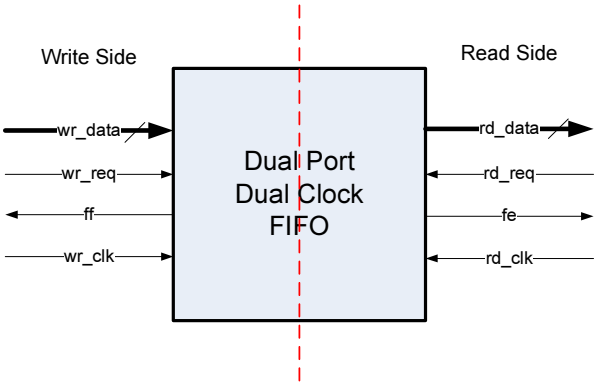
פתרון המבחן (חלקי)

שאלה 1 (75 נקודות)

תכנן FIFO (רכיב זיכרון הפועל לפי המשטר First In First Out המכונה גם טור) המתווך בין שני מערכות הפועלות בקצבים שונים. ה-FIFO מוצג באיור 1.



איור 1

משמעות של כל פורט של FIFO מפורטת בטבלה הבאה:

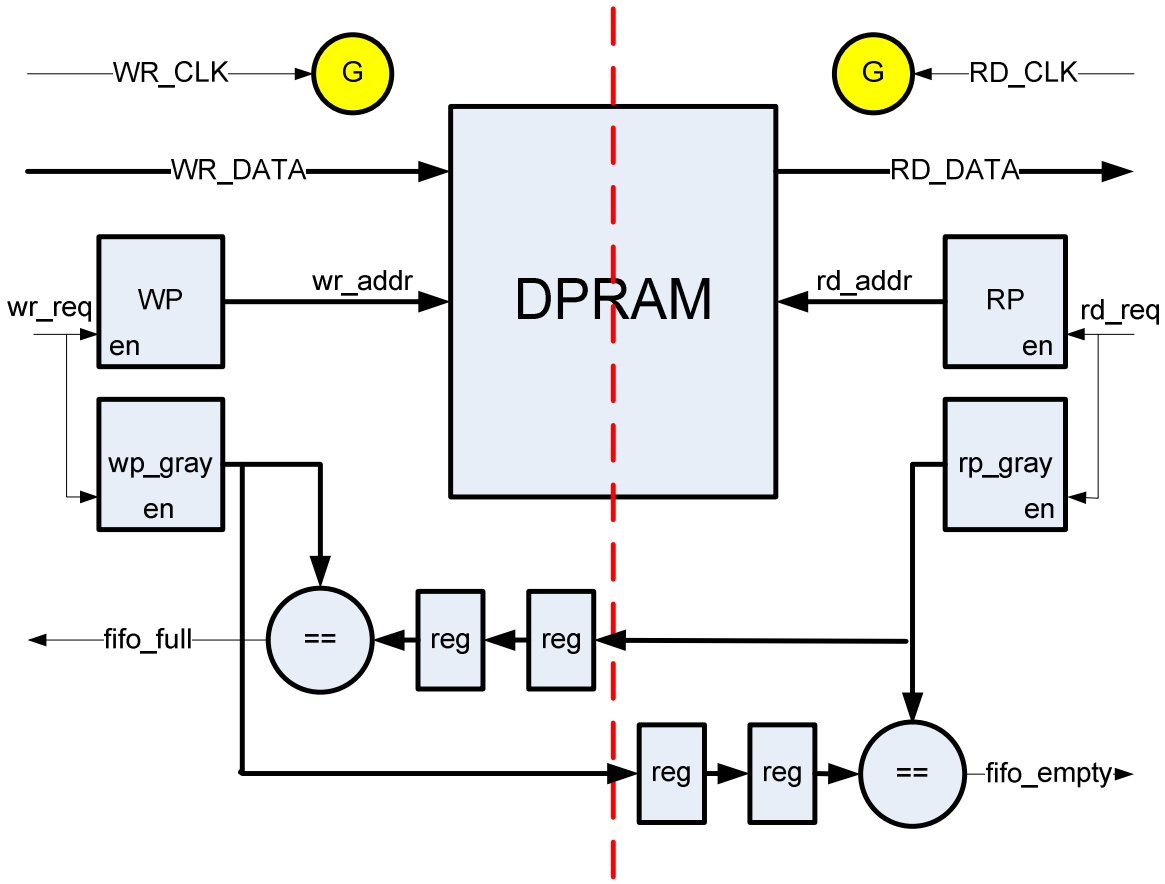
Write side		Read side	
wr_clk	Write clock	rd_clock	Read clock
wr_req	Write request	rd_req	Read request
ff	Fifo full flag	fe	Fifo empty flag
wr_data	Write data	rd_data	Read data

במימוש ה-FIFO יש להשתמש בטיפוס מערך לייצוג זיכרון הפנימי של ה-FIFO כאשר גודל המילה בזיכרון (FIFO_WIDTH) ועומק ה-FIFO (FIFO_DETPH) קרי מספר המילים המרבי היכול להימצא ב-FIFO נקבעים ע"י **generic** וה-FIFO_DETPH הוא חיזקה שלמה של 2. אין למנוע (בלוגיקה של ה-FIFO) כתיבה ל-FIFO במצב **ff** וקריאה מה-FIFO במצב **fe**. יש להתייחס לעובדה שה-FIFO משני צדדיו מנוהל ע"י **<u>שעונים שונים</u>** **!!!**, דבר המשפיע על עדכון ה-**ff** וה-**fe**. על מנת לממש לוגיקה של ה-full וה-empty יש לנהל בנוסף למונים המשמשים ככתובות לכתיבה **wr_pointer** (בצד הכתיבה) והקריא **rd_pointer** (בצד הקריא) עוד שני מונים ב-Gray Code העוברים בהצלבה מצד לצד. מונים אלה נדגמים פעמיים בכל צד לפי השעון שלו ורק אז מתבצעת בדיקה הקובעת את ה-**ff** וה-**fe**.

<u>המשימה:</u> ממש מערכת הנ"ל בשלבים הבאים:

1. שרטט דיאגראמת בלוקים מפורטת של המערכת **(10 נקודות)**
2. כתוב קוד בר-סינתזה של המערכת כולה ב-VHDL **(40 נקודות)**
3. כתוב Test-Bench הבודק את ה-FIFO, הבדיקה חייבת להתייחס למצבי ה-ff וה-fe **(25 נקודות)**

דיאגראמת בלוקים של ה-FIFO



```
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 dp_fifo is
7     generic (
8        FIFO_WIDTH: integer; FIFO_DEPTH_LOG2: integer);
9     port (
10       rst     : in  std_logic;
11       -- write side
12       wr_clk     : in  std_logic;
13       wr_data : in  std_logic_vector((FIFO_WIDTH-1) downto 0);
14       wr_req  : in  std_logic;
15       ff      : out boolean;
16       -- read side
17       rd_clk     : in  std_logic;
18       rd_data : out std_logic_vector((FIFO_WIDTH-1) downto 0);
19       rd_req  : in  std_logic;
20       fe      : out boolean);
21 end entity dp_fifo;
22
23 architecture arc_dp_fifo of dp_fifo is
24 type ram is array (((2**FIFO_DEPTH_LOG2)-1) downto 0) of std_logic_vector(wr_data'range);
25 signal fifo_buff : ram;
26 signal wp,wp_gray,wp_gray_s0,wp_gray_s1 : std_logic_vector(FIFO_DEPTH_LOG2 downto 0);
27 signal rp,rp_gray,rp_gray_s0,rp_gray_s1 : std_logic_vector(FIFO_DEPTH_LOG2 downto 0);
28 alias wr_addr : std_logic_vector((FIFO_DEPTH_LOG2-1)downto 0) is wp((FIFO_DEPTH_LOG2-1) downto 0);
29 alias rd_addr : std_logic_vector((FIFO_DEPTH_LOG2-1)downto 0) is rp((FIFO_DEPTH_LOG2-1) downto 0);
30 begin
31    wr_side: process (wr_clk, rst) is
32       variable bcnt, nextb : std_logic_vector(FIFO_DEPTH_LOG2 downto 0);
33    begin
34       if (rst = '0') then
35          wp<=(others => '0');
36          wp_gray<=(others => '0');
37          rp_gray_s0<=(others => '0');
38          rp_gray_s1<=(others => '0');
39       elsif rising_edge(wr_clk) then
40          if (wr_req='1') then
41             fifo_buff(conv_integer(wr_addr))<=wr_data;
42             wp<=wp+1;
43          end if;
44          for i in bcnt'range loop
45             if (i=bcnt'high) then
46                bcnt(i):=wp_gray(i);
47             else
48                bcnt(i):=wp_gray(i) xor bcnt(i+1);
49             end if;
50          end loop;
51          nextb:=bcnt+wr_req;
52          wp_gray<=nextb xor ('0' & nextb(nextb'left downto 1));
53          rp_gray_s0<=rp_gray;
54          rp_gray_s1<=rp_gray_s0;
55       end if;
56    end process wr_side;
57
58    rd_side: process (rd_clk, rst) is
59       variable bcnt, nextb : std_logic_vector(FIFO_DEPTH_LOG2 downto 0);
60    begin
61       if (rst = '0') then
62          rp<=(others => '0');
63          rp_gray<=(others => '0');
64          wp_gray_s0<=(others => '0');
65          wp_gray_s1<=(others => '0');
66          rd_data<=(others => '0');
67       elsif rising_edge(rd_clk) then
68          if (rd_req='1') then
69             rd_data<=fifo_buff(conv_integer(rd_addr));
70             rp<=rp+1;
71          end if;
72          for i in bcnt'range loop
73             if (i=bcnt'high) then
74                bcnt(i):=rp_gray(i);
75             else
76                bcnt(i):=rp_gray(i) xor bcnt(i+1);
77             end if;
78          end loop;
79          nextb:=bcnt+rd_req;
80          rp_gray<=nextb xor ('0' & nextb(nextb'left downto 1));
81          wp_gray_s0<=wp_gray;
82          wp_gray_s1<=wp_gray_s0;
83       end if;
84    end process rd_side;
85    ff<= (wp_gray(wp_gray'left)/=rp_gray_s1(rp_gray_s1'left)) and
86          (wp_gray((wp_gray'left-1) downto 0)=rp_gray_s1((rp_gray_s1'left-1) downto 0));
87    fe<= (rp_gray=wp_gray_s1);
88 end architecture arc_dp_fifo;
89
90
```

שאלה 2 (25 נקודות)

יש לכתוב שתי פונקציות הבאות:

1. **(15 נקודות)** פונקציה המקבלת מערך של שלושה **std_logic_vector**(7 **downto** 0) ומחזירה תוצאה מטיפוס **std_logic_vector**. הפונקציה מחשבת חציון (median) של המערך הנתון, כלומר ערך שהוא גדול או שווה לערך המזערי במערך ושווה או קטן מהערך המרבי במערך. פונקציה זו משמשת כפונקצית עזר לפונקצית אם מסעיף הבא.
2. **(10 נקודות)** פונקציה אשר מחשבת ומחזירה (באמצעות הטיפוס **std_logic_vector**) חציון במערך שאלמנט בסיסי שבו הוא מערך מסעיף קודם, כלומר הארגומנט של הפונקציה הוא מערך של מערכים (מטריצה).

על מנת לממש פונקציות הנ"ל, תחילה יש להגדיר טיפוס מסוג מערך בעבור כל פונקציה בהתאם למוסבר לעיל. במימוש הפונקציות אין להגביל ארגומנטים, כלומר אין לקבוע מה הוא ה-range של הארגומנט, הפונקציות אמורות לקרוא מהארגומנט את המאפיינים (attributes) שלו ולהשתמש בהם.

```
type curr_color is array (natural range <>) of bit_vector(7 downto 0);
type matrix2d is array (0 to 2) of curr_color(0 to 2);

function padding (arg : matrix2d) return bit_vector is
   variable buff : curr_color(0 to 2);
   function median(d : curr_color(0 to 2)) return bit_vector is
      variable temp : bit_vector(7 downto 0);
   begin
      if (d(2) >= d(1) and d(2) <= d(0)) then
         temp:= d(2);
      elsif (d(2) >= d(0) and d(2) <= d(1)) then
         temp:= d(2);
      elsif (d(1) >= d(0) and d(1) <= d(2)) then
         temp:= d(1);
      elsif (d(1) >= d(2) and d(1) <= d(0)) then
         temp:= d(1);
      elsif (d(0) >= d(1) and d(0) <= d(2)) then
         temp:= d(0);
      elsif (d(0) >= d(2) and d(0) <= d(1)) then
         temp:= d(0);
      else
         temp:= d(0);
      end if;
      return temp;
   end function median;
begin
   for i in arg'range loop
      buff(i):=median(arg(i));
   end loop;
   return median(buff);
end function padding;
```