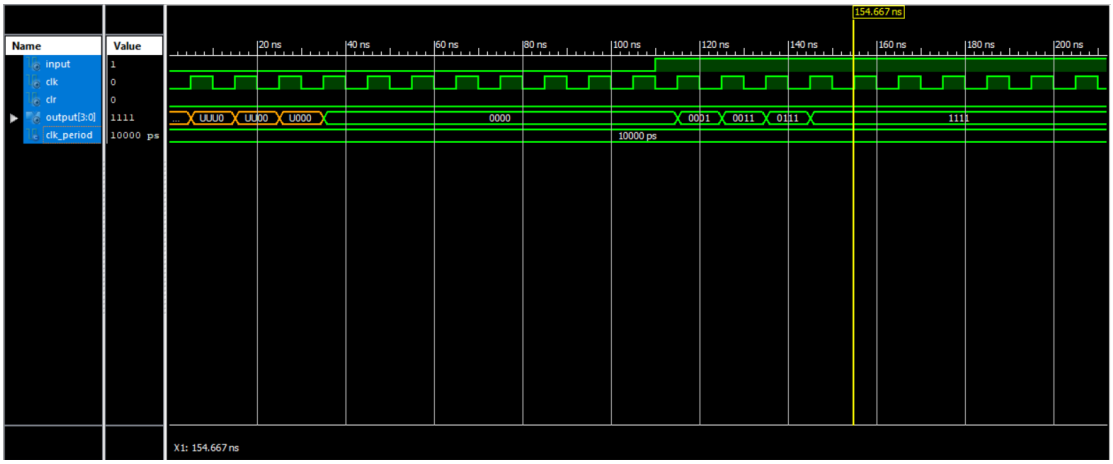
به نام خدا

محمدرضا صاحب زاده
عرفان زارع
امیررضا ملکوتی فر

SIPO:

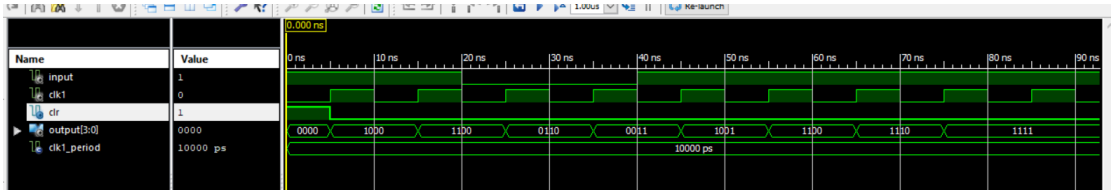



```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shift_register_4bit is
   generic (n: integer :=4);
   port(
      input :in std_logic := '0';
      clk ,clr :in std_logic;
      output : out std_logic_vector( n-1 downto 0)
      );
end shift_register_4bit;

architecture Behavioral of shift_register_4bit is
component dff is
port(
   d , reset , clk:in std_logic;
   q ,q_prime:out std_logic
   );
end component;
begin
   process (clk)
   variable t_out : std_logic_vector(n-1 downto 0);
   begin
      if(clr = '1')then
         output <= "0000";
      elsif (clk = '1' )then--TODO
         t_out := t_out(n-2 downto 0) & input;
      end if;
      output <= t_out;
   end process;


end Behavioral;
```

Shift_register_test_bench:

```
21
22      --Outputs
23      signal output : std_logic_vector(3 downto 0);
24
25      -- Clock period definitions
26      constant clk_period : time := 10 ns;
27
28  BEGIN
29
30      -- Instantiate the Unit Under Test (UUT)
31      uut: shift_register_4bit PORT MAP (
32             input => input,
33             clk => clk,
34             clr => clr,
35             output => output
36           );
37
38      -- Clock process definitions
39      clk_process :process
40      begin
41         clk <= '0';
42         wait for clk_period/2;
43         clk <= '1';
44         wait for clk_period/2;
45      end process;
46
47
48      -- Stimulus process
49      stim_proc: process
50      begin
51
52         -- hold reset state for 100 ns.
53         wait for 10 ns;
54
55         wait for clk_period*10;
56         input <= '1';
57         -- insert stimulus here
58
59         wait;
60      end process;
61
62  END;
63
```

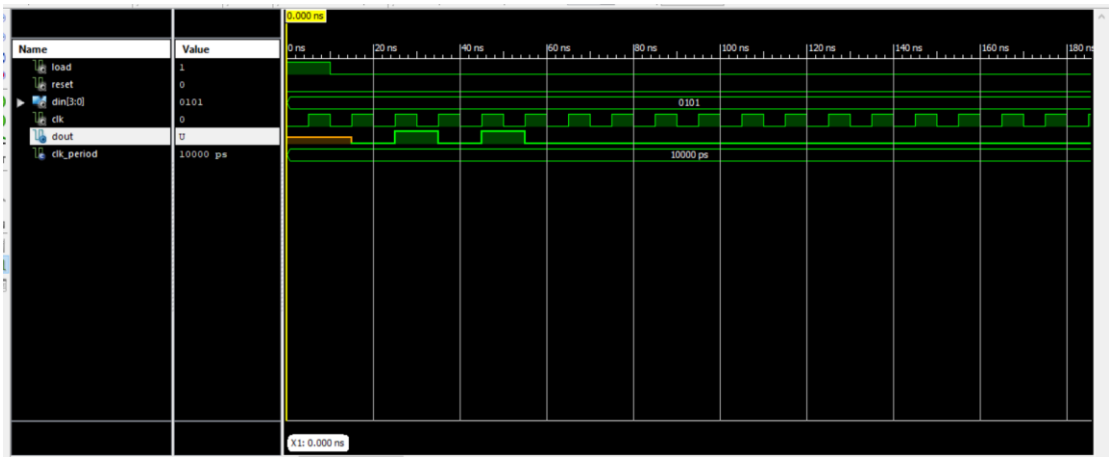
SISO:



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

entity SISO is
   --generic (n: integer :=4);
   port(
      input :in std_logic ;--:='1';
      clk1 :in std_logic;
      clr :inout std_logic:='1';
      output : out std_logic_vector(3 downto 0)
      );
end SISO;

architecture Behavioral of SISO is

    component dff is
   port(
      d , reset , clk:in std_logic;
      q ,q_prime:out std_logic
   );
    END COMPONENT;
    signal t1 ,tt1 , t2 ,tt2 , t3 , tt3 ,t4 , tt4:std_logic:='0';
    --signal clk : std_logic:='0';
begin
   --clk <= not clk after 10 ns;

   dff1 : dff port map(input , clr , clk1 , tt1, open);
   t1<= tt1 after 5 ns;
   dff2 : dff port map(t1   , clr , clk1 , tt2, open);
   t2 <= tt2 after 5 ns;
   dff3 : dff port map(t2 , clr , clk1 , tt3, open);
   t3 <= tt3 after 5 ns;
   dff4 : dff port map(t3 , clr , clk1 , tt4, open);

   output <= tt1&tt2&tt3&tt4;
   clr <= '0' after 5 ns;
end Behavioral;
```

```
31      -- Clock period definitions
32      constant clk1_period : time := 10 ns;
33
34  BEGIN
35
36      -- Instantiate the Unit Under Test (UUT)
37      uut: SISO PORT MAP (
38             input => input,
39             clk1 => clk1,
40             clr => clr,
41             output => output
42           );
43
44      -- Clock process definitions
45      clk1_process :process
46      begin
47         clk1 <= '0';
48         wait for clk1_period/2;
49         clk1 <= '1';
50         wait for clk1_period/2;
51      end process;
52
53
54      -- Stimulus process
55      stim_proc: process
56      begin
57         -- hold reset state for 100 ns.
58         input <= '1';
59         wait for 10 ns;
60         input<= '0' after 10 ns;
61         wait for 10 ns;
62         input<= '1' after 20 ns;
63
64         wait for clk1_period*10;
65
66         -- insert stimulus here
67
68         wait;
69      end process;
70
71  END;
```

PISO:



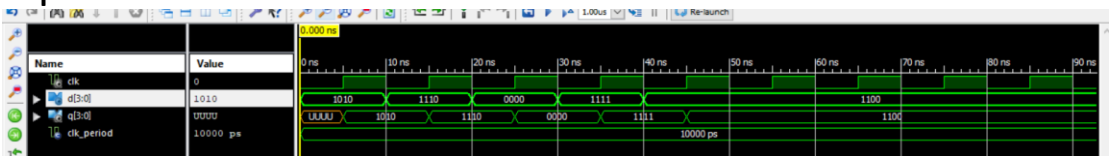
Test bench:

```
47          );
48
49      -- Clock process definitions
50      clk_process :process
51      begin
52          clk <= '0';
53          wait for clk_period/2;
54          clk <= '1';
55          wait for clk_period/2;
56      end process;
57
58
59      -- Stimulus process
60      stim_proc: process
61      begin
62          din<= "0101";
63          load<= '1';
64
65
66          wait for 10 ns;
67          load <= '0';
68          --din<= '0';
69          --wait for 10 ns;
70          --din<= '1';
71
72          wait for clk_period*10;
73
74          -- insert stimulus here
75
76          wait;
77      end process;
78
```

Code:

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

entity piso is
Port ( load : in STD_LOGIC;
reset : in STD_LOGIC;
din : in STD_LOGIC_VECTOR(3 downto 0);
clk : in STD_LOGIC;
dout : out STD_LOGIC);
end piso;
architecture Behavioral of piso is
begin
process (clk,reset,load,din) is
variable temp : std_logic_vector (din'range);
begin
if (reset='1') then
temp := (others=>'0');
elsif (load='1') then
temp := din ;
elsif (clk'event and clk='1') then
dout <= temp(3);
temp := temp(2 downto 0) & '0';
end if;
end process;
end Behavioral;
```

Pipo:



Test bench:

```
32    -- Instantiate the Unit Under Test (UUT)
33    uut: PIPO_shift_register PORT MAP (
34           clk => clk,
35           D => D,
36           Q => Q
37         );
38
39    -- Clock process definitions
40    clk_process :process
41    begin
42       clk <= '0';
43       wait for clk_period/2;
44       clk <= '1';
45       wait for clk_period/2;
46    end process;
47
48
49    -- Stimulus process
50    stim_proc: process
51    begin
52       D<="1010";
53       wait for 10 ns;
54       D<="1110";
55       wait for 10 ns;
56       D<="0000";
57       wait for 10 ns;
58       D<="1111";
59       wait for 10 ns;
60       D<="1100";
61
62       wait for clk_period*10;
63
```

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity PIPO_shift_register is
5      generic(n : positive := 4);
6      port(
7          clk : in std_logic;
8          D : in std_logic_vector(n-1 downto 0); -- D == input
9          Q : out std_logic_vector(n-1 downto 0) -- Q == output
10 );
11 end PIPO_shift_register;
12
13 architecture Behavioral of PIPO_shift_register is
14
15 begin
16
17     process (clk)
18         variable temp_out : std_logic_vector(n-1 downto 0);
19     begin
20
21         if (clk'event and clk='1') then
22             temp_out := D; -- new inputs replaced with an old one.
23         end if;
24         Q <= temp_out;
25     end process;
26
27 end Behavioral;
```

کاربرد ثبات ها در کامپیوتر:

میتوان گفت ثباتها مهمترین و کاربردی ترین حافظه در امر پردازش هستند؛ زیرا بسیاری از دستورات نمیتوانند به صورت مستقیم توسط واحد پردازشی CPU پردازش شده و خروجی را تحویل دهند. این دستورات نیازمند پردازش دستور یا دستوراتی دیگر قبل از ارائه خروجی هستند. در این حالت، CPU با استفاده از واحد اسمبلر (Unit Assambler ،به معنای واحد تبدیل کننده به اسمبلی) دستور اصلی را به مجموعهای از دستورات تبدیل میکند. مجموعه دستورات بدست آمده،
دستورات فرعی نام دارند.
نکته: در برنامههای نوشته شده به زبان اسمبلی، برنامه نویس میتواند به صورت مستقیم به ثباتهای عمومی دسترسی داشته باشد. بنابراین واحد اسمبلر از پروسه پردازش کنار گذاشته میشود. تنها در زبان سطح پایین اسمبلی میتوان به رجیسترها به صورت مستقیم دسترسی داشت و در سایر زبانهای سطح بال، کامپایلرها و واحد اسمبلر این وظیفه را بر عهده دارند.در ادامه پروسه پردازش، پردازنده هر یک از دستورات فرعی را با استفاده از واحد ALU) مخفف Logic & Arithmetic Unit و به معنای واحد محاسبه منطق) پردازش کرده و نتیجه خروجی را در یکی از ثباتهای خود ذخیره میکند. هر یک از دستورات فرعی ممکن است بر خروجی دستور فرعی دیگر تاثیر بگذارد. به همین دلیل، ثباتها برخلف سایر حافظهها به صورت منطقی (Logic (کار میکنند؛ یعنی یک ثبات میتواند بر ثبات دیگر اثر بگذارد. بنابراین ثباتها وظیفه دارند تا خروجیهای هر یک از دستورات فرعی را درخود (به صورت موقت) ذخیره کنند