שפת תיאור חומרה VHDL: פתרון המבחן

שאלה 1 -60 נקודות: להלן תאור של מערכת ספרתית סינכרונית המבצעת כפל בין שני ארגומנטים a (הנכפל) ו-b (הכופל) של 16 ביט כל אחד לפי אלגוריתם Radix 16 סידרתי:

$$A = \{a_{n-1}, a_{n-2}, \ldots, a_1, a_0\} \text{ the multiplicand,}$$

$$B = \{b_{n-1}, b_{n-2}, \ldots, b_1, b_0\} \text{ the multiplier,}$$

$$P = \sum_{i=0}^{\lceil (n-1)/4 \rceil + next_carry} (A << 4i) \cdot F(C), \text{ where}$$

$$F(C) = \begin{cases} C, C<9 \\ (C\text{-}16), \text{ otherwise} \end{cases}$$

$$C = ((b_{(4i+3)} b_{(4i+2)} b_{(4i+1)} b_{2i} + prev_carry) \bmod 16)$$

$$next_carry = \begin{cases} 1, C > 8 \\ prev_carry, C=0 \\ 0, otherwise \end{cases}$$

פעולת המערכת מתחילה מאות start ברמה לוגית '1' המתקבל למחזור שעון אחד, המערכת מודיעה על סיום הפעולה באמצעות אות done העולה ל-'1' למחזור שעון אחד. באלגוריתם זה בכל מחזור מתבצעת פעולת חישוב של תוצאת ביניים (המתוארת בטבלה) על הנכפל במלואו ב-4 ביט תורניים (Current nibble) ונאגרת באוגר P. ה-Current nibble מהווה חיבור אריתמטי של 4 ביט הנמוכים (LSB) של הכופל וה-prev_carry (ממחזור קודם), במקביל מחושב ה-next_carry הנכפל זז שמאלה ב-4 ביט והכופל ימינה ב-4 ביט. ה-next_carry במחזור הנוכחי הוא ה-prev_carry במחזור הבא. פעולות אלה מתבצעות כל עוד גם הכופל שונה מ-0 או הכופל התנוון ל-0 וה-Current nibble שונה מ-0.

Current nibble	Action	next_carry
0000	No action	prev_carry
0001	+a	
0010	+2a	
0011	+3a (calculated beforehand)	
0100	+4a	
0101	+5a (calculated beforehand)	
0110	+6a (2*3a)	
0111	+7a (calculated beforehand)	
1000	+8a	
1001	-7a (calculated beforehand)	1
1010	-6a (2*3a)	1
1011	-5a (calculated beforehand)	1
1100	-4a	1
1101	-3a (calculated beforehand)	1
1110	-2a	1
1111	-a	1

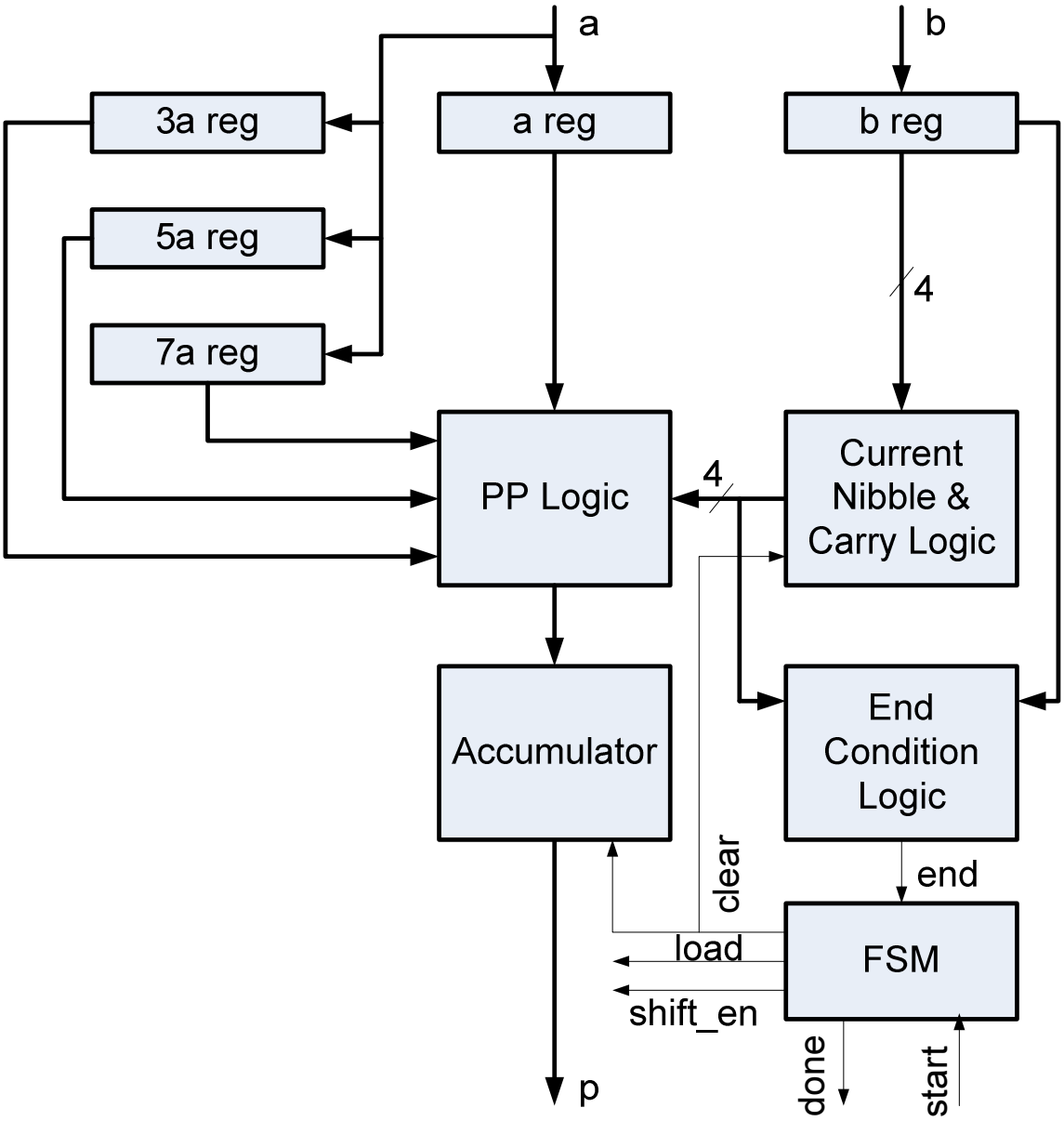
בטבלה לעיל משתנה a מתייחס לנכפל המוזז שמאלה בהתאם, 3a, 5a, 7a הם האוגרים אשר ערכם חושב מראש (בתחילת העבודה) ומוזזים שמאלה יחד עם האוגר של הנכפל.

המשימה: ממש מערכת הנ"ל בשלבים הבאים:

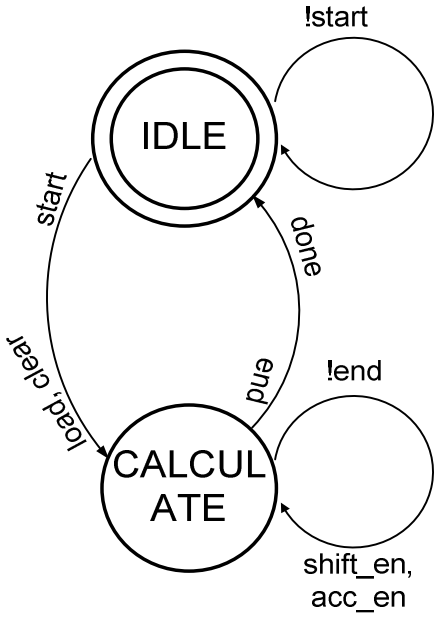
1. בנה ארכיטקטורה של המערכת ע"י חלוקה ליחידת בקרה הממומשת כ-FSM ויחידות ביצוע. שרטט דיאגראמת בלוקים של המערכת **(10 נקודות)**
2. שרטט דיאגראמת מצבים של יחידת בקרה **(10 נקודות)**
3. כתוב קוד VHDL מלא **(40 נקודות)**

קוד הנכתב אמור ליהות בר סינתזה!!!

דיאגרמת בלוקים של המערכת:



דיאגרמת מצבים של יחידת בקרה:



שאלה 2 -20 נקודות: לכתוב פונקציה רקורסיבית המקבלת כארגומנט bit_vector. פונקציה מחזירה bit_vector באותו הגודל אשר מהווה היפוך של ווקטור כניסה סביב המרכז. לדוגמה בעבור ווקטור "111000110" פונקציה מחזירה "011000111" . **יש לכתוב פונקציה הנתונה סינתזה.**

הפתרון:

```
function reverse_vec (d : std_logic_vector) return std_logic_vector is
   alias t : std_logic_vector(0 to (d'length-1)) is d;
   variable res : std_logic_vector(t'range);
begin
   case d'length is
      when 1 => res(0):=t(0);
      when 2 => res:=t(1) & t(0);
      when others => res:=t(t'right) & reverse_vec(t(1 to (t'right-1))) & t(t'left);
   end case;
   return res;
end function reverse_vec;
```

שאלה 3 -20 נקודות: נתונה מערכת צירופית המקבלת בשתי כניסות שלה a ו -b ערכים של 8 ביט המוצגים בקוד Johnson הכולל מספר רציף של ביטים ברמה לוגית '1' מוצמדים לימין, שאר הביטים נמצאים ב-'0'. לקוד של הכניסות מיוחסים ערכים עשרוניים לפי טבלה הבאה:

קוד כניסה	ערך עשרוני
00000000	0
00000001	1
00000011	2
00000111	3
00001111	4
00011111	5
00111111	6
01111111	7
11111111	8

מתבצע חיבור אריתמטי בין שני ערכים העשרוניים המיוחסים לערכים של כניסות , ותוצאת החיבור מוצגת במוצע C של 16 ביט לפי דוגמה הבאה: נתונים A="00001111" ו B="00111111" קוד של פורט A מזוהה עם ערך עשרוני 4, והקוד של פורט -B מזוהה עם ערך עשרוני 6. תוצאת החיבור האריתמטי בניהם היא 10. התוצאה תוצג במוצא באופן הבא: C="0000001111111111". במידע ולפחות אחד מהכניסות לא תוצג בקוד המוגדר בטבלה, אזי במוצא C מופיע "1000000000000000".

יש לכתוב תיאור של מערכת הנ"ל ב-VHDL באופן מלא, הכולל הצהרת ENTITY ו ARCHITECTURE הפתרון:

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_arith.all;
4  entity johnson_add is
5     port (a,b : in  std_logic_vector(7 downto 0);
6           c : out std_logic_vector(15 downto 0));
7  end entity johnson_add;
8  architecture behavioral of johnson_add is
9   signal tempa, tempb : integer range 0 to 15;
10  signal sum_ab       : integer range 0 to 31;
11 begin
12    process (a) is
13    begin
14       tempa<=15;
15       for i in 0 to 8 loop
16          if (((2**i)-1)=a) then
17             tempa<=i;
18          end if;
19       end loop;
20    end process;
21    process (b) is
22    begin
23       tempb<=15;
24       for j in 0 to 8 loop
25          if (((2**j)-1)=b) then
26             tempb<=j;
27          end if;
28       end loop;
29    end process;
30    sum_ab<= tempa + tempb;
31    c<=x"8000"  when (tempa=15 or tempb=15) else conv_std_logic_vector(((2**sum_ab)-1),c'range);
32 end architecture behavioral;
33
```