CPU Architecture

LAB 4 assignment

FPGA based Digital Design

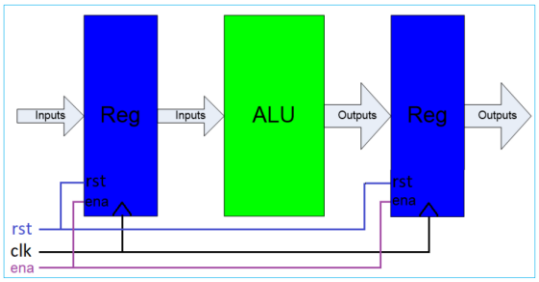
מגישים:

יבגני יגודין 324432988

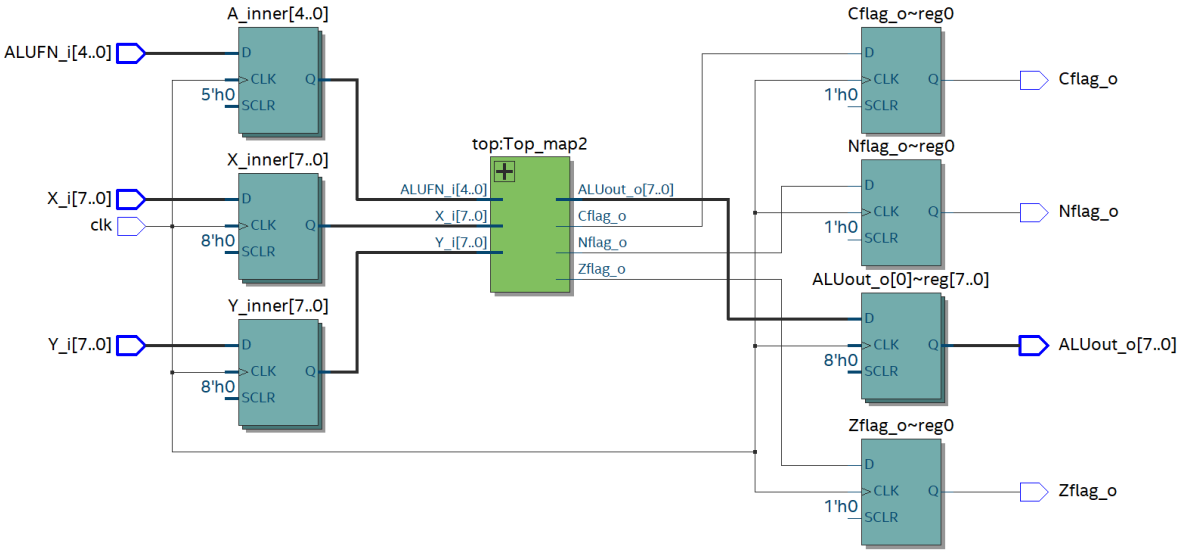
אור יעקבי 206827164

חישוב התדר המקסימלי:

נעטוף ברגיסטרים את ה-ALU שלנו (TOP) כפי שהתבקש באיור הבא:



איור 1א: הRTL של המערכת שלנו בצורה הזאת.



איור 1ב: נריץ את TIME QUEST וקיבלנו 108.53Mhz.

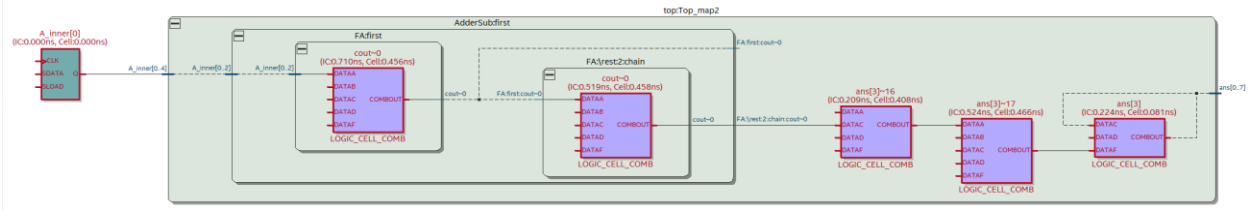
	Fmax	Restricted Fmax	Clock Name	Note
1	108.53 MHz	108.53 MHz	A_inner[3]	

תדר מקסימלי מקביל את המערכת שלנו בכך שאם יהיה יותר מידי גבוה, התוצאה של החישוב הסופי לא תספיק להתייצב בגלל מוגבלויות פיזיקליות ומכאן המערכת לא תיתן תשובה/תוצאה נכונה. לכל רכיב יש מדדים כמו זמן השהייה וכו'. ככלל לוקחים את מסלול צוואר בקבוק (קריטי) ולפי המוגבלויות שלו קובעים את התדר המקסימלי.

מסלול קריטי & frequency limiting operation

הבדיקה מראה שהמסלול הקריטי עובר דרך המחבר-מחסר .עובדה זאת לא מפתיעv כי הרכיב הזה מומש באופן תורי עם ה-FA שיש לו בפנים, ה-cout צריך עובר באופן גלי-תורי דרך כל ה-FA ומכאן ההאטה. לאומת זאת שיפטר ולוגיקה מומשו על ידי MUX-ים ובאופן מקבלי (או בקירוב מקבילי אם להזניח FANIN). זאת הסיבה שהמסלול קריטי של מחבר הוא כלול במסלול קריטי של כל המערכת.
מסלול קריטי של יחידה לוגית ושפטר, ניתן לראות בהמשך בRTL שלהם, שבגלל שמומשו באופן מקבלי וMUX-ים כל המסלולים הפנימיים זהים כמעט. לכן המסלול הקריטי של מחבר הוא הכי מעניין.

איור 2: המסלול הקריטי של המערכת.



Logic resource usage summary

Analysis & Synthesis Resource Usage Summary

<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	78
2		
3	Combinational ALUT usage for logic	110
1	-- 7 input functions	0
2	-- 6 input functions	42
3	-- 5 input functions	20
4	-- 4 input functions	22
5	-- <=3 input functions	26
4		
5	Dedicated logic registers	32
6		
7	I/O pins	33
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	A_inner[1]
12	Maximum fan-out	50
13	Total fan-out	622
14	Average fan-out	2.99

אופטימיזציות שנעשו בשביל ה-FPGA

כפי שהוגדר במטלה, נדרש "לצרוב" את ה-ALU על ערכת FPGA, איך ישנו צורך בהתאמות.
התאמה ראשונה שנעשה היא בעקבות כך המערכת שלנו בעלת 3 כניסות, אבל סט הקלט שנבחר הוא מתגים והסט הזה הוא יחיד.
1) הוחלט ליצור ממשק קלט משוטף ל-ALUFN,Y,X אבל בתוספת סט REGISTER+(ENA)KEY כך שכל פעם כשנכניס קלט למערכת, נוכל להגדיר שמה שהקלדנו הוא למשל X ואז בפעם הבאה שנרצה להקליד Y, שוב נקליד את הערך ונלחץ על KEY שמשויך לרגיסטר של Y ולבסוף באותו אופן נכניס את הקוד של פעולה רצויה.
(אין חשיבות לסדר הכנסת קלטים אפשר להתחיל מכל קלט, עקב הסימטריה והרגיסטרים ששומרים עת הערכים)

