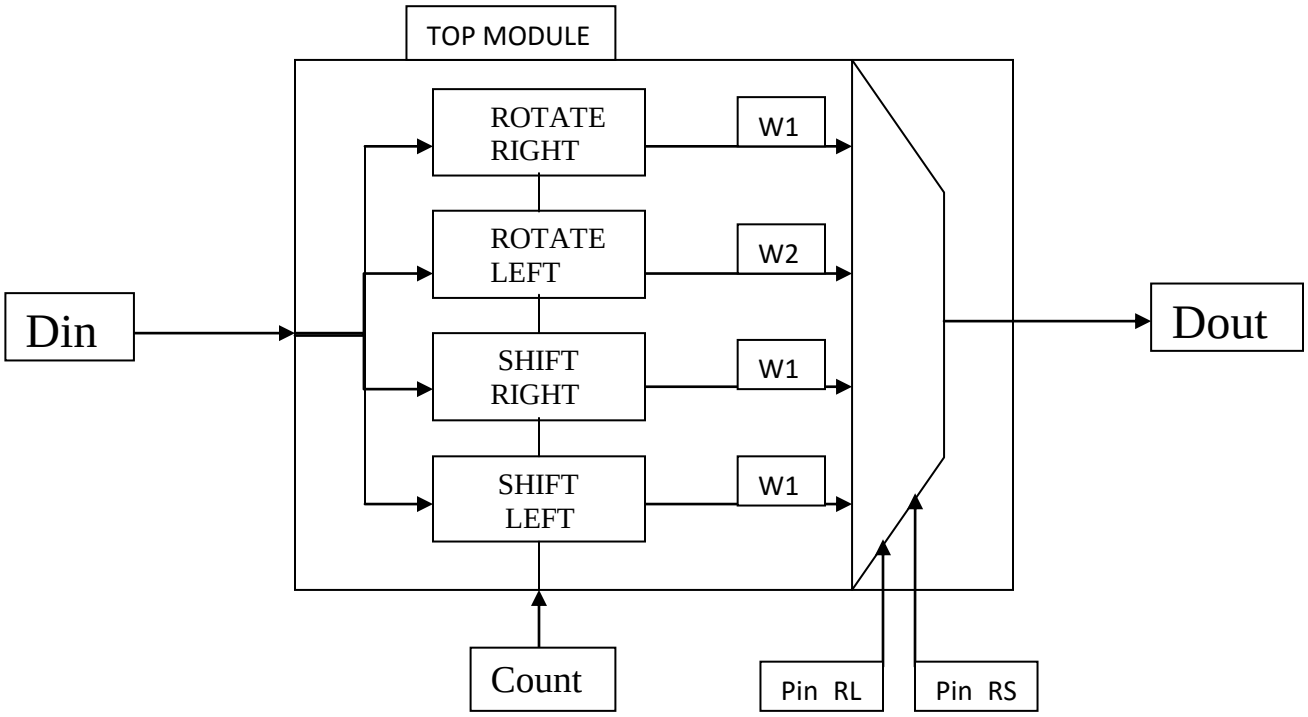
FRONT END VLSI TRAINING - 2015

ASSIGNMENT NO - 4

TOPIC - BARREL SHIFTER

NAME- AJINKYA PADWAD

BLOCK DIAGRAM-



WORKING -

Rotate Right Block -

This block receives 8-bit data input from 'Din' signal of the main module and rotates the input towards right as per the number given by the 'count' signal from the main module.

The right rotate operation is carried out by using the method of concatenation of the bits required to be rotated. It shifts the input bits from MSB to LSB and the last LSB bit is concatenated back to MSB.

Syntax : Dout <= {Din[1:0],Din[7:2]}; //concatenation example 1

Rotate Left Block -

This block receives 8-bit data input from 'Din' signal of the main module and rotates the input towards left as per the number given by the 'count' signal from the main module.

The left rotate operation is carried out by using the method of concatenation of the bits required to be rotated. It shifts the input bits from LSB to MSB and the last MSB bit is concatenated back to LSB.

Syntax : Dout <= {Din[6:0],Din[7]}; //concatenation example 2

Shift Right Block -

This block receives 8-bit data input from 'Din' signal of the main module and shifts the input towards right as per the number given by the 'count' signal from the main module. This is a logical shift and a '0' is added at each bit shifted.

