

# ELEX 7660

---

ASS1

Nicholas Scott AKA "White Cheddar"  
A01255181 | JAN 30<sup>TH</sup>, 2024

## Contents

Problem 1 .....	2
'encoder.sv' .....	2
'encoder_tb.sv' .....	3
Problem 2 .....	6
'shiftReg.sv' .....	6
'shiftReg_tb.sv' .....	9

## Table of Figures

Figure 1 - 'encoder.sv' Compilation Report .....	3
Figure 2 - 'encoder.sv' RTL Netlist .....	3
Figure 3 - 'encoder_tb' Simulation Waveforms .....	5
Figure 4 - 'encoder.sv' Simulation Transcript .....	5
Figure 5 - 'shiftReg.sv' Compilation Report .....	7
Figure 6 - 'shiftReg.sv' RTL Netlist .....	8
Figure 7 - 'shiftReg_tb.sv' Simulation Waveforms .....	11
Figure 8 - 'shiftReg_tb.sv' Simulation Transcript .....	11

## Problem 1

I'm so excited.

### 'encoder.sv'

I wrote the following code for 'encoder.sv':

```
// ELEX 7660 Ass1
// Nicholas Scott AKA "White Cheddar"
// Jan. 30th, 2024
// Instructor: Sweet Bobby T

module encoder (output logic [1:0] y,
                output logic valid,
                input logic [3:0] a);

    always_comb begin
        // prepare for a case statement
        case (a)
            0 : begin
                y = 0;
                valid = 0;
            end
            1 : begin
                y = 0;
                valid = 1;
            end
            2, 3 : begin
                y = 1;
                valid = 1;
            end
            4, 5, 6, 7 : begin
                y = 2;
                valid = 1;
            end
            8, 9, 10, 11, 12, 13, 14, 15 : begin
                y = 3;
                valid = 1;
            end
        endcase
    end
endmodule
```

The code is very straightforward... I couldn't think of any meaningful comments to add. This code yielded the following compilation report and RTL Netlist:

Flow Summary	
<<Filter>>	
Flow Status	Successful - Wed Jan 31 14:18:06 2024
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	encoder
Top-level Entity Name	encoder
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	2 / 56,480 (< 1 %)
Total registers	0
Total pins	7 / 268 (3 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 1 - 'encoder.sv' Compilation Report

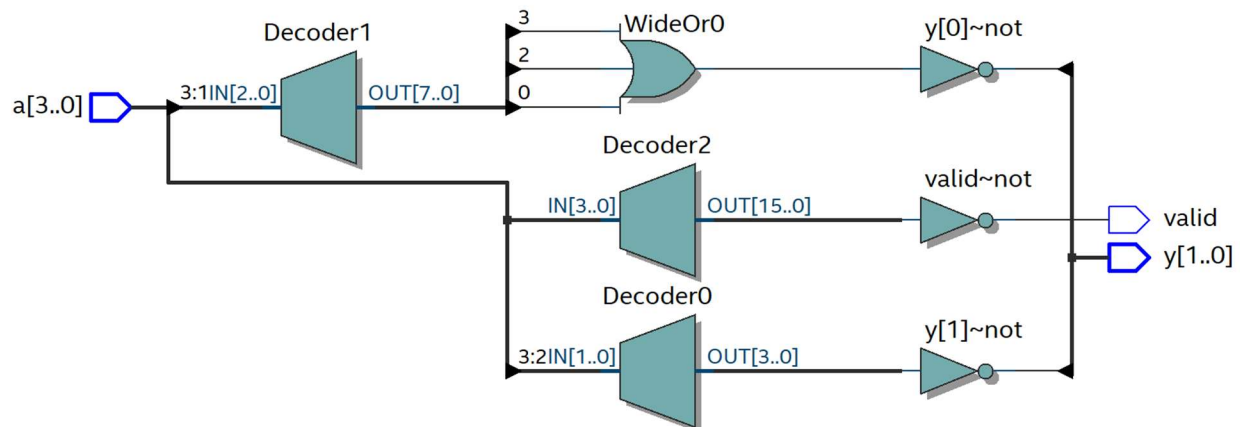


Figure 2 - 'encoder.sv' RTL Netlist

Now for the fun part...

### 'encoder\_tb.sv'

I implemented the following code for the encoder test bench:

```
// ELEX 7660 Ass1
// Nicholas Scott AKA "White Cheddar"
// Jan. 30th, 2024
// Instructor: Sweet Bobby T
```

```
`timescale 1ms/1ms
```

```
function logic check_value (int expected_value, int actual_value); // your handy  
lil function
```

```
    if (expected_value != actual_value) begin  
        $display("FAIL: expected value is %d => actual value is %d",  
expected_value, actual_value) ;  
        check_value = 1;  
    end else  
        check_value = 0;
```

```
endfunction
```

```
module encoder_tb ;
```

```
    // expected values, in sequence from a = 0 to a = 15  
    int yExpectedValues [0:15] = '{0,0,1,1,2,2,2,2,3,3,3,3,3,3,3};  
    int validExpectedValues [0:15] = '{0,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
```

```
    // encoder outputs  
    logic [1:0] y;  
    logic valid;
```

```
    logic [3:0] a = 0; // encoder input
```

```
    logic tb_fail = 0;  
    logic clk = 1;
```

```
    encoder dut (.*);
```

```
    initial begin
```

```
        for (int i = 0; i <=15; i++) begin
```

```
            a = i; // increment a from 0 to 15  
            repeat(1) @(negedge clk); // allow some time for output to stabilize
```

```
            // check both outputs  
            tb_fail |= check_value (yExpectedValues[i], y);  
            tb_fail |= check_value (validExpectedValues[i], valid);
```

```
            repeat(1) @(negedge clk); // allow some time again
```

```
        end
```

```
        if (tb_fail)
```

```
$display("DUMBASS!") ;  
else  
    $display("YEEEEEEAH BOIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII") ;  
$stop;  
  
end  
  
// 1 Hz clock  
always  
    #1000ms clk = ~clk ;  
  
endmodule
```

When simulated, this test bench yielded the following waveforms and simulation transcript. Not the y output changing as expected, *tb\_fail* remaining logic low, and the yeeeeeeah boiiiiiiiiiiiiiiiiiiii.

