106061151 劉安得

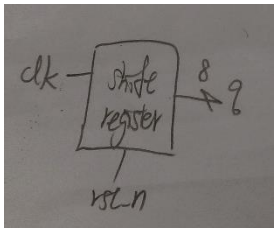
Prelab4

1. **Design Specification**

   8-bits shift register

   Input : clk, rst_n

   Output : q[7:0]



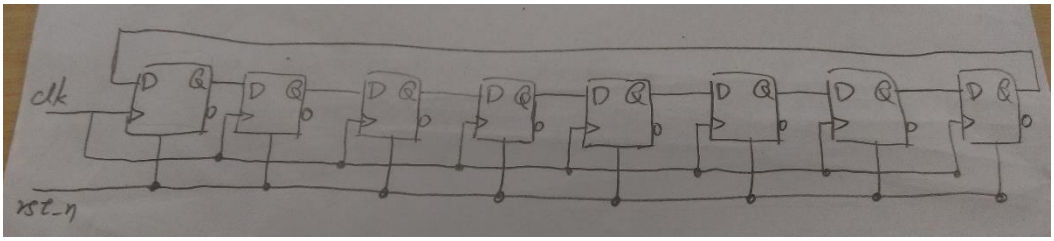
2. **Design implementation**

   Ring shifter

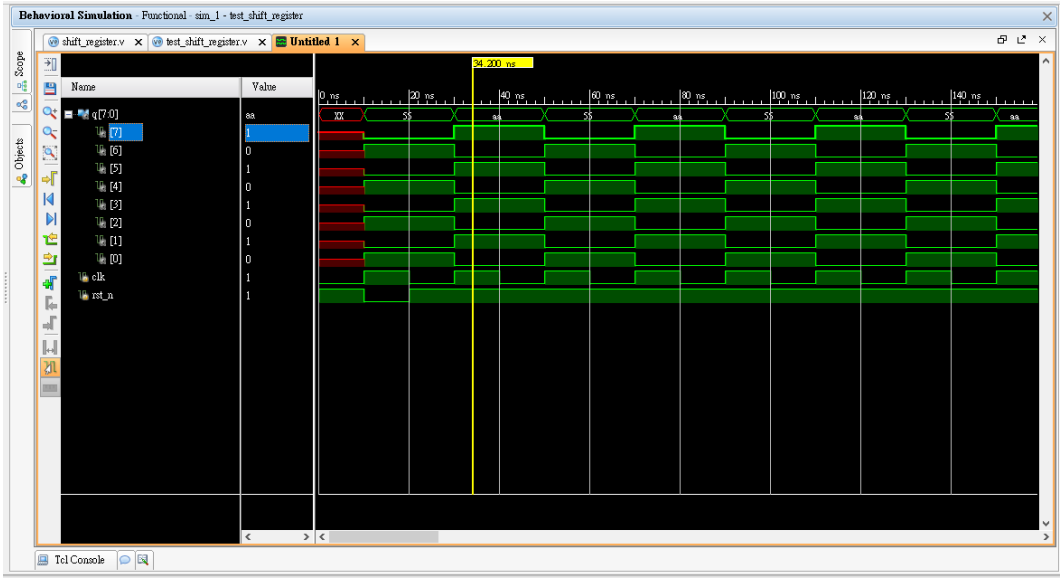
   Connect the out Q to input D

   If(rst_n = 0), q = 01010101

   Else, q shift right 1-bit



3. simutation



01010101 -> 10101010 -> 01010101 repeat

4. **discussion**

看似短短的 verilog code，其實裡面有 implement 到 8 個 DFF。之後應該還要寫個 load 功能，讓他可以從外面賦值，可以暫存外面所存入的值。

5. **conclusion**

做完這次實驗，讓我學會如何 implement 簡單的 shift register

6. **reference**

04_verilog(3) p.12

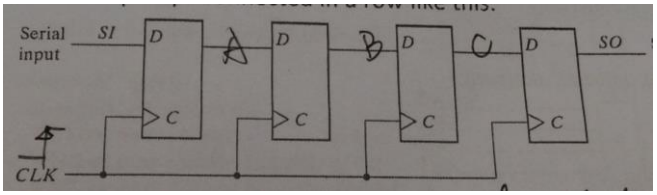
the Verilog of shift register

```
`define BIT_WIDTH 4
module shifter(
  q, // shifter output
  clk, // global clock
  rst_n  // active low reset
);

output [`BIT_WIDTH-1:0] q;  // output
input clk;  // global clock
input rst_n;  // active low reset

reg [`BIT_WIDTH-1:0] q;  // output

// Sequential logics: Flip flops
always @(posedge clk or negedge rst_n)
  if (~rst_n)
  begin
    q<=`BIT_WIDTH'b0101;
  end                initial value 0101
  else
  begin
    q[0]<=q[3];
    q[1]<=q[0];
    q[2]<=q[1];
    q[3]<=q[2];
  end
endmodule
```

Logic Design Lecture 8 Registers and Counters

The schematic of shift register

這個 shift register 只有 4-bit，並且有 serial input



the Verilog of shift register

同樣的，它也只有 4-bits，並且有 sin 這個 serial input。還有一點不同處是，它 reset 完，全部設為 0

```
module Shifter_Register1(clk, rst, sin, state);
 parameter n=4;
 input clk;
 input rst;
 input sin;
 output [n-1:0] state;

 wire [n-1:0] next = rst ? {n{1'b0}} :
 {state[n-2:0], sin};

 DFF #(n) cnt(.clk(clk), .in(next), .out(out));

endmodule
```

sin

```
module Shifter_Register2(clk, rst, sin, state);
 parameter n=4;
 input clk;
 input rst;
 input sin;
 output [n-1:0] state;

 wire [n-1:0] next = {state[n-2:0], sin};

 always @(posedge clk)
  if (rst)
   state <= n'b0;
 else
  state <= next;

endmodule
```

serial In