דו"ח מעבדה דיגיטלית 1

יואב אשד 305384869

גלעד בינו 302252101

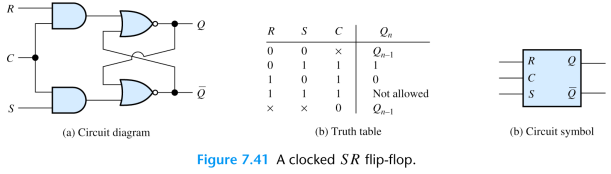
1. מימוש מסוג .

-רקע:

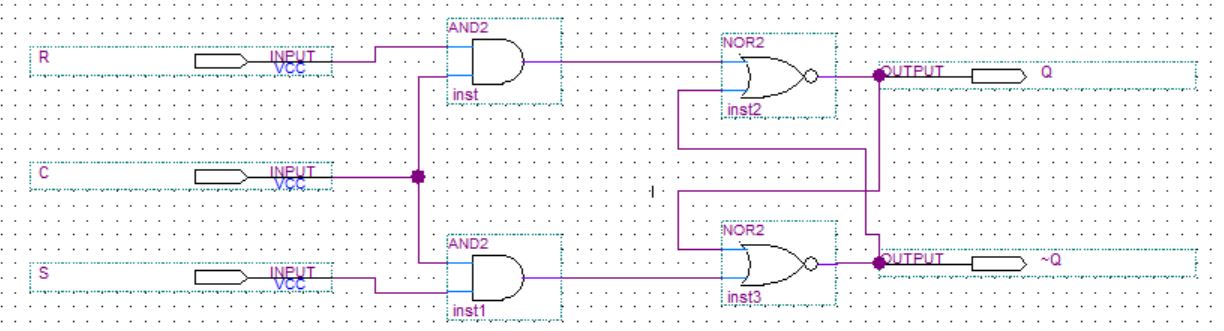
בתרגיל נתנה לנו טבלת האמת של מסוג , ומימוש שלו באמצעות שערים לוגיים עליו התבססנו לטובת יצירת המעגל ב.

*מסוג הוא אחד מהמעגלים הלוגים הבסיסיים ביותר. כזה מקבל ערך שני קלטים , ו , כאשר עבורו המוצא יהיה "1", ו עבורו המוצא יהיה "0".*

*תיאור הFF נקרא . זאת מאחר והקלט של הreset מאפס את הFF למצב המקורי שלו עם מוצא שיהיה בערך לוגי "1" או ערך לוגי "0" , תלוי במצב*

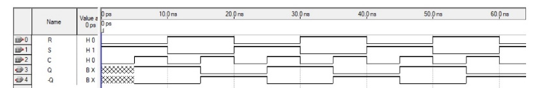
*ערך הFF ביציאה משתנה עבור עליית שעון, עבור "0" בשני הכניסות המצב לא ישתנה, עבור ערכים שונים בכניסות נקבל שינוי מצב של שתי כניסות "1" נקבל FF לא יציב ולכן המצב הזה אסור.*

-תכנון המעגל ב:



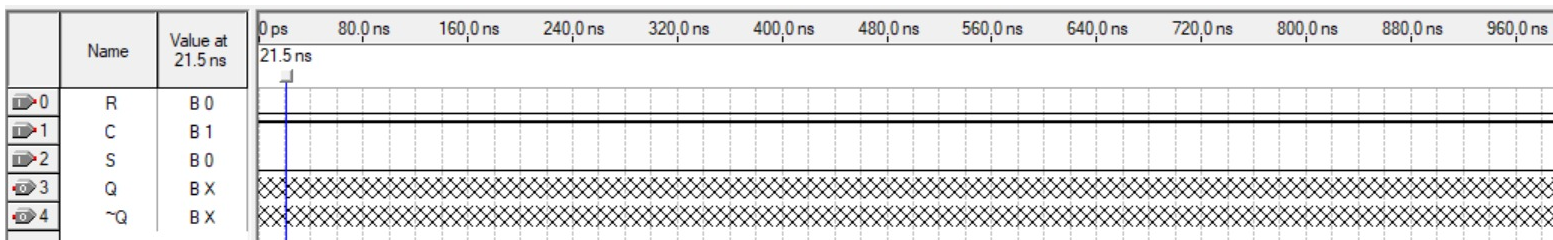
- סימולציית functional

מהסימולציות שבוצעו ניתן לשים לב כי התנהגות הFF תלויה בשעון, והמוצא משתנה עבור עליית השעון , כלומר הדגימה מתרחשת בעליית השעון , ונקבעת לפי ערך הכניסה. לכן הרכיב הינו רכיב סינכרוני.

 עבור כניסות :

ניתן לראות כי בכל דגימת שעון , המוצא משתנה בהתאם לכניסות ומקיים את דרישות טבלת האמת, כאשר המוצא מראה 1, ועבור *.*

*עבור כניסות*

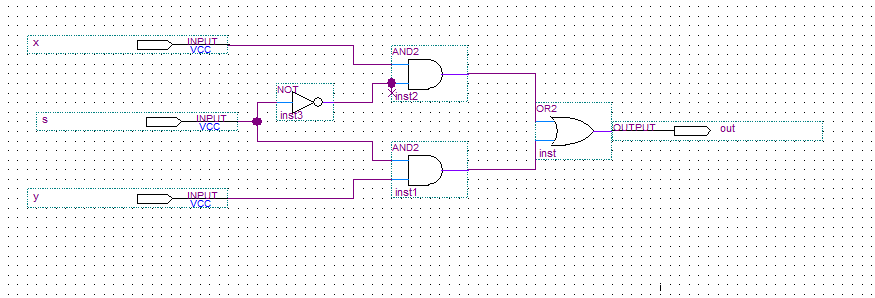


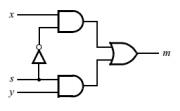
ניתן לראות כי המערכת לא עובדת במצופה

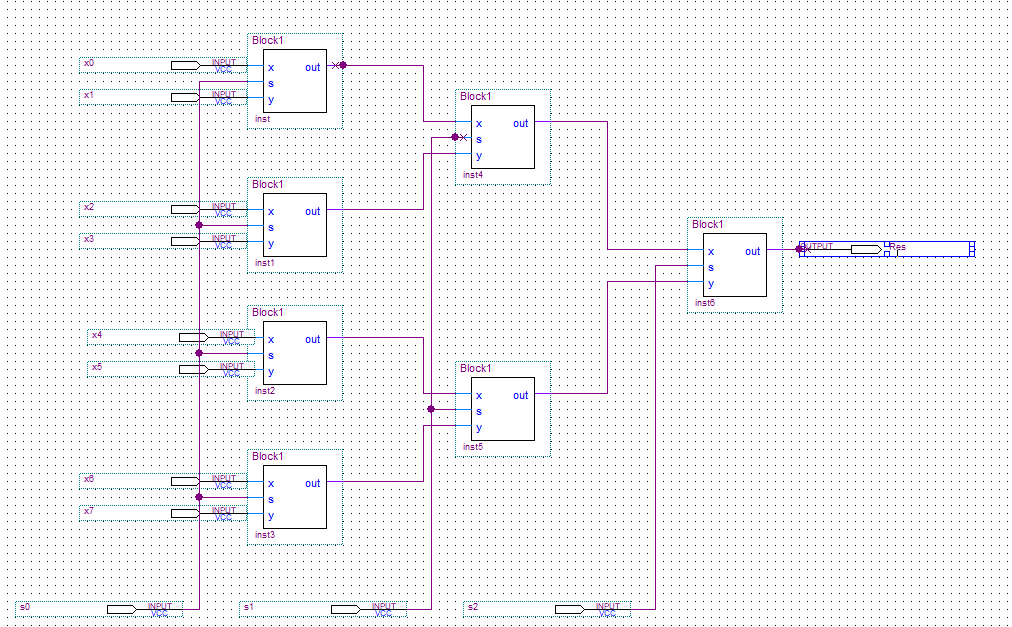
1. סלקטור 8 ל1 באמצעות סלקטור 2 ל1.

-רקע: או הינו רכיב עם קלטים מרובים ומוצא יחיד. לmux כניסות select הקובעים אילו מכניסות המידע מחוברות למוצא וכן את כמות המידע שיכולה לעבור בפרק זמן מסוים. ים יכולים לשמש ביישומים אנלוגיים או דיגיטליים , כאשר ביישומים אנלוגיים muxים מורכבים מטרנזיסטורים ושנאים, וביישומים דיגיטליים muxים מורכבים משערים דיגיטליים.

1. מימוש באמצעות דיאגרמת בלוקים, כאשר השתמשנו בשרטוט שסופק בתרגיל.

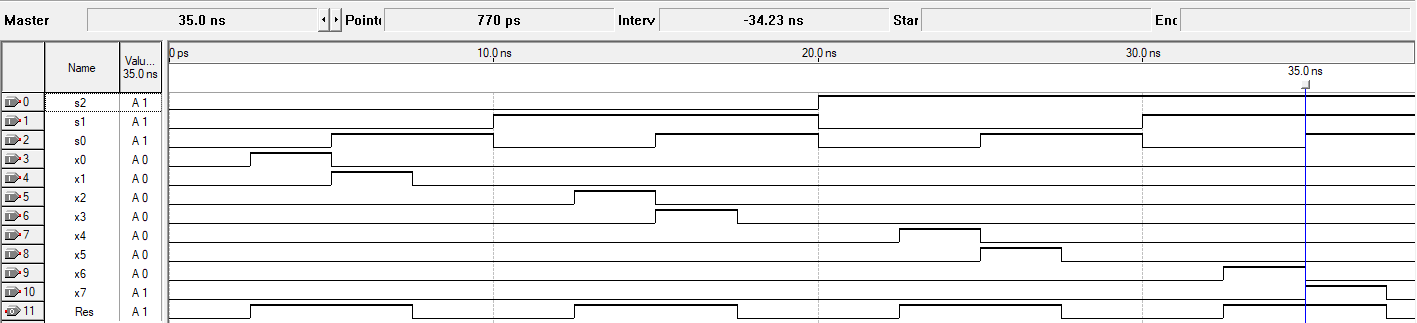
שרטוט שרטוט בquartus



ב. מימוש ע"י בלוקים של

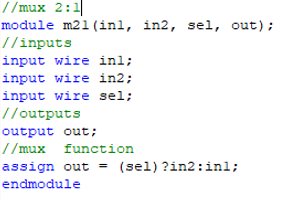
בחרנו לחבר את הmux 8:1 באופן זה מאחר וכך אנו יכולים להשתמש בבלוקים של אותו י יצרנו.

