

**המחלקה להנדסת חשמל ואלקטרוניקה**

**התקנים לוגיים מתוכננים 31551**

**פרויקט סוף**

**תאריך : 02.07.2023**

תוכן עניינים:

3.....	הקדמה.....
4.....	סכמת בЛОקים.....
5.....	בלוק- push_button
9.....	בלוק- controller
14.....	בלוק- timing_generator
18.....	בלוק - data_generator
24.....	בלוק - clock_generator
26.....	בלוק- two_flop_synchronizer
27.....	בלוק- .bcd_to_7seg
28.....	בלוק - image_processor
38.....	תיאור תקלות.....
39.....	סיכום ומסקנות.....

## דרישות הפרויקט

### רעיון כללי:

היעוד הראשי של המערכת הוא: להציג תמונה על מסך HDMI ולסובב אותה. את המידע שמננו התמונה מרכיבת נשמר בזיכרון SRAM, הנמצא ברכיב ה-FPGA. בעזרת הרכיב נשלוט על סיבוב התמונה בין אם זה בצורה אוטומטית או ידנית, נשלוט על כיוון הסיבוב (עם או נגד כיוון השעון).

### דרישות טכניות:

המערכת עובדת בתדר שעון [MHz].25.

שנה כניסה איפוס אשר עובדת בנמו.

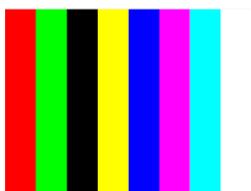
לחץ (key3) העובד בנמו אשר קובע אם נכנסים לסיבוב אוטומטי מעל 2 שניות כל עוד לחצים עליו או סיבוב ידני אם נלחץ על הלחץ לפחות מ2 שניות.

לחץ (key0) מאפס את המערכת.

לחץ (key9) מחליף לנו את התצוגה על המסך בkr שמאלה לנו תצוגת צבעים.

מפסק (key4) סיבוב אוטומט .

מפסק (key2) קובע את כיוון סיבוב.



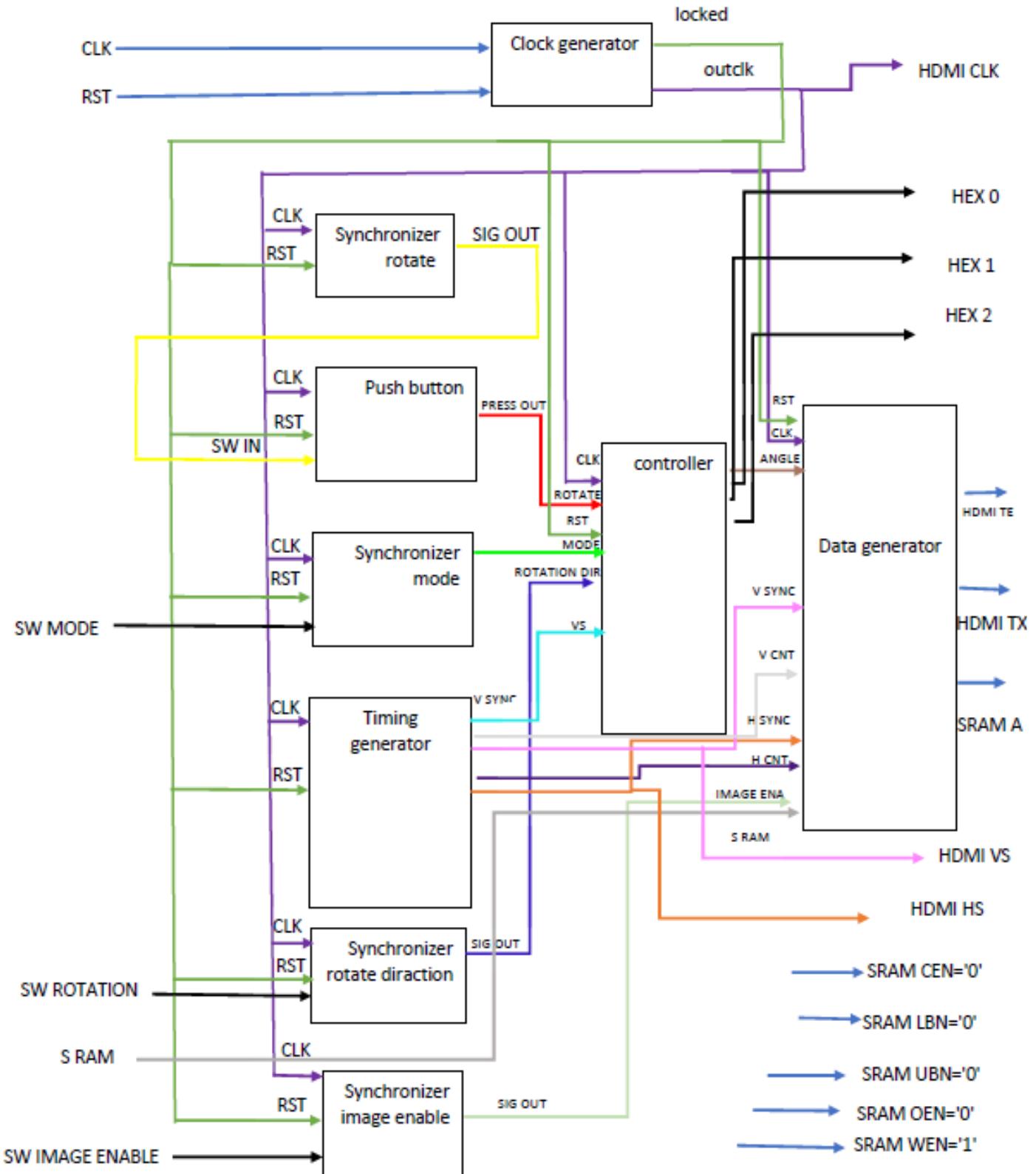
### ציוד נדרש:

ערכת Cyclone V Starter Kit

מסך HDMI + קבל HDMI

.מחשב עם תוכנת MODELSIM ו-QUARTUS-II

## סכמת בלוקים



## בלוק: push boutton

### הסבר:

בבלוק push boutton מישנו באמצעות מכונת מצבים.

הגדנו 4 מצבים :

1. Idle מושך במצב התחלתי
2. Press מושך במצב לחיצה
3. sec pulse מושך במצב של לחיצה ארוכה
4. clk pulse מושך במצב של לחיצה קצרה

המצב הראשון נקרא idle מצב זה הוא התחלתי כאשר אנו נלחץ על הלוח נקלט נüber למצב לחיצה (press) , במצב זה אנו בודקים את משך זמן הלחיצה במידה והלחיצה תהיה קצרה מ-2 שניות אנו עבור במצב clk pulse שטרכתו להוציא פולס לפי השעון של הבלוק. במידה וזמן הלחיצה יהיה גדול מ-2 שניות אנו עוברים במצב של sec pulse שטרכתו להוציא פולס בכל שנייה ככלمر אנחנו נחכה 2 שניות ואז עוברים במצב sec pulse עד שנייה ואז יוציא פולס בכל שנייה.

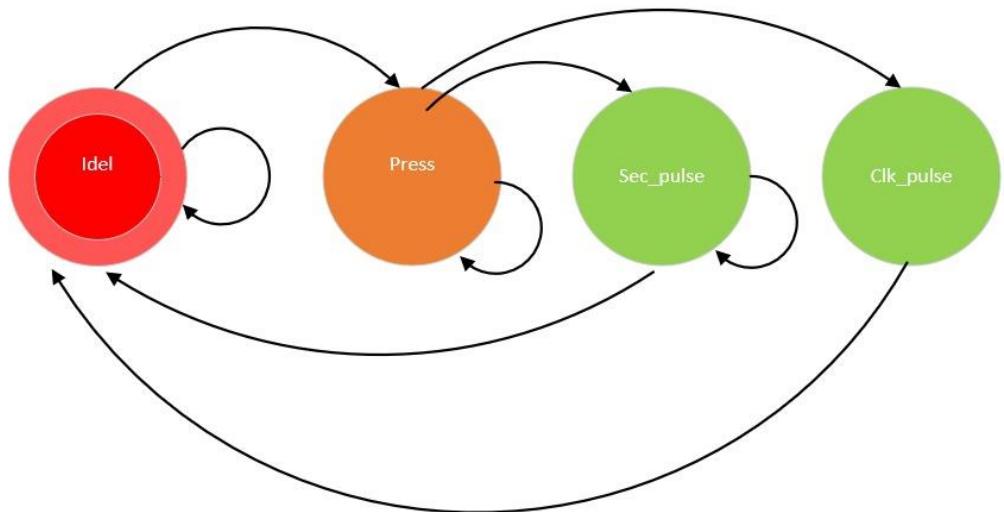
### סיכום משאבים:

Flow Status	Successful - Wed Jun 28 15:00:22 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	push_button
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	31 / 29,080 (< 1 % )
Total registers	43
Total pins	4 / 364 ( 1 % )
Total virtual pins	0
Total block memory bits	0 / 4,567,040 ( 0 % )
Total DSP Blocks	0 / 150 ( 0 % )
Total HSSI RX PCSS	0 / 6 ( 0 % )
Total HSSI PMA RX Deserializers	0 / 6 ( 0 % )
Total HSSI TX PCSS	0 / 6 ( 0 % )
Total HSSI PMA TX Serializers	0 / 6 ( 0 % )
Total PLLs	0 / 12 ( 0 % )
Total DLLs	0 / 4 ( 0 % )

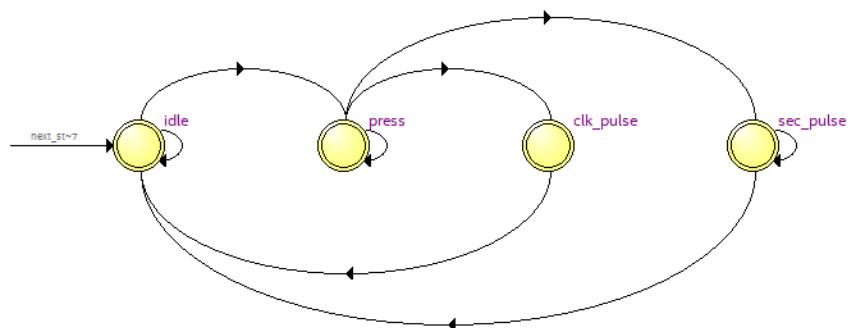
### סיכום עמידה בתדר – Fmax

Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	260.28 MHz	260.28 MHz	CLK	

מכונת מצבים:

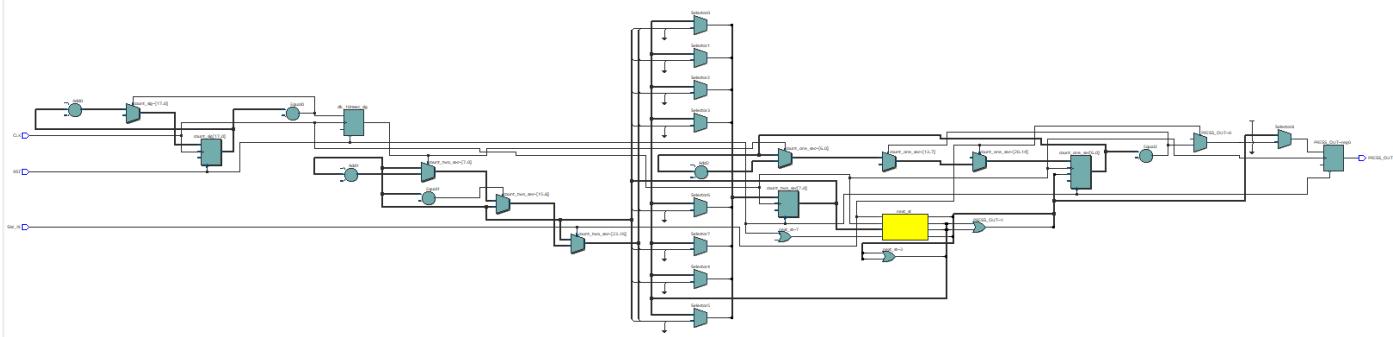


מכונת מצבים של ה-quartos:

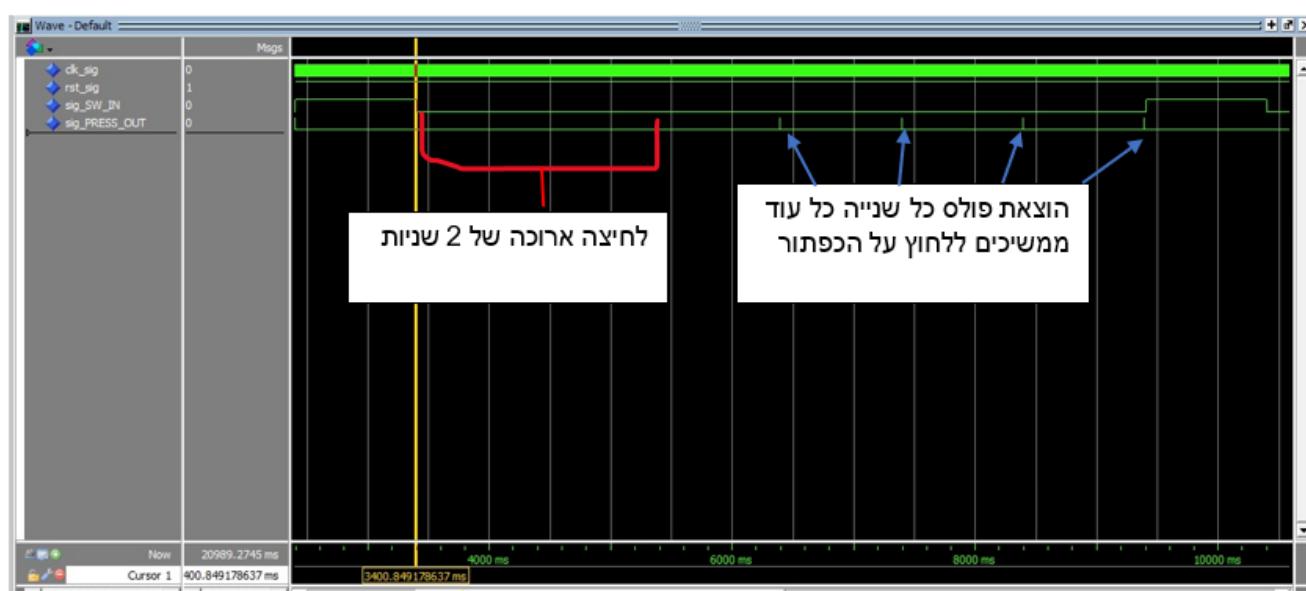


	Name	sec_pulse	clk_pulse	press	idle
1	idle	0	0	0	0
2	press	0	0	1	1
3	clk_pulse	0	1	0	1
4	sec_pulse	1	0	0	1

