گزارشکار آزمایش چهارم درس آزمایشگاه معماری

اعضای گروه:

حوریه سبزواری، عرشیا آرین‌نژاد، الناز رضایی

هدف آزمایش:

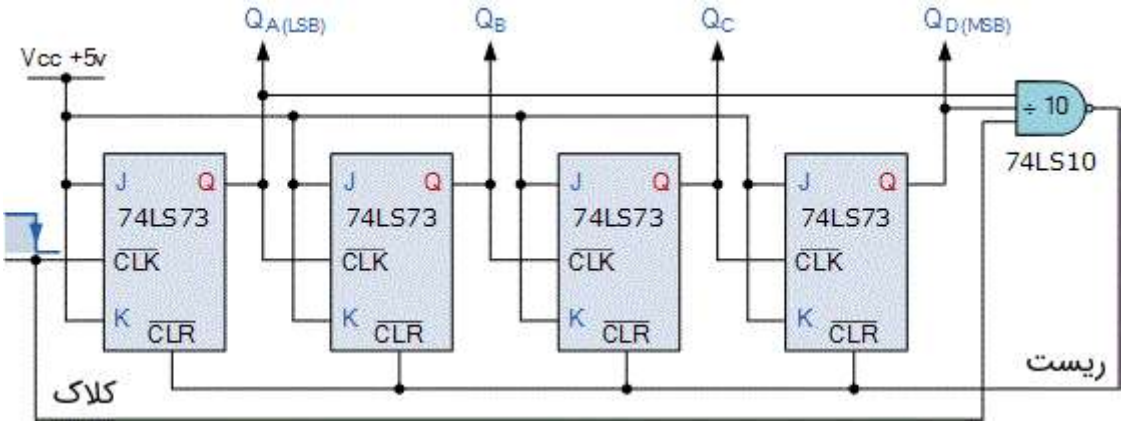
شمارنده آسنکرون (Asynchronous Counter) از فلیپ فلاپ‌هایی که به صورت سری به یکدیگر متصل شده‌اند، بهره می‌گیرد تا بتواند پالس کلاک ورودی را به صورت موجی در طول شمارنده حرکت دهد. در این آزمایش به بیان اصول کاری شمارنده‌های آسنکرون می‌پردازیم.

گر یک شمارنده آسنکرون با MOD ۱۶ را انتخاب کنیم و آن را با گیت‌های منطقی اضافی اصلاح کنیم، می‌توانیم به خروجی شمارنده دهدهی (مدار مقسم بر ده) دست یابیم و از آن در شمارنده‌های دسیمال استاندارد و مدارات حسابی (Arithmetic) استفاده کنیم.

این شمارنده‌ها معمولا تحت عنوان شمارنده‌های ده دهی (Decade Counter) شناخته می‌شوند. در یک شمارنده ده دهی نیاز است تا زمانی که شمارش به ضرایب ده برسد، خروجی به مقدار صفر ریست شود. به عبارت دیگر خروجی باید برابر با DCBA=1010 باشد. برای رسیدن به این هدف، باید وقوع این حالت را به ورودی ریست مدار فیدبک دهیم. یک شمارنده با رشته شمارش از ۰۰۰۰ باینری (BCD = 0)تا ۱۰۰۱(BCD=9) ، شمارنده BCD یا شمارنده دسیمال کد باینری (Binary Coded Decimal) نامیده می‌شود؛ زیرا دنباله ده حالتی خروجی آن همان کد BCD است. اما شمارنده‌های ده دهی باینری بسیار متداول‌تر هستند.

یک مثال بسیار خوب از شمارنده MOD-m که از مدار ترکیبی خارجی برای تولید یک شمارنده MOD-10 استفاده می‌کند، شمارنده ده دهی است. شمارنده ده دهی (مقسم بر ۱۰) مانند TTL 74LS90 دارای ده حالت در دنباله شمارش خود است و در نمایش‌گرهای ارقام به عنوان رابط کاربری بسیار کاربرد دارد.

شمارنده ده دهی دارای ۴ خروجی است و می‌تواند اعداد ۴ بیتی باینری تولید کند. همچنین با استفاده از گیت‌های AND و OR می‌توان وقوع شمارش ۹ را تشخیص داد و شمارنده را به مقدار ۰ ریست کرد. همانند سایر شمارنده‌هایMOD ، این شمارنده نیز پالس‌های کلاک را یکی پس از دیگری دریافت می‌کند و از ۰ تا ۹ شمارش را انجام می‌دهد. زمانی که شمارش به ۹ دسیمال برسد، شمارنده به جای اینکه ۱۰ دسیمال یا ۱۰۱۰ باینری را بشمارد، به مقدار ۰ ریست می‌شود.



شمارنده‌ی بالاشمار دهدهی آسنکرون با استفاده از T_FlipFlop

کد vhdl :

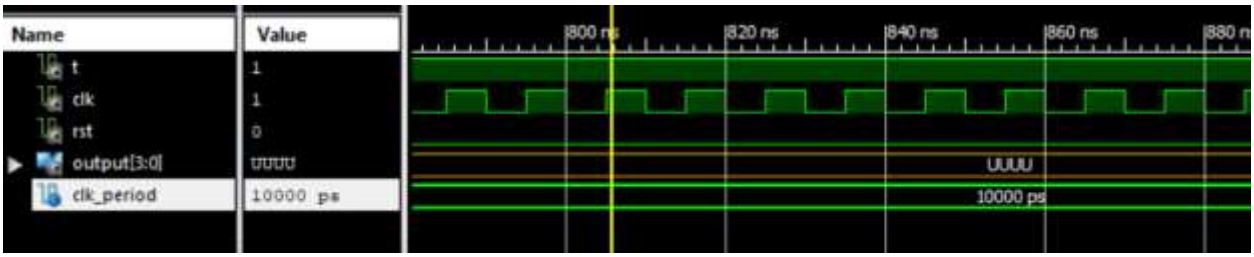
```
32  entity counter is
33     port(
34       T: in std_logic;
35       clk: in std_logic;
36       rst: inout std_logic;
37       output: out std_logic_vector(3 downto 0)
38     );
39  end counter;
40
41  architecture Behavioral of counter is
42
43  component T_FF is
44    port(
45     T, reset, clk: in std_logic;
46      Q: inout std_logic
47    );
48  end component T_FF;
49
50  component NAND1 is
51    port(
52     A, B: in std_logic;
53      C :out std_logic
54    );
55  end component NAND1;
56
57  signal Q0, Q1, Q2, Q3: std_logic;
58  signal res : std_logic;
59
60  begin
```

```
61
62  T0: T_FF port map(T => T, reset => res, clk => not clk, Q => Q0);
63  T1: T_FF port map(T => T, reset => res, clk => not Q0, Q => Q1);
64  T2: T_FF port map(T => T, reset => res, clk => not Q1, Q => Q2);
65  T3: T_FF port map(T => T, reset => res, clk => not Q2, Q => Q3);
66  n1 : nand1 port map (A => Q1, B => Q3, C => res);
67  output(0) <= Q0;
68  output(1) <= Q1;
69  output(2) <= Q2;
70  output(3) <= Q3;
71
72   end Behavioral;
```

```
33  entity T_FF is
34    port(
35     T, reset, clk: in std_logic;
36      Q: inout std_logic
37    );
38  end entity T_FF;
39
40  architecture Behavioral of T_FF is
41    begin
42    process (reset, clk) is begin
43        if reset = '1' then
44         Q <= '0';
45        elsif(rising_edge(clk)) then
46           Q <= Q xor T;
47        end if;
48    end process;
```

```
34
35  ENTITY counter_test IS
36  END counter_test;
37
38  ARCHITECTURE behavior OF counter_test IS
39
40      -- Component Declaration for the Unit Under Test (UUT)
41
42      COMPONENT counter_async
43      PORT(
44           T : IN  std_logic;
45           clk : IN  std_logic;
46           rst : INout  std_logic;
47           output : OUT  std_logic_vector(3 downto 0)
48          );
49      END COMPONENT;
50
51
52     --Inputs
53     signal T : std_logic := '1';
54     signal clk : std_logic := '0';
55     signal rst : std_logic := '0';
56
57     --Outputs
58     signal output : std_logic_vector(3 downto 0);
59
60     -- Clock period definitions
61     constant clk_period : time := 10 ns;
62
63  BEGIN
```

```
64
65     -- Instantiate the Unit Under Test (UUT)
66     uut: counter_async PORT MAP (
67            T => T,
68            clk => clk,
69            rst => rst,
70            output => output
71          );
72
73     -- Clock process definitions
74     clk_process :process
75     begin
76        clk <= '0';
77        wait for clk_period/2;
78        clk <= '1';
79        wait for clk_period/2;
80     end process;
81
82
83     -- Stimulus process
84     stim_proc: process
85     begin
86        -- hold reset state for 100 ns.
87        wait for 100 ns;
88
89        rst <= '1' after 50 ns, '0' after 100 ns;
90
91        wait for clk_period*10;
92
93        -- insert stimulus here
94
95        wait;
96     end process;
```



یک شمارنده‌ی پایین‌شمار دودویی ۴بیتی:

کد vhdl :

```
entity counter2 is
   port(
     T: in std_logic;
     clk: in std_logic;
     rst: in std_logic;
     output: out std_logic_vector(3 downto 0)
   );
end counter2;

architecture Behavioral of counter2 is

component T_FF is
  port(
   T, reset, clk: in std_logic;
    Q: inout std_logic
  );
end component T_FF;

signal Q0, Q1, Q2, Q3: std_logic;

begin
T0: T_FF port map(T => T, reset => rst, clk => clk, Q => Q0);
T1: T_FF port map(T => T, reset => rst, clk => Q0, Q => Q1);
T2: T_FF port map(T => T, reset => rst, clk => Q1, Q => Q2);
T3: T_FF port map(T => T, reset => rst, clk => Q2, Q => Q3);

output(0) <= Q0;
output(1) <= Q1;
output(2) <= Q2;
output(3) <= Q3;
```

```
64
65      -- Instantiate the Unit Under Test (UUT)
66      uut: counter_async_decreasing PORT MAP (
67             T => T,
68             clk => clk,
69             rst => rst,
70             output => output
71           );
72
73      -- Clock process definitions
74      clk_process :process
75      begin
76         clk <= '0';
77         wait for clk_period/2;
78         clk <= '1';
79         wait for clk_period/2;
80      end process;
81
82
83      -- Stimulus process
84      stim_proc: process
85      begin
86         -- hold reset state for 100 ns.
87         wait for 100 ns;
88
89         rst <= '1' after 100 ns, '0' after 200 ns;
90
91         wait for clk_period*10;
92
93         -- insert stimulus here
94
95         wait;
96      end process;
97
98  END;
```