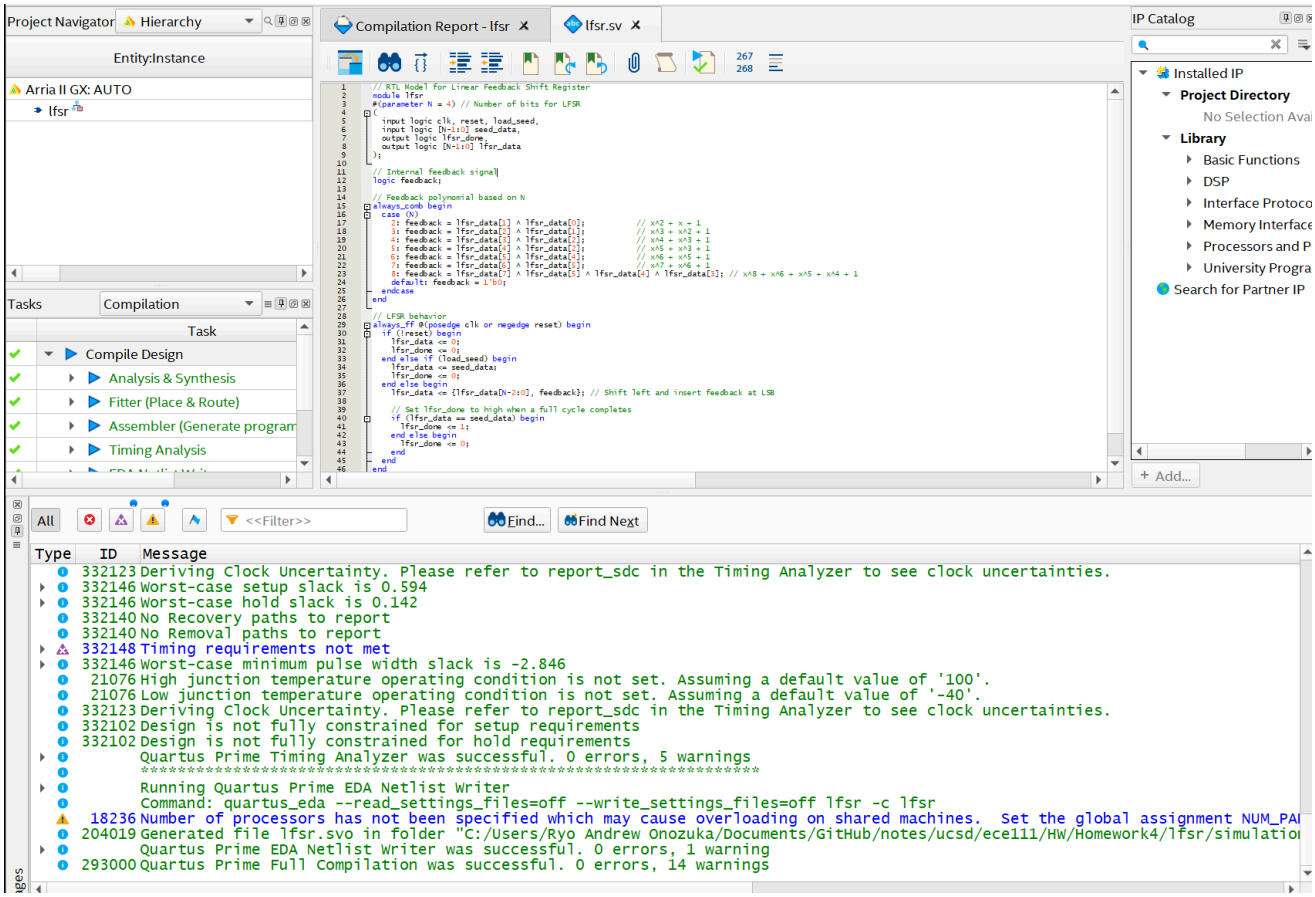
Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

