

# 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](http://www.yilectronics.com/Courses/CE433/s2022/lectures/week4_dataStorage/week4_DataStorage.html)

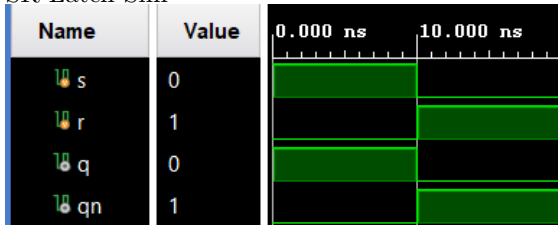
### 3 Results

#### 3.1 Task 1

SR Latch Code

```
3 module SRLatch(s,r,q,qn);
4 input s,r;
5 output reg q;
6 output reg qn;
7 always @(s or r)
8     if(s) {q,qn} <= 2'b10;
9     else if(r) {q,qn} <= 2'b01;
0 endmodule
1
2 module SRLatch_tb;
3 reg s,r;
4 wire q,qn;
5 SRLatch UUT(.q(q),.qn(qn),.s(s),.r(r));
6 initial begin
7     {s,r} = 2'b10; #10;
8     {s,r} = 2'b01; #10;
9 end
0 endmodule
```

SR Latch Sim



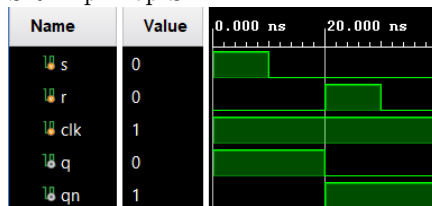
### SR Flip Flop Code

```

23 module SRFF(s,r,q,qn,clk);
24 input s,r,clk;
25 output reg q;
26 output reg qn;
27 always @(s or r or clk)
28     if(clk & s) {q,qn} <= 2'b10;
29     else if(clk & r) {q,qn} <= 2'b01;
30 endmodule
31
32 module SRFF_tb;
33 reg s,r,clk;
34 wire q,qn;
35 SRFF UUT(.q(q),.qn(qn),.s(s),.r(r),.clk(clk));
36 initial begin
37     {s,r,clk} = 3'b101; #10;
38     {s,r,clk} = 3'b001; #10;
39     {s,r,clk} = 3'b011; #10;
40     {s,r,clk} = 3'b001; #10;
41 end
42 endmodule

```

### SR Flip Flop Sim



### D Latch Code

```

45 module Dlatch(d,c,q,qn);
46 input d,c;
47 output reg q;
48 output reg qn;
49 always @(d or c)
50     if(c) {q,qn} <= {d,~d};
51 endmodule
52
53 module Dlatch_tb;
54 reg d,c;
55 wire q,qn;
56 Dlatch UUT(.q(q),.qn(qn),.d(d),.c(c));
57 initial begin
58     {d,c} = 2'b10; #10;
59     {d,c} = 2'b01; #10;
60     {d,c} = 2'b11; #10;
61     {d,c} = 2'b00; #10;
62 end
63 endmodule

```

### D Latch Sim

| Name | Value | 0.000 ns | 10.000 ns | 20.000 ns | 30.000 ns |
|------|-------|----------|-----------|-----------|-----------|
| d    | 0     |          |           |           |           |
| c    | 0     |          |           |           |           |
| q    | 1     |          |           |           |           |
| qn   | 0     |          |           |           |           |

### D Flip Flop Code

```

66 module DFF(d,clr,q,qn,clk);
67 input d,clr,clk;
68 output reg q;
69 output reg qn;
70 always @(posedge clk or negedge clr)
71     if(clk == 0) {q,qn} <= 2'b01;
72     else {q,qn} <= {d,~d};
73 endmodule
74
75 module DFF_tb;
76 reg d=0;
77 reg clr=1;
78 reg clk=0;
79 wire q,qn;
80 integer i;
81 DFF UUT(.q(q),.qn(qn),.d(d),.clr(clr),.clk(clk));
82 initial begin
83     for(i=0;i<20;i=i+1)begin
84         clk=~clk;
85         if(i>3&&i<10) d=1;
86         if(i==15) clr=0;
87         #10;
88     end
89 end
90 endmodule

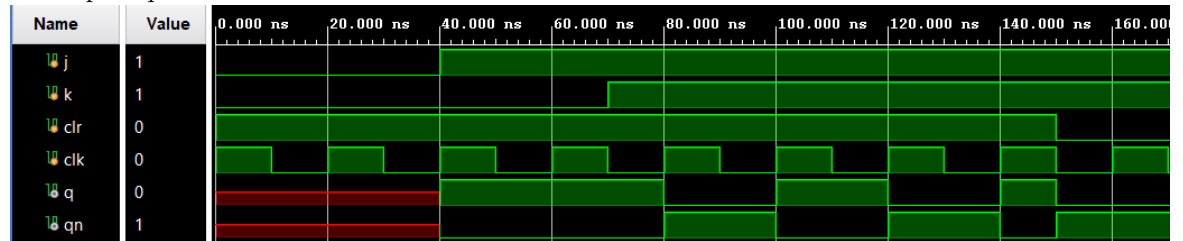
```

