

lfsr.sv

The screenshot displays the Quartus II IDE interface. The top-left pane shows the 'Entity/Instance' hierarchy with 'Arria II GX: AUTO' and 'lfsr'. The top-right pane shows the 'Compilation Report - lfsr' and the 'lfsr.sv' source code. The bottom-left pane shows the 'Tasks' window with 'Compilation' selected, listing tasks like 'Compile Design', 'Analysis & Synthesis', 'Fitter (Place & Route)', 'Assembler (Generate program)', and 'Timing Analysis'. The bottom-right pane shows the 'Messages' window with a list of messages, including warnings and errors related to clock uncertainty, timing requirements, and the successful completion of the Quartus Prime EDA Netlist Writer.

```
1 // RTL Model for Linear Feedback Shift Register
2 module lfsr
3     #parameter N = 4 // Number of bits for LFSR
4     (
5         input logic clk, reset, load_seed,
6         input logic [N-1:0] seed_data,
7         output logic lfsr_done,
8         output logic [N-1:0] lfsr_data
9     );
10    // Internal feedback signal
11    logic feedback;
12    // Feedback polynomial based on N
13    // always_comb begin
14    //     case (N)
15    //         2: feedback = lfsr_data[1] ^ lfsr_data[0]; // x^2 + x + 1
16    //         3: feedback = lfsr_data[2] ^ lfsr_data[1]; // x^3 + x^2 + 1
17    //         4: feedback = lfsr_data[3] ^ lfsr_data[2]; // x^4 + x^3 + 1
18    //         5: feedback = lfsr_data[4] ^ lfsr_data[3]; // x^5 + x^4 + 1
19    //         6: feedback = lfsr_data[5] ^ lfsr_data[4]; // x^6 + x^5 + 1
20    //         7: feedback = lfsr_data[6] ^ lfsr_data[5]; // x^7 + x^6 + 1
21    //         8: feedback = lfsr_data[7] ^ lfsr_data[6]; // x^8 + x^7 + x^6 + x^5 + x^4 + 1
22    //     default: feedback = 1'b0;
23    //     endcase
24    // end
25    // LFSR behavior
26    always_ff @(posedge clk or negedge reset) begin
27        if (!reset) begin
28            lfsr_data <= 0;
29            lfsr_done <= 0;
30        end else if (load_seed) begin
31            lfsr_data <= seed_data;
32            lfsr_done <= 0;
33        end else begin
34            lfsr_data <= {lfsr_data[N-1:0], feedback}; // Shift left and insert feedback at LSB
35        end
36        // Set lfsr_done to high when a full cycle completes
37        if (lfsr_data == seed_data) begin
38            lfsr_done <= 1;
39        end else begin
40            lfsr_done <= 0;
41        end
42    end
43 end
```

Messages:

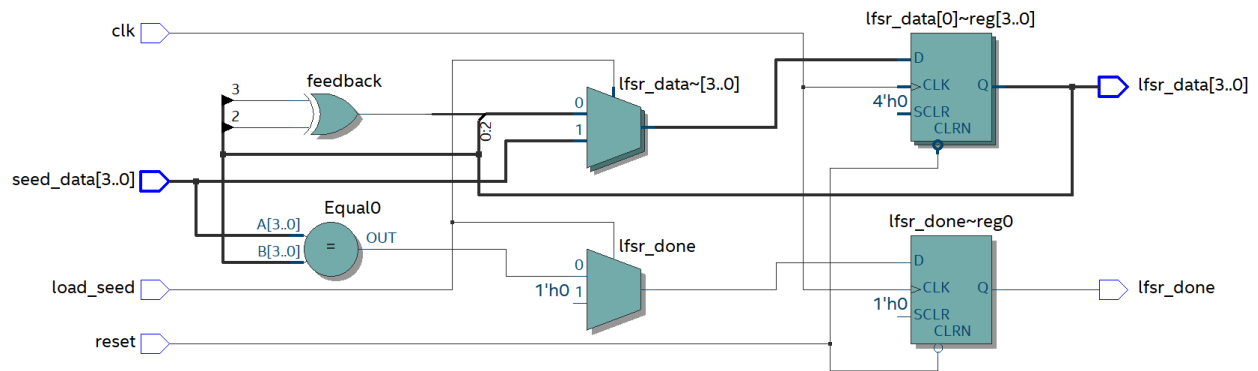
- 332123 Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
- 332146 Worst-case setup slack is 0.594
- 332146 Worst-case hold slack is 0.142
- 332140 No Recovery paths to report
- 332140 No Removal paths to report
- 332148 Timing requirements not met
- 332146 Worst-case minimum pulse width slack is -2.846
- 21076 High junction temperature operating condition is not set. Assuming a default value of '100'.
- 21076 Low junction temperature operating condition is not set. Assuming a default value of '-40'.
- 332123 Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
- *****
- Running Quartus Prime EDA Netlist Writer
- Command: quartus_eda --read_settings_files=off --write_settings_files=off lfsr -c lfsr
- 18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCESSORS to the number of processors to use.
- 204019 Generated file lfsr.svo in folder "C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ece111/HW/Homework4/lfsr/simulation"
- Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings

```
ucsd > ece111 > HW > Homework4 > lfsr > lfsr.sv
1 // RTL Model for Linear Feedback Shift Register
2 module lfsr
3 #(parameter N = 4) // Number of bits for LFSR
4 (
5     input logic clk, reset, load_seed,
6     input logic [N-1:0] seed_data,
7     output logic lfsr_done,
8     output logic [N-1:0] lfsr_data
9 );
10
11 // Internal feedback signal
12 logic feedback;
13
14 // Feedback polynomial based on N
15 always_comb begin
16     case (N)
17         2: feedback = lfsr_data[1] ^ lfsr_data[0]; // x^2 + x + 1
18         3: feedback = lfsr_data[2] ^ lfsr_data[1]; // x^3 + x^2 + 1
19         4: feedback = lfsr_data[3] ^ lfsr_data[2]; // x^4 + x^3 + 1
20         5: feedback = lfsr_data[4] ^ lfsr_data[2]; // x^5 + x^3 + 1
21         6: feedback = lfsr_data[5] ^ lfsr_data[4]; // x^6 + x^5 + 1
22         7: feedback = lfsr_data[6] ^ lfsr_data[5]; // x^7 + x^6 + 1
23         8: feedback = lfsr_data[7] ^ lfsr_data[5] ^ lfsr_data[4] ^ lfsr_data[3]; // x^8 + x^6 + x^5 + x^4 + 1
24         default: feedback = 1'b0;
25     endcase
26 end
27
28 // LFSR behavior
29 always_ff @(posedge clk or negedge reset) begin
30     if (!reset) begin
31         lfsr_data <= 0;
32         lfsr_done <= 0;
33     end else if (load_seed) begin
34         lfsr_data <= seed_data;
35         lfsr_done <= 0;
36     end else begin
37         lfsr_data <= {lfsr_data[N-2:0], feedback}; // Shift left and insert feedback at LSB
38
39         // Set lfsr_done to high when a full cycle completes
40         if (lfsr_data == seed_data) begin
41             lfsr_done <= 1;
42         end else begin
43             lfsr_done <= 0;
44         end
45     end
46 end
47
48 endmodule: lfsr
```

