שאלה 1

בשאלה זאת יש לתכנן מכפל בינארי הפועל בשיטת Radix 4.

הכופל B (multiplier) והנכפל A (multiplicand) הם נתונים של 16 ביט, תוצאות ביניים (PP- partial products) והתוצאה P (product) הן של 32 ביט. מעגל כפל Radix 4 מוצג באיור 1, לפי האלגוריתם זה מחשבים partial products בעבור 2 ביט הימניים של הכופל ומזיזים הנכפל ב-2 ביט שמאלה בכל מחזור. ה-PP מחושב לפי ערך בינארי של 2 ביט של הכופל באופן הבא:

$$PP = 0,\ when\ "00"$$
$$PP = A,\ when\ "01"$$
$$PP = 2A,\ when\ "10"$$
$$PP = 3A,\ when\ "11"$$

כל ה-partial products מתקבלים באופן פשוט פרט למקרה של "11" ב-2 ביט המנותחים של הכופל. במקרה זה יש צורך לחשב $3A$. אלגוריתם Radix 4 מציע במקום חיבור של $3A$ הניתן לחישוב כ- $(4A - A)$, לחבר רק $-A$ (במשלים ל-2), ואת $4A$ הנותרים להשלים במחזור הבא ע"י הוספה של A אשר עד אז יהיה מוזז שמאלה ב-2 ביט ויהיה שווה ל- $4A$ במחזור הנוכחי. אינדיקציה על פעולה זאת מתקבלת ע"י סיגנל carry העולה ל-'1'. מספר פעולות חיבור/הזזה לפי שיטה זו הוא $(n/2)+1$ לעומת n בגישה קנונית. אלגוריתם Radix 4 מודגם להלן:

נתון נכפל A והכופל B שניהם 4 ביט כאשר: $A = 1011, B = 1011$.

מחזור ראשון: מתחילים מצובר המאותחל ב-'0'. מחשבים PP, זוג הביטים הימניים של הכופל הם "11", יש לבצע חיבור של $-A$ ומדליקים carry='1', $PP = -A = 11110101$, מזיזים שמאלה ב-2 ביט את A ואז $A = 00101100$, מזיזים את B ימינה ב-2 ביט ואז $B = 0010$. מחברים בצובר את PP, התוצאה היא 11110101

מחזור שני: זוג הביטים הימניים של הכופל הוא "10" אך ה-carry הוא 1, לכן יש לחבר אותו ולאחר חיבור הזוג הנוכחי "11" ולפיו אנו מחשבים את PP. יש לבצע חיבור של $-A$ (במשלים ל-2) ולהדליק carry='1', $PP = -A = 11010100$, מזיזים שמאלה ב-2 ביט את A ואז $A = 10110000$, מזיזים את B ימינה ב-2 ביט ואז $B = 0000$. מחברים בצובר את PP, התוצאה היא 11001001.

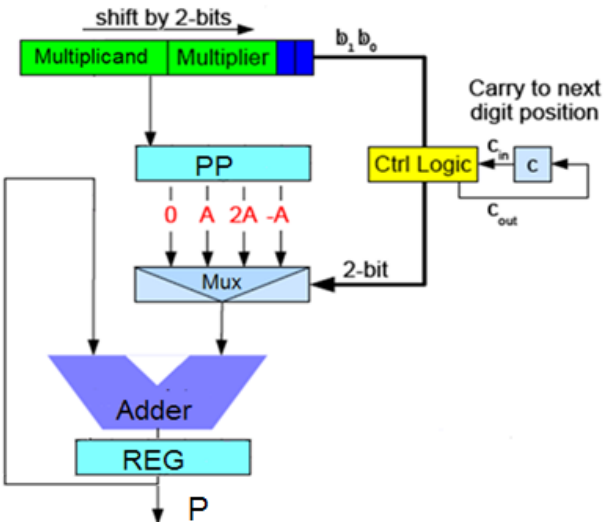
מחזור השלישי: הוא המחזור האחרון כי הכופל הוא של 4-ביט וכולל שתי זוגות של ביטים- קרי 2 מחזורים, אבל יש לבדוק את ה-carry. אם ה-carry הוא '0' אז זהו סוף הפעולה, אחרת יש לבצע עוד חיבור אחד של A. בדוגמה זאת carry='1' לכן יש לחבר את A ששווה ל- 10110000 אחרי שתי הזזות ב-2 ביט לערך אותו צברנו עד כה והתוצאה הסופית היא 01111001.