### D Flip Flop Sim

| Name | Value | 0.000 ns | 20.000 ns | 40.000 ns | 60.000 ns | 80.000 ns | 100.000 ns | 120.000 ns | 140.000 ns | 160.000 ns | 180.000 ns |
|------|-------|----------|-----------|-----------|-----------|-----------|------------|------------|------------|------------|------------|
| d    | 1     |          |           |           |           |           |            |            |            |            |            |
| clr  | 0     |          |           |           |           |           |            |            |            |            |            |
| clk  | 0     |          |           |           |           |           |            |            |            |            |            |
| q    | 1     |          |           |           |           |           |            |            |            |            |            |
| qn   | 0     |          |           |           |           |           |            |            |            |            |            |

### 3.2 Task 2

JK Flip Flop Sim



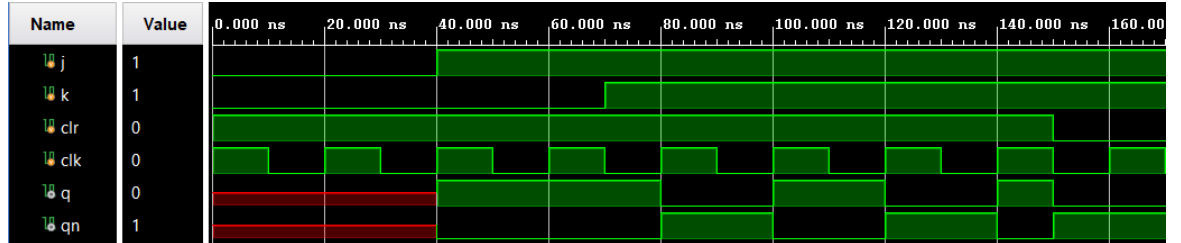
JK Flip Flop Code

```

94 module JKFF(j,k,clr,q,qn,clk);
95 input j,k,clr,clk;
96 output reg q;
97 output reg qn;
98 always @(posedge clk, negedge clr)
99     if(clr == 0) {q,qn} <= 2'b01;
100     else
101     case({j,k})
102     2'b01: {q,qn} <= 2'b01;
103     2'b10: {q,qn} <= 2'b10;
104     2'b11: {q,qn} <= {qn,q};
105     endcase
106 endmodule
107
108 module JKFF_tb;
109 reg j=0;
110 reg k=0;
111 reg clr=1;
112 reg clk=0;
113 wire q,qn;
114 integer i;
115 JKFF UUT(.q(q),.qn(qn),.j(j),.k(k),.clr(clr),.clk(clk));
116 initial begin
117     for(i=0;i<20;i=i+1)begin
118         clk=~clk;
119         if(i>3&i<9) j=1;
120         if(i>6&i<9) k=1;
121         if(i==15) clr=0;
122         #10;
123     end
124 end
125 endmodule

```

### T Flip Flop Sim



### T Flip Flop Code

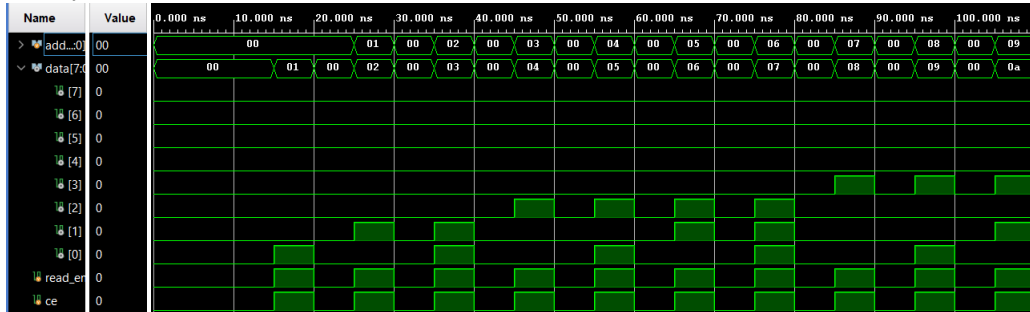
```

129 module TFF(t,clr,q,qn,clk);
130 input t,clr,clk;
131 output reg q;
132 output reg qn;
133 always @(posedge clk, negedge clr)
134     if(clr == 0) {q,qn} <= 2'b01;
135     else
136     begin
137         q<= t^q;
138         qn<=~(t^q);
139     end
140 endmodule
141
142 module TFF_tb;
143 reg t=0;
144 reg clr=1;
145 reg clk=0;
146 wire q,qn;
147 integer i;
148 TFF UUT(.q(q),.qn(qn),.t(t),.clr(clr),.clk(clk));
149 initial begin
150     for(i=0;i<20;i=i+1)begin
151         clk=~clk;
152         if(i>3&i<10) t=1;
153         if(i==15) clr=0;
154         #10;
155     end
156 end
157 endmodule

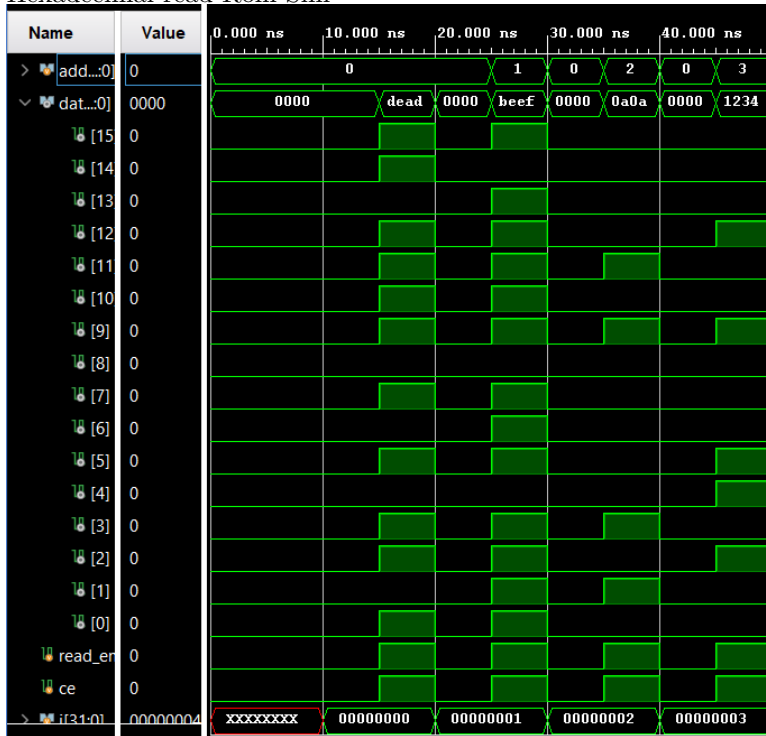
```