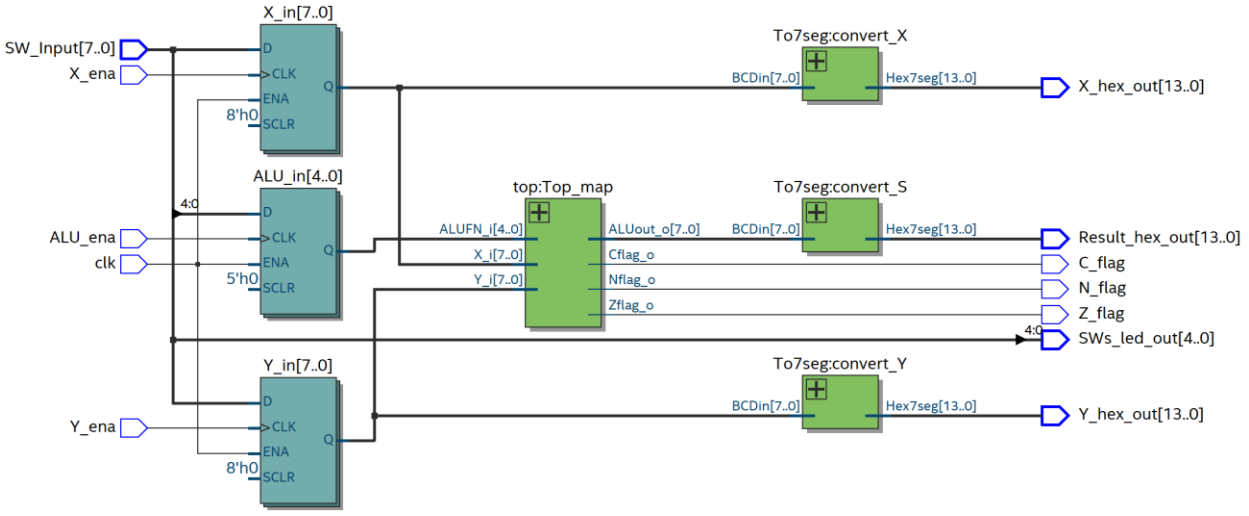
2) נדרש להציג על הלדים וגם על מסך HEX את שלושת הקלטים שנכנסו ל-ALU. Y,X נציג על מסך HEX בייצוג הקסדצימלי ואת ערך ALU נציג בייצוג בינארי על הלדים. ייצוג הלדים הוא ישיר, פשוט הוצאנו את האות החוצה כפלט נוסף שממופה ללדים. את הייצוג של Y,X נדרש להציג על גבי זוג מסכים 7 סגמנטים כל אחד. לכן יצרנו ישות חדשה שממירה 4 ביטים ל-7 ביטים לפי טבלת אמת שיצרנו תוך הסתכלות על מספרי סגמנטים בחוברת.
לאחר מכן הרחבנו את הישות שתמיר מ-8 ל-14 ביטים שיתאים למספר הביטים של X, Y.

3) באותו אופן כמו בסעיף הקדום עשינו עבור פלט המערכת. התוצאה תוצג בייצוג HEX על גבי מסך HEX אחרי ההמרה והדגים יוצגו כביט בודד כל אחד על גבי לדים.

לסיכום: ערכי X, Y והתוצאה יוצגו על 6 מסכי HEX, כל אחד מקבל 2 מסכי HEX. ערכי ALU והדגלים יוצגו על גבי לדים. הקלט יתקבל ממתגים כך שכל פעם נוכל להגיד מה הוקלד ולסמן איזה מה מבין ALUFN,Y,X הוקלד

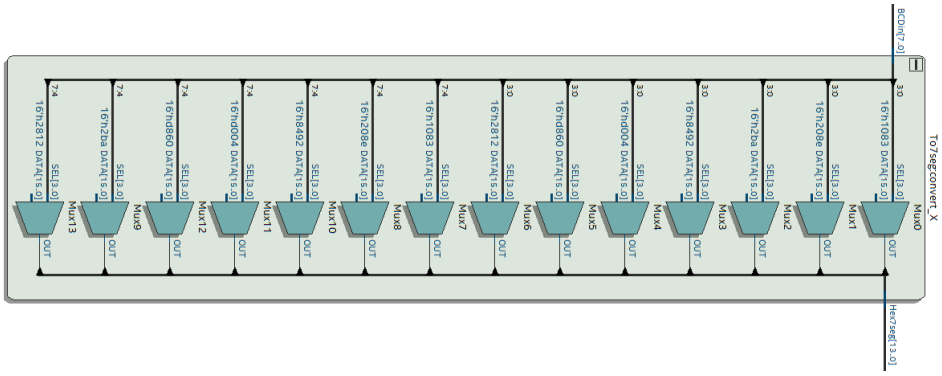
בעמוד זה וב-2 עמודים הבאים נציג את הRTL של הרכיבים.

איור 3: המעטפת שיצרנו סביב ה-ALU כדי להתממשק לערכה והתאמה ל-IO שלה.



