מעבדה 1

[מעבדה 1 2](#_Toc167527548)

[הקדמה 2](#_Toc167527549)

[תיאור הקוים הנכנסים למערכת 3](#_Toc167527550)

[מודלים 4](#_Toc167527551)

[Top 4](#_Toc167527552)

[מחבר מחסר ושולל 6](#_Toc167527553)

[שיפטר 8](#_Toc167527554)

[Logic 10](#_Toc167527555)

[סיכום כללי 12](#_Toc167527556)

# מעבדה 1

## הקדמה

במעבדה זו נלמדת השפה VHDL המתארת חומרה המשמשת לתיאור של מעגלים ספרתיים. בפרט, יושמה מערכת המבצעת מספר מודלים הכוללים מחבר, מחסר, מודל האחראי על פעולות לוגיות בין זוג וקטורים שיפטרים ועוד. בעזרת פקודות הניתנות לALUFN, קו וקטורי הנכנס למערכת, הלה מיישמת כל אחת מהפעולות הנזכרות מעלה על שני וקטורי כניסה אחרים, X,Y.

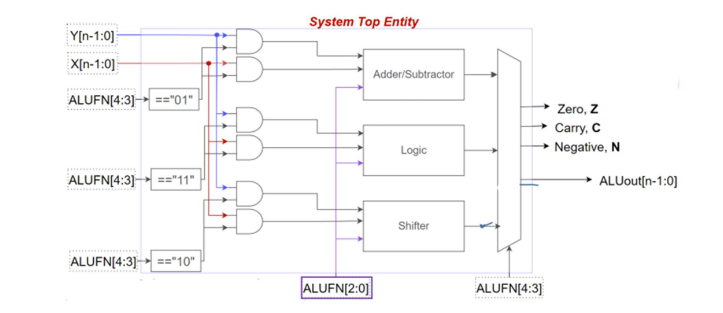


Figure 1 -סכימת המערכת שיושמה

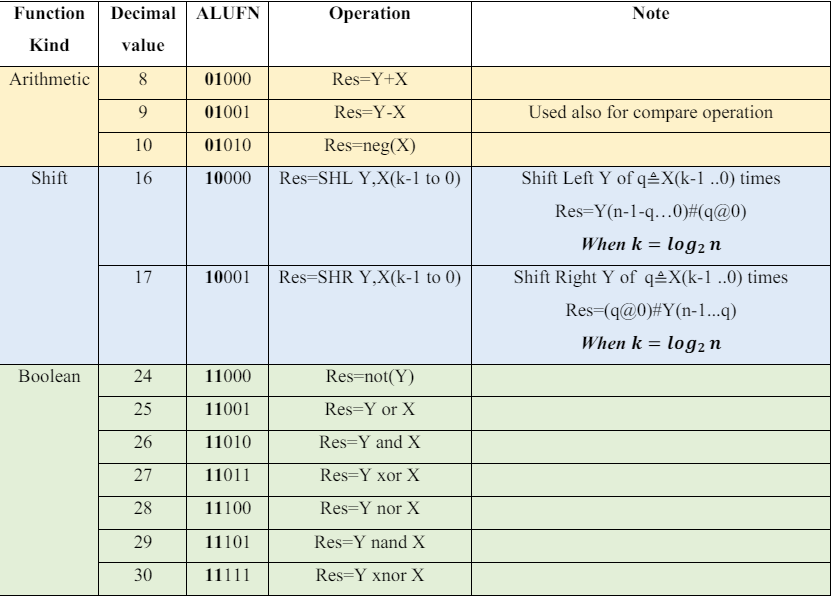


Figure 2 - תיאור מלא לאופן פעולת המערכת

### תיאור הקוים הנכנסים למערכת

ישנם שלושה קוים עיקריים אשר נכנסים למערכת:

* וקטור X
* וקטור Y
* ALUFN

**Y[n-1:0]**

* תיאור: קו כניסה למערכת המייצג וקטור בן n ביטים.
* שימוש: וקטור זה מספק את אחד הערכים שישמשו בפעולות האריתמטיות או הלוגיות במודולים הפנימיים של הALU

**X[n-1:0]**

* תיאור: קו כניסה נוסף למערכת המייצג גם הוא וקטור בן n ביטים.
* שימוש: וקטור זה מספק את הערך השני שישמש בפעולות האריתמטיות או הלוגיות במודולים הפנימיים של הALU.

**ALUFN[4:0]**

* תיאור: קו כניסה המייצג קוד פקודה בן 5 ביטים המגדיר את הפעולה שהALU צריכה לבצע.
* שימוש:
  + **ALUFN[4:3]** שני הביטים הגבוהים משמשים לבחירת המודול הפנימי שיבצע את הפעולה.
    - "01" מפעיל את מודול ה Adder/Subtractor
    - "10" מפעיל את מודול ה Shifter
    - "11" מפעיל את מודול ה Logic .
  + **ALUFN[2:0]** שלושת הביטים הנמוכים מגדירים את סוג הפעולה הספציפית בתוך המודול הנבחר. לדוגמה, חיבור, חיסור, הזזה שמאלה או ימינה ופעולות לוגיות.

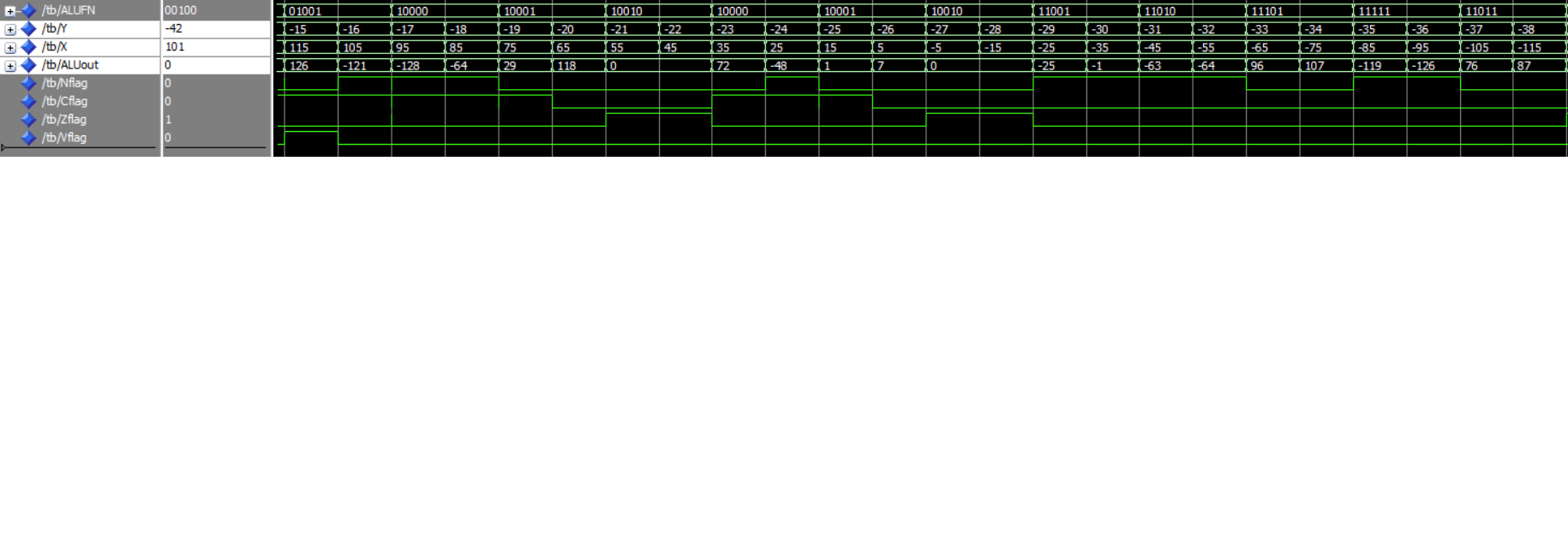
## מודלים

### Top

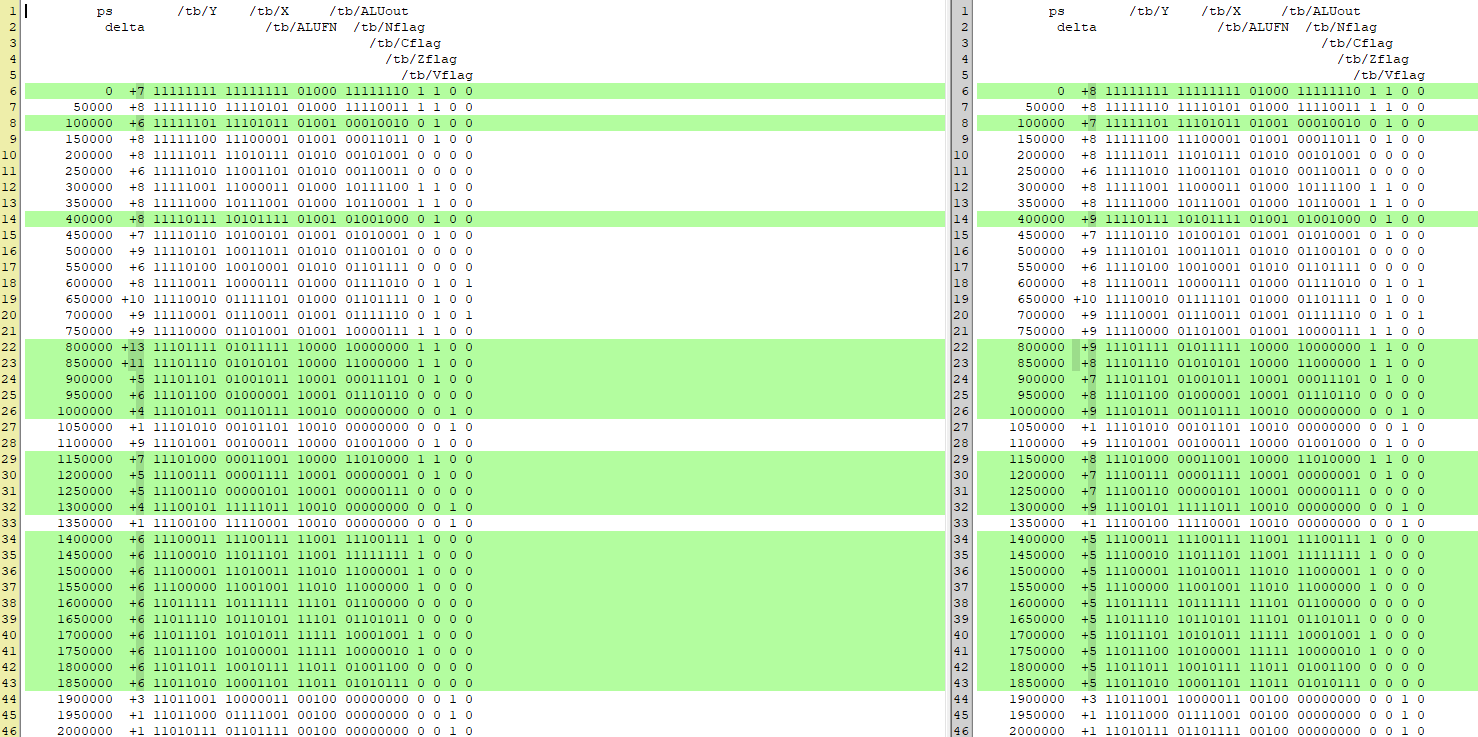
1. **פרמטרים גנריים**:
   * :n גודל וקטור הכניסות והיציאות.
   * :k מספר ביטים לייצוג לוגריתם של n.
   * :m ערך מחושב המבוסס על k.
2. **כניסות**:
   * Y\_i, X\_i וקטורי כניסות בגודל n.
   * ALUFN\_i וקטור כניסה בגודל 5 ביטים המגדיר את הפונקציה האריתמטית/לוגית שתבוצע.
3. **יציאות**:
   * ALUout\_o וקטור יציאה בגודל n .
   * Nflag\_o, Cflag\_o, Zflag\_o, Vflag\_o דגלי יציאה לציון תוצאות חישוב מיוחדות (שליליות, נשיאה, אפס, וחריגה).

**מבנה הארכיטקטורה:**

1. **קבועים וסוגים**:
   * NUM\_OF\_MODULES מספר המודולים הפנימיים.
   * vector, matrix הגדרות סוגים עבור וקטורים ומטריצות של STD\_LOGIC
2. **אותות פנימיים**:
   * אותות עבור כניסות ויציאות המודולים הפנימיים.
   * אותות עבור דגלי חישוב מיוחדים כמו Vflag\_add וVflag\_sub.
3. **אינסטנציות של תת-מודולים**:
   * AdderSub\_inst מבצע חיבור/חיסור.
   * Shifter\_inst מבצע פעולות הזזה.
   * Logic\_inst מבצע פעולות לוגיות.
4. **היגיון חיווט והפעלה**:
   * חיווט כניסות למודולים לפי ערכי ALUFN\_i
   * קביעת ערכים לדגלים בהתאם לתוצאות החישוב.
5. **היגיון יציאה**:
   * בחירת תוצאת המודול המתאים והעברתה ליציאה ALUout\_o
   * קביעת ערכים לדגלי היציאה Nflag\_o, Zflag\_o, Cflag\_o, ו- Vflag\_o
6. **הערות כלליות**
   * כדי לאפשר מינימום אנרגיה בזמן פעולה, רק מצב אחד יכול להיות דולק, וכאשר אף מצב לא דולק ישנם שיערי כניסה שיקבעו כי המערכת כבויה
7. **תוצאות סימולציה**

****

* + אפשר לראות את פעולות המודל בעת כניסות שונות, שהרי הוא עושה פעולות שונות לפי הטבלה בהקדמה.



* + להלן התוצאות בהשוואה למודל הזהב.

### מחבר מחסר ושולל

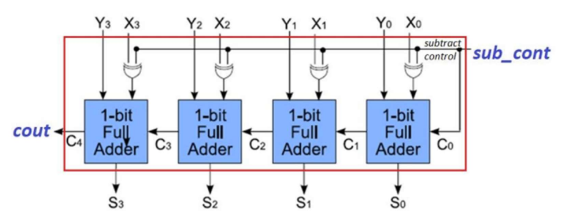


Figure 3- שרטוט של המודל המיושם

1. **פרמטרים גנריים**:
   * n גודל וקטור הכניסות והיציאות, ברירת מחדל היא 8 ביטים.
2. **כניסות**:
   * x\_adderSub\_in, y\_adderSub\_in וקטורי כניסות בגודל n ביטים.
   * ALUFN וקטור כניסה בגודל 3 ביטים המגדיר את הפעולה שתבוצע (חיבור או חיסור).
3. **יציאות**:
   * adderSub\_cout יציאת נשיאה (Carry Out).
   * adderSub\_out וקטור יציאה בגודל n ביטים המייצג את תוצאת החישוב.

**מבנה הארכיטקטורה:**

1. **קבועים וסוגים**:
   * הגדרת סוגים עבור וקטורים של STD\_LOGIC
   * אותות פנימיים כגון reg, x\_adderSub\_xor, y\_adderSub\_gated ו sub\_control.
2. **אותות פנימיים**:
   * sub\_control אות המשמש לשליטה האם מתבצע חיסור (ערך '1') או חיבור (ערך '0').
   * x\_adderSub\_xor וקטור שיכול להיות מושלך עם sub\_control עבור פעולת חיסור.
   * y\_adderSub\_gated וקטור שמאפשר שליטה על הערך של y\_adderSub\_in ל היות 0 במקרים של פעולות לא חוקיות.
3. **תהליך לוגי**:
   * קביעת ערך sub\_control בהתאם ל ALUFN
   * קביעת ערך y\_adderSub\_gated בהתאם לערך ALUFN
   * הפקת וקטור x\_adderSub\_xor עם אפשרות להיפוך ביטים עבור חיסור.
4. **אינסטנציות של סכום מלא (FA)**
   * **first** אינסטנציה של סכום מלא לביט הראשון עם כניסת נשיאה ראשונית (sub\_control)
   * **Rest** אינסטנציות של סכום מלא עבור כל הביטים הנותרים, מקשרים כל ביט עם הנשיאה מהביט הקודם.
5. **היגיון יציאה**:
   * יציאת נשיאה (adderSub\_cout) נקבעת מהביט האחרון של הוקטור reg .
6. **תוצאות:**

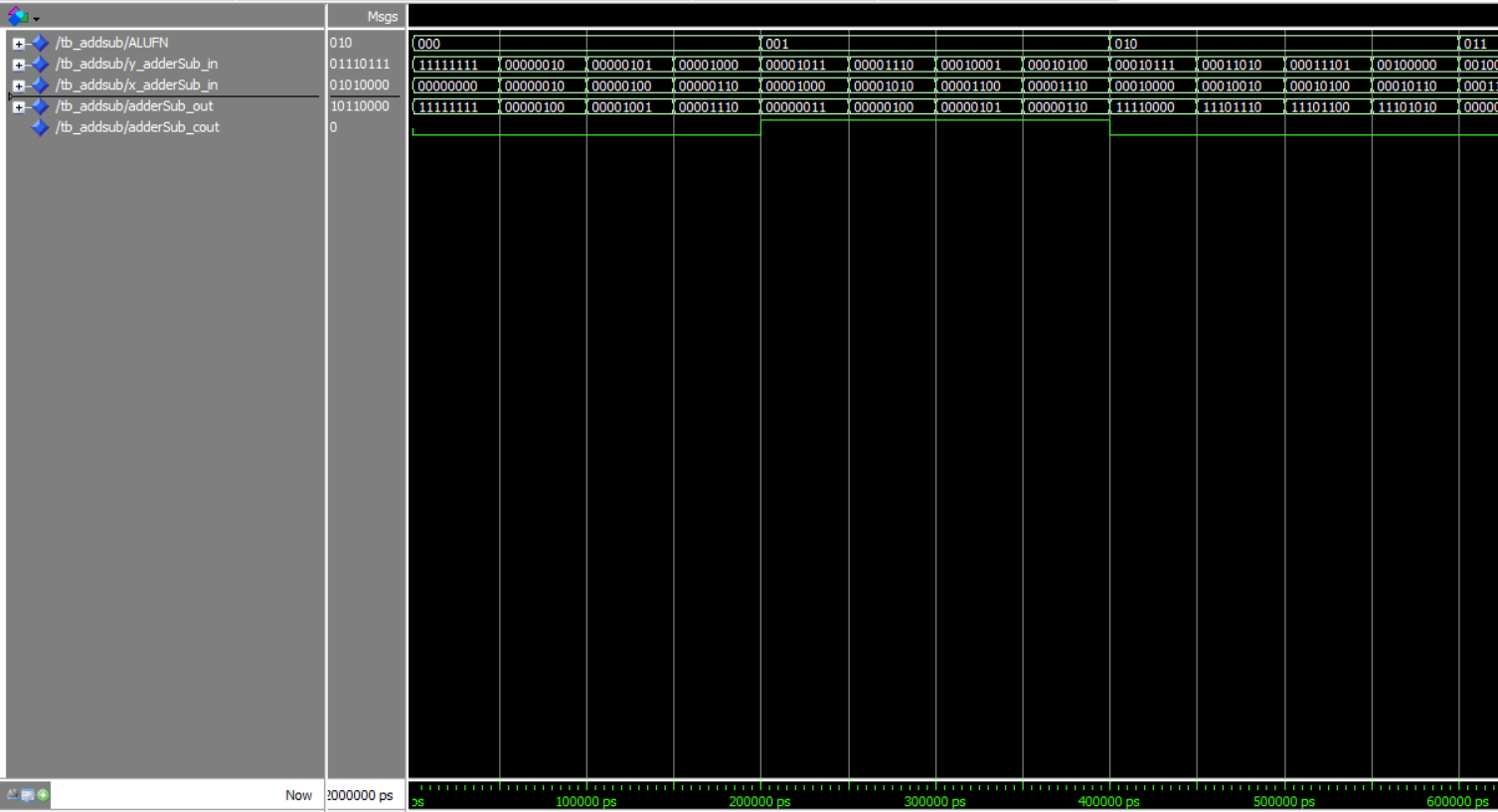


Figure 4 - אינפוטים תקינים של המודל



Figure 5 - אינפוטים לא תקינים של המודל

### שיפטר

**מבנה כללי:**

המודול Shifter מבצע פעולות של הזזה שמאלה (Shift Left) והזזה ימינה (Shift Right) על וקטור כניסה בגודל n ביטים, כאשר כמות ההזזה מוגדרת על ידי וקטור כניסה נוסף בגודל k ביטים.

**רכיבי Shifter**

1. **פרמטרים גנריים**:
   * n גודל וקטור הכניסות והיציאות, ברירת מחדל היא 8 ביטים.
   * k מספר ביטים לייצוג כמות ההזזה (log2(n)), ברירת מחדל היא 3 ביטים.
2. **כניסות**:
   * y\_Shifter\_in וקטור כניסה בגודל n ביטים המייצג את הנתונים שיש להזיז.
   * x\_Shifter\_in וקטור כניסה בגודל k ביטים המייצג את כמות ההזזה.
   * ALUFN וקטור כניסה בגודל 3 ביטים המגדיר את סוג הפעולה שתבוצע (הזזה שמאלה או ימינה).
3. **יציאות**:
   * Shifter\_cout ביט יציאה המייצג את הביט ש"זז החוצה" במהלך ההזזה.
   * Shifter\_out וקטור יציאה בגודל n ביטים המייצג את תוצאת ההזזה.

**מבנה הארכיטקטורה:**

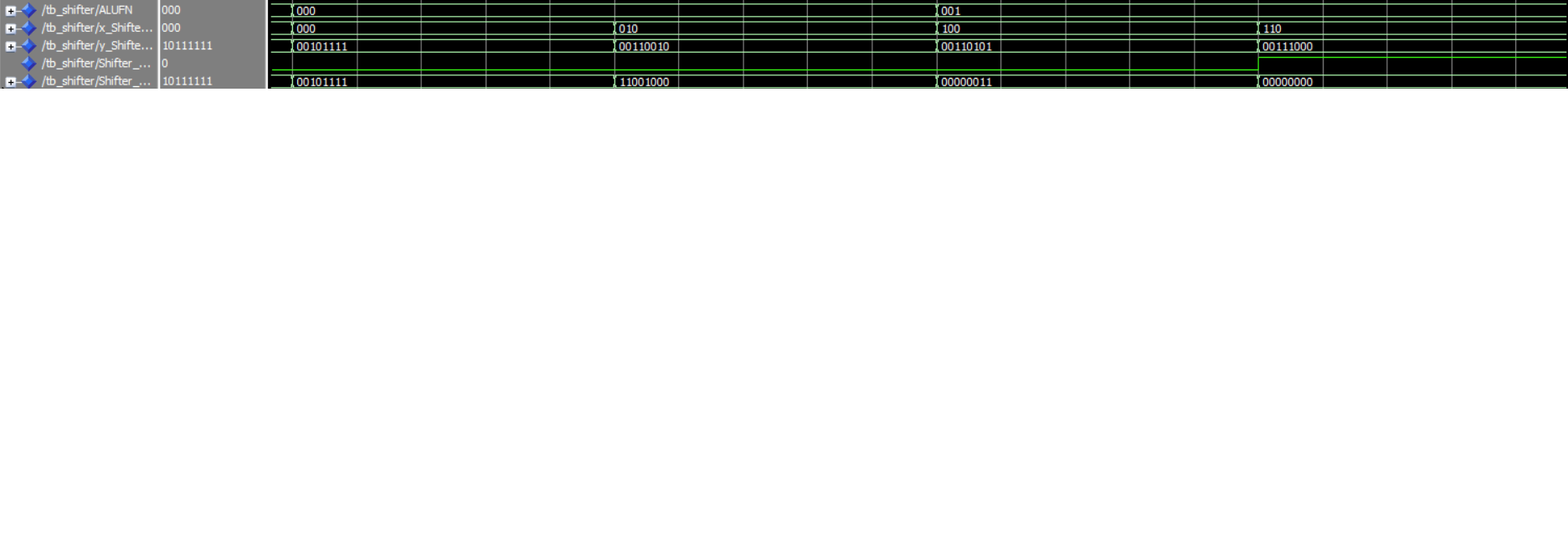
1. **קבועים וסוגים**:
   * הגדרת סוגים עבור וקטורים של std\_logic\_vector
   * הגדרת מטריצה עבור הוקטורים של הביטים המוזזים שמאלה וימינה.
2. **אותות פנימיים**:
   * Shifter\_cout\_vector\_left, Shifter\_cout\_vector\_right וקטורים לאיסוף הביטים שנזזו החוצה במהלך ההזזה.
   * row\_left, row\_right מטריצות עבור התהליך ההזזה שמאלה וימינה בהתאמה.
3. **תהליך לוגי**:
   * **הזזה שמאלה**: כל ביט בוקטור y\_Shifter\_in מוזז שמאלה, והביט שזז נאסף בוקטור Shifter\_cout\_vector\_left.
   * **הזזה ימינה**: כל ביט בוקטור y\_Shifter\_in מוזז ימינה, והביט שזז נאסף בוקטור Shifter\_cout\_vector\_right.
4. **הפקת תוצאה ויציאה**:
   * **Shifter\_out** התוצאה הסופית של ההזזה (שמאלה או ימינה) נקבעת לפי ערך x\_Shifter\_in ו- ALUFN.
   * **Shifter\_cout** הביט ש"זז החוצה" במהלך ההזזה נקבע לפי ערך x\_Shifter\_in ו- ALUFN.
5. **הערות**
   * המודל מיושם באופן דומה לסרטוני ההדרכה, בעזרת מטריצת עזר לתוצאה ווקטור עזר לקרי.
6. **תוצאות**

Figure 6 - תוצאות של אינפוטים תקינים המראות גם את הקרי

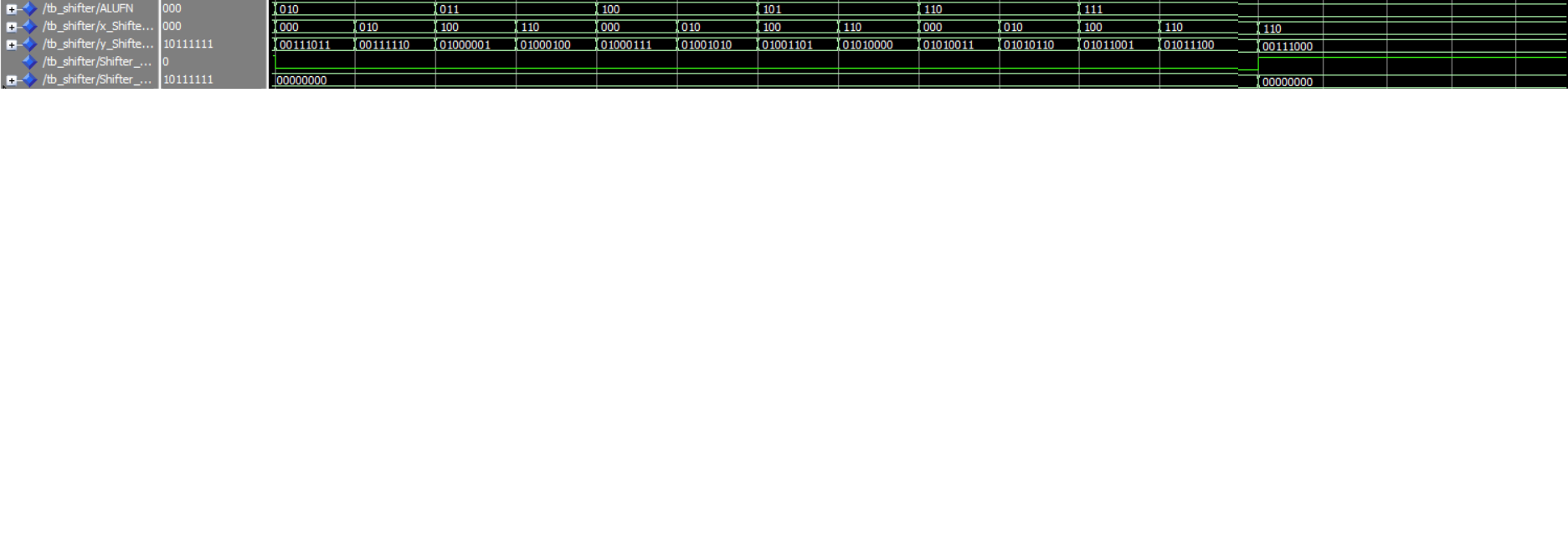


Figure 7 - תוצאות של אינפוטים לא תקינים

### Logic

**מבנה כללי:**

המודול Logic מבצע פעולות לוגיות על שני וקטורי כניסה של n ביטים, כאשר סוג הפעולה נקבע על ידי וקטור ALUFN . המודול מסוגל לבצע מגוון פעולות לוגיות כמו NOT, OR, AND, XOR, NOR, NAND, ו-XNOR.

**רכיבי Logic**

1. **פרמטרים גנריים**:
   * n גודל וקטור הכניסות והיציאות, ברירת מחדל היא 8 ביטים.
   * OP מספר הפעולות הלוגיות הנתמכות פלוס אחת (עבור תוצאה של אפסים), ברירת מחדל היא 8.
2. **כניסות**:
   * x\_logic, y\_logic וקטורי כניסה בגודל n ביטים עליהם מתבצעות הפעולות הלוגיות.
   * ALUFN וקטור כניסה בגודל 3 ביטים המגדיר את הפעולה הלוגית שתבוצע.
3. **יציאות**:
   * Logic\_out וקטור יציאה בגודל n ביטים המייצג את תוצאת הפעולה הלוגית.

**מבנה הארכיטקטורה:**

1. **סוגים ואותות פנימיים**:
   * result\_matrix מטריצה המייצגת את כל התוצאות האפשריות עבור כל הפעולות הלוגיות הנתמכות.
   * output\_matrix אות פנימי של המטריצה המכילה את כל תוצאות הפעולות הלוגיות.
2. **חישוב תוצאות**:
   * לכל ערך אפשרי של ALUFN (מ-000 עד 111) נקבעת הפעולה הלוגית המתאימה:
     + output\_matrix(0) מקבל את תוצאת NOT על y\_logic
     + output\_matrix(1) מקבל את תוצאת OR בין x\_logic ל- y\_logic
     + output\_matrix(2) מקבל את תוצאת AND בין x\_logic ל- y\_logic
     + output\_matrix(3) מקבל את תוצאת XOR בין x\_logic ל- y\_logic
     + output\_matrix(4) מקבל את תוצאת NOR בין x\_logic ל y\_logic
     + output\_matrix(5) מקבל את תוצאת NAND בין x\_logic ל y\_logic
     + output\_matrix(6) מקבל את תוצאת XNOR בין x\_logic ל y\_logic
     + output\_matrix(7) מקבל את וקטור האפסים (ברירת מחדל לפקודות לא חוקיות).
3. **הפקת תוצאה ויציאה**:
   * בהתאם לערך של ALUFN, המודול בוחר את התוצאה המתאימה מהמטריצה ומעביר אותה ליציאה Logic\_out
4. **תוצאות**

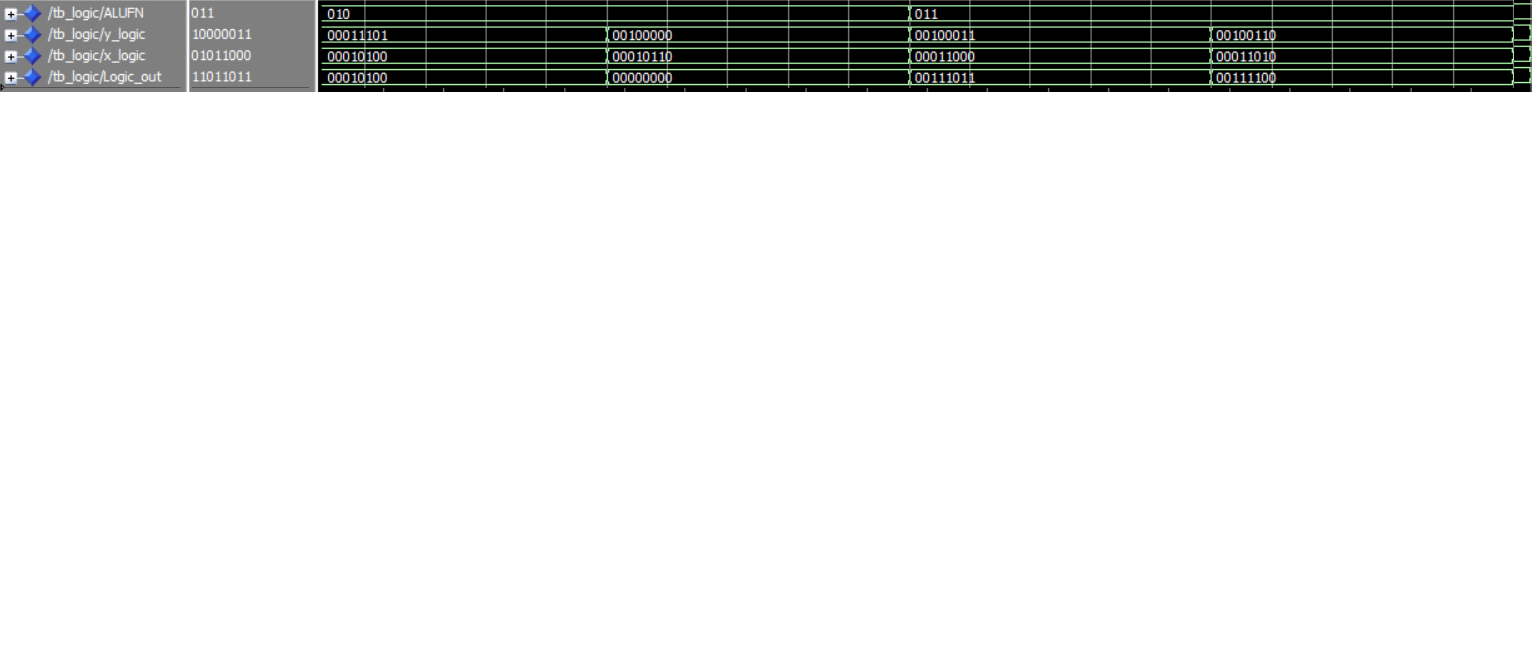


Figure 8 - פעולות לוגיות של and, xor

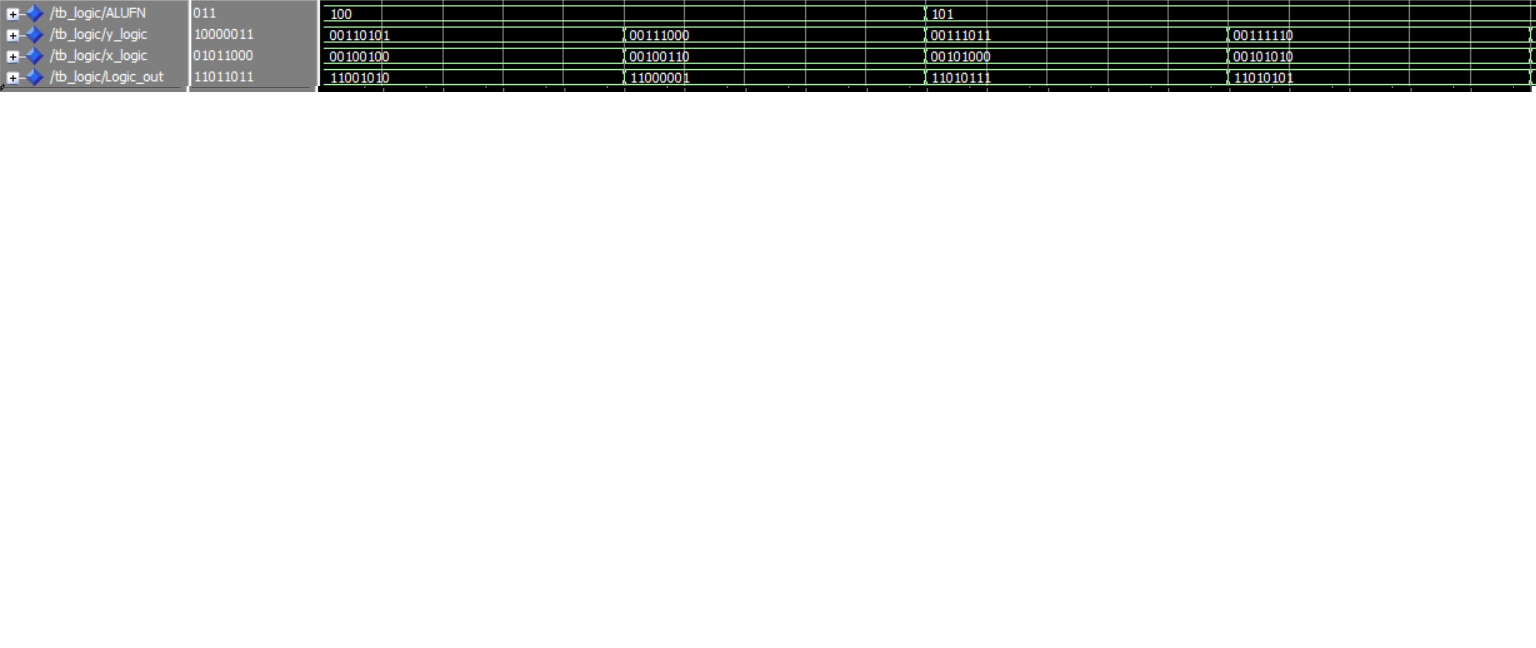


Figure 9 - פעולות לוגיות של nor, nand

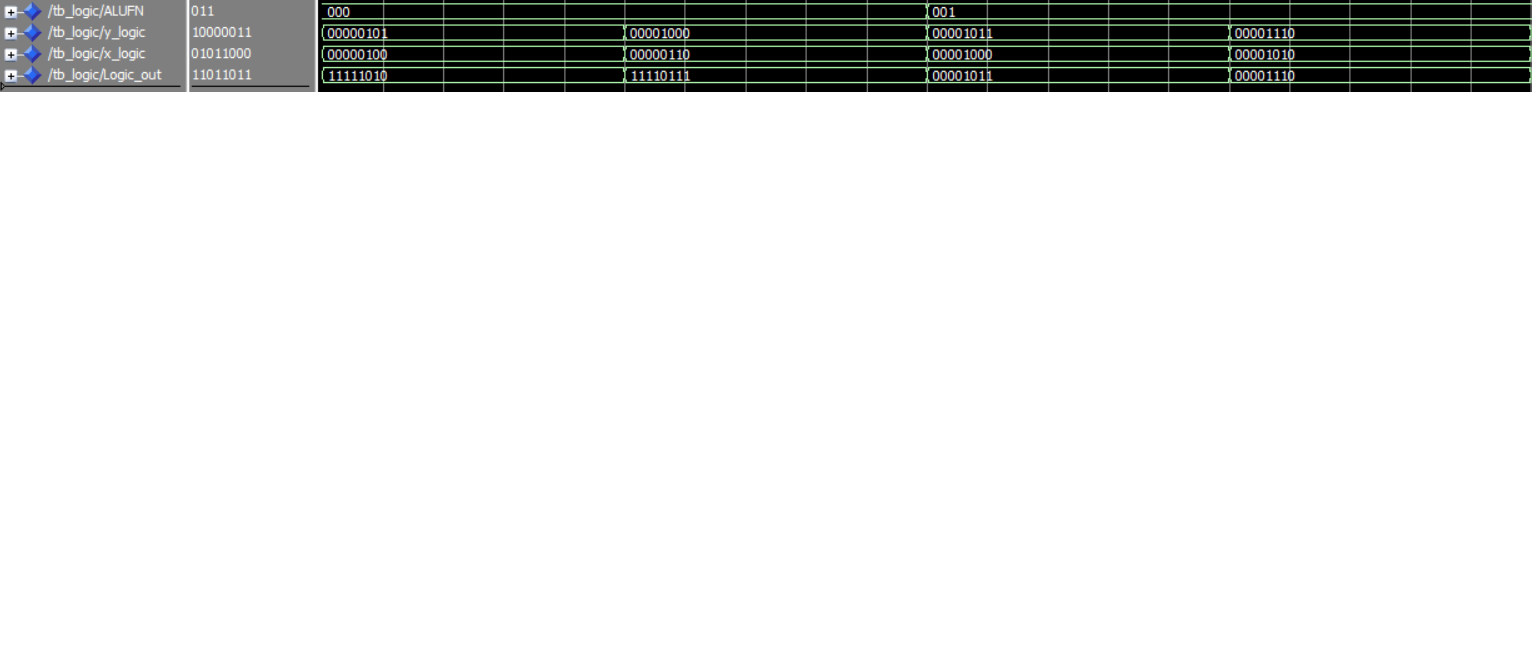


Figure 10 - פעולה לוגית not

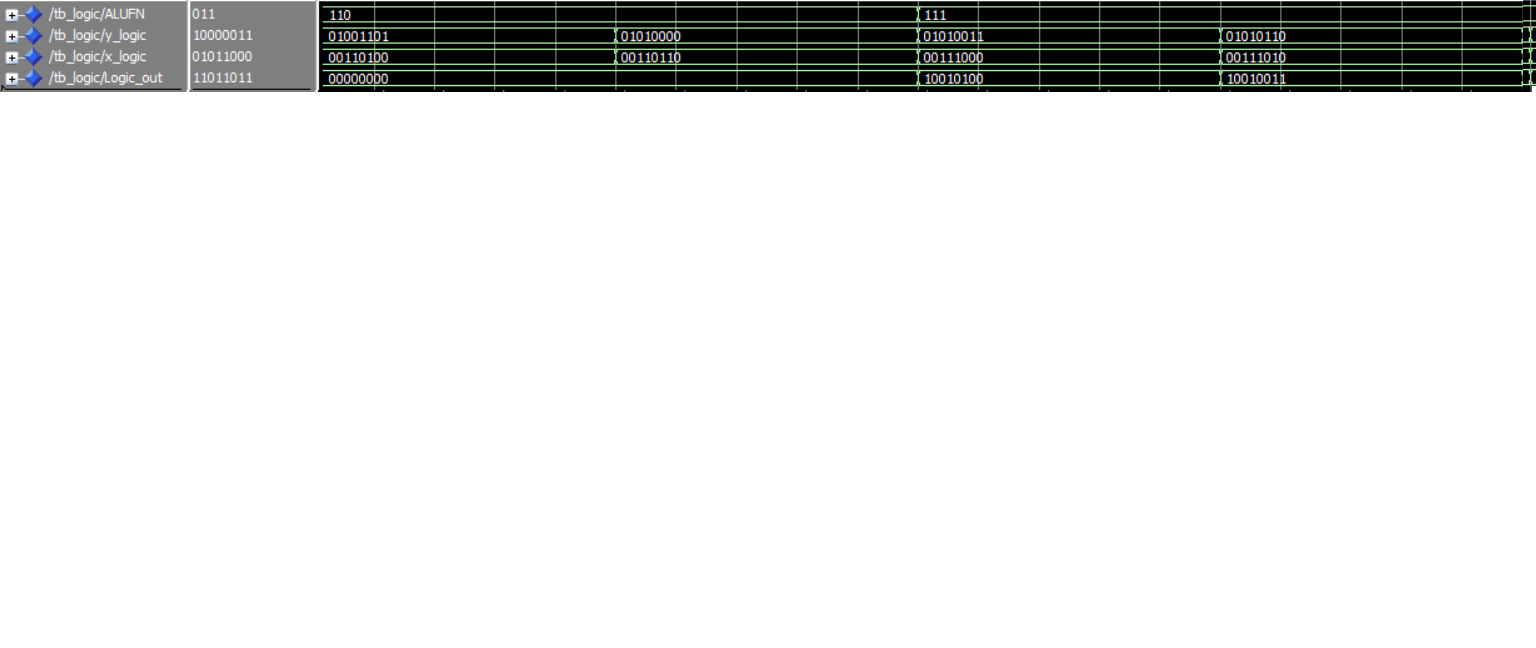


Figure 11 - אינפוטים אסורים עבור nor

## סיכום כללי

המבנה המשולב של המודולים מאפשר מימוש יחידה אריתמטית-לוגית (ALU) גמישה ורבת-יכולות, אשר מתאימה לשימוש במערכות דיגיטליות כמו מעבדים ומערכות. כל אחד מהמודולים מבצע פעולה ספציפית בצורה יעילה, והשילוב ביניהם מאפשר לממש פעולות מורכבות בהתאם לצרכים שונים.