### 3.3 Task 3

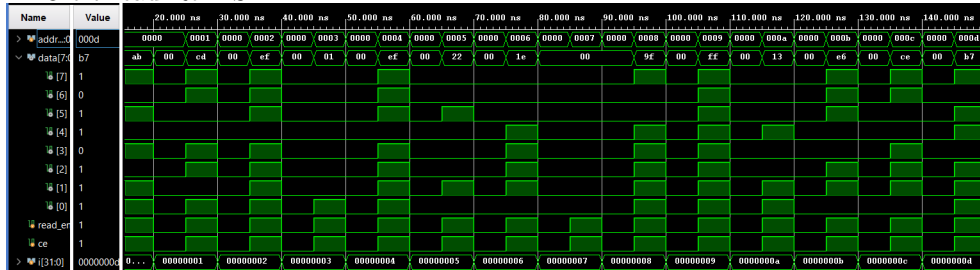
#### Binary read Rom Sim



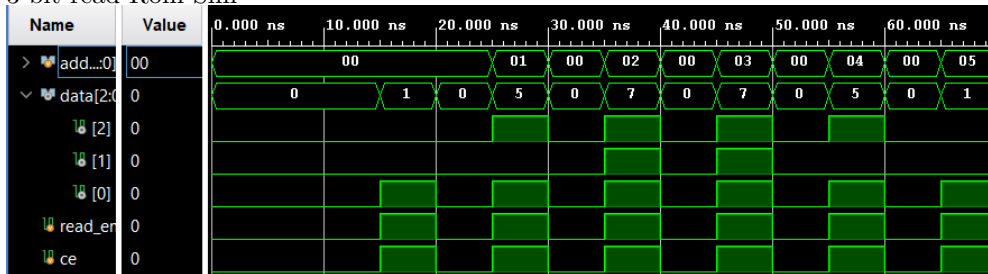
#### Hexadecimal read Rom Sim



### 8-bit read Rom Sim



### 3-bit read Rom Sim



### Vivado IP Block Code

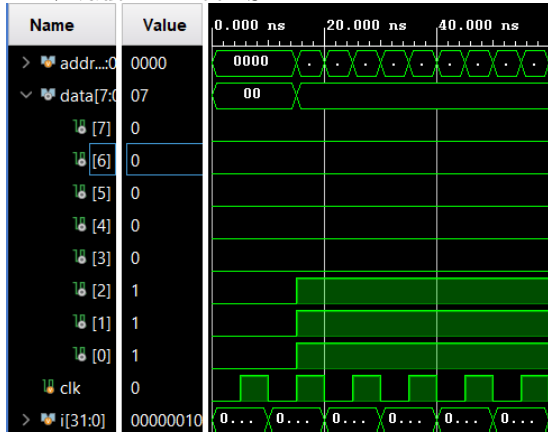
```

194 module myRom_tb;
195 reg [15:0] address;
196 wire [7:0] data;
197 reg clk;
198 integer i;
199 initial begin
200     address=0;
201     clk =0;
202     for(i=0;i<16;i=i+1)begin
203         #5 address =i;
204         #5 address=0;
205     end
206 end
207 always #5 clk=~clk;
208 myRom U(.addr(address),.dout(data),.clka(clk));
209 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.