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

entity accumulator is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        addr  : in  std_logic_vector(2 downto 0);
        din   : in  std_logic;
        dout  : out std_logic_vector(15 downto 0)
    );
end accumulator;

architecture behavioral of accumulator is

    signal shift_reg : std_logic_vector(6 downto 0);
    signal data_cnt  : unsigned(2 downto 0);

    -- Simple FSM: '1' - operating, '0' - idle
    signal state     : std_logic;
    signal load      : std_logic;
    signal reset_s   : std_logic;

    signal acc       : unsigned(15 downto 0);

begin

    -- Input shift register
    process(clk,reset)
    begin
        if reset = '1' then
            shift_reg <= (others => '0');
        elsif clk'event and clk = '1' then
            shift_reg <= shift_reg(5 downto 0) & din;
        end if;
    end process;

    -- Input data counter
    process(clk,reset)
    begin
        if reset = '1' then
            data_cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            if state = '0' then
                data_cnt <= (others => '0');
            else
                data_cnt <= data_cnt + 1;
            end if;
        end if;
    end process;

    -- Control logic (simple FSM)
    process(clk,reset)
    begin
        if reset = '1' then
            state <= '0';
        elsif clk'event and clk = '1' then
            if state = '0' and din = '1' then
                state <= '1';
            elsif state = '1' and data_cnt = 7 then
```

```
62                  state <= '0';
63              end if;
64          end if;
65      end process;
66
67      load    <= '1' when shift_reg(6 downto 4) = addr and data_cnt = 7 else '0';
68      reset_s <= '1' when shift_reg(6 downto 4) = "111" and data_cnt = 7 else '0';
69
70      -- Accumulator logic
71      process(clk,reset)
72      begin
73          if reset = '1' then
74              acc <= (others => '0');
75          elsif clk'event and clk = '1' then
76              if reset_s = '1' then
77                  acc <= (others => '0');
78              else
79                  if load = '1' then
80                      acc <= acc + unsigned(shift_reg(3 downto 0));
81                  end if;
82              end if;
83          end if;
84      end process;
85
86      -- Output connection
87      dout <= std_logic_vector(acc);
88
89  end behavioral;
90
```

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

entity top is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        e     : in  std_logic;
        s     : out std_logic_vector(15 downto 0);
        t     : out std_logic_vector(15 downto 0)
    );
end top;

architecture structural of top is

    signal addr_a : std_logic_vector(2 downto 0);
    signal addr_b : std_logic_vector(2 downto 0);

begin

    addr_a <= "110";
    addr_b <= "011";

    bloque_a: entity work.accumulator
    port map(
        clk   => clk,
        reset => reset,
        addr  => addr_a,
        din   => e,
        dout  => s
    );

    bloque_b: entity work.accumulator
    port map(
        clk   => clk,
        reset => reset,
        addr  => addr_b,
        din   => e,
        dout  => t
    );

end structural;

```

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity accumulator_tb is
 5  end accumulator_tb;
 6
 7  architecture testbench of accumulator_tb is
 8
 9      signal clk   : std_logic;
10      signal reset : std_logic;
11      signal addr  : std_logic_vector(2 downto 0);
12      signal din   : std_logic;
13      signal dout  : std_logic_vector(15 downto 0);
14
15      signal aux   : std_logic_vector(6 downto 0);
16
17      constant clk_period : time := 10 ns;
18
19  begin
20
21      uut: entity work.accumulator
22      port map(
23          clk   => clk,
24          reset => reset,
25          addr  => addr,
26          din   => din,
27          dout  => dout
28      );
29
30      proc_clk: process
31      begin
32          clk <= '1';
33          wait for clk_period/2;
34          clk <= '0';
35          wait for clk_period/2;
36      end process;
37
38      proc_stim: process
39      begin
40          reset <= '1';
41          addr <= "000";
42          aux <= "0000000";
43          din <= '0';
44          wait for clk_period*100.5;
45
46          reset <= '0';
47          wait for clk_period*100;
48
49          addr <= "011";
50          aux <= "0111010";
51
52          din <= '1';
53          wait for clk_period;
54          for i in 6 downto 0 loop
55              din <= aux(i);
56              wait for clk_period;
57          end loop;
58          din <= '0';
59          wait for clk_period*100;
60
61          addr <= "110";
```

```
62            aux <= "1100011";
63
64            din <= '1';
65            wait for clk_period;
66            for i in 6 downto 0 loop
67                din <= aux(i);
68                wait for clk_period;
69            end loop;
70            din <= '0';
71            wait for clk_period*100;
72
73            addr <= "110";
74            aux <= "1111111";
75
76            din <= '1';
77            wait for clk_period;
78            for i in 6 downto 0 loop
79                din <= aux(i);
80                wait for clk_period;
81            end loop;
82            din <= '0';
83            wait for clk_period*100;
84
85            wait;
86        end process;
87
88    end testbench;
89
```

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

entity lfsr is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        dout  : out std_logic
    );
end lfsr;

architecture behavioral of lfsr is

    constant C_DEPTH : integer := 3; -- In b is 65
    constant C_TAP_1 : integer := 3; -- in b is 65
    constant C_TAP_2 : integer := 2; -- In b is 47

    signal shift_register : std_logic_vector(C_DEPTH-1 downto 0);
    signal din            : std_logic;

begin

    process(clk,reset)
    begin
        if reset = '1' then
            shift_register <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            shift_register <= shift_register(C_DEPTH-2 downto 0) & din;
        end if;
    end process;

    din  <= shift_register(C_TAP_1-1) xor shift_register(C_TAP_2-1);
    dout <= shift_register(C_DEPTH-1);

end behavioral;
```

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

entity random is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        dout  : out std_logic
    );
end random;

architecture behavioral of random is

    constant C_DEPTH : integer := 3; -- In b is 65
    constant C_TAP_1 : integer := 3; -- in b is 65
    constant C_TAP_2 : integer := 2; -- In b is 47

    signal shift_register : std_logic_vector(C_DEPTH-1 downto 0);
    signal din            : std_logic;
    signal enable         : std_logic;

    constant C_MAX_CNT : integer := 8*10**6;
    signal cnt : integer range 0 to C_MAX_CNT-1;

begin

    process(clk,reset)
    begin
        if reset = '1' then
            cnt <= 0;
        elsif clk'event and clk = '1' then
            if cnt = C_MAX_CNT-1 then
                cnt <= 0;
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

    enable <= '1' when cnt = C_MAX_CNT-1 else '0';

    process(clk,reset)
    begin
        if reset = '1' then
            shift_register <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            if enable = '1' then
                shift_register <= shift_register(C_DEPTH-2 downto 0) & din;
            end if;
        end if;
    end process;

    din  <= shift_register(C_TAP_1-1) xor shift_register(C_TAP_2-1);
    dout <= shift_register(C_DEPTH-1);

end behavioral;
```