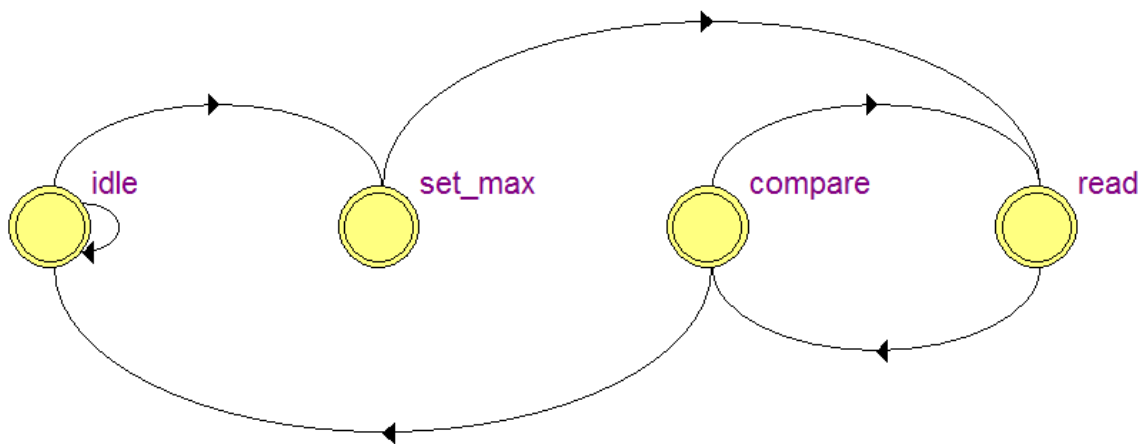


שאלה 1

דיאגרמת מצבים



הצהרת entity ו-architecture

```
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 mem_search is
7>   port (
8>     clk      : in  std_logic;
9>     rst      : in  std_logic;
10>    start    : in  std_logic;
11>    start_addr : in  std_logic_vector(7 downto 0);
12>    gap      : in  std_logic_vector(7 downto 0);
13>    rdata    : in  std_logic_vector(7 downto 0);
14>    re       : out std_logic;
15>    raddr    : out std_logic_vector(7 downto 0);
16>    max_addr : out std_logic_vector(7 downto 0);
17>    done     : out std_logic);
18> end entity mem_search;
19>
20>
21> architecture arc_mem_search of mem_search is
22>   type fsm_st is (idle, set_max, read, compare);
23>   signal cs      : fsm_st;
24>   signal addr_cnt : std_logic_vector(7 downto 0);
25>   signal end_addr : std_logic_vector(7 downto 0);
26>   signal max_val  : std_logic_vector(7 downto 0);
27> begin
```

איתחול בערכי ברירת מחול

```

27> raddr <= start_addr when (cs = idle) else addr_cnt;
28>
29> fsm : process (clk, rst) is
30>     variable ns : fsm_st;
31>     variable addr_cnt_en, addr_cnt_srst, last_addr, latch_addr, max_upd : boolean;
32> begin
33>     if (rst = '0') then
34>         cs <= idle;
35>         re <= '0';
36>         done <= '0';
37>         addr_cnt <= (others => '0');
38>         end_addr <= (others => '0');
39>         max_val <= (others => '0');
40>     elsif rising_edge(clk) then
41>         addr_cnt_en := false;
42>         last_addr := (addr_cnt = end_addr);
43>         latch_addr := false;
44>         addr_cnt_srst := false;
45>         max_upd := false;
46>         ns := cs;
47>         re <= '0';
48>         done <= '0';
49>

```

לוגיקה של מבב הבא והמוצאים

```

49> case cs is
50>     when idle => if (start = '1') then
51>         ns := set_max;
52>         latch_addr := true;
53>         re <= '1';
54>     else
55>         addr_cnt_srst := true;
56>     end if;
57>     when set_max => max_upd := true;
58>         ns := read;
59>     when read => re <= '1';
60>         ns := compare;
61>     when compare => max_upd := (max_val < rdata);
62>         if (last_addr) then
63>             ns := idle;
64>             done <= '1';
65>         else
66>             ns := read;
67>         end if;
68>     when others => ns := idle;
69> end case;
70> cs <= ns;
71>

```

המבנים הנשלטים

```

71> cs <= ns;
72> if(max_upd) then
73>     max_val <= rdata;
74>     max_addr <= addr_cnt;
75> end if;
76> if (addr_cnt_srst) then
77>     addr_cnt <= (others => '0');
78> elsif (latch_addr) then
79>     addr_cnt <= start_addr;
80> elsif (addr_cnt_en) then
81>     addr_cnt <= addr_cnt+1;
82> end if;
83> if (latch_addr) then
84>     end_addr <= start_addr+gap;
85> end if;
86> end if;
87> end process fsm;

```

שאלה 2

```
type bv_arr is array (natural range <>) of bit_vector(31 downto 0);

function double_swap ( a : bv_arr) return bv_arr is
    variable temp_arr : bv_arr;
begin
    for i in in a'range loop
        for j in a(i)'range loop
            temp_arr(i)(j):=a(a'high-i)(a(a'high-i)'high-j);
        end loop;
    end loop;
    return temp_arr;
end function double_swap;
```

שאלה 3

```
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 timer is
7>     port (
8>         clk          : in      std_logic;
9>         rst          : in      std_logic;
10>         tens_of_sec  : buffer std_logic_vector(2 downto 0);
11>         seconds      : buffer std_logic_vector(3 downto 0);
12>         sec_div_10   : buffer std_logic_vector(3 downto 0);
13>         sec_div_100  : buffer std_logic_vector(3 downto 0));
14> end entity timer;
15>
16> architecture arc_timer of timer is
17>     signal clock_div : std_logic_vector(16 downto 0);
18>     constant c_sec_div_100 : integer := 99999;
19> begin
20>     process(clk, rst) is
21>     begin
22>         if (rst = '0') then
23>             clock_div <= (others => '0');
24>             tens_of_sec <= (others => '0');
25>             seconds <= (others => '0');
26>             sec_div_10 <= (others => '0');
27>             sec_div_100 <= (others => '0');
28>         elsif rising_edge(clk) then
29>             if (clock_div = c_sec_div_100) then
30>                 clock_div <= (others => '0');
31>                 if (sec_div_100 = 9) then
32>                     sec_div_100 <= (others => '0');
33>                     if (sec_div_10 = 9) then
34>                         sec_div_10 <= (others => '0');
35>                         if (seconds = 9) then
36>                             seconds <= (others => '0');
37>                             if (tens_of_sec = 5) then
38>                                 tens_of_sec <= (others => '0');
39>                             else
40>                                 tens_of_sec <= tens_of_sec+1;
41>                             end if;
42>                         else
43>                             seconds <= seconds+1;
44>                         end if;
45>                     else
46>                         sec_div_10 <= sec_div_10+1;
47>                     end if;
48>                 else
49>                     sec_div_100 <= sec_div_100+1;
50>                 end if;
51>             else
52>                 clock_div <= clock_div+1;
53>             end if;
54>         end if;
55>     end process;
56> end architecture arc_timer;
```