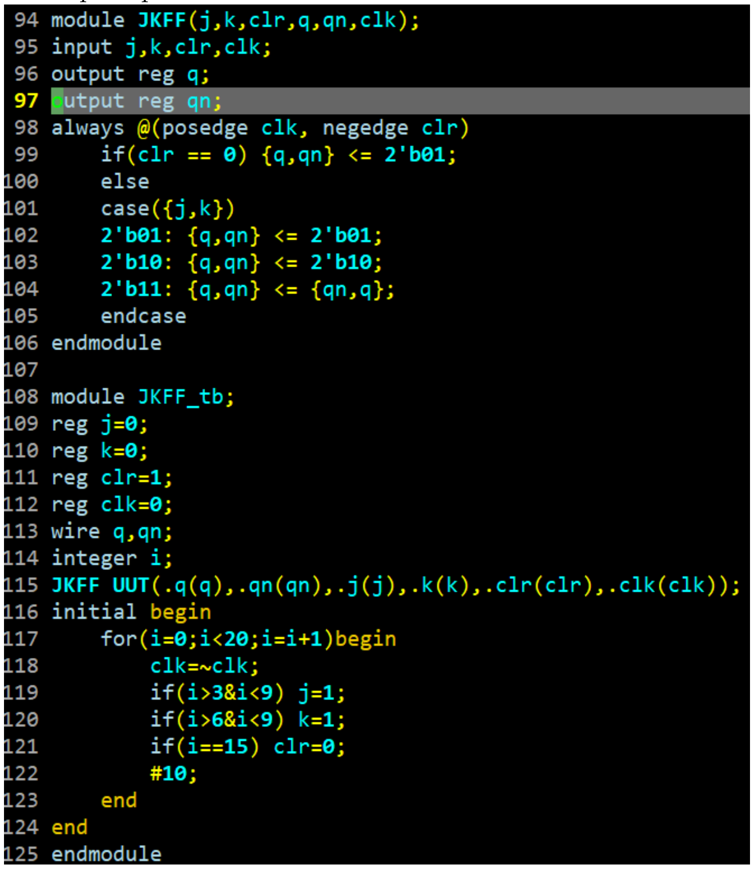


3.2 Task 2

JK Flip Flop Sim

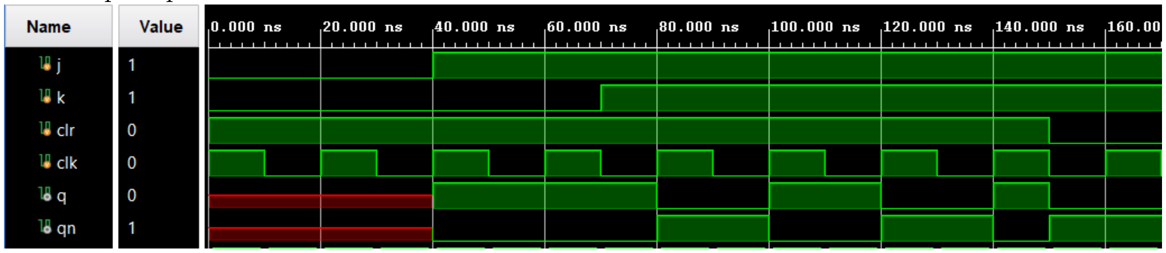


JK Flip Flop Code

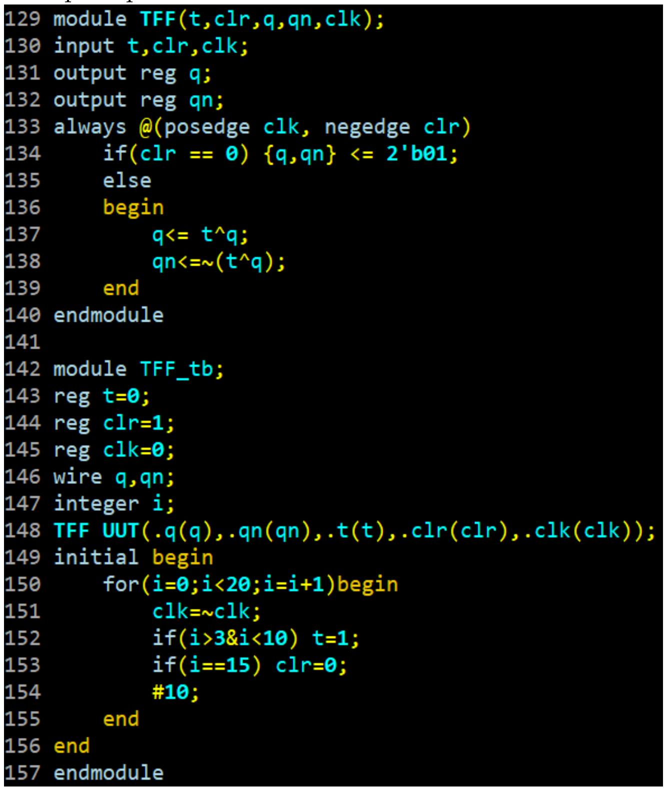


```
module JKFF(j,k,clr,q,qn,clk);
input j,k,clr,clk;
output reg q;
output reg qn;
always @(posedge clk, negedge clr)
    if(clr == 0) {q,qn} <= 2'b01;
    else
    case({j,k})
    2'b01: {q,qn} <= 2'b01;
    2'b10: {q,qn} <= 2'b10;
    2'b11: {q,qn} <= {qn,q};
    endcase
endmodule

module JKFF_tb;
reg j=0;
reg k=0;
reg clr=1;
reg clk=0;
wire q,qn;
integer i;
JKFF UUT(.q(q),.qn(qn),.j(j),.k(k),.clr(clr),.clk(clk));
initial begin
    for(i=0;i<20;i=i+1)begin
        clk=~clk;
        if(i>3&i<9) j=1;
        if(i>6&i<9) k=1;
        if(i==15) clr=0;
        #10;
    end
end
endmodule
```