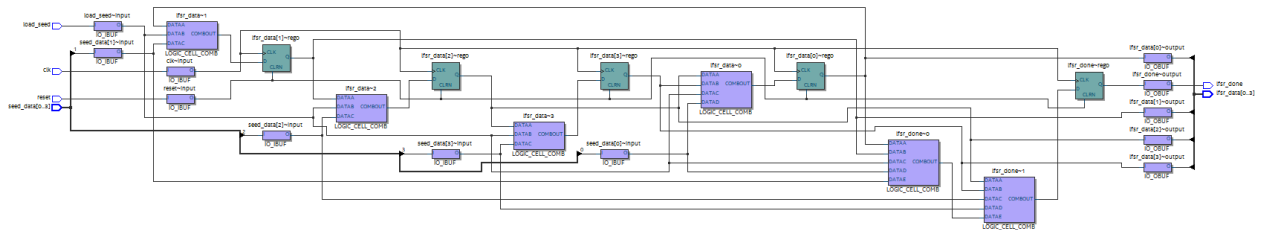
lfsr.sv Resource Usage Summary

ucsd > ece111 > HW > Homework4 > lfsr > ≡ lfsr-Resource Usage Summary.rpt		
34	+-----+-----+-----+	
35	; Analysis & Synthesis Resource Usage Summary	
36	+-----+-----+-----+	
37	; Resource	; Usage
38	+-----+-----+-----+	
39	; Estimated ALUTs Used	; 6
40	; -- Combinational ALUTs	; 6
41	; -- Memory ALUTs	; 0
42	; -- LUT_REGS	; 0
43	; Dedicated logic registers	; 5
44		
45	; Estimated ALUTs Unavailable	; 0
46	; -- Due to unpartnered combinational logic	; 0
47	; -- Due to Memory ALUTs	; 0
48		
49	; Total combinational functions	; 6
50	; Combinational ALUT usage by number of inputs	
51	; -- 7 input functions	; 0
52	; -- 6 input functions	; 0
53	; -- 5 input functions	; 2
54	; -- 4 input functions	; 1
55	; -- <=3 input functions	; 3
56		
57	; Combinational ALUTs by mode	
58	; -- normal mode	; 6
59	; -- extended LUT mode	; 0
60	; -- arithmetic mode	; 0
61	; -- shared arithmetic mode	; 0
62		
63	; Estimated ALUT/register pairs used	; 6
64		
65	; Total registers	; 5
66	; -- Dedicated logic registers	; 5
67	; -- I/O registers	; 0
68	; -- LUT_REGS	; 0
69		
70		
71	; I/O pins	; 12
72		
73	; DSP block 18-bit elements	; 0
74		
75	; Maximum fan-out node	; load_seed~input
76	; Maximum fan-out	; 5
77	; Total fan-out	; 55
78	; Average fan-out	; 1.57
79	+-----+-----+-----+	

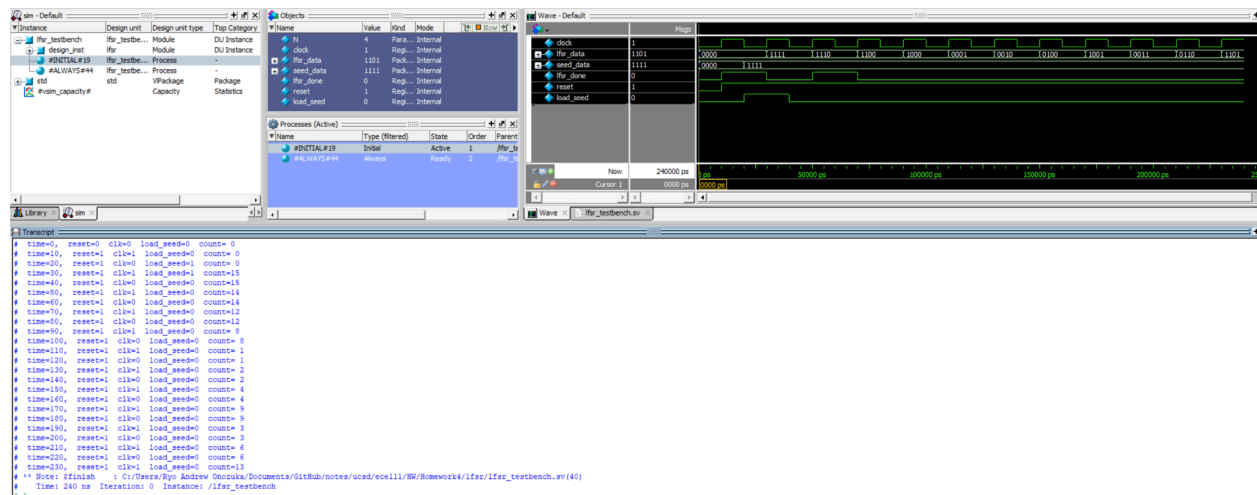
lfsr.v sv RTL viewer



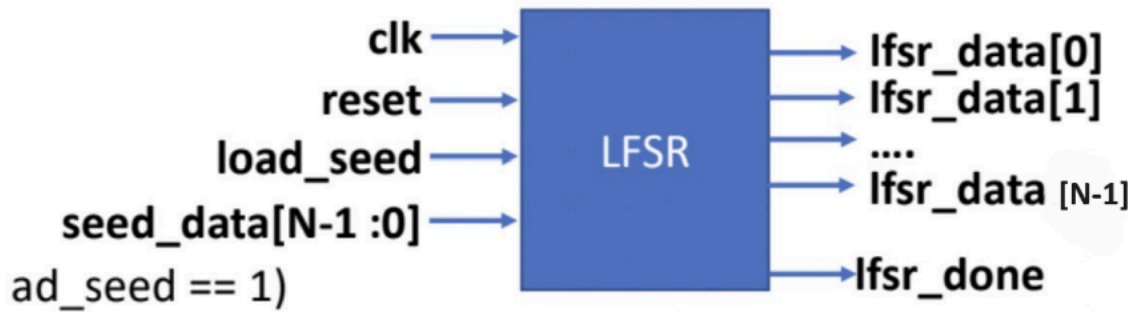
lfsr.v post mapping viewer



simulation wavelength results explanation:

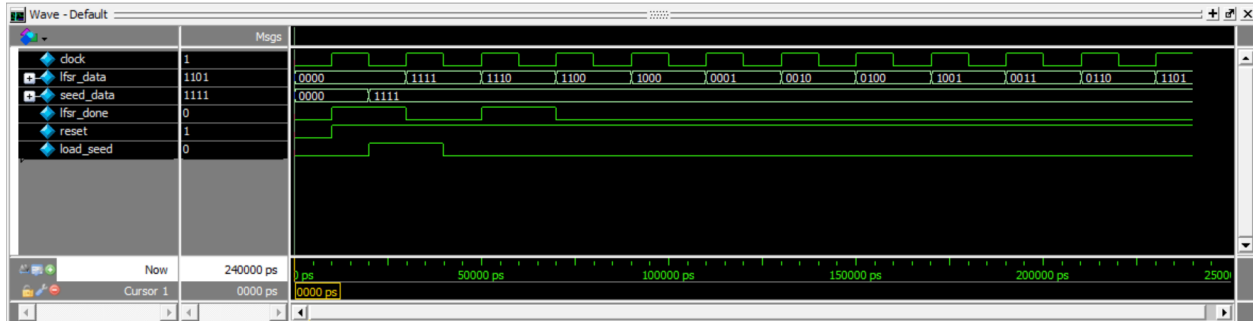


- Here is the block diagram:

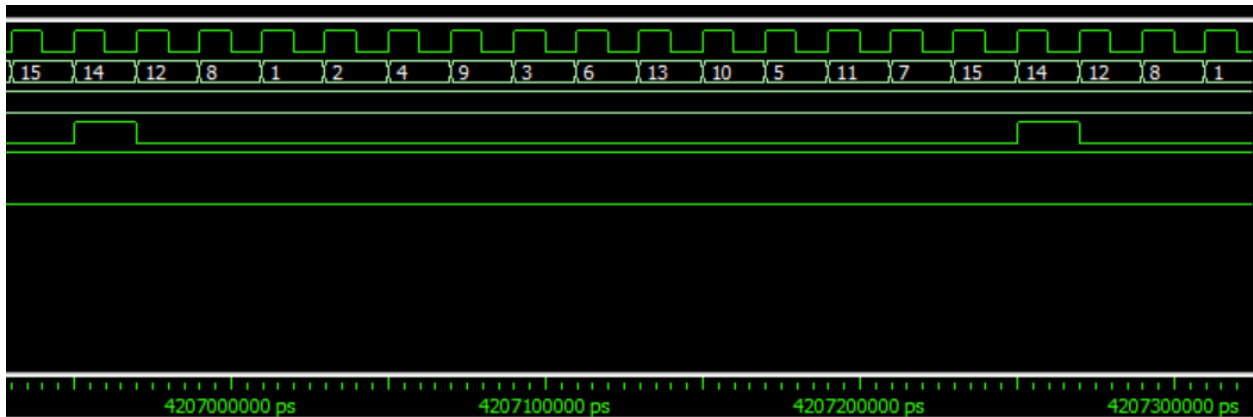


Bits(n)	Feedback Polynomial	Max Length / Period ($2^N - 1$)
2	$x^2 + x^1 + 1$	3
3	$x^3 + x^2 + 1$	7
4	$x^4 + x^3 + 1$	15
5	$x^5 + x^3 + 1$	31
6	$x^6 + x^5 + 1$	63
7	$x^7 + x^6 + 1$	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	255

The 4-bit Linear Feedback Shift Register (LFSR) uses the feedback polynomial $x^4 + x^3 + 1$, where the feedback bit is calculated by XORing bits 3 and 2 of the current lfsr_data value. This feedback bit is then shifted into the least significant bit (LSB) of the register on each clock cycle. In the simulation, we initialize the LFSR with a seed value of 4'b1111 (decimal 15) when load_seed is high, loading it into lfsr_data. After the initial load, each clock cycle shifts the bits of lfsr_data left and inserts the feedback bit at the LSB, generating a unique pseudo-random sequence.



As observed in the waveform, the sequence generated follows the maximum-length behavior, covering all possible 4-bit states (from 1 to 15) without repetition until it completes a full cycle. The expected sequence, characteristic of the polynomial, is pseudo-random and includes values, until it eventually returns to the starting seed. This behavior demonstrates the periodic nature and maximum-length cycle of the 4-bit LFSR, validating its functionality as a pseudo-random sequence generator.



We can see here our pattern in our extended simulation (see bottom for how long the simulation was run)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
15	14	12	8	1	2	4	9	3	6	13	10	5	11	7

The simulation confirms that the LFSR design accurately shifts, applies feedback, and produces a sequence according to the specified polynomial.

Andrew Onozuka A16760043

ECE 111 HW4 | 10/30/2024

barrel_shifter.v

The screenshot displays the Quartus II IDE interface. The top pane shows the source code for `barrel_shifter.v`, which implements a barrel shifter with shift and rotate operations. The middle-left pane lists compilation tasks, including 'Compile Design', 'Analysis & Synthesis', 'Fitter (Place & Route)', 'Assembler (Generate program)', 'Timing Analysis', and 'EDA Netlist Writer'. The bottom pane shows a list of messages and warnings generated during the compilation process.

