PES UNIVERSITY

FINAL SEMESTER ASSESSMENT (FSA)
B.TECH. (CSE)
III SEMESTER

UE18CS206 – DIGITAL DESIGN & COMPUTER ORGANIZATION LABORATORY

PROJECT REPORT

ON

"DESIGN AND IMPLEMENTATION OF A 16-BIT BARREL SHIFTER"

SUBMITTED BY:

NAME:	SRN:
Naik Bhavan Chandrashekhar	PES2201800047
Varun Seshu	PES2201800074
Achyuth K Kowshik	PES2201800022
Hritik Shanbhag	PES2201800082

AUGUST – DECEMBER 2019

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ELECTRONIC CITY CAMPUS,

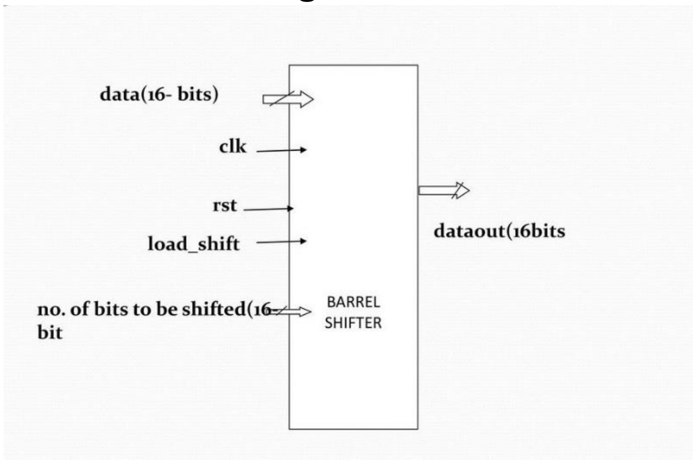
BENGALURU – 560100, KARNATAKA, INDIA

ABSTRACT OF THE PROJECT:

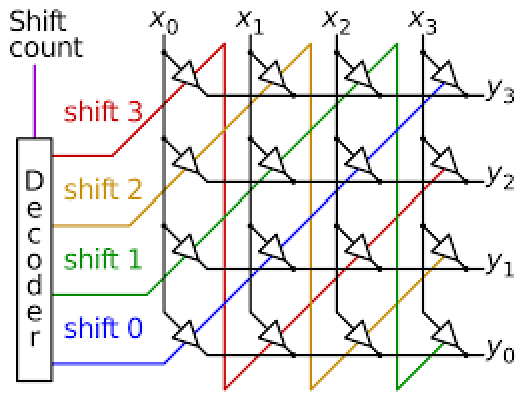
- We will be designing and implementing a 16-bit barrel shifter for our project.
- A barrel shifter is a digital circuit that can shift a data word by a specified number of bits without the use of any sequential logic, only pure combinational logic.
- It is also defined as a specialized digital electronic circuit with the purpose of shifting an entire data word by a specified number of bits by only using combinational logic, with no sequential logic used.
- The simplest way of achieving this is by using a series of multiplexers where one output is connected to the input of the next multiplexer in the chain, in a specific manner that depends on the amount of shift specified.
- It is implemented with a sequence of shift multiplexers, each shifting a word by 2k bit positions for different values of k.
- We have used right shift operation in our project.

CIRCUIT DIAGRAMS:

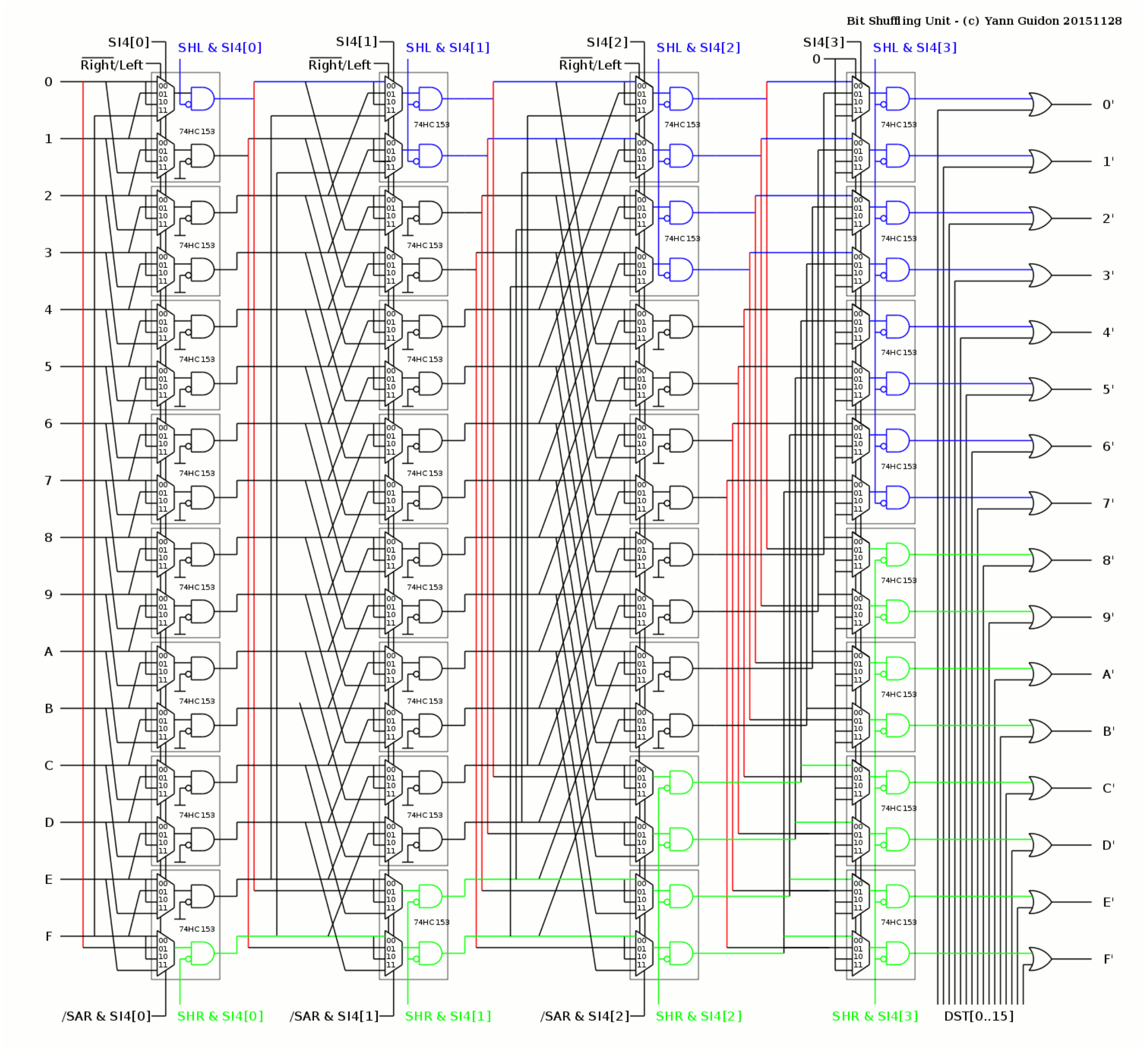
a. Basic Diagram:



b. Basic barrel shifter design:



c. 16-bit barrel shifter diagram:



MAIN VERILOG CODE:

```
module mux2 (input wire i0, i1, j, output wire o);
  assign o = (j==0)?i0:i1;
endmodule

module barrel_shift_16bit (in, ctrl, out);
  input  [15:0] in;
  input [3:0] ctrl;
  output [15:0] out;
  wire [15:0] x,y,z;


//8bit shift right
mux2 mux_15(in[15],1'b0,ctrl[3],x[15]);
mux2 mux_14(in[14],1'b0,ctrl[3],x[14]);
```

```
mux2 mux_13(in[13],1'b0,ctrl[3],x[13]);
mux2 mux_12(in[12],1'b0,ctrl[3],x[12]);
mux2 mux_11(in[11],1'b0,ctrl[3],x[11]);
mux2 mux_10(in[10],1'b0,ctrl[3],x[10]);
mux2 mux_9(in[9],1'b0,ctrl[3],x[9]);
mux2 mux_8(in[8],1'b0,ctrl[3],x[8]);
mux2 mux_7(in[7],in[15],ctrl[3],x[7]);
mux2 mux_6(in[6],in[14],ctrl[3],x[6]);
mux2 mux_5(in[5],in[13],ctrl[3],x[5]);
mux2 mux_4(in[4],in[12],ctrl[3],x[4]);
mux2 mux_3(in[3],in[11],ctrl[3],x[3]);
mux2 mux_2(in[2],in[10],ctrl[3],x[2]);
mux2 mux_1(in[1],in[9],ctrl[3],x[1]);
mux2 mux_0(in[0],in[8],ctrl[3],x[0]);

//4bit shift right
mux2 mux_31(x[15],1'b0,ctrl[2],y[15]);
mux2 mux_30(x[14],1'b0,ctrl[2],y[14]);
mux2 mux_29(x[13],1'b0,ctrl[2],y[13]);
mux2 mux_28(x[12],1'b0,ctrl[2],y[12]);
mux2 mux_27(x[11],x[15],ctrl[2],y[11]);
mux2 mux_26(x[10],x[14],ctrl[2],y[10]);
mux2 mux_25(x[9],x[13],ctrl[2],y[9]);
mux2 mux_24(x[8],x[12],ctrl[2],y[8]);
mux2 mux_23(x[7],x[11],ctrl[2],y[7]);
mux2 mux_22(x[6],x[10],ctrl[2],y[6]);
mux2 mux_21(x[5],x[9],ctrl[2],y[5]);
mux2 mux_20(x[4],x[8],ctrl[2],y[4]);
mux2 mux_19(x[3],x[7],ctrl[2],y[3]);
mux2 mux_18(x[2],x[6],ctrl[2],y[2]);
mux2 mux_17(x[1],x[5],ctrl[2],y[1]);
mux2 mux_16(x[0],x[4],ctrl[2],y[0]);

//2bit shift right
mux2 mux_47(y[15],1'b0,ctrl[1],z[15]);
mux2 mux_46(y[14],1'b0,ctrl[1],z[14]);
mux2 mux_45(y[13],y[15],ctrl[1],z[13]);
mux2 mux_44(y[12],y[14],ctrl[1],z[12]);
mux2 mux_43(y[11],y[13],ctrl[1],z[11]);
mux2 mux_42(y[10],y[12],ctrl[1],z[10]);
mux2 mux_41(y[9],y[11],ctrl[1],z[9]);
mux2 mux_40(y[8],y[10],ctrl[1],z[8]);
mux2 mux_39(y[7],y[9],ctrl[1],z[7]);
mux2 mux_38(y[6],y[8],ctrl[1],z[6]);
mux2 mux_37(y[5],y[7],ctrl[1],z[5]);
mux2 mux_36(y[4],y[6],ctrl[1],z[4]);
mux2 mux_35(y[3],y[5],ctrl[1],z[3]);
mux2 mux_34(y[2],y[4],ctrl[1],z[2]);
```

```
mux2 mux_33(y[1],y[3],ctrl[1],z[1]);
mux2 mux_32(y[0],y[2],ctrl[1],z[0]);

//1bit shift right
mux2 mux_63(z[15],1'b0,ctrl[0],out[15]);
mux2 mux_62(z[14],z[15],ctrl[0],out[14]);
mux2 mux_61(z[13],z[14],ctrl[0],out[13]);
mux2 mux_60(z[12],z[13],ctrl[0],out[12]);
mux2 mux_59(z[11],z[12],ctrl[0],out[11]);
mux2 mux_58(z[10],z[11],ctrl[0],out[10]);
mux2 mux_57(z[9],z[10],ctrl[0],out[9]);
mux2 mux_56(z[8],z[9],ctrl[0],out[8]);
mux2 mux_55(z[7],z[8],ctrl[0],out[7]);
mux2 mux_54(z[6],z[7],ctrl[0],out[6]);
mux2 mux_53(z[5],z[6],ctrl[0],out[5]);
mux2 mux_52(z[4],z[5],ctrl[0],out[4]);
mux2 mux_51(z[3],z[4],ctrl[0],out[3]);
mux2 mux_50(z[2],z[3],ctrl[0],out[2]);
mux2 mux_49(z[1],z[2],ctrl[0],out[1]);
mux2 mux_48(z[0],z[1],ctrl[0],out[0]);
endmodule
```

TEST-BENCH FILE:

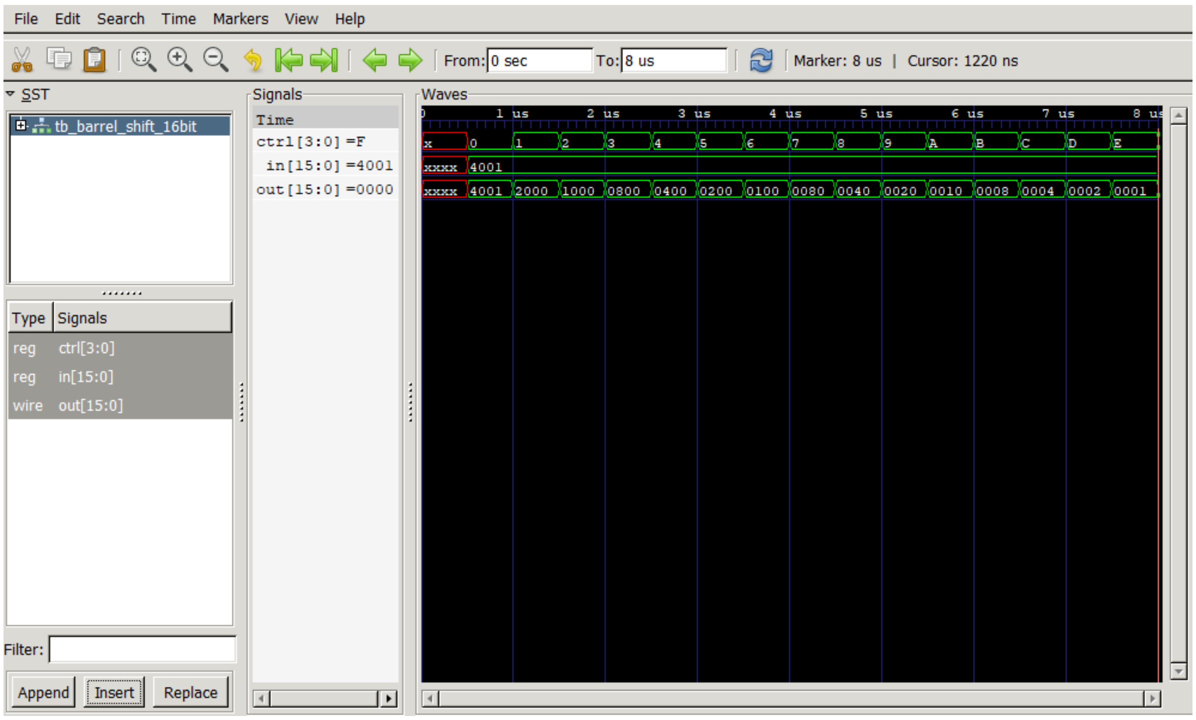
```
`timescale 100 ns / 100 ps
module tb_barrel_shift_16bit;
  reg [15:0] in;
  reg [3:0] ctrl;
  wire [15:0] out;
  initial begin $dumpfile("tb_project.vcd");
 $dumpvars(0,tb_barrel_shift_16bit);
 end
barrel_shift_16bit bs(.in(in), .ctrl(ctrl), .out(out));

initial
 begin
     $display($time, " << Starting the Simulation >>");    //Monitor display
     /*   in=16'd000; ctrl=4'd0; //no shift
    #10 in=16'd128; ctrl=4'd1; //shift by 1 bit
    #10 in=16'd128; ctrl=4'd2; //shift 2 bits
    #10 in=16'd128; ctrl=4'd4; //shift 4 bits
    #10 in=16'd255; ctrl=4'd7; //shift by 7 bits
    #10 in=16'd255; ctrl=4'd8; //shift by 8 bits
    #10 in=16'd511; ctrl=4'd15;//shift by 15 bits*/

    //16385 is 2 to the power of 14
        #5 in=16'd16385; ctrl=4'd0; //shift 0 bits;
        #5 in=16'd16385; ctrl=4'd1; //shift 1 bits;
        #5 in=16'd16385; ctrl=4'd2; //shift 2 bits;
        #5 in=16'd16385; ctrl=4'd3; //shift 3 bits;
        #5 in=16'd16385; ctrl=4'd4; //shift 4 bits;
        #5 in=16'd16385; ctrl=4'd5; //shift 5 bits;
        #5 in=16'd16385; ctrl=4'd6; //shift 6 bits;
        #5 in=16'd16385; ctrl=4'd7; //shift 7 bits;
        #5 in=16'd16385; ctrl=4'd8; //shift 8 bits;
        #5 in=16'd16385; ctrl=4'd9; //shift 9 bits;
        #5 in=16'd16385; ctrl=4'd10; //shift 10 bits;
        #5 in=16'd16385; ctrl=4'd11; //shift 11 bits;
        #5 in=16'd16385; ctrl=4'd12; //shift 12 bits;
        #5 in=16'd16385; ctrl=4'd13; //shift 13 bits;
        #5 in=16'd16385; ctrl=4'd14; //shift 14 bits;
        #5 in=16'd16385; ctrl=4'd15; //shift 15 bits;
  end
    initial begin
      $monitor("Input=%d, Control=%d, Output=%d",in,ctrl,out);    // this stam
ent is used for direct display
    end
endmodule
```

SCREEN-SHOTS OF THE OUTPUT:

a. Console Output:

```
VCD info: dumpfile tb_project.vcd opened for output.
                   0 << Starting the Simulation >>
Input=    x, Control= x, Output=    x
Input=16385, Control= 0, Output=16385
Input=16385, Control= 1, Output= 8192
Input=16385, Control= 2, Output= 4096
Input=16385, Control= 3, Output= 2048
Input=16385, Control= 4, Output= 1024
Input=16385, Control= 5, Output=  512
Input=16385, Control= 6, Output=  256
Input=16385, Control= 7, Output=  128
Input=16385, Control= 8, Output=   64
Input=16385, Control= 9, Output=   32
Input=16385, Control=10, Output=   16
Input=16385, Control=11, Output=    8
Input=16385, Control=12, Output=    4
Input=16385, Control=13, Output=    2
Input=16385, Control=14, Output=    1
Input=16385, Control=15, Output=    0
```

b. GTKWAVE OUTPUT:



From: 0 sec To: 8 us | Marker: 8 us | Cursor: 1220 ns

tb_barrel_shift_16bit

Type	Signals
reg	ctrl[3:0]
reg	in[15:0]
wire	out[15:0]

```
Time
ctrl[3:0] =F
 in[15:0] =4001
out[15:0] =0000
```

```
0       1 us    2 us    3 us    4 us    5 us    6 us    7 us    8 us
x    0    1    2    3    4    5    6    7    8    9    A    B    C    D    E
xxxx 4001
xxxx 4001 2000 1000 0800 0400 0200 0100 0080 0040 0020 0010 0008 0004 0002 0001
```