T Flip Flop Sim



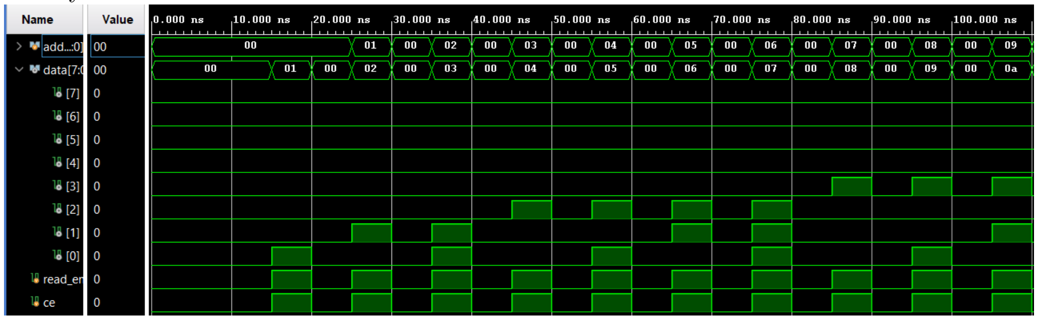
T Flip Flop Code



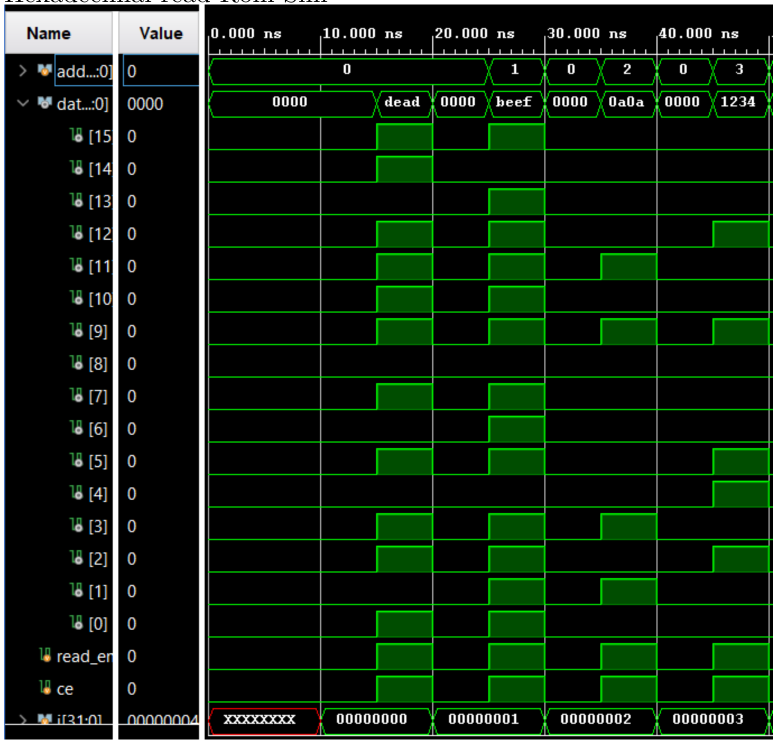
```
module TFF(t,clr,q,qn,clk);
input t,clr,clk;
output reg q;
output reg qn;
always @(posedge clk, negedge clr)
    if(clr == 0) {q,qn} <= 2'b01;
    else
    begin
        q<= t^q;
        qn<=~(t^q);
    end
endmodule

module TFF_tb;
reg t=0;
reg clr=1;
reg clk=0;
wire q,qn;
integer i;
TFF UUT(.q(q),.qn(qn),.t(t),.clr(clr),.clk(clk));
initial begin
    for(i=0;i<20;i=i+1)begin
        clk=~clk;
        if(i>3&i<10) t=1;
        if(i==15) clr=0;
        #10;
    end
end
endmodule
```

3.3 Task 3

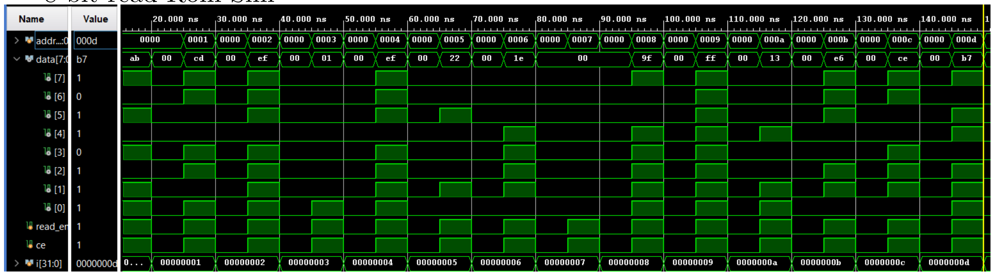
Binary read Rom Sim



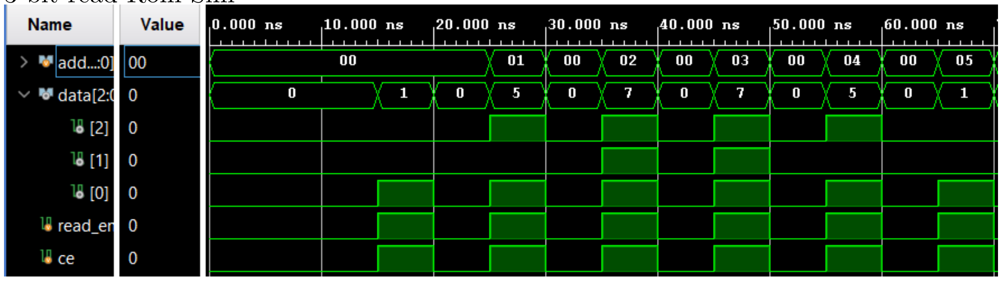
Hexadecimal read Rom Sim



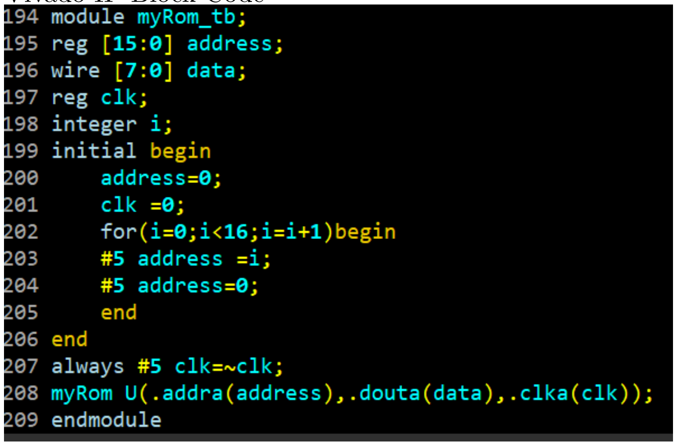
8-bit read Rom Sim



3-bit read Rom Sim

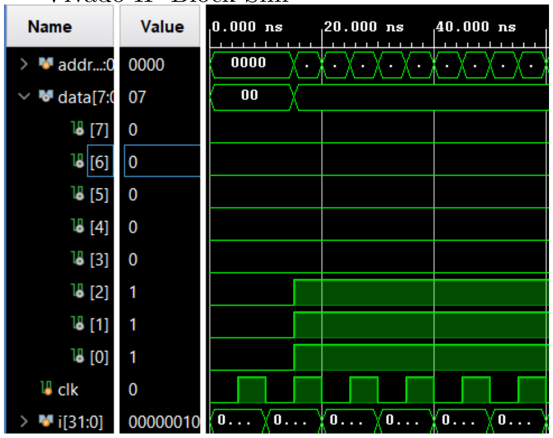


Vivado IP Block Code



```
module myRom_tb;
reg [15:0] address;
wire [7:0] data;
reg clk;
integer i;
initial begin
    address=0;
    clk =0;
    for(i=0;i<16;i=i+1)begin
    #5 address =i;
    #5 address=0;
    end
end
always #5 clk=~clk;
myRom U(.addra(address),.douta(data),.clka(clk));
endmodule
```

Vivado IP Block Sim



4 Discussion

All of this was fairly simple just to follow your tutorial. I had no trouble figuring out the problems.