The right shift operation is carried out by using the logical shift '>>' operator. It shifts the input bits from MSB to LSB and adds '0' to the vacant bits from MSB side.

Syntax: Dout <= Din >> 1; //Shifted right by 1 bit

Shift Left Block -

This block receives 8-bit data input from 'Din' signal of the main module and shifts the input towards left as per the number given by the 'count' signal from the main module. This is a logical shift and a '0' is added at each bit shifted.

The left shift operation is carried out by using the logical shift '<<' operator. It shifts the input bits from LSB to MSB and adds '0' to the vacant bits from LSB side.

Syntax: Dout <= Din << 1; //Shifted left by 1 bit

Main Module Working -

1. Receives 'Din' signal as the 8-bit input to be shifted/rotated left/right by number of bits specified by 'count'
2. The data is fed to the four blocks mentioned above and also the count.
3. Each block independently carries out the shifting or rotating operation.
4. The operation to be carried out is selected using if-else logic via select pins.
5. Pin_RS selects the rotate or shift operation.
6. Pin_RL selects the left or right operation.
7. The different cases are evaluated at the main module and given out to the final output 'Dout'.

VERILOG CODES-

Rotate Right Block

```
module RRight(
   input [7:0] Din,
         output reg [7:0] Dout,
   input [2:0] count

   );

always@(*)
begin
case(count)
3'b000: Dout <= Din;
3'b001: Dout <= {Din[0],Din[7:1]};
3'b010: Dout <= {Din[1:0],Din[7:2]};
3'b011: Dout <= {Din[2:0],Din[7:3]};
3'b100: Dout <= {Din[3:0],Din[7:4]};
3'b101: Dout <= {Din[4:0],Din[7:5]};
3'b110: Dout <= {Din[5:0],Din[7:6]};
default: Dout <= {Din[6:0],Din[7]};
endcase
end
endmodule
```

Rotate Left Block

```
module RLeft(
   input [7:0] Din,
         output reg [7:0] Dout,
   input [2:0] count

   );

always@(*)
begin
case(count)

3'b000: Dout <= Din;
3'b001: Dout <= {Din[6:0],Din[7]};
3'b010: Dout <= {Din[5:0],Din[7:6]};
3'b011: Dout <= {Din[4:0],Din[7:5]};
3'b100: Dout <= {Din[3:0],Din[7:4]};
3'b101: Dout <= {Din[2:0],Din[7:3]};
3'b110: Dout <= {Din[1:0],Din[7:2]};
default: Dout <= {Din[0],Din[7:1]};
endcase
end
endmodule
```

Shift Right Block

```
module SRight(
   input [7:0] Din,
         output reg [7:0] Dout,
   input [2:0] count

   );
always@(*)
begin
case(count)
3'b000: Dout <= Din >> 0;
3'b001: Dout <= Din >> 1;
3'b010: Dout <= Din >> 2;
3'b011: Dout <= Din >> 3;
3'b100: Dout <= Din >> 4;
3'b101: Dout <= Din >> 5;
3'b110: Dout <= Din >> 6;
default: Dout <= Din >> 7;
endcase
end
endmodule
```

