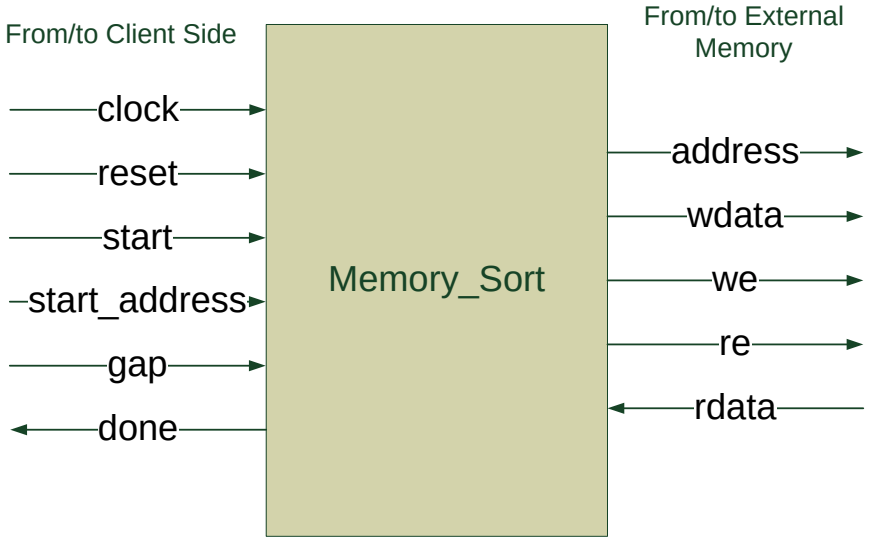
תכן חומרה ספרתית בVHDL
פתרון המבחן

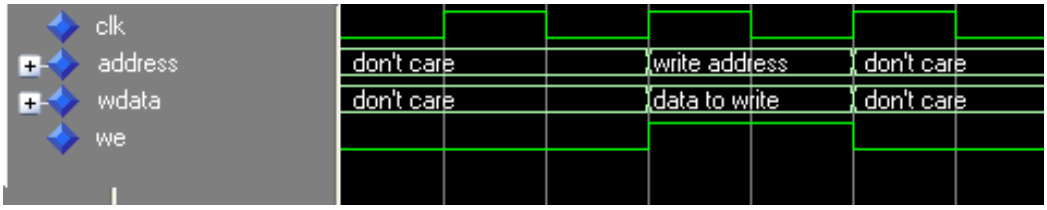
שאלה 1

בשאלה זו יש לתכנן מערכת ספרתית, סינכרונית המוצגת באיור 1.



איור 1

מערכת זו מקבלת כתובת התחלתית (start_address) ותחום של כתובות רציפות (gap) החל מ-start_address, בתחום זה היא מבצעת מיון של הנתונים השמורים בזכרון חיצוני לפי אלגוריתם Bubble Sort המתואר בהמשך. המערכת מתחילה את המיון לפי אות start המתקבל למחזור שעון אחד ומודיעה על סיום ע״י אות done. בסיום המיון בכתובת הראשונה של תחום מיון המבוקש יימצא הנתון המיזערי, בכתובת הבאה –נתון גדול מימנו (או שווה לו) וכך עד סוף תחום המיון, בכתובת האחרונה יימצא נתון המרבי. פנייה לזכרון חיצוני נעשת באמצעות קווים הבאים: כתובת לקריאה/כתיבה (address), נתון לכתיבה (wdata), איפשור כתיבה (we), איפשור קריאה (re) ונתון הנקרא מזכרון (rdata). קווי הכתובת משני הצדדים הם קווים של 10 ביט, תחום המיון (gap) הוא קוו של 8 ביט, wdata וה-rdata בעלי רוחב של 8 ביט. מחזור קריאה וכתיב מהזכרון מוצגים להלן:



בזיכרון הנתון אין אפשרות לביצוע כתיבה וקריאה בו זמנית.
להלן קטע קוד בשפת C המתאר את אלגוריתם Bubble Sort:

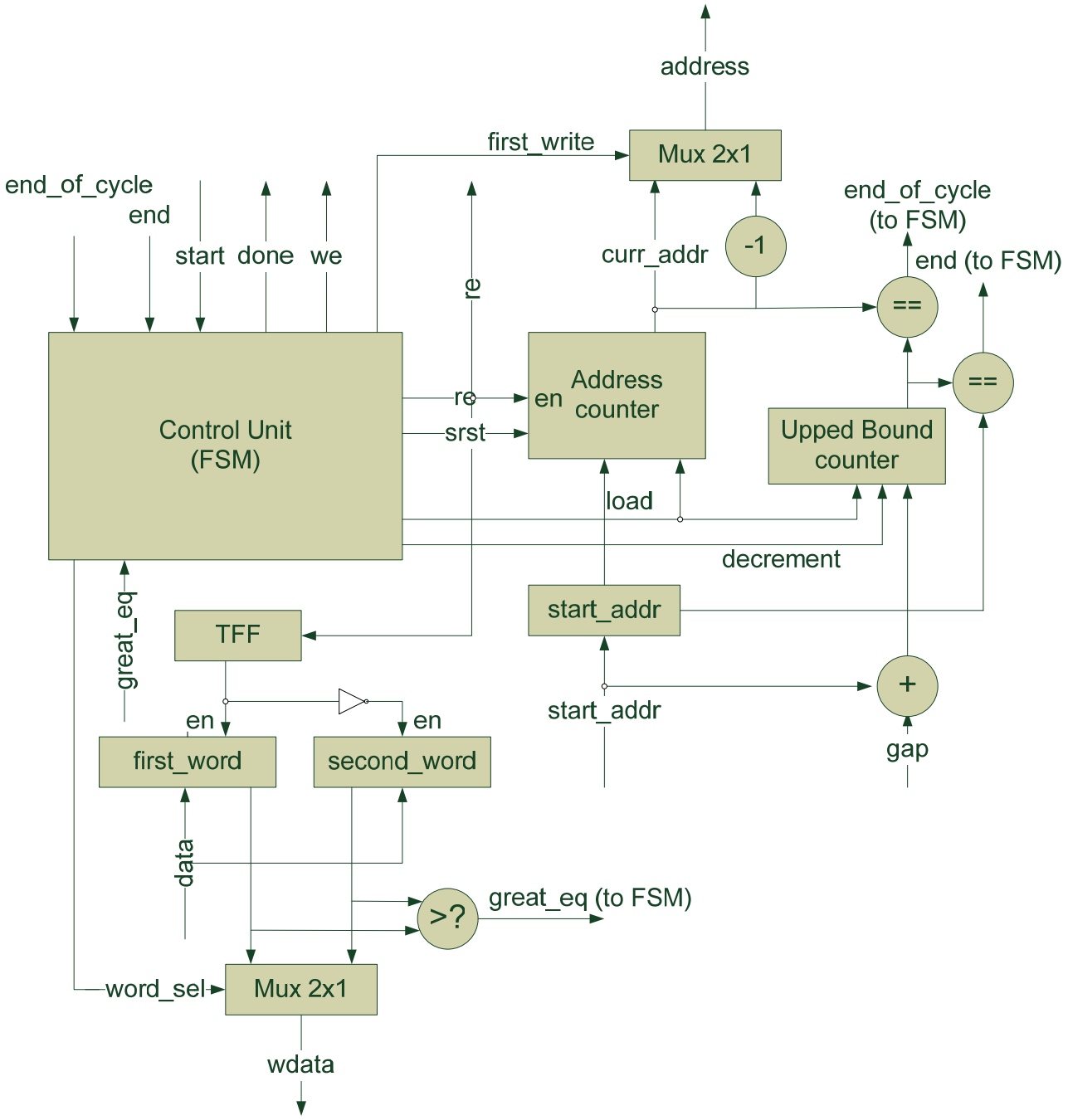
```
for(i=0;i<SORT_BOUNDARY;i++)
{
   for (j=0;j<(SORT_BOUNDARY-i);j++)
   {
       if (num[j+1]<num[j])
       {
           temp = num[j];
           num[j] = num[j+1];
           num[j+1] = temp;
       }
   }
}
```

המשימה: ממש את מערכת ע"י חלוקה לתת מבנים: יחידת בקרה ויחידות הביצוע. את יחידת הבקרה יש לממש כמכונת מצבים (FSM). **אין לעסוק בבדיקת תקינות הקלט של מערכת.**
שלבי העבודה מוערכים באופן הבא:

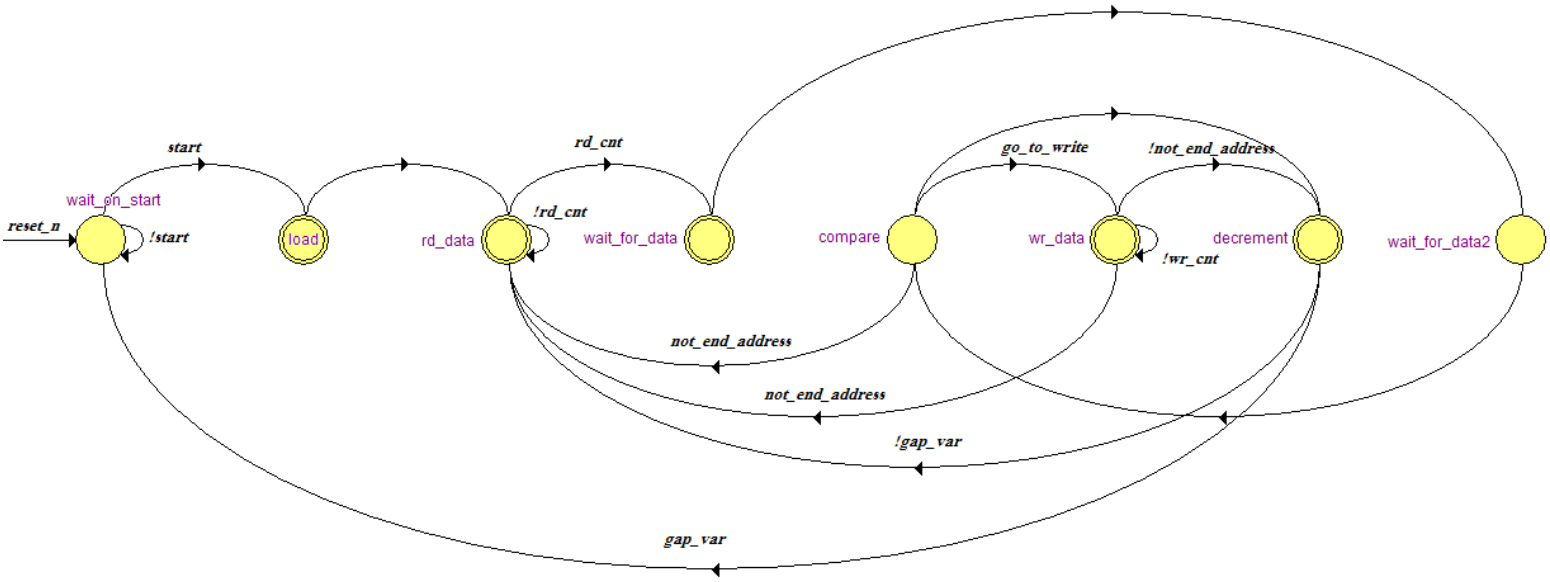
א) חלוקה לתת מבנים וסרטוט של המערכת ברמת תת מבנים -**10 נקודות**.
ב) תיאור דיאגרמאת מצבים של יחידת הבקרה (FSM)-**10 נקודות**
ג) כתיבת קוד בר סינתזה הכולל entity, architecture המתארים את המערכת-**30 נקודות**

הפתרון:

א) דיאגרמאת בלוקים של המערכת



ב) דיאגרמת מצבים



ג) להלן תאור המערכת ב-VHDL

```
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  entity bubble_sort is
6     port (
7        clk           : in  std_logic;
8        reset_n       : in  std_logic;
9        start         : in  std_logic;
10       start_address : in  std_logic_vector(9 downto 0);
11       gap           : in  std_logic_vector(9 downto 0);
12       read_data     : in  std_logic_vector(7 downto 0);
13       write_data    : out std_logic_vector(7 downto 0);
14       wr_en         : buffer std_logic;
15       rd_en         : buffer std_logic;
16       wr_address    : out std_logic_vector(9 downto 0);
17       rd_address    : buffer std_logic_vector(9 downto 0)
18       );
19 end entity bubble_sort;
20
21 architecture arc_bubble_sort of bubble_sort is
22    signal end_address, gap_var, current_address                : std_logic_vector(9 downto 0);
23    signal rd_cnt, wr_cnt, reg_en, not_end_address, go_to_write : std_logic;
24    signal ld_cnt, en_cnt, ld_dec, en_dec                       : std_logic;
25    signal rd_data_temp                                         : std_logic_vector(7 downto 0);
26    type fsm_states is (wait_on_start, load, rd_data, wait_for_data,
27                        wait_for_data2, compare, wr_data, decrement);
28    signal curr_state, next_state : fsm_states;
29 begin
30    wr_address      <= rd_address-wr_cnt;
31    rd_address      <= current_address;
32    end_address     <= start_address+gap_var;
33    go_to_write     <= '1'  when (read_data < rd_data_temp)       else '0';
34    not_end_address <= '1'  when (current_address /= end_address) else '0';
35    write_data      <= read_data when (wr_cnt='1') else rd_data_temp;
36
```

```
37  process (clk, reset_n) is
38  begin
39     if (reset_n = '0') then
40        rd_cnt          <= '0';
41        wr_cnt          <= '0';
42        rd_data_temp    <= (others => '0');
43        current_address <= (others => '0');
44        gap_var         <= (others => '0');
45        curr_state      <= wait_on_start;
46     elsif rising_edge(clk) then
47        rd_cnt <= rd_cnt xor rd_en;
48        wr_cnt <= wr_cnt xor wr_en;
49        curr_state<= next_state;
50        if (reg_en = '1') then
51           rd_data_temp <= read_data;
52        end if;
53        if ((ld_cnt or en_dec) = '1') then
54           current_address <= start_address;
55        else current_address <= current_address+en_cnt;
56        end if;
57        if (ld_dec = '1') then
58           gap_var <= gap;
59        elsif (en_dec = '1') then
60           if (gap_var /= 0) then
61              gap_var <= gap_var-1;
62           end if;
63        end if;
64     end if;
65  end process;
```

```
67    process(curr_state, start, rd_cnt, go_to_write, not_end_address, wr_cnt, gap_var) is
68    begin
69       next_state <= curr_state;
70       ld_dec<= '0'; ld_cnt <= '0'; en_dec <= '0';en_cnt <= '0';
71       wr_en <= '0'; rd_en <= '0'; reg_en     <= '0';
72       case (curr_state) is
73          when wait_on_start => if (start = '1') then
74                                   next_state <= load;
75                                end if;
76          when load => ld_dec <= '1'; ld_cnt     <= '1';
77                       next_state <= rd_data;
78          when rd_data => rd_en <= '1'; reg_en <= '1';
79                          if (rd_cnt = '0') then
80                             en_cnt <= '1';
81                          else next_state <= wait_for_data;
82                          end if;
83          when wait_for_data => reg_en <= '1';
84                                next_state <= wait_for_data2;
85          when wait_for_data2 => next_state <= compare;
86          when compare        => if (go_to_write = '1') then
87                                    next_state <= wr_data;
88                                 elsif (not_end_address = '1') then
89                                    next_state <= rd_data;
90                                 else next_state <= decrement;
91                                 end if;
92          when wr_data => wr_en <= '1';
93                          if (wr_cnt = '1') then
94                             if (not_end_address = '1') then
95                                next_state <= rd_data;
96                             else next_state <= decrement;
97                             end if;
98                          end if;
99          when decrement => en_dec <= '1';
100                           if (gap_var = 1) then
101                              next_state <= wait_on_start;
102                           else next_state <= rd_data;
103                           end if;
104       end case;
105    end process;
106 end architecture arc_bubble_sort;
```

שאלה 2 (30 נקודות).

נתון תיאור של מערכת כל שהיא.

```
1  package my_pack is
2     function get_y_size (constant x_width : integer) return integer;
3  end package my_pack;
4
5  package body my_pack is
6     function get_y_size (constant x_width : integer) return integer is
7     begin
8        for i in 1 to x_width loop
9           if ((2**i) >= (x_width+i+1)) then
10             return (x_width+i);
11          end if;
12       end loop;
13       return 0;
14    end function get_y_size;
15 end package body my_pack;
16
17 library ieee;
18 use ieee.std_logic_1164.all;
19 use ieee.std_logic_arith.all;
20 use ieee.std_logic_unsigned.all;
21 library work;
22 use work.my_pack.all;
23
24 entity trouble is
25    generic (x_width : integer := 12);
26    port(
27       clk      : in  std_logic;
28       rst      : in  std_logic;
29       x_en     : in  std_logic;
30       x        : in  std_logic_vector((x_width-1) downto 0);
31       y_val      : out std_logic;
32       y : out std_logic_vector((get_y_size(x_width)-1) downto 0)
33       );
34 end entity trouble;
```

```
36  architecture arc_trouble of trouble is
37  begin
38     process (clk, rst) is
39        variable a                 : std_logic_vector(0 to ((y'length-x'length)-1));
40        variable y_ascending_range : std_logic_vector(1 to y'length);
41        variable v                 : std_logic_vector((a'length-1) downto 0);
42        variable k                 : integer;
43        alias x_ascending_range    : std_logic_vector(1 to x'length) is x;
44     begin
45        if (rst ='0') then
46           y <= (others => '0');
47           y_val<= '0';
48        elsif rising_edge(clk) then
49           y_ascending_range := (others => '0');
50           a := (others => '0');
51           k := 1;
52           for n in 1 to a'high loop
53              for m in ((2**n)+1) to ((2**(n+1))-1) loop
54                 if k <= x'length then
55                    y_ascending_range(m) := x_ascending_range(k);
56                    k := k+1;
57                 end if;
58              end loop;
59           end loop;
60           for i in a'range loop
61              for j in y_ascending_range'range loop
62                 v := conv_std_logic_vector(j, v'length);
63                 if (v(i) = '1') then
64                    a(i):= a(i) xor y_ascending_range(j);
65                    y_ascending_range(2**i) := a(i);
66                 end if;
67              end loop;
68           end loop;
69           y_val <= x_en;
70           if (x_en = '1') then
71             y <= y_ascending_range;
72           end if;
73        end if;
74     end process;
75  end architecture arc_trouble;
```

א) הסביר/י את הפעולות המתבצעות במערכת

מערכת הנ"ל היא מקודד בקוד Hamming, מתבצע חישוב גודל של מילת המוצע באמצעות פונקציה get_y_size, לפי גודל זה נקבעים מיקומים וערכב של ביטי ה-parity ומילה סוסית המלווה באות y_val נדגמת ברגיסטר המוצע.

ב) האם הקוד הנ"ל הוא בר סנתזה? כן/לא נמק/י

הקוד הנ"ל הוא כן בר סינתזה על פי כללים הבאים:

- הלולאות מתבצעות מספר פעמים קבוע ומוגדר מראש.
- הלולאות אינן מייצגות חומרה! כל משמתבצע בהן הוא חישובים של הקבועים אשר נעשים בזמן הקומפילציה.
- הערכים המחושבים בהן משמשים לחיבור ישיר בן הכניסה וליציא או לחיבור של מספר סופי של ביטים בשערי XOR.
- אין שימוש בטיפוסים שונים מלבד שימוש ב-std_logic
- אין שימוש ב-wait

ג) סרטט/י את המעגל המתואר בקוד זה בעבור x_width=5

שאלה 3 (20 נקודות).

כתוב/י פונקציה הנתונה לסינתזה **ללא שימוש בפונקציות סטנדרטיות**, המקבלת ווקטור בינארי (bit_vector) ומחזירה ערך עשרוני שלו כ-integer . יש להתייחס לאפשרות שגודל הוקטור כניסה ייעלה על גודל ה-integer (32 ביט, אשר ה-MSB הוא סימן)

להלן דוגמה לפונקציה אשר ממירה מ-bit_vector ל-integer ולא מאפשרת גלישה מעבר ל- 32 bit

```
2   function bitvector2int (arg : bit_vector) return integer is
3      variable res          : integer:= 0;
4      alias arg_descending : bit_vector(arg'length-1 downto 0) is arg;
5      variable buff         : bit_vector(31 downto 0) := (others => '0');
6   begin
7      for j in 0 to arg_descending'left loop
8         if (j <= 31) then
9            buff(j) := arg(j);
10        end if;
11     end loop;
12     for i in buff'range loop
13        res := res+res;
14        if buff(i) = '1' then
15           res := res + 1;
16        end if;
17     end loop;
18     return res;
19  end function bitvector2int;
```