A screenshot of a video game

Description automatically generated**Top block:**

Figure 1: RTL view of top.vhd

כפי שניתן לראות ב-figure 1, הכניסות של המתגים והכפתורים מחוברים ישירות לבלוקים המתאימים. ה-ALU עצמו מקבל מידע מהבלוקים ולפי המידע הנתון על A,B,OPC הוא מבצע את הפעולה המתאימה. החיבור היחיד ל-ALU הוא הכניסה של key\_3 והוא מתחבר ל-clkMAC – כלומר הוא מתפקד בתור אות השעון שמפעיל את ה-MAC.  
היציאות של המערכת הם בעצם HI,LO,STATUS, והם יוצאים ישירות החוצה גם ללדים וגם עוברים דרך מודול שממיר אותם לתצוגה של SEVEN SEGMENT.

A picture containing clock, meter

Description automatically generated**OP\_REGISTER:**

Figure 2: RTL view of registerOP.vhd

כפי שניתן לראות ב-figure 2, רגיסטר OP הוא בעצם DFF פשוט שדואג להעביר את קוד הפעולה בצורה סינכרונית (בהתאם למתי שהמשתמש לוחץ על הכפתור המתאים).

A screenshot of a cell phone

Description automatically generated**B\_REGISTER:**

Figure 3: RTL view of registerB.vhd

A screenshot of a cell phone

Description automatically generatedכפי שניתן לראות ב-figure 3, רגיסטר B הוא בעצם DFF פשוט שדואג להעביר את כניסה B בצורה סינכרונית (בהתאם למתי שהמשתמש לוחץ על הכפתור המתאים).

**A\_REGISTER:**

Figure 4: RTL view of registerA.vhd

כפי שניתן לראות ב-figure 4, רגיסטר A הוא בעצם DFF פשוט שדואג להעביר את כניסה A בצורה סינכרונית (בהתאם למתי שהמשתמש לוחץ על הכפתור המתאים).

**sevenSegment block:**

A close up of a map

Description automatically generated

Figure 5: RTL view of sevenSegment.vhd

כפי שניתן לראות ב-figure 5, ה- sevenSegmentמורכב מאוסף של muxים שממירים את הנתון שמוצג כווקטור בינארי לנתון שיפעיל ספרה/אות של seven segment.

A screenshot of a cell phone

Description automatically generated**ALU block:**

Figure 6: RTL view of ALU.vhd

כפי שניתן לראות ב-figure 6, ה-ALU מורכב מ-3 תתי מודולים שהוא מאחד והם בעצם אחראים על כל תפקודו, נפרט עליהם בהמשך.

A screenshot of a video game

Description automatically generated**shifter block:**

Figure 7: RTL view of shifter.vhd

בלוק זה למעשה אחראי על 4 פקודות ההזזה השונות (RRA,RRC,RLA,RLC).  
כפי שניתן לראות ב-figure 7, ה-shifter למעשה מורכב מאוסף של MUXים ובעזרתם הוא מבצע את כל ההזזות. בנוסף, ניתן לראות שיש כמה מודולי השוואה (הם למעשה בוררים איזו פעולה לעשות מבין ה-4).

A screenshot of a video game

Description automatically generated**outputSelector block:**

Figure 8: RTL view of outputSelector.vhd

כפי שניתן לראות ב-figure 8, ה-outputSelector פשוט אחראי על השוואה של OPC והוצאה של תוצאה מתאימה בהתאם לפקודה שהתקבלה בכניסה.

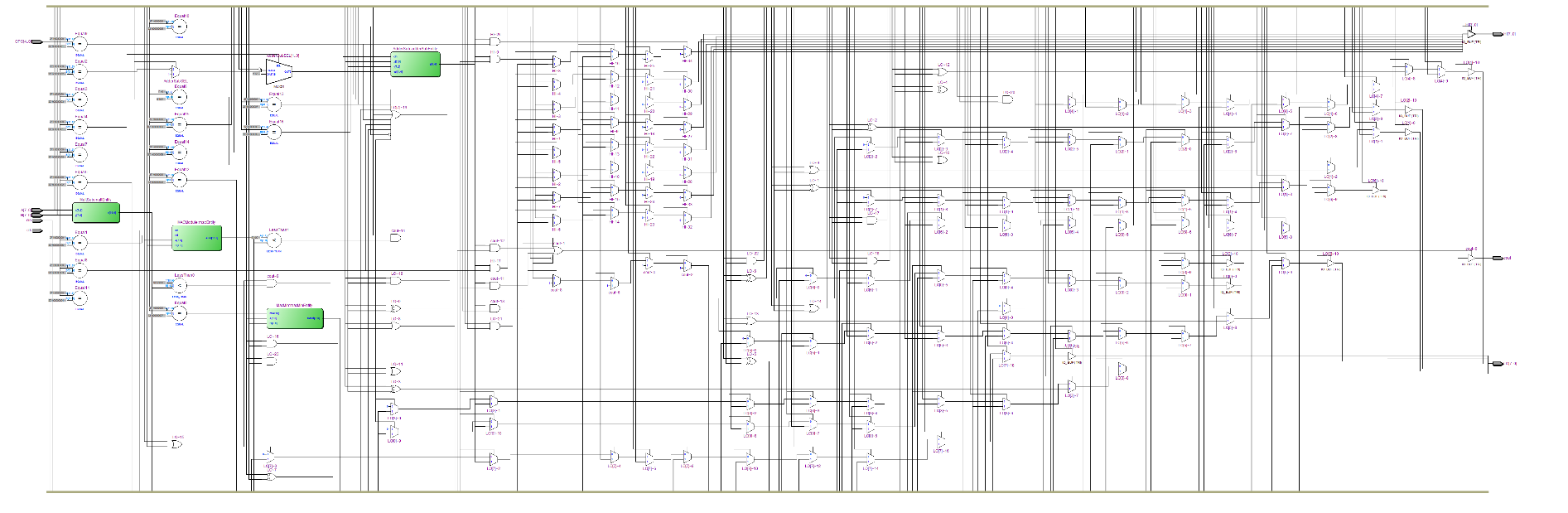
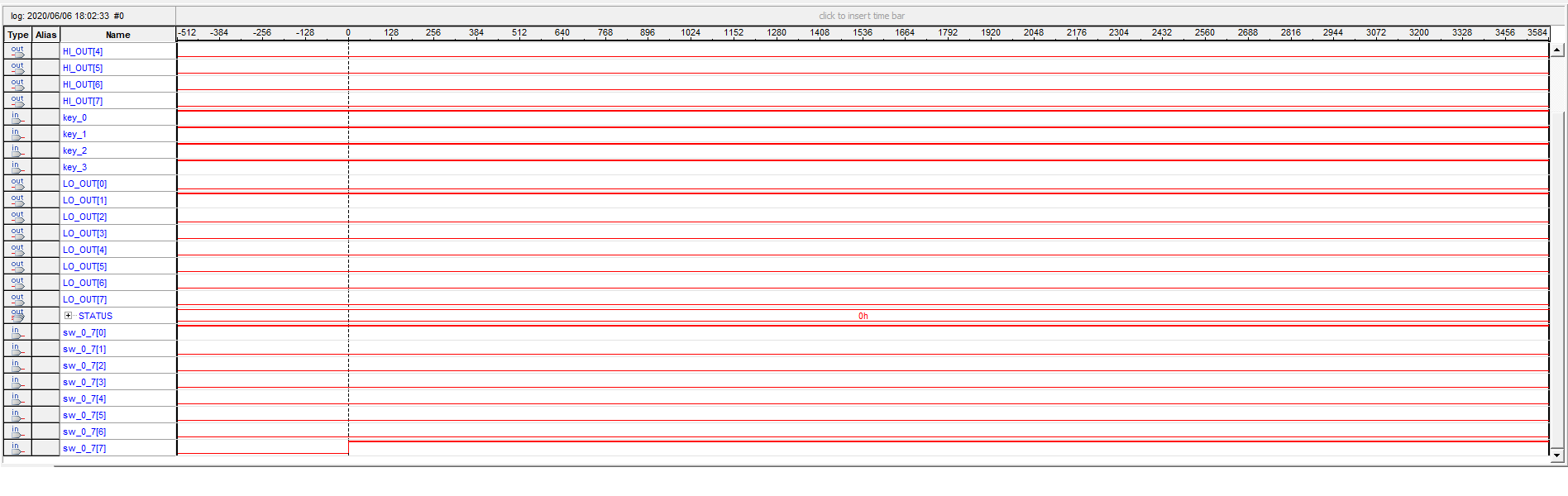
**ArithLogic block:**

Figure 9: RTL view of ArithLogic.vhd

כפי שניתן לראות ב-figure 9, ה- ArithLogicמכיל הרבה מודולים שמבצעים לוגיקה – וזאת בהתאם לבלוק – זה הבלוק שמבצע את כל הפעולות הלוגיות שיש במערכת. בנוסף, הוא מכיל בתוכו מודול של מכפל, מחבר מחסר, MAC, ומודול שבודק מינימום/מקסימום.  
המודול מקבל את הכניסות ואת קוד הפעולה ומבצע את הפעולה ומוציא תוצאה בהתאם לקוד הפעולה שהיה בכניסה.

**תרשים מ-signal tap:**

לאחר שהגדררנו את הכל כמו שצריך, התקבל המסך הבא ב-signal tap:

מה שעשינו זה פשוט תכנית שעושה 1+1 וזהו, ואכן ניתן לראות ש-LO[1] עם "1" לוגי ולכן התוצאה היא באמת 2.

**ביצועי המערכת:**

התדר המקסימלי:

כדי לחשב את התדר המקסימלי של המערכת, לקחנו את הפרויקט של מטלה 1 (עטוף ברגיסטרים) והרצנו עליו את כל הבדיקות, כי רק בצורה זו ניתן למדוד את ביצועי המערכת.

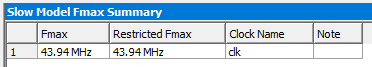
תדר העבודה המקסימלי שהתקבל הינו

Figure 10: Maximum operating frequency

ע"י שינוי קובץ sdc קבענו את השעון לעבוד במחזור של 23ns וכך גרמנו למערכת לעבוד בתדר .  
ההגבלה נוצרת בגלל הגבלה על זרימת המידע במערכת, כלומר לרכיב מסויים יש דילאיי מסויים ואם נשדר אליו יותר מדי מידע הוא לא יספיק לעבד אותו ונקבל תוצאות שגויות. לכן, יש לעבוד מתחת לתדר המקסימלי או בדיוק בתדר המקסימלי המחושב.

Logic usage (עבור TOP):

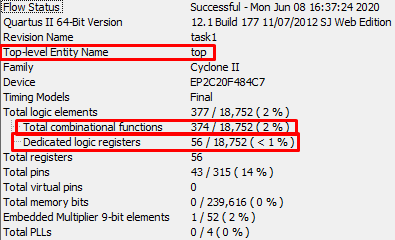
לאחר קומפילציה, התקבל המסך הבא:

Figure 11: Logical usage for top block

כפי שניתן לראות ב-figure 11, קיבלנו את השימוש ברכיבים שונים על כרטיס ה-FPGA. אכן, קיבלנו שיש שימוש רק במכפל 1, ואכן אמור להיות רק אחד.

מסלול קריטי (עבור TOP):

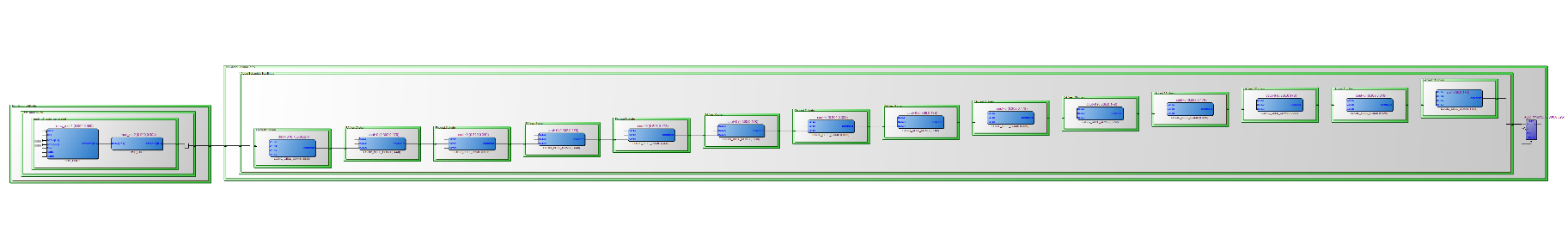
בדקנו את המסלול הארוך ביותר בעזרת ה-Timequest Analyzer, והתקבל המסלול הבא:

Figure 12: Critical path for top block

המסלול שהתקבל הוא מסלול שעובר דרך ה-MAC והמכפל שנמצא בתוכו.  
ניתן לומר כי זה היה צפוי, מכיוון שרכיב ה-MAC מבצע גם פעולת הכפלה, גם פעולת חיבור, והוא עושה זאת עבור וקטורים באורך 2n, בניגוד לשאר הפעולות במעבד שעובדות על וקטורים באורך n.

מסלול קריטי (עבור shifter):

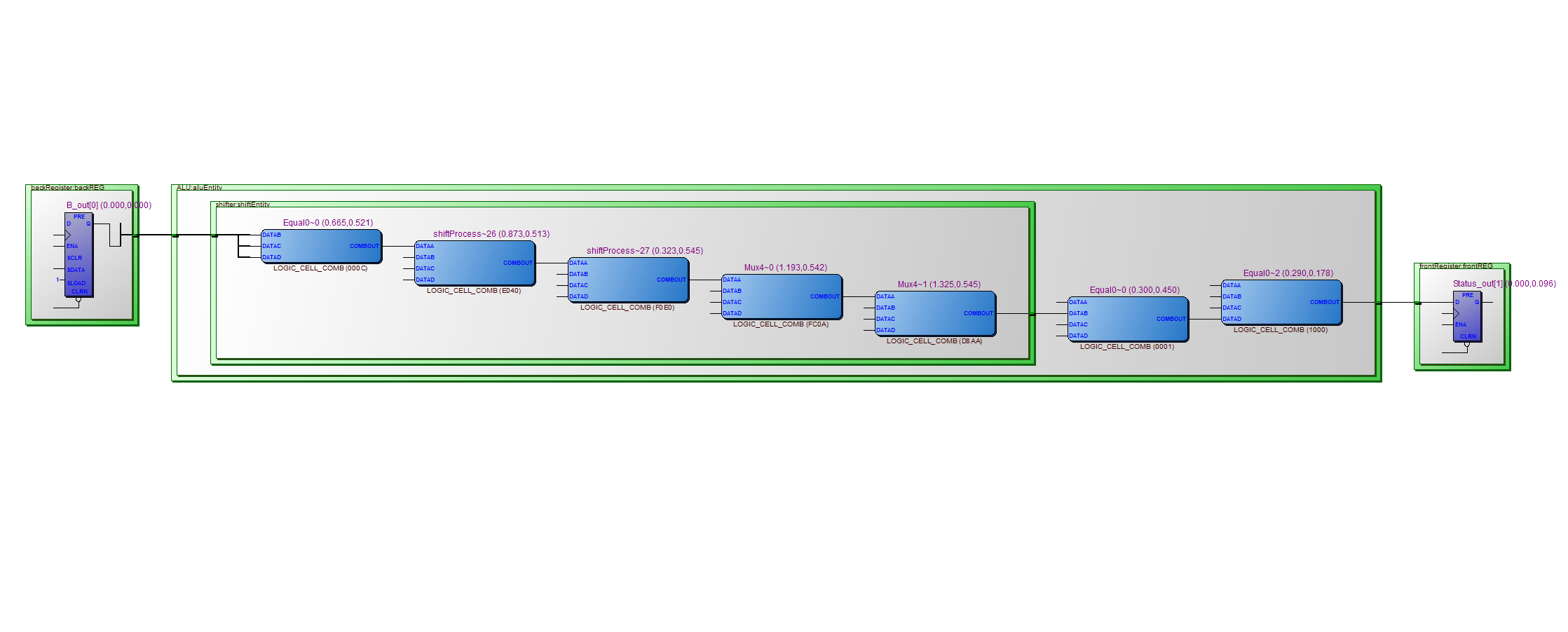
בדקנו את המסלול הארוך ביותר בעזרת ה-Timequest Analyzer, והתקבל המסלול הבא:

Figure 13: Critical path for shifter block

המסלול שהתקבל הוא מסלול עבור הזזה שמאלה ללא CARRY – RLA.  
אינטואיטיבית, המסלול שהיה אמור להיות הכי איטי הוא הזזה ימינה (מכיוון שנדרש גם להפוך את הביטים של HI), אבל מכיוון שאנחנו מטפלים בסיביות אלו במודול outputSelector אז זה לא בא כאן בחשבון.

Logic usage (עבור shifter):

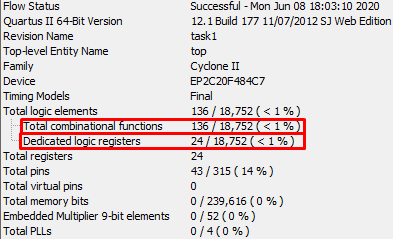
לאחר קומפילציה, התקבל המסך הבא:

Figure 13: Logical usage for shifter block

ניתן לראות שירדו הרבה מאוד רגיסטרים (וגם יחידות לוגיות, אבל זה לא אומר לנו הרבה). זאת מכיוון שאין יחידות סינכרוניות בתוך ה-shifter, אז כמות הרגיסטרים הנ"ל היא בעצם כמות הרגיסטרים שעוטפים את ה-top.

מסלול קריטי (עבור arithLogic):

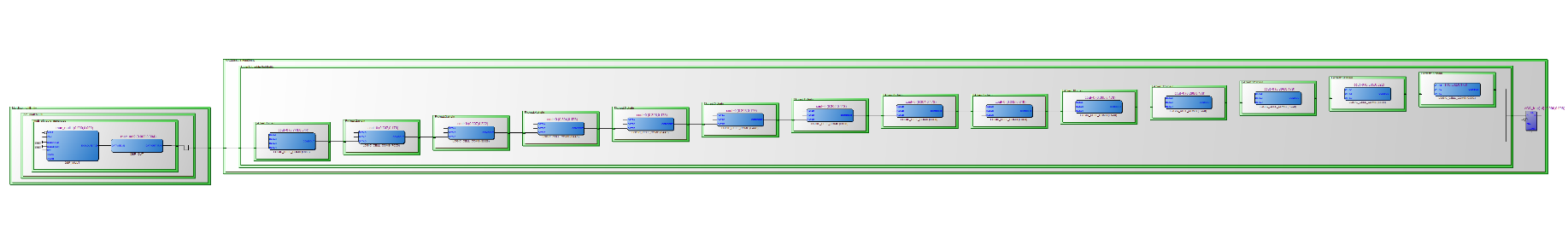
בדקנו את המסלול הארוך ביותר בעזרת ה-Timequest Analyzer, והתקבל המסלול הבא:

Figure 14: Critical path for arithLogic block

כצפוי, המסלול הכי איטי זה המסלול של ה-MAC.

Logic usage (עבור arithLogic):

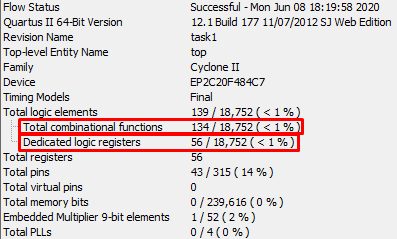
לאחר קומפילציה, התקבל המסך הבא:

Figure 15: Logical usage for arithLogic block

כפי שניתן לראות, מספר הרגיסטרים נשאר זהה (כי מודול זה מכיל את כל הרכיבים הסינכרוניים – והם בתוך ה-MAC). ואכן גם קיבלנו שיש שימוש במכפל 1.