שפת תיאור חומרה VHDL

פתרון המבחן

שאלה 1 (60 נקודות)

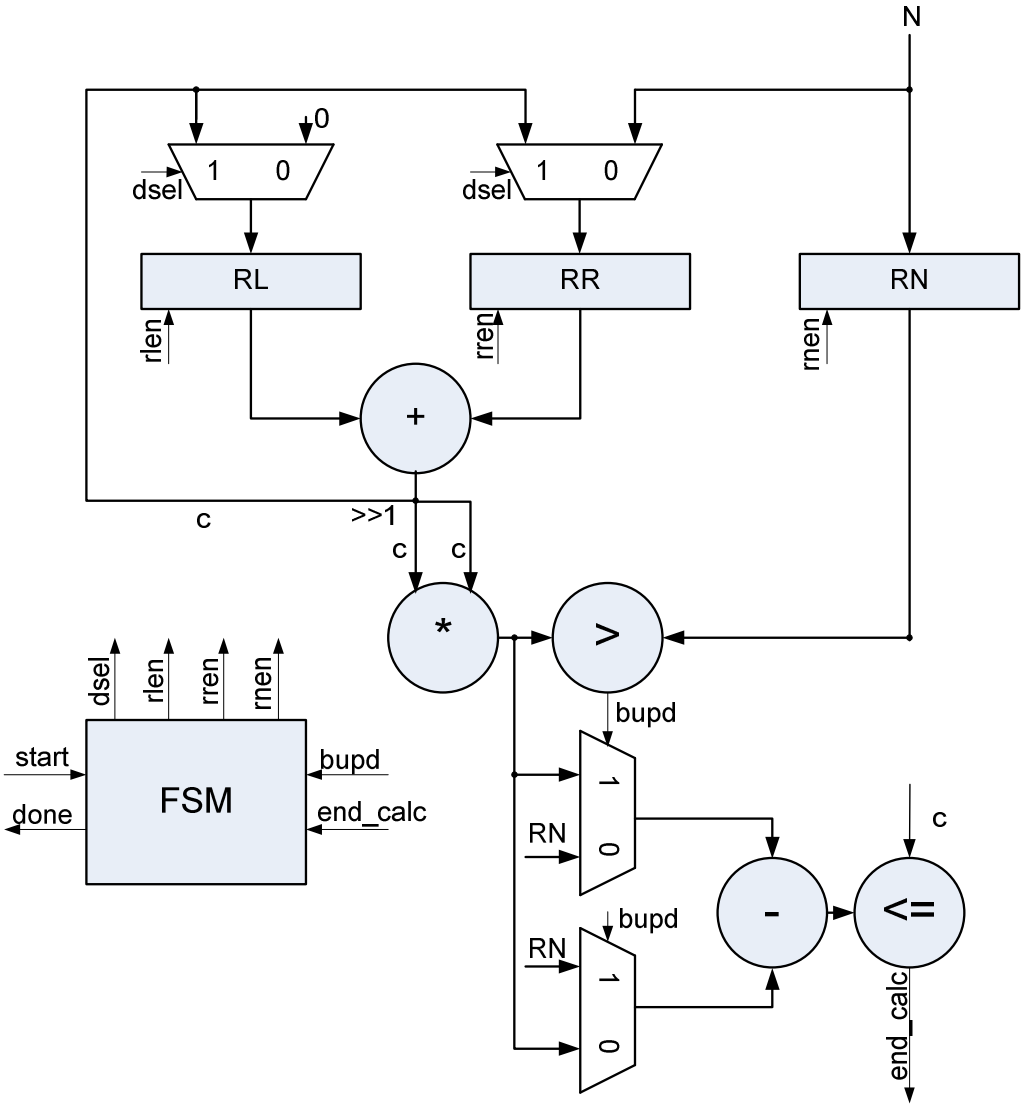
בשאלה זו נעסוק בתכן מערכת ספרתית וסינכרונית המחשבת שורש ריבועי ממספר שלם (**start_num**) בשיטת חיפוש בינארי המכונה גם "שיטת אריה במדבר" המפורטת כדלהלן. פעולה מתחילה כאשר מתקבל אות חיצוני **start** המגיע ל מחזור שעון אחד וערך בינארי **start_num** של 10 ביט. להלן אלגוריתם חיפוש בינארי המודגם ע"י חישוב שורש ממספר 50 (**start_num**). חשוב לציין ***שבמהלך החיפוש בכל איטרציה לפני ביצוע הפעולות הקשורות לחיפוש ניבדק תנאי העצירה*** המוזכר כדלהלן (סעיף 11):

1. נגדיר גבולות החיפוש. התוצאה נמצאת בין 0 (גבול השמאלי L) לבין 50 (גבול הימני R).
2. נבחר אמצע התחום החיפוש כתוצאת ביניים $C = (L+R)/2 = 25$.
3. נבדוק את רמת הקירוב של תוצאת ביניים בריבוע ל-**start_num**. $25^2 = 625 > 50$, תוצאת הביניים גדולה מה-**start_num** , לכן יש לצמצם את גבולות החיפוש. במקרה זה L נשאר ללא שינוי וה- R מקבל את C.
4. הגבול השמאלי (L) הוא 0, גבול הימני (R) הוא 25, ותוצאת ביניים החדשה היא $C = (L+R)/2 = 12$
5. נבדוק את רמת הקירוב של תוצאת ביניים בריבוע ל-**start_num**. $12^2 = 144 > 50$, שוב תוצאת ביניים גדולה מה-**start_num**, שוב יש לצמצם את גבולות החיפוש.
6. הגבול השמאלי (L) הוא 0, גבול הימני (R) הוא 12, ותוצאת ביניים החדשה היא $C = (L+R)/2 = 6$
7. נבדוק את רמת הקירוב של תוצאת ביניים בריבוע ל-**start_num**. $6^2 = 36 < 50$, הפעם תוצאת ביניים קטנה מה-**start_num**, יש לצמצם את גבול החיפוש, רק הפעם R נשאר ללא שינוי וה-L מקבל את C.
8. הגבול השמאלי (L) הוא 6, גבול הימני (R) הוא 12, ותוצאת ביניים החדשה היא $C = (L+R)/2 = 9$
9. נבדוק את רמת הקירוב של תוצאת ביניים בריבוע ל-**start_num**. $9^2 = 81 > 50$, שוב תוצאת ביניים גדולה מה-**start_num**, יש לצמצם את גבולות החיפוש.
10. הגבול השמאלי (L) הוא 6, גבול הימני (R) הוא 9, ותוצאת ביניים החדשה היא $C = (L+R)/2 = 7$
11. נבדוק את רמת הקירוב של תוצאת ביניים בריבוע ל-**start_num**. $7^2 = 49 < 50$, תוצאת הביניים קטנה מה-**start_num**, אך זו היא איטרציה האחרונה, כי הפעם מתקיים תנאי עצירת החיפוש $abs(C^2 - start_num) \le C$, לכן התוצאה הסופית היא 7.

המשימה: ממש מערכת הנ"ל בשלבים הבאים:

1. בנה ארכיטקטורה של המערכת ע"י חלוקה למבנים המבצעים את האלגוריתם וליחידת בקרה. שרטט דיאגראמת בלוקים מפורטת של המערכת **(10 נקודות)**.
2. שרטט דיאגראמת מצבים של-FSM של יחידת בקרה **(10 נקודות)**.
3. כתוב קוד של מערכת כולה ב-VHDL בסגנון מאפשר לסינתזה **(40 נקודות)**

דיאגראמת בלוקים של המערכת

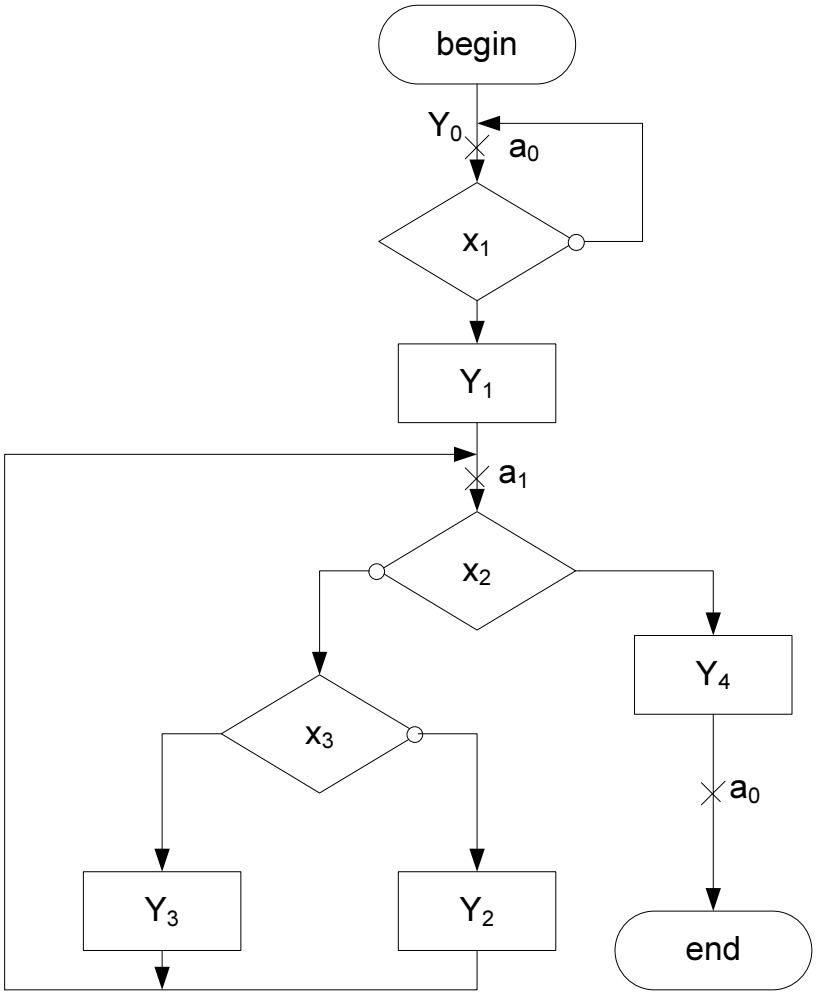


הגדרת של אותות בקרה, אותות סטאטוס ומילות בקרה

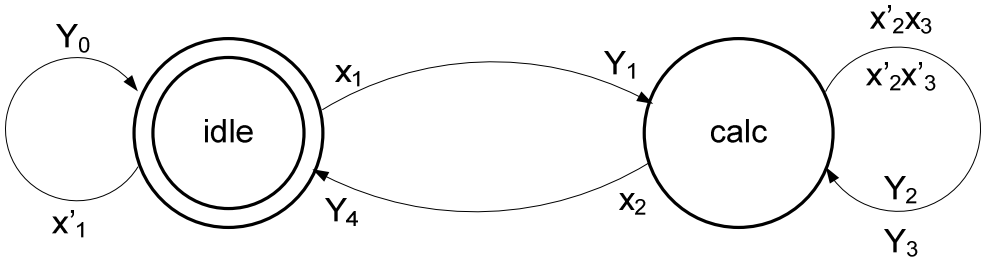
שם הסיגנל	start	end_calc	bupd
סימון ב-ASM	x_1	x_2	x_3

מילת בקרה	rlen	rren	rnen	dsel	done
Y_0	0	0	0	0	0
Y_1	1	1	1	0	0
Y_2	1	0	0	1	0
Y_3	0	1	0	1	0
Y_4	0	0	0	-	1

ASM של יחידת בקרה



דיאגראמת מצבים של ה-FSM של יחידת בקרה

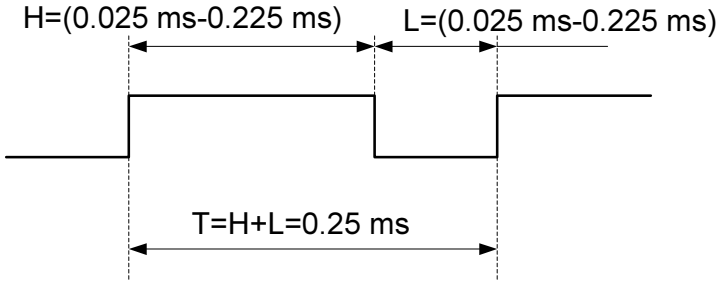


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity square_root is
   port (
      clk   : in  std_logic;
      rst   : in  std_logic;
      num   : in  std_logic_vector(9 downto 0);
      start : in  std_logic;
      res   : out std_logic_vector(9 downto 0);
      done  : out std_logic);
end entity square_root;

architecture arc_square_root of square_root is
   type   fsm_st is (idle, calculate);
   signal cs         : fsm_st;
   signal rl, rr, rn : std_logic_vector(num'range);
begin
   process (clk, rst) is
      variable rlen, rren, rnen, dsel,bupd,end_calc : boolean;
      variable rl_plus_rr         : std_logic_vector(10 downto 0);
      alias c                     : std_logic_vector(9 downto 0) is rl_plus_rr(10 downto 1);
      variable a,b,c_pow_2        : std_logic_vector(19 downto 0);
      variable rl_data, rr_data   : std_logic_vector(9 downto 0);
   begin
      if (rst = '0') then
         cs   <= idle;
         rl   <= (others => '0');
         rr   <= (others => '0');
         rn   <= (others => '0');
         done <= '0';
         res  <= (others => '0');
      elsif rising_edge(clk) then
            -- control signals default assignment
         done       <= '0';
         rlen       := false;
         rren       := false;
         rnen       := false;
         dsel       := false;
            ---------------------------------------
         rl_plus_rr := ('0' & rl) + ('0' & rr);
         c_pow_2    := c*c;
         bupd       := (c_pow_2 > ("0000000000" & rn));
```

```
        -- end of calculate condition logic abs(c**2-num)<=c
        if (bupd) then
          a := c_pow_2;
          b := "0000000000" & rn;
        else
          a := "0000000000" & rn;
          b := c_pow_2;
        end if;
        end_calc := ((a-b) <= ("0000000000" & c));
        -- control fsm----------------------
        case cs is
          when idle => if (start = '1') then
                         rlen := true;
                         rren := true;
                         rnen := true;
                         cs   <= calculate;
                       end if;
          when calculate => if (end_calc) then
                              done <= '1';
                              cs   <= idle;
                            elsif (bupd) then
                              dsel := true;
                              rren := true;
                            else
                              dsel := true;
                              rlen := true;
                            end if;
          when others => null;
        end case;
        -- left & right bounds select and registers
        if (dsel) then
          rl_data := c;
          rr_data := c;
        else
          rl_data := (others => '0');
          rr_data := num;
        end if;
        if (rlen) then
          rl <= rl_data;
        end if;
        if (rren) then
          rr <= rr_data;
        end if;
        if (rnen) then
          rn <= num;
        end if;
        res<=c;
      end if;
    end process;
  end architecture arc_square_root;
```

שאלה 2 (40 נקודות)

ממש מערכת ספרתית סינכרונית הפועלת לפי השעון אשר זמן המחזור שלו מוגדר באמצעות generic מטיפוס time, לדוגמה בעבור השעון של 100MHz - ; **generic** (clock_period: **time**:=10 **ns**) . מערכת זו מפעילה מנוע סרוו (servo) לפי השיטה PWM (Pulse Width Modulation). באיור 1 מוצגת דוגמה של דיאגראמת זמנים של מוצא המערכת בתדר 4 KHz (מחזור 0.25 ms) זמן זה גם יקבע באמצעות generic. באות המוצא ה-duty-cycle משתנה בצורה מבוקרת ע"י אותות up_ndwn (Up not Down) וה- sc_en (Speed Change Enable) בגבולות בין 10% לבין 90% מהמחזור המלא בצעדים של 10%. בהתקבל הצירוף sc_en =1 וה-up_ndwn =1 המהירות גדלה (הduty-cycle משתנה כלפי מעלה אך אינו עובר את ה-90%) ובצירוף sc_en =1 וה- up_ndwn =0 המהירות קטנה (הduty-cycle משתנה כלפי מטה אך אינו עובר את ה-10%), כאשר sc_en =0 המערכת אינה משנה את אות המוצא. בעת ההפעלה, מערכת מייצרת אות עם duty-cycle 50%.



איור 2.

<u>המשימה:</u>

ממש מערכת הנ"ל בשלבים הבאים:

1. בנה ארכיטקטורה של המערכת ע"י חלוקה לתת-מבנים.
   שרטט דיאגראמת בלוקים מפורטת של המערכת **(10 נקודות)**
2. כתוב קוד VHDL הנתון לסינתזה **(30 נקודות)**

```
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 pwm is
7     generic( CLK_PERIOD : time := 20 ns; PWM_PERIOD : time := 250 us);
8     port(
9        clk          : in  std_logic;
10       rst          : in  std_logic;
11       speed_change : in  std_logic;
12       up_ndwn      : in  std_logic;
13       pwm_out      : out std_logic
14       );
15 end entity pwm;
16
17 architecture arc_pwm of pwm is
18    constant clk_div_ratio   : integer := PWM_PERIOD/CLK_PERIOD;
19    constant pwm_hp          : integer := clk_div_ratio/2;
20    constant pwm_low_limit   : integer := clk_div_ratio/10;
21    constant duty_cycle_step : integer := pwm_low_limit;
22    constant pwm_high_limit  : integer := clk_div_ratio-pwm_low_limit;
23    function log2 (data      : integer) return integer is
24    begin
25       for i in 0 to 31 loop
26          if ((2**i) >= data) then
27             return i;
28          end if;
29       end loop;
30       return 1;
31    end function log2;
32    signal clock_cnt,duty_cycle_cnt: std_logic_vector(log2(clk_div_ratio) downto 1);
33 begin
34    process(clk, rst) is
35    begin
36       if(rst = '0') then
37          clock_cnt      <= (others => '0');
38          duty_cycle_cnt <= conv_std_logic_vector(pwm_hp, duty_cycle_cnt'length);
39          pwm_out        <= '0';
40       elsif (rising_edge(clk)) then
41          if(clock_cnt = clk_div_ratio) then
42             clock_cnt <= (others => '0');
43          else
44             clock_cnt <= clock_cnt+1;
45          end if;
46          if (speed_change = '1') then
47             if (up_ndwn = '1') then
48                if(duty_cycle_cnt < pwm_high_limit) then
49                   duty_cycle_cnt <= duty_cycle_cnt + duty_cycle_step;
50                end if;
51             elsif (duty_cycle_cnt > pwm_low_limit) then
52                duty_cycle_cnt <= duty_cycle_cnt - duty_cycle_step;
53             end if;
54          end if;
55          if(clock_cnt < duty_cycle_cnt) then
56             pwm_out <= '1';
57          else
58             pwm_out <= '0';
59          end if;
60       end if;
61    end process;
62 end architecture arc_pwm;
63
```