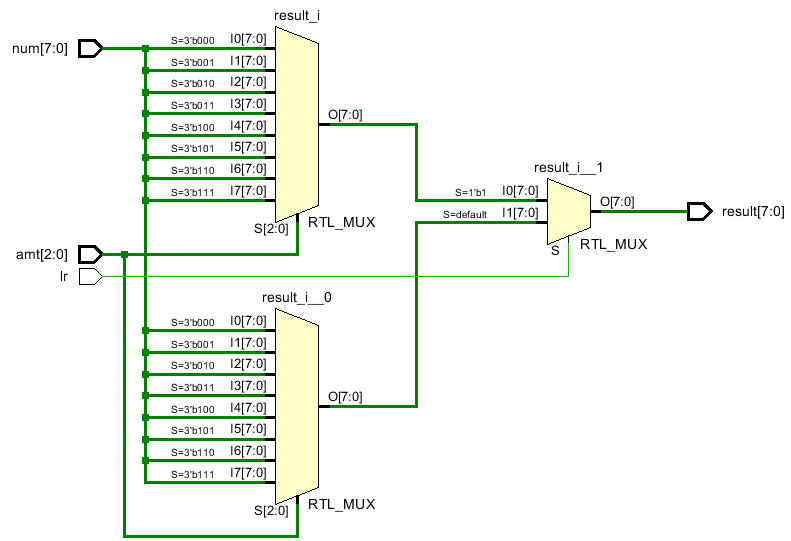
**3.12.1:**

1. A 2-1 mux was used to select a either left shifting or right shifting (shown here):



1. Testbench Code / Waveform:

`timescale 1 ns / 10 ps

module Testbench();

// signal declaration

logic [7:0] numIn, numOut;

logic [2:0] amt;

logic lr;

// instantiate barrel shifter

BarrelShifter uut (.num(numIn), .amt(amt), .lr(lr), .result(numOut));

// generate test values

initial begin

numIn = 8'b1001\_0110; // right shift once

amt = 1;

lr = 1'b1;

# 1

numIn = 8'b1001\_0110; // right shift twice

amt = 2;

lr = 1'b1;

# 1

numIn = 8'b1001\_0110; // left shift once

amt = 1;

lr = 1'b0;

# 1

numIn = 8'b1001\_0110; // left shift twice

amt = 2;

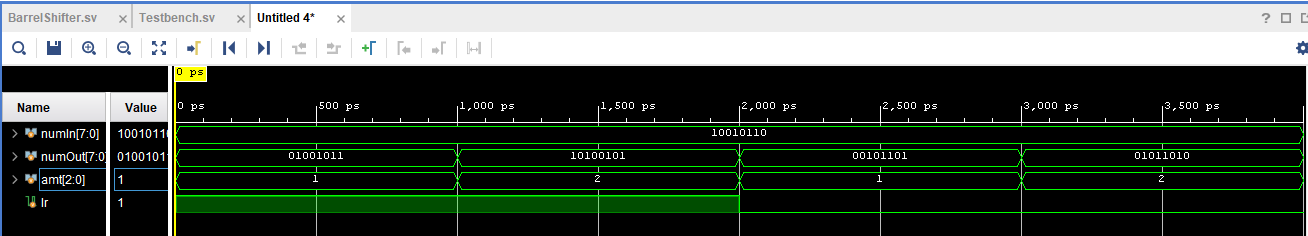
lr = 1'b0;

# 1

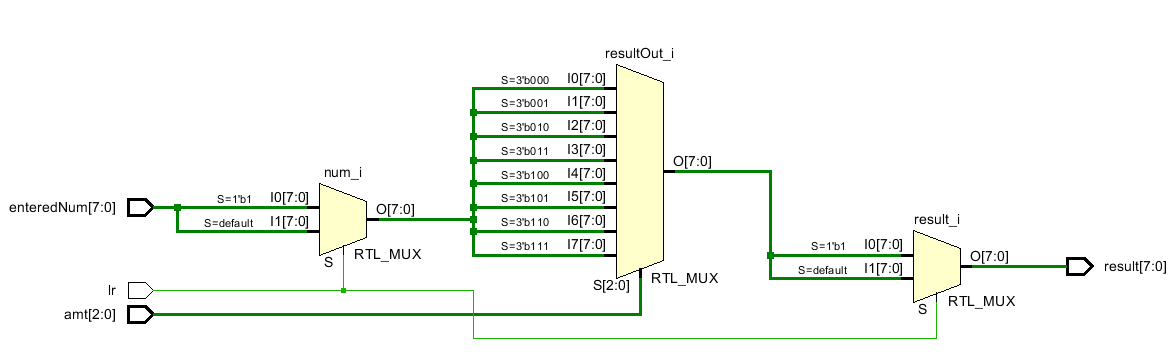
$stop;

end

endmodule



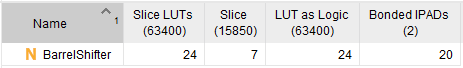
1. Video Link:
2. Reverser Barrel Shifter:
   1. Schematic:

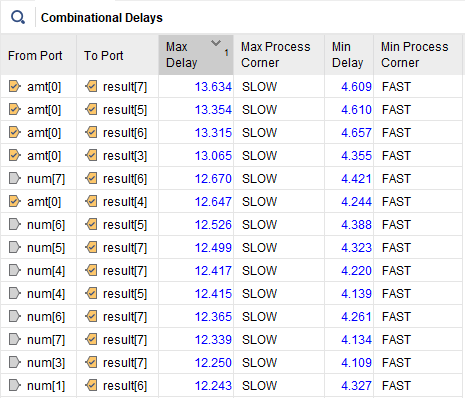


* 1. Testbench

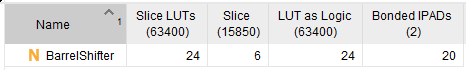
**( Same as previous part)**

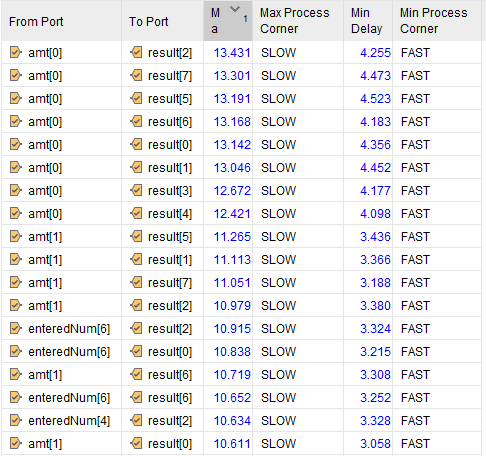
1. Video Demo
   1. <https://youtu.be/gUIKF-4KoiA>
2. Reports (logic cells and propagation delays):
   1. Left / Right Shifter Report



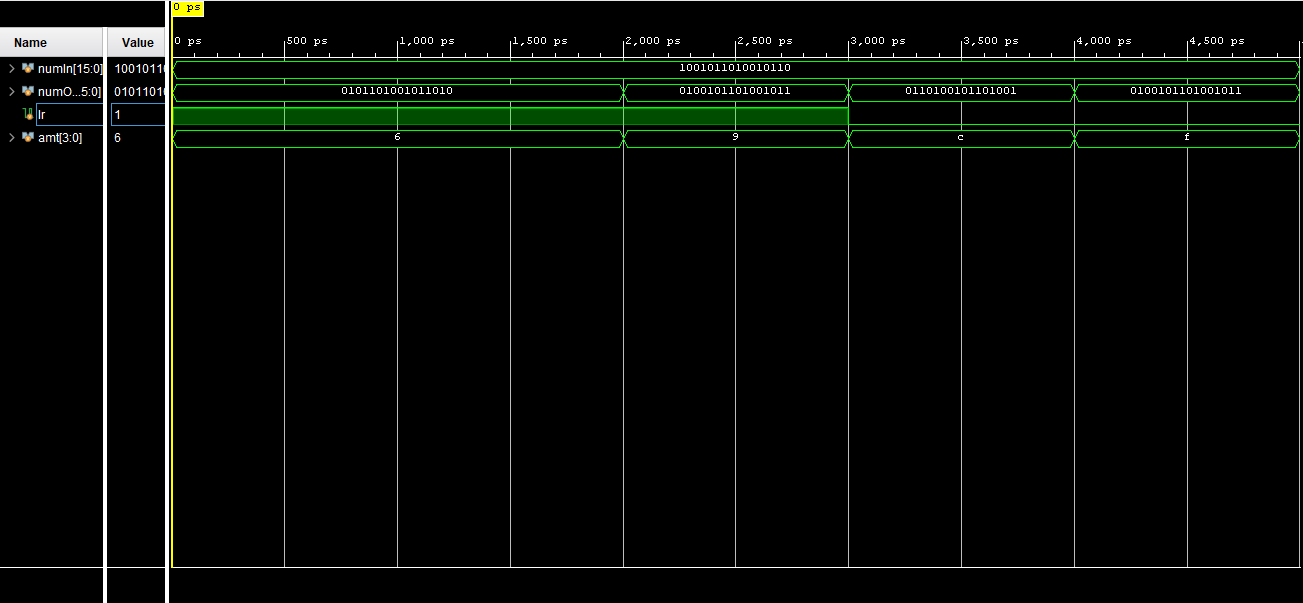


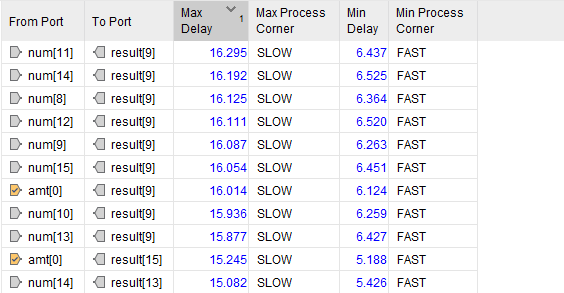
* 1. Reverser Barrel Shifter Report

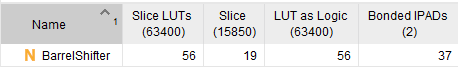




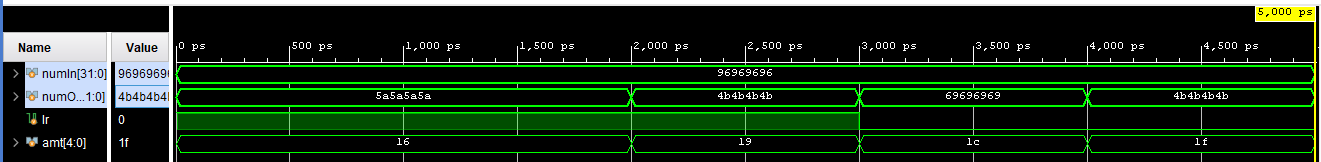
1. 16 bit expansion:

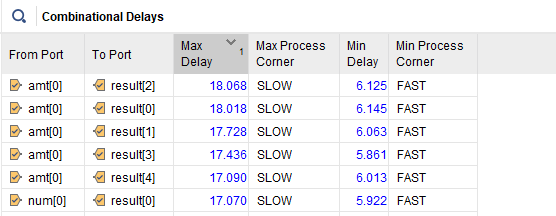


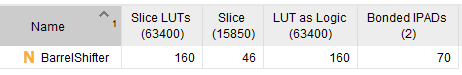




1. 32 bit expansion:

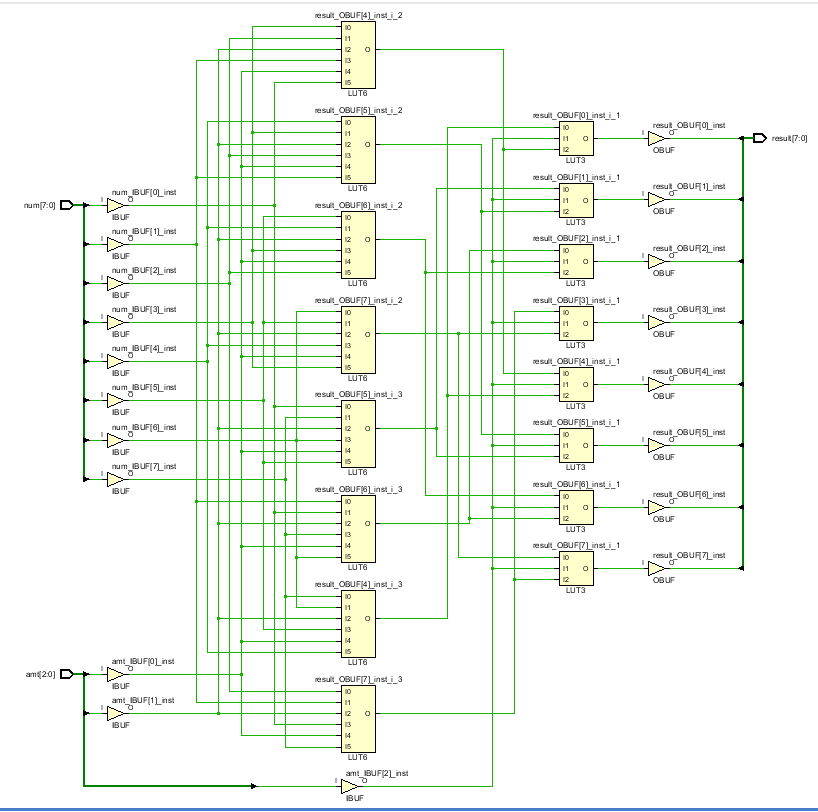






**3.12.2:**

1. Parametrized Circuit Design and Code:



// 3.12.1 - Part 7

module BarrelShifter

# (parameter N = 3) // 2 ^ N = bits for number to shift

(

input logic [(2\*\*N) - 1:0] num,

input logic [N - 1:0] amt,

output logic [(2\*\*N - 1):0] result

);

// used to dynamically create an 'N' amount of stages (s0, s1, ... sN)

logic [(2\*\*N) - 1:0] s [N:0];

// generate shifting 2-1 multiplexer dynamically

generate

genvar i;

for(i = 0; i < N; i = i + 1)

begin

if(i == 0)

assign s[0] = amt[0] ? {num[0], num[(2\*\*N)-1:1]} : num;

else

assign s[i] = amt[i] ? {s[i-1][(2\*\*i)-1:0], s[i-1][(2\*\*N)-1:(2\*\*i)]} : s[i-1];

end

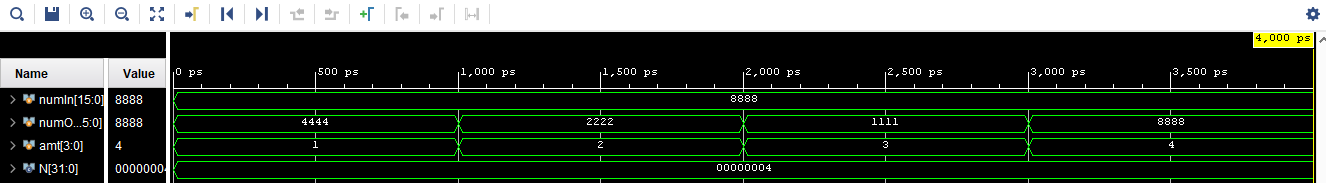
endgenerate

// assign result to the final stage of the shifter

assign result = s[N-1];

endmodule

1. Testbench and Simulation



`timescale 1 ns / 10 ps

module Testbench();

// signal declaration

localparam N = 4;

logic [(2\*\*N)-1:0] numIn, numOut;

logic [N-1:0] amt;

// instantiate barrel shifter

BarrelShifter #(.N(N)) uut (.num(numIn), .amt(amt), .result(numOut));

// generate test values

initial begin

numIn = 16'h8888; // right shift once

amt = 1;

# 1

numIn = 16'h8888; // right shift twice

amt = 2;

# 1

numIn = 16'h8888; // right shift three times

amt = 3;

# 1

numIn = 16'h8888; // right shift four times

amt = 4;

# 1

$stop;

end

endmodule

1. 16-Bit Barrel Shifter Code

(see number 1 of 3.12.2 for 16-bit shifter code)

* Parameter N = 4, which equates to 16 bits wide

1. FPGA Implementation

<https://youtu.be/TIN8IrNSeCA>