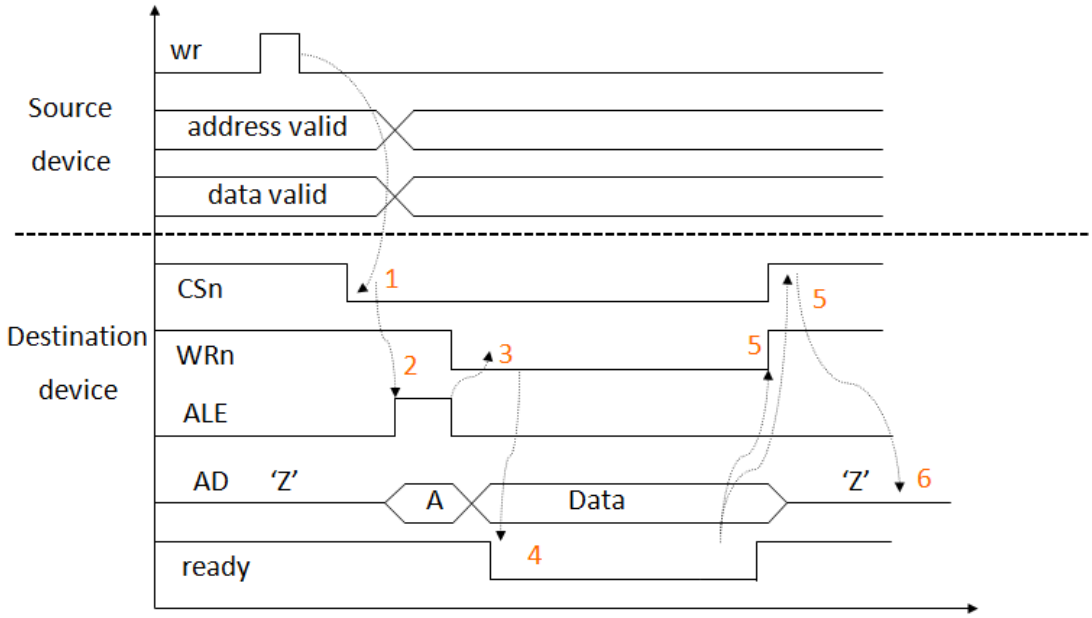
שאלה 1

ממש מערכת ספרתית סינכרונית המשמשת כמתווך בין שני רכיבים (source device ו- destination device) אשר הרכיב היוזם (source device) כותב לרכיב השני (destination device).

אותות בקרה של שני הרכיבים ורצף האירועים המתחולל מתאורים להלן:



איור 1.

שני הרכיבים פועלים לפי אותו השעון. הרכים היוזם פונה לפי צרכיו לרכים השני בכתיבה באמצעות קו wr העולה ל-'1' למחזור שעון אחד בלבד אך ורק כאשר רכיב השני מוכן לכך- קו ready נמצא ברמה לוגית '1', באותה עת קווי address וה-data (8 ביט כל אחד) אמורים להיות בערך רצוי.

רכיב המקבל מצוייד בקווי בקרה הבאים:

- Chip select (CS) הפעיל ברמה לוגית '0'
- write (wr)
- address latch enable (ale)
- address/data (ad) אפיק זה של 8 ביט נמצא ב -'Z' לפני ואחרי כתיבה לרכיב
- Ready

יש לממש מערכת המתווכת בין שני הרכיבים המוזכרים לעיל. המערכת תזהה שרכיב אחד רוצה לכתוב לרכיב השני (wr עולה לאחד) והרכים השני זמין לכתיבה (ready נימצא באחד) ותחולל רצף הבא:

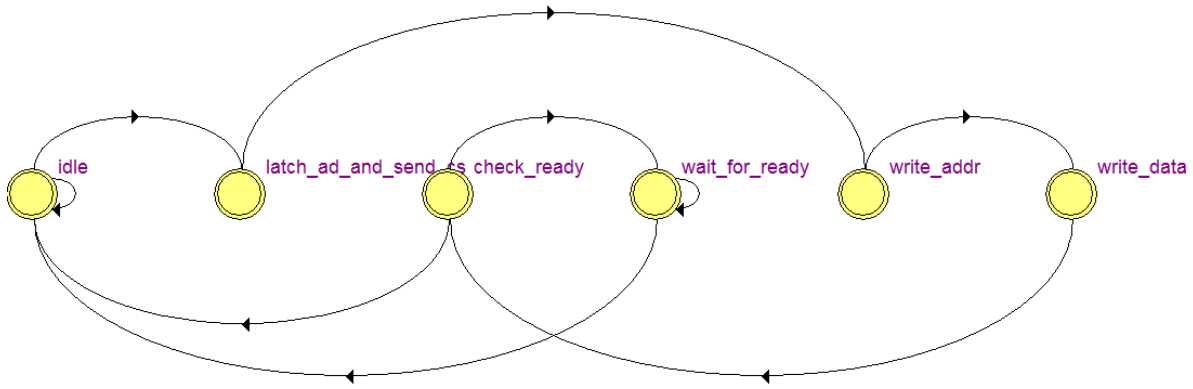
1. קו cs יורד לאפס.
2. קו ale עולה לאחד למחזור שעון אחד, בזמן זה address המתקבל מהרכיב היוזם מועבר לקו AD.
3. קו wr יורד לאפס, כתוצאה מכך קו ready יורד לאפס, אחרת יש לסיים טרנזקציה (ראה מחזור 6) ולהדליק ביט בשם error.
4. ה-data המתקבל מרכיב הכותב מועבר לקו AD.

5. מתבצעת המתנה לעליה של ready לאחד, המתנה זו לא תימשך מעבר ל-5 מחזורי שעון של מערכת מהירידה של ready לאפס. במידה וה-ready לא עלה, יש לסיים טרנזקציה (ראה מחזור 6) בכל זאת ולהדליק ביט בשם error.
6. עליה של ready גורמת ל-cs וה-wr לחזור לאחד. חזרתם של שני קווים אלא לאחד מסמנת סיום טרנזקציה ומחזירה קוו AD למצב של אכבה גבוהה.

בפתרון הבעיה יש לבצע שלבים הבאים:

1. לחלק מערכת לשני חלקים- המבצע והמבקר ולשרטט דיאגרמת בלוקים **(10 נקודות)**
2. לשרטט דיאגרמת מצבים של מכונת המצבים בחלק המבקר **(10 נקודות)**
3. לכתוב קוד VHDL מלא (entity, architecture, library) המייצג את המערכת כולה, כולל צד הבקרה הוצד המבוקר **(40 נקודות)**

דיאגרמת מצבים של יחידת הבקרה



הצהרת ה-entity

```
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 host_if is
7     port (
8        clk   : in  std_logic;
9        rst   : in  std_logic;
10       wr    : in  std_logic;
11       addr  : in  std_logic_vector(7 downto 0);
12       data  : in  std_logic_vector(7 downto 0);
13       ready : in  std_logic;
14       cs    : out std_logic;
15       wrn   : out std_logic;
16       ale   : out std_logic;
17       ad    : out std_logic_vector(7 downto 0);
18       err   : out std_logic);
19 end entity host_if;
20
```

הצהרת architecture והמשאבים הפנימיים

```
21
22  architecture arc_host_if of host_if is
23     signal data_reg, addr_reg, ad_sig : std_logic_vector(ad'range);
24     signal oe                         : std_logic;
25     type   fsm_st is (idle, latch_ad_and_send_cs, write_addr,
26                       write_data, check_ready, wait_for_ready);
27     signal curr_st                    : fsm_st;
28     signal cnt                        : std_logic_vector(2 downto 0);
29  begin
30
```

ניהול הבוס הדו-כיווני ופעולות איתחול של מכונת המצבים

```
30     ad <= ad_sig when (oe = '1') else (others => 'Z');
31
32     fsm : process (clk, rst) is
33        variable next_st                  : fsm_st;
34        variable ad_sel, ad_latch, cnt_en : std_logic;
35        variable timeout_end              : boolean;
36     begin
37        if (rst = '0') then
38           curr_st  <= idle;
39           oe       <= '0';
40           cs       <= '1';
41           wrn      <= '1';
42           ad_sig   <= (others => '0');
43           data_reg <= (others => '0');
44           addr_reg <= (others => '0');
45           ale      <= '0';
46           err      <= '0';
47
```

ערכי ברירת מחדל של המכונה

```
47        elsif rising_edge(clk) then
48           next_st     := curr_st;
49           oe          <= '0';
50           cs          <= '0';
51           wrn         <= '1';
52           ale         <= '0';
53           err         <= '0';
54           ad_sel      := '0';
55           ad_latch    := '0';
56           cnt_en      := '0';
57           timeout_end := (cnt = 5);
```

לוגיקה של מצב הבא ומוצאים של מכונת המצבים

```
58          case curr_st is
59             when idle => if ((wr and ready) = '1') then
60                              next_st := latch_ad_and_send_cs;
61                           end if;
62                           cs <= '1';
63             when latch_ad_and_send_cs => next_st:= write_addr; ad_latch := '1';
64             when write_addr => ad_sel:= '1';  ale<= '1';
65                                next_st:= write_data; oe<= '1';
66             when write_data => wrn <= '0'; cnt_en := '1';
67                                next_st:= check_ready; oe<= '1';
68             when check_ready => if (ready = '1') then
69                                     err<= '1'; cs <= '1';
70                                     next_st := idle;
71                                  else
72                                     wrn <= '0'; oe<= '1';
73                                     cnt_en  := '1';
74                                     next_st := wait_for_ready;
75                                  end if;
76             when wait_for_ready => if (ready = '1') then
77                                      next_st := idle;
78                                      wrn <= '0'; oe <= '1';
79                                     elsif (timeout_end) then
80                                      next_st := idle;
81                                      err <= '1'; cs <= '1';
82                                     else
83                                      cnt_en:= '1'; wrn <= '0'; oe<= '1';
84                                     end if;
85             when others => next_st:= idle; cs<='1';
86          end case;
87          curr_st<=next_st;
88
```

המבנים הנשלטים ע״י מכונת המצבים

```
89             if (ad_latch = '1') then
90                addr_reg <= addr;
91                data_reg <= data;
92             end if;
93             if (ad_sel = '1') then
94                ad_sig <= addr_reg;
95             else
96                ad_sig <= data_reg;
97             end if;
98             -- timeout counter
99             if (cnt_en = '0') then
100               cnt <= (others => '0');
101            else
102               cnt <= cnt+1;
103            end if;
104         end if;
105      end process fsm;
106   end architecture arc_host_if;
107
```

שאלה 2: לכתוב פונקציה רקורסיבית המקבלת כארגומנט bit_vector ומחזירה מיקום של '1' השמאלי ביותר, כאשר ספירה מתבצעת מימין לשמאל. ערך המוחזר של פונקציה הוא integer.

הפתרון מתבסס על הנחת ייסוד שארגומנט המתקבל ע"י פונקציה כולל יותר מביט אחד.

```
function find_first_one (a : bit_vector) return integer is
   alias aa            : bit_vector(a'length-1 downto 0) is a;
   variable pos        : integer;
   variable bitwise_or : bit := '0';
begin
   for i in aa'range loop
      bitwise_or := bitwise_or or aa(i);
   end loop;
   if (bitwise_or = '0') then
      pos := 0;
   elsif (aa(aa'left) = '1') then
      pos := aa'left;
   else
      pos := find_first_one(aa(aa'left-1 downto 0));
   end if;
   return pos;
end function find_first_one;
```

שאלה 3: לרשותך component המייצג מסכם למחצה (half adder) המצוייד הכניסות a ו-b של ביט אחד והיציאות s ו-co גם כן של ביט אחד. ממש באמצעות ה-generate וה-component הנתון מערכת המקבלת בכניסה ווקטור אשר גודלה נקבע באמצעות generic ומחזירה את המשלים ל-2 שלו.

```
1  entity twos_comp is
2     generic (size : integer);
3     port ( a : in  bit_vector(size-1 downto 0);
4            b : out bit_vector(size-1 downto 0);
5  end entity twos_comp;
6  architecture arc_twos_comp of twos_comp is
7  component ha is
8     port ( a  : in  bit;
9            b  : in  bit;
10           s  : out bit;
11           co : out bit);
12 end component ha;
13 signal carry : bit_vector(b'length downto b'low);
14 begin
15    carry(0)<='1';
16    g1: for i in b'reverse_range generate
17       signal not_a : bit_vector(b'range);
18    begin
19       u1: ha
20          port map (
21             a  => a(i),
22             b  => '1',
23             s  => not_a(i),
24             co => open);
25       u2: ha
26          port map (
27             a  => not_a(i),
28             b  => carry(i),
29             s  => b(i),
30             co => carry(i+1));
31    end generate g1;
32 end architecture arc_twos_comp;
33
```