DATE

Experiment No-10
Design of Counters using Xilinx Vivado Tool

(i) Asynchronous Counters.

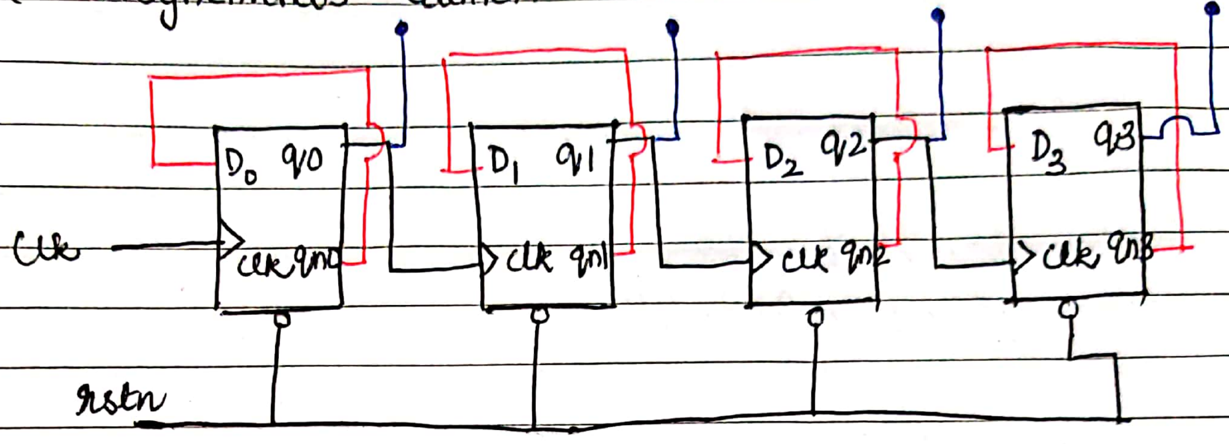
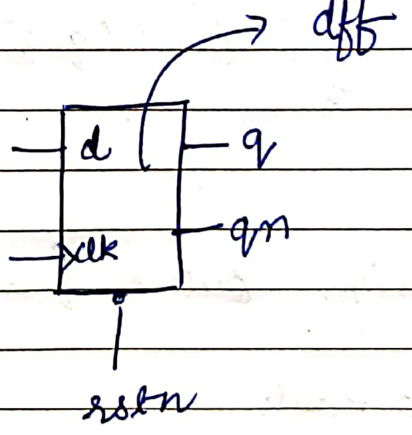


Fig: Asynchronous Down Counter.

Verilog Code (Structural Style)

```
module dff (d, clk, rstn, q, qn);
input d, clk, rstn;
output q, qn;
 reg q, qn;
always @ (posedge clk or negedge rstn)
  begin
     if (! rstn)
       q <= 0
     else
       q <= d;
   end
  assign qn = ~q;
 endmodule
```



DATE

Verilog code for asynchronous Counter

```
module asynchronous (clk, rstn, out);
input clk, rstn;
Output [3:0] out;

wire q0, q1, q2, q3;
wire qn0, qn1, qn2, qn3;

dff ff0 (.d(qn0), .clk(clk), .rstn(rstn), .q(q0),
                           .qn (qn0));

dff ff1 (.d(qn1), .clk(q0), .rstn(rstn), .q(q1),
                           .qn(qn1));

dff ff2 (.d(qn2), .clk(q1), .rstn(rstn), .q(q2),
                           .  qn(qn2));

dff ff3 (.d(qn3), .clk(q2), .rstn(rstn), .q(q3),
                           . qn (qn3));

assign out = { q3, q2, q1, q0 };

endmodule
```

DATE

Test Bench

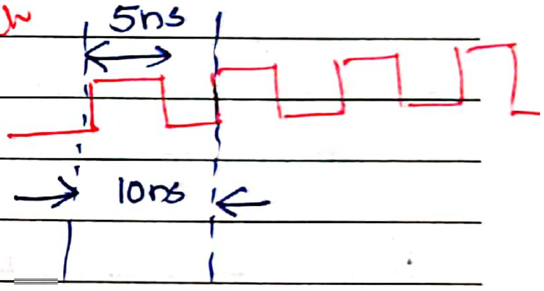
```
module asynchronous_tb;
reg clk, rstn;
wire [3:0] out;

asynchronous ~~a0~~ dut ( clk, rstn, out);
initial
begin
        rstn = 0;
        clk  = 0;
#20  rstn = 1;
end

always
   #5 clk = ~clk;

endmodule.
```

Generating a clock with a period of 10 ns

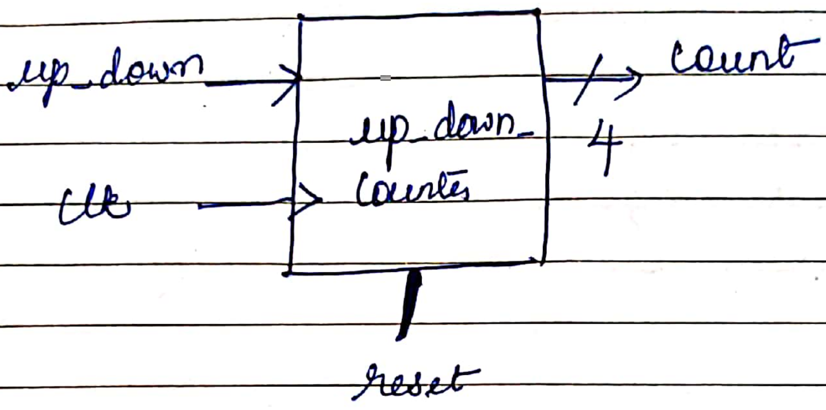


DATE

(ii) Synchronous Counters

⟶ [up down counter]

⟶ Behavioural Style



Verilog Code for Synchronous up-down counter

```
module up_down_counter (up_down, clk, reset,
                        Count);
input up_down, clk, reset;
Output [3:0] Count;
reg    [3:0] count_up_down;
always@ (posedge clk or posedge reset)
begin
    if (reset)
        Count_up_down <= 4'h0;
```

Am registering a temporary variable

DATE

```
else if (up_down)
    count_up_down <= count_up_down + 4'd1;
else
    count_up_down <= count_up_down ~~+~~ 4'd1;
end
    assign count = count_up_down;
endmodule
```

Test Bench code

```
module up_down_counter_tb;
  reg clk, reset, up_down;
  wire [3:0] count;

up_down_counter dut( up_down, clk, reset,
                     count);

 initial
  begin
   clk = 0;
   forever #5 clk = ~clk;
  end
```

DATE

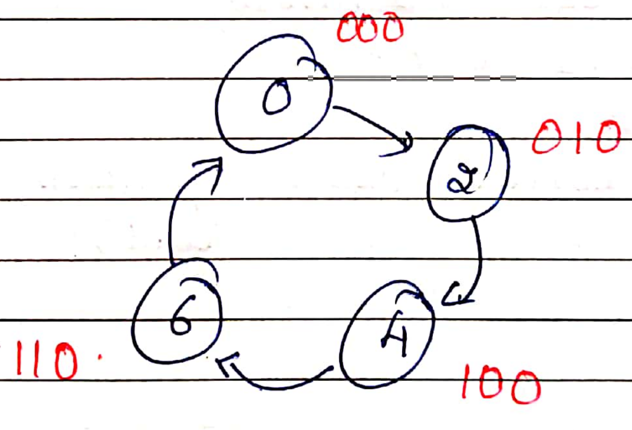
```
initial
 begin

   reset = 1;
   up_down = 0;
#20  reset = 0;
#200  up_down = 1;
  end
endmodule
```

3 Any sequence counter

→ To count the sequence 0,2,4,6.

1 State diagram



I will choose D-ff for the design of any sequence counter.

State table

Present state $q_2\ q_1\ q_0$	Next state $q_2\ q_1\ q_0$	flip flop Input equations $d_0\ d_1\ d_0$
0 0 0	0 1 0	0 1 0
0 0 1		
0 1 0	1 0 0	1 0 0
0 1 1		
1 0 0	1 1 0	1 1 0
1 0 1		
1 1 0	0 0 0	0 0 0
1 1 1		

~~d_0~~

$q_2 \backslash q_1 q_0$	00	01	11	10
0	0	X	X	1
1	1	X	X	0

$\rightarrow \bar{q}_2 q_1$

$q_2 \bar{q}_1$

$= \bar{q}_2 q_1 + q_2 \bar{q}_1 = \boxed{q_2 \oplus q_1}$

$d_0 = q_2 \wedge q_1 \rightarrow$ equ $\textcircled{1}$

d_1

$q_2 \backslash q_1 q_0$	00	01	11	10
0	1	X	X	0
1	1	X	X	0

~~$\bar{q}_1 \bar{q}_0$~~ $\quad \bar{q}_1$

~~$d_1 = \sim(q_1 \vee q_0)$~~ $\rightarrow$ eq $\textcircled{2}$

$\bar{q}_1$

$\boxed{d_1 = \sim q_1 \rightarrow \text{equ} \textcircled{2}}$

d_2

$q_2 \backslash q_1 q_2$	00	01	11	10
0	0	X	X	0
1	0	X	X	0

$d_2 = 0 \rightarrow$ eq $\textcircled{3}$

DATE

Verilog code for anysequence counter

```
module anysequencecounter ( q, rst, clk );
 input clk, rst;
 Output [2:0] q;
 reg q;

  wire [2:0] d;
  assign d[0] = q[2] ^ q[1];
  assign d[1] = ~~~( q[1] & q[2]);~~ ~q[1];
  assign d[2] = 1'b0;

  always @ ( posedge clk or negedge rst)
  begin
       if (! rst)

        q <= 1'b0;

        else

        q <= d;

       end
  endmodule
```