lfsr.sv



Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

ucsd > ece111 > HW > Homework4 > lfsr > lfsr.sv

```
// RTL Model for Linear Feedback Shift Register
module lfsr
#(parameter N = 4) // Number of bits for LFSR
(
  input logic clk, reset, load_seed,
  input logic [N-1:0] seed_data,
  output logic lfsr_done,
  output logic [N-1:0] lfsr_data
);

// Internal feedback signal
logic feedback;

// Feedback polynomial based on N
always_comb begin
  case (N)
    2: feedback = lfsr_data[1] ^ lfsr_data[0];           // x^2 + x + 1
    3: feedback = lfsr_data[2] ^ lfsr_data[1];           // x^3 + x^2 + 1
    4: feedback = lfsr_data[3] ^ lfsr_data[2];           // x^4 + x^3 + 1
    5: feedback = lfsr_data[4] ^ lfsr_data[2];           // x^5 + x^3 + 1
    6: feedback = lfsr_data[5] ^ lfsr_data[4];           // x^6 + x^5 + 1
    7: feedback = lfsr_data[6] ^ lfsr_data[5];           // x^7 + x^6 + 1
    8: feedback = lfsr_data[7] ^ lfsr_data[5] ^ lfsr_data[4] ^ lfsr_data[3]; // x^8 + x^6 + x^5 + x^4 + 1
    default: feedback = 1'b0;
  endcase
end

// LFSR behavior
always_ff @(posedge clk or negedge reset) begin
  if (!reset) begin
    lfsr_data <= 0;
    lfsr_done <= 0;
  end else if (load_seed) begin
    lfsr_data <= seed_data;
    lfsr_done <= 0;
  end else begin
    lfsr_data <= {lfsr_data[N-2:0], feedback}; // Shift left and insert feedback at LSB

    // Set lfsr_done to high when a full cycle completes
    if (lfsr_data == seed_data) begin
      lfsr_done <= 1;
    end else begin
      lfsr_done <= 0;
    end
  end
end

endmodule: lfsr
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

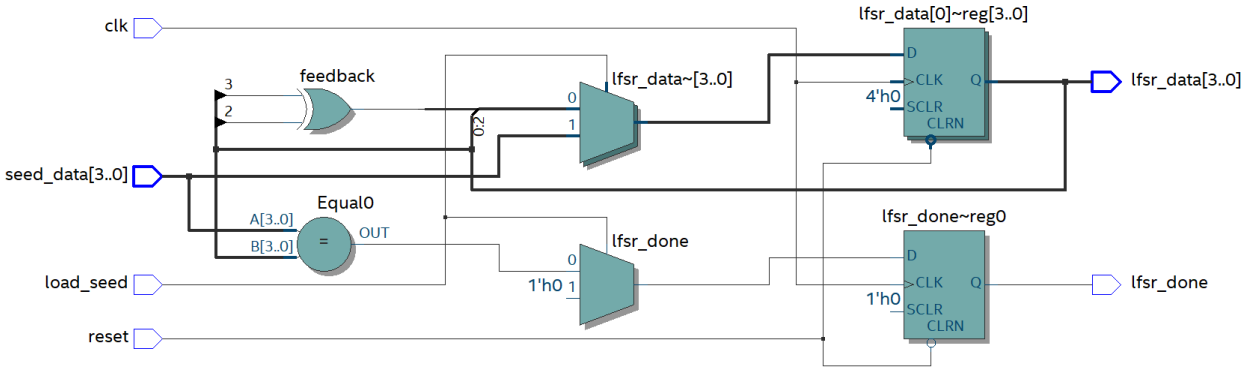
lfsr.sv Resource Usage Summary

ucsd > ece111 > HW > Homework4 > lfsr > lfsr-Resource Usage Summary.rpt

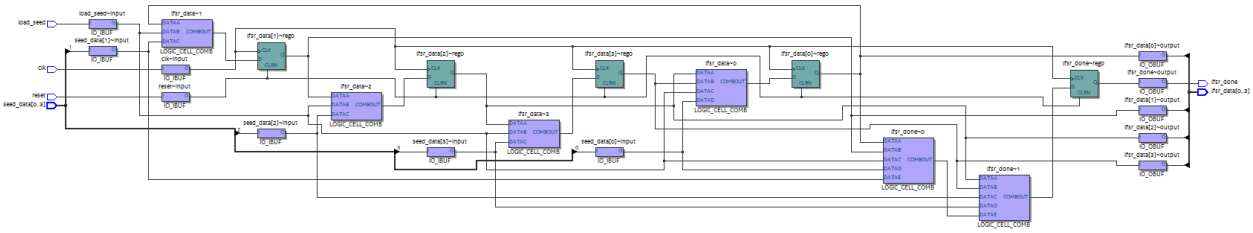
```
+-----------------------------------------------------------------------+
; Analysis & Synthesis Resource Usage Summary                           ;
+-----------------------------------------------+-----------------------+
; Resource                                      ; Usage                 ;
+-----------------------------------------------+-----------------------+
; Estimated ALUTs Used                          ; 6                     ;
;     -- Combinational ALUTs                    ; 6                     ;
;     -- Memory ALUTs                          ; 0                     ;
;     -- LUT_REGs                              ; 0                     ;
; Dedicated logic registers                     ; 5                     ;
;                                               ;                       ;
; Estimated ALUTs Unavailable                   ; 0                     ;
;     -- Due to unpartnered combinational logic ; 0                     ;
;     -- Due to Memory ALUTs                    ; 0                     ;
;                                               ;                       ;
; Total combinational functions                 ; 6                     ;
; Combinational ALUT usage by number of inputs  ;                       ;
;     -- 7 input functions                      ; 0                     ;
;     -- 6 input functions                      ; 0                     ;
;     -- 5 input functions                      ; 2                     ;
;     -- 4 input functions                      ; 1                     ;
;     -- <=3 input functions                    ; 3                     ;
;                                               ;                       ;
; Combinational ALUTs by mode                   ;                       ;
;     -- normal mode                           ; 6                     ;
;     -- extended LUT mode                     ; 0                     ;
;     -- arithmetic mode                       ; 0                     ;
;     -- shared arithmetic mode                ; 0                     ;
;                                               ;                       ;
; Estimated ALUT/register pairs used            ; 6                     ;
;                                               ;                       ;
; Total registers                               ; 5                     ;
;     -- Dedicated logic registers             ; 5                     ;
;     -- I/O registers                         ; 0                     ;
;     -- LUT_REGs                              ; 0                     ;
;                                               ;                       ;
;                                               ;                       ;
; I/O pins                                      ; 12                    ;
;                                               ;                       ;
; DSP block 18-bit elements                     ; 0                     ;
;                                               ;                       ;
; Maximum fan-out node                          ; load_seed~input       ;
; Maximum fan-out                               ; 5                     ;
; Total fan-out                                 ; 55                    ;
; Average fan-out                               ; 1.57                  ;
+-----------------------------------------------+-----------------------+
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

