


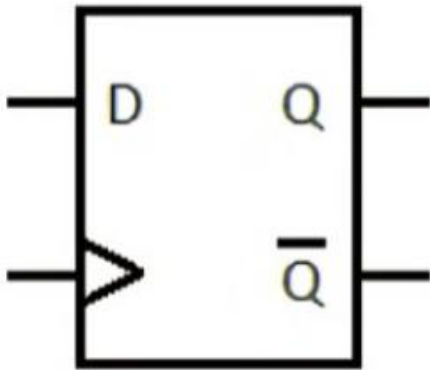
Họ tên sinh viên	MSSV	Lớp (thứ - tiết)	
<b>Hoàng Ngọc Dung</b>	23139006	<b>Thứ 7 - Tiết 7 - 9</b>	

**Chú ý:** Sinh viên thay bằng QR code của mã số sinh viên (ví dụ: 23119012), có thể tham khảo tại <https://barcode.tec-it.com>)

### Quick question : chapter 5

*Lưu ý: Trong mỗi thiết kế yêu cầu sinh viên thực hiện*

- Sơ đồ khối (nguyên lý, cấu trúc)
  - Bảng trạng thái
  - Mô tả bằng ngôn ngữ Verilog cho module cần thiết kế,
  - Mô tả Verilog cho module dùng để kiểm tra thiết kế
  - Kết quả mô phỏng quá trình kiểm tra, có phân tích
  - Module test được đặt tên theo cú pháp: *tensv\_testbench\_tenmodule*, ví dụ để test module *encoder*, sinh viên *Nguyen Van An* phải đặt tên module test như sau:  
*An\_testbench\_encoder*. Các kết quả mô phỏng phải được chụp màn hình bao gồm cả tên của module test trong đó có tên sinh viên thì mới hợp lệ
1. Thiết kế và mô phỏng kiểm chứng mạch Flip Flop D



**Bảng trạng thái  $Q(n+1) = D(n)$**

Clk	D	Q(n+1)
0	x	Q(n)
1	0	0
1	1	1

**Mô tả bằng ngôn ngữ Verilog**

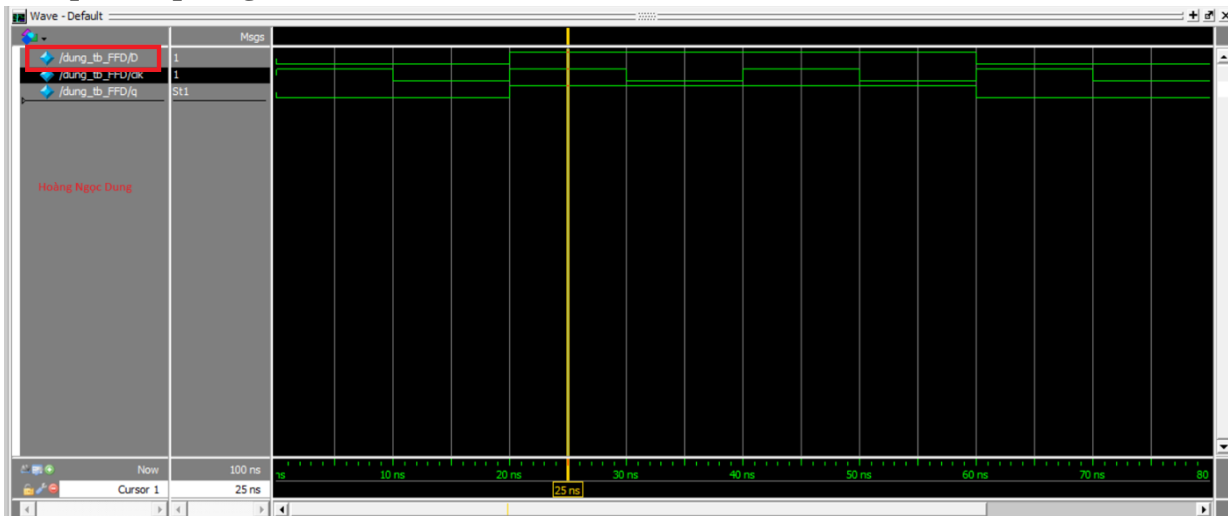
```
module FF_D(input wire d,clk, output reg q);
always @(posedge clk) // positive edge
begin
    q <= d // Non-Blocking Assignment
end
```

```
end  
endmodule
```