Messages and Warnings:

- 332140 No Minimum Pulse Width paths to report
- 21076 High junction temperature operating condition is not set. Assuming a default value of '100'.
- 21076 Low junction temperature operating condition is not set. Assuming a default value of '-40'.
- 332142 No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
- 332096 The command derive_clocks did not find any clocks to derive. No clocks were created or changed.
- 332068 No clocks defined in design.
- 332154 The derive_clock_uncertainty command did not apply clock uncertainty to any clock-to-clock transfers.
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 6 warnings
- Running Quartus Prime EDA Netlist Writer
- Command: quartus_eda --read_settings_files=off --write_settings_files=off barrel_shifter -c barrel_shifter
- 18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PA...
- 204019 Generated file barrel_shifter.svo in folder "C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ece111/Hw/Homework4/barr...
- Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 16 warnings

```
ucsd > ece111 > HW > Homework4 > barrel_shifter > barrel_shifter.sv
1  // Barrel Shifter RTL Model
2  `include "mux_2x1_behavioral.sv"
3  module barrel_shifter (
4      input logic select, // select=0 shift operation, select=1 rotate operation
5      input logic direction, // direction=0 right move, direction=1 left move
6      input logic[1:0] shift_value, // number of bits to be shifted (0, 1, 2 or 3)
7      input logic[3:0] din,
8      output logic[3:0] dout
9  );
10
11  logic [3:0] temp_shift, temp_rotate;
12
13  // Shift Logic
14  always_comb begin
15      if (direction) begin // Left shift
16          case (shift_value)
17              2'b00: temp_shift = din;
18              2'b01: temp_shift = {din[2:0], 1'b0};
19              2'b10: temp_shift = {din[1:0], 2'b00};
20              2'b11: temp_shift = {din[0], 3'b000};
21              default: temp_shift = din;
22          endcase
23      end else begin // Right shift
24          case (shift_value)
25              2'b00: temp_shift = din;
26              2'b01: temp_shift = {1'b0, din[3:1]};
27              2'b10: temp_shift = {2'b00, din[3:2]};
28              2'b11: temp_shift = {3'b000, din[3]};
29              default: temp_shift = din;
30          endcase
31      end
32  end
33
34  // Rotate Logic
35  always_comb begin
36      if (direction) begin // Left rotate
37          case (shift_value)
38              2'b00: temp_rotate = din;
39              2'b01: temp_rotate = {din[2:0], din[3]};
40              2'b10: temp_rotate = {din[1:0], din[3:2]};
41              2'b11: temp_rotate = {din[0], din[3:1]};
42              default: temp_rotate = din;
43          endcase
44      end else begin // Right rotate
45          case (shift_value)
46              2'b00: temp_rotate = din;
47              2'b01: temp_rotate = {din[0], din[3:1]};
48              2'b10: temp_rotate = {din[1:0], din[3:2]};
49              2'b11: temp_rotate = {din[2:0], din[3]};
50              default: temp_rotate = din;
51          endcase
52      end
53  end
54
55  // Output Mux: Select between Shift and Rotate
56  mux_2x1 mux0 (.sel(select), .in0(temp_shift[0]), .in1(temp_rotate[0]), .out(dout[0]));
57  mux_2x1 mux1 (.sel(select), .in0(temp_shift[1]), .in1(temp_rotate[1]), .out(dout[1]));
58  mux_2x1 mux2 (.sel(select), .in0(temp_shift[2]), .in1(temp_rotate[2]), .out(dout[2]));
59  mux_2x1 mux3 (.sel(select), .in0(temp_shift[3]), .in1(temp_rotate[3]), .out(dout[3]));
60
61  endmodule: barrel_shifter
```