כל הפעולות:

00001011_b (A)
*
00001011_b (B)

11110101_b (-A) -11_d
+
11010100_b (-A *4) -44_d
+
10110000_b (A *16) $+ 176_d$

$01111001_b = 121_d$



איור 1

נא לקרוא בעיון את ההסבר הנ"ל לפני מתן הפתרון!

<u>המשימה:</u> יש לממש מערכת ספרתית, **סינכרונית** המבצעת כפל בין שני מילים של 16 ביט כל אחד לפי אלגוריתם Radix 4. פעולת המערכת מתחילה ע"י אות **start** העולה ל- '1' למחזור שעון אחד בלבד. מערכת מודיע על סיום פעולת ע"י אות **done** העולה ל-'1' למחזור שעון אחד בלבד, שאר הזמן אות זה נמצא ב-'0'.

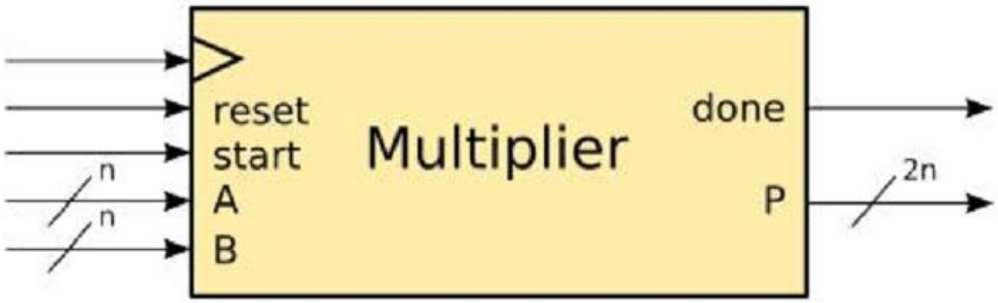
א) חלק את המערכת ליחדת הביצוע ויחידת הבקרה וסרטט דיאגרמת בלוקים **(5 נקודות)**

ב) סרטט דיגרמת מצבים של ה-FSM של יחידת הבקרה **(10 נקודות)**

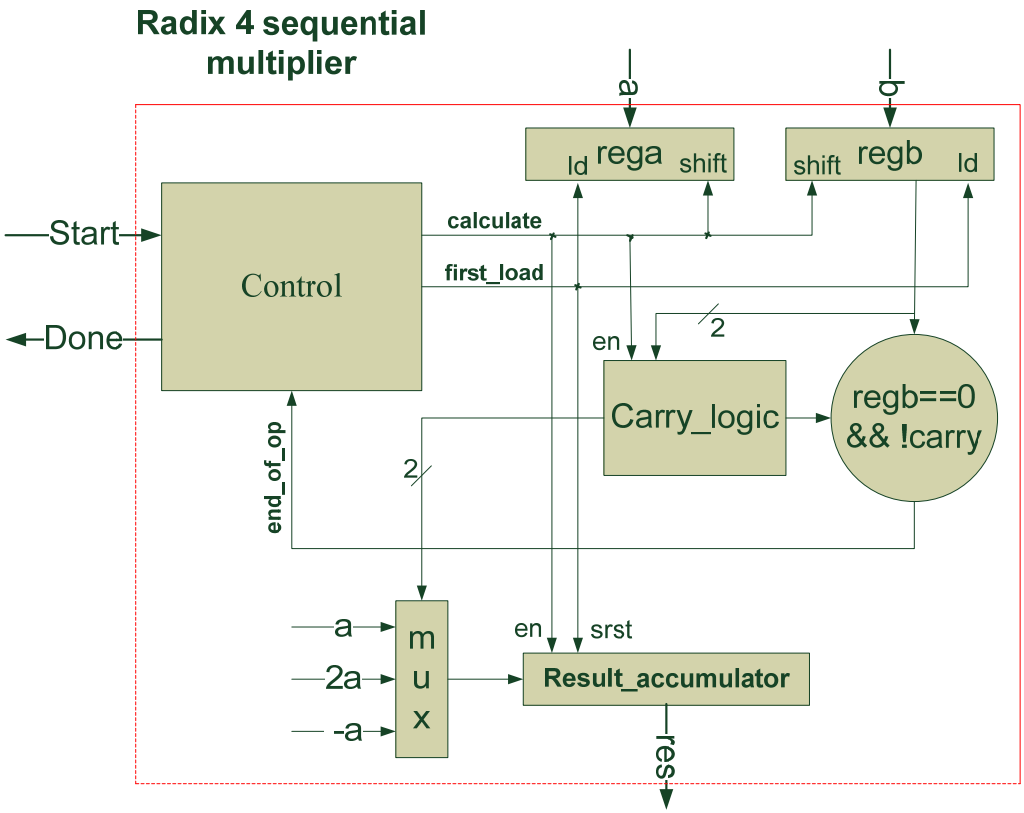
ג) תאר את המערכת בשפת VHDL באופן מלא הכולל entity ו-architecture. יש לכתוב מודל הקביל לסנתזה **(35 נקודות)**

הפתרון:

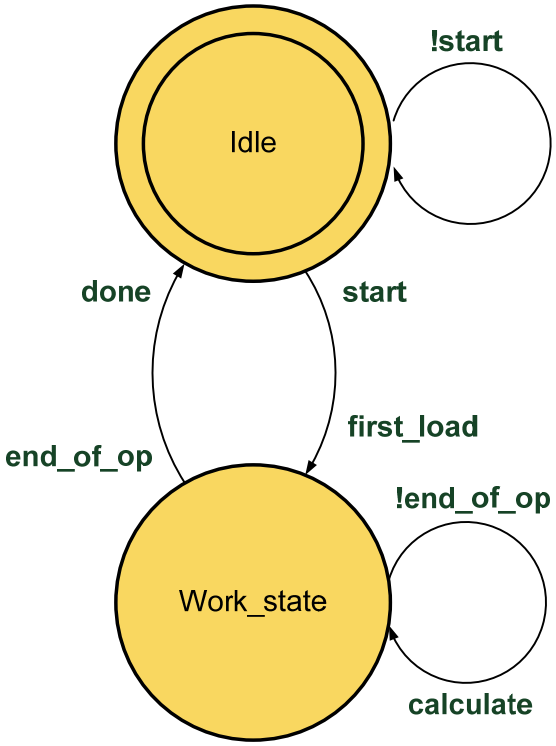
א) הגדרת הכניסות והיצואות של המערכת



דיאגראמת בלוקים :



ב) דיאגרמאת מצבים



ג) 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
6  entity radix4_mul is
7     port (
8        clk   : in     std_logic;
9        rst   : in     std_logic;
10       start : in     std_logic;
11       a     : in     std_logic_vector(15 downto 0);
12       b     : in     std_logic_vector(15 downto 0);
13       res   : buffer std_logic_vector(31 downto 0);
14       done  : out    std_logic
15       );
16 end entity radix4_mul;
17
18 architecture arc_radix4_mul of radix4_mul is
19    signal carry                          : std_logic;
20    signal end_of_op, first_load, calculate : boolean;
21    signal areg, breg                     : std_logic_vector(res'range);
22    alias  curr_pair                      : std_logic_vector(1 downto 0) is breg(1 downto 0);
23    type   state_type is (idle, work_state);
24    signal cs, ns : state_type;
25 begin
26    state_reg : process (clk, rst) is
27    begin
28       if (rst = '0') then
29          cs <= idle;
30       elsif rising_edge(clk) then
31          cs <= ns;
32       end if;
33    end process state_reg;
```

```
35    comb_logic : process (cs, start, end_of_op) is
36    begin
37       first_load <= false;
38       calculate  <= false;
39       done       <= '0';
40       ns         <= cs;
41       case cs is
42          when idle =>        if(start = '1') then
43                                ns <= work_state;
44                                first_load <= true;
45                              end if;
46          when work_state => if (not end_of_op) then
47                                calculate <= true;
48                              else
49                                done <= '1';
50                                ns   <= idle;
51                              end if;
52          when others => null;
53       end case;
54    end process comb_logic;
```