## Mô tả Verilog cho module dùng để kiểm tra thiết kế

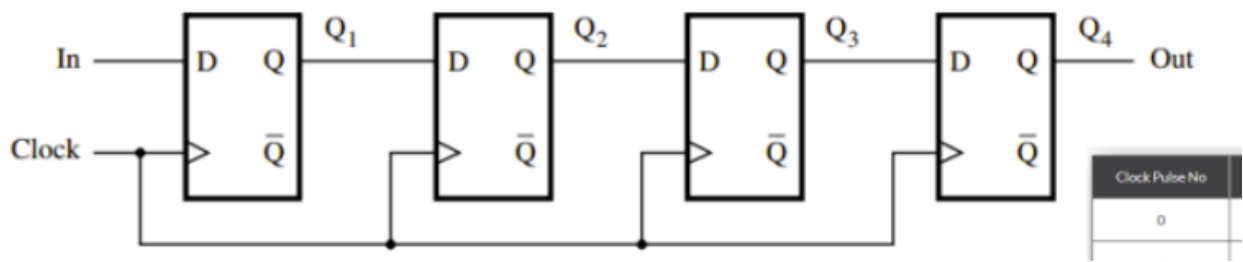
```
`timescale 1ns/1ns // define timescale  
module dung_tb_FFD();  
reg D;  
reg clk;  
wire q;  
  
initial begin  
clk = 1;  
D = 0;  
end  
  
always forever #20 clk = ~clk;  
always forever #40 D = ~D;  
FF_D m0(D,clk,q);  
endmodule
```

## Kết quả mô phỏng



Hình 1: Hình ảnh mô tả kết quả cấp xung đầu vào  $D$  và xung  $Clk$ . Tại thời điểm  $25ns$ ,  $D = 1$ ,  $Clk = 1$  thì  $q = 1$  (kích cạnh lên) trạng thái trước đó  $D = 0$ ,  $clk = 1$  thì  $q = 0 \Rightarrow$  Thiết lập mạch flip flop  $D$  đúng

- Thiết kế và mô phỏng kiểm chứng thanh ghi dịch 8 bit sử dụng FlipFlop D



Hình 2: 4-bit SISO shift register (tương tự cho 8 – bit)

### Bảng trạng thái

Clk Pulse	D	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	x	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0
3	1	1	1	0	0	0	0	0
4	1	1	1	1	0	0	0	0
5	1	1	1	1	1	0	0	0
6	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0
8	1	1	1	1	1	1	1	1

### Mô tả bằng ngôn ngữ Verilog

```

module FF_D(input wire d, clk, output reg q);
    always @(posedge clk)
        q <= d;
endmodule

module SR_8bit(input wire in, clk, output wire[7:0] out);

    FF_D F0 (in, clk, out[0]);
    FF_D F1 (out[0], clk, out[1]);
    FF_D F2 (out[1], clk, out[2]);
    FF_D F3 (out[2], clk, out[3]);
    FF_D F4 (out[3], clk, out[4]);
    FF_D F5 (out[4], clk, out[5]);
    FF_D F6 (out[5], clk, out[6]);
    FF_D F7 (out[6], clk, out[7]);
endmodule

```

### Mô tả Verilog cho module dùng để kiểm tra thiết kế

```

`timescale 1ns/1ns // define timescale
module HoangNgocDung_tb_FFD_SR_8bit();
    reg in;
    reg clk;

```

```

wire [7:0] out;
initial begin
clk = 0;
in = 0;
end
always #10 clk = ~clk;
always #10 in = 1;

SR_8bit m0 (in,clk,out);
endmodule