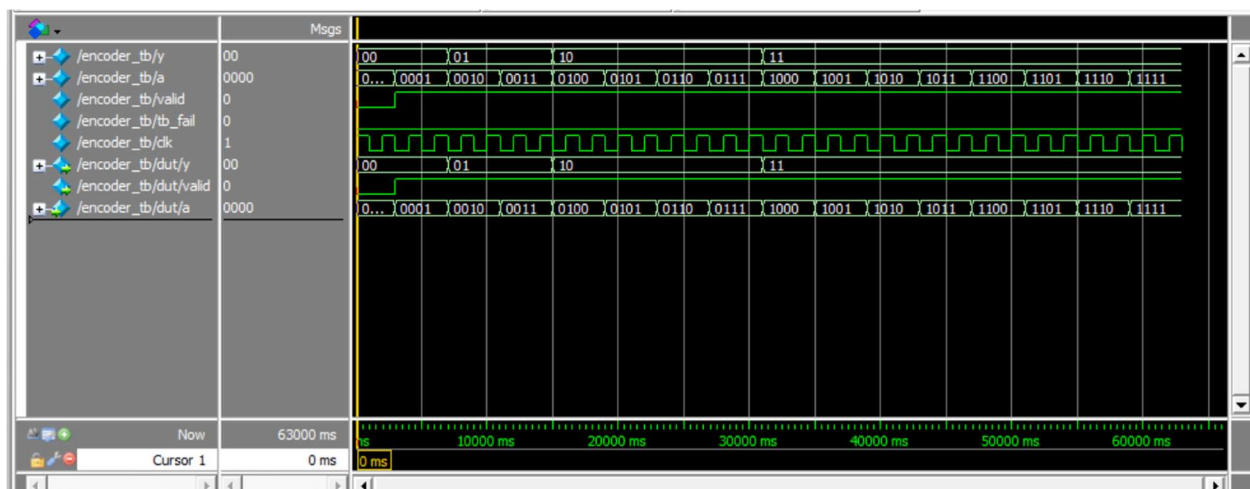


Figure 3 - 'encoder\_tb' Simulation Waveforms

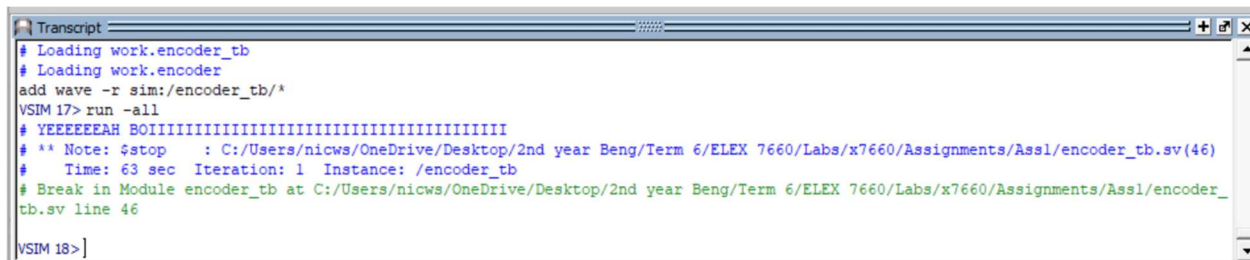


Figure 4 - 'encoder.sv' Simulation Transcript

## Problem 2

I then did the next thing.

### 'shiftReg.sv'

I wrote the following code for the shift register:

```
// ELEX 7660 Ass1
// Nicholas Scott AKA "White Cheddar"
// Jan. 30th, 2024
// Instructor: Sweet Bobby T

module shiftReg (output logic [7:0] q,
                 input logic [7:0] a, input logic [1:0] s,
                 input logic shiftIn, clk, reset_n);

    always_ff @( posedge clk or negedge reset_n ) begin

        if(!reset_n)
            q <= a; // what I assume you want the reset to do

        else begin // if active low reset not pressed

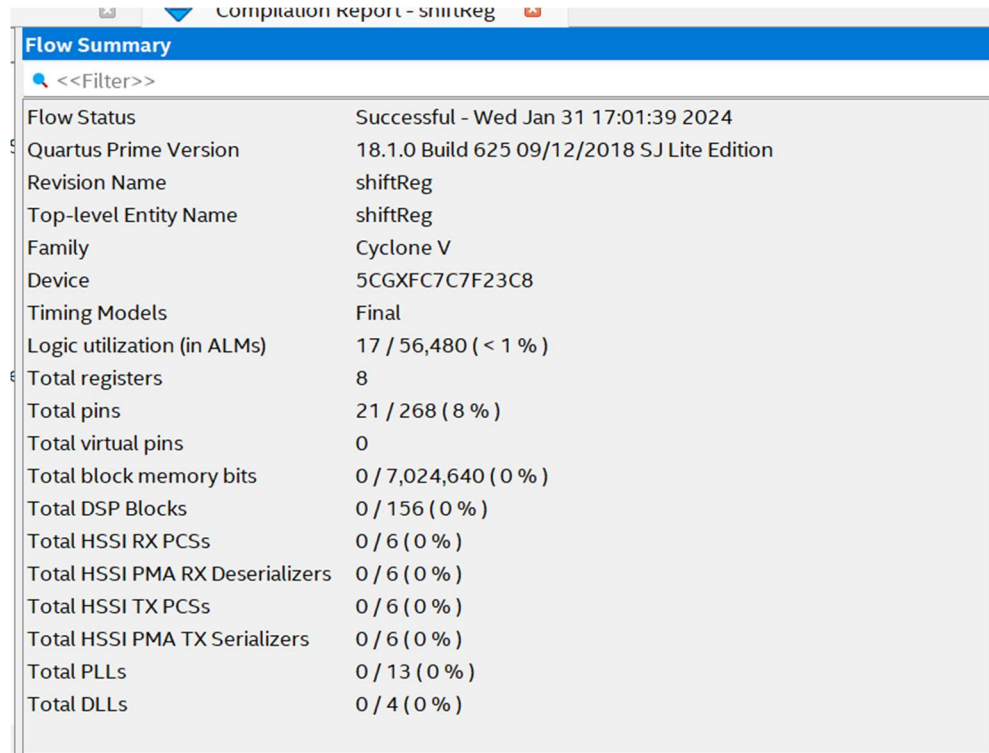
            case(s)
                0 : q <= a; // follow a
                1 : begin
                    q >>= 1; // make room for shiftIn variable
                    q[7] <= shiftIn; // get in there
                end
                2 : begin
                    q <<= 1; // make some room on the right side
                    q[0] <= shiftIn; // shift that thang in there
                end
                3 : q <= q; // maintain current state
            endcase

        end

    end

endmodule
```

I threw that shit in Quartus and compiled its ass. I got the following compilation report and RTL netlist as evidence:



Flow Summary	
<<Filter>>	
Flow Status	Successful - Wed Jan 31 17:01:39 2024
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	shiftReg
Top-level Entity Name	shiftReg
Family	Cyclone V
Device	5CGXFC7C7F23C8
Timing Models	Final
Logic utilization (in ALMs)	17 / 56,480 (< 1 %)
Total registers	8
Total pins	21 / 268 (8 %)
Total virtual pins	0
Total block memory bits	0 / 7,024,640 (0 %)
Total DSP Blocks	0 / 156 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 13 (0 %)
Total DLLs	0 / 4 (0 %)

Figure 5 - 'shiftReg.sv' Compilation Report



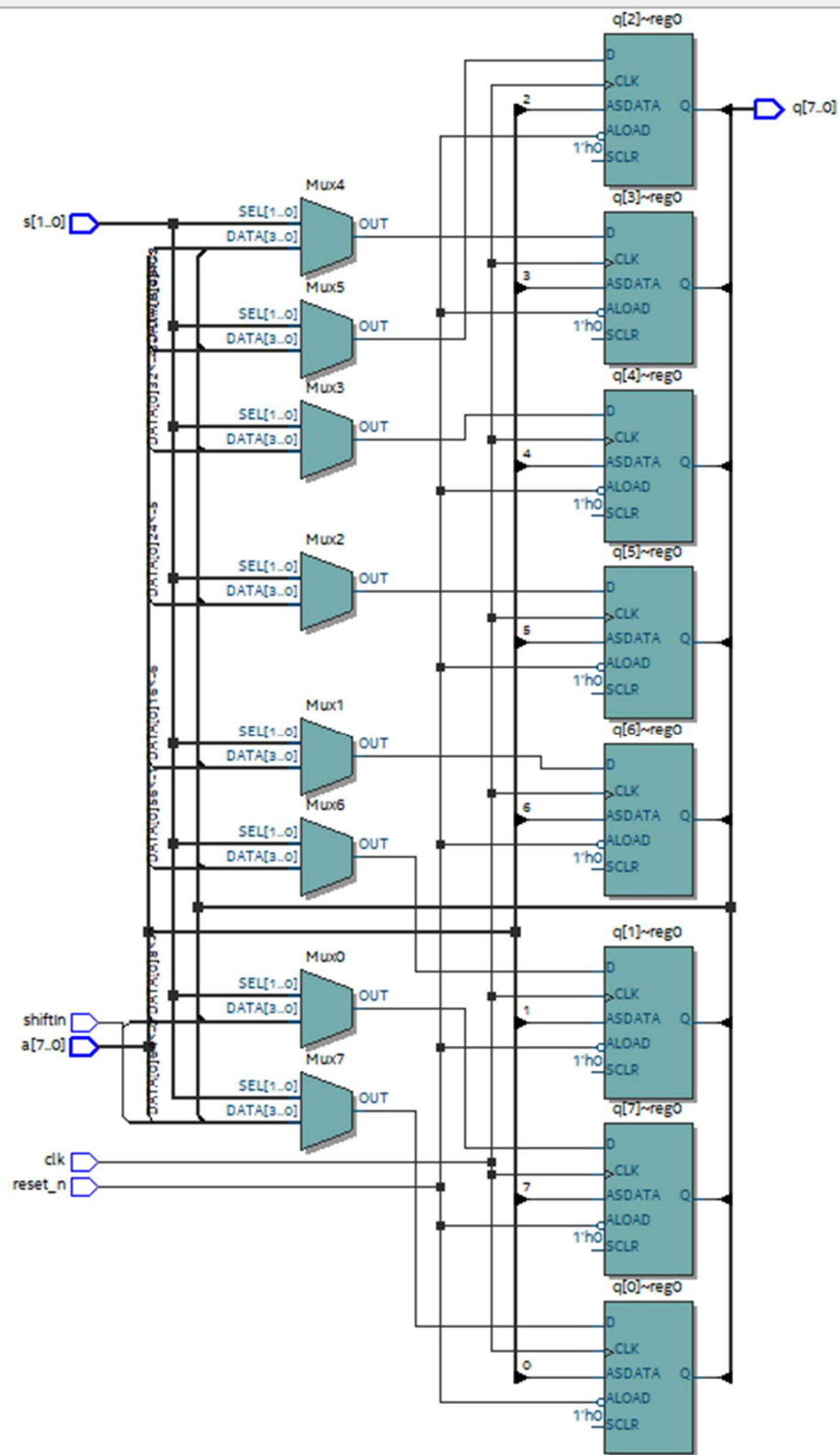


Figure 6 - 'shiftReg.sv' RTL Netlist

**'shiftReg\_tb.sv'**

I also made a cute little test bench.

```
// ELEX 7660 Ass1
// Nicholas Scott AKA "White Cheddar"
// Jan. 30th, 2024
// Instructor: Sweet Bobby T

`timescale 1ms/1ms

function logic check_value (int expected_value, int actual_value); // your handy lil function

    if (expected_value != actual_value) begin
        $display("FAIL: expected value is %d => actual value is %d", expected_value,
actual_value) ;
        check_value = 1;
    end else
        check_value = 0;

endfunction

module shiftReg_tb;

    logic [7:0] q;
    logic [7:0] a = 'b00001000; // almost palindromic
    logic [1:0] s;
    logic reset_n = 1;
    logic clk = 1;
    logic shiftIn;
    logic tb_fail = 0;

    // sequence of expected outputs
    logic [7:0] qExpectedValues[0:6] = '{ 'b00001000,
'b00000100, 'b10000010, 'b00000100, 'b00001001, 'b00001001, 'b00001000};

    shiftReg dut (.*)

    initial begin

        // set q to a
        s = 0;
        repeat(1) @(negedge clk);
