Create system-verilog bitmap

במסמך זה תיעוד והנחיות שימוש בפונקציה VerilogBitmapArray.m.

נכתב ע"י רון טייכנר, 25.03.2019

[מטרת ויכולות הפונקציה](#_qxo3gm1s2cw3)

[הנחיות שימוש](#_y0xyltmbboxy)

[כניסות לפונקציה](#_pnj09nkvfttb)

[דוגמא לאופן הקריאה לפונקציה](#_l9lcpetrvz9x)

[פירוט הכניסות לפונקציה](#_fwk8lq5moqyl)

[מוצא הפונקציה - הדפסות למסך](#_6d8rngxt8blj)

[הודעות בנושא תכונת שקיפות בתמונה](#_tzhrd3ra1qng)

[הודעת כתיבת קובץ sv](#_xf080y8ee91k)

[הודעות שגיאה / אי ביצוע פעולה אפשריות](#_g1yotyyfbv47)

[מוצא הפונקציה - גרפים](#_n0xaha7d0mha)

[מוצא הפונקציה - קובץ sv](#_7f395etl9b2u)

[ליבת המוצא עבור תמונה בגודל 5x5 קוונטיזציה ל - 8 ביט](#_atbk9hugo7j4)

[ליבת המוצא עבור תמונה בינארית בגודל 5x5](#_ld44senqu4n7)

[שילוב קובץ SV](#_u2hxkw1ltndj)

[דוגמת שימוש](#_ghmpg3bp6joa)

[תכולת הספרייה](#_t6edyvl5uzlh)

# 

# מטרת ויכולות הפונקציה

מטרת הפונקציה היא המרה של תמונה מפורמט סטנדרטי של קובץ תמונה אל מערך קובץ בשפת system-verilog.  
לאחר המרת התמונה ייכתב לספריית העבודה קובץ system-verilog ובו מקטע קוד המכיל מערך ובו ערכי התמונה (RGB או binary). את מקטע הקוד יש להעתיק אל קובץ system verilog מלא איתו אתם עובדים בפרויקט. אותו יש לשלב בפרויקט.

יכולות הפונקציה:

1. תמיכה בפורמטים הבאים: jpg, tif, png.   
   (פורמטים נוספים של תמונה נתמכים גם כן, אך רק jpg, tif, png נבדקו).
2. ביצוע חיתוך (crop) לתמונה לאזור עניין המוגדר ע"י המשתמש.
3. ביצוע שינוי גודל לתמונה לגודל (מספר שורות, מספר עמודות) המוגדר ע"י המשתמש
4. ביצוע קוונטיזציה לתמונה לתמונת צבע או לתמונת שחור לבן
5. קביעת רמת שקיפות בינארית עבור תמונות המכילות רמות שקיפות.

הפונקציה מדפיסה למסך את התמונה לאורך שלבי העיבוד השונים וכך למשתמש יש את האפשרות לשחק עם הפרמטרים השונים עד לקבלת התמונה הסופית (שתרשם לקובץ sv) לפי שביעות רצונו.

# 

# הנחיות שימוש

## כניסות לפונקציה - שימוש בקובץ exe

### דוגמא לאופן הקריאה לפונקציה

יש להכין שני קבצי טקסט הכוללים את כלל הכניסות לפונקציה. על קבצי הטקסט להיות ממוקמים באותה ספרייה בה ממוקמת הפונקציה VerilogBitmapArray.exe .

קובץ טקסט ראשון נקרא   
VerilogBitmapArrayTxtFileNameInpus.txt ומכיל את שמות הקבצים:  
שורה ראשונה - שם קובץ התמונה  
שורה שנייה - שם קובץ ה - sv שייכתב.  
תוכן קובץ לדוגמא:

"corn.tif"

"smileyBitMap.sv"

קובץ טקסט שני נקרא VerilogBitmapArrayTxtNumericInpus.txt ומכיל את השדות בסדר הבא:

sProcessing.binaryTransparencyTh = 1; % [%]

sProcessing.sCrop.enable = 1;

sProcessing.sCrop.xyPortions = [50,50]; % [%]

sProcessing.sCrop.xyCenter = [50,50]; % [%] % x values are left to   
 % right; y values are up to down

sProcessing.sResize.enable = true;

sProcessing.sResize.new\_xy = [32,32];

sProcessing.quantize\_nBits = 8; % {8 - 3Red, 3Green, 2Blue}

תוכן קובץ לדוגמא:

1

1

[25,25]

[50,45]

1

[32,16]

4

## כניסות לפונקציה - שימוש בקובץ m

### דוגמא לאופן הקריאה לפונקציה

inputImageFileName = "corn.tif";

outputVerilogFileName = "cornBitmap.sv";

sProcessing.binaryTransparencyTh = 1; % [%]

sProcessing.sCrop.enable = true;

sProcessing.sCrop.xyPortions = [50,50]; % [%]

sProcessing.sCrop.xyCenter = [50,50]; % [%] % x values are left to   
 % right; y values are up to down

sProcessing.sResize.enable = true;

sProcessing.sResize.new\_xy = [32,32];

sProcessing.quantize\_nBits = 8; % {8 - 3Red, 3Green, 2Blue}

VerilogBitmapArray(inputImageFileName,outputVerilogFileName,sProcessing)

הערה: דוגמא זו עובדת תקין במטלב גרסת R2018b.   
במטלב גרסת R2018a נדרש לקרוא לתמונה בעזרת גרש יחיד:

inputImageFileName = ‘corn.tif’;

### פירוט הכניסות לפונקציה

|  |  |
| --- | --- |
| inputImageFileName | מחרוזת - שם קובץ התמונה. על הקובץ להימצא בספריית העבודה או לחלופין ניתן לספק path מלא. דוגמא (matlab R2018b):  “corn.tif” “C:\corn.tif”  Matlab R2018a: ‘corn.tif’ |
| outputVerilogFileName | מחרוזת - שם הקובץ אליו יירשם מקטע הקוד בשפת system verilog ובו ה - bitmap של התמונה. דוגמא:  “cornBitmap.sv” |
| sProcessing.sCrop.enable | שדה בולאני, בהינתן שהוא true יבוצע crop לתמונה בהתאם לנתונים בשדות הבאים. |
| sProcessing.sCrop.xyCenter | וקטור בעל מימדים [1x2].  הכניסה הראשונה היא המיקום בציר המאוזן המהווה את המרכז סביבו מתבצע החיתוך. היחידות הן אחוזים, כלומר מספר בתחום [0,100]. הכניסה השנייה היא המיקום בציר המאונך המהווה את המרכז סביבו מתבצע החיתוך. היחידות הן אחוזים, כלומר מספר בתחום [0,100]. |
| sProcessing.sCrop.xyPortions | וקטור בעל מימדים [1x2].  הכניסה הראשונה היא חלק התמונה בציר המאוזן הנשאר בתמונה בעת ביצוע החיתוך. היחידות הן אחוזים, כלומר מספר בתחום [0,100].  הכניסה השנייה היא חלק התמונה בציר המאונך הנשאר בתמונה בעת ביצוע החיתוך. היחידות הן אחוזים, כלומר מספר בתחום [0,100]. |
| sProcessing.sResize.enable | שדה בולאני, בהינתן שהוא true יבוצע שינוי גודל (מספר שורות, מספר עמודות) לתמונה (או לתמונה לאחר crop) בהתאם לנתונים בשדות הבאים. |
| sProcessing.sResize.new\_xy | וקטור בעל מימדים [1x2].  הכניסה הראשונה היא מספר העמודות בתמונה שתירשם בקובץ sv.  הכניסה השנייה היא מספר השורות בתמונה שתירשם בקובץ sv. |
| sProcessing.quantize\_nBits | בשדה זה קובעים את אופן ביצוע בקוונטיזציה לתמונה. אפשריים הערכים הבאים:  8 - תיווצר תמונה בעלת הייצוג הבא: [3bitsRed, 3bitsGreen, 2bitsBlue]  4 - תיווצר תמונה בעלת הייצוג הבא: [2bitsRed, 1bitsGreen, 1bitsBlue]  1 - תיווצר תמונת שחור לבן בינארית |
| sProcessing.binaryTransparencyTh | סקלר בתחום [0,100].  חלק מתמונות בפורמט png תומכות בשקיפות. השקיפות בתמונה הינה ברמה משתנה.  ב - system verilog אנו תומכים עבור כל פיקסל בשקיפות בינארית - שקוף לחלוטין או פיקסל התמונה המקורית. בשדה זה אנו מכוונים את הסף - רק ערכי שקיפות הגדולים מהסף יהיו שקופים בקובץ sv המיוצר. היחידות הן אחוזים, כלומר מספר בתחום [0,100]. |

## מוצא הפונקציה - הדפסות למסך

### הודעות בנושא תכונת שקיפות בתמונה

הפונקציה מדפיסה למסך הודעה המציינת האם התמונה שהתקבלה מכילה שדה שקיפות.  
במידה וקיים שדה שקיפות תודפס ההודעה:

image has transparency; scale binary threshold using sProcessing.binaryTransparencyTh

במידה ולא קיים שדה שקיפות תודפס ההודעה:

image has no transparency

### הודעת כתיבת קובץ sv

בסיום פעולה תקינה של הפונקציה תודפס הודעה על כתיבת קובץ לדיסק. לדוגמא:

ngc6543aBitmap.sv written to disk

### הודעות שגיאה / אי ביצוע פעולה אפשריות

במידה ושדה enable של חיתוך / שינוי גודל הוא במצב false תתקבל הודעה:

sProcessing.sCrop.enable is false - image was not cropped

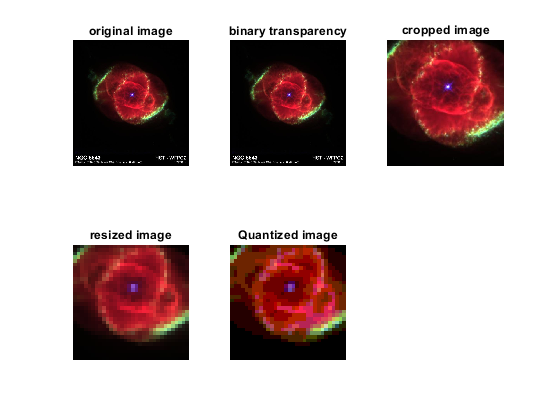
במידה ואחד השדות בקלט חסר תתקבל הודעה בסגנון:

sProcessing.sCrop.xyPortions or sProcessing.sCrop.xyCenter are not a [2x1] vector - image was not cropped

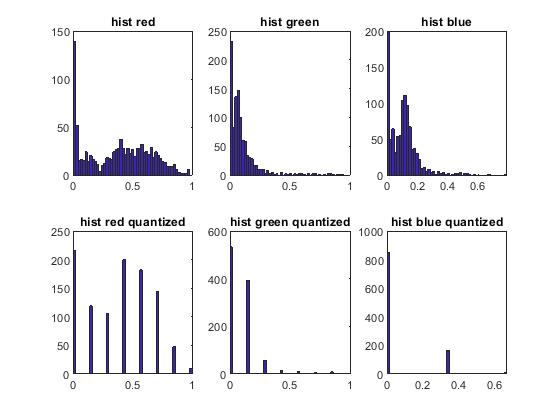
## 

## מוצא הפונקציה - גרפים

הפונקציה יוצרת שני גרפים.   
הראשון מכיל את התמונה שהתקבלה לאורך העיבוד:



השני מוקדש לנושא הקוונטיזציה ומדגים את פילוג הערכים לכל צבע לפני ואחרי קוונטיזציה:



## מוצא הפונקציה - קובץ sv

### ליבת המוצא עבור תמונה בגודל 5x5 קוונטיזציה ל - 8 ביט

**localparam** **int** OBJECT\_HEIGHT\_Y **=** 5**;**

**localparam** **int** OBJECT\_WIDTH\_X **=** 5**;**

**logic** **[**0**:** OBJECT\_WIDTH\_X **-**1**]** **[**0**:** OBJECT\_HEIGHT\_Y **-**1**]** **[**8**-**1**:**0**]** object\_colors **=** **{**

**{**8'hC5**,** 8'hC9**,** 8'hC9**,** 8'hA9**,** 8'hC9 **},**

**{**8'hC9**,** 8'hC9**,** 8'hC9**,** 8'hC9**,** 8'hC9 **},**

**{**8'h89**,** 8'hA5**,** 8'hA5**,** 8'hC5**,** 8'h85 **},**

**{**8'h85**,** 8'hC5**,** 8'hA5**,** 8'hA5**,** 8'h85 **},**

**{**8'hC9**,** 8'hC9**,** 8'hC9**,** 8'hA9**,** 8'hA9 **}**

**};**

**wire** **[**7**:**0**]** red\_sig**,** green\_sig**,** blue\_sig**;**

**assign** red\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**7**:**5**]** **,** 5'd0**};**

**assign** green\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**4**:**2**]** **,** 5'd0**};**

**assign** blue\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**1**:**0**]** **,** 6'd0**};**

### ליבת המוצא עבור תמונה בגודל 5x5 קוונטיזציה ל - 4 ביט

**localparam** **int** OBJECT\_HEIGHT\_Y **=** 5**;**

**localparam** **int** OBJECT\_WIDTH\_X **=** 5**;**

**logic** **[**0**:** OBJECT\_WIDTH\_X **-**1**]** **[**0**:** OBJECT\_HEIGHT\_Y **-**1**]** **[**4**-**1**:**0**]** object\_colors **=** **{**