```

## Kết quả mô phỏng



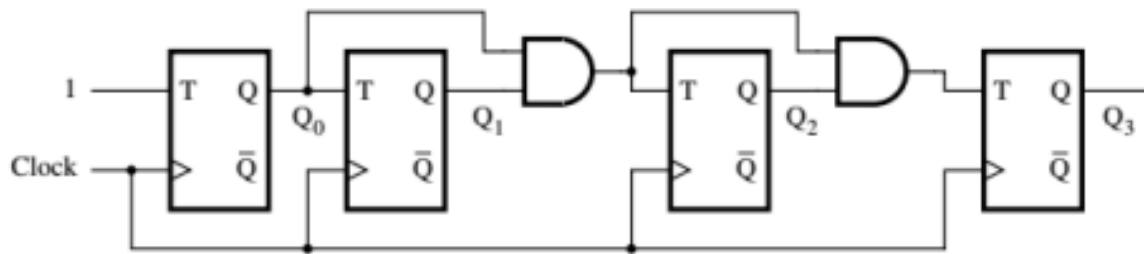
Hình 3: Tín hiệu in duy trì ở mức 1 sau 10ns, nên tại mỗi cạnh lên của clock sau đó, bit '1' sẽ được dịch vào thanh ghi từ vị trí out[0]. Tại cạnh lên clock đầu tiên (khoảng 20ns), out[0] chuyển thành 1.

Tại cạnh lên clock thứ hai (khoảng 40ns), out[0] vẫn là 1 (vì in là 1), và '1' cũ từ out[0] dịch sang out[1], làm cho out[1] cũng thành 1. Bây giờ out là xx...x11. Quá trình này tiếp diễn. Sau mỗi cạnh lên clock, một bit '1' mới được thêm vào out[0] và các bit '1' cũ dịch chuyển sang trái.

Sau 8 cạnh lên của clock (khoảng 160ns), toàn bộ thanh ghi out[7:0] chứa giá trị 8'b11111111 (tất cả các bit đều là 1), thể hiện rằng 8 bit '1' đã được dịch hoàn toàn vào thanh ghi.

Dạng sóng mô phỏng cho thấy module SR\_8bit hoạt động chính xác như một thanh ghi dịch trái (left shift register) 8-bit.

### 3. Thiết kế và mô phỏng kiểm chứng mạch đếm đồng bộ 8 bit sử dụng FlipFlop T



Hình 4: Mạch đếm đồng bộ 8 bit này sẽ sử dụng T-Flip-Flop (T-FF) và sẽ đếm từ 00000000 đến 11111111 (từ 0 đến 255 trong hệ thập phân). Mỗi xung đồng hồ sẽ làm tăng giá trị của bộ đếm lên 1.

Clock Pulse No	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1
6	0	0	0	0	0	1	1	0
7	0	0	0	0	0	1	1	1
8	0	0	0	0	1	0	0	0
9	0	0	0	0	1	0	0	1
10	0	0	0	0	1	0	1	0
11	0	0	0	0	1	0	1	1
12	0	0	0	0	1	1	0	0
13	0	0	0	0	1	1	0	1
14	0	0	0	0	1	1	1	0
15	0	0	0	0	1	1	1	1
16	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...
255	1	1	1	1	1	1	1	1

### Mô tả bằng ngôn ngữ Verilog

```

module t_flip_flop (
    input t, clk,
    output reg q,qb
);
initial
begin
q = 0;
qb = 1;
end

always @(posedge clk)
    if (t) begin

```

```

        q = ~q;
        qb = !qb;
    end
endmodule

module up_down_counter_8bit (
    input clk, up,
    output [7:0] count
);
    wire t0, t1, t2, t3, t4, t5, t6, t7;
    wire q0, q1, q2, q3, q4, q5, q6, q7;

    // T flip-flop logic for the counter
    assign t0 = 1;
    assign t1 = (up) ? q0 : ~q0;
    assign t2 = (up) ? (q0 & q1) : (~q0 & ~q1);
    assign t3 = (up) ? (q0 & q1 & q2) : (~q0 & ~q1 & ~q2);
    assign t4 = (up) ? (q3 & q2 & q1 & q0) : (~q0 & ~q1 & ~q2 & ~q3);
    assign t5 = (up) ? (q4 & q3 & q2 & q1 & q0) : (~q0 & ~q1 & ~q2 & ~q3 &
~q4);
    assign t6 = (up) ? (q5 & q4 & q3 & q2 & q1 & q0) : (~q0 & ~q1 & ~q2 & ~q3 &
~q4 & ~q5);
    assign t7 = (up) ? (q6 & q5 & q4 & q3 & q2 & q1 & q0) : (~q0 & ~q1 & ~q2 & ~q3 &
~q4 & ~q5 & ~q6);

    t_flip_flop ff0 (t0, clk, q0);
    t_flip_flop ff1 (t1, clk, q1);
    t_flip_flop ff2 (t2, clk, q2);
    t_flip_flop ff3 (t3, clk, q3);
    t_flip_flop ff4 (t4, clk, q4);
    t_flip_flop ff5 (t5, clk, q5);
    t_flip_flop ff6 (t6, clk, q6);
    t_flip_flop ff7 (t7, clk, q7);

    assign count = {q7, q6, q5, q4, q3, q2, q1, q0};
endmodule

```

### Mô tả Verilog cho module dùng để kiểm tra thiết kế

```

`timescale 1ns/1ns // define timescale
module HoangNgocDung_tb_up_down_FFT();
    reg clk;
    reg up;
    wire [7:0] count;

    initial begin
        clk = 0;
    end
endmodule

```

```

up = 0;
end

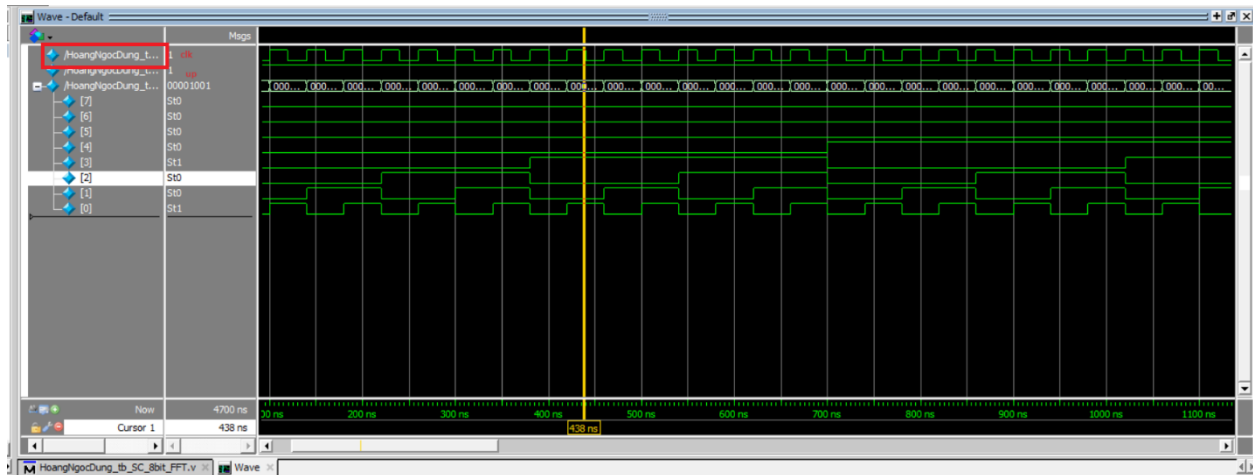
always forever #20 clk = ~clk;
always forever #50 up = 1;

up_down_counter_8bit counter (
    .clk(clk),
    .up(up),
    .count(count)
);

endmodule

```

## Kết quả mô phỏng



Hình 5: Tại thời điểm 50ns, tín hiệu *up* chuyển từ thấp (0) lên cao (1). Điều này báo hiệu cho bộ đếm chuyển sang chế độ đếm lên. Cạnh lên của clock đầu tiên sau khi *up* lên 1 (ví dụ: cạnh lên tiếp theo sau 50ns là tại 60ns), giá trị của bộ đếm bắt đầu tăng lên.

Quan sát dạng sóng, giá trị của *count* tăng lên từng đơn vị tại mỗi cạnh lên của clock: Tại ~60ns: *count* chuyển từ 00000000 lên 00000001 (1). Tại ~100ns: *count* chuyển từ 00000001 lên 00000010 (2). Tại ~140ns: *count* chuyển từ 00000010 lên 00000011 (3).

Kết quả hiển thị là bộ đếm đang đếm lên từ 00000000, tăng giá trị của nó theo từng xung clock khi *up* ở mức cao. Dạng sóng cho thấy bộ đếm hoạt động đúng với chức năng đếm lên đồng bộ 8-bit