## מבנה לוגי של הבלוק:



## סימולציה של הבלוק:



## :Test bench

```
push_button_tb.vhd [3]
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.data_image_package.all;
4  entity push_button_tb is
5  end entity;
6
7  architecture behave of push_button_tb is
8  -- constants declaration
9  constant C_CLK_PRD           : time := 40 ns;
10 component push_button is
11   port (
12     RST      : in std_logic;
13     CLK      : in std_logic;
14     SW_IN    : in std_logic;
15     PRESS_OUT : out std_logic
16   );
17 end component;
18
19 signal clk_sig      : std_logic := '0';
20 signal rst_sig      : std_logic := '0';
21 signal sig_SW_IN   : std_logic;
22 signal sig_PRESS_OUT : std_logic;
23
24 begin
25   dut: push_button
26     port map (
27       RST      => rst_sig,
28       CLK      => clk_sig,
29       SW_IN    => sig_SW_IN,
30       PRESS_OUT => sig_PRESS_OUT
31     );
32
33   clk_sig <= not clk_sig after C_CLK_PRD / 2;
34   rst_sig <= '0', '1' after 20 ns;
35
36   process
37   begin
38     -----Short press-----
39     for i in 0 to 5 loop
40       sig_SW_IN<='1' after 0 ms, '0' after 20 ms, '1' after 60 ms,'0' after 110 ms,
41       '1' after 200 ms,'0' after 300 ms;
42       wait for 400 ms;
43     end loop;
44
45     -----Long press-----
46     for i in 0 to 2 loop
47       sig_SW_IN<='1' after 0 ns, '0' after 1 sec, '1' after 7 sec;
48       wait for 7 sec;
49     end loop;
50     wait;
51   end process;
52 end architecture;
```

## בלוק Controller:

### הסבר:

S7 מייצג לנו ENABLE והוא מגייע מ- timing generator ואומר לנו מתי יהיה הסוף של הפרים.

לכן רק כאשר S7=1 ניתן להיכנס לשאר הממצבים הבאים :

כאשר אנחנו נמצאים במצב MODE=1 זאת אומרת שהמפסק עלה ל1 ואז מתחילה העיבר את הזריות בכל שנייה ככלمر יעבד על מצב אוטומט, לאחר בדיקת המוד אנו נבדוק באיזה מצב נמצא המפסק dir rotation שטרתו לקבוע את כיוון הסיבוב של התמונה (נגד השעון או עם כיוון השעון) , לאחר שבדקנו את כל הממצבים אנו מגדרים את הזריות ככלמר על כל תצוגת התמונה הבלוק יראה לנו באיזה זווית אנחנו (0,90,180,270).

כאשר MODE=0 אז אנחנו נכנסים למצב ידני ובודק באיזה תנאי נמצא הrotate שישתמש אותנו כלחץ שנוכל להפוך ידני את התמונה לאחר שבדקנו את ה- rotate נבדוק את באיזה מצב נמצא dir rotation כפי שהזכירנו מטרתו לקבוע את הכיוון של היפוך התמונה ולאחר מכן יציב את הזריות הנדרשת.

הערה: נציין שעשינו מיפוי של הבלוק 7seg\_BCD לבלוק זה על מנת לשלוח את העריכים של הזריות למכשיר האALTERה במקום להגדירו בرمאה העליונה.

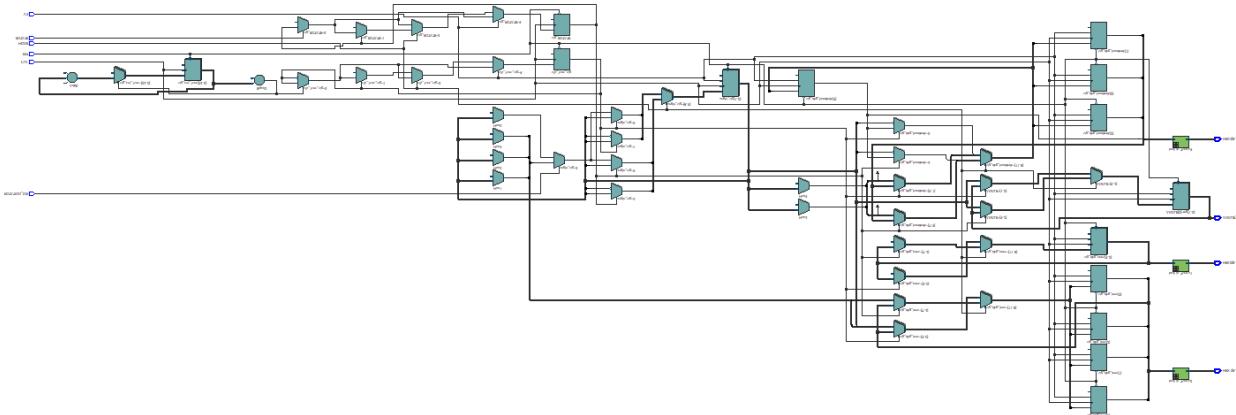
### סיכום משאבים:

Flow Status	Successful - Wed Jun 28 14:41:57 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	controller
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	29 / 29,080 (< 1 %)
Total registers	39
Total pins	29 / 364 (8 %)
Total virtual pins	0
Total block memory bits	0 / 4,567,040 (0 %)
Total DSP Blocks	0 / 150 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 4 (0 %)

### סיכום עמידה בתדר – Fmax :

Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	245.88 MHz	245.88 MHz	CLK	

## מבנה לוגי של הבלוק:



## סימולציה של הבלוק:

Mode=1 מצב אוטומטי

270 מעילות

כניסת ריבט  
(עובד  
בנמוך)

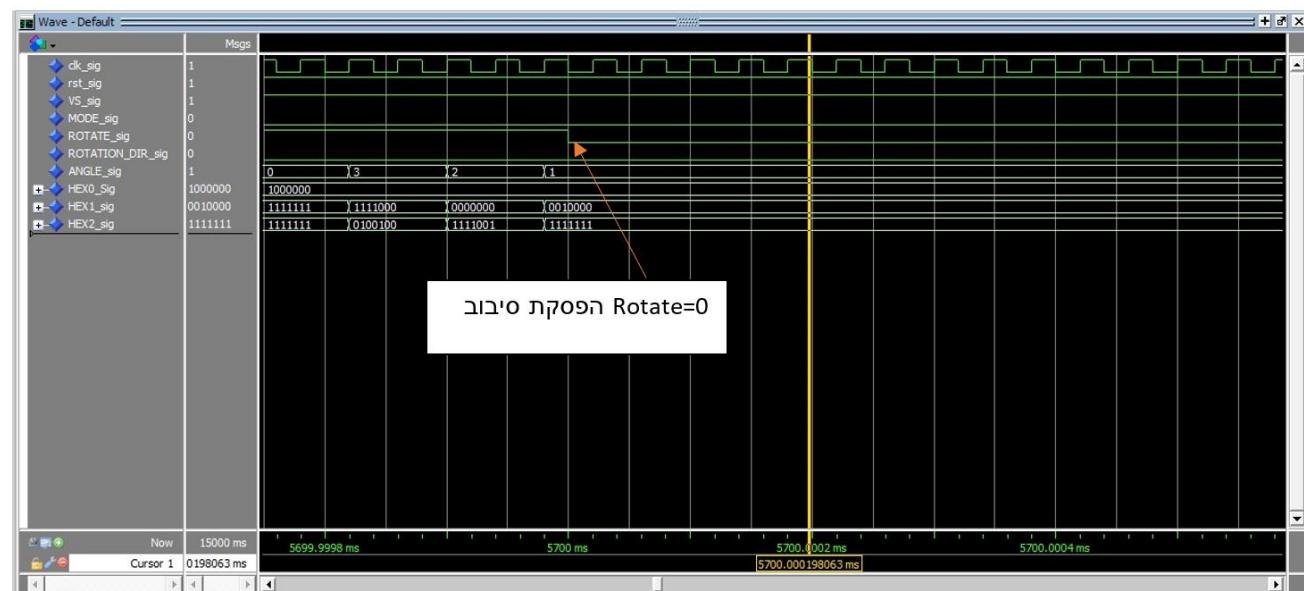
Rotation dir=0  
סיבוב עם כיוון  
השעון

תחילת סיבוב כל  
שנייה עם כיוון השעון

180 מעילות



Mode=0 מצב ידני



## :Test bench

```
controller_tb.vhd x]
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use work.data_image_package.all;
4 entity controller_tb is          -- The Testbench entity is empty. No ports.
5 end entity;
6
7 architecture behave of controller_tb is    -- This is the architecture of the testbench
8
9 -- constants declaration
10 constant C_CLK_PRD      : time := 40 ns;
11
12 component controller is           -- This is the component declaration.
13 port (
14     CLK      : in std_logic; -- clock
15     RST      : in std_logic; -- reset
16     VS       : in std_logic;
17     MODE     : in std_logic;
18     ROTATE   : in std_logic;
19     ROTATION_DIR : in std_logic;
20     ANGLE    : out integer range 0 to 3;
21     HEX0     : out std_logic_vector(6 downto 0);
22     HEX1     : out std_logic_vector(6 downto 0);
23     HEX2     : out std_logic_vector(6 downto 0)
24 );
25 end component;
26
27
28 -- signals declaration
29 signal clk_sig      : std_logic := '0';
30 signal rst_sig      : std_logic := '0';
31 signal VS_sig       : std_logic:= '0';
32 signal MODE_sig     : std_logic;
33 signal ROTATE_sig   : std_logic;
34 signal ROTATION_DIR_sig : std_logic;
35 signal ANGLE_sig    : integer range 0 to 3;
36 signal HEX0_Sig     : std_logic_vector(6 downto 0);
37 signal HEX1_sig      : std_logic_vector(6 downto 0);
38 signal HEX2_sig      : std_logic_vector(6 downto 0);
39
40 begin
41
42     dut: controller
43     port map (
44         rst  => rst_sig,
45         clk  => clk_sig,
46         VS   => VS_sig,
47         MODE => MODE_sig, -- output
48         ROTATE=>ROTATE_sig,
49         ROTATION_DIR=>ROTATION_DIR_sig,
50         ANGLE=>ANGLE_sig,
51         HEX0=>HEX0_Sig,
52         HEX1=>HEX1_sig,
53         HEX2=>HEX2_sig
54     );
55 
```

```
56 clk_sig <= not clk_sig after C_CLK_PRD /2;      -- clk_sig toggles every C_CLK_PRD/2 ns
57 rst_sig <= '0', '1' after 100 ns;
58 VS_sig <='0' after 0 ns , '1' after  50 ns,  '0' after  5 sec,'1' after  5510 ms;
59 process
60 begin
61   for i in 0 to 10 loop
62     MODE_sig<='1' after 0 ns , '0' after  4 sec;
63     ROTATE_sig<='0' after 0 ns , '1' after  2000 ms;
64     ROTATION_DIR_sig<='0' after 0 ns,  '1' after  3000 ms;
65     wait for 500 ms;
66   end loop;
67
68   for i in 0 to 10 loop
69     MODE_sig<='0' after 0 ns, '1' after  2 sec;
70     ROTATEE_sig<='0' after 0 ns , '1' after  100 ms,  '0' after  300 ms,'1' after  600 ms,
71     '0' after 700 ms , '1' after  800 ms,  '0' after  1000 ms,'1' after  1100 ms;
72     ROTATION_DIR_sig<='0' after 0 ns , '1' after  500 ms,  '0' after  700 ms,'1' after  1000 ms;
73     wait for 200 ms;
74   end loop;
75   wait;
76 end process;
77 end architecture;
```

## בלוק : TIMING GENERATOR

### הסבר:

בבלוק זה אנו מייצרים מונה אופקי CNT\_H סופר את מספר הפיקסלים ( פיקסלים 800 ) במסך .  
 מונה אנכי CNT\_V סופר את מספר השורות ( 525 שורות ) במסך .  
 בעזרתם ניתן לבדוק את המיקום הנוכחי ולתחום את המספר , ובכך לדעת האם נמצאים בתחום הנראה על ידי שימוש ב chs\_h sync\_v .  
 כאשר chs\_h אומר לנו שנגמר שורה ו- chs\_v אומר לנו שנגמר הפרים .  
 ככלומר ברגע שנקלט "0" לוגי באחד מהם זאת אומרת שאנו בתחום לא ניראה של המסך .  
 בנוסף לכך יש לנו פולס זז שמשמש אותנו כגוזר כלומר עולה ל 1 לוגי  
 (למישר מחזור אחד של השעון) כאשר נגמר הפרים .