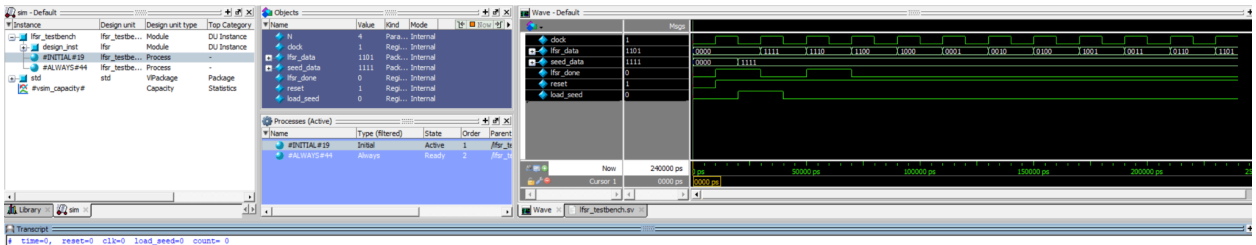
lfsr.sv RTL viewer



lfsr.sv post mapping viewer



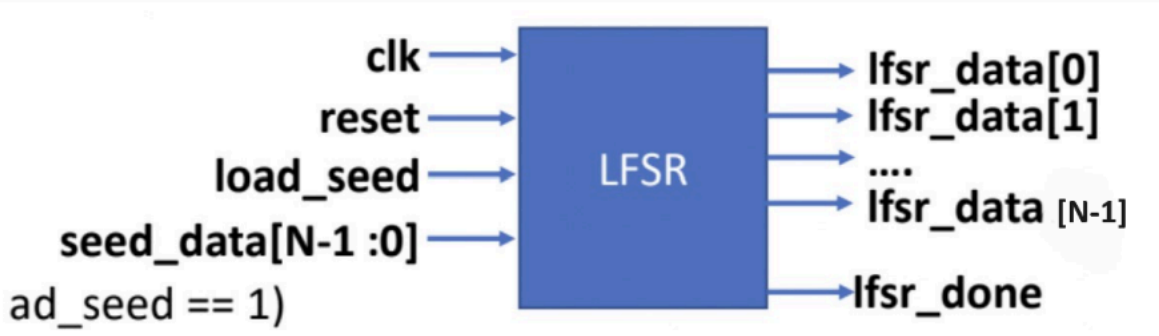
simulation wavelength results explanation:



```
# time=0,   reset=0  clk=0  load_seed=0  count= 0
# time=10,  reset=1  clk=1  load_seed=0  count= 0
# time=20,  reset=1  clk=0  load_seed=1  count= 0
# time=30,  reset=1  clk=1  load_seed=1  count=15
# time=40,  reset=1  clk=0  load_seed=0  count=15
# time=50,  reset=1  clk=1  load_seed=0  count=14
# time=60,  reset=1  clk=0  load_seed=0  count=14
# time=70,  reset=1  clk=1  load_seed=0  count=12
# time=80,  reset=1  clk=0  load_seed=0  count=12
# time=90,  reset=1  clk=1  load_seed=0  count= 8
# time=100,  reset=1  clk=0  load_seed=0  count= 8
# time=110,  reset=1  clk=1  load_seed=0  count= 1
# time=120,  reset=1  clk=0  load_seed=0  count= 1
# time=130,  reset=1  clk=1  load_seed=0  count= 2
# time=140,  reset=1  clk=0  load_seed=0  count= 2
# time=150,  reset=1  clk=1  load_seed=0  count= 4
# time=160,  reset=1  clk=0  load_seed=0  count= 4
# time=170,  reset=1  clk=1  load_seed=0  count= 9
# time=180,  reset=1  clk=0  load_seed=0  count= 9
# time=190,  reset=1  clk=1  load_seed=0  count= 3
# time=200,  reset=1  clk=0  load_seed=0  count= 3
# time=210,  reset=1  clk=1  load_seed=0  count= 6
# time=220,  reset=1  clk=0  load_seed=0  count= 6
# time=230,  reset=1  clk=1  load_seed=0  count=13
# ** Note: $finish    : C:/Users/Ryo Andrew Onozuka/Documents/GitHub/notes/ucsd/ece111/HW/Homework4/lfsr/lfsr_testbench.sv(40)
#    Time: 240 ns  Iteration: 0  Instance: /lfsr_testbench
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

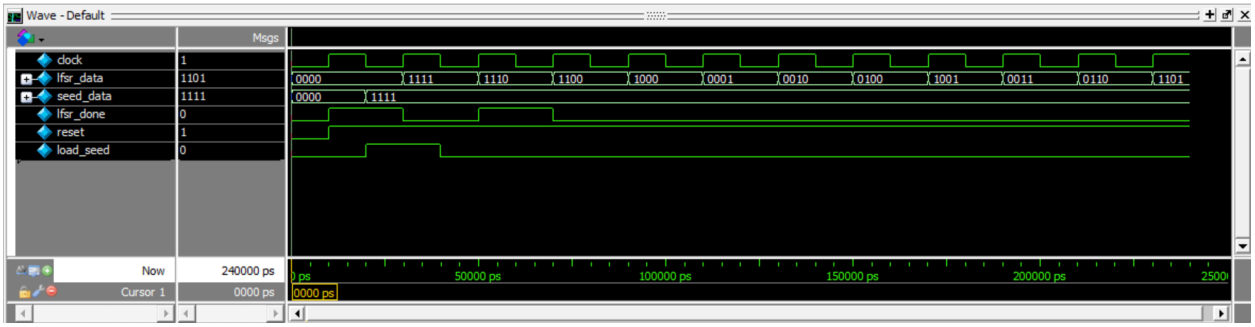
- **Here is the block diagram:**



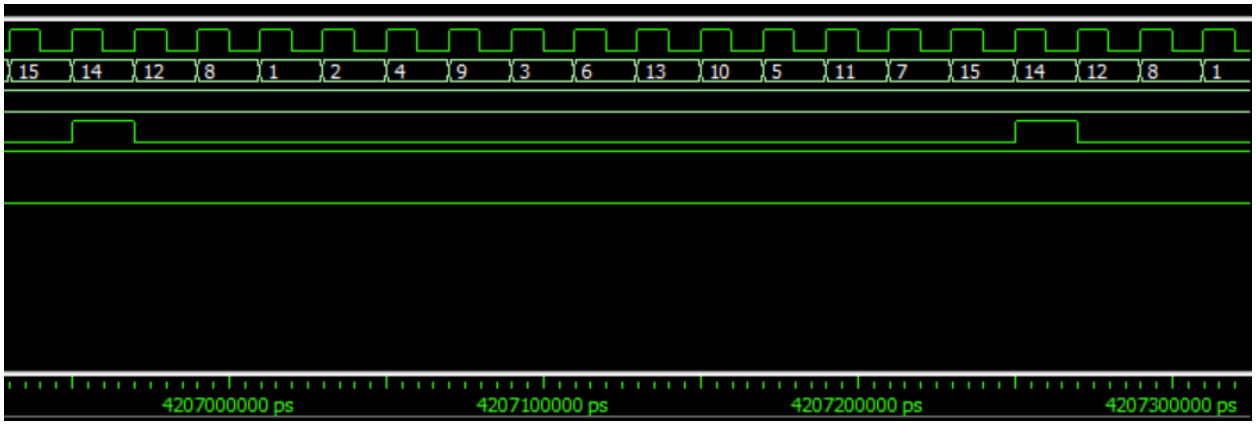
Bits(n)	Feedback Polynomial	Max Length / Period ($2^N - 1$)
2	$x^2 + x^1 + 1$	3
3	$x^3 + x^2 + 1$	7
4	$x^4 + x^3 + 1$	15
5	$x^5 + x^3 + 1$	31
6	$x^6 + x^5 + 1$	63
7	$x^7 + x^6 + 1$	127
8	$x^8 + x^6 + x^5 + x^4 + 1$	255

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.

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024



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.



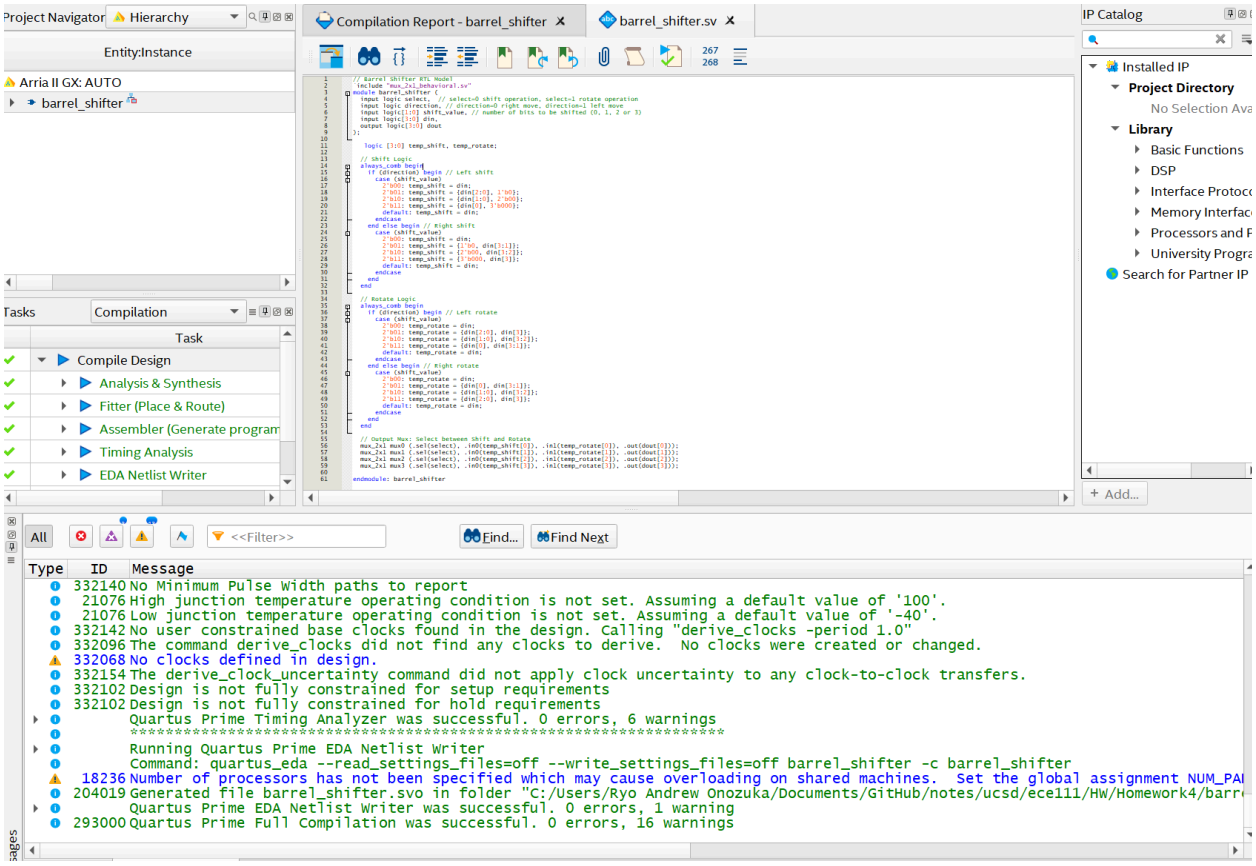
We can see here our pattern in our extended simulation (see bottom for how long the simulation was run)

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.sv



Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

ucsd > ece111 > HW > Homework4 > barrel_shifter > barrel_shifter.sv

```
// Barrel Shifter RTL Model
`include "mux_2x1_behavioral.sv"
module barrel_shifter (
  input logic select,  // select=0 shift operation, select=1 rotate operation
  input logic direction, // direction=0 right move, direction=1 left move
  input logic[1:0] shift_value, // number of bits to be shifted (0, 1, 2 or 3)
  input logic[3:0] din,
  output logic[3:0] dout
);

  logic [3:0] temp_shift, temp_rotate;

  // Shift Logic
  always_comb begin
    if (direction) begin // Left shift
      case (shift_value)
        2'b00: temp_shift = din;
        2'b01: temp_shift = {din[2:0], 1'b0};
        2'b10: temp_shift = {din[1:0], 2'b00};
        2'b11: temp_shift = {din[0], 3'b000};
        default: temp_shift = din;
      endcase
    end else begin // Right shift
      case (shift_value)
        2'b00: temp_shift = din;
        2'b01: temp_shift = {1'b0, din[3:1]};
        2'b10: temp_shift = {2'b00, din[3:2]};
        2'b11: temp_shift = {3'b000, din[3]};
        default: temp_shift = din;
      endcase
    end
  end

  // Rotate Logic
  always_comb begin
    if (direction) begin // Left rotate
      case (shift_value)
        2'b00: temp_rotate = din;
        2'b01: temp_rotate = {din[2:0], din[3]};
        2'b10: temp_rotate = {din[1:0], din[3:2]};
        2'b11: temp_rotate = {din[0], din[3:1]};
        default: temp_rotate = din;
      endcase
    end else begin // Right rotate
      case (shift_value)
        2'b00: temp_rotate = din;
        2'b01: temp_rotate = {din[0], din[3:1]};
        2'b10: temp_rotate = {din[1:0], din[3:2]};
        2'b11: temp_rotate = {din[2:0], din[3]};
        default: temp_rotate = din;
      endcase
    end
  end

  // Output Mux: Select between Shift and Rotate
  mux_2x1 mux0 (.sel(select), .in0(temp_shift[0]), .in1(temp_rotate[0]), .out(dout[0]));
  mux_2x1 mux1 (.sel(select), .in0(temp_shift[1]), .in1(temp_rotate[1]), .out(dout[1]));
  mux_2x1 mux2 (.sel(select), .in0(temp_shift[2]), .in1(temp_rotate[2]), .out(dout[2]));
  mux_2x1 mux3 (.sel(select), .in0(temp_shift[3]), .in1(temp_rotate[3]), .out(dout[3]));

endmodule: barrel_shifter
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

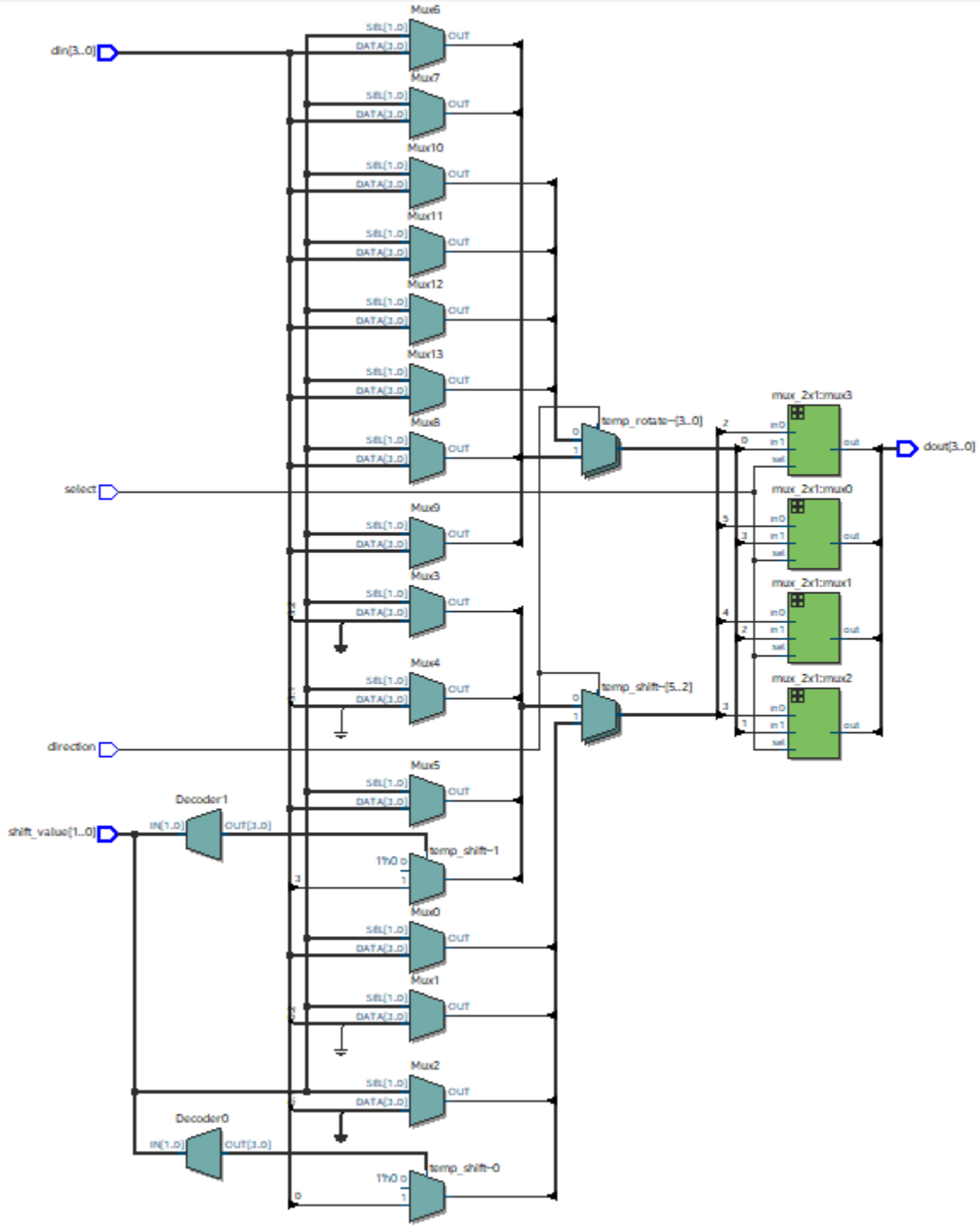
barrel_shifter.sv Resource Usage Summary

