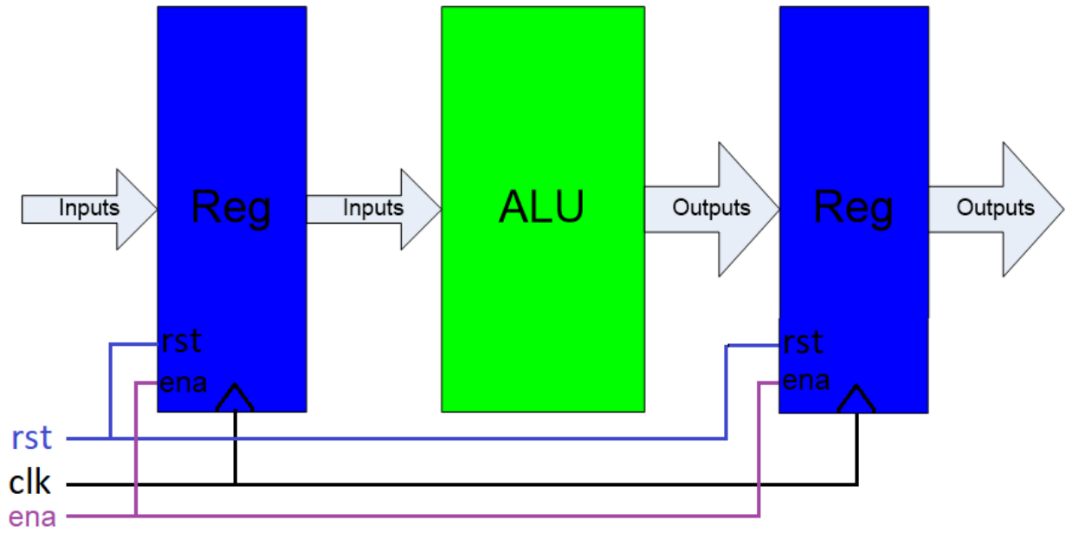
**top.vhd:**

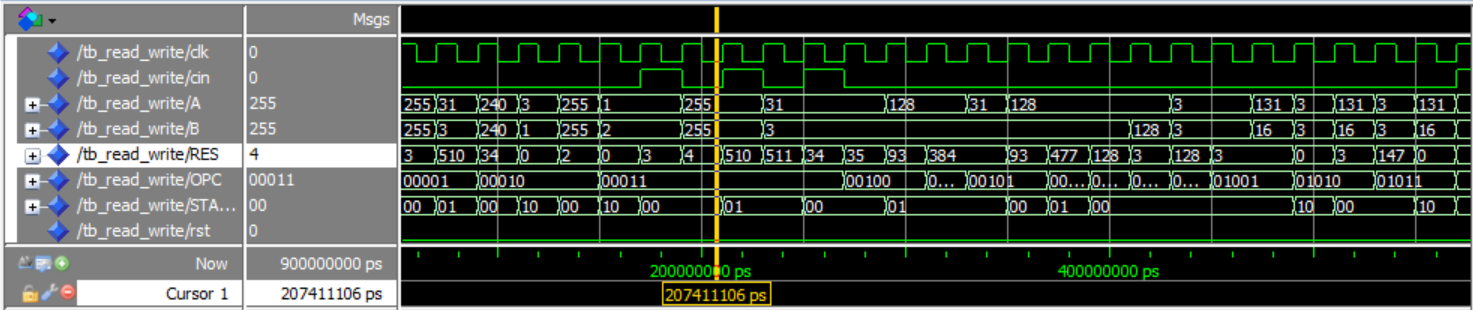
בקובץ הזה נמצא Entity בשם top בעל הכניסות והמוצאים הבאים:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ביט שתפקידו לאפס את המערכת | Input bit | rst |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| סיגנל CARRY שניתן לחישוב מתוך STATUS או להזין מבחוץ | input/feedback | c |
| סיגנל אשר מפעיל את פעולת המערכת | Input bit | ena |
| ווקטור הכניסה A למעבד. | Input vector (n bits) | A |
| ווקטור הכניסה B למעבד. | Input vector (n bits) | B |
| ווקטור כניסה למעבד שמכיל מידע לגבי הפעולה שנרצה לבצע. | Input vector (m bits) | OPC |
| ווקטור תוצאה שמתקבלת מהמעבד. | Output vector (2n bits) | RES |
| ווקטור סטטוס שמתקבל מהמעבד (מביא מידע על ה-carry או התוצאה). | Output vector (k bits) | STATUS |

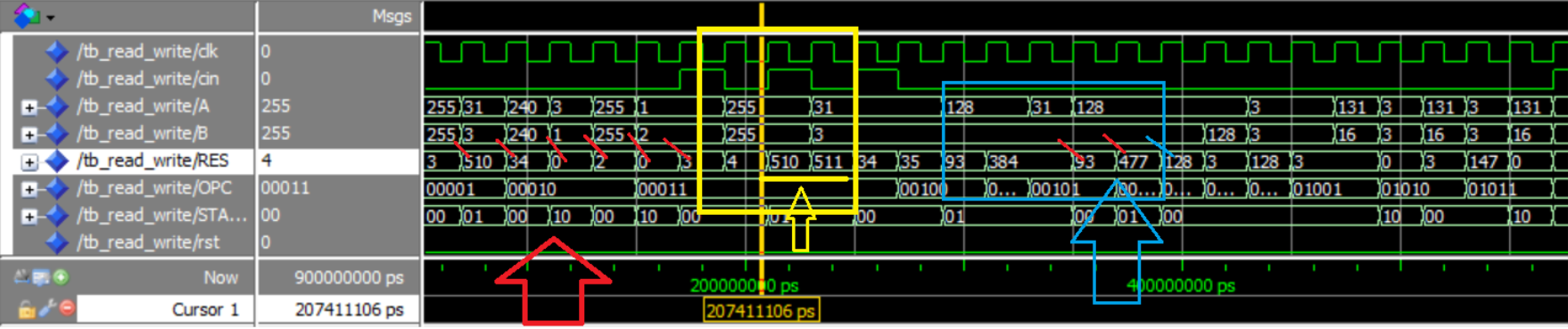
מודול זה בעצם עוטף את כלל המערכת והוא מכיל את המודולים הבאים: BACKregister, FRONTregister, ALU.

איור המודול:

**דיאגרמה נקייה:**



**דיאגרמה עם הסבר:**



מתוך הטסט שבודק את כל המקרים בחרנו כמה דברים להציג ולתת להם הסבר יותר מפורט. דבר ראשון תחת חץ אדום ניתן לראות פעולות אריתמטיות כמו חיבור חיסור. 255+255 יוצא 510 .240-240 יוצא 0. בתוך המערכת קיים דילאי של מחזור כפי שציפינו בשלב התכנון. במלבן צהוב ניתן לראות חיבור בהתחשבות של CIN,חשוב לציין שבמערכת קיים פידבק מתוך ALU לרגיסטר 1 על מנת לספר C המתאים, בנוסף ניתן להזין CIN מבחוץ. במלבן כחול ניתן לראות תוצאה של איוס הMAC שבזמן האיפוס הMAC פלט נשאר כמו במחזור לפני זה(384) ולאחר מכן ניתן לראות תוצאה של ACC. בצד בימני של הציור ניתן לראות פעולות של הזזה**.(בהמשך נפרט את הטסט עבור כל מודול)**

**דיאגרמת ה-LIST כדי שנוכל לראות את טבלת האמת של גרף ה-wave:**



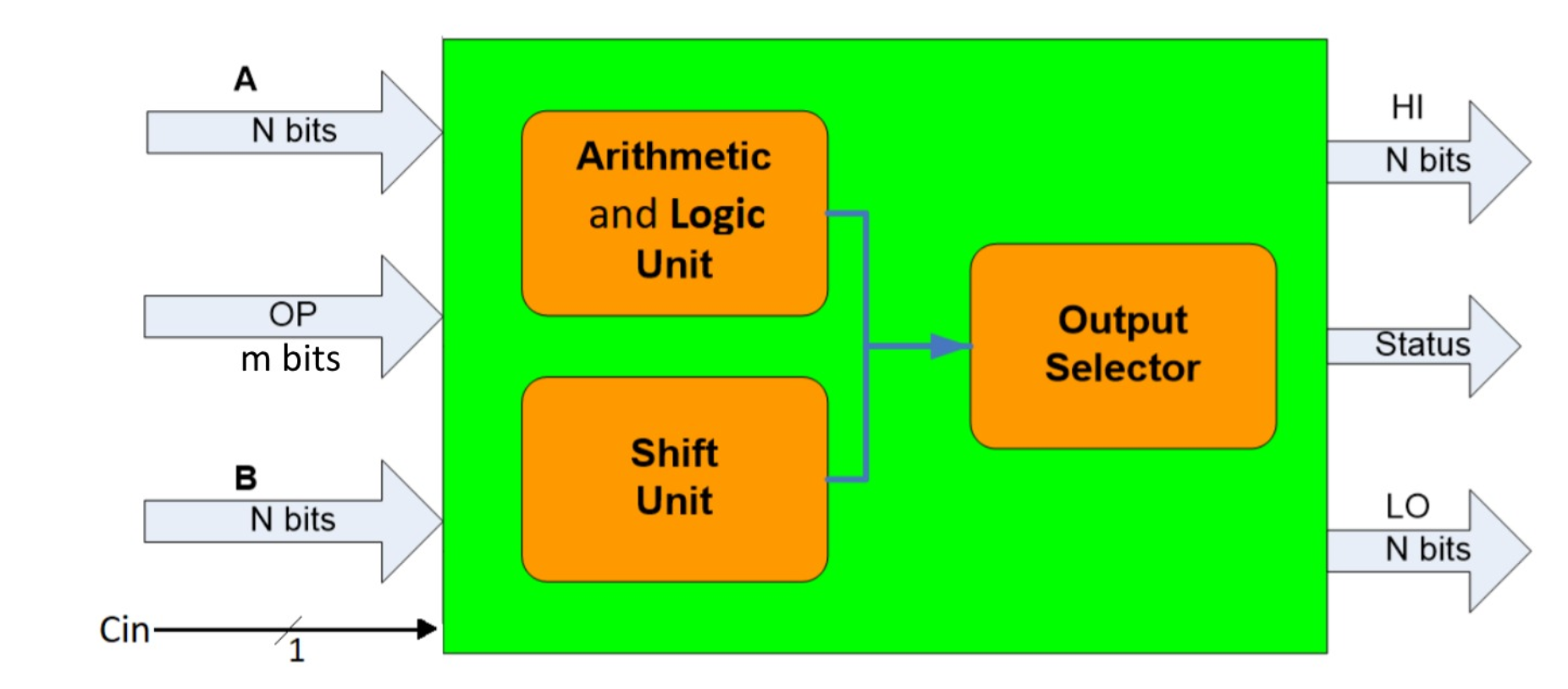
**ALU.vhd**

בקובץ הזה נמצא Entity בשם ALU בעל הכניסות והמוצאים הבאים:

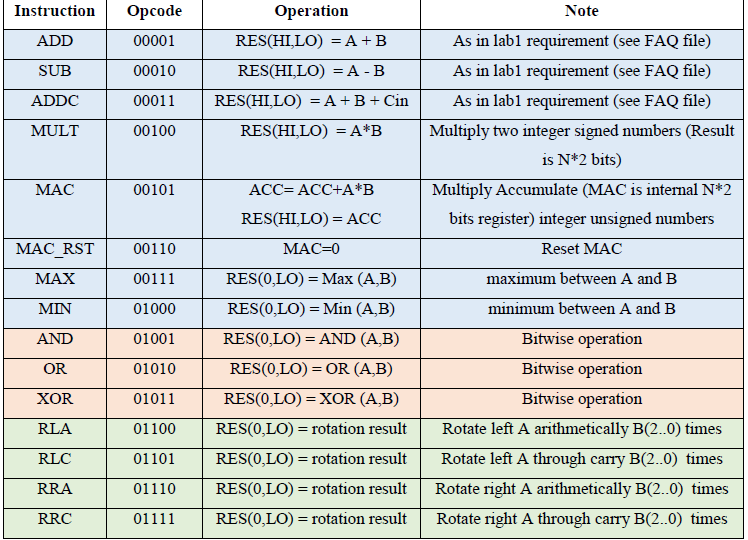
|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| ווקטור הכניסה A למעבד. | Input vector (n bits) | A |
| ווקטור הכניסה B למעבד. | Input vector (n bits) | B |
| ווקטור כניסה למעבד שמכיל מידע לגבי הפעולה שנרצה לבצע. | Input vector (m bits) | OPC |
| ווקטור תוצאה שמתקבלת מהמעבד. | Output vector (2n bits) | RES |
| ווקטור סטטוס שמתקבל מהמעבד (מביא מידע על ה-carry או התוצאה). | Output vector (k bits) | STATUS |

בתוך המודול הזה נמצאים INSTANCES הבאים: arithAndLogicEntity, selectorEntity, shiftEntity.

המודול הנוכחי הוא מקבל 2 וקטורי כניסה, CIN וOPCODE של הפקודה ומבצע חישוב המתאים. בסוף הוא מוציא וקטור פלט וסטטוס.במודל הזה קיימת כניסה של השעון עבור חישוב של MAC.



ALU הנוכחי מתחזק פעולות הבאות:



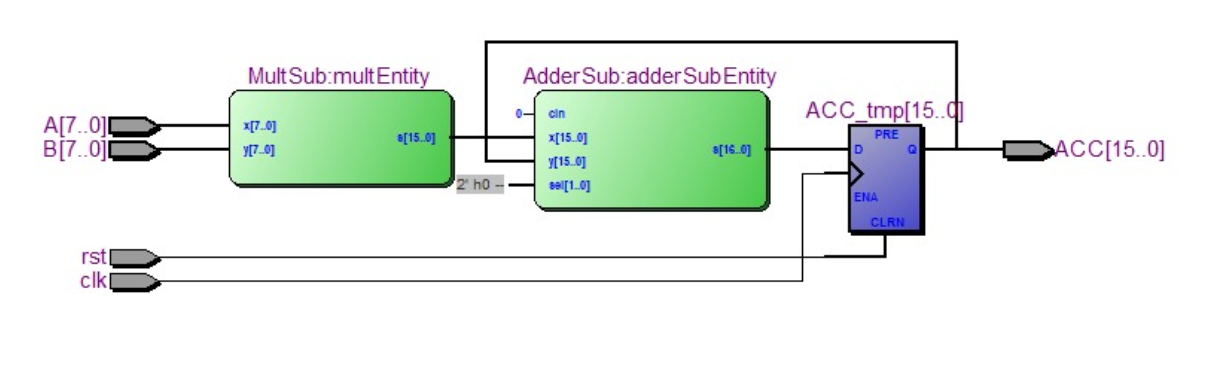
**MAC.vhd**

בקובץ הזה נמצא Entity בשם MAC בעל הכניסות והמוצאים הבאים:

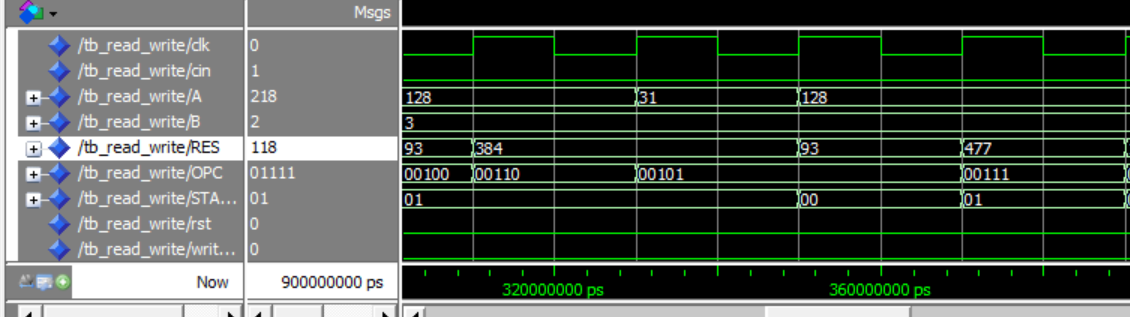
|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| סיגנל שמגיע מתוך רשימת פקודות של ALU שמאפס אתACC | Input bit | rst |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| ווקטור הכניסה A למעבד. | Input vector (n bits) | A |
| ווקטור הכניסה B למעבד. | Input vector (n bits) | B |
| ווקטור תוצאה שמתקבלת מהמעבד. | Output vector (2n bits) | ACC |

מודול ששומר את תוצאה של ACC ועבור פקודות מתאימות הוא מוציא את התוצאה או שמאפס שת ACC.מודול הזה הוא מודל היחידי בתוך CPU שדורש כניסה של CLK. מודל הנוכחי עבר סינטזה.

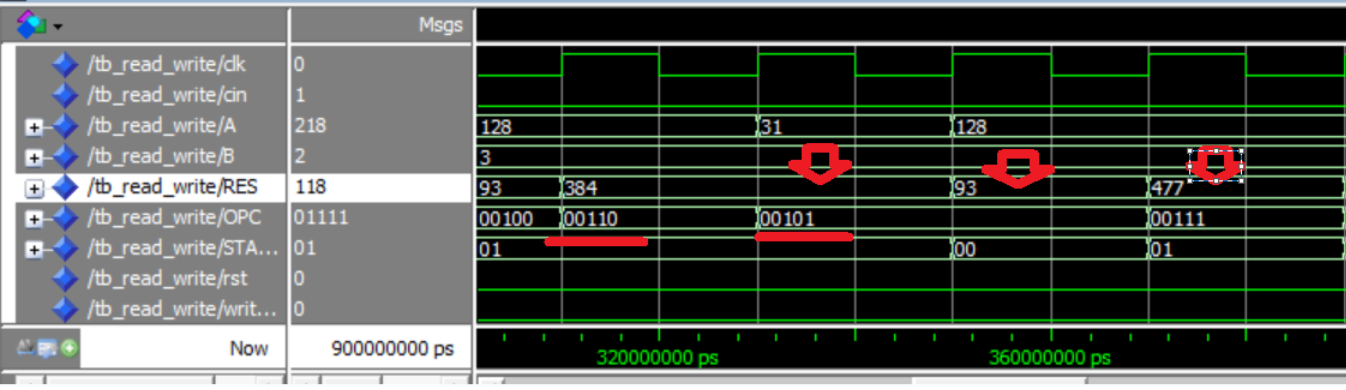
**ציור של MAC מסונטז:**



**דיאגרמה נקייה:**



**דיאגרמה עם ההסבר:**



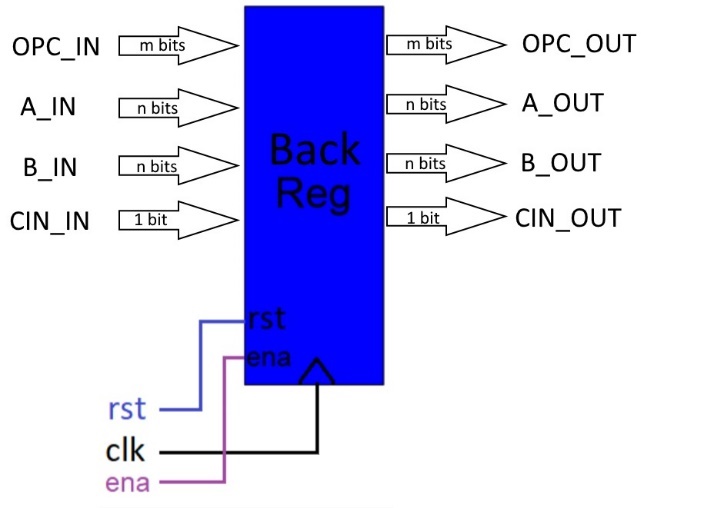
ניתן לראות פעולת MAC בדיאגרמה שלמעלה. בהתחלה מופעלת פקודה של איפוס הACC ולכן במחזור שעון הזה ערך שהיה מקודם נשמר(384) לאחר מכן מופעל פעמיים פעולה של לקבל ערך של ACC. במחזור הראשון זה

0+3\*31=93 במחזור של פקודה הבא יוצא 31+128\*3=477.את הערכים האלב ניתן לראות בגרף, קיבלנו בדיוק את מה שציפינו לקבל.

**backRegister.vhd:**

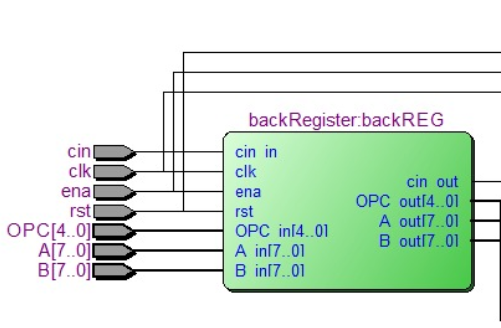
מודול זה אחראי על פליטת הקלט למעבד בסנכרון לאות השעון, אות האיפוס ואות האיפשור (enable).  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ביט שתפקידו לאפס את המערכת | Input bit | rst |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| סיגנל אשר מפעיל את פעולת המערכת | Input bit | ena |
| ווקטור הכניסה A למעבד. | Input vector (n bits) | A\_in |
| ווקטור הכניסה B למעבד. | Input vector (n bits) | B\_in |
| ווקטור כניסה למעבד שמכיל מידע לגבי הפעולה שנרצה לבצע. | Input vector (m bits) | OPC\_in |
| ביט כניסה למעבד עם ה-carry שאותו רוצים להכניס.\פידבק מתוך הסטטוס | Input bit | cin\_in/cin\_total |
| ווקטור הכניסה A למעבד. (מסונכרן לפי שעון) | Output vector (n bits) | A\_out |
| ווקטור הכניסה B למעבד. (מסונכרן לפי שעון) | Output vector (n bits) | B\_out |
| ווקטור כניסה למעבד שמכיל מידע לגבי הפעולה שנרצה לבצע. (מסונכרן לפי שעון) | Output vector (m bits) | OPC\_out |
| ביט כניסה למעבד עם ה-carry שאותו רוצים להכניס. (מסונכרן לפי שעון) | Output bit | cin\_out |



**איור המודול:**

**דיאגרמה לאחר סינטזה של הרגיסטר**:



חשוב לציין שמודול הזה כפי שתכננו מטפל בפידבק שמגיע מתוך הסטטוס ובנוסף לכך הוא מחלטים האם לקחת CIN של ההזנה חיצונית או של CIN מתוך הסטטוס. בחלטנו לתת להזנה מבחוץ ההעדפה על פני CIN שמגע מתוך המערכת. זאת אומרת שCIN שהגיע מתוך המערכת יכול להיות 1 אך אם באותו רגע בטסט מוזן CIN שהוא 0 אזי תיבחר 0.

**דוגמא**:

קלט:

00001 00000001 11111111 x

00011 00000001 11111111 0

פלט:

00000001 00000000 11

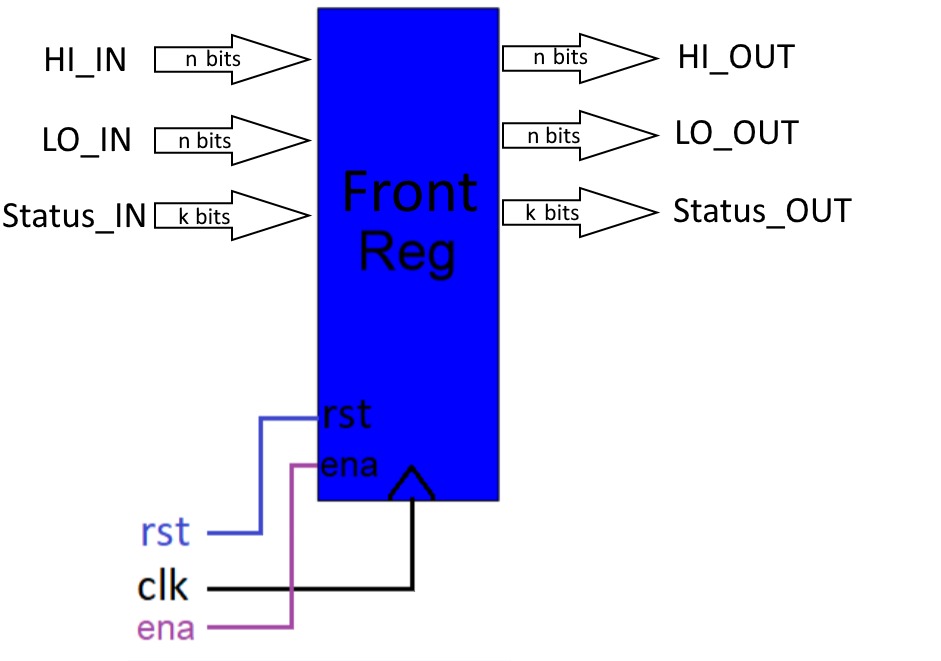
00000001 00000000 11

למרות שC בתוך הסטטוס שווה ל1 אנו התעלמנו ממנו מכיוון שהגיע מתוך הטסט CIN שהוא 0.

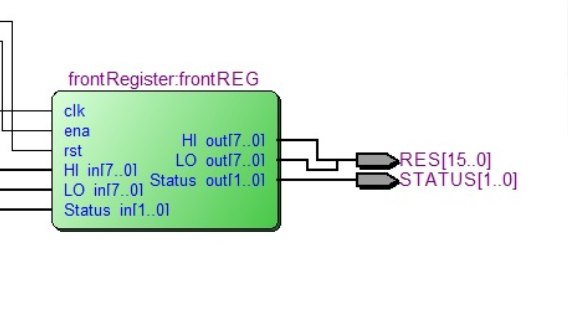
**frontRegister.vhd:**

מודול זה אחראי על פליטת הקלט מהמעבד החוצה בסנכרון לאות השעון, אות האיפוס ואות האיפשור (enable).  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| ווקטור היציאה מהמעבד (n סיביות MSB של התוצאה). | Input vector (n bits) | HI\_IN |
| ווקטור היציאה מהמעבד (n סיביות LSB של התוצאה). | Input vector (n bits) | LO\_IN |
| ווקטור סטטוס שמתקבל מהמעבד (מביא מידע על ה-carry או התוצאה). | Input vector (k bits) | STATUS\_IN |
| ווקטור היציאה מהמעבד (n סיביות MSB של התוצאה). (מסונכרן לפי שעון) | Output vector (n bits) | HI\_OUT |
| ווקטור היציאה מהמעבד (n סיביות LSB של התוצאה). (מסונכרן לפי שעון) | Output vector (n bits) | LO\_OUT |
| ווקטור סטטוס שמתקבל מהמעבד (מביא מידע על ה-carry או התוצאה). (מסונכרן לפי שעון) | Output vector (k bits) | STATUS\_OUT |

  
**איור המודול:**

**דיאגרמה של המודול לאחר סיטזה:**



מודול הנוכחי דואג להוציא את התוצאות של ALU החוצה.בנוסף מבצע חישוב של סטטוס כפי שהוגדר בתרגיל.

**דוגמא**:

קלט:

00001 00000001 11111111 x

00011 00000001 11111111 0

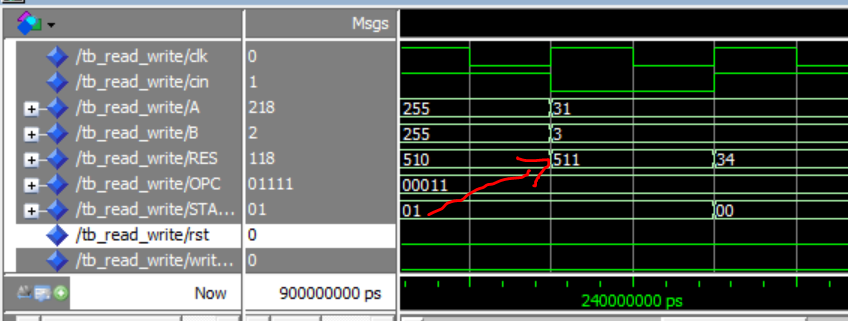
פלט:

00000001 00000000 11

00000001 00000000 11

ניתן לראות שכאשר LO הוא אפסים וגם יש CARRY אזי סטטוס 11,בדיוק כפי שציפינו.

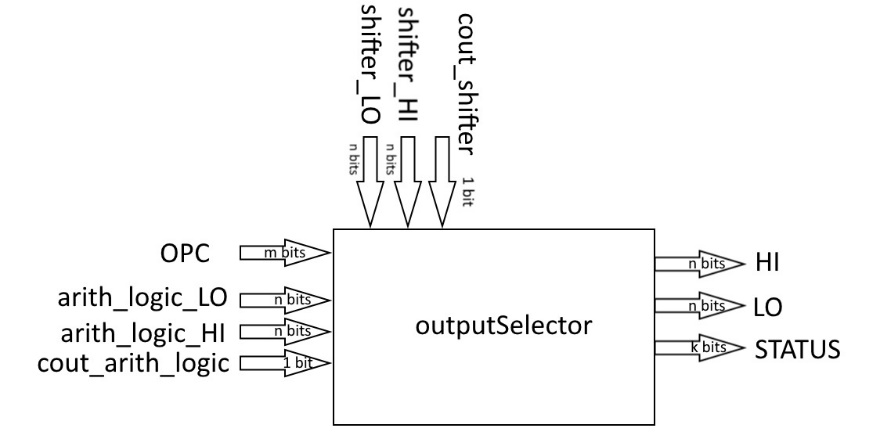
דוגמא נוספת שעבור חישוב סטטוס 255 + 255 =510 וזה גורם לCARRY לעלות ל1, לכן סטטוס הופך 01 ולאחר מכן משתמשים בזה בחישוב הבא, בדיוק כמו שרצינו שיהיה.



**outputSelector.vhd:**

מודול זה בעצם בוחר איזה פלט המעבד יוציא לפי קוד ה-OPC שהיה בכניסה. בנוסף, הוא גם מקבל את ה-Carry של שני המודולים ומוציא בהתאם גם את ווקטור ה-Status המתאים בהתאם למצב הנתון.  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ווקטור כניסה למעבד שמכיל מידע לגבי הפעולה שנרצה לבצע. | Input vector (m bits) | OPC |
| ווקטור היציאה מיחידת ה-shifter (n סיביות MSB של התוצאה). | Input vector (n bits) | Shifter\_HI |
| ווקטור היציאה מיחידת ה-shifter (n סיביות LSB של התוצאה). | Input vector (n bits) | Shifter\_LO |
| ביט מוצא מיחידת ה-shifter שמכיל את הנשא של התוצאה. | Input bit | Cout\_shifter |
| ווקטור היציאה מיחידת ה-ArithLogic (n סיביות MSB של התוצאה). | Input vector (n bits) | Arith\_logic\_HI |
| ווקטור היציאה מיחידת ה- ArithLogic(n סיביות LSB של התוצאה). | Input vector (n bits) | Arith\_logic \_LO |
| ביט מוצא מיחידת ה- ArithLogicשמכיל את הנשא של התוצאה. | Input bit | Cout\_shifter |
| ווקטור היציאה המתאים (n סיביות MSB של התוצאה). | Output vector (n bits) | HI |
| ווקטור היציאה המתאים (n סיביות LSB של התוצאה). | Output vector (n bits) | LO |
| ווקטור סטטוס שנקבע ע"י התוצאה המתאימה (מביא מידע על ה-carry או התוצאה). | Output vector (k bits) | STATUS |

  
איור המודול:

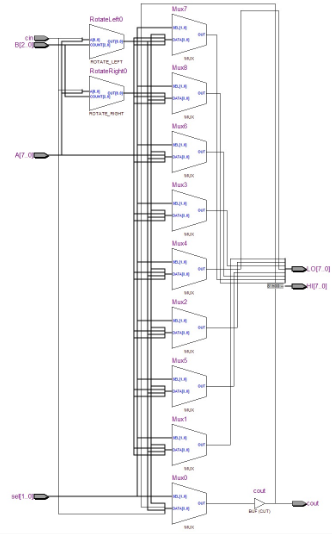
**shifter.vhd:**

מודול זה בעצם אחראי על כל פעולות ההזזה של המעבד. הוא בוחר איזה סוג הזזה לבצע לפי ה-sel שהיה בכניסה והוא עושה את ההזזה הנבחרת על ווקטור A לפי המספר שמכיל B. בנוסף, הוא גם מקבל Carry של שני המודולים ומוציא בהתאם גם את ביט ה-carry החדש שנוצר.  
עבור sel = “00” – ביצוע RLA.  
עבור sel = “01” – ביצוע RLC.  
עבור sel = “10” – ביצוע RRA.  
עבור sel = “11” – ביצוע RRC.  
פירוט הכניסות והמוצאים של המודול:

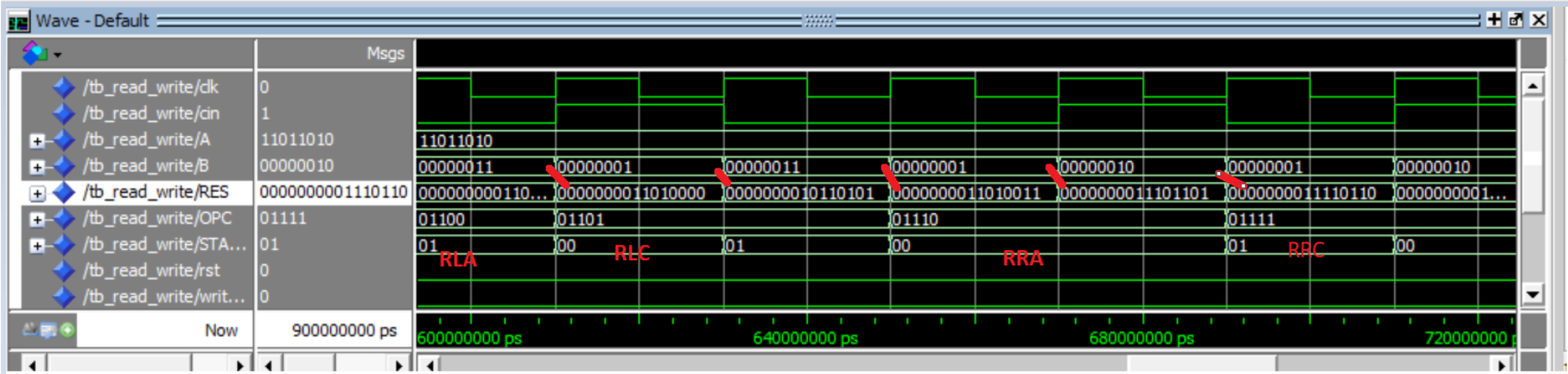
|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ווקטור הכניסה A למודול (עליו מבצעים את ההזזה). | Input vector (n bits) | A |
| ווקטור הכניסה B למודול (מכיל מידע על כמה פעמים להזיז את A). | Input vector (3 bits) | B |
| ווקטור כניסה למודול שמורה על איזה סוג הזזה לבצע. | Input vector (2 bits) | sel |
| ביט כניסה למודול עם ה-carry שאותו רוצים להכניס. | Input bit | cin |
| ווקטור היציאה מהמודול (n סיביות MSB של התוצאה). | Output vector (n bits) | HI |
| ווקטור היציאה מהמודול (n סיביות LSB של התוצאה). | Output vector (n bits) | LO |
| ביט יציאה מהמודול עם ה-carry שנוצר. | Output bit | cout |

  
איור המודול:

דיאגרמה של המודול לאחר סינטזה:



**דיאגרמה עם הסבר:**

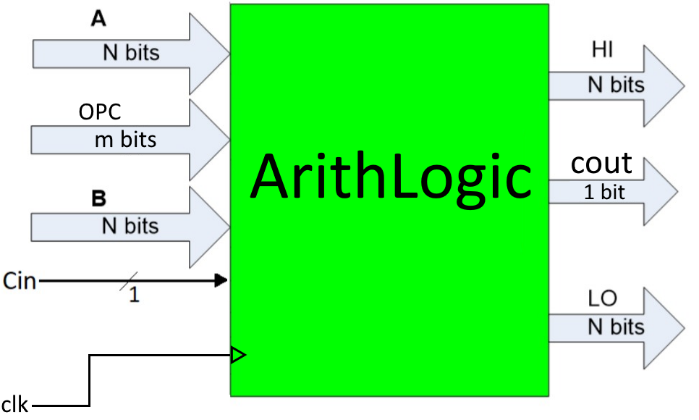


בדיאגרמה ניתן לראות פעולה תקינה של כל אחד מ4 סוגי הSHIFT. עבור כל אחד מ4 מקרים ניתן לראות את ההתחשבות עבור CARRY ועבור פעולה קודמת. מודול עובד כפי שתכננו אותו וכפי שהוגדר בתרגיל.

**ArithLogic.vhd:**

מודול זה בעצם אחראי על כל הפעולות הלוגיות והאריתמטיות של המעבד. הוא בוחר איזה סוג פעולה לבצע לפי קוד ה-OPC שהיה בכניסה, והוא מכיל את המודולים הבאים: MaxMin, MACModule, AdderSub, MultSub.  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| סיגנל מחזורי שמסנכרן את כל המערכת | Input bit | clk |
| ווקטור הכניסה A למודול. | Input vector (n bits) | A |
| ווקטור הכניסה B למודול. | Input vector (n bits) | B |
| ווקטור כניסה למודול שמורה על איזה סוג פעולה לבצע. | Input vector (m bits) | OPC |
| ביט כניסה למודול עם ה-carry שאותו רוצים להכניס. | Input bit | cin |
| ווקטור היציאה מהמודול (n סיביות MSB של התוצאה). | Output vector (n bits) | HI |
| ווקטור היציאה מהמודול (n סיביות LSB של התוצאה). | Output vector (n bits) | LO |
| ביט יציאה מהמודול עם ה-carry שנוצר. | Output bit | cout |

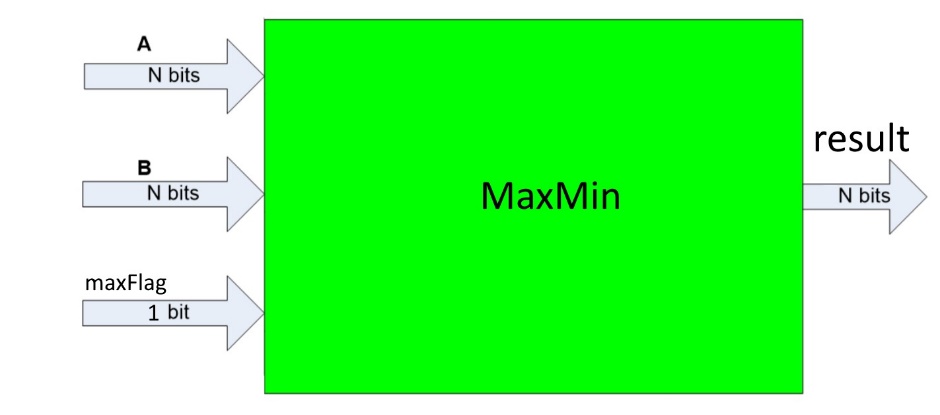
  
איור המודול:

פעולות תקינה של המודול ניתן לראות בפעולות תקינה של התת מודולים שמצויינים בדוח הנוכחי.

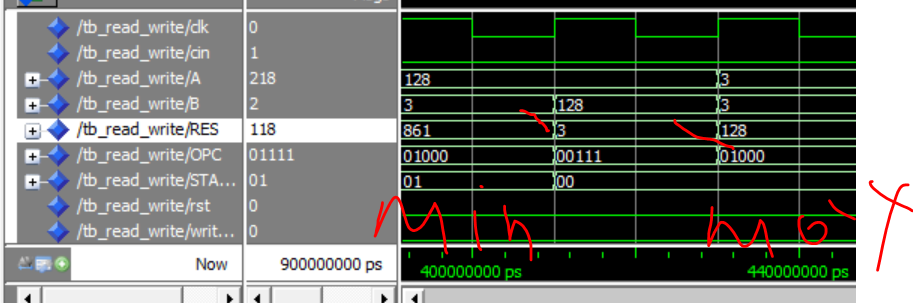
**MaxMin.vhd:**

מודול זה בעצם אחראי על השוואה בין הכניסות A ו-B ולהוציא את המינימום/מקסימום שלהם בהתאם לסיגנל maxFlag (אם הוא '1' אז להוציא את המקסימום, אם הוא '0' אז להוציא את המינימום).  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ווקטור הכניסה A למודול. | Input vector (n bits) | A |
| ווקטור הכניסה B למודול. | Input vector (n bits) | B |
| ביט כניסה למודול שמורה על איזה מהווקטורים להוציא כפלט. | Input bit | maxFlag |
| ווקטור היציאה מהמודול. | Output vector (n bits) | result |

  
איור המודול:

**דיאגרמה עם הסבר:**

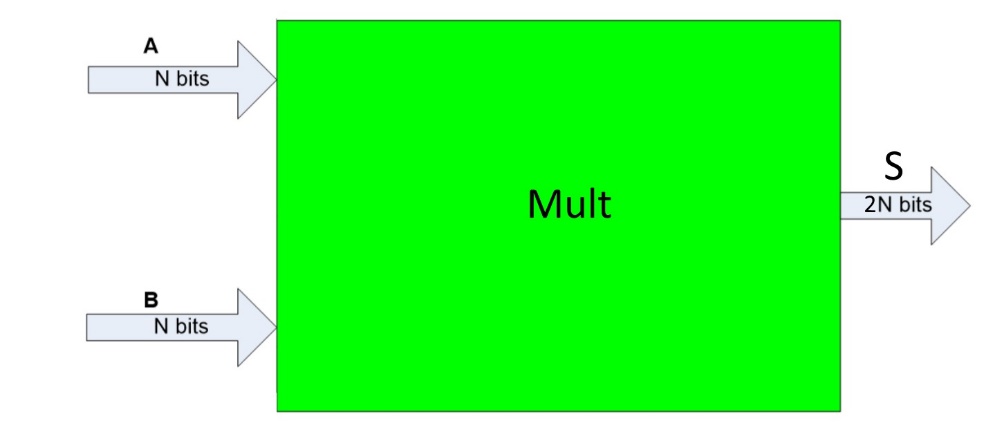


בדוגמא ניתן לראות פעולות MIN MAX עבור זוג מספרים 128 ו3. עבור MIN פלט 3 עבור MAX פלט 128.

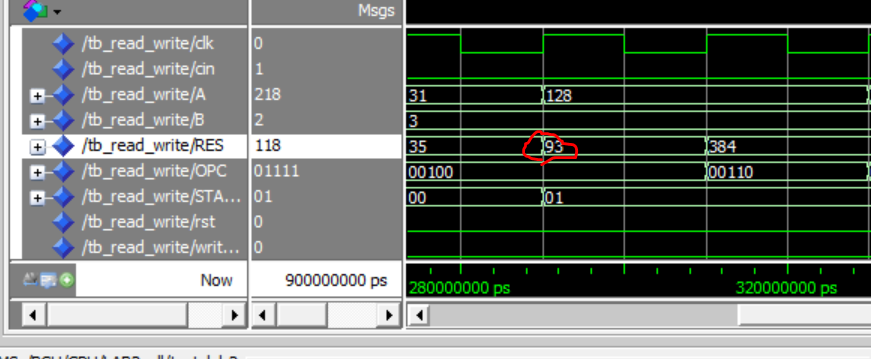
**Mult.vhd:**

מודול זה בעצם אחראי על פעולת הכפל בין שתי הכניסות x ו-y.  
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ווקטור הכניסה x למודול. | Input vector (n bits) | x |
| ווקטור הכניסה y למודול. | Input vector (n bits) | y |
| ווקטור היציאה מהמודול (2n סיביות, תוצאת הכפל בין x ל-y). | Output vector (2n bits) | s |

  
איור המודול:

**דיאגרמה עם הסבר:**

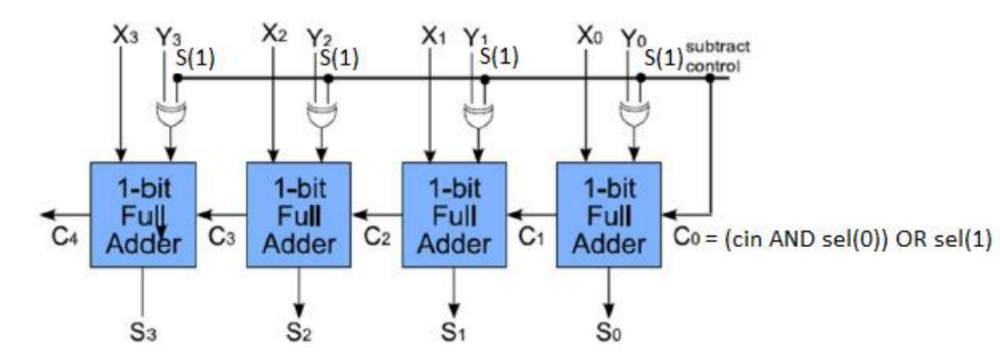


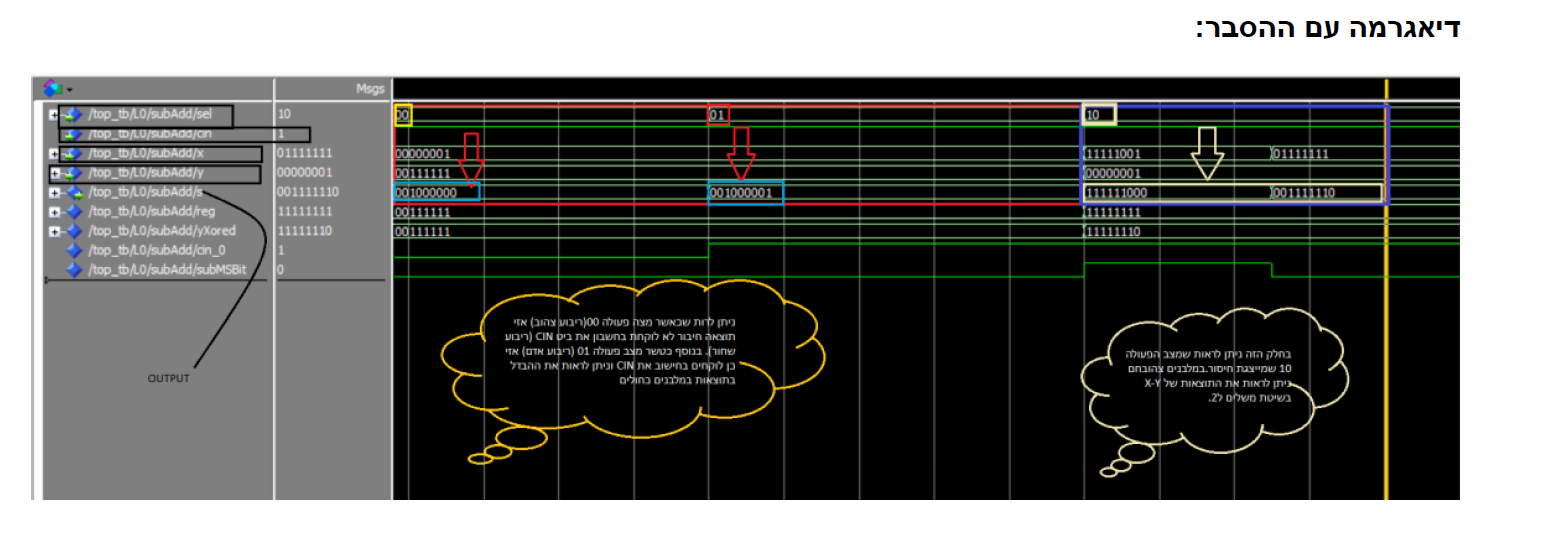
בדוגמא ניתן לראות תוצאה של פעולת כפל 31 \*3 =93. בדיוק המספר שאנו רואים בדיאגרמה.

**AdderSub.vhd:**

מודול זה בעצם אחראי על פעולות חיבור וחיסור של המעבד. הוא בוחר איזה סוג פעולה לבצע לפי קוד ה-sel שהיה בכניסה, והוא מכיל מודולים של FA.   
פירוט הכניסות והמוצאים של המודול:

|  |  |  |
| --- | --- | --- |
| הסבר | סוג | שם הפורט |
| ווקטור הכניסה x למודול. | Input vector (n bits) | x |
| ווקטור הכניסה y למודול. | Input vector (n bits) | y |
| ווקטור כניסה למודול שמורה על איזה סוג פעולה לבצע. | Input vector (2 bits) | sel |
| ביט כניסה למודול עם ה-carry שאותו רוצים להכניס. | Input bit | cin |
| ווקטור היציאה מהמודול (n סיביות MSB של התוצאה). | Output vector (n+1 bits) | s |

  
איור המודול (דוגמה עבור 4 ביטים):  
במודול שלנו זה בעצם .



המודול נלקח מתרגיל הראשון.

**סיכום:**

במהלך המשימה הצלחנו לקבל קישורים בכתיבת קוד VHDL בצורה יותק מתקדמת ולקבל ההבנה של שילוב מערכות שונות בפרויקט אחד. בנוסף למדנו איך לממש טסטים מתקדמים בעזרת קריאה וכתיבה לתוך הקובץ. הבנו חשיבות של בדיקות והצלחנו לממש את מערכת שעומדת בדרישות המשימה. אנחנו מבינים שקוד שכתבנו הוא אינו אופטימלי ובמעבדות הבאות נשים לב גם למסלול הקריטי וננסה לשפר אותו.