# ארכיטקטורת CPU

פרויקט גמר

Histogram &

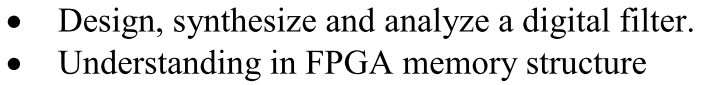
Edge detection

מגישים: אייר גלעד 300309937

דוד אבו 038109484

תוכן עניינים

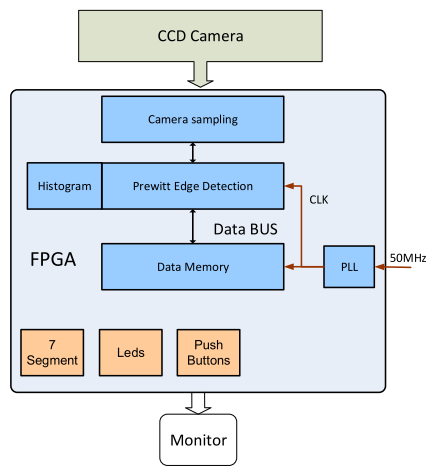
1. מטרת ותיאור הפרויקט..................................................................3
2. …………………………………………………………….Edge Detection4
3. היסטוגרמה.....................................................................................8
4. מסקנות.........................................................................................10

מטרת הפרויקט

תיאור הפרויקט

משימת הפרויקט הייתה לתכנן ולממש היסטוגרמה ו Edge Detection על גבי בקר FPGA.

קיבלנו נתוני תמונה ממצלמה שחוברה לבקר והוצאנו למסך תמונה מעובדת בצורת היסטוגרמה ומזהה קצוות, כאשר ההחלפה ביניהם בוצעה על ידי כפתור שבורר בין המצבים.



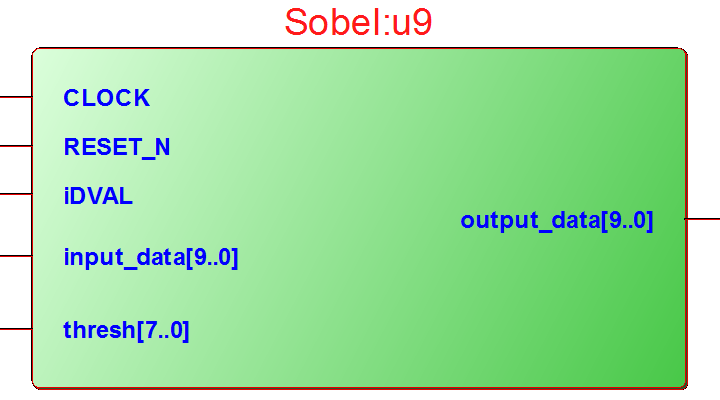
איור 0: סכימה כללית, Top Level של המערכת

Edge Detection

יחידת ה Edge detection היא היחידה אשר מתרגמת תמונה לתמונה בגווני שחור לבן שמסמנים קצוות בין עצמים בתמונה.

יחידה זו מזהה שינויים קיצוניים בין פיקסלים צמודים ע"י שימוש באלגוריתם של Sobel.

במקומות מסויימים נתייחס ליחידת ה Edge detection כפי שקראנו לה בקוד – Sobel.

נתחיל בתיאור חיצוני של היחידה:

איור 1: יחידת ה Edge detection

כניסות:

CLOCK שעון ראשי.

RESET\_N קו ריסט, מאפס את יחידות הזיכרון הפנימיות.

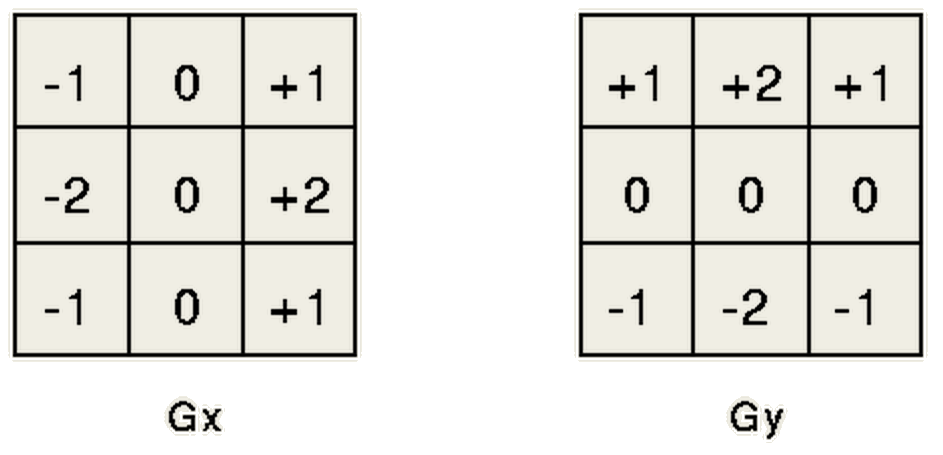
Idval קו בקרה המתריע על סיום פריים שלם שהתקבל מהזיכרון.