ucsd > ece111 > HW > Homework4 > barrel_shifter > barrel_shifter-Resource Usage Summary.rpt

```
+-----------------------------------------------------------------------------+
; Analysis & Synthesis Resource Usage Summary                                 ;
+-----------------------------------------------+-----------------------------+
; Resource                                      ; Usage                       ;
+-----------------------------------------------+-----------------------------+
; Estimated ALUTs Used                          ; 14                          ;
;     -- Combinational ALUTs                    ; 14                          ;
;     -- Memory ALUTs                          ; 0                           ;
;     -- LUT_REGs                              ; 0                           ;
; Dedicated logic registers                     ; 0                           ;
;                                               ;                             ;
; Estimated ALUTs Unavailable                   ; 8                           ;
;     -- Due to unpartnered combinational logic ; 8                           ;
;     -- Due to Memory ALUTs                    ; 0                           ;
;                                               ;                             ;
; Total combinational functions                 ; 14                          ;
; Combinational ALUT usage by number of inputs  ;                             ;
;     -- 7 input functions                      ; 2                           ;
;     -- 6 input functions                      ; 6                           ;
;     -- 5 input functions                      ; 4                           ;
;     -- 4 input functions                      ; 2                           ;
;     -- <=3 input functions                    ; 0                           ;
;                                               ;                             ;
; Combinational ALUTs by mode                   ;                             ;
;     -- normal mode                           ; 12                          ;
;     -- extended LUT mode                     ; 2                           ;
;     -- arithmetic mode                       ; 0                           ;
;     -- shared arithmetic mode                ; 0                           ;
;                                               ;                             ;
; Estimated ALUT/register pairs used            ; 22                          ;
;                                               ;                             ;
; Total registers                               ; 0                           ;
;     -- Dedicated logic registers             ; 0                           ;
;     -- I/O registers                         ; 0                           ;
;     -- LUT_REGs                              ; 0                           ;
;                                               ;                             ;
;                                               ;                             ;
; I/O pins                                      ; 12                          ;
;                                               ;                             ;
; DSP block 18-bit elements                     ; 0                           ;
;                                               ;                             ;
; Maximum fan-out node                          ; shift_value[1]~input        ;
; Maximum fan-out                               ; 12                          ;
; Total fan-out                                 ; 94                          ;
; Average fan-out                               ; 2.47                        ;
+-----------------------------------------------+-----------------------------+
```

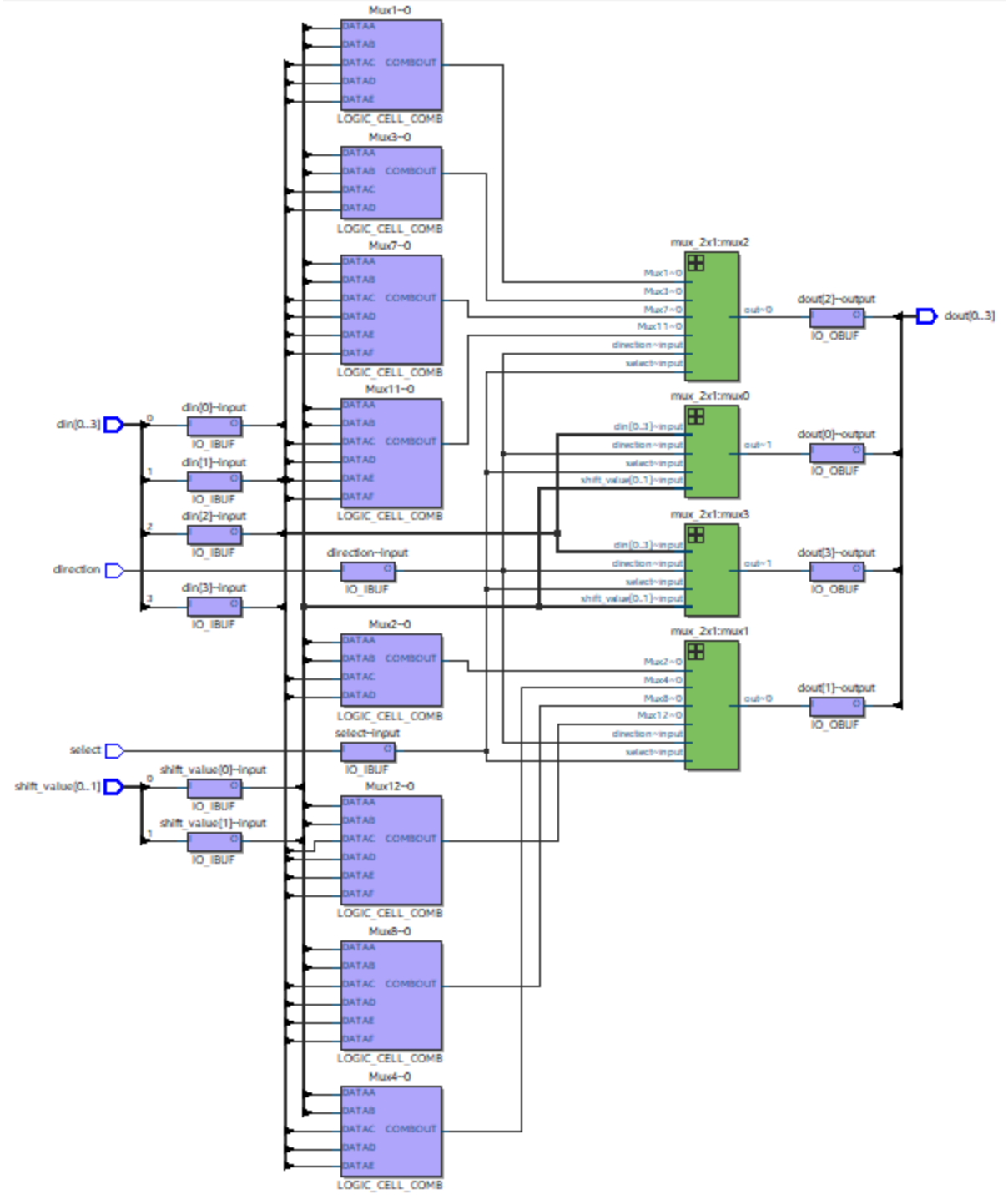
Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

barrel_shifter.sv RTL viewer



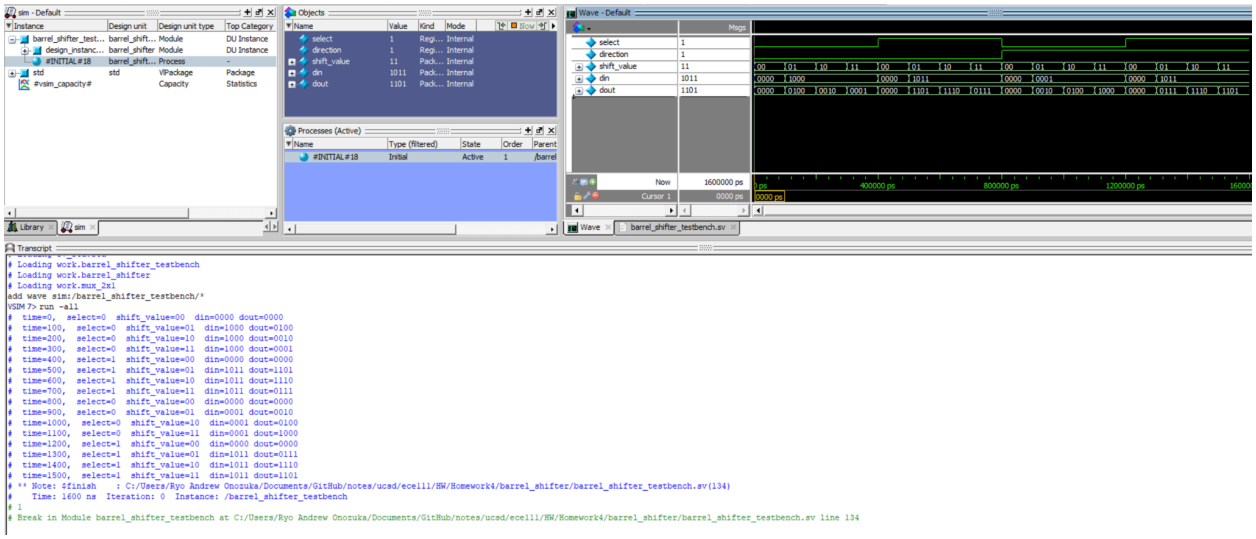
Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

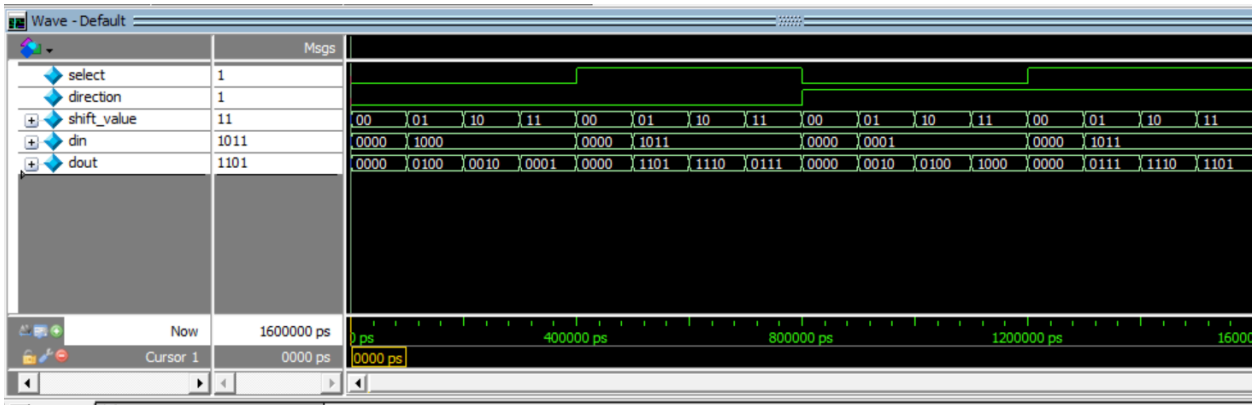
barrel_shifter.sv post mapping viewer



Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

simulation wavelength results explanation:

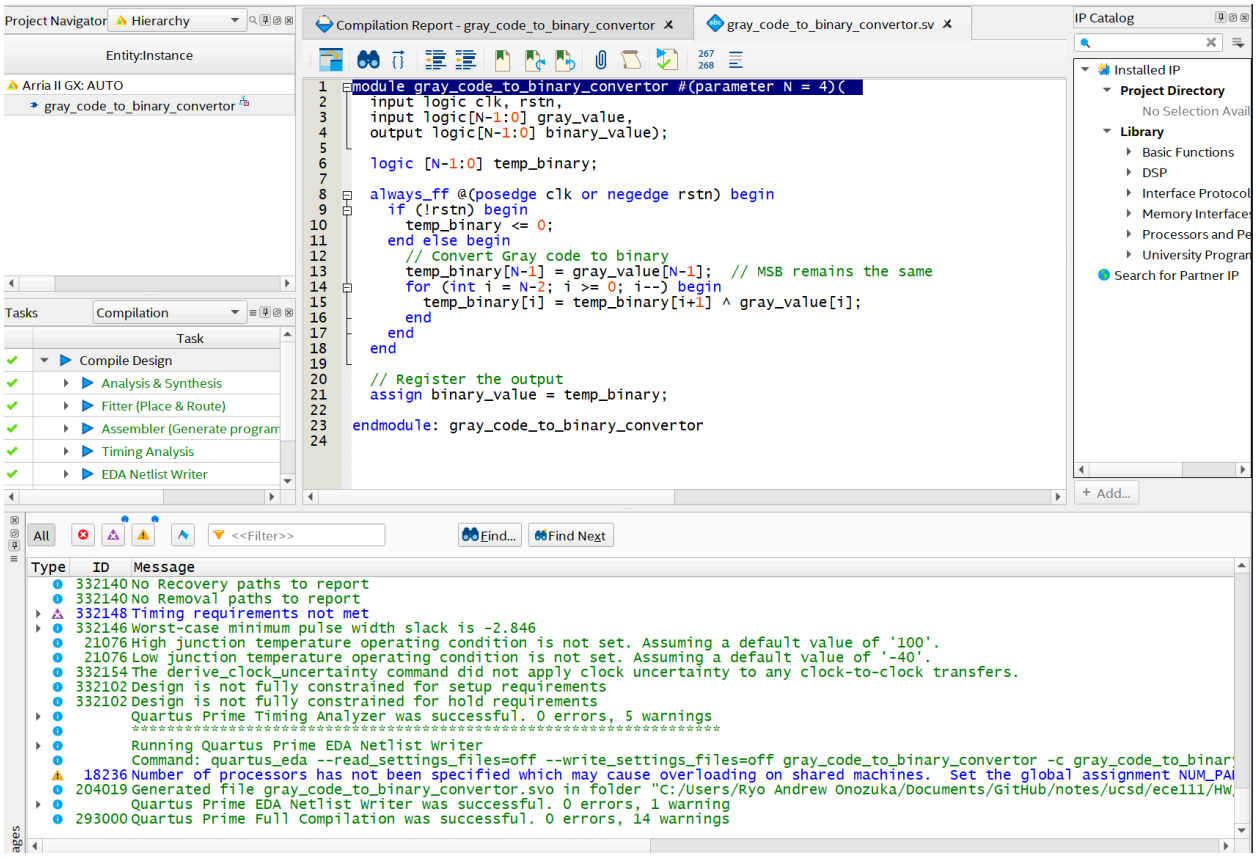




The simulation output aligns with the expected behavior for both shift and rotate operations in the barrel shifter. For shift operations (select = 0), right shifts with din = 1000 produce correct results for each shift_value: shifting by 1, 2, and 3 yields 0100, 0010, and 0001, respectively. Similarly, 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_code_convertor.sv