**{**20'hCC888**},**

**{**20'hCCCC8**},**

**{**20'h88888**},**

**{**20'h88888**},**

**{**20'h88888**}**

**};**

**wire** **[**7**:**0**]** red\_sig**,** green\_sig**,** blue\_sig**;**

**assign** red\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**3**:**2**]** **,** 6'd0**};**

**assign** green\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**1**:**1**]** **,** 7'd0**};**

**assign** blue\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**][**0**:**0**]** **,** 7'd0**};**

### 

### ליבת המוצא עבור תמונה בינארית בגודל 32x16

**localparam** **int** OBJECT\_HEIGHT\_Y **=** 16**;**

**localparam** **int** OBJECT\_WIDTH\_X **=** 32**;**

**logic** **[**0**:** OBJECT\_WIDTH\_X **-**1**]** **[**1**\***32**-**1**:**0**]** object\_colors **=** **{**

**{**32'b00000000000000000000000001110000**},**

**{**32'b10011110011111001111100011110001**},**

**{**32'b00011000000000000000000000000000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b10001111000111110011111000111110**},**

**{**32'b10011111000111110011111100111100**},**

**{**32'b00010000000000000000000000000000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b10001111110001000111100011111000**},**

**{**32'b00001111110001000111100001111000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b00000000000000000000000000000000**},**

**{**32'b11110011111001111100011110001111**},**

**{**32'b00000001111000111110001111000011**}**

**};**

**wire** **[**7**:**0**]** red\_sig**,** green\_sig**,** blue\_sig**;**

**assign** red\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**]** **,** 6'd0**};**

**assign** green\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**]** **,** 7'd0**};**

**assign** blue\_sig **=** **{**object\_colors**[**offsetY**][**offsetX**]** **,** 7'd0**};**

### שילוב קובץ SV

שילוב הקובץ הינו זהה לכל אחת מהקוונטיזציות האפשריות.

זאת מפני שבכל מקרה הקובץ מסתיים בהצבת הפיקסל הנתון לשלושה משתנים ברוחב 8bit (אחד עבור כל צבע).

# 

# דוגמת שימוש

נרצה לטעון את התמונה, לבצע חיתוך סביב הנקודה המרכזית' להקטין לגודל של   
[32x32] ולבצע קוונטיזציה ל - 8bit.

הכניסות לפונקציה:

inputImageFileName = "ngc6543a.jpg";

outputVerilogFileName = "ngc6543aBitmap.sv";

sProcessing.sCrop.enable = true;

sProcessing.sCrop.xyPortions = [25,25]; % [%]

sProcessing.sCrop.xyCenter = [50,45]; % [%]

sProcessing.sResize.enable = true;

sProcessing.sResize.new\_xy = [32,32];

sProcessing.quantize\_nBits = 8;

sProcessing.binaryTransparencyTh = 1; % [%]

הרצת הפונקציה ע"י double click על הקובץ:

VerilogBitmapArray

עבור קובץ exe או הרצה במטלב של:

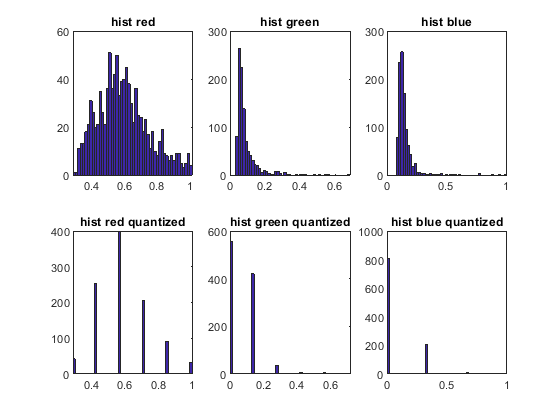
VerilogBitmapArray(inputImageFileName,outputVerilogFileName,sProcessing)

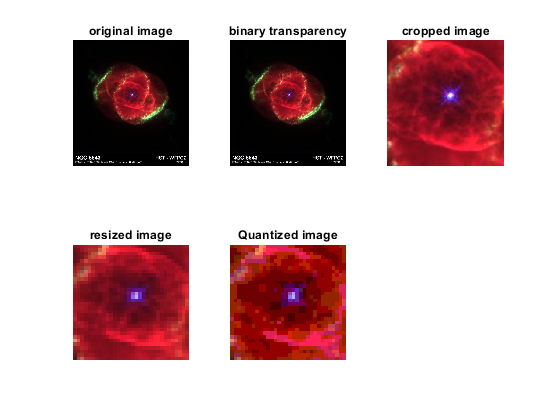
פלט הפונקציה למסך:

image has no transparency

ngc6543aBitmap.sv written to disk

הגרפים המתקבלים:





# 

# תכולת הספרייה

1. הפונקציה VerilogBitmapArray.exe ודוגמת הרצה המפורטת בשני קבצי הטקסט הכלולים בספרייה
2. הפונקציה VerilogBitmapArray.m ודוגמת הרצה בסקריפט RunVerilogBitmapArray\_example.m
3. תמונות בפורמטים jpg, png, tif
4. מסמך ההדרכה Create system-verilog bitmap.pdf