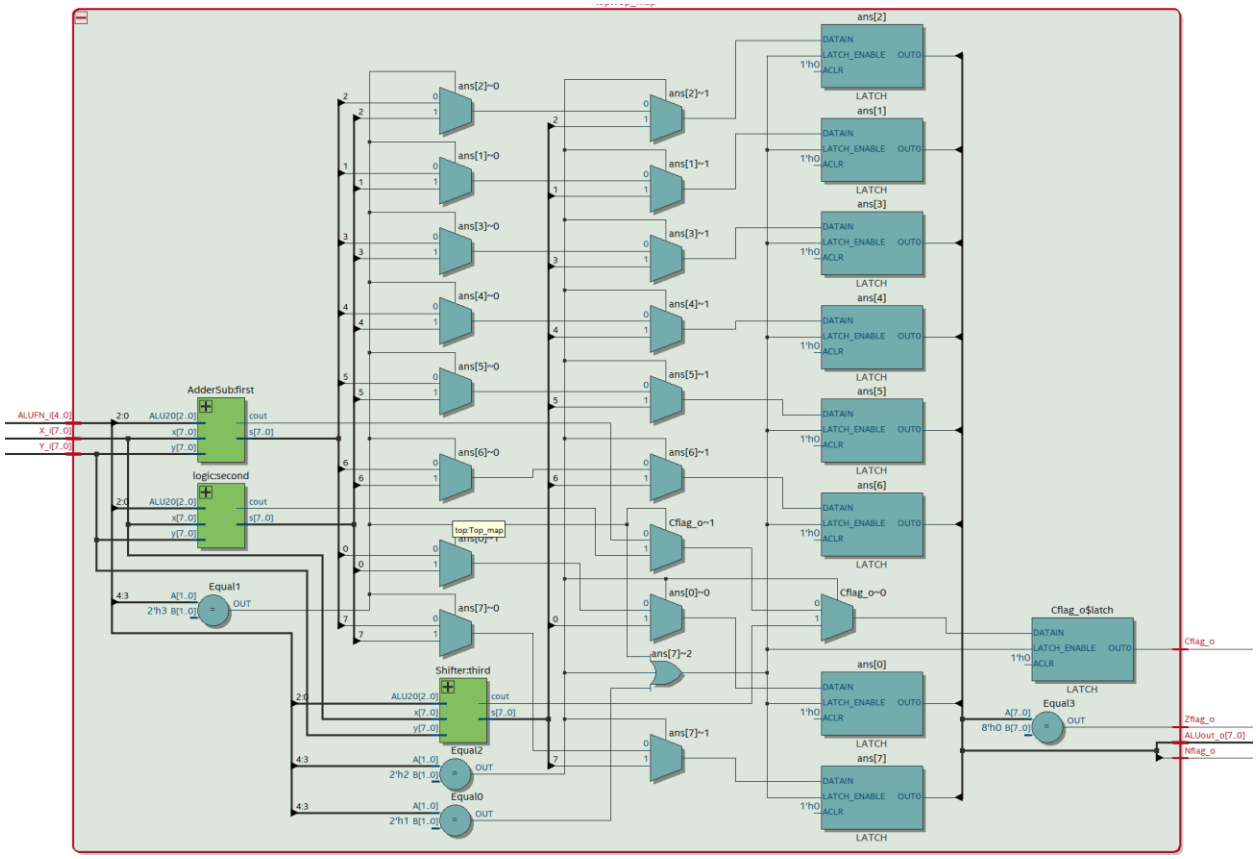
הערה: המערכת א-סינכרונית ולכן אין באמת CLK האיור נעשה לפני שתיקנו את זה.

איור 4: על מנת שנוכל להציג 8 ביטים על גבי זוג 7-seg-display נמיר אותם ל-14 ביטים באמצעות רכיבי MUX.



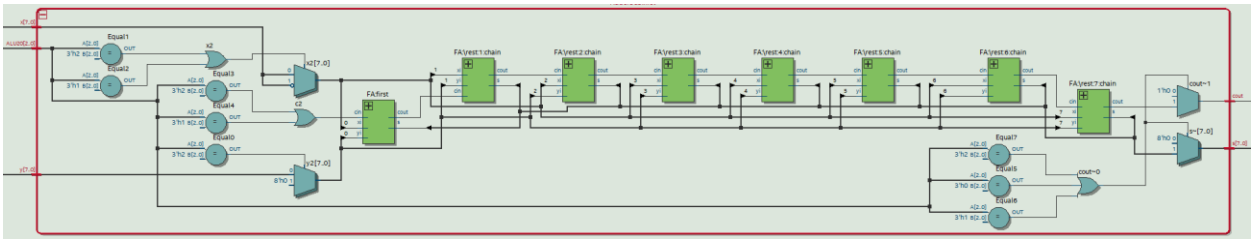
ה-top הוא למעשה ה-ALU ממעבדה 1 שלנו. בפלט יש לנו 2 אותות: Y,X ואות שמגדיר את הפעולה שמתבצעת ביניהם (חיבור, חיסור, הזזות, פעולות bit-wise). הפלט הוא התוצאה יחד עם דגלים. נשים לב שכאן יש לאטצ'ים והסיבה לקיומם היא שנדרש במטלה להחזיק את הפלט הקודם במידה והפקודה ש-ALU מקבלת אינה מוגדרת.

איור 5: ה-TOP או ALU.



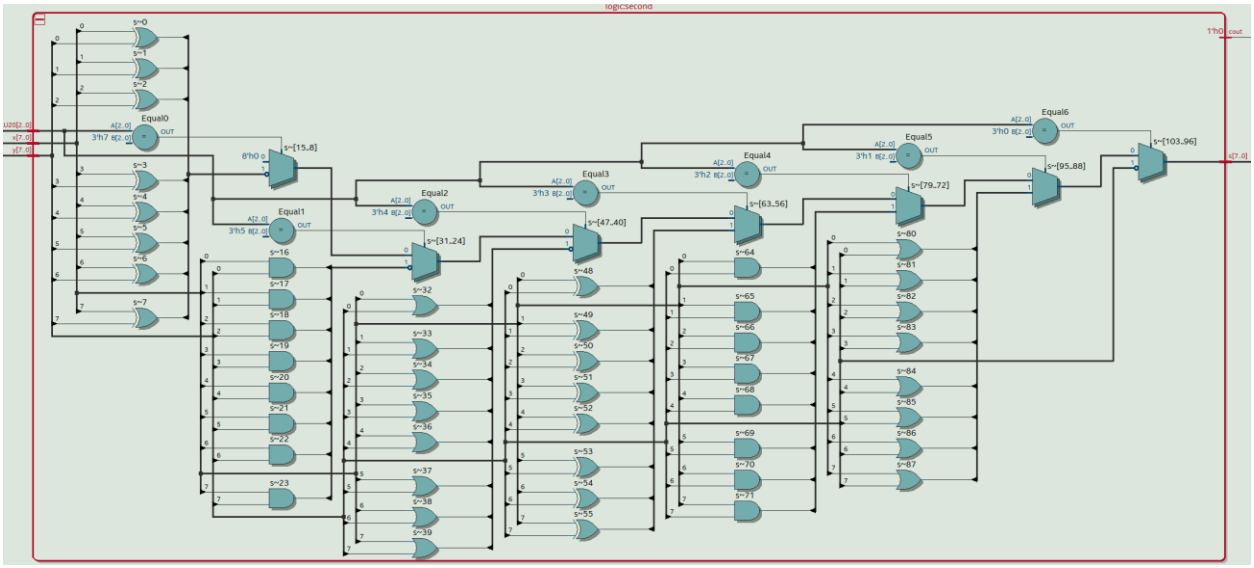
הרכיב הראשון ב-ALU הוא המחבר/מחסר בין 2 ווקטורים, מומש על ידי FA פשוטים משורשים.

איור 6: ה-מחבר/מחסר



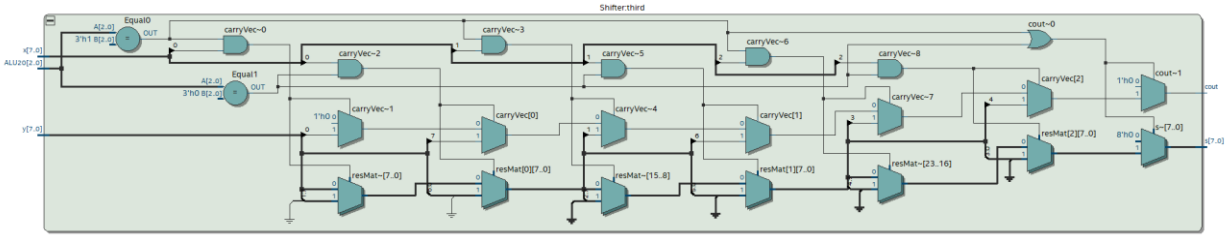
הרכיב הבא היא יחידת הלוגיקה, מבצעת פעולות bitwise פשוטות בין הקלטים.