Input\_data[9..0] קווי ה input, דרכם מתקבלים הנתונים שיעובדו.

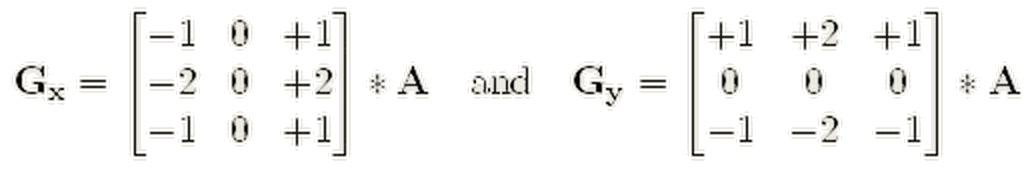
Thresh[7..0] קווי בקרה לשילטה על רמת החדות של עיבוד התמונה.

יציאות:

Output\_data[9..0] קווי ה output, מהם יוצאים הנתונים לאחר עיבוד התמונה.

באלגוריתם זה מתבצעת קונבולוציה בין שתי מטריצות (בגודל 3X3 במקרה שלנו) בצירי ה – X וה –Y : מטריצה אחת היא מטריצת הגרעין של Sobel עבור כל ציר, והמטריצה השניה היא מטריצה המכילה 9 פיקסלים מהתמונה.

איור 2: מטריצות הגרעין של Sobel

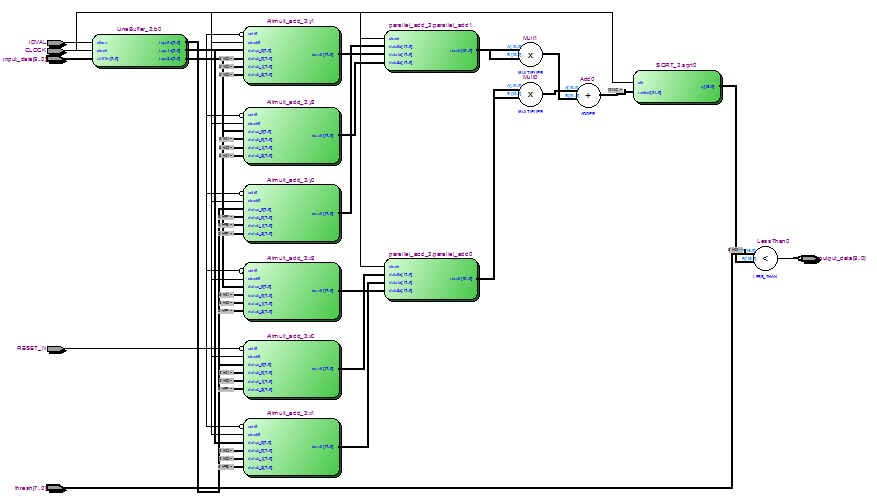
והחישוב שמתבצע(קונבולוציה):

איור 3: קונבולוציית המטריצות

כאשר לבסוף |G| = |Gx| + |Gy| (קירוב של משפט פיתגורס), ואם |G| גדול מערך כניסת ה Thresh, אזי נחשיב את הפיקסל כקצה.