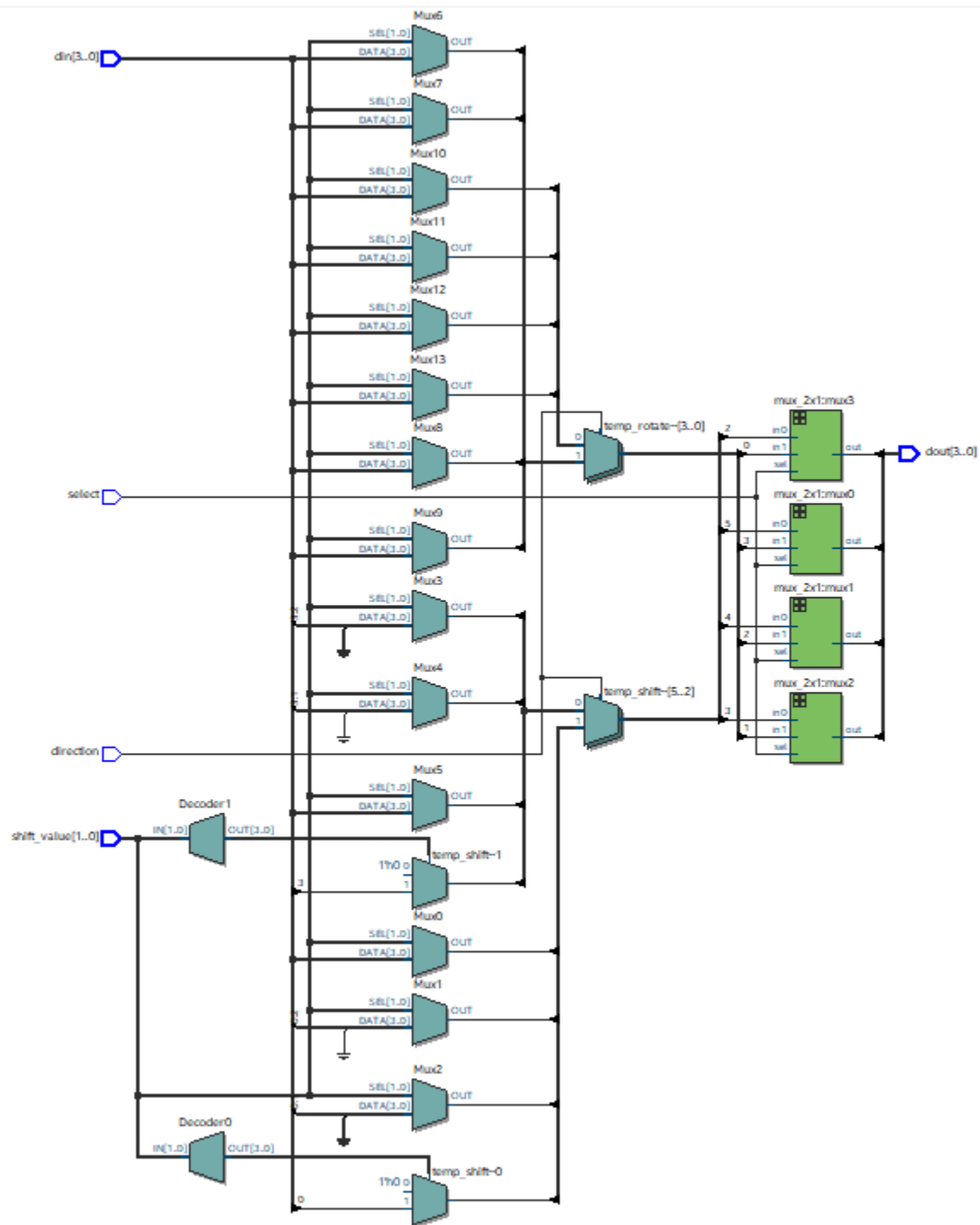

barrel_shifter.sv Resource Usage Summary

```

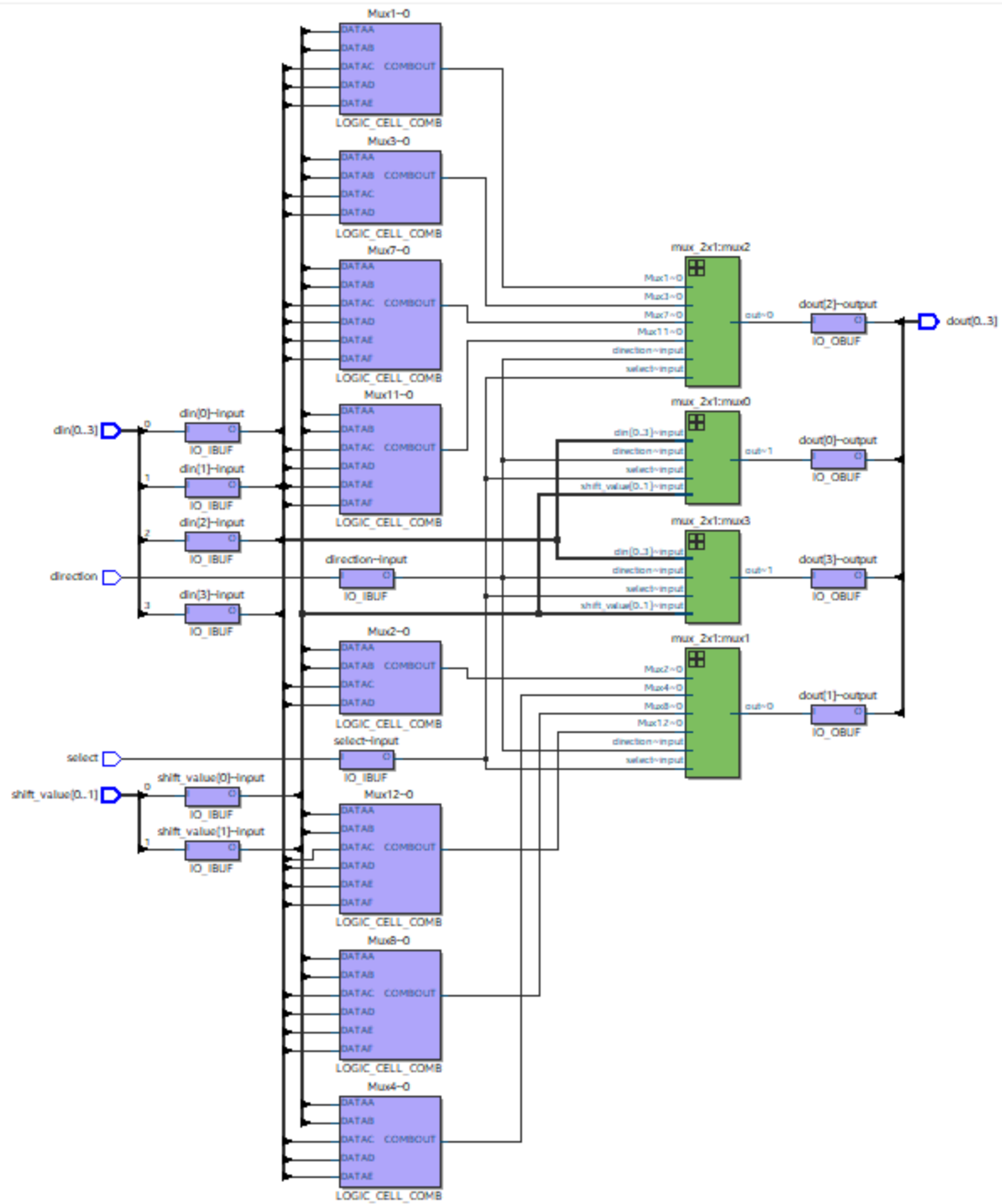
ucsd > ece111 > HW > Homework4 > barrel_shifter > barrel_shifter-Resource Usage Summary.rpt
34 +-----+
35 ; Analysis & Synthesis Resource Usage Summary ;
36 +-----+
37 ; Resource ; Usage ;
38 +-----+
39 ; Estimated ALUTs Used ; 14 ;
40 ; -- Combinational ALUTs ; 14 ;
41 ; -- Memory ALUTs ; 0 ;
42 ; -- LUT_REGS ; 0 ;
43 ; Dedicated logic registers ; 0 ;
44 ; ; ;
45 ; Estimated ALUTs Unavailable ; 8 ;
46 ; -- Due to unpartnered combinational logic ; 8 ;
47 ; -- Due to Memory ALUTs ; 0 ;
48 ; ; ;
49 ; Total combinational functions ; 14 ;
50 ; Combinational ALUT usage by number of inputs ; ;
51 ; -- 7 input functions ; 2 ;
52 ; -- 6 input functions ; 6 ;
53 ; -- 5 input functions ; 4 ;
54 ; -- 4 input functions ; 2 ;
55 ; -- <=3 input functions ; 0 ;
56 ; ; ;
57 ; Combinational ALUTs by mode ; ;
58 ; -- normal mode ; 12 ;
59 ; -- extended LUT mode ; 2 ;
60 ; -- arithmetic mode ; 0 ;
61 ; -- shared arithmetic mode ; 0 ;
62 ; ; ;
63 ; Estimated ALUT/register pairs used ; 22 ;
64 ; ; ;
65 ; Total registers ; 0 ;
66 ; -- Dedicated logic registers ; 0 ;
67 ; -- I/O registers ; 0 ;
68 ; -- LUT_REGS ; 0 ;
69 ; ; ;
70 ; ; ;
71 ; I/O pins ; 12 ;
72 ; ; ;
73 ; DSP block 18-bit elements ; 0 ;
74 ; ; ;
75 ; Maximum fan-out node ; shift_value[1]~input ;
76 ; Maximum fan-out ; 12 ;
77 ; Total fan-out ; 94 ;
78 ; Average fan-out ; 2.47 ;
79 +-----+
80

```

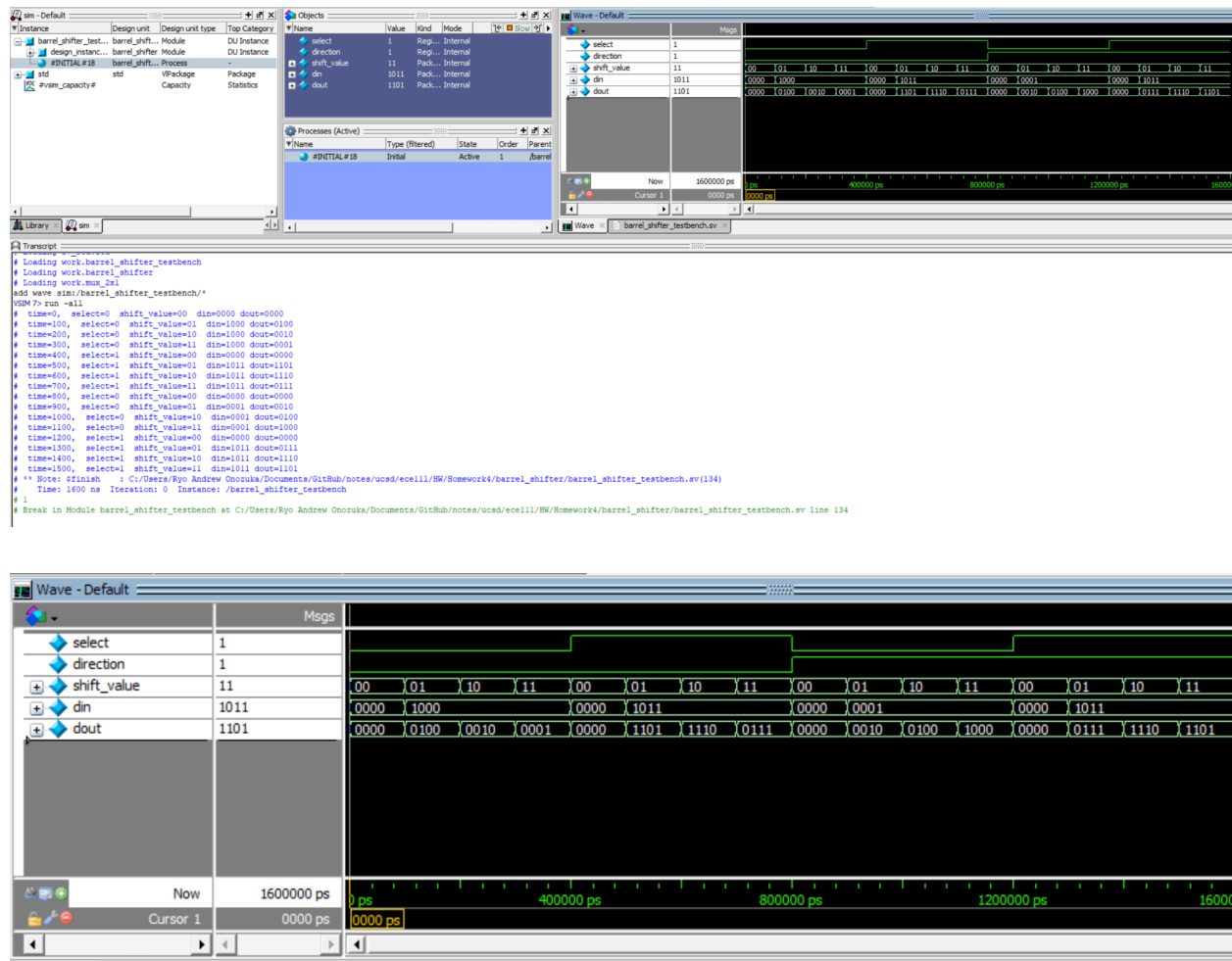
barrel_shifter.sv RTL viewer



barrel_shifter.v sv post mapping viewer



simulation wavelength results explanation:



The simulation output aligns with the expected behavior for both shift and rotate operations in the barrel shifter. For shift operations (select = 0), right shifts with din = 1000 produce correct results for each shift_value: shifting by 1, 2, and 3 yields 0100, 0010, and 0001, respectively. Similarly, left shifts with din = 0001 and varying shift_value correctly result in 0010, 0100, and 1000. For rotate operations (select = 1), left rotations with din = 1011 produce accurate outputs: rotating by 1, 2, and 3 yields 1101, 1110, and 0111, as expected. This confirms that the barrel shifter module is functioning correctly for both shifting and rotating, following the design requirements.

Andrew Onozuka A16760043

ECE 111 HW4 | 10/30/2024

gray_to_binary_convertor.v

The screenshot shows the Quartus Prime IDE interface. The top pane displays the Verilog code for the `gray_code_to_binary_convertor` module. The code defines a module with parameters `N` and `4`, inputs `clk`, `rstn`, and `gray_value`, and output `binary_value`. It uses an `always_ff` block to convert Gray code to binary by XORing adjacent bits. The bottom pane shows the compilation report with various messages, including timing warnings and successful compilation status.

```
1 module gray_code_to_binary_convertor #(parameter N = 4)(
2     input logic clk, rstn,
3     input logic[N-1:0] gray_value,
4     output logic[N-1:0] binary_value);
5
6     logic [N-1:0] temp_binary;
7
8     always_ff @(posedge clk or negedge rstn) begin
9         if (!rstn) begin
10             temp_binary <= 0;
11         end else begin
12             // Convert Gray code to binary
13             temp_binary[N-1] = gray_value[N-1]; // MSB remains the same
14             for (int i = N-2; i >= 0; i--) begin
15                 temp_binary[i] = temp_binary[i+1] ^ gray_value[i];
16             end
17         end
18     end
19
20     // Register the output
21     assign binary_value = temp_binary;
22 endmodule: gray_code_to_binary_convertor
```

Compilation Report - gray_code_to_binary_convertor

Task: Compile Design

Messages:

- 332140 No Recovery paths to report
- 332140 No Removal paths to report
- 332148 Timing requirements not met
- 332146 Worst-case minimum pulse width slack is -2.846
- 21076 High junction temperature operating condition is not set. Assuming a default value of '100'.
- 21076 Low junction temperature operating condition is not set. Assuming a default value of '-40'.
- 332154 The derive_clock_uncertainty command did not apply clock uncertainty to any clock-to-clock transfers.
- 332102 Design is not fully constrained for setup requirements
- 332102 Design is not fully constrained for hold requirements
- Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
- Running Quartus Prime EDA Netlist Writer
- Command: quartus_eda --read_settings_files=off --write_settings_files=off gray_code_to_binary_convertor -c gray_code_to_bin
- 18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PA
- 204019 Generated file gray_code_to_binary_convertor.svo in folder "C:/Users/Ryo Andrew Onozuka/Documents/Github/notes/ucsd/ece111/HW
- Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
- 293000 Quartus Prime Full Compilation was successful. 0 errors, 14 warnings

```
ucsd > ece111 > HW > Homework4 > gray_code_to_binary_convertor > gray_code_to_binary_convertor.v
1 module gray_code_to_binary_convertor #(parameter N = 4)(
2     input logic clk, rstn,
3     input logic[N-1:0] gray_value,
4     output logic[N-1:0] binary_value);
5
6     logic [N-1:0] temp_binary;
7
8     always_ff @(posedge clk or negedge rstn) begin
9         if (!rstn) begin
10             temp_binary <= 0;
11         end else begin
12             // Convert Gray code to binary
13             temp_binary[N-1] = gray_value[N-1]; // MSB remains the same
14             for (int i = N-2; i >= 0; i--) begin
15                 temp_binary[i] = temp_binary[i+1] ^ gray_value[i];
16             end
17         end
18     end
19
20     // Register the output
21     assign binary_value = temp_binary;
22
23 endmodule: gray_code_to_binary_convertor
24
```

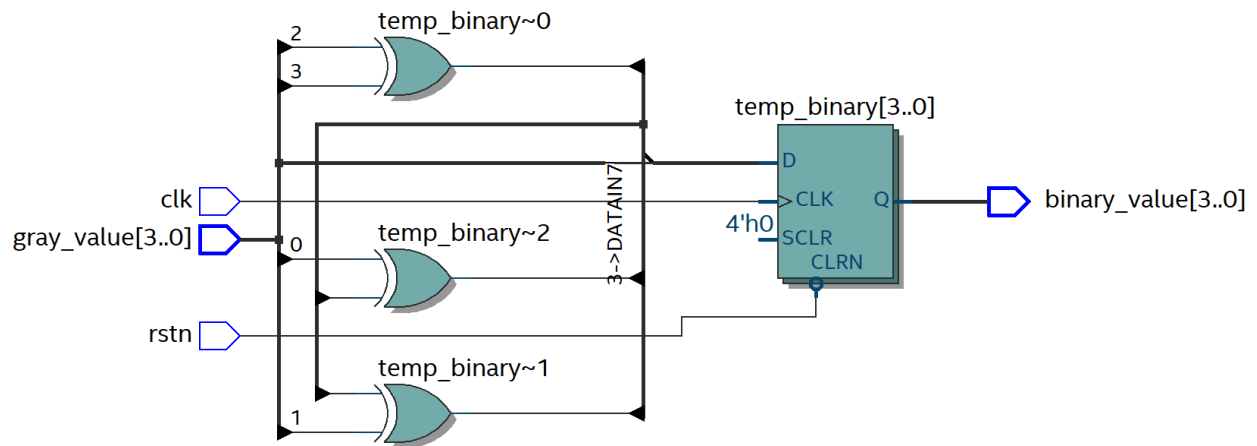
gray_to_binary_code_convertor.sv Resource Usage Summary

```

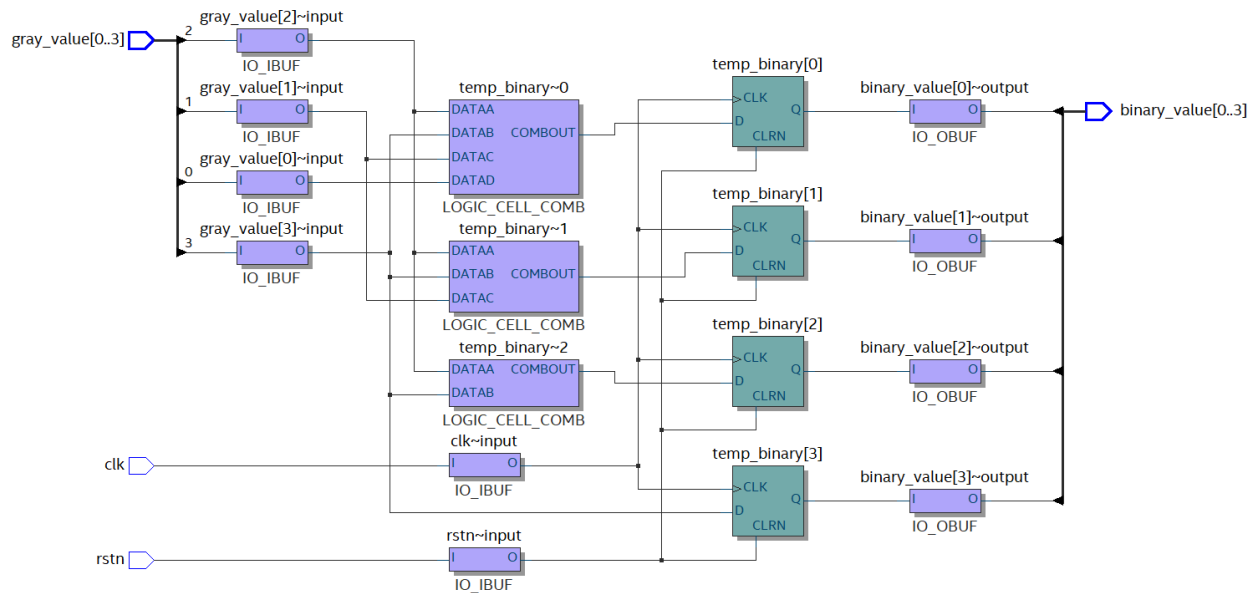
ucsd > ece111 > HW > Homework4 > gray_code_to_binary_convertor > gray_code_to_binary_convertor-Resource Usage Summary.rpt
34  +-----+
35  ; Analysis & Synthesis Resource Usage Summary ;
36  +-----+
37  ; Resource ; Usage ;
38  +-----+
39  ; Estimated ALUTs Used ; 3 ;
40  ; -- Combinational ALUTs ; 3 ;
41  ; -- Memory ALUTs ; 0 ;
42  ; -- LUT_REGS ; 0 ;
43  ; Dedicated logic registers ; 4 ;
44  ; ; ;
45  ; Estimated ALUTs Unavailable ; 0 ;
46  ; -- Due to unpartnered combinational logic ; 0 ;
47  ; -- Due to Memory ALUTs ; 0 ;
48  ; ; ;
49  ; Total combinational functions ; 3 ;
50  ; Combinational ALUT usage by number of inputs ; ;
51  ; -- 7 input functions ; 0 ;
52  ; -- 6 input functions ; 0 ;
53  ; -- 5 input functions ; 0 ;
54  ; -- 4 input functions ; 1 ;
55  ; -- <=3 input functions ; 2 ;
56  ; ; ;
57  ; Combinational ALUTs by mode ; ;
58  ; -- normal mode ; 3 ;
59  ; -- extended LUT mode ; 0 ;
60  ; -- arithmetic mode ; 0 ;
61  ; -- shared arithmetic mode ; 0 ;
62  ; ; ;
63  ; Estimated ALUT/register pairs used ; 4 ;
64  ; ; ;
65  ; Total registers ; 4 ;
66  ; -- Dedicated logic registers ; 4 ;
67  ; -- I/O registers ; 0 ;
68  ; -- LUT_REGS ; 0 ;
69  ; ; ;
70  ; ; ;
71  ; I/O pins ; 10 ;
72  ; ; ;
73  ; DSP block 18-bit elements ; 0 ;
74  ; ; ;
75  ; Maximum fan-out node ; gray_value[3]~input ;
76  ; Maximum fan-out ; 4 ;
77  ; Total fan-out ; 35 ;
78  ; Average fan-out ; 1.30 ;
79  +-----+
80

```

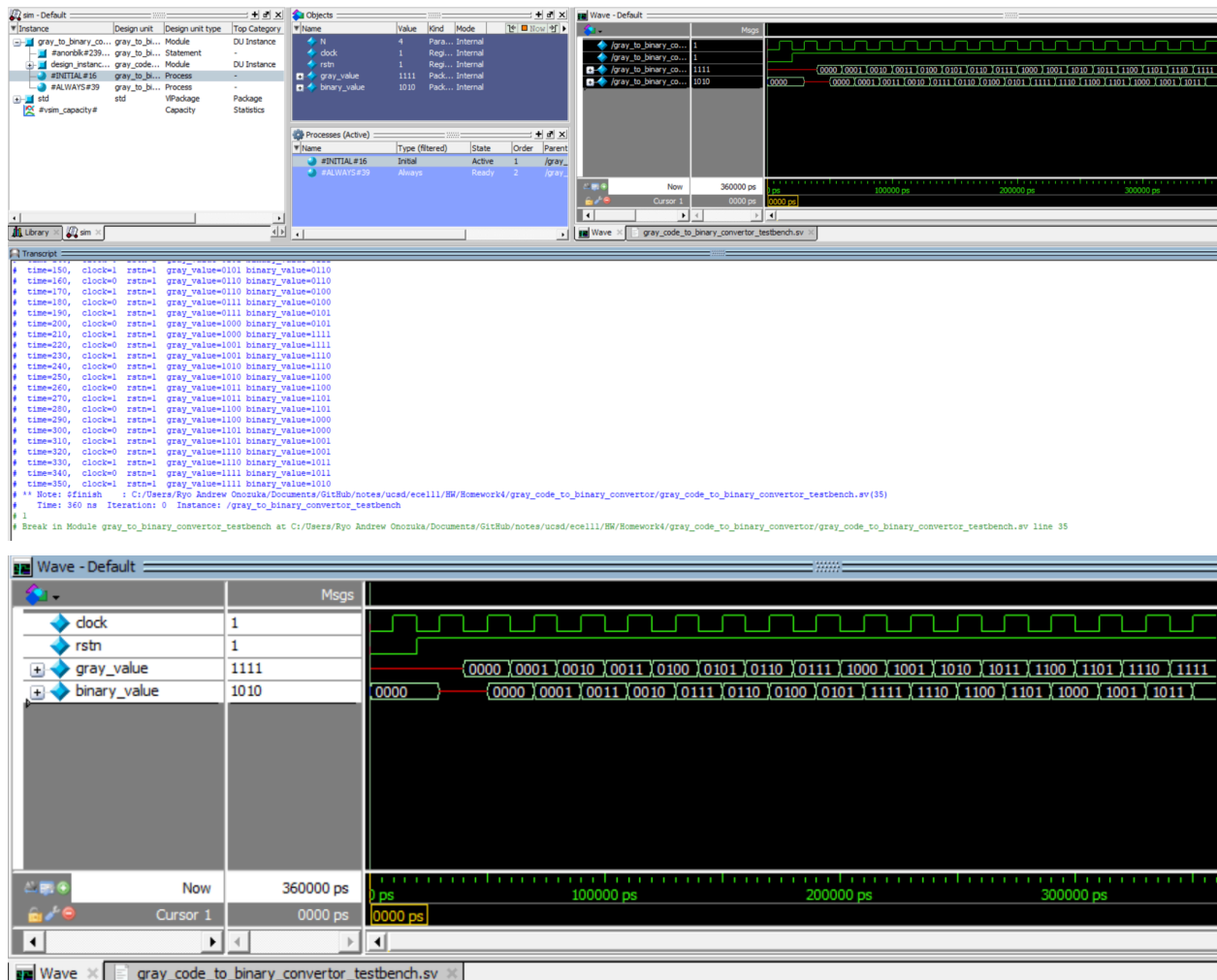
gray_to_binary_code_convertor.sv RTL viewer



gray_to_binary_code_convertor.sv post mapping viewer



simulation wavelength results explanation:



The simulation output confirms the correct operation of the Gray to Binary Code Converter. Initially, with `rstn=0`, the `binary_value` is reset to 0000. Once `rstn` is set to 1, the converter begins translating `gray_value` inputs to their binary equivalents on each clock cycle. For example, when `gray_value=0000`, the output `binary_value=0000` is correct, as both Gray and binary are the same for zero. As the `gray_value` increments, each corresponding `binary_value` accurately reflects the expected binary conversion, confirming the functionality of the XOR-based conversion logic. Specific cases, such as `gray_value=1000` producing `binary_value=1111` and `gray_value=1111` yielding `binary_value=1010`, demonstrate that the module produces the correct binary equivalents for various Gray code inputs, indicating that the design is working as intended.