```
56    end_of_op <= (breg = 0) and (carry = '0');
57
58    shift_accumulate : process (clk, rst) is
59       variable curr: std_logic_vector(1 downto 0);
60    begin
61       if (rst = '0') then
62          res   <= (others => '0');
63          carry <= '0';
64          areg  <= (others => '0');
65          breg  <= (others => '0');
66       elsif rising_edge(clk) then
67          curr  := curr_pair + carry;
68          if (first_load) then
69             carry <= '0';
70             res   <= (others => '0');
71             areg  <= x"0000" & a;
72             breg  <= x"0000" & b;
73          elsif (calculate) then
74             -- shift left of multiplicant and shift right of multpiler
75             areg  <= areg(29 downto 0) & "00";
76             breg  <= "00" & breg(31 downto 2);
77             -- carry calculation
78             carry<=curr_pair(1) and (carry or curr_pair(0));
79             -- accumulate result
80             if (curr = "01") then
81                res <= res + areg;
82             elsif (curr = "10") then
83                res <= res + (areg(30 downto 0) & '0');
84             elsif (curr = "11") then
85                res <= res - areg;
86             end if;
87          end if;
88       end if;
89    end process shift_accumulate;
90 end architecture arc_radix4_mul;
```

שאלה 2

נתון תיאור של מערכת כל שהיא.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity trouble is
   generic (size : integer := 8);
   port(
      din  : in  std_logic_vector(size -1 downto 0);
      dout : out std_logic_vector(1 downto 0)
      );
end entity trouble;
architecture arc_trouble of trouble is
   function do_something (x : std_logic_vector) return std_logic_vector is
      variable y : std_logic_vector(1 downto 0);
      variable t : std_logic_vector(3 downto 0);
      variable m : integer;
   begin
      case x'length is
         when 2 => y := x;
         when 3 => y(y'high):= (x(x'high) or x(x'high-1));
                   y(y'low) := ((x(x'high) and x(x'high-1)) or x(x'low));
         when 4 => y(y'high):= (x(x'high) or x(x'high-1)) or (x(x'low+1) and x(x'low));
                   y(y'low) := (x(x'high) and x(x'high-1)) or (x(x'low+1) or x(x'low));
         when others =>
                   m := (x'length + 1)/2 + x'right;
                   t(t'high downto (t'length/2))    := do_something(x(x'high downto m));
                   t(((t'length/2)-1) downto t'low):= do_something(x(m-1 downto x'low));
                   y := do_something(t);
      end case;
      return y;
   end function do_something;
begin
   dout <= do_something(din);
end architecture arc_trouble;
```

א) הסביר את הפעולות המתבצעות במערכת הנ"ל .

מערכת זו בודקת האם מילה בינרית המתקבלת בכניסתה מוצגת בקוד One-Hot או לא.

מהו מספר האיטרציות המתבצעות בעבור size=8?

מערכת ממומשת באמצעות פונקציה רקורסיבית, אשר בעבור מילת כניסה של 8 ביט תבצע 2 איטרציות.
באטרציה ראשונה תחולק המערכת לשני תת-מבנים 4 ביט כניסה ושני בית יציאה כל אחת.
באטרציה השנייה המוצאים של דרגות הקודמות יחוברו ככניסה ליחידה נוספת ויוחזר ערך סופי.

ב) חשב את ערך המוצא בעבור צירופים הבאים בכניסה

1. "x"08 התוצאה היא '01'
2. "x"41 התוצאה היא '11'

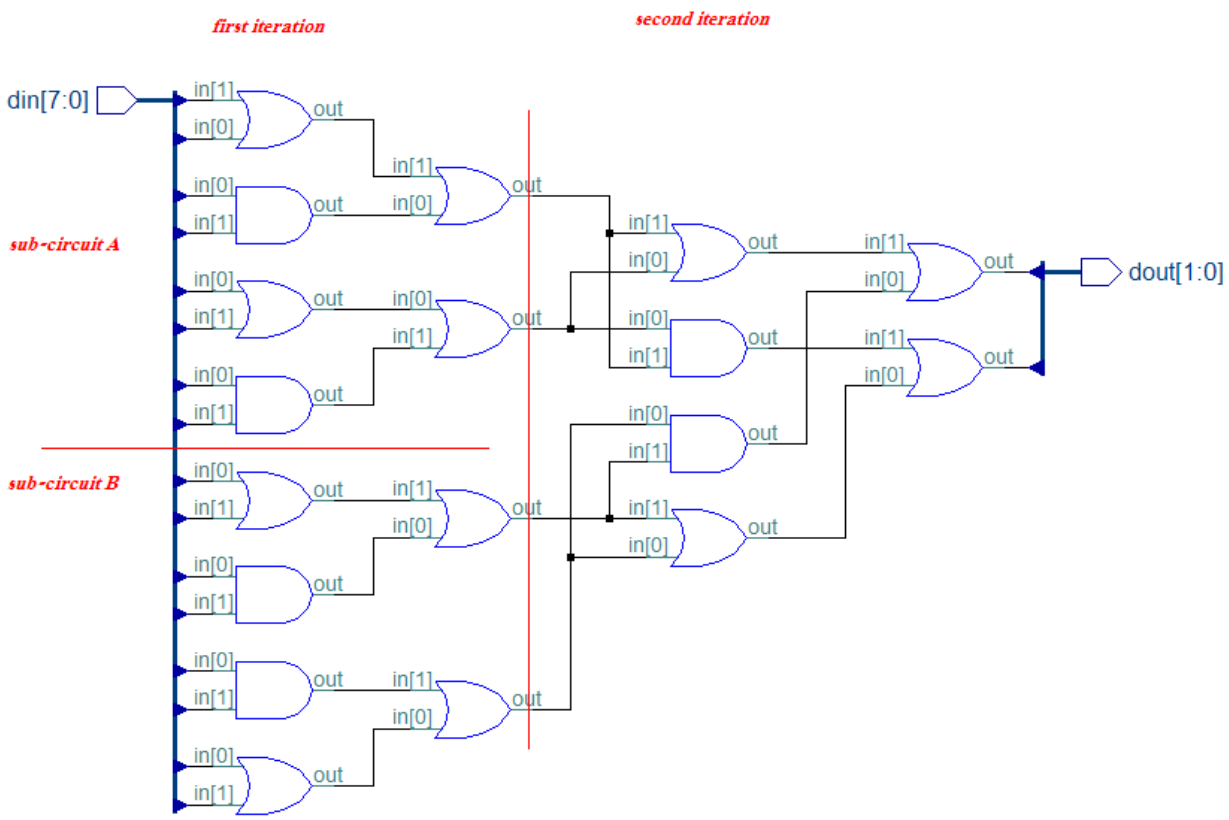
ג) האם הקוד הנ"ל הוא בר סנתזה?

הקוד הנ"ל כן בר סינתזה. המערכת ממומשת האמצעות פונקציה רקורסיבית אשר מקיימת שלושת הכללי רקורסיה: תנאי עצירה; ביצוע פעולה על המשתני כניסה; קריאה לפונקציה עם ארגומנטים חדש. בנוסף, אין שימוש בטיפוסים הלא ניתנים לסינתזה וכל הפעולות המתוארות הן פעולות לוגיות בלבד.

ד) הסבר/י האם קוד הנ"ל נותן מענה לכל האפשרויות של גודל הכניסה?

בקוד הנתון אין התייחסות לאפשות שמילת כניסה תיהיה ברוחב ביט אחד.

ה) סרטט/י את המעגל המתואר בקוד זה בעבור size=8, ברמת שערים לוגיים



שאלה 3

כתוב/י פונקציה **הנתונה לסינתזה** המקבלת ווקטור בינארי din מטיפוס std_logic_vector ומחזירה את ה- $\log_2 din$ כערך עשרוני (integer) המעוגל כלפי מעלה

```
function log2 (d : std_logic_vector) return integer is
begin
   for i in 0 to d'length-1 loop
      if (2**i>=d) then
         return i;
      end if;
   end loop;
   return 0;
end function log2;
```