        tb_fail |= check_value (qExpectedValues[0], q);

        // right shift in a 0
```

```
s = 1;
shiftIn = 0;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[1], q);

// right shift in a 1
s = 1;
shiftIn = 1;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[2], q);

// left shift in a 0
s = 2;
shiftIn = 0;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[3], q);

// left shift in a 1
s = 2;
shiftIn = 1;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[4], q);

// maintain q
s = 3;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[5], q);

// reset
reset_n = 0;
repeat(1) @(negedge clk);
tb_fail |= check_value (qExpectedValues[6], q);

if (tb_fail)
    $display("DUMBASS!") ;
else
    $display("YEEEEEEAH BOIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII") ;
$stop;

end

// 1 Hz clock
always
    #100ms clk = ~clk ;

endmodule
```

It ended up being a bit long but it did the trick. I slapped it into ModelSim and got the following results:

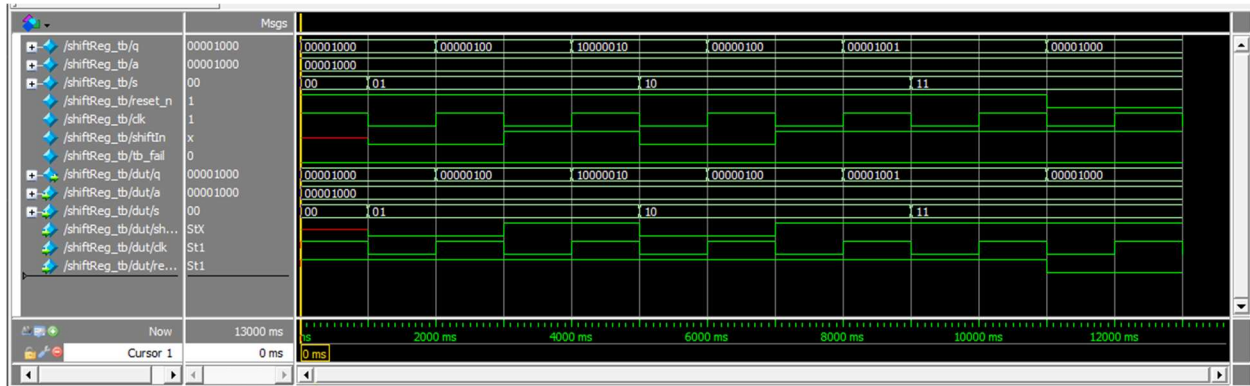


Figure 7 - 'shiftReg\_tb.sv' Simulation Waveforms

```
VSIM i8> run -all  
# YEEEEEEAH BOIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII  
** Note: $stop      : C:/Users/nicws/OneDrive/Desktop/2nd year Beng/Term 6/ELEX 7660/Labs/x7660/Assignments/Assl/shiftReg_tb.sv(85)  
# Time: 13 sec Iteration: 1 Instance: /shiftReg_tb  
# Break in Module shiftReg_tb at C:/Users/nicws/OneDrive/Desktop/2nd year Beng/Term 6/ELEX 7660/Labs/x7660/Assignments/Assl/shiftReg_tb.sv line 85  
  
VSIM i9>
```

Figure 8 - 'shiftReg\_tb.sv' Simulation Transcript