Shift Left Block

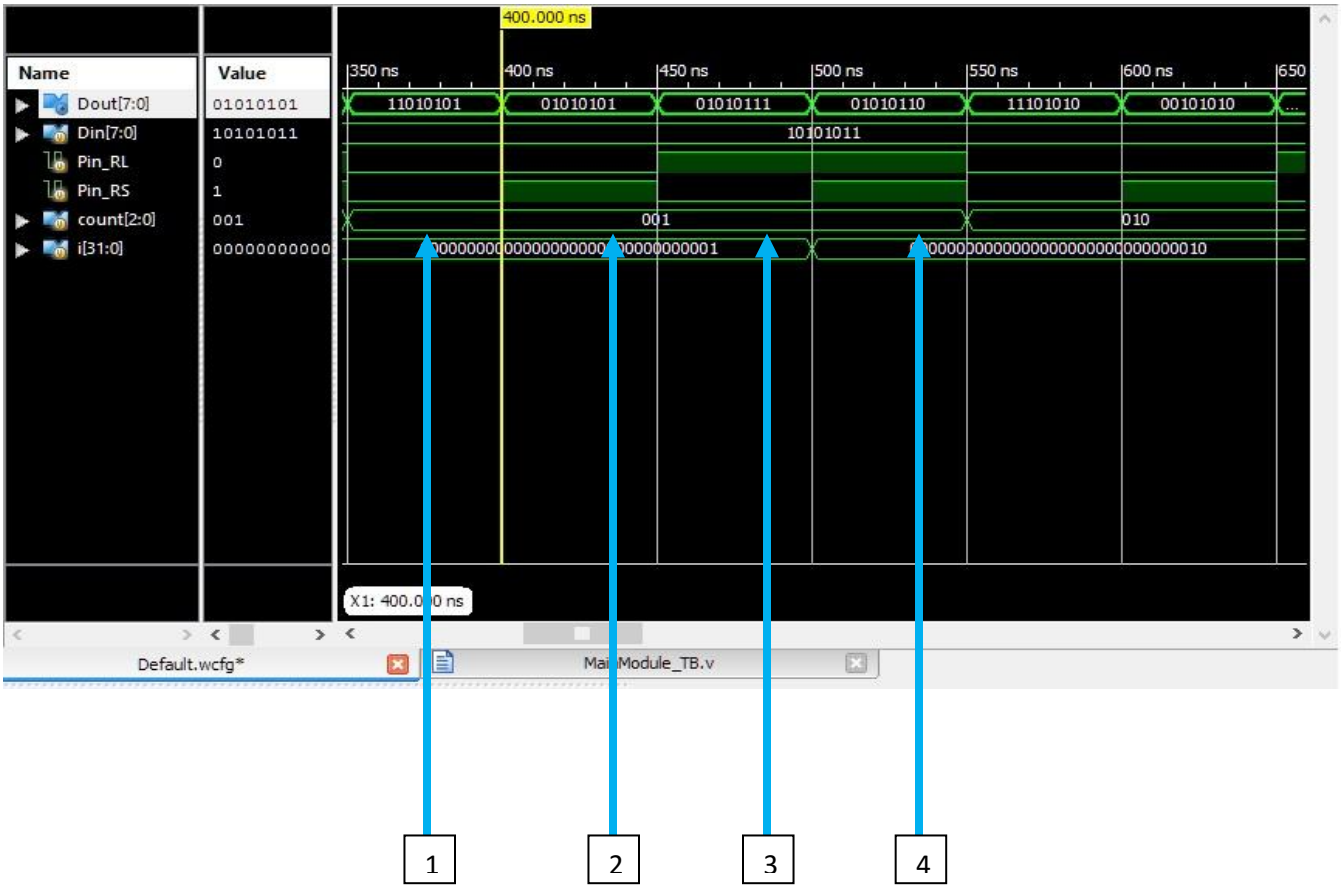
```
module SLeft(
   input [7:0] Din,
         output reg [7:0] Dout,
   input [2:0] count
   );

always@(*)
begin
case(count)
3'b000: Dout <= Din << 0;
3'b001: Dout <= Din << 1;
3'b010: Dout <= Din << 2;
3'b011: Dout <= Din << 3;
3'b100: Dout <= Din << 4;
3'b101: Dout <= Din << 5;
3'b110: Dout <= Din << 6;
default: Dout <= Din << 7;
endcase
end
endmodule
```

Main Module

```
module MainModule(
   input [7:0] Din,
   input Pin_RL,
   input Pin_RS,
   input [2:0] count,
   output reg [7:0] Dout
   );
wire [7:0] w1,w2,w3,w4;
RLeft RL (.Din(Din),.count(count),.Dout(w1));          // Rotate Left block instantiation
RRight RR (.Din(Din),.count(count),.Dout(w2));         // Rotate Right block instantiation
SLeft SL (.Din(Din),.count(count),.Dout(w3));          // Shift Left block instantiation
SRight SR (.Din(Din),.count(count),.Dout(w4));         // Shift Right block instantiation

always@(*)
begin

if (Pin_RL & Pin_RS == 0) //left rotate
Dout = w1;
else if (Pin_RL == 0 & Pin_RS == 0)  //right rotate
Dout = w2;
else if (Pin_RL & Pin_RS) //left shift
Dout = w3;
else
Dout = w4; //right shift
end

endmodule
```

TEST BENCH SIMULATION-

(Operation for 1 bit shift/rotate)



1. Rotate Right
2. Shift Right
3. Rotate Left
4. Shift Left