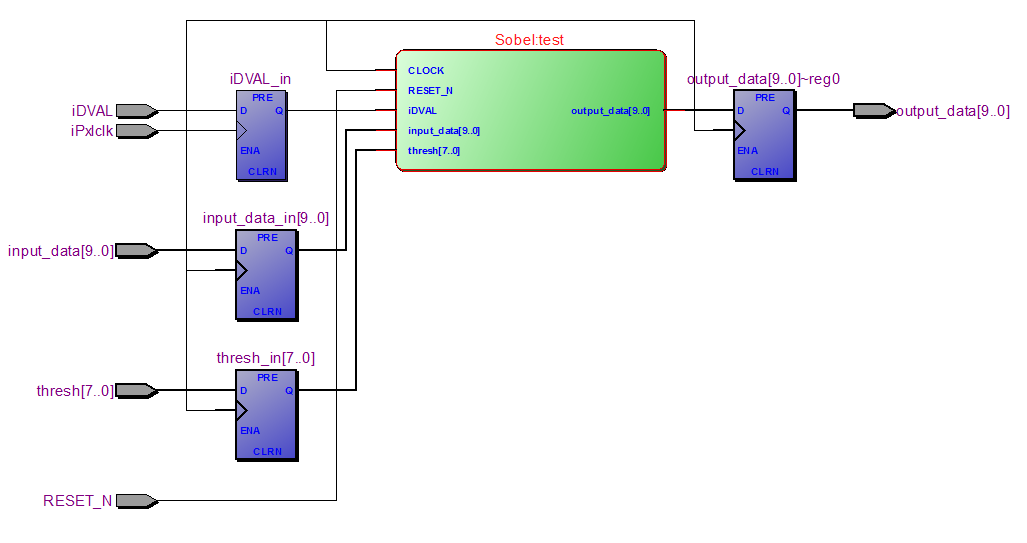
עבור המטריצה הנלקחת מהתמונה אנו נדרשים לשמור מידע זמני מספיק גדול שנשמר ע"י יחידת LineBuffer.

להלן תאור פנימי של יחידת ה Edge detection:



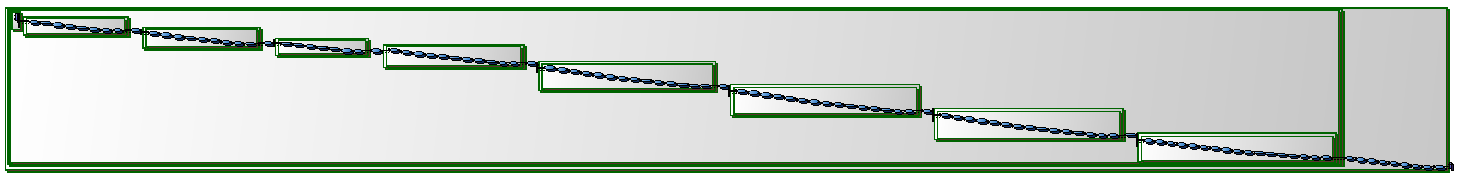
איור 4: תרשים RTL של יחידת ה Edge detection

עבור מציאת Critical path ותדר מקסימלי, נעטוף את היחידה ברגיסטרים:



איור 5: יחידת ה Sobel בין רגיסטרים

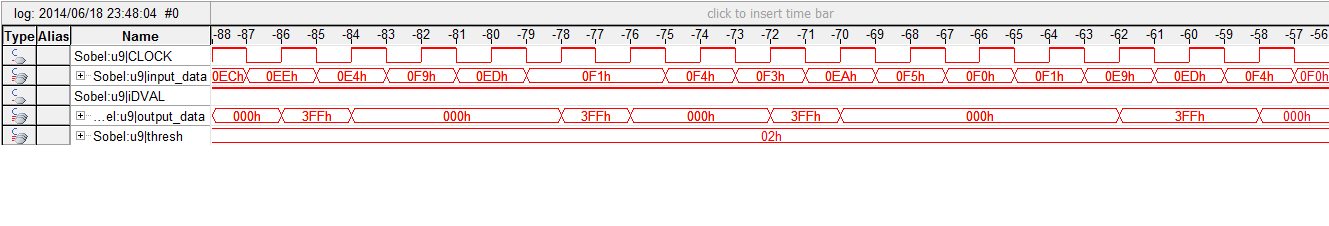
ה Critical path שמתקבל:



