30 Days of RTL Coding

-By T Tushar Shenoy

Day 16

Problem Statement: Implementing Shift Registers –Part I

1. Serial In Serial Out
2. Serial In Parallel Out

Using Structural Style of Implementation.

Theory:

A group of flip flops which is used to store multiple bits of data and the data is moved from one flip flop to another is known as **Shift Register**. The bits stored in registers shifted when the clock pulse is applied within and inside or outside the registers. To form an n-bit shift register, we have to connect n number of flip flops. So, the number of bits of the binary number is directly proportional to the number of flip flops. The flip flops are connected in such a way that the first flip flop's output becomes the input of the other flip flop.

A **Shift Register** can shift the bits either to the left or to the right. A **Shift Register**, which shifts the bit to the left, is known as **"Shift left register"**, and it shifts the bit to the right, known as **"Right left register"**.

The shift register is classified into the following types:

- Serial In Serial Out
- Serial In Parallel Out
- Parallel In Serial Out
- Parallel In Parallel Out

Serial IN Serial OUT (SISO)

In "Serial Input Serial Output", the data is shifted "IN" or "OUT" serially. In SISO, a single bit is shifted at a time in either right or left direction under clock control.

Initially, all the flip-flops are set in "reset" condition i.e. $Q_3 = Q_2 = Q_1 = Q_0 = 0$. If we pass the binary number 1111, the LSB bit of the number is applied first to the Din bit. The D0 input of the 0th flip flop, i.e., FF-0, is directly connected to the serial data input D0. The output Q_0 is passed to the data input D1 of the next flip flop. This process remains the same for the remaining flip flops. The block diagram of the **"Serial IN Serial OUT"** is given below.

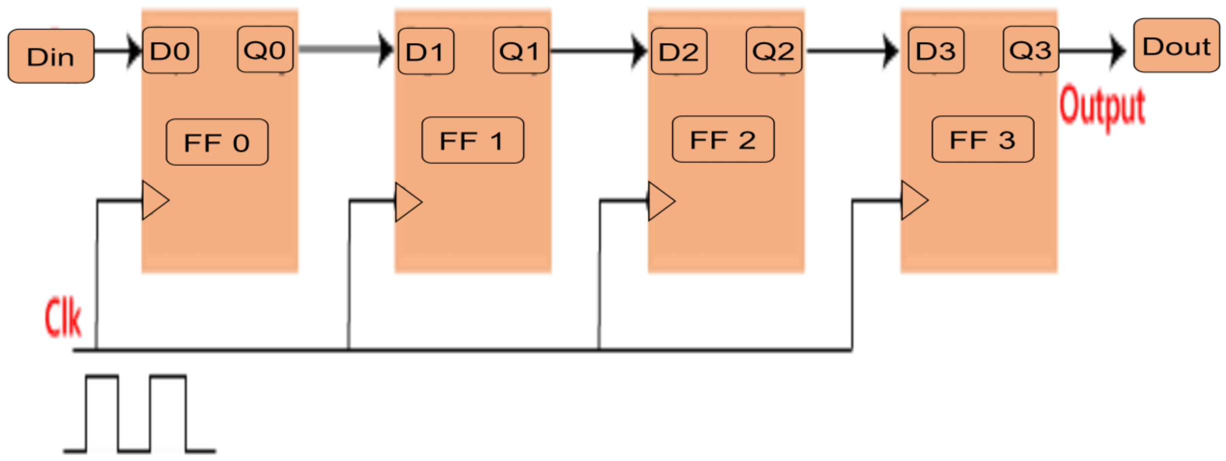


FIG: Serial IN Serial OUT Shift Register

Verilog Code:

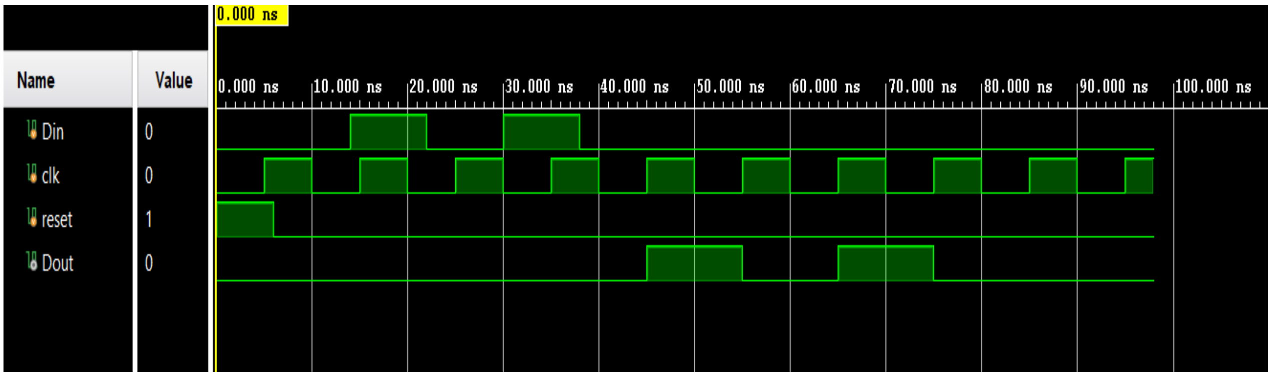
```
//Verilog Code for D Flip Flop
module D_Flip_Flop(D, reset, clk, Q, Qb);
 input D, reset, clk;
 output reg Q, Qb;

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

```
//Verilog Code for Serial in Serial Out Register
module SISO_Shift_Register(Din,clk,reset,Dout);

input Din,clk,reset;
output Dout;
wire [2:0]Q;

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

D_Flip_Flop FF2(.D(Q[0]),.reset(reset),.clk(clk),.Q(Q[1]));

D_Flip_Flop FF3(.D(Q[1]),.reset(reset),.clk(clk),.Q(Q[2]));

D_Flip_Flop FF4(.D(Q[2]),.reset(reset),.clk(clk),.Q(Dout));

endmodule
```

Testbench Code:

```
//Testbench Code for Serial in Serial Out Register
module SISO_Shift_Register_tb();

reg Din,clk,reset;
wire Dout;

SISO_Shift_Register dut(.Din(Din),.clk(clk),.reset(reset),.Dout(Dout));

initial begin
Din=1'b0;
clk=1'b0;
reset=1'b1;
#6  reset=1'b0;
#8  Din=1'b1;
#8  Din=1'b0;
#8  Din=1'b1;
#8  Din=1'b0;
//Add More test Cases Here
#60 $finish;

end

always #5 clk=~clk;

endmodule
```

Simulation Output:



Serial IN Parallel OUT (SIPO)

In the "Serial IN Parallel OUT" shift register, the data is passed serially to the flip flop, and outputs are fetched in a parallel way. The data is passed bit by bit in the register, and the output remains disabled until the data is not passed to the data input. When the data is passed to the register, the outputs are enabled, and the flip flops contain their return value

Below is the block diagram of the 4-bit serial in the parallel-out shift register. The circuit having four D flip-flops contains a clear and clock signal to reset these four flip flops. In SIPO, the input of the second flip flop is the output of the first flip flop, and so on. The same clock signal is applied to each flip flop since the flip flops synchronize each other. The parallel outputs are used for communication.

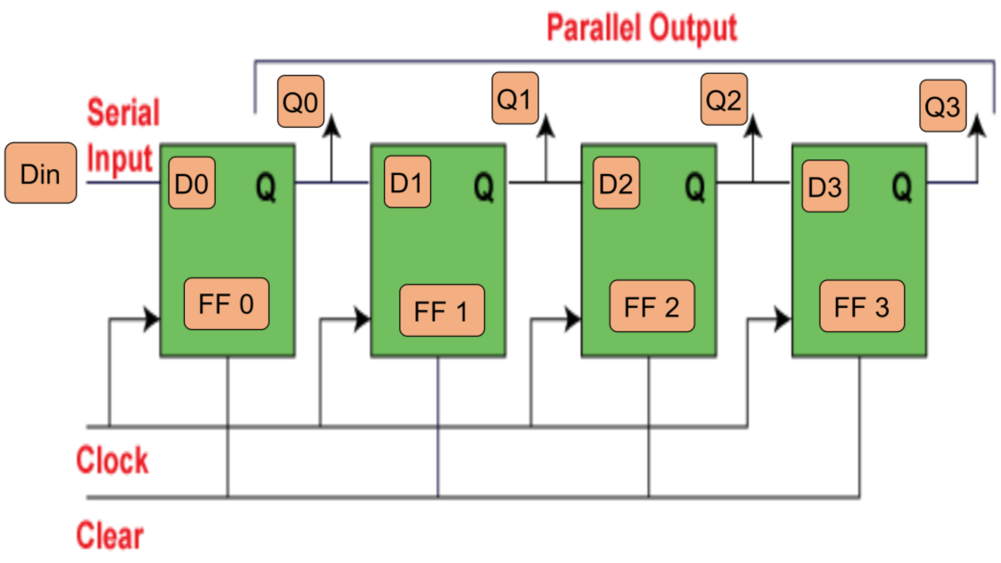


FIG: Serial IN Parallel OUT Shift Register

Verilog Code:

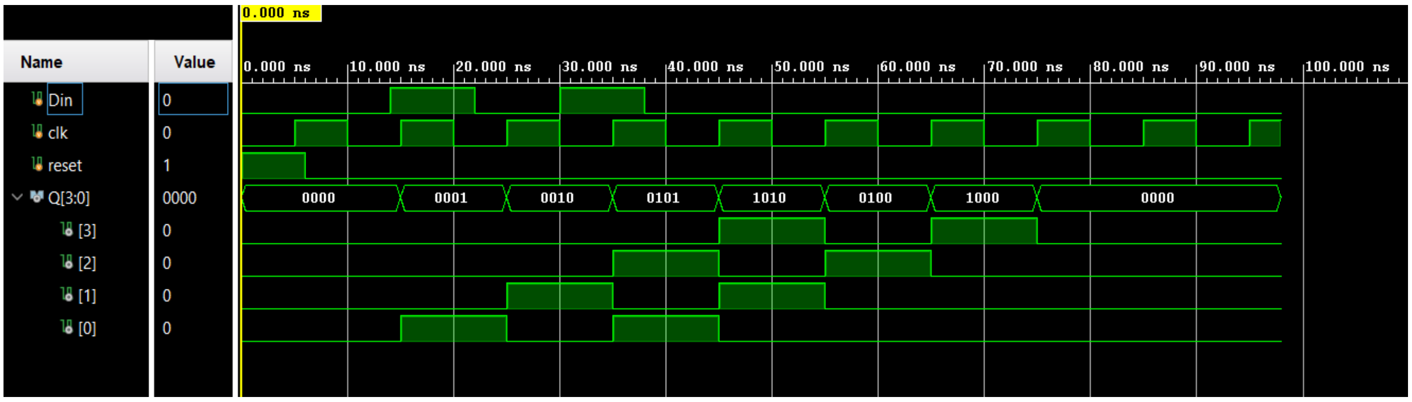
```
//Verilog Code for D Flip Flop
module D_Flip_Flop(D, reset, clk, Q, Qb);
 input D, reset, clk;
 output reg Q, Qb;

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

```
//Verilog Code for Serial in Parallel Out Register
module SIPO_Shift_Register(Din,clk,reset,Q);

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

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

D_Flip_Flop FF2(.D(Q[0]),.reset(reset),.clk(clk),.Q(Q[1]));

D_Flip_Flop FF3(.D(Q[1]),.reset(reset),.clk(clk),.Q(Q[2]));

D_Flip_Flop FF4(.D(Q[2]),.reset(reset),.clk(clk),.Q(Q[3]));

endmodule
```

Testbench Code:

```
//Testbench Code for Serial in Parallel Out Register
module SIPO_Shift_Register_tb();

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

SIPO_Shift_Register dut(.Din(Din),.clk(clk),.reset(reset),.Q(Q));

initial begin
Din=1'b0;
clk=1'b0;
reset=1'b1;
#6  reset=1'b0;
#8  Din=1'b1;
#8  Din=1'b0;
#8  Din=1'b1;
#8  Din=1'b0;
//Add More test Cases Here
#60 $finish;

end

always #5 clk=~clk;

endmodule
```

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-16-Shift-Registers-I