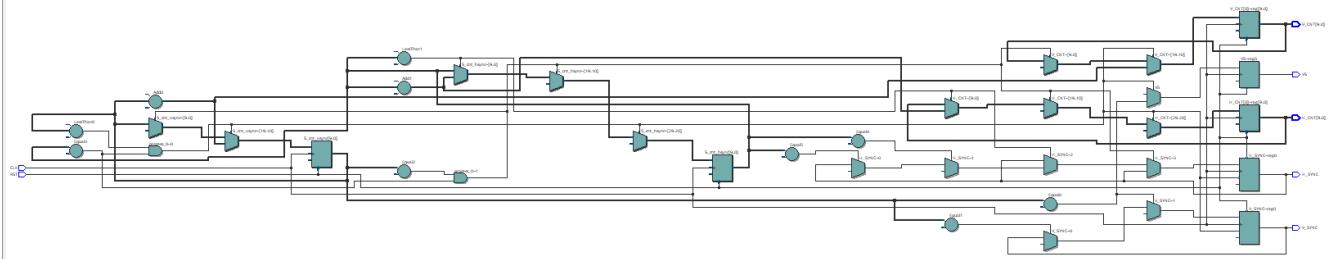
### סיכום משאבים:

Flow Status	Successful - Wed Jun 28 14:55:45 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	timing_generator
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	21 / 29,080 (< 1 %)
Total registers	32
Total pins	25 / 364 ( 7 %)
Total virtual pins	0
Total block memory bits	0 / 4,567,040 ( 0 %)
Total DSP Blocks	0 / 150 ( 0 %)
Total HSSI RX PCSs	0 / 6 ( 0 %)
Total HSSI PMA RX Deserializers	0 / 6 ( 0 %)
Total HSSI TX PCSs	0 / 6 ( 0 %)
Total HSSI PMA TX Serializers	0 / 6 ( 0 %)
Total PLLs	0 / 12 ( 0 %)
Total DLLs	0 / 4 ( 0 %)

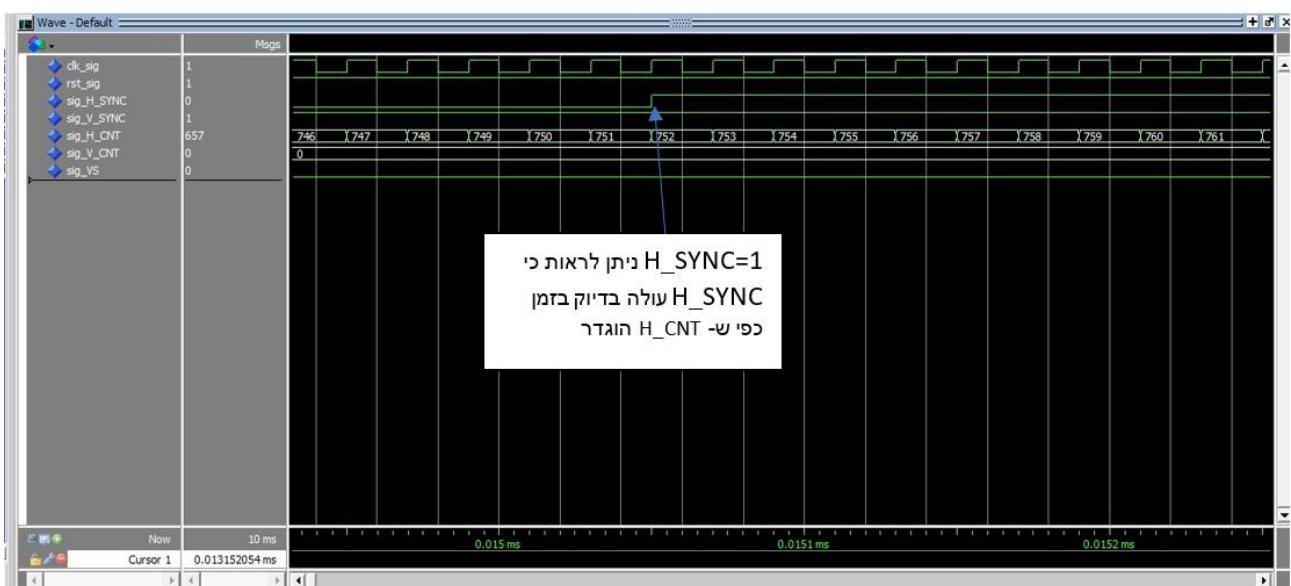
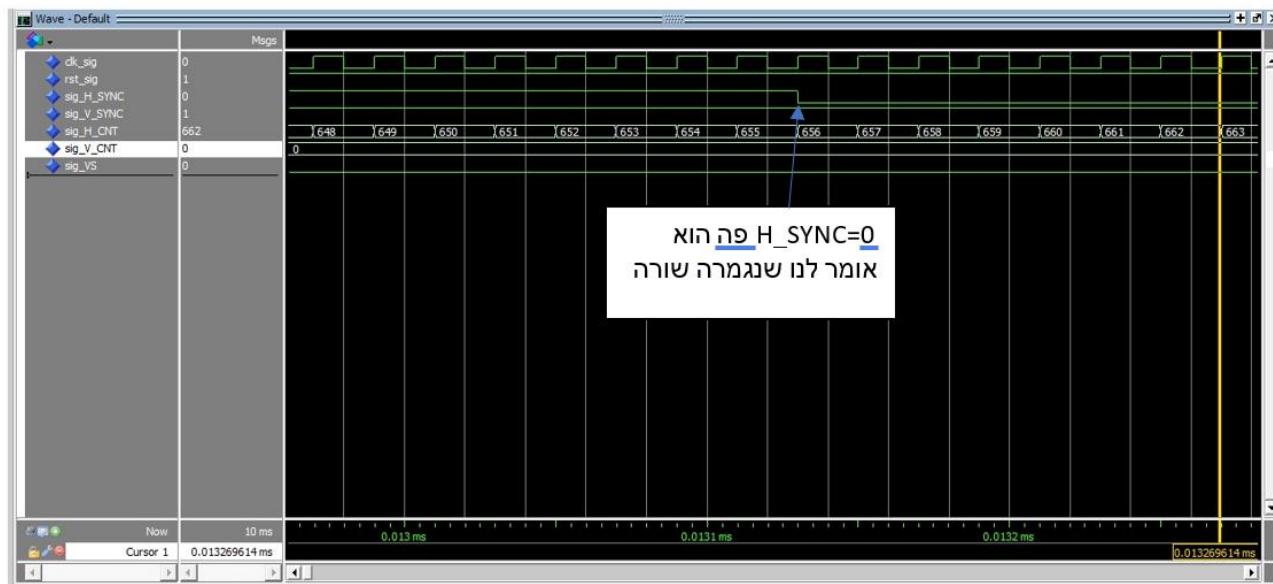
### סיכום עמידה בתדר – Fmax :

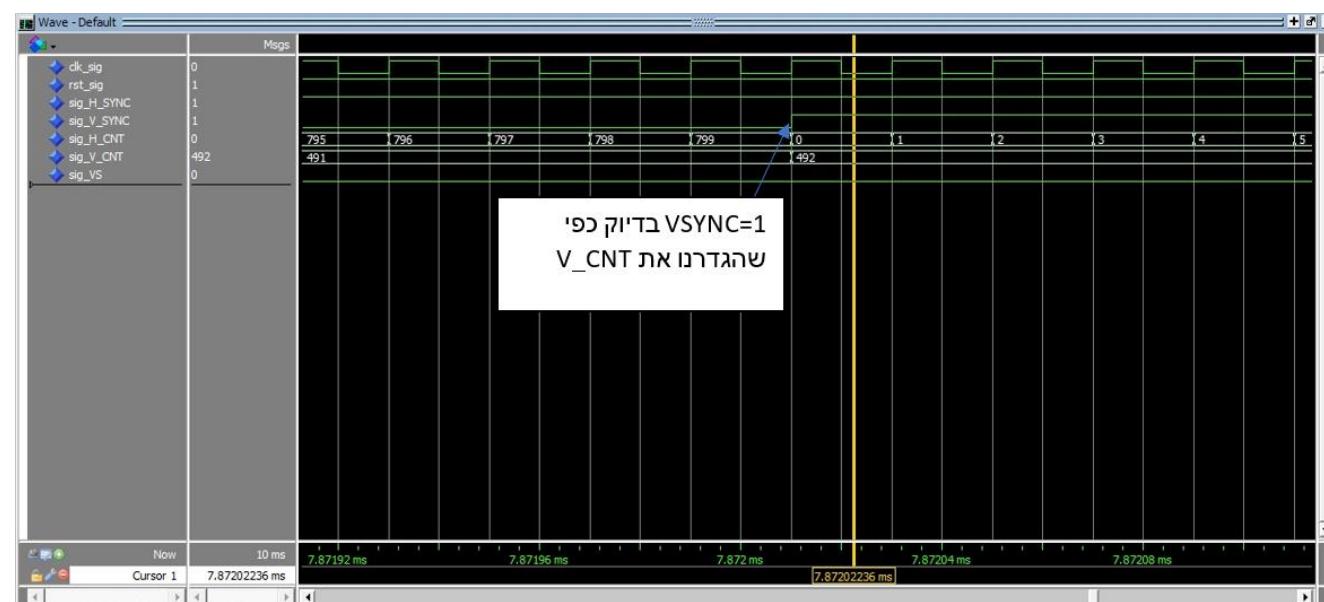
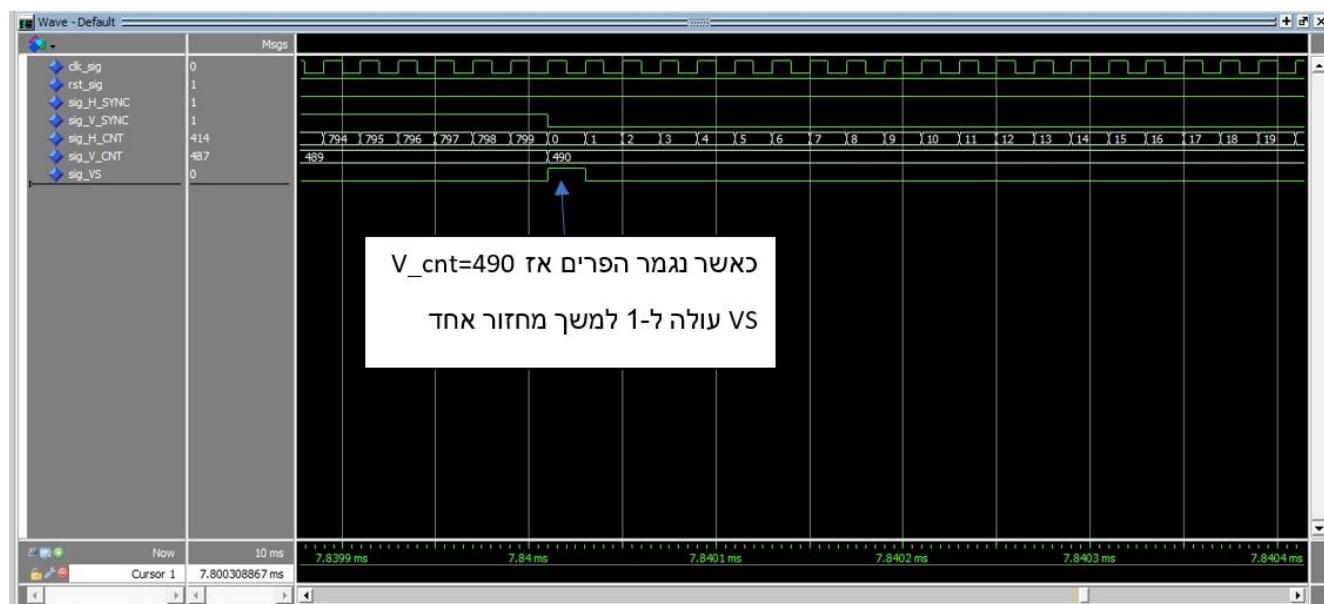
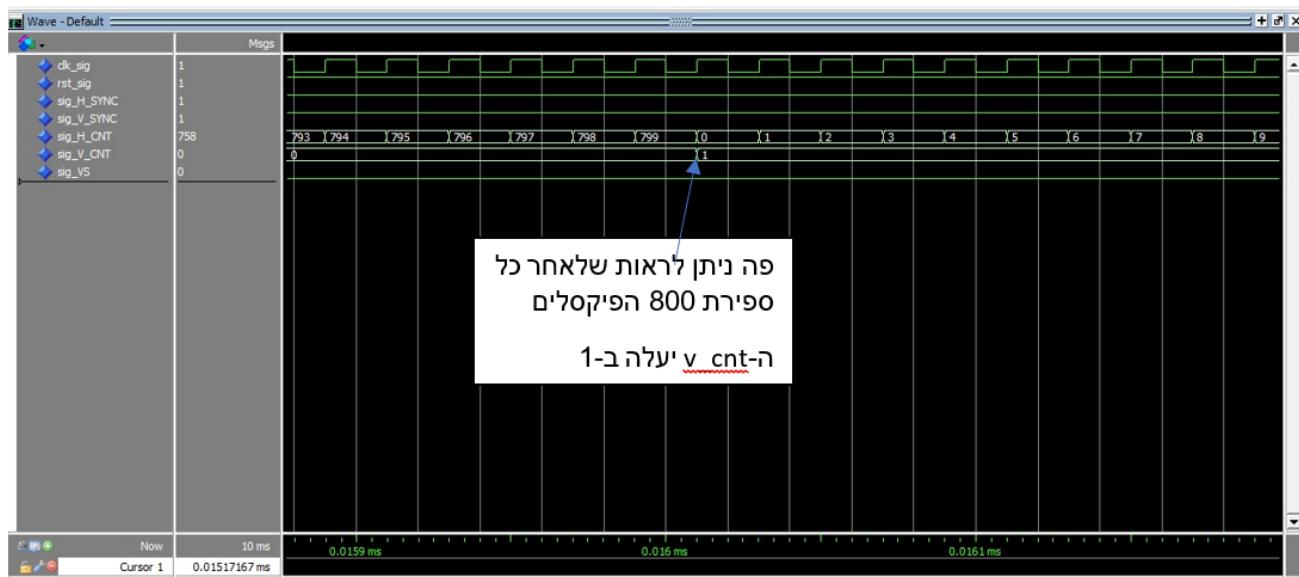
	Fmax	Restricted Fmax	Clock Name	Note
1	231.54 MHz	231.54 MHz	CLK	

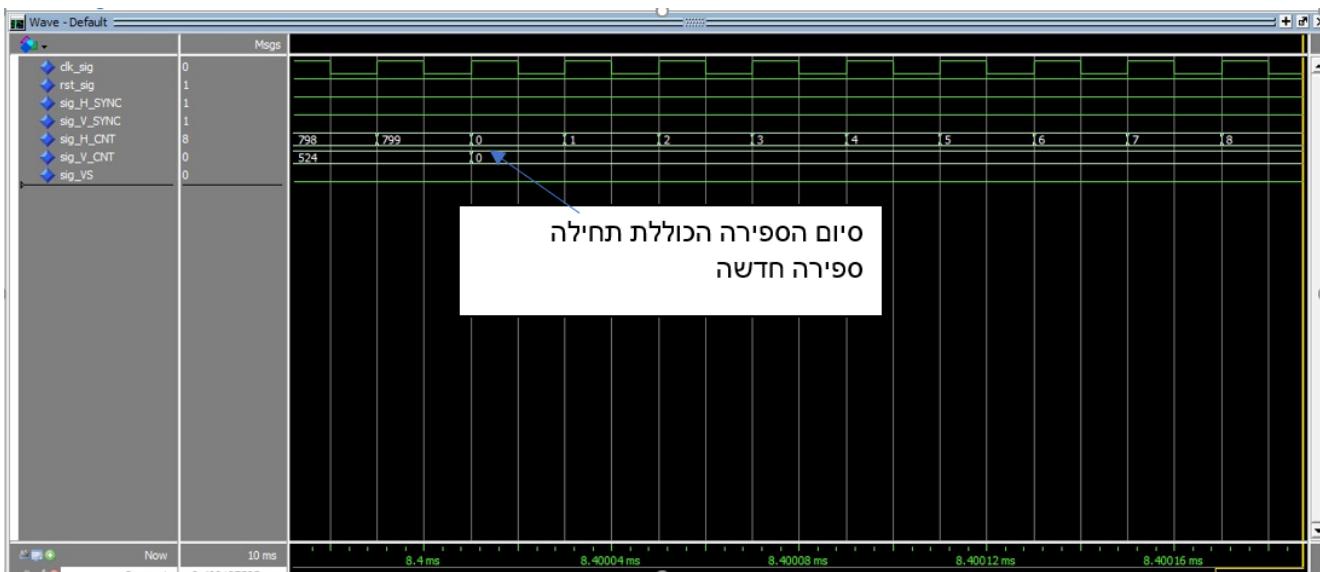
## מבנה לוגי של הבלוק:



## סימולציה של הבלוק:







## :test bench

```

library ieee;
use ieee.std_logic_1164.all;
use work.data_image_package.all;

entity timing_generator_tb is          -- The Testbench entity is empty. No ports.
end entity;

architecture behave of timing_generator_tb is      -- This is the architecture of the testbench
-- constants declaration
constant C_CLK_PRD           : time := 20 ns;

component timing_generator is           -- This is the component declaration.
port (
    RST      : in    std_logic;
    CLK      : in    std_logic;
    H_SYNC   : out   std_logic;
    V_SYNC   : out   std_logic;
    H_CNT    : out   integer range 0 to c_pixels_per_line-1;--800
    V_CNT    : out   integer range 0 to c_lines_per_frame-1;--525
    VS       : out   std_logic
);
end component;

-- signals declaration
signal clk_sig : std_logic := '0';
signal rst_sig : std_logic := '0';
signal sig_H_SYNC : std_logic:='0';
signal sig_V_SYNC : std_logic:='0';
signal sig_H_CNT : integer range 0 to c_pixels_per_line-1;--800
signal sig_V_CNT : integer range 0 to c_lines_per_frame-1;--525
signal sig_VS   : std_logic:='0';

begin

dut: timing_generator           -- This is the component instantiation. dut is the instance name of the component timing_generator
port map (
    RST      => rst_sig, -- The RST input of the dut instance of the pulse generator component is connected to rst_sig signal
    CLK      => clk_sig, -- The CLK input of the dut instance of the pulse generator component is connected to clk_sig signal
    H_SYNC   => sig_H_SYNC,
    V_SYNC   => sig_V_SYNC,
    H_CNT    => sig_H_CNT,
    V_CNT    => sig_V_CNT,
    VS       => sig_VS
);

clk_sig <= not clk_sig after C_CLK_PRD / 2;      -- clk_sig toggles every C_CLK_PRD/2 ns
rst_sig <= '0', '1' after 20 ns;                  -- the 100 ns is arbitrary

end architecture;

```

בלוק: data generator

## הסבר:

בבלוק זה מתבצעת ייצרת התמונה לכל זווית והקצתה הכתובה ל'זיכרון המram  
בשלב הראשון אנו בודקים שאנו לא באיזור ההחשה מיידית וכן אין DATA\_DE=0  
אחרת DATA\_DE=1 לאחר שבדקנו שאנו לא נמצא באיזור ההחשה נבדוק באיזה מצב  
נמצא

image\_ena\_int

עבורו לוגי : ניצור עמודות פסים בכל 8 פיקסלים כאשר כל פס מתאר צבע אחר, לאחר שהכנו את הנתונים ל-`tx_data` נעביר כל 8 סיביות (סך הכל 24 סיביות מקבל הAKER (HDMI) לצבעים הבסיסיים DATA R,G,B וaz נקבל במסגרת ה-

uboR 1 לוגי : ניצור סיגנלים של DATA\_R,G,B,\_DATA 5,6,5 סיביות בהतאמה מכיוון ש-  
 p\_sram (משמש לשילוח נתונים הצבעים זה DATA שישוב ב-RAM) נתן לנו 16 סיביות.  
 ולאחר מכן נבצע המרה על ידי פונקציה שיצרנו בחיבור, הפונקציה עשויה המרה 5,6,5  
 ל-8,8 סיביות הנטישה לכך שאנו עושים זאת היא בגל שבק HDMI מקבל 24 סיביות ולא  
 16 סיביות.

**תנאי להציג התמונה :**

כשר אנחנו נמצאים באיזור התמונה מהצד האופקי ומהצד ההאנכי אנו איז אנחנו יכולים להציג את התמונה כלומר אנחנו מבצעים על `h-active` ו- `v-active` פועלות AND כלומר `image_active<=image_h_active and image_v_active` במידה ונקלט 1 לוגי נכון להציג את התמונה

**cut נסbir על הקצאת הכתובות לזכרון RAM**

הגדירו סיגナル של 9 סיביות לפיקסלים ולשורות כלומר `image_line_num` ו-`image_pixel_num` סיגנליים שלב של יצירת התמונה.

כלומר כאשר אנחנו נמצאים בזווית של 0 מעלות אני מקבלים את הערך ההתחלתי מתוך המערכת שהגדכנו בחבילה שמרתתו לספק לנו את התנאים עבור הפיקסל ושורטת שמהם ניתן לסייע, ככלומר מעלים ב-1 או מחסרים ב-1 בהתאם לזוויות של התמונה שאוותה אותנו מטענים לייצר, לאחר כל פעולה אנחנו שולחים אותה ל- `a_sram` שהוא בעצם הזכרן שאליו מזמנים את הכתובות על ידי שרשור הסיגנלים `num_line` ו- `image_pixel`.

ועבור כל זווית נבעע את שרשרת הנכוון כפי שהגדכנו בקדם.

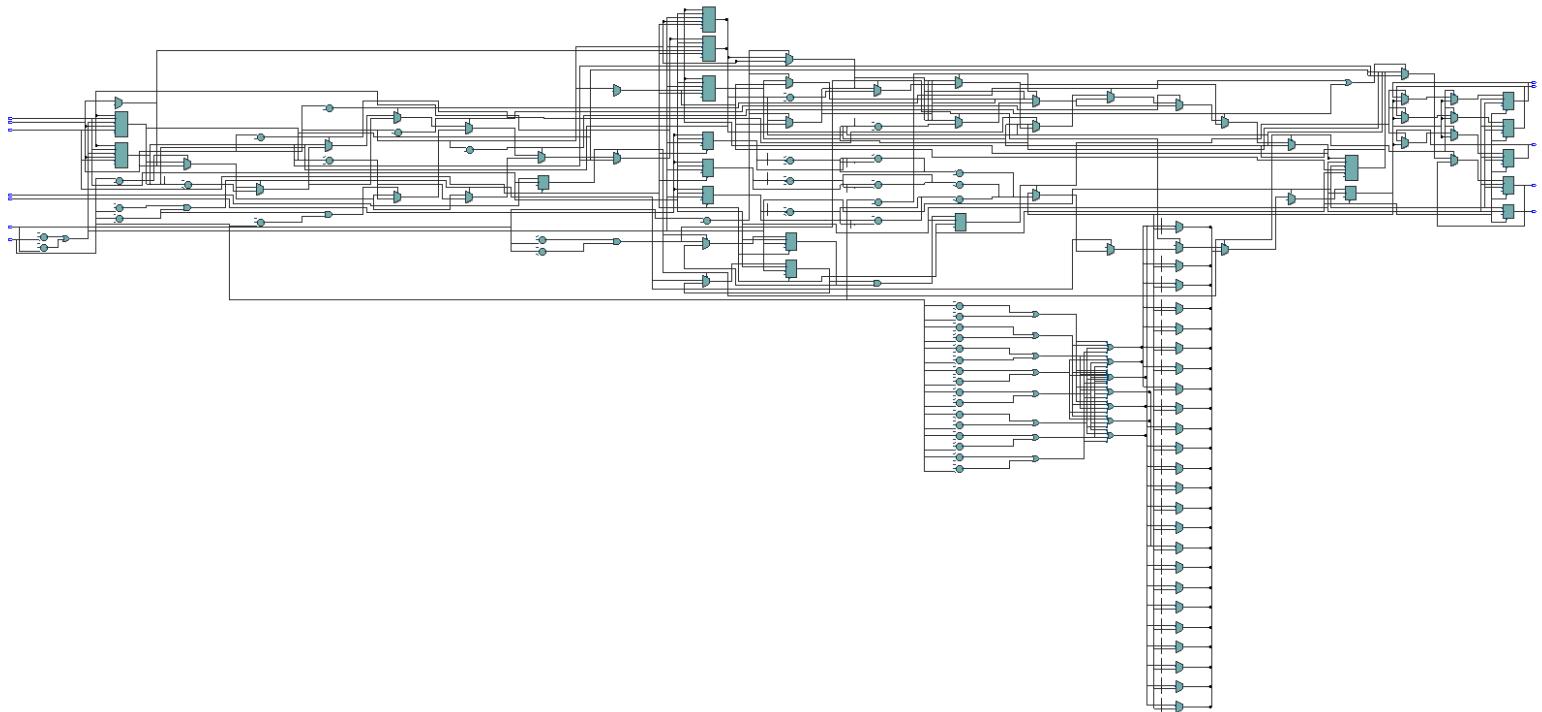
### **סיכון משאבים:**

Flow Status	Successful - Wed Jun 28 15:06:34 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	data_generator
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	372 / 29,080 ( 1 % )
Total registers	105
Total pins	84 / 364 ( 23 % )
Total virtual pins	0
Total block memory bits	0 / 4,567,040 ( 0 % )
Total DSP Blocks	0 / 150 ( 0 % )
Total HSSI RX PCSs	0 / 6 ( 0 % )
Total HSSI PMA RX Deserializers	0 / 6 ( 0 % )
Total HSSI TX PCSs	0 / 6 ( 0 % )
Total HSSI PMA TX Serializers	0 / 6 ( 0 % )
Total PLLs	0 / 12 ( 0 % )
Total DLLs	0 / 4 ( 0 % )

### **סיכון עמידה בתדר – Fmax**

Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	29.69 MHz	29.69 MHz	CLK	

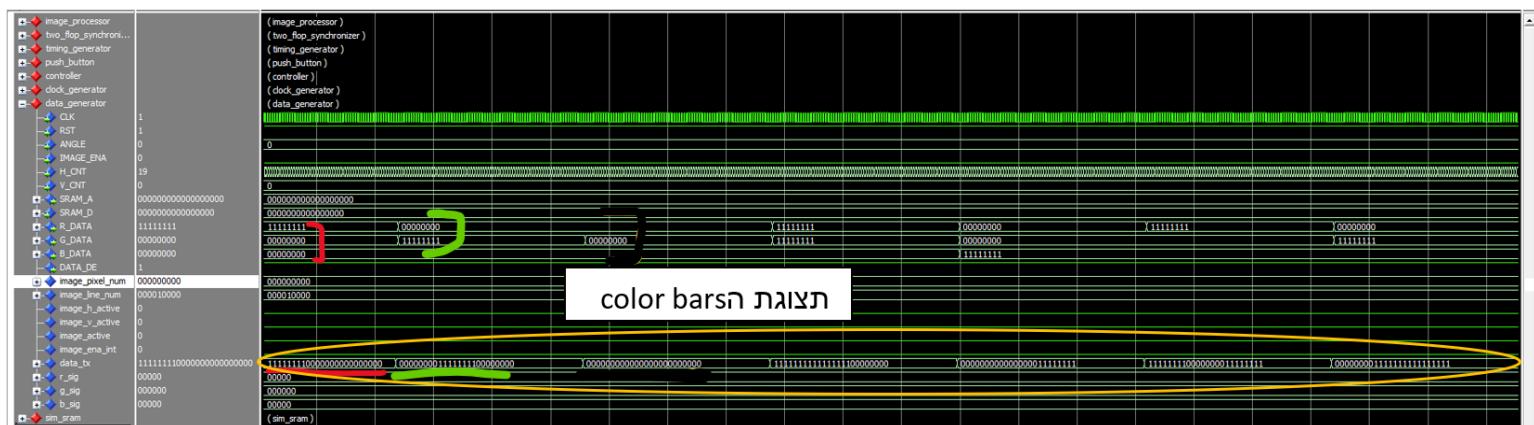
## מבנה לוגי של הבלוק:



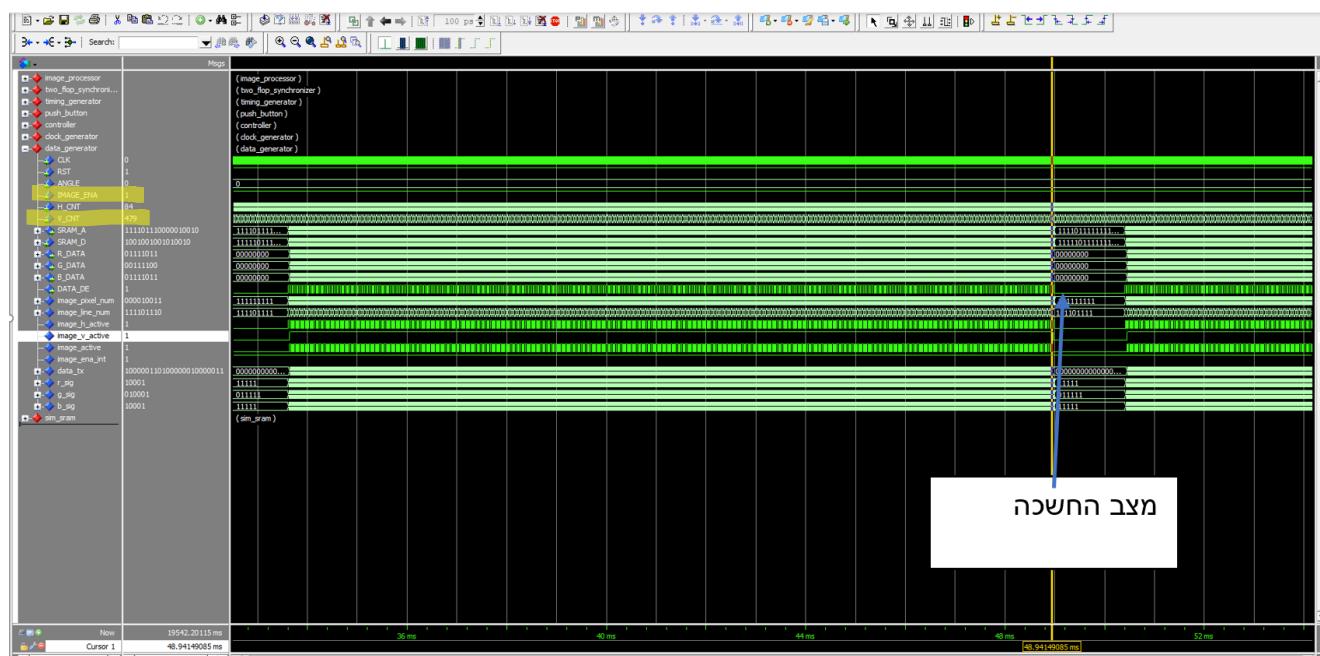
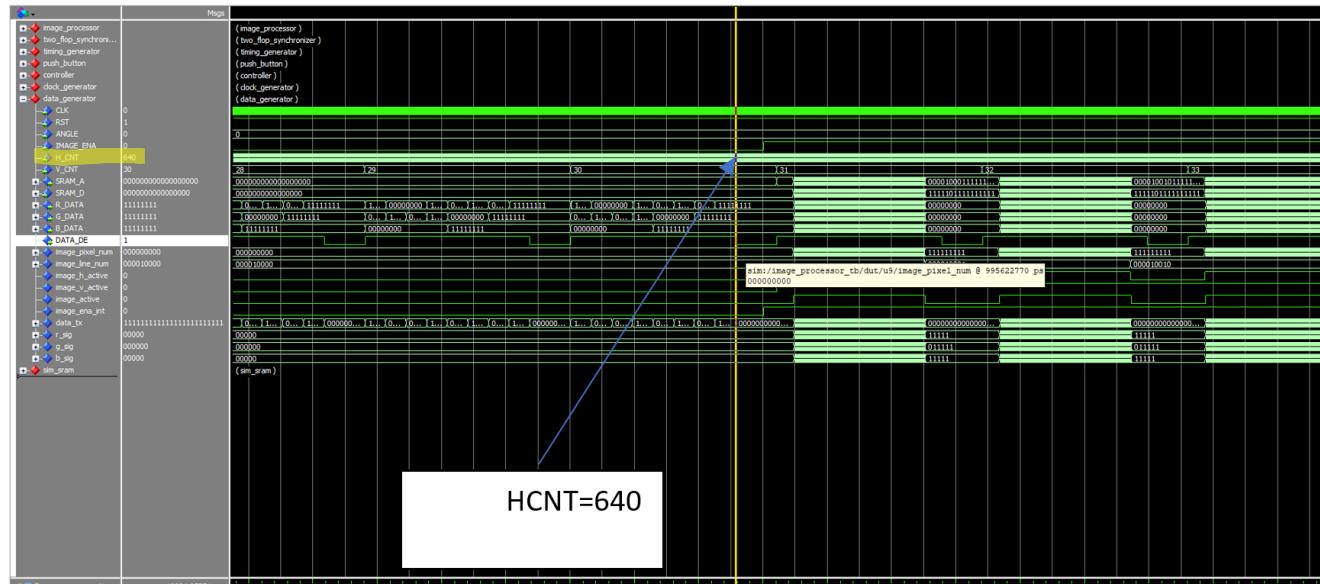
## סימולציה של הבלוק:

כאשר IMAGE\_ENA='0' ניתן לראות את הייצור של ה color bars על TX\_DATA וazz העברתם לשלווש צבעי הבסיס DATA\_B, DATA\_G, DATA\_R (RGB).

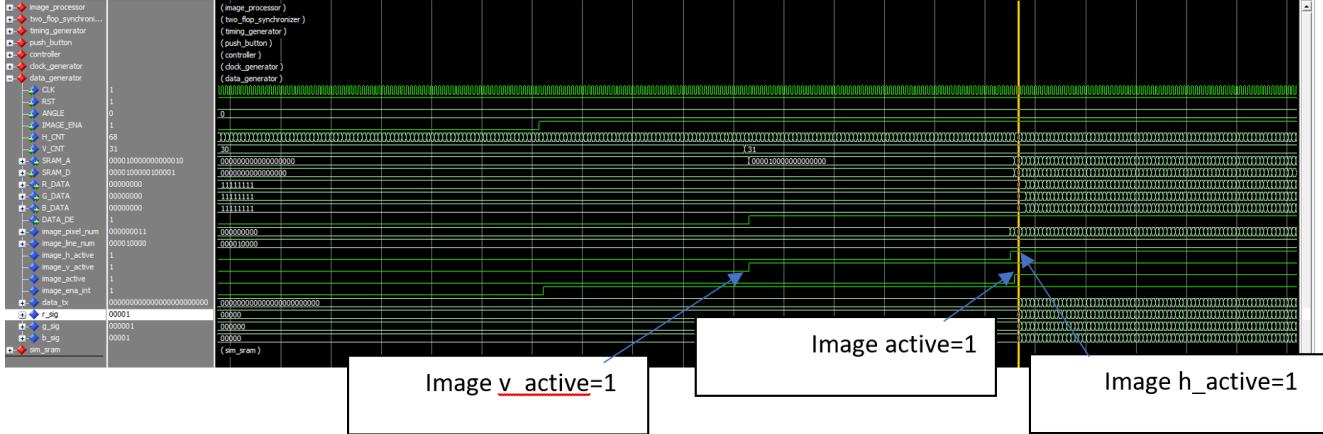
על ידי רמת הצבועים הבסיסית נוכל ליצור צבעים אחרים כפי שניתן לראות בתמונה הבאה.



בשני התמונות הבאות נראה שלאחר שהפעלו את ה `image_ena` ל 1 לוגי זאת אומרת שאנוחנו עברנו במצב של צפיה בתמונה, אך כאשר `DE_DATA` יורד ל- "0" לוגי אנחנו יכולים לראות שאנחנו נכנסים במצב החשכה כלומר ש- `T_CNT_H` גודל מ- 640 או ש- `v_cnt` גודל מ- 480.



בתמונה הבאה ניתן לראות כאשר IMAGE\_ACTIVE שווה ל 1 לוגי זאת אומרת שנחנו יכולם להתחיל להכניס את הערcis בכל מחרוז של שעון ל num\_pixel\_num ול- NUM\_PIXELS ובתנאי עומדים באיזור של התמונה כולם HCNT ו- VCNT לא חורגים מאזור של התמונה. עבור IMAGE\_ACTIVE הוא יעלה ל-1" לוגי רק כאשר החלק האופקי IMAGE\_H והחלק האנכי IMAGE\_V עולים ל"1" לוגי כלומר נמצאים באיזור של התמונה.



בתמונה הבאה ניתן לראות את סיום הקצאת הכתובות עבור זוויות 180 מעלות (2=ANGLE)

תחילת הספירה של הפיקסל הייתה " 111111111111 " (511) לאחר סיום הספירה נקבל

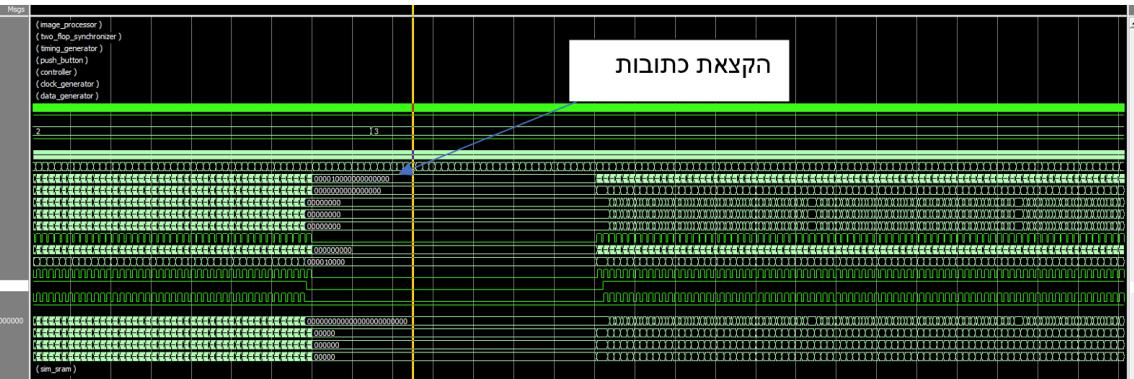
$$000000000 = 0 \text{ שזה } 511 - 575 + 64$$

וחילת הספירה של השורות הייתה " 1111011111 " (495) לאחר סיום הספירה נקבל :

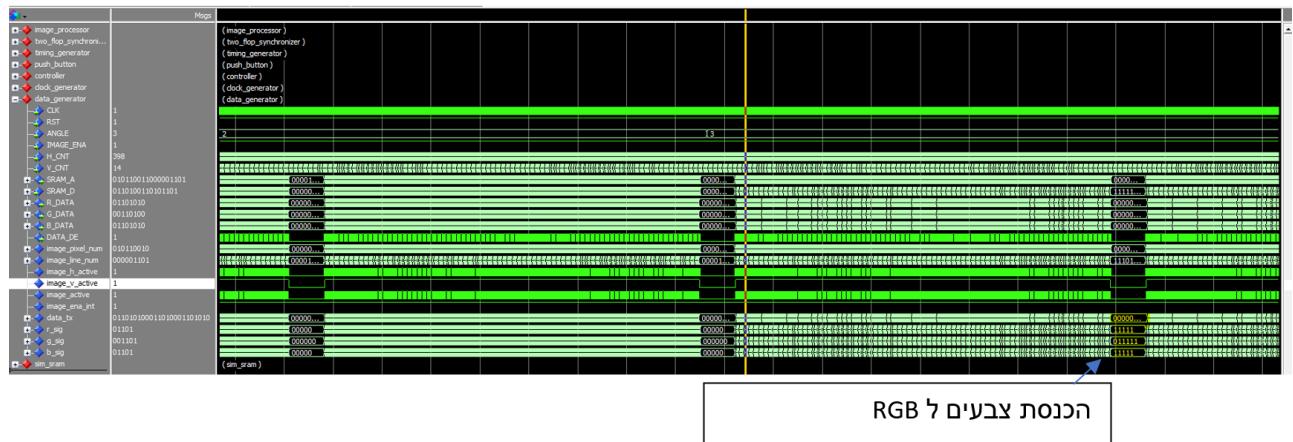
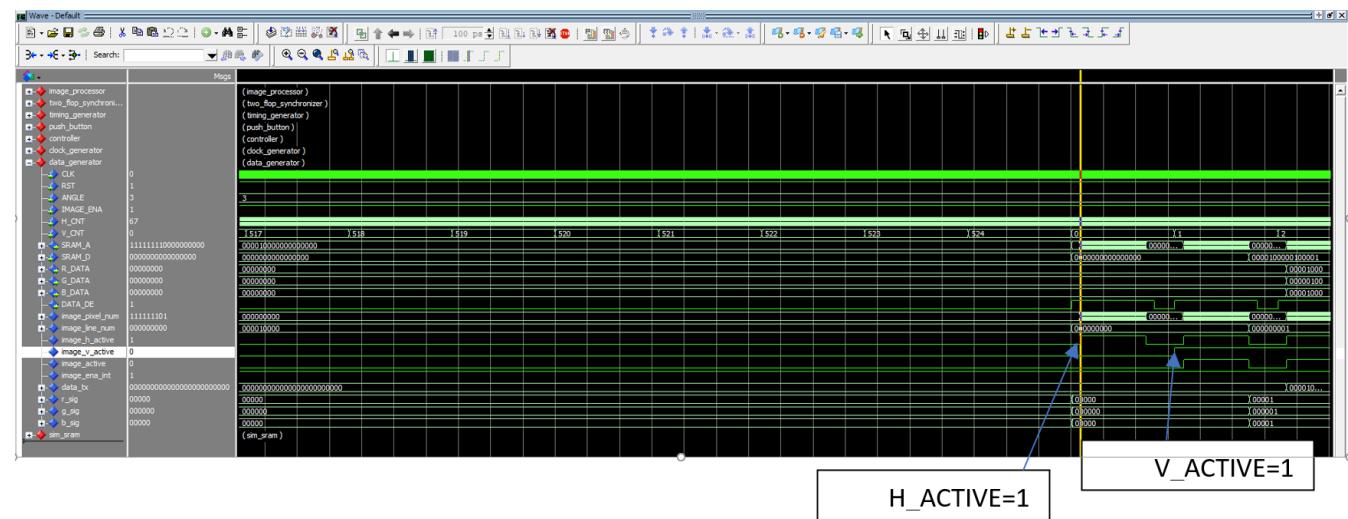
$$000010000 = 16 \text{ שזה } 495 - 479 - 0$$

זה בדיק התוצאה שקיבלנו בסם\_sram

כפי שהזכרנו לפני בזווית 0 ו-180 אנחנו משרירים מהשורה ועד לפיקסלים כלומר lines&pixels



בתמונה הראשונה. ניתן לראות שברגע ש-`h_active` שווה ל- "1" לוגי אז נכנס את הערך ההתחלתי של הפיקסלים שהוא "1111111111" ולאחר מכן שאגם `v_active` יעלה ל- "1" אז נכנס את הערך "0000000000" וכאשר `image_active` יעלה ל- "1" אנחנו ניראה את הצבעים שמקבלים RG B DATA (בתמונה השנייה) .



## בלוק: clock generator

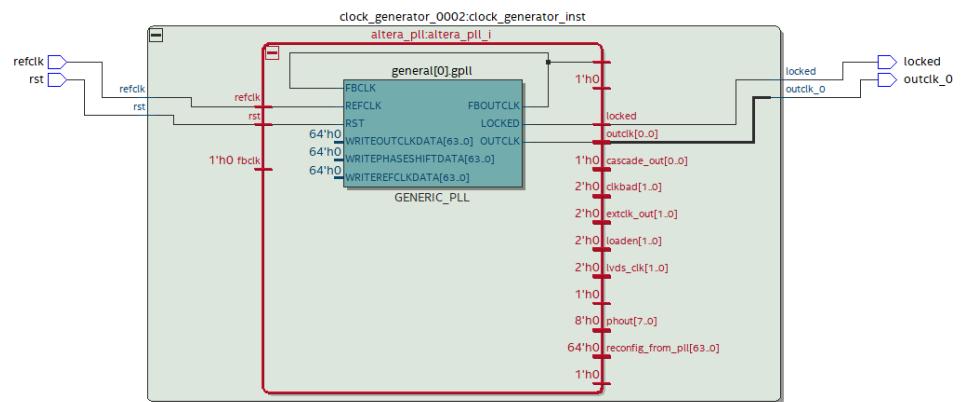
### הסבר:

מטרת הבלוק היא להמיר את התדר מ- 50MHZ מהמCSIr לתדר של 25MHz  
הסיבה לכך היא מכיוון שהוHDMI עובד על 25MHz

### סיכום משאבים:

Flow Status	Successful - Wed Jun 28 14:21:30 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	clock_generator
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	1 / 29,080 (< 1 %)
Total registers	0
Total pins	4 / 364 ( 1 % )
Total virtual pins	0
Total block memory bits	0 / 4,567,040 ( 0 % )
Total DSP Blocks	0 / 150 ( 0 % )
Total HSSI RX PCSS	0 / 6 ( 0 % )
Total HSSI PMA RX Deserializers	0 / 6 ( 0 % )
Total HSSI TX PCSS	0 / 6 ( 0 % )
Total HSSI PMA TX Serializers	0 / 6 ( 0 % )
Total PLLs	1 / 12 ( 8 % )
Total DLLs	0 / 4 ( 0 % )

## מבנה לוגי של הבלוק:



## בלוק: two flop synchronizer

הסבר: מטרת הבלוק היא לסנכרן את הלחצנים מהמכשור לשעון.

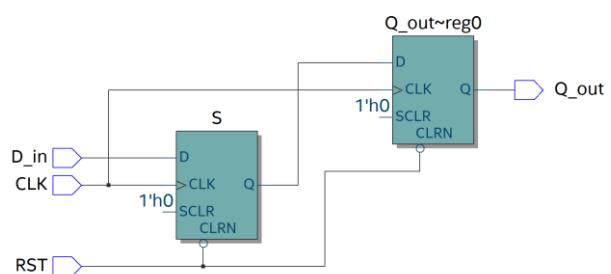
### סיכום משאבים:

Flow Status	Successful - Fri Jun 30 03:25:06 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	two_flop_synchronizer
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	1 / 29,080 (< 1 %)
Total registers	2
Total pins	4 / 364 (1 %)
Total virtual pins	0
Total block memory bits	0 / 4,567,040 (0 %)
Total DSP Blocks	0 / 150 (0 %)
Total HSSI RX PCSs	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSs	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 4 (0 %)

### סיכום עמידה בתדר – Fmax

Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	907.44 MHz	650.2 MHz	CLK	(lim...in)

### מבנה לוגי של הבלוק:



## בלוק: bcd to 7seg

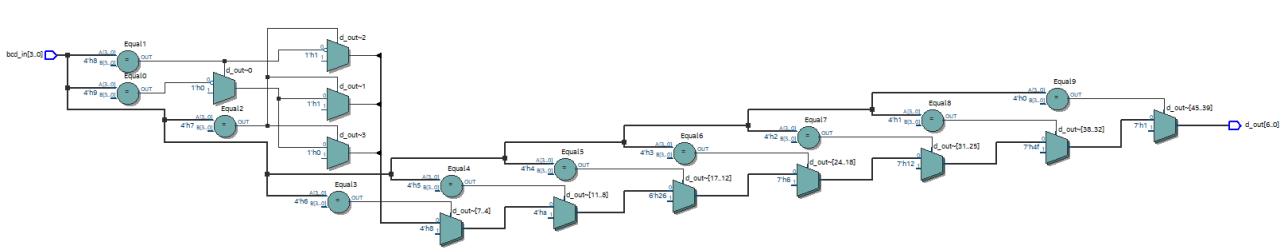
### הסבר:

כפי שהסבירנו בבלוק של controller-ה controller זה משמש לתרגום הערכים של הזרויות לערכים של בינריים שמרתם לייצג את המספרים במכשיר אחרת.

### סיכום משבבים:

Flow Status	Successful - Fri Jun 30 03:36:32 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	bcd_to_7seg
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	4 / 29,080 (< 1 %)
Total registers	0
Total pins	11 / 364 (3 %)
Total virtual pins	0
Total block memory bits	0 / 4,567,040 (0 %)
Total DSP Blocks	0 / 150 (0 %)
Total HSSI RX PCSS	0 / 6 (0 %)
Total HSSI PMA RX Deserializers	0 / 6 (0 %)
Total HSSI TX PCSS	0 / 6 (0 %)
Total HSSI PMA TX Serializers	0 / 6 (0 %)
Total PLLs	0 / 12 (0 %)
Total DLLs	0 / 4 (0 %)

### מבנה לוגי של הבלוק:



## בלוק: image processor (הרמה העליונה)

### הסבר:

מטרת הבלוק הינה לחבר בין כל הבלוקים שעשינו למעלה , לשЛОח ערcisム לבקור ה- HDMI ו- sram.

הבלוק מורכב מ 2 לחצנים ו 3 מפסקים

מטרתו להציג לנו את הפעולות המערכת על ידי שילוב כל הבלוקים שהגדכנו

מעבר מפסק IMAGE\_ENA\_SW כאשר הוא ב 0 לוגי המערכת תציג לנו את colors bars

כלומר נקבל את ייצור הפסים מתוך ה - data\_generator והוא מעביר את זה ל – TX\_HDMI

וממנו מתחבר לבקור ה HDMI ושם אנחנו רואים את תצוגת הפסים על המסך.

כאשר אנחנו ב 1 לוגי ה – TX\_HDMI את נתונים התמונה ואז אנו רואים במסך את התמונה

מעבר מפסק MODE\_SW כאשר הוא מקבל 1 לוגי אז בשלב זה המערכת תבצע כל שנייה החלפה של התמונה בהתאם לערך הלוגי שמקבל DIR\_SW ( שתפקידו לקבוע את כיוון של היפוך התמונה )

כאשר אנחנו נמצאים ב 0 לוגי אין היפוך תמונה ובשלב זה ניתן להיפוך את התמונה באופן ידני על ידי הלחץ KEY\_ROTATE , אשר יכול לבצע שני פועלות הבאות :

1. כל לחיצה קצרה כלומר פחות מ 2 שניות התמונה תתהפך באופן ישיר ככלمر לפ' השען של CLOCK\_GENERATOR
2. לחיצה ממיל 2 שניות מתחילה לספר כל שנייה ואז הופכת את התמונה בכל שנייה

לחץ אחרון הוא ה-RST שמטרתו לאפס את המCSR ולהחזיר למצב ההתחלתי כלומר תמונה בזריזות של 0 מעלות.

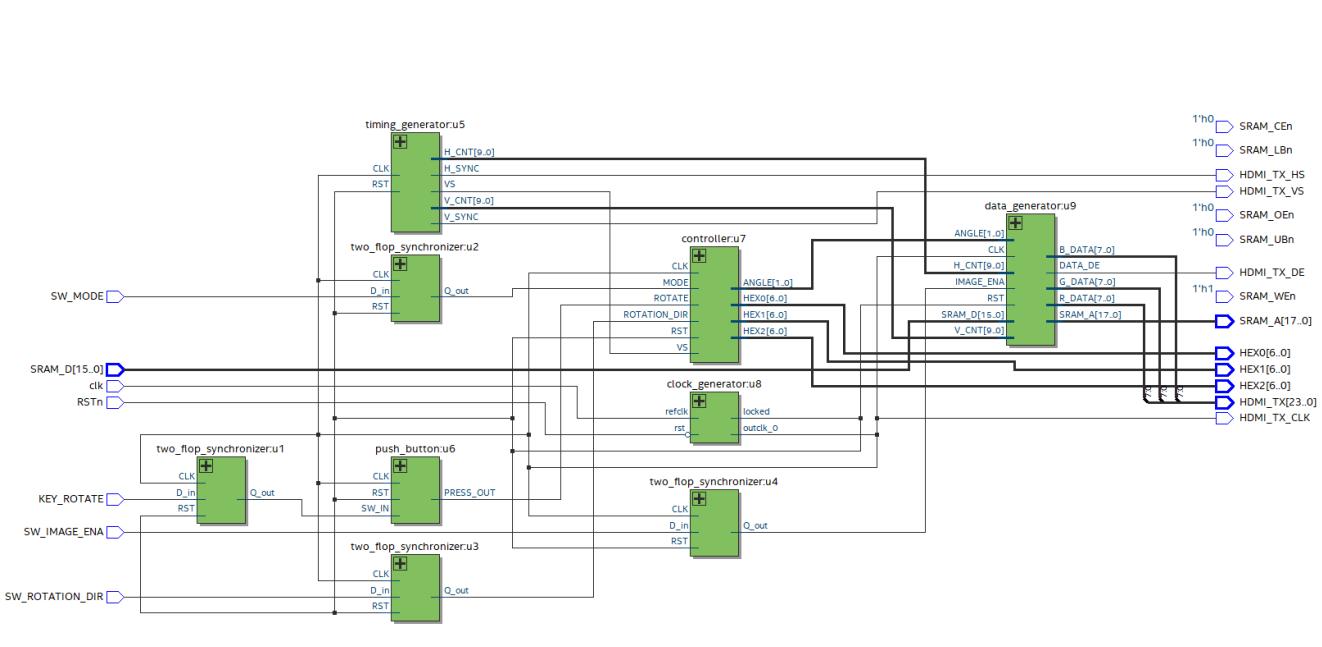
### סיכום משאבים:

Flow Status	Successful - Wed Jun 28 15:10:44 2023
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	image_processor
Top-level Entity Name	image_processor
Family	Cyclone V
Device	5CGXFC5C6F27C7
Timing Models	Final
Logic utilization (in ALMs)	455 / 29,080 ( 2 % )
Total registers	222
Total pins	94 / 364 ( 26 % )
Total virtual pins	0
Total block memory bits	0 / 4,567,040 ( 0 % )
Total DSP Blocks	0 / 150 ( 0 % )
Total HSSI RX PCSs	0 / 6 ( 0 % )
Total HSSI PMA RX Deserializers	0 / 6 ( 0 % )
Total HSSI TX PCSs	0 / 6 ( 0 % )
Total HSSI PMA TX Serializers	0 / 6 ( 0 % )
Total PLLs	1 / 12 ( 8 % )
Total DLLs	0 / 4 ( 0 % )

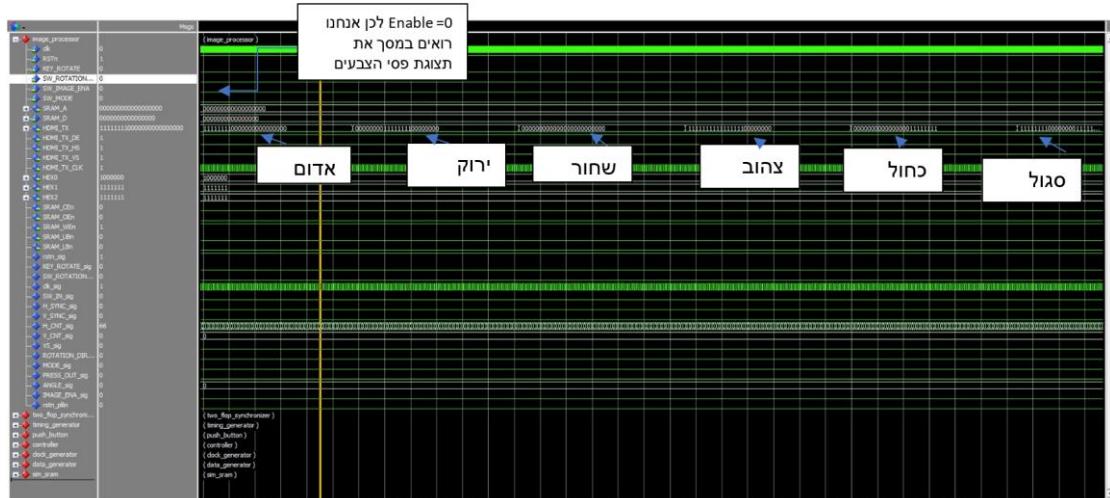
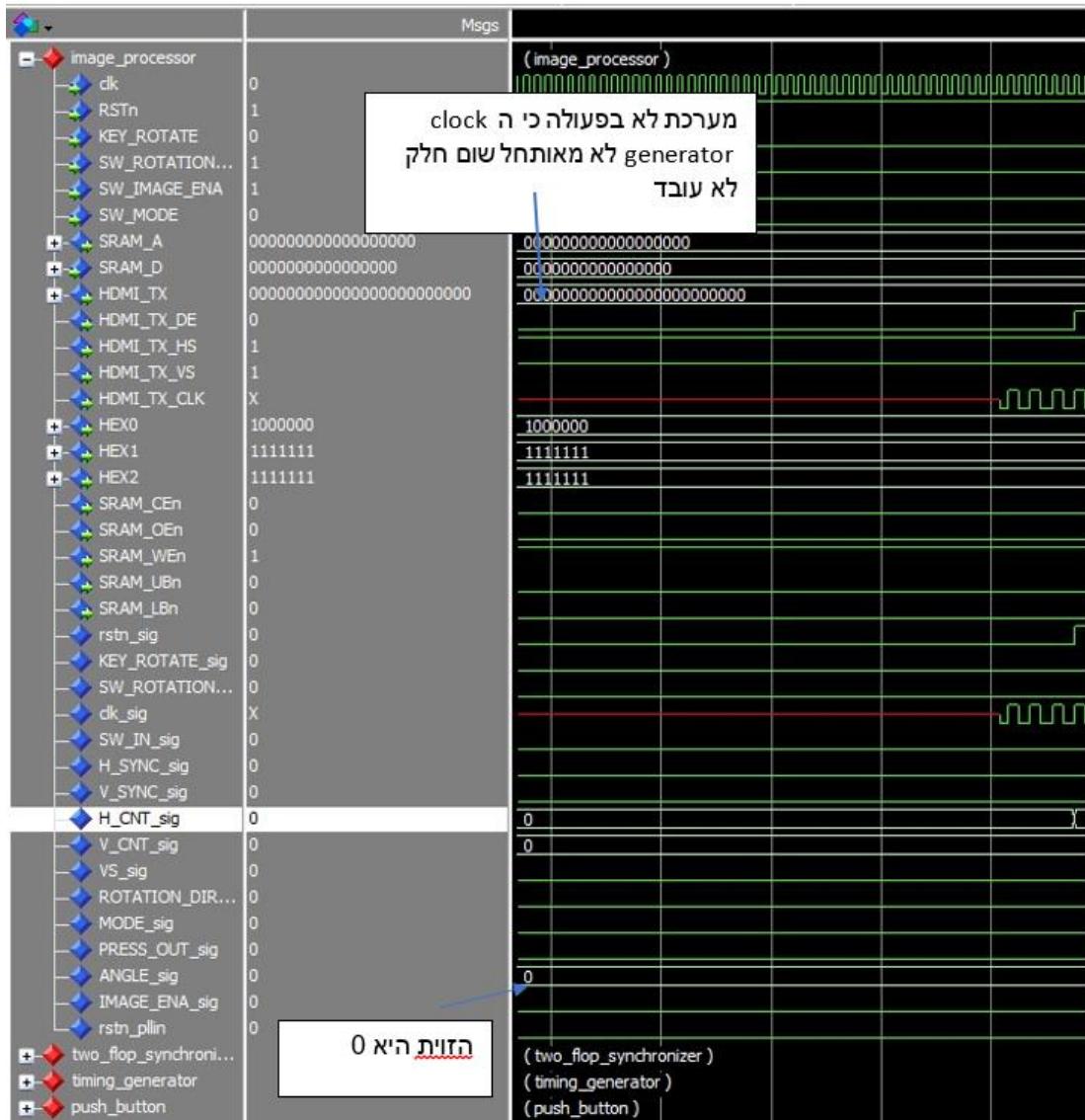
## סיכון עמידה בתדר – Fmax

Slow 1100mV 85C Model Fmax Summary				
	Fmax	Restricted Fmax	Clock Name	Note
1	29.96 MHz	29.96 MHz	u8 clock divclk	

## מבנה לוגי של הבלוק:

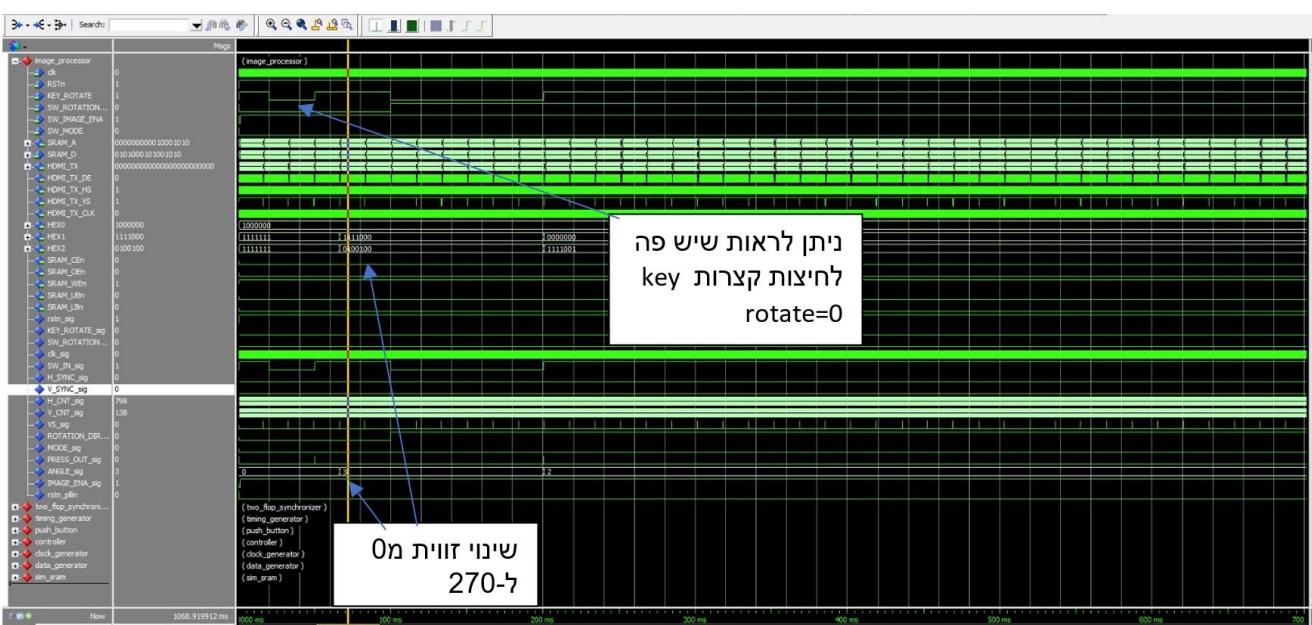


## סימולציה של הבלוק:



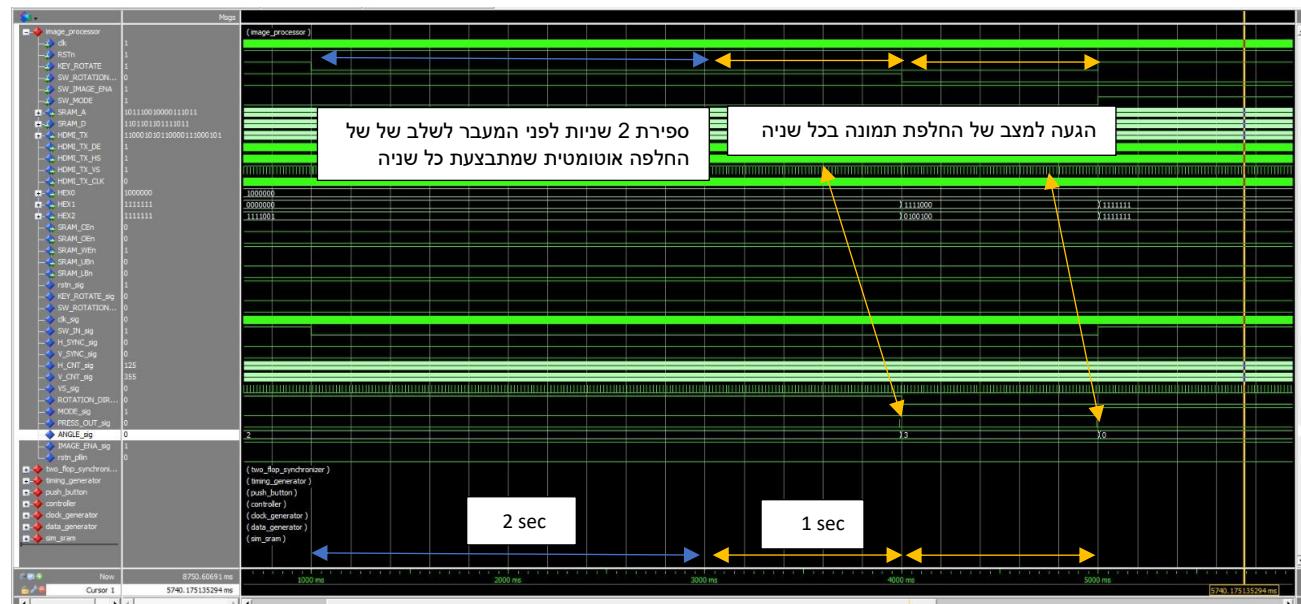


בתמונה הבאה ניתן לראות שהשינוי בזווית רק כאשר קיבלנו פולס ולאחר מכן קיבלנו את הפולס מהתא PRESS\_OUT ש- $V=1$  הוא תחלף, כפי שניתן לראות יש החלפה מ- 0 ל-270.





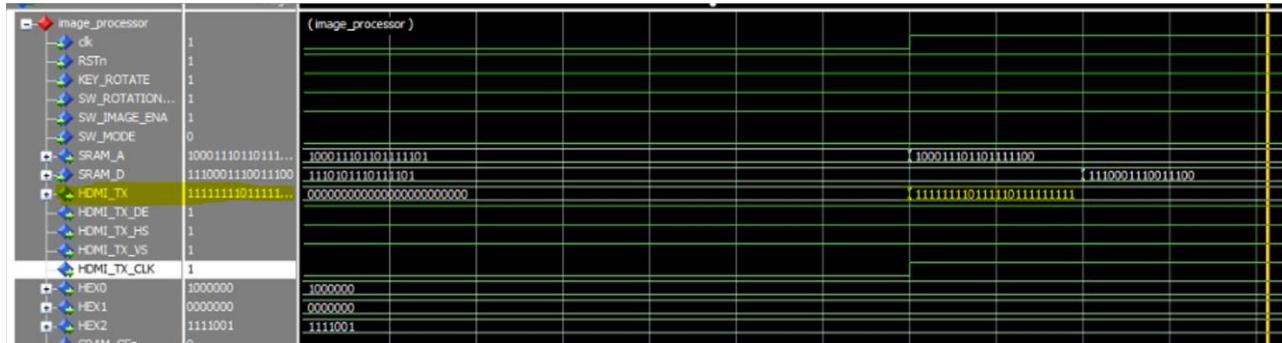
בתמונה הבאה ניתן לראות אתימוש הלחיצה הארכאה שנעשית בעזרת הלחץ key .rotate



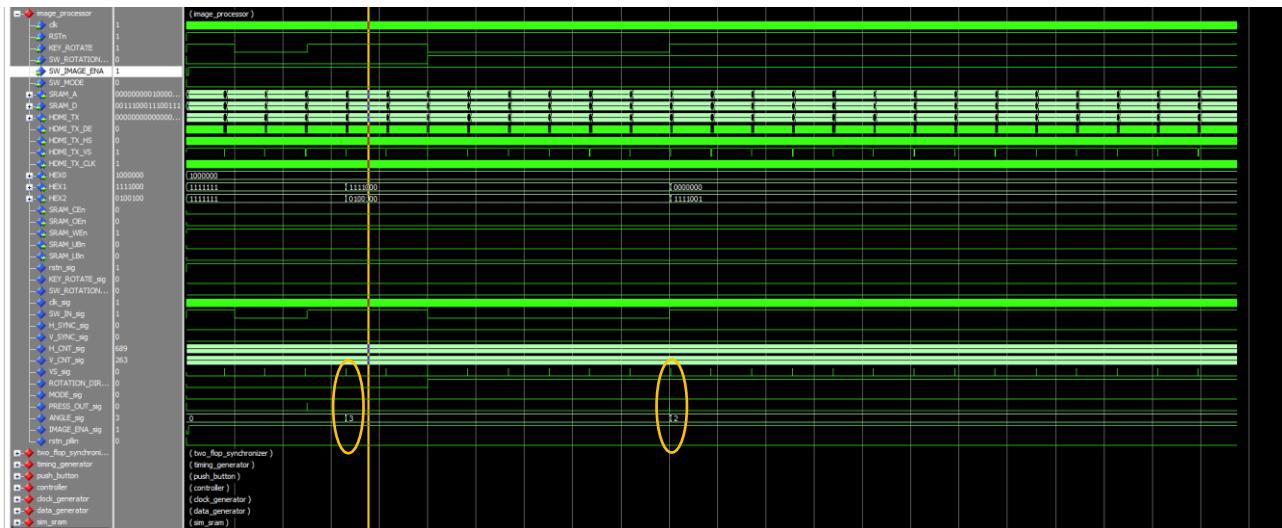
בתמונה הבאה נציג דוגמא של מידע היוצא מ-`hdmi_tx` המתאר מידע עבור פיקסל כלשהו באזור של התמונה מאופשרה:

כארס "11111111 01111101 11111111 hdmi\_tx = סדר הצבעים משמאלי לימין הוא RGB.

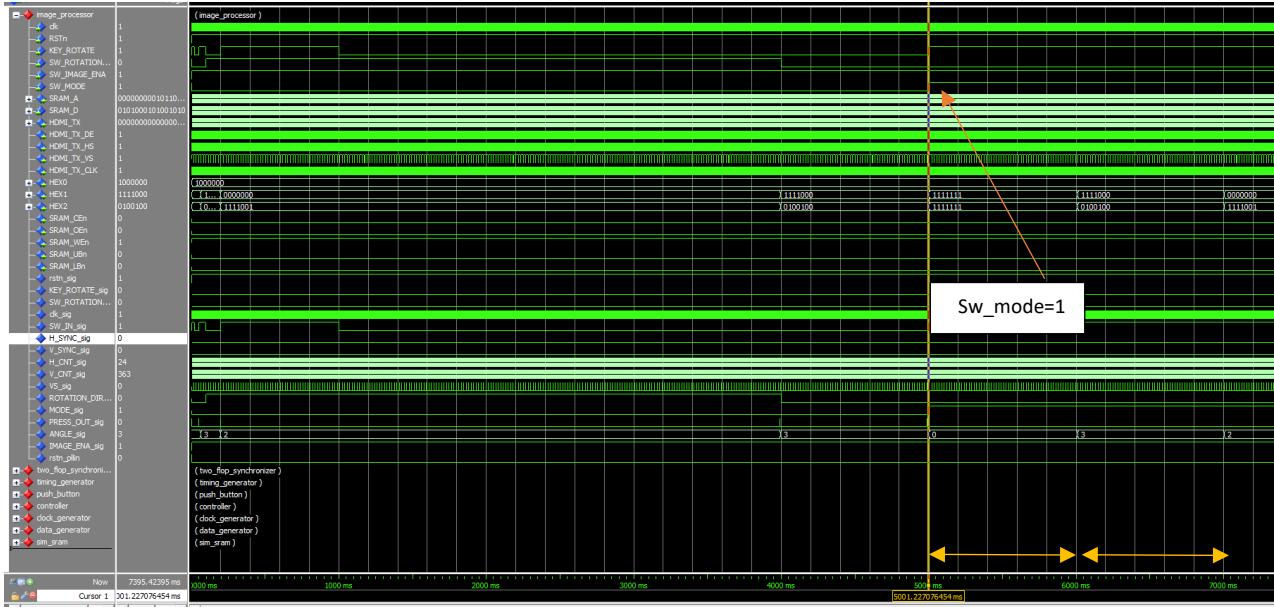
- הצבע האדום חזק ביותר ("11111111")
  - הצבע הירוק בינוני ("01111101")
  - הצבע הכחול חזק ביותר ("11111111")



בתמונה הבאה ניתן לראות שرك כאשר  $SV=1$  אז תהיה החלפת הזוגית.



בתמונה הבאה נראה שכאשר נכנס למצב אוטומט היפוך התמונה תהיה אוטומטית בכל שנייה.



#### קובץ DO לרמה העליונה:

```

image_processor.do

1
2
3
4 # Clock files:
5 vcom ./src/clock_generator_sim.vhd
6
7 # components files:
8
9 vcom ./src/data_image_package.vhd
10 vcom ./src/push_button.vhdl
11 vcom ./src/sim_sram.vhd
12 vcom ./src/controller.vhdl
13 vcom ./src/timing_generator.vhd
14 vcom ./src/data_generator.vhd
15 vcom ./src/two_flop_synchronizer.vhd
16 vcom ./src/image_processor.vhd
17 vcom ./src/image_processor_tb.vhd
18 vsim image_processor_tb
19
20
21 add wave -group image_processor image_processor_tb/dut/*
22 add wave -group two_flop_synchronizer image_processor_tb/dut/u1/*
23 add wave -group two_flop_synchronizer image_processor_tb/dut/u2/*
24 add wave -group two_flop_synchronizer image_processor_tb/dut/u3/*
25 add wave -group two_flop_synchronizer image_processor_tb/dut/u4/*
26 add wave -group timing_generator image_processor_tb/dut/u5/*
27 add wave -group push_button image_processor_tb/dut/u6/*
28 add wave -group controller image_processor_tb/dut/u7/*
29 add wave -group clock_generator image_processor_tb/dut/u8/*
30 add wave -group data_generator image_processor_tb/dut/u9/*
31
32 add wave -group sim_sram image_processor_tb/dut2/*
33
34
35 restart -f
36 run 20 sec

```

## לרמה העליונה: TEST BENCH

```
image_processor_tb.vhd | 1 library ieee;
2 use ieee.std_logic_1164.all;
3 use work.data_image_package.all;
4
5 entity image_processor_tb is          -- The Testbench entity is empty. No ports.
6 end entity;
7
8 architecture behave of image_processor_tb is      -- This is the architecture of the testbench
9
10-- constants declaration
11  constant C_CLK_PRD    : time := 20 ns;
12  constant C_image_file   : string := "sec_cols_fixed_lines_mem.bin";
13
14 component image_processor is                  -- This is the component declaration.
15 port(
16    clk : in std_logic; -- clock
17    RSTn : in std_logic; -- reset
18    KEY_ROTATE : in std_logic;
19    SW_ROTATION_DIR: in std_logic;
20    SW_IMAGE_ENA: in std_logic;
21    SW_MODE: in std_logic ;
22    SRAM_D : in std_logic_vector (15 downto 0);
23    SRAM_A : out std_logic_vector(17 downto 0);
24    SRAM_CEEn:out std_logic ;
25    SRAM_OEn:out std_logic ;
26    SRAM_WEn:out std_logic ;
27    SRAM_UBn:out std_logic ;
28    SRAM_LBn:out std_logic ;
29    HDMI_TX :out std_logic_vector (23 downto 0);
30    HDMI_TX_DE:out std_logic ;
31    HDMI_TX_HS:out std_logic ;
32    HDMI_TX_VS:out std_logic ;
33
34    HDMI_TX_CLK:out std_logic ;
35    HEX0: out std_logic_vector (6 downto 0);
36    HEX1: out std_logic_vector (6 downto 0);
37    HEX2: out std_logic_vector (6 downto 0)
38  );
39 end component;
40
41 component sim_sram is
42 generic ( ini_file_name: string := "UNUSED");
43 port (
44    SRAM_ADDR      : in  std_logic_vector(17 downto 0) := (others=>'0'); -- sram address
45    SRAM_DQ       : inout std_logic_vector(15 downto 0); -- sram data
46    SRAM_WE_N     : in  std_logic;                         -- sram write enable
47    SRAM_OE_N     : in  std_logic;                         -- sram output enable
48    SRAM_UB_N     : in  std_logic;                         -- sram upper byte enable
49    SRAM_LB_N     : in  std_logic;                         -- sram Lower byte enable
50    SRAM_CE_N     : in  std_logic;                         -- sram chip enable
51  );
52 end component;
53
54
55
56-- signals declaration
57 signal clk_sig:  std_logic:='0'; -- clock
58 signal RSTn_sig: std_logic:='0'; -- reset
59
60 signal KEY_ROTATE_sig: std_logic:='0';
61 signal SW_ROTATION_DIR_sig: std_logic:='0';
62 signal SW_IMAGE_ENA_sig: std_logic:='0';
63 signal SW_MODE_sig: std_logic:='0' ;
64 signal SRAM_D_sig : std_logic_vector (15 downto 0);
```

```

65      signal SRAM_A_sig : std_logic_vector(17 downto 0);
66      signal SRAM_CEn_sig: std_logic:='0' ;
67      signal SRAM_OEn_sig: std_logic:='0' ;
68      signal SRAM_WEn_sig: std_logic:='1' ;
69      signal SRAM_UBn_sig: std_logic:='0' ;
70      signal SRAM_LBn_Sig: std_logic:='0' ;
71      signal HDMI_TX_sig : std_logic_vector (23 downto 0);
72      signal HDMI_TX_DE_sig: std_logic:='0' ;
73      signal HDMI_TX_HS_sig: std_logic:='0' ;
74      signal HDMI_TX_VS_sig: std_logic:='0' ;
75      signal HDMI_TX_CLK_sig: std_logic:='0' ;
76      signal HEX0_sig: std_logic_vector (6 downto 0);
77      signal HEX1_sig: std_logic_vector (6 downto 0);
78      signal HEX2_sig: std_logic_vector (6 downto 0);
79
80
81 begin
82
83     dut: image_processor
84     port map (
85         RSTn => RSTn_sig,
86         clk => clk_sig,
87         KEY_ROTATE =>KEY_ROTATE_sig,
88         SW_ROTATION_DIR => SW_ROTATION_DIR_sig,
89         SW_IMAGE_ENA =>SW_IMAGE_ENA_sig,
90         SW_MODE =>SW_MODE_sig,
91         SRAM_D =>SRAM_D_sig,
92         SRAM_A => SRAM_A_sig,
93         SRAM_CEn =>SRAM_CEn_sig,
94         SRAM_OEn =>SRAM_OEn_sig,
95         SRAM_WEn => SRAM_WEn_sig,
96         SRAM_UBn => SRAM_UBN_SIG,
97         SRAM_LBn=> SRAM_LBn_Sig,
98         HDMI_TX => HDMI_TX_sig,
99         HDMI_TX_DE => HDMI_TX_DE_sig,
100        HDMI_TX_HS => HDMI_TX_HS_sig,
101        HDMI_TX_VS => HDMI_TX_VS_sig,
102        HDMI_TX_CLK => HDMI_TX_CLK_sig,
103        HEX0 =>HEX0_sig,
104        HEX1 =>HEX1_sig,
105        HEX2 =>HEX2_sig
106    );
107    dut2: sim_sram
108
109
110    generic map (ini_file_name=> C_image_file)
111    port map (
112        SRAM_ADDR          => SRAM_A_sig,
113        SRAM_DQ            => SRAM_D_sig,
114        SRAM_WE_N          => SRAM_WEn_sig,
115        SRAM_OE_N          => SRAM_OEn_sig,
116        SRAM_UB_N          => SRAM_UBN_sig,
117        SRAM_LB_N          => SRAM_LBn_sig,
118        SRAM_CE_N          => SRAM_CEn_sig
119    );
120
121

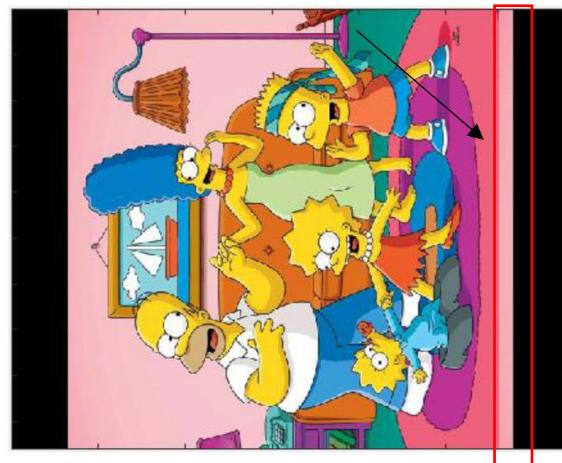
```

```
122
123
124
125     clk_sig <= not clk_sig after C_CLK_PRD /2;      -- clk_sig toggles every C_CLK_PRD/2 ns
126     RSTn_sig <= '0', '1' after 15 ns;
127
128 process
129 begin
130
131     for i in 0 to 10 loop
132         KEY_ROTATE_sig<='1' after 0 ns, '0' after 20 ms, '1' after 50 ms, '0' after 100 ms,
133         '1' after 200 ms, '0' after 1000 ms, '1' after 5000 ms;
134
135         SW_ROTATION_DIR_sig<= '1', '0' after 500 ns, '1' after 100 ms, '0' after 4000 ms;
136
137         SW_MODE_sig<= '1', '0' after 20 ns, '1' after 5000 ms;
138
139         SW_IMAGE_ENA_sig<='1', '0' after 100 ns, '1' after 1 ms;
140         wait for 10000 ms;
141     end loop;
142 end process;
143 end architecture;
144
145
```

## סיכון ומסקנות

### תיאור תקלות שעלו בדוח

1. פס שמויף בתמונה כאשר אנחנו ב90 מעלות



פתרון: מכיוון שמתחילה לספור מ16 בפיקסלים  
ו511 בשורות אז

עבור הפיקסלים : התנאי צריך להתחילה לספור  $64+64+64+64=495$  כדי שנגיע ל-  
כלומר צריך להוסיף 1 בכל שלב שאנו נכנסים לתנאי זה.  
עבור השורות: צריך להחסיר ב-1 בכל שלב.

```

if(H_CNT>c_image_start_h-c_image_start_v and H_CNT< c_image_end_h)
|   image_pixel_num<=image_pixel_num+1;----the index start from 1
end if;

if H_CNT=c_image_end_h then
|   if(V_CNT>0 and V_CNT< c_active_lines) then
|       image_line_num<=image_line_num-1;----the index start from
|       end if;
end if;

```

2. חוסר סyncron בין הבלוק של ה – controller push\_button ל- controller נוצר מצב שכאשר לווחצים על  
הלהצן KEY\_ROTATE אך אין שינוי בהיפיכת התמונה דבר זה נובע מחוסר סyncron בין שני  
הבלוקים.

פתרון : נגדיר "דגל" CONTROLLER שמטרתו ליצור את הסyncron בין שני הבלוקים.  
כלומר ניצור סיגנל שעולה ל-1 כאשר יש פולס מה-push\_button וירד ל-0 כאשר נכנסים  
לפעולה של ROTATE כלומר כאשר VS=1.

בצורה הבאה :

```

70
71     if (ROTATE = ACTIVE_HIGH)
72         sig_ROTATE <= ROTATE;
73     end if;
74

148
149     else-----here we starting use to rotate_key (short/long press)
150         if(sig_ROTATE=ACTIVE_HIGH) then
|             sig_ROTATE<=ACTIVE_LOW;

```

3. חלק מה ממשקים של הבלוק `image_processor` לא היו מאותחלים בהתחלה כדי לטפל בבעיה הכנסו ערך התחלתי לכל ממשק.

4. כאשר הרצינו את התמונה מתחלת לעבוד אוטומטית מבל' שלחצנו על ה-`rotate`

פתרון: היינו בטוחים שהגדכנו נכון את ה-SWITCH-S בבלוק של ה-`button_push` אך החלטנו לבדוק את הבעה בסיגל טפ מכיוון שלא היינו בטוחים באיזה בלוק הבעה ושם איתרנו את הבעה שכן הייתה ב-`button_push` כלומר ה-SWITCH לא הוגדר כראוי היה צריך להיות 0 לוגי ולא 1.

5. בהירות התמונה לא מוצגת כראוי



פתרון: ראשית חשוב לציין שהבעיה לקוחה המון זמן לפתור את הבעיה.

הפתרון הוא חוסר סוגרים בחישוב של הפונקציה אך חשוב לעבוד מסודר בכתיבת הקוד.

```
p_out := std_logic_vector(to_unsigned((to_integer(unsigned(p_in)) * 255) / c,p_out'length));
```

### מסקנות

1. כדי למצוא את הביעות השתמשוigli שנקרא סיגנל `signal tap` כדי חיוני שמאפשר לנו לאתר את הבעיה שmagiu מהמשתק או לשורת הקוד למשל `SWITCH` שנמצא ב-`button_push` שהוא צריך להציגו אוטו כ-0 לוגי ולא אחד או צמצום אפשרות של הבעויות שחשבנו שגרמו לחוסר סyncron בין הבלוק `button_push` של ה-`controller` בין הבלוק של ה-`push_button`.

2. יש לתכנן קודם על הדף תרשימי זרימה בעיקר לקודים מורכבים כמו ה-`datta_generator` מכיוון שלא עבדנו לפיה כלל זה נוכנסו להמון תקלות שנבעו מחוסר הבנה בכך לאחר שבנו תרשימים זרימה הדבר מאוד קל علينו בבניית הקוד.

3. לאחר צירבת התוכנית במידה ונתקלנו בעיות רצוי לבדוק את הבעיה לאתר תקלות רצוי לבדוק את הסימולציה לאותו הקוד ושם נוכל לאתר איפה בדיק יש בעיה. כמו למשל בעית האינדקסים שהתרחשה בזווית 90 של התמונה.

### סיכום

הפרויקט היה מאתגר ומורכב מאוד אך למדנו ממנו המון.

הפרויקט נתן לנו ניסיון רב בהתקנה של כתיבת הקוד איתור תקלות אפשריות, שימוש בכלים שתורמים מאוד לאייתור התקלות האפשריות. וניסיון בミימוש מתוכנה לחומרה.