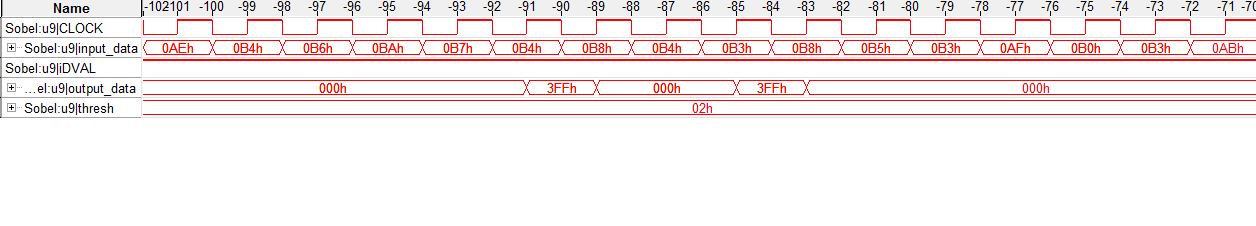
איור 6: Critical path עבור יחידת ה Sobel

והתדר המקסימלי שמקבל הוא: MHz 32.59

תמונות מסך מתוכנת ה Signal Tap:







סכמה כללית של ההיסטוגרמה והסבר

יצירת ההיסטוגרמה מתבצעת בשלושה שלבים:

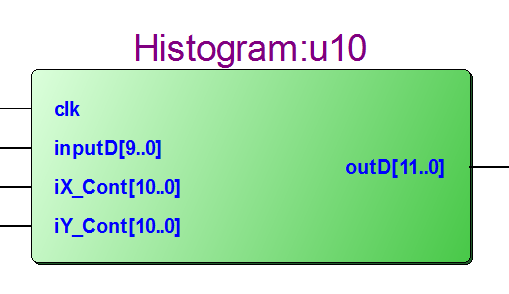
* עיבוד פיקסל – כאשר נכנס פיקסל של frame לרכיב, הספקטרום שלו מחולק ל128 גוונים,

החלוקה מתבצעת באופן הבא: מכל 8 גוונים "קרובים" נקבל גוון אחד

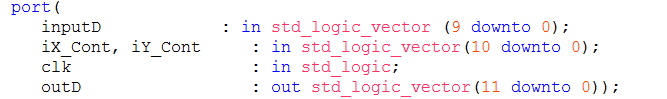
כאשר קיימים בסך הכל 1024 גוונים שונים, משמע 1024/8=128,

לכל גוון יש אקומולטור כשאר נקבל גוון מסוים נוסיף לאקומולטור אחד, מערך האקומולטורים – באפר שבסוף כל frame מחזיק את ההיסטוגרמה.

* נירמוליזציה של ערכי ההיסטוגרמה – בכדי שיהיה ניתן להציג על המסך את הערכים שסכמנו נצטרך (כל תא מצביע על גובה ציר Y) לעשות נירמול היות והגובה המקסימלי של המסך שניתן להציג הוא 480 פיקסלים וכאשר התמונה בעלת גוון אחיד נקבל באקומולטור ספציפי 640\*480 = 307200 , ובכדי שזה יהיה ערך עליון במסך נחלק ב 640 וכך נקבלה גובה מקסימלי של 480.
* הצגת ההיסטוגרמה - על מנת שיהיה ניתן להציג ולעבד בו זמנית קיים באפר אשר שומר את ההיסטוגרמה של ה frame הקודם, זאת אומרת בהינתן frame חדש נעתיק את ההיסטוגרמה של frame הקודם לבאפר ולכל frame נוכחי שעושים עיבוד נציג את ההיסטוגרמה השמורה בבאפר, כאשר רוחב המסך 640 פיקסלים והמערך הקיים בגודל 128 , 640/128 =5 לכן כל גוון המוצג הוא ברוחב 5 פיקסלים,

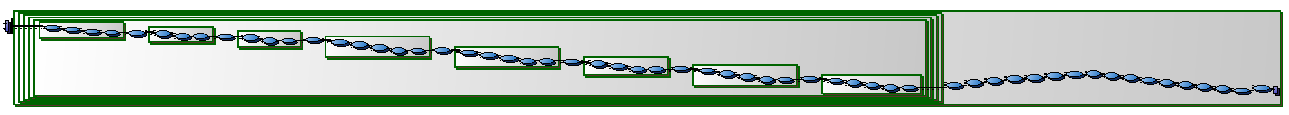


איור 7: סכימת יחידת ההיסטוגרמה



תרשים הRTL של ההיסטוגרמה הינו סבוך מאוד ודורש שטח רב בשביל להציגו בצורה מניחה את הדעת ולכן נוותר עליו במסמך הזה.

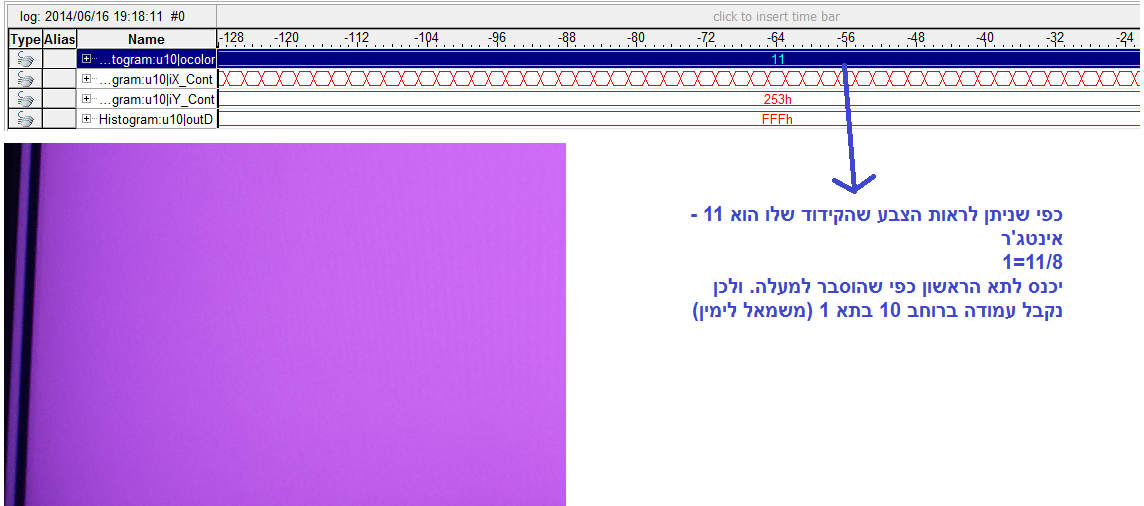
**מסלול קריטי של הרכיב :**



**איור 8:** Critical path עבור יחידת ההיסטוגרמה

**תדר עבודה מקסימלי: MHz 35.95**

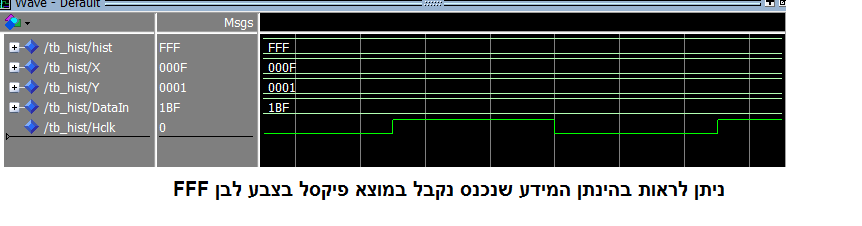
**Signal tap**



איור 9: תמונת מסך מתוכנת ה **Signal tap**

**ModelSim**

(הקבצים מצורפים בתיקיה)



איור 10: תמונת מסך מתוכנת ה **ModelSim**

מסקנות

מסמך זה תיאר את הפרויקט האחרון בקורס.

בפרויקט תיכננו ביצענו וניתחנו היסטוגרמה ומזהה קצוות.

בנוסף שולבו אלמנטים של זיכרון.

ביצוע הפרויקט המחיש את היתרון של ביצוע משימות בצורה חומרתית במקום תוכנתית.

בעוד שבפרויקט הקודם נדרשנו לממש מעבד ולכתוב עבורו תוכנה, בפרויקט הזה לקחנו שני אלגוריתמים – היסטוגרמה וזיהוי קצוות ומימשנו אותם בלבד על גבי בקר ה FPGA.

מכיוון שביצועי חומרה עבור פעולה ספציפית טובים בהרבה מאלה של תוכנה שמפעילה מעבד המבצע פעולות קבועות, המסקנה המתבקשת היא שאם ישנן פעולות שידוע מראש שנרצה לבצע רק אותן והרבה פעמים נוכל לעשות זאת בצורה חומרתית יעילה ביותר.

בנוסף, הפרויקט הכיר לנו את מונח ההיסטוגרמה ואת האלגוריתמים שמימשנו ופתח אותנו לתחום של עיבוד תמונה.

אייר ודוד