30 Days of RTL Coding

-By T Tushar Shenoy

Day 17

Problem Statement: Implementing Shift Registers –Part II

3. Parallel In Serial Out
4. Parallel In Parallel Out

Using Structural Style of Implementation.

Theory:

Parallel IN Serial OUT

In the "Parallel IN Serial OUT" register, the data is entered in a parallel way, and the outcome comes serially. A four-bit "Parallel IN Serial OUT" register is designed below. The input of the flip flop is the output of the previous Flip Flop. The input and outputs are connected through the combinational circuit. Through this combinational circuit, the binary input B_0, B_1, B_2, B_3 are passed. The shift mode and the load mode are the two modes in which the "PISO" circuit works.

Load mode

The bits B_0, B_1, B_2, and B_3 are passed to the corresponding flip flops when the second, fourth, and sixth "AND" gates are active. These gates are active when the shift or load bar line set to 0. The binary inputs B0, B1, B2, and B3 will be loaded into the respective flip-flops when the edge of the clock is high. Thus, parallel loading occurs.

Shift mode

The second, fourth, and sixth gates are inactive when the load and shift line set to 0. So, we are not able to load data in a parallel way. At this time, the first, third, and fifth gates will be activated, and the shifting of the data will be left to the right bit. In this way, the "Parallel IN Serial OUT" operation occurs.

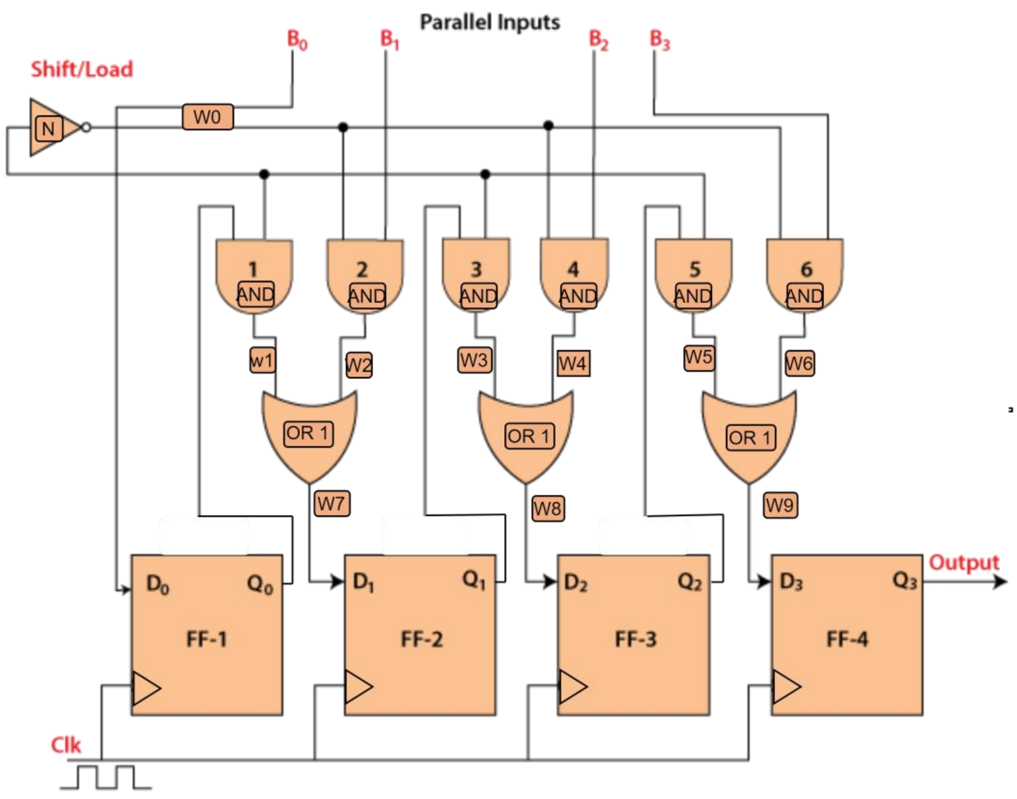


FIG: Parallel IN Serial OUT Shift Register

Verilog Code:

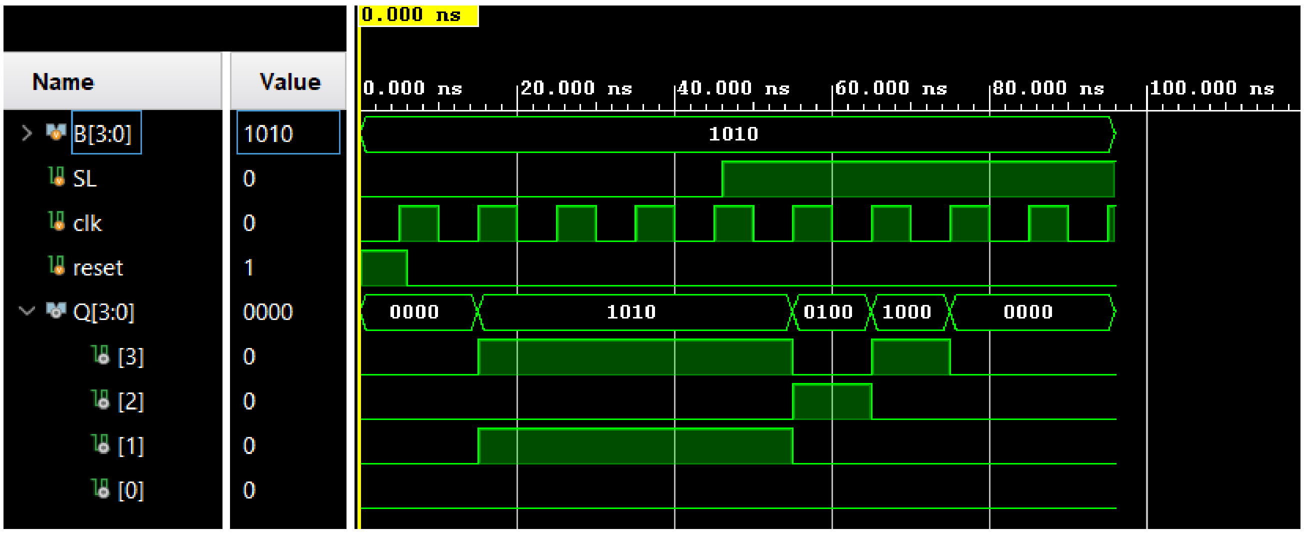
```
//Verilog Code for D Flip Flop
module D_Flip_Flop(D, reset, clk, Q, Qb);
 input D, reset, clk;
 output reg Q, Qb;

 always @(posedge clk,posedge reset)
 begin
 if (reset == 1)
 Q<=1'b0;
 else
 begin
 Q<=D;
 end
 Qb<=~Q;
 end
endmodule

//Verilog Code for  Parallel in Serial out Shift Register
module PISO_Shift_Register(B,SL,clk,reset,Q);

input [3:0]B;
input SL,clk,reset;
output[3:0]Q;

wire [9:0]w;
not N(w[0],SL);
```

```
and A1(w[1],SL,Q[0]);
and A2(w[2],w[0],B[1]);
and A3(w[3],SL,Q[1]);
and A4(w[4],w[0],B[2]);
and A5(w[5],SL,Q[2]);
and A6(w[6],w[0],B[3]);

or O1(w[7],w[1],w[2]);
or O2(w[8],w[3],w[4]);
or O3(w[9],w[5],w[6]);

D_Flip_Flop FF1(.D(B[0]),.reset(reset),.clk(clk),.Q(Q[0]));

D_Flip_Flop FF2(.D(w[7]),.reset(reset),.clk(clk),.Q(Q[1]));

D_Flip_Flop FF3(.D(w[8]),.reset(reset),.clk(clk),.Q(Q[2]));

D_Flip_Flop FF4(.D(w[9]),.reset(reset),.clk(clk),.Q(Q[3]));

endmodule
```

Testbench Code:

```
//Testbench Code for Serial in Parallel in Serial out Shift Register
module PISO_Shift_Register_tb();

reg [3:0]B;
reg SL,clk,reset;
wire [3:0]Q;

PISO_Shift_Register dut(.B(B),.SL(SL),.clk(clk),.reset(reset),.Q(Q));

initial begin
SL=1'b0;
B=4'b1010;
clk=1'b0;
reset=1'b1;
#6  reset=1'b0;
#40 SL=1'b1;
//Add More test Cases Here
#50 $finish;

end

always #5 clk=~clk;

endmodule
```

Simulation Output:



Parallel IN Parallel OUT

In "Parallel IN Parallel OUT", the inputs and the outputs come in a parallel way in the register. The inputs B_0, B_1, B_2, and B_3, are directly passed to the data inputs D_0, D_1, D_2, and D_3 of the respective flip flop. The bits of the binary input is loaded to the flip flops when the Positive clock edge is applied. The clock pulse is required for loading all the bits. At the output side, the loaded bits appear.

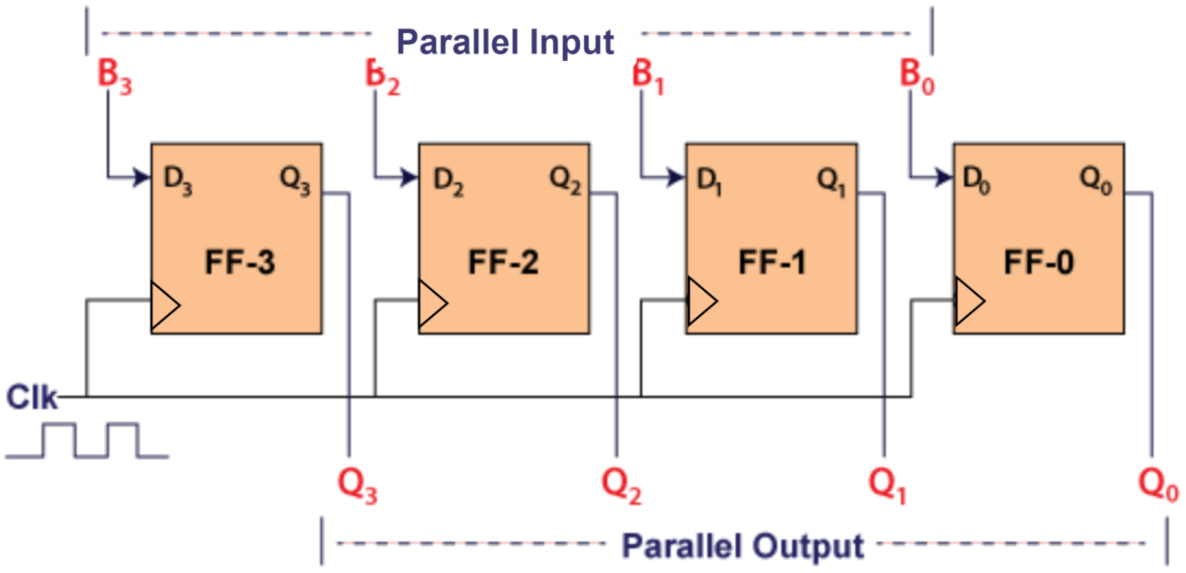


FIG: Parallel IN Parallel OUT Shift Register

Verilog Code:

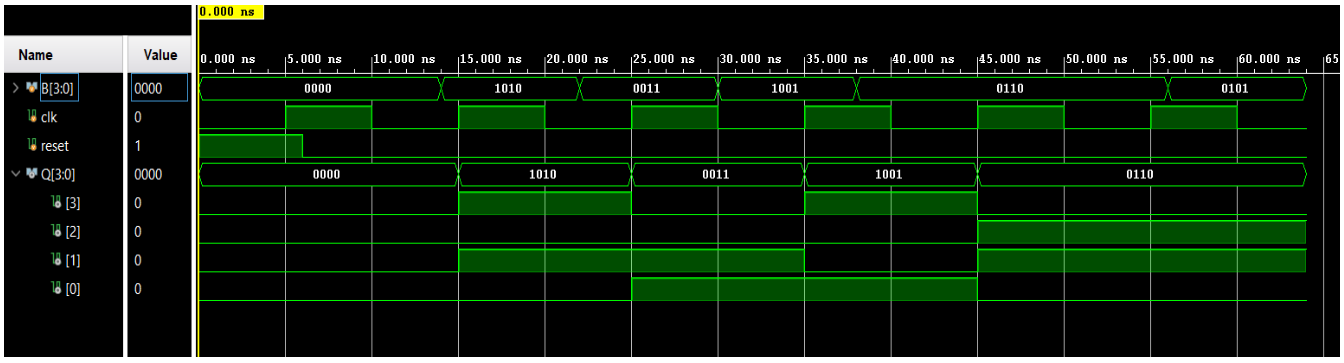
```
//Verilog Code for D Flip Flop
module D_Flip_Flop(D, reset, clk, Q, Qb);
 input D, reset, clk;
 output reg Q, Qb;

 always @(posedge clk,posedge reset)
 begin
 if (reset == 1)
 Q<=1'b0;
 else
 begin
 Q<=D;
 end
 Qb<=~Q;
 end
endmodule
```

```
//Verilog Code for Parallel in Parallel out Shift Register
module PIPO_Shift_Register(B,clk,reset,Q);

input [3:0]B;
input clk,reset;
output[3:0]Q;


D_Flip_Flop FF3(.D(B[3]),.reset(reset),.clk(clk),.Q(Q[3]));

D_Flip_Flop FF2(.D(B[2]),.reset(reset),.clk(clk),.Q(Q[2]));

D_Flip_Flop FF1(.D(B[1]),.reset(reset),.clk(clk),.Q(Q[1]));

D_Flip_Flop FF0(.D(B[0]),.reset(reset),.clk(clk),.Q(Q[0]));

endmodule
```

Testbench Code:

```
//Testbench Code for Parallel in Parallel out Shift Register
module PIPO_Shift_Register_tb();

reg [3:0]B;
reg clk,reset;
wire [3:0]Q;

PIPO_Shift_Register dut(.B(B),.clk(clk),.reset(reset),.Q(Q));

initial begin
B=4'b0000;
clk=1'b0;
reset=1'b1;
#6  reset=1'b0;
#8 B=4'b1010;
#8 B=4'b0011;
#8 B=4'b1001;
#8 B=4'b0110;
#18 B=4'b0101;
//Add More test Cases Here
#8 $finish;

end

always #5 clk=~clk;

endmodule
```

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-17-Shift-Registers-II