איור 7: יחידה לוגית



הרכיב האחרון הוא השיפטר, מזיז את הקלט ימינה או שמאלה לפי הקלט השני, מומש על ידי רכיבי MUX.

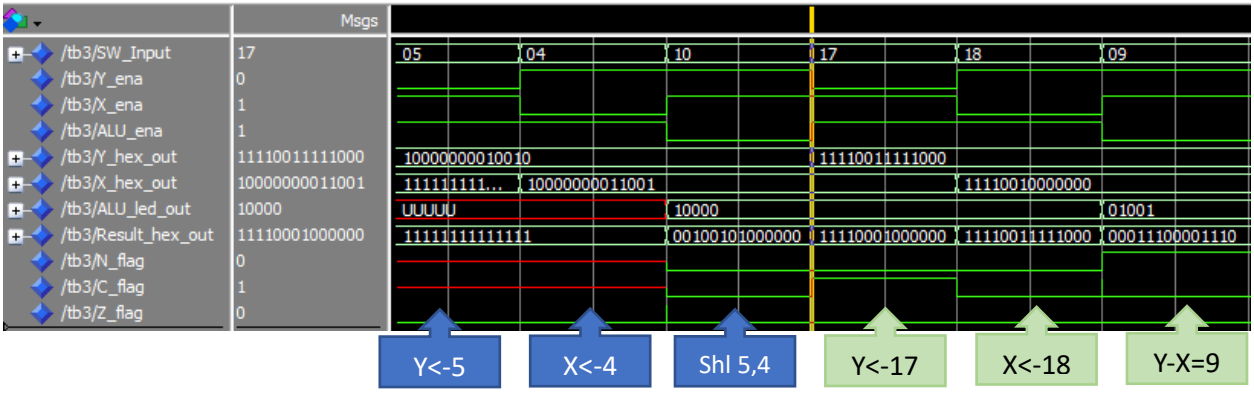
איור 8: יחידה שיפטר



The testbench

נבצע את הבדיקה על TB של MODELSIM על המערכת הכי גבוהה, זאת אומרת המערכת שכבר מותאמת לFPGA שלנו, נשים לב שעדיין מדובר ב-ALU שכבר דיברנו עליו וביצענו עליו הרבה בדיקות ולכן המטרה המרכזית של הTB הזה היא לבדוק את הלוגיקה הנוספת שהוספנו, מסכי HEX, לדים, מתגים.

איור 9: הצגת תוצאת הסימולציה



הקו מציג עם "HEX" בשם מוצגים אחרי המרה למסך HEX.

נריץ בדיקות:

בדיקה 1: SHIFTER

$$Y = \underbrace{1000000}_{0}\underbrace{0010010}_{5} = 05h\,;X = \underbrace{1000000}_{0}\underbrace{0011001}_{4} = 04h\,;Result = \underbrace{0010010}_{5}\underbrace{1000000}_{0} = 50h = 80$$

$$\rightarrow so\ the\ calculation\ is:\ 5 * 2^4 = 80 \rightarrow correct$$

בדיקה 2: ADDER

$$Y = \underbrace{1111001}_{1}\underbrace{1111000}_{7} = 17h\,;X = \underbrace{1111001}_{1}\underbrace{0000000}_{8}\,;Result = \underbrace{0001110}_{F}\underbrace{0001110}_{F} = FFh \underbrace{=}_{2's} -1$$

$$\rightarrow so\ the\ calculation\ is:\ 17h - 18h = FFh \underbrace{=}_{2's} -1 \rightarrow correct + Nflag$$

<u>הערה 1:</u> סוגריים מסולסלות אופקיות מסמנות המרה למסך HEX ולא ייצוג HEX. זה נעשה על ידי ישות converter שממירה באופן שמוצג משמאל (קטע קוד).

<u>הערה 2:</u> אותות מסוג EN ואות התוצאה מוצגים כ-not מהסיבה שכבה מוגדרת הערכה, וכפתורי KEY שקשורים ל-EN הם מקבלים 0 לוגי בלחיצה, בדומה גם הפלט למסכי HEX.

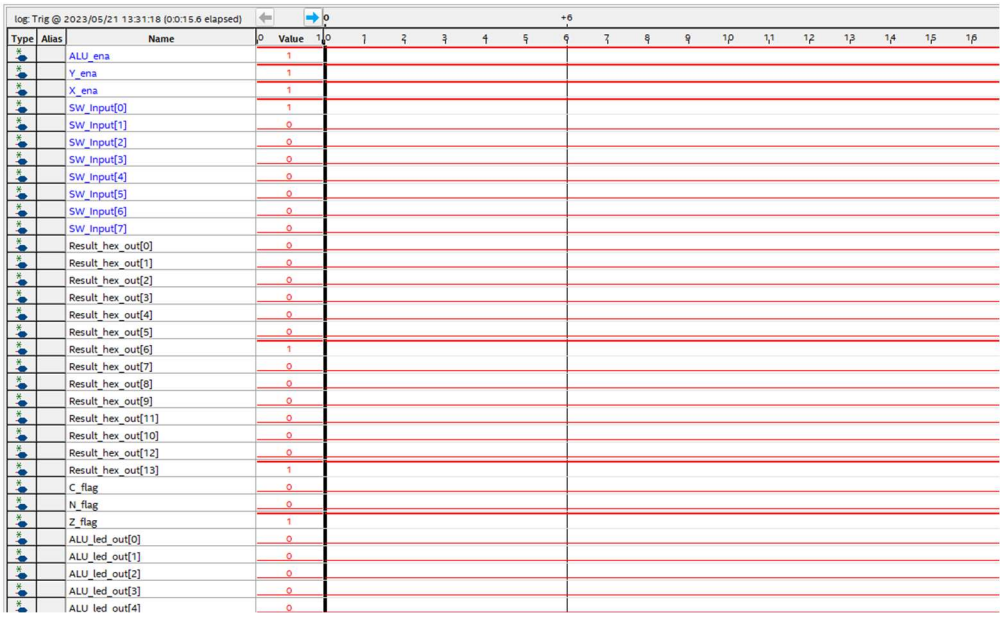
```
when "0000" =>Hex7seg(6 downto 0) <= not"0111111"; ---0
when "0001" =>Hex7seg(6 downto 0) <= not"0000110"; ---1
when "0010" =>Hex7seg(6 downto 0) <= not"1011011"; ---2
when "0011" =>Hex7seg(6 downto 0) <= not"1001111"; ---3
when "0100" =>Hex7seg(6 downto 0) <= not"1100110"; ---4
when "0101" =>Hex7seg(6 downto 0) <= not"1101101"; ---5
when "0110" =>Hex7seg(6 downto 0) <= not"1111101"; ---6
when "0111" =>Hex7seg(6 downto 0) <= not"0000111"; ---7
when "1000" =>Hex7seg(6 downto 0) <= not"1111111"; ---8
when "1001" =>Hex7seg(6 downto 0) <= not"1101111"; ---9
when "1010" =>Hex7seg(6 downto 0) <= not"1110111"; ---A
when "1011" =>Hex7seg(6 downto 0) <= not"1111100"; ---b
when "1100" =>Hex7seg(6 downto 0) <= not"0111001"; ---C
when "1101" =>Hex7seg(6 downto 0) <= not"1011110"; ---d
when "1110" =>Hex7seg(6 downto 0) <= not"1111001"; ---E
when "1111" =>Hex7seg(6 downto 0) <= not"1110001"; ---F
```

בדיקת signal tap

בשלה הוריפיקציה יש צורך לבדוק את האותות קלט-פלט של FPGA.
הערכה שלנו כן עובדת כנדרש, וה-SIGNAL TAP תופס את הקלט והפלט של הערכה אבל התצוגה של האות לא נראת לעין בגלל הסקופ' של הזמן שנמדד בננו שניות,כאשר מהירות הלחצות שלנו היא הבסגר גודל של שניות.

באיורים הבאים נציג הוכחה שהערכה התחברה ל-SIGNAL TAP

איור 10: כאשר אנו לוחצים על מקשים רואים בלייב שהעמודה VALUE מקבלית ערכים



איור 11: אפשרי לראות שאם מעבירים מתג אז יש תגובה ורואים מעין "debounce"