ucsd > ece111 > HW > Homework4 > gray_code_to_binary_convertor > gray_code_to_binary_convertor.sv

```sv
module gray_code_to_binary_convertor #(parameter N = 4)(
  input logic clk, rstn,
  input logic[N-1:0] gray_value,
  output logic[N-1:0] binary_value);

  logic [N-1:0] temp_binary;

  always_ff @(posedge clk or negedge rstn) begin
    if (!rstn) begin
      temp_binary <= 0;
    end else begin
      // Convert Gray code to binary
      temp_binary[N-1] = gray_value[N-1];  // MSB remains the same
      for (int i = N-2; i >= 0; i--) begin
        temp_binary[i] = temp_binary[i+1] ^ gray_value[i];
      end
    end
  end

  // Register the output
  assign binary_value = temp_binary;

endmodule: gray_code_to_binary_convertor
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

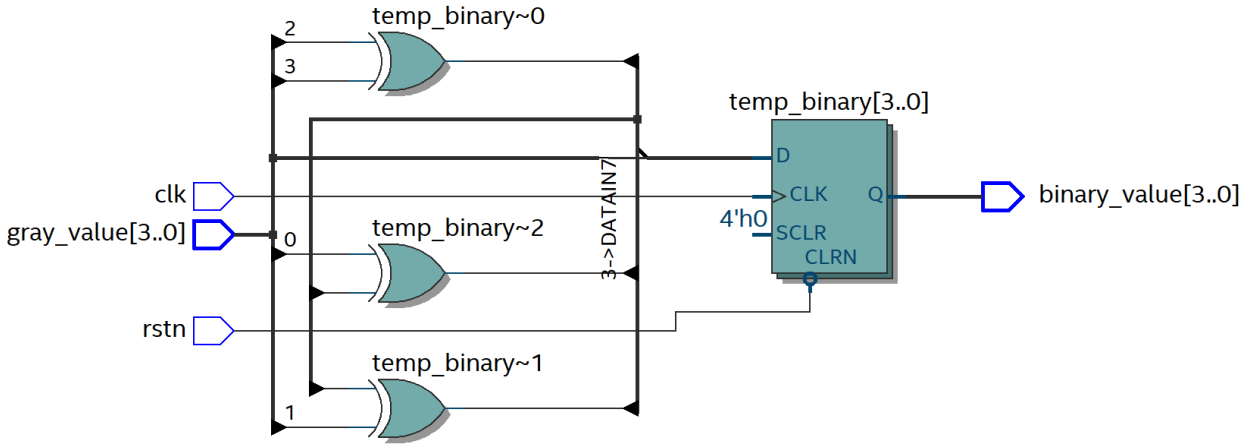
gray_to_binary_code_convertor.sv Resource Usage Summary

ucsd > ece111 > HW > Homework4 > gray_code_to_binary_convertor > gray_code_to_binary_convertor-Resource Usage Summary.rpt

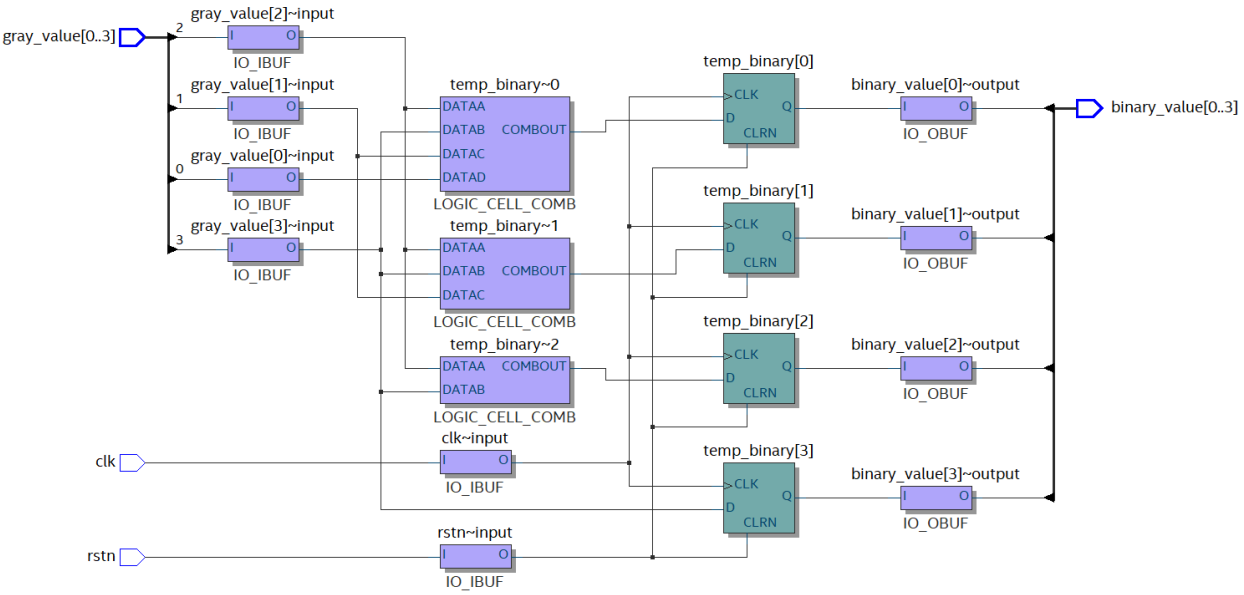
```
+-----------------------------------------------------------------------+
; Analysis & Synthesis Resource Usage Summary                          ;
+-----------------------------------------------+-----------------------+
; Resource                                      ; Usage                 ;
+-----------------------------------------------+-----------------------+
; Estimated ALUTs Used                          ; 3                     ;
;     -- Combinational ALUTs                    ; 3                     ;
;     -- Memory ALUTs                           ; 0                     ;
;     -- LUT_REGs                               ; 0                     ;
; Dedicated logic registers                     ; 4                     ;
;                                               ;                       ;
; Estimated ALUTs Unavailable                   ; 0                     ;
;     -- Due to unpartnered combinational logic ; 0                     ;
;     -- Due to Memory ALUTs                    ; 0                     ;
;                                               ;                       ;
; Total combinational functions                 ; 3                     ;
; Combinational ALUT usage by number of inputs  ;                       ;
;     -- 7 input functions                      ; 0                     ;
;     -- 6 input functions                      ; 0                     ;
;     -- 5 input functions                      ; 0                     ;
;     -- 4 input functions                      ; 1                     ;
;     -- <=3 input functions                    ; 2                     ;
;                                               ;                       ;
; Combinational ALUTs by mode                   ;                       ;
;     -- normal mode                           ; 3                     ;
;     -- extended LUT mode                     ; 0                     ;
;     -- arithmetic mode                       ; 0                     ;
;     -- shared arithmetic mode                ; 0                     ;
;                                               ;                       ;
; Estimated ALUT/register pairs used            ; 4                     ;
;                                               ;                       ;
; Total registers                               ; 4                     ;
;     -- Dedicated logic registers              ; 4                     ;
;     -- I/O registers                          ; 0                     ;
;     -- LUT_REGs                               ; 0                     ;
;                                               ;                       ;
;                                               ;                       ;
; I/O pins                                      ; 10                    ;
;                                               ;                       ;
; DSP block 18-bit elements                     ; 0                     ;
;                                               ;                       ;
; Maximum fan-out node                          ; gray_value[3]~input   ;
; Maximum fan-out                               ; 4                     ;
; Total fan-out                                 ; 35                    ;
; Average fan-out                               ; 1.30                  ;
+-----------------------------------------------+-----------------------+
```

Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

gray_to_binary_code_convertor.sv RTL viewer

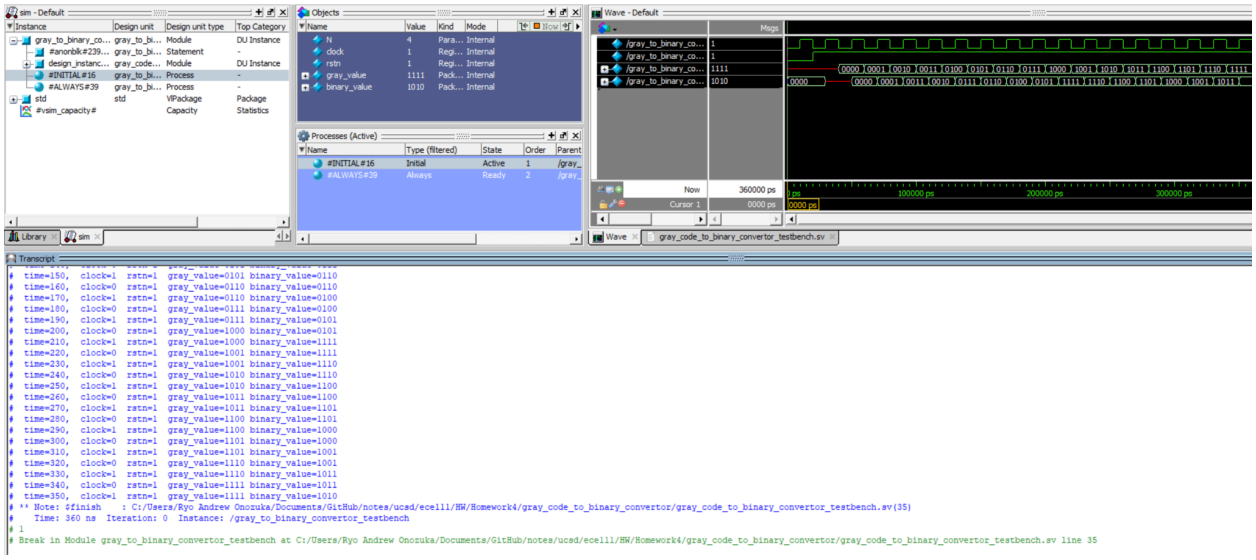


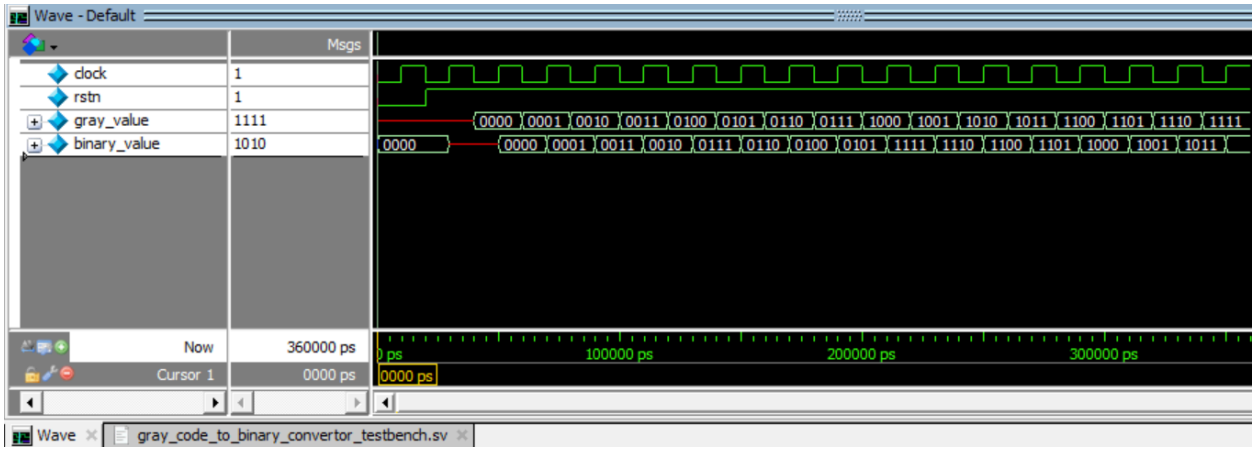
gray_to_binary_code_convertor.sv post mapping viewer



Andrew Onozuka A16760043
ECE 111 HW4 | 10/30/2024

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.