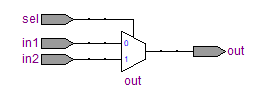
סימולציית תקינות ל



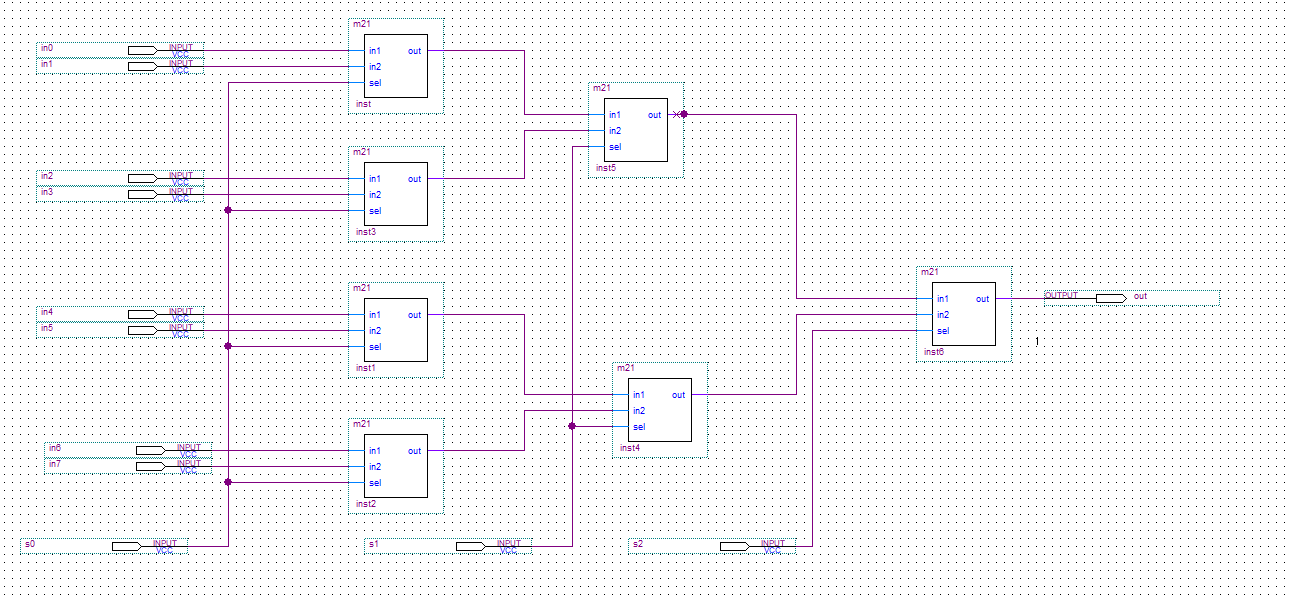
ניתן לראות כי שלושת הביטים של כניסת ה שולטים איזו כניסה מראה את היציאה. בדקנו את כל המקרים האפשריים עבור שלוש כניסות וניתן לראות שיש אות ביציאה רק כאשר יש אות באחת משמונה הכניסות וזה אכן תואם את טבלת האמת של כנדרש.

1. מימוש בלוק של mux2:1 בקוד ורילוג





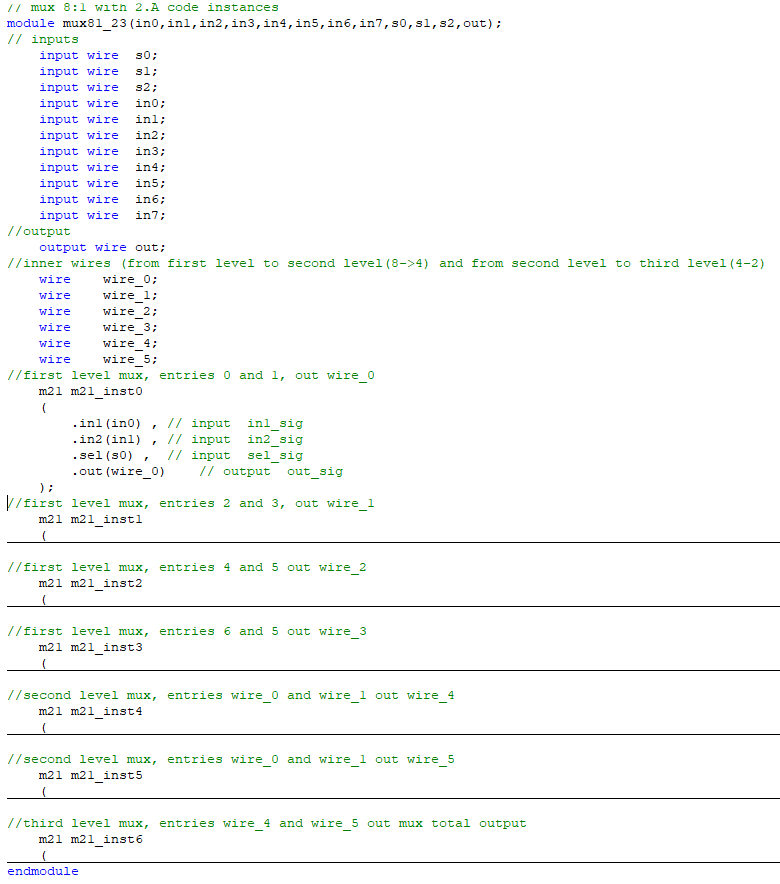
1. מימוש באמצעות בלוקים של הmux2:1 שתוכנן בקוד בסעיף הקודם , התכנון נובע מאותו שיקול של הסעיף הקודם.



.

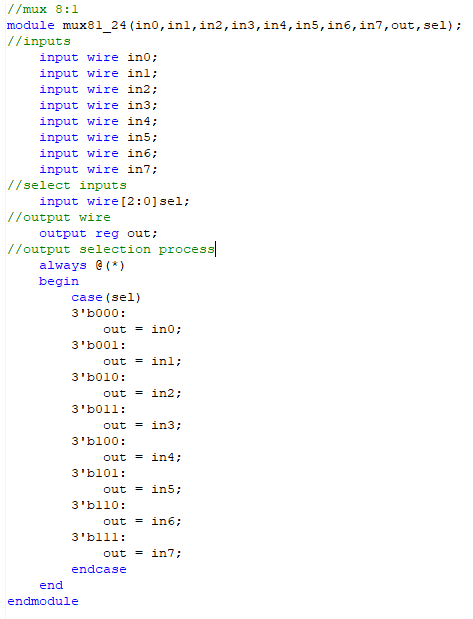
1. מימוש באמצעות מופעים של המודול מסעיף 2א'

כל מופע מייצג mux 2:1 וכמו בסעיפים הקודמים, 4 muxים מובילים ל2 וה2 מובילים ל1 שממנו מקבלים את אות היציאה.

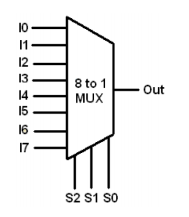


1. במימוש על ידי קוד.

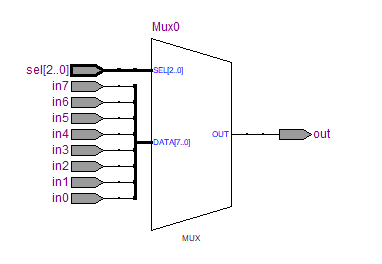
על מנת לממש בצורה הפשוטה ביותר בחרנו להשתמש בצורה של מקרים עבור כל צירוף אותות בקרה אפשריים.



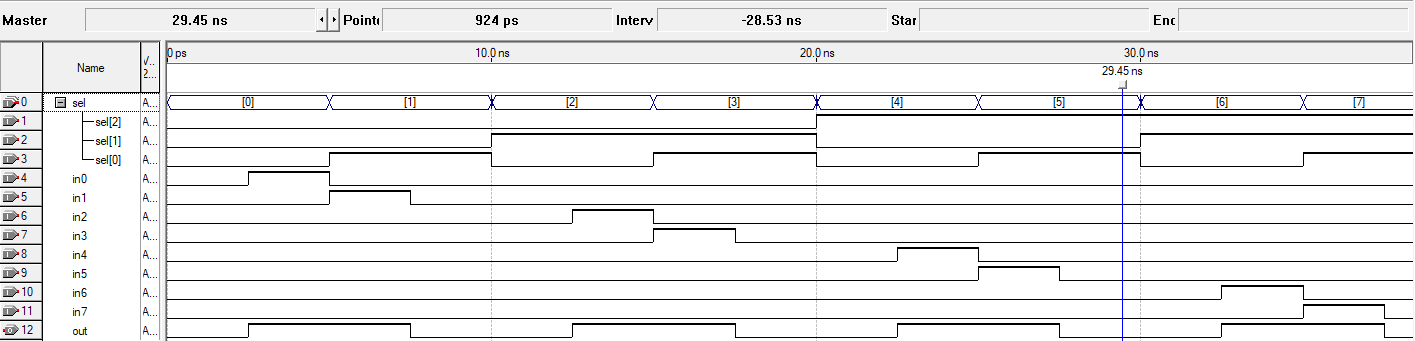
שרטוט שסופק בתרגיל



שרטוט



סימולציית פעולה:



ניתן לראות כי התוצאה של הסימולציה דומה עד זהה, גם כאן שלושת הביטים של כניסת ה שולטים איזו כניסה מראה את היציאה, וגם כאן היציאה עולה רק כאשר מוכנס אות באחת הכניסות וכי זה מקיים את טבלת האמת של ה, ולכן הmux מתפקד כנדרש.

-יתרונות לשימוש בשפת VHDL:

* מורכבות: כאשר יש לממש מעגל מורכב , אם מבחינת פונקציונאליות ואם מבחינת מספר הכניסות והיציאות של הרכיב, אז קל יותר לעשות זאת באמצעות כתיבת קוד מאשר באמצעות שרטוט. כך מתקבל סיכוי נמוך יותר לחיבורים שגויים או יצירת סכמה מסורבלת ולא קריאה.
* גמישות: על מנת לממש מעגל בסכמת בלוקים, נרצה להיעזר ברכיבים קיימים(כפי שעשינו בתרגיל הנ"ל). בדיאגרמת בלוקים אנו מוגבלים לפונקציות הקיימות בספרייה הקיימת ואם נרצה להוסיף נצטרך להוסיף את החסר באמצעות קוד. על ידי שימוש בקוד איננו נתונים למגבלות הסביבה.

-יתרונות בשימוש בסכמת בלוקים:

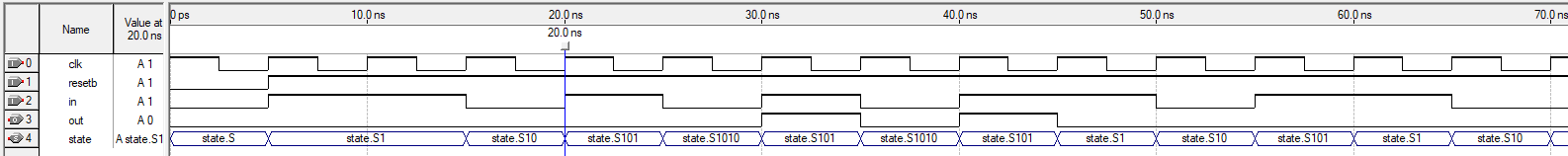
* נוחות השימוש: מימוש מעגל באמצעות דיאגרמת בלוקים הוא אינטואיטיבי ומהיר יחסית עבור מעגלים פשוטים.
* מבט על המערכת – יותר קל להבין דיאגרמת בלוקים מאשר להתחיל לעבור על קוד על מנת להבין כיצד המכונה עובדת.

3. מכונת מצבים FSM

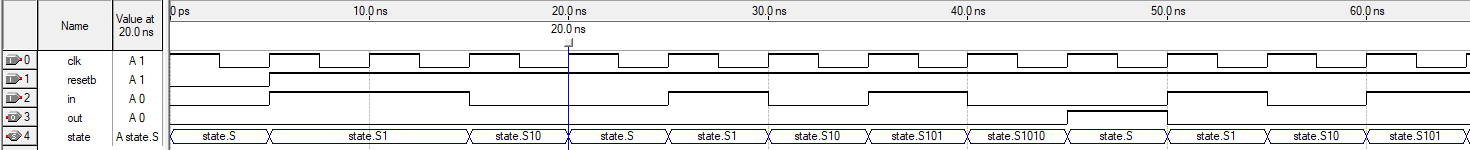
מימוש מכונת מצבים המזהה את הרצף 1010. המכונה ממומשת בקובץ יחיד (מצורף) , כאשר בתור קלט היא מקבלת אות כניסה , אות ריסט ואות שעון, ומוציאה אות מוצא שמשתנה מ0 ל1 רק כאשר הרצף 1010 נדגם.

את המצבים בחרנו לממש בצורה של מקרים (case) כאשר כל מייצג מצב במכונת המצבים והם מתחלפים בהתאם לשינוי באות הכניסה.

תוצאות עבור אות הכניסה אותו התבקשנו להכניס:



כמו כן ניסינו רצף מספרים נוסף על מנת לוודא את תקינות .

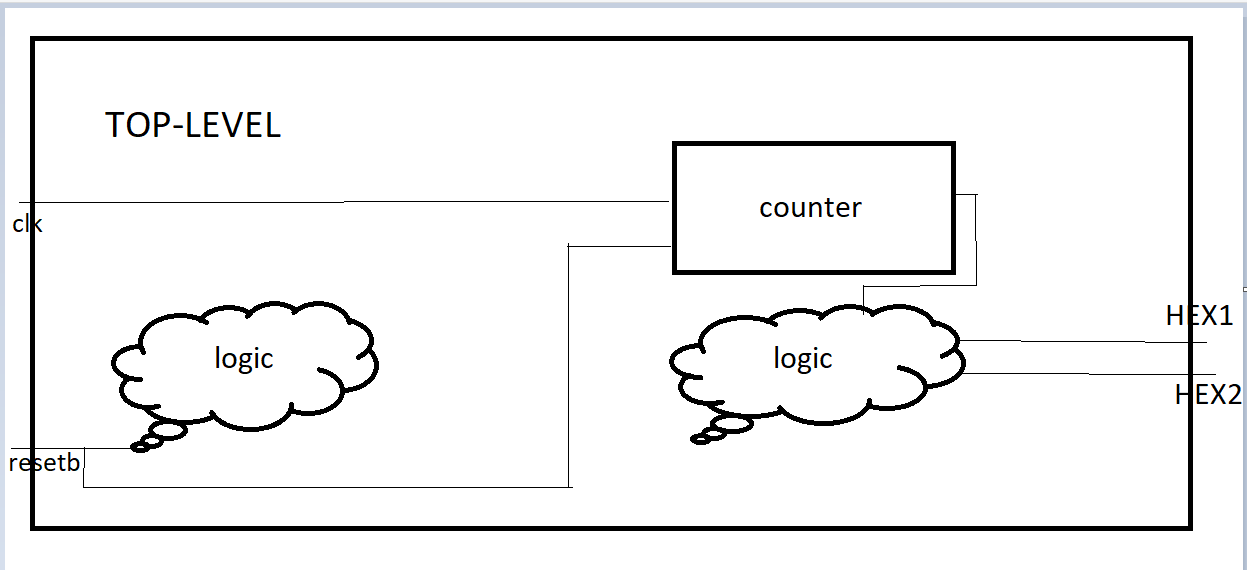


ניתן לראות כי מכונת המצבים עוברת ממצב למצב כנדרש ולפי ההגדרות אשר ניתנו בתרגיל.

4. מונה 99 שניות

נרצה להציג את המימוש שלנו למונה 99 שניות בתדר של 27Mhz. המימוש כולל TOP LEVEL ועוד תת תוכנית המכילה קאונטר.

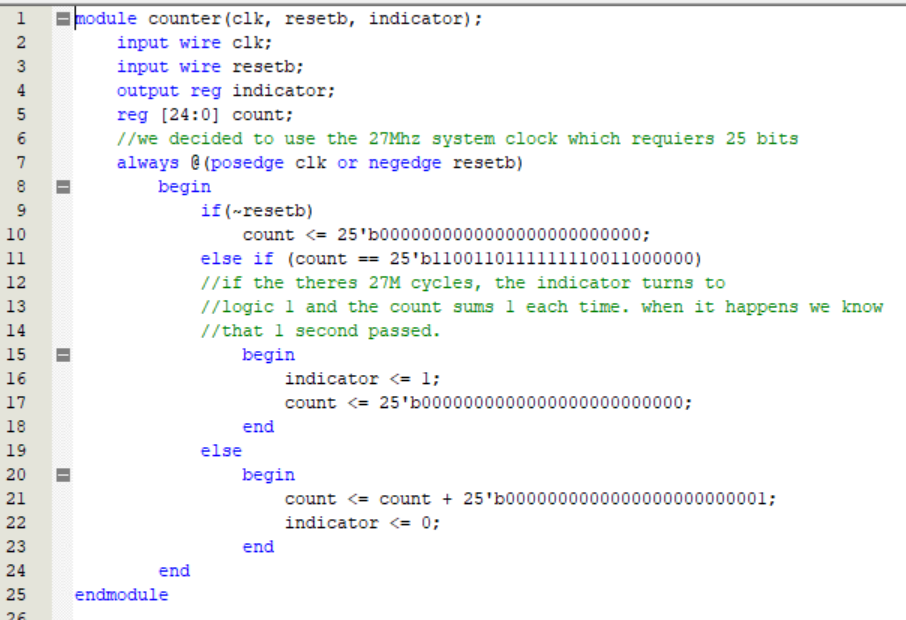
סכמת הבלוקים של המימוש שלנו:



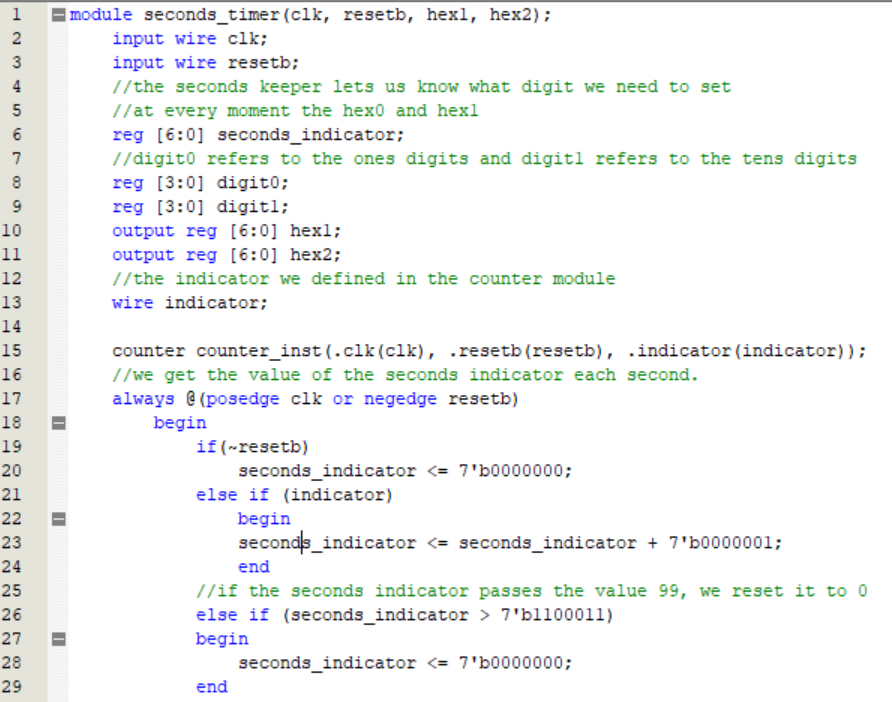
**נציין כי בבדיקות שלנו לא ספרנו 27 מיליון עליות שעון כי זהו מספר גדול מאוד. לכן בחרנו לספור 14 עליות שעון. מאחר והמימוש עבד עבור מספר עליות שעון זה, כפי שנראה בהמשך, נדע כי הוא יעבוד גם עבור 27 מיליון עליות שעון.**

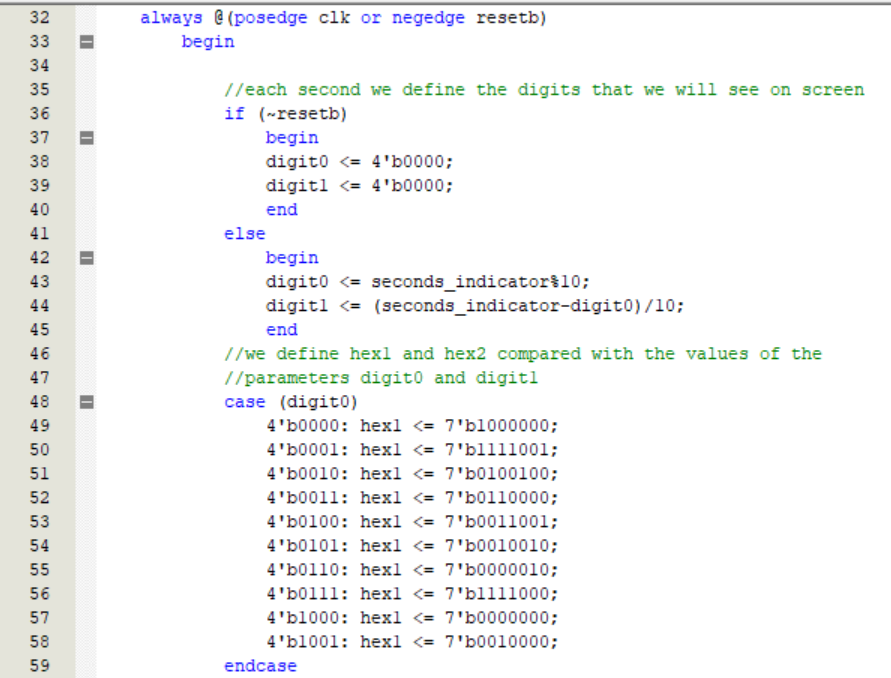
תיאור המימוש:

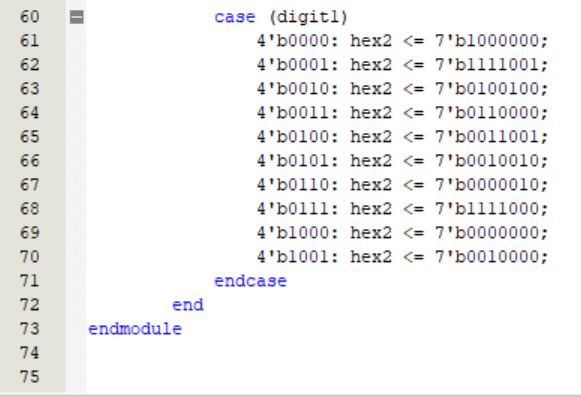
מטרתת תת התוכנית (COUNTER) הינה לספור שניות כך שהשעון הפנימי סופר 27 מיליון עליות שעון. כאשר השעון הגיע למספר זה הגדרנו אינדיקטור שיוחלף ל-1 לוגי ואז מתחיל לספור מחדש. ניתן לראות זאת בקוד המצורף:



כעת נסתכל על הקוד של ה- TOP LEVEL של התוכנית. שכאמור, היא מוגדרת לספור עד 99 ולאפס את המונה כאשר הוא מגיע למספר זה. בקוד שלנו הגדרנו אינדיקטור שמעלה את עצמו ב-1 עבור כל '1' לוגי שמתקבל מתת התוכנית COUNTER. בנוסף הגדרנו פרמטר לספרת האחדאות ופרמטר למספר העשרות ((digit0, digit1 שמשנים את ערכם לפי האינדיקטור. פרמטרים אלה בסופו של דבר קובעים את ערכם של היציאות HEX.



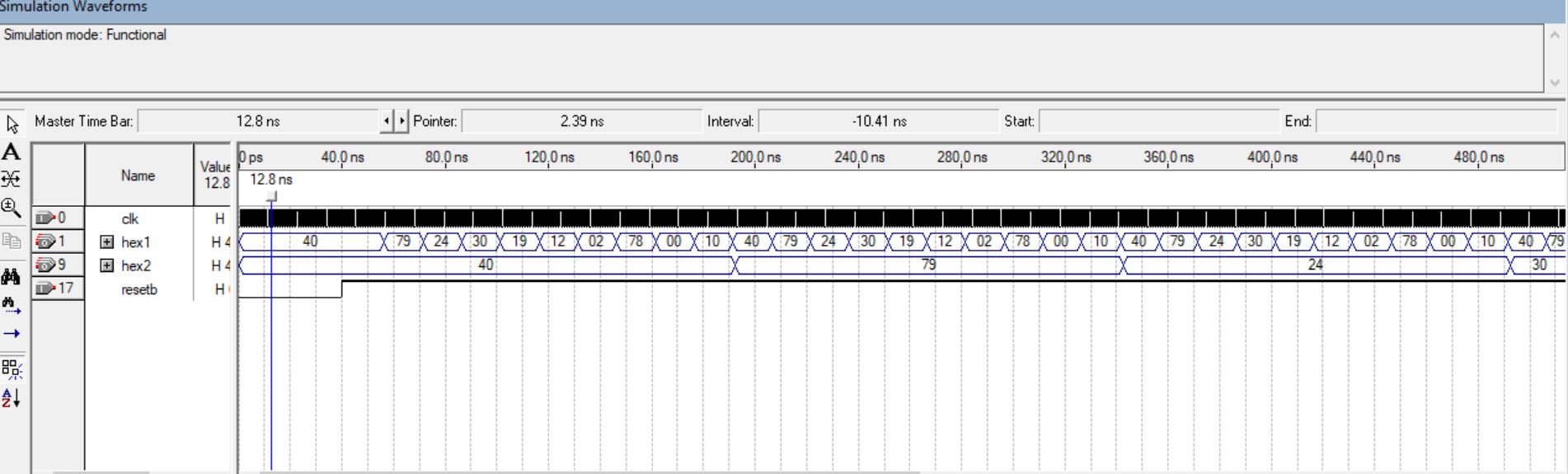


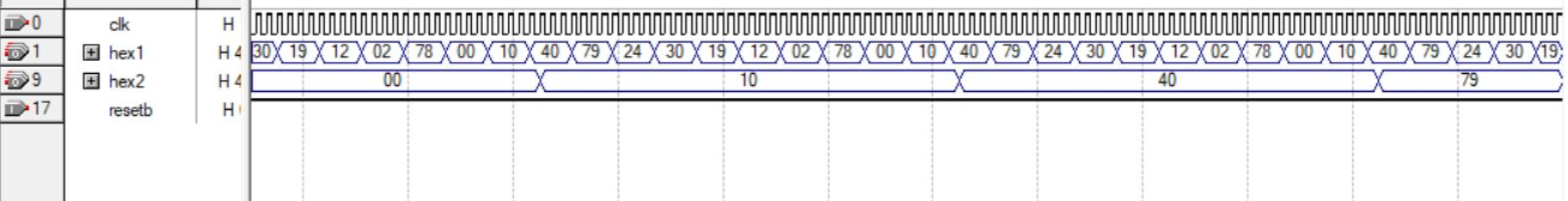


כעת נעבור לסימולציות של התוכנית.

הגדרנו תחילה את RESETB להיות 0 בשביל להראות שאכן כלום לא משתנה. בשלב זה שתי הספרות של המונה הן 0, כלומר 40 עבור HEX.

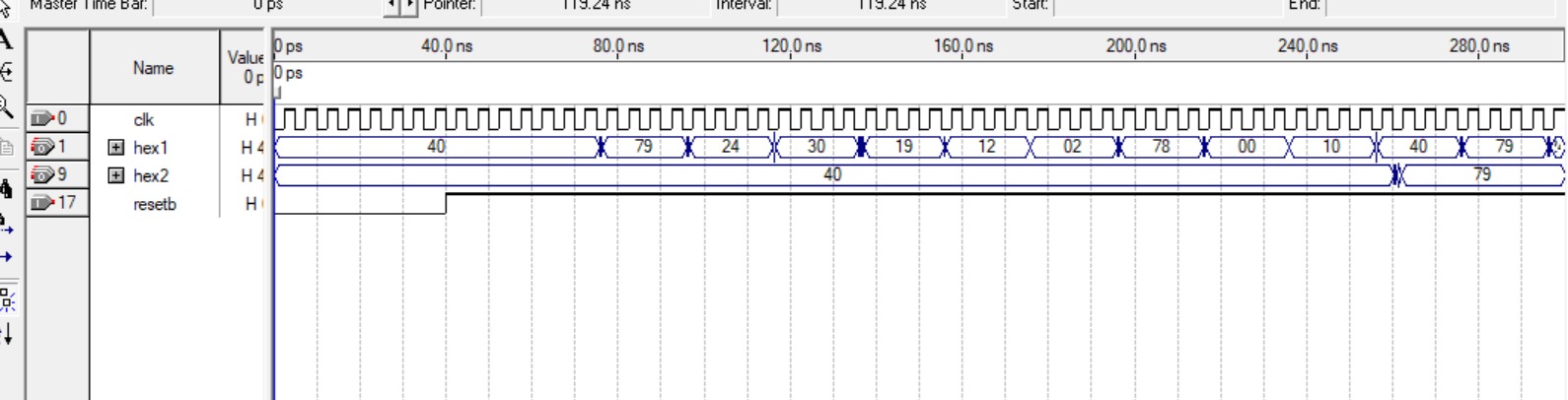
לאחר מכן RESETB משתנה ל-1 וספרת האחדות 1HEX מתחילה לגדול עד שמגיעה ל-9 ואז מתