به نام یکتای در کمال

دانشکده مهندسی کامپیوتر

گزارش‌کار سوم – آزمایشگاه معماری کامپیوتر

استاد : سرکار خانم دکتر محبتی

امیرحسین ملکوتی – محمدرضا صاحب‌زاده – عرفان زارع

آذر ۱۴۰۱

در این سری از آزمایش ها ما ابتدا به سراغ ساخت Flip-flop های پایه در مدارات منطقی رفته‌ایم.

همانگونه که در دروس مدارات منطقی و معماری کامپیوتر آموختیم، فلیپ فلاپ‌ها یک سری المان ذخیره کننده هستند که در CLK ورودی در صورت تغییر در داده‌ها، داده ها را تغییر می‌دهند و در غیر اینصورت مقدار بیت را در خود نگه‌داری می‌کنند.

با استفاده از جدول فوق ما خواص ۳ مورد از فلیپ فلاپ ها را داریم.

فلیپ فلاپ D

D	Q(t+1)	
0	0	بازنشانی
1	1	نشاندن

فلیپ فلاپ JK

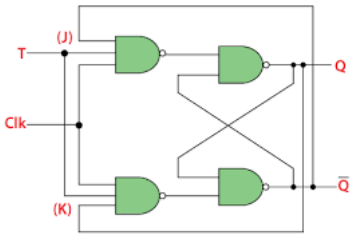
J	K	Q(T+1)	
0	0	Q(t)	بلا تغییر
0	1	0	بازنشانی
1	0	1	نشاندن
1	1	Q´(t)	متمم

فلیپ فلاپ T

T	Q(t+1)	
0	Q(t)	بلا تغییر
1	Q´(t)	متمم

حال با استفاده از این جدول و شکل مدارات متناظر با هر کدام کد VHDL را خواهیم نوشت.

فلیپ فلاپ T :

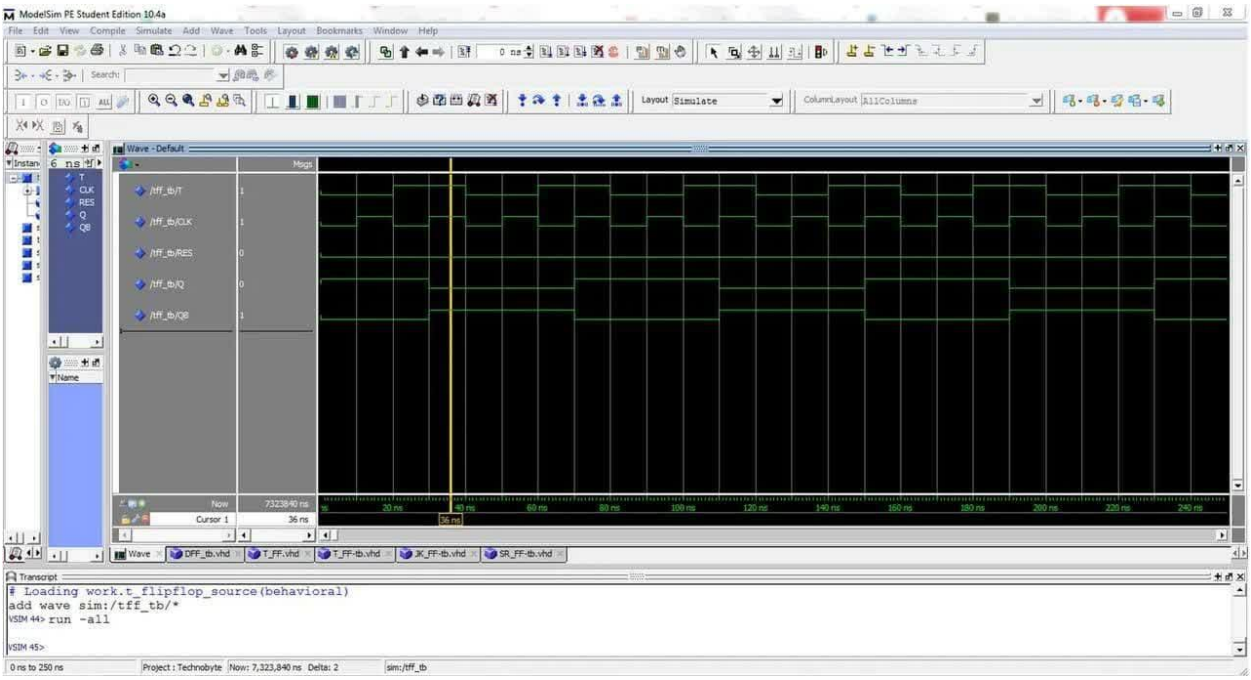


همانطور که مشاهده کردیم این فلیپ فلاپ در اصل از مدل JK ساخته می‌شود.

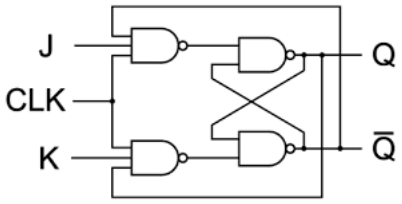
در ادامه کد VHDL و سیگنال خروجی این فلیپ فلاپ را خواهیم داشت:

```
   library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity T_FLIPFLOP_SOURCE is
   Port ( T,CLK,RES,TEMP : in  STD_LOGIC;
          Q,QB : out STD_LOGIC);
end T_FLIPFLOP_SOURCE;

architecture Behavioral of T_FLIPFLOP_SOURCE is

begin

PROCESS(T,CLK,RES)

VARIABLE TEMP:STD_LOGIC:='0';

BEGIN

IF(RES='1')THEN
TEMP:='0';
ELSIF(RISING_EDGE(CLK))THEN
IF(T='1')THEN
TEMP:= NOT TEMP;

END IF;
END IF;
Q<= NOT TEMP;
QB<= TEMP;

END PROCESS;
END BEHAVIORAL;
```

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity TFF_tb is
7  end entity;
8
9  architecture tb of TFF_tb is
10 component T_FLIPFLOP_SOURCE is
11 Port ( T,CLK,RES : in STD_LOGIC;
12 Q,QB : out STD_LOGIC);
13 end component;
14
15 signal T,CLK,RES,Q,QB : STD_LOGIC;
16
17 begin
18 uut: T_FLIPFLOP_SOURCE port map(
19 T => T,
20 CLK => CLK,
21 RES => RES,
22 Q => Q,
23 QB => QB);
24
25 clock : process
26 begin
27
28 CLK <= '0';
29 wait for 10 ns;
30 CLK <= '1';
31 wait for 10 ns;
32
33 end process;
34
35 stim: process
36 begin
37
38 RES <= '0';
39
40 T <= '0';
41 wait for 20 ns;
42
43 T <= '1';
44 wait for 20 ns;
45
46 end process;
47 end tb;
```



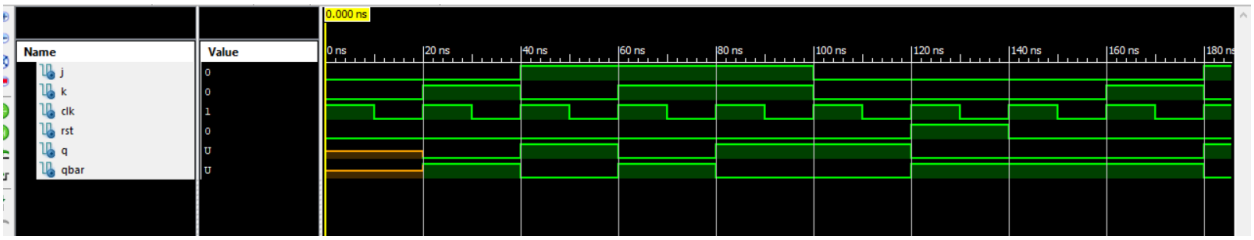
فلیپ فلاپ JK :



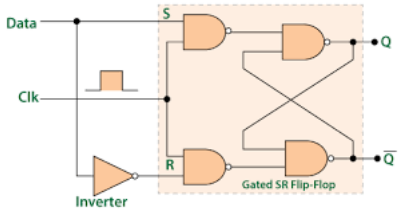
در ادامه کد VHDL و سیگنال خروجی این فلیپ فلاپ را خواهیم داشت:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity JK_FF is
7     port( J, K, clk, rst : in std_logic;
8        Q, Qbar : out std_logic);
9  end JK_FF;
10
11 architecture behavioral of JK_FF is
12 begin
13 process(clk, rst)
14 variable qn : std_logic;
15 begin
16 if(rst = '1')then
17 qn := '0';
18 elsif(clk'event and clk = '1')then
19 if(J='0' and K='0')then
20 qn := qn;
21 elsif(J='0' and K='1')then
22 qn := '0';
23 elsif(J='1' and K='0')then
24 qn := '1';
25 elsif(J='1' and K='1')then
26 qn := not qn;
27 else
28 null;
29 end if;
30 else
31 null;
32 end if;
33 Q <= qn;
34 Qbar <= not qn;
35
36 end process;
37 end behavioral;
```

```
library ieee;
use ieee.std_logic_1164.all;

entity JK_FF_tb is
end JK_FF_tb;

architecture testbench of JK_FF_tb is

component JK_FF is
port(J, K, clk, rst : in std_logic;
Q, Qbar : out std_logic
);
end component;

signal J, K, clk, rst : std_logic;
signal Q, Qbar : std_logic;

begin
uut: JK_FF port map(
J => J,
K => K,
clk => clk,
rst => rst,
Q => Q,
Qbar => Qbar);

clock: process
begin
clk <= '1';
wait for 10 ns;
clk <= '0';
wait for 10 ns;
end process;

Force: process
begin
J <= '0';
K <= '0';
rst <= '0';
wait for 20 ns;

J <= '0';
K <= '1';
rst <= '0';
wait for 20 ns;

J <= '1';
K <= '0';
rst <= '0';
```



فلیپ فلاپ D :

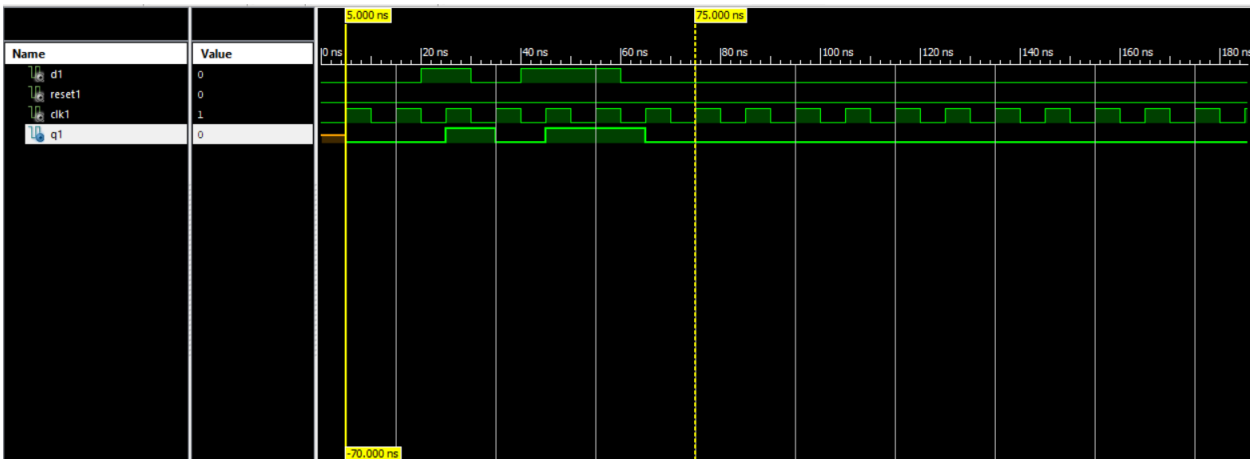


همانطور که در این فلیپ فلاپ مشاهده می‌کنیم، این فلیپ فلاپ هم از JK می‌تواند ساخته شود.

در ادامه کد VHDL و سیگنال خروجی این فلیپ فلاپ را خواهیم داشت:

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5
6  entity dff is
7  port(
8     d , reset , clk:in std_logic;
9     q                :out std_logic
10    );
11 end dff;
12
13 architecture dff_behav of dff is
14
15 begin
16 q<= '0' when reset = '1' else
17    d when clk'event and clk  = '1';
18
19 end dff_behav;
20
21
```

```
ENTITY dff_testbench IS
END dff_testbench;

ARCHITECTURE behavior OF dff_testbench IS

    COMPONENT dff
    PORT(
      d , reset , clk:in std_logic;
      q              :out std_logic
        );
    END COMPONENT;


   --constant <clock>_period : time := 10 ns;
   signal d1 , reset1 , clk1 :std_logic := '0';
   signal q1 :std_logic :='0';
BEGIN
   uut: dff PORT MAP (d1 , reset1 , clk1 , q1
        );

   -- Clock process definitions
   --<clock>_process :process
   --begin
      --<clock> <= '0';
      --wait for <clock>_period/2;
      --<clock> <= '1';
      --wait for <clock>_period/2;
   --end process;


   -- Stimulus process
   --stim_proc: process
   --begin
      -- hold reset state for 100 ns.
     -- wait for 100 ns;

      --wait for <clock>_period*10;
   clk1 <= not clk1 after 5 ns;
   d1<= '1' after 20 ns , '0' after 30 ns , '1' after 40 ns,  '1' after 50 ns ,'0' after 60 ns ;

      -- insert stimulus here

      --wait;
```



در سری بعدی آزمایش ما سراغ یک شمارنده با استفاده از فلیپ فلاپ می‌رویم.

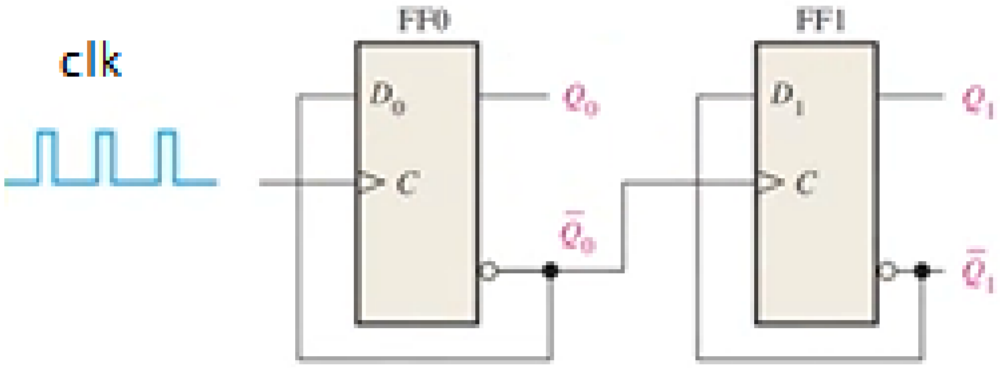
شمارنده ها سری‌ای از فلیپ فلاپ ها هستند که اولا به صورت ورودی سریال (Serial) و خروجی موازی (Parallel) هستند و معمولا با هر CLK مقدار آن‌ها تغییر می‌کند.

از شمارنده ها ما در تابلو امتیازات و ساعت و ... استفاده می‌کنیم.

نکته‌ی جالب در خصوص طراحی شمارنده ها آزادی عمل در خصوص استفاده از فلیپ فلاپ‌ها خواهیم داشت. این نکته‌ی به شدت ساده هنگام طراحی مدارات با اندازه‌های بزرگ‌تر کمک بسیار بزرگی به ما خواهد کرد.

در ابتدا ما شمارنده آسنکرون ۲ بیتی طراحی می‌کنیم. این شمارنده مقادیر ۰۰ تا ۱۱ را می‌شمارد و بعد دوباره از ۱۱ به ۰۰ برمی‌گردد.

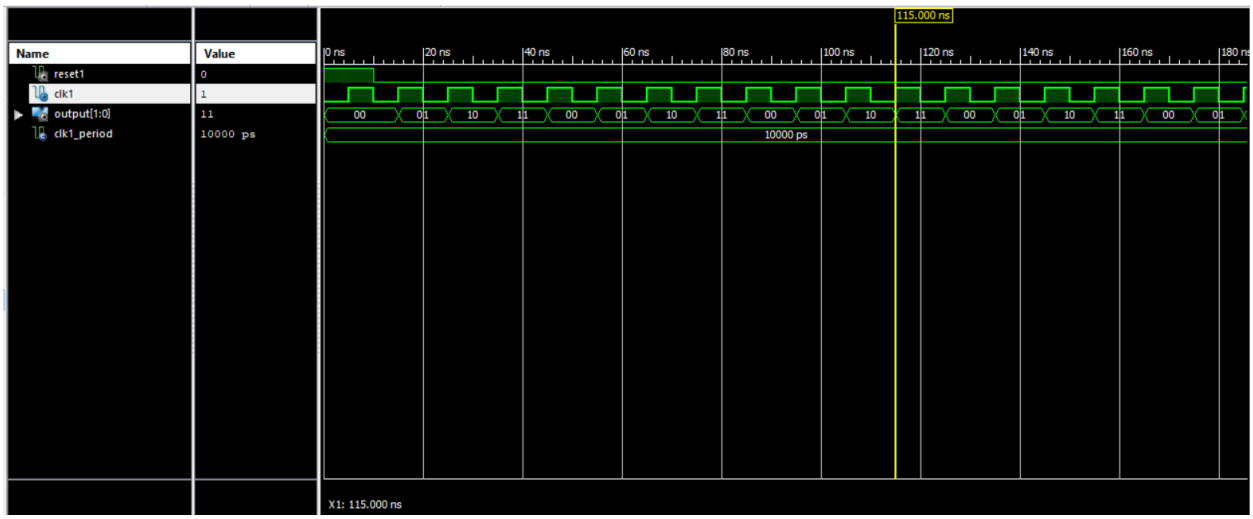
نمای این شمارنده به صورت زیر است :



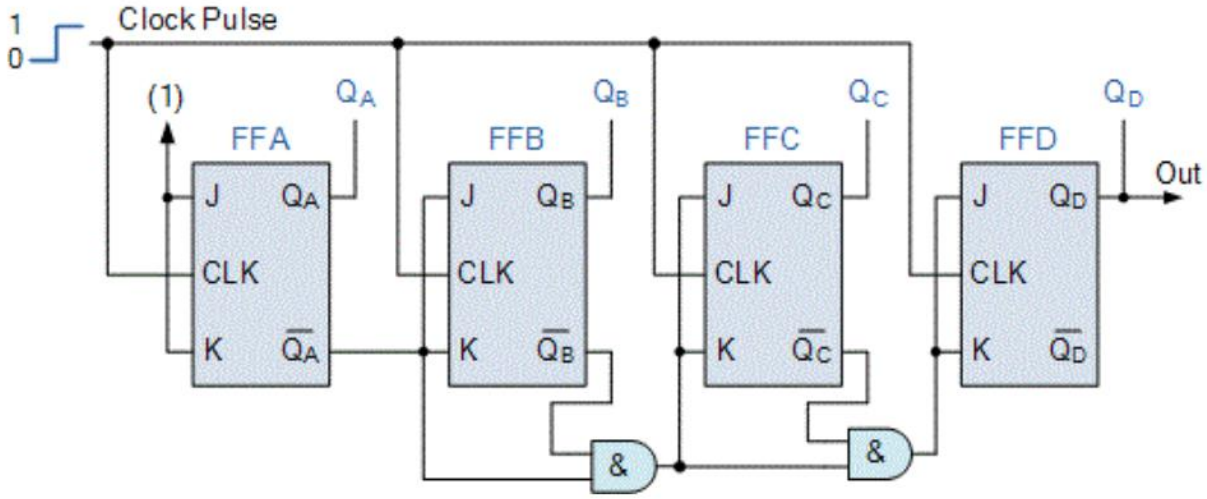
در ادامه کد VHDL و سیگنال خروجی شمارنده ۲ بیتی آسنکرون را خواهیم داشت:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity async_2bit_counter is
   port(
      reset1  : inout std_logic:='1';
      clk1 : in std_logic;
      output :out std_logic_vector(1 downto 0)
      );
end async_2bit_counter;

architecture Behavioral of async_2bit_counter is
component dff is
port(
   d , reset , clk:in std_logic;
   q ,q_prime :out std_logic
   );
end component;
   signal temp1 , out1:std_logic:= '1';
   signal temp2 , out2:std_logic:= '1';

begin
   reset1 <= '0' after 10 ns;

      dff_back : dff port map (d => temp1 ,reset =>reset1 , clk => clk1 , q_prime => out1);
      output(0) <= not out1;
      temp1 <= out1 after 2 ns;
      dff_forward : dff port map (d => temp2 , reset => reset1 , clk => out1 , q_prime => out2);
      output(1)<= not out2;
      temp2 <= out2 after 2 ns;

end Behavioral;
```

```
19          output : OUT  std_logic_vector(1 downto 0)
20        );
21     END COMPONENT;
22
23
24     --Inputs
25     signal reset1 : std_logic := '0';
26     signal clk1 : std_logic := '0';
27
28     --Outputs
29     signal output : std_logic_vector(1 downto 0);
30
31     -- Clock period definitions
32     constant clk1_period : time := 10 ns;
33
34  BEGIN
35
36     -- Instantiate the Unit Under Test (UUT)
37     uut: async_2bit_counter PORT MAP (
38            reset1 => reset1,
39            clk1 => clk1,
40            output => output
41          );
42
43     -- Clock process definitions
44     clk1_process :process
45     begin
46        clk1 <= '0';
47        wait for clk1_period/2;
48        clk1 <= '1';
49        wait for clk1_period/2;
50     end process;
51
52
53     -- Stimulus process
54     stim_proc: process
55     begin
56        -- hold reset state for 100 ns.
57        wait for 100 ns;
58
59        wait for clk1_period*10;
60
61        -- insert stimulus here
62
63        wait;
64     end process;
65
66  END;
67
```



سپس در ادامه ما شمارنده پایین شمار سنکرون دودویی ۴ بیتی را خواهیم داشت.

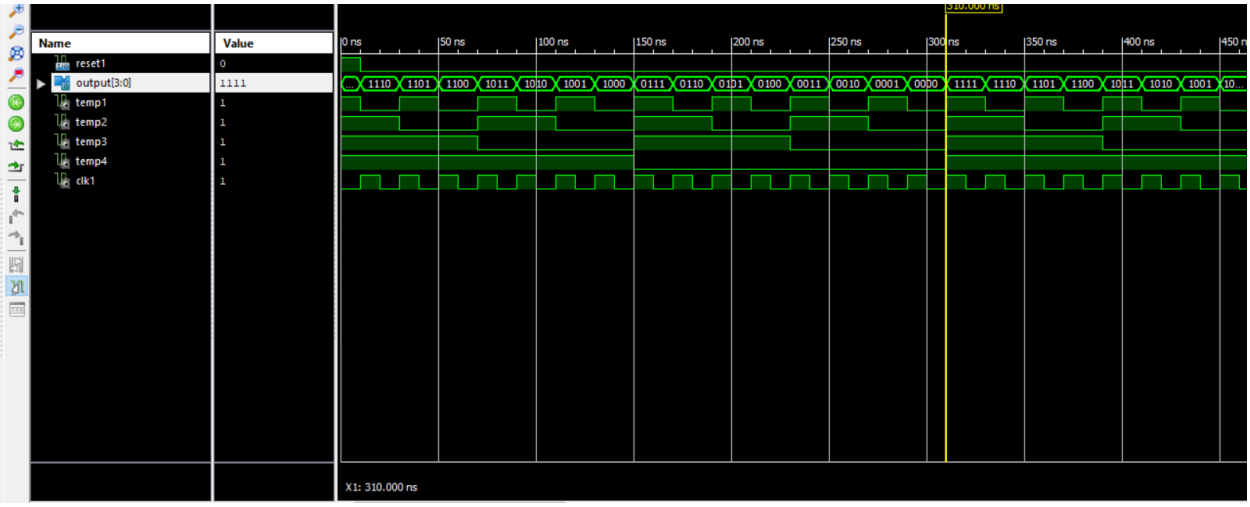


شمارنده بالا یک شمارنده پایین‌شمار سنکرون است به این معنی که اولا مقادیر تمامی فلیپ فلاپ ها با یکدیگر تغییر می‌کند و ثانیا از مقدار ۱۱۱۱ به مقدار ۰۰۰۰ شمارش می‌کند و بعد دوباره به ۱۱۱۱ پرش می‌کند.

در ادامه ما شاهد کد VHDL و سیگنال این شمارنده خواهیم بود:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity async_4bit_down_counter is
   port(
   clk1  : in std_logic;
   reset1: inout std_logic:='1';
   output:out std_logic_vector(3 downto 0)
   );
end async_4bit_down_counter;

architecture Behavioral of async_4bit_down_counter is
   component dff is
   port(
      d , reset , clk:in std_logic;
      q ,q_prime :out std_logic
      );
   end component;
   signal temp1 , temp2 :std_logic :='1';
   signal temp3 , temp4:std_logic :='1';

begin
   reset1 <= '0' after 10 ns;
   dff_1 : dff port map (d => temp1 ,reset =>reset1 , clk => clk1 , q_prime => temp1);
      output(0) <= temp1;
   dff_2 : dff port map (d => temp2 ,reset =>reset1 , clk => temp1 , q_prime => temp2);
      output(1) <= temp2;
   dff_3 : dff port map (d => temp3 ,reset =>reset1 , clk => temp2 , q_prime => temp3);
      output(2) <= temp3;
   dff_4 : dff port map (d => temp4 ,reset =>reset1 , clk => temp3 , q_prime => temp4);
      output(3) <= temp4;
end Behavioral;

```

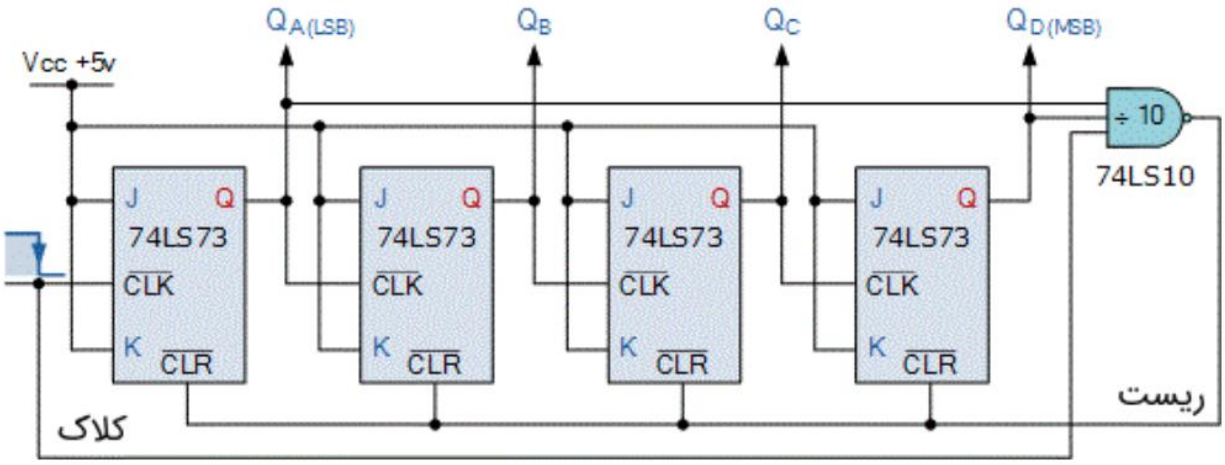
```
ARCHITECTURE behavior OF async_4bit_down_counter_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT async_4bit_down_counter
    PORT(
         clk1 : IN  std_logic;
         reset1 : INOUT  std_logic;
         output : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal clk1 : std_logic := '0';

   --BiDirs
   signal reset1 : std_logic;

   --Outputs
   signal output : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk1_period : time := 10 ns;

BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: async_4bit_down_counter PORT MAP (
          clk1 => clk1,
          reset1 => reset1,
          output => output
        );

   -- Clock process definitions
   clk1_process :process
   begin
      clk1 <= '0';
      wait for clk1_period/2;
      clk1 <= '1';
      wait for clk1_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
```



به عنوان آخرین مورد از پیاده‌سازی ها برای این گزارش‌کار ما شاهد یک شمارنده بالاشمار دهدهی آسنکرون خواهیم بود. از این مدل شمارنده ها در محاسبات Decimal استفاده می‌کنیم و مدارات حساب معمول را با استفاده از این مدل طراحی خواهیم کرد.

در این شمارنده ها ما از ۰۰۰۰ شروع به محاسبه می‌کنیم و سپس تا ۱۰۰۱ یا ۹ دسیمال ادامه می‌دهیم. زمانی که شمارنده‌ی ما به ۱۰۱۰ رسید باید ریست فعال شود. این مکانیزم به شکل خیلی ساده با یک گیت And متصل به ریست می‌تواند پیاده‌سازی شود در این حالت خروجی کم ارزش ترین بیت و پر ارزش‌ترین بیت با یکدیگر به گیت And دو ورودی وصل می‌شود و با فعال شدن هر دو گیت ریست روشن شده و خروجی بعد از ۹، صفر می‌شود.

شماتیک این شمارنده به شکل زیر خواهد بود:



در ادامه کد VHDL این مدل شمارنده به همراه سیگنال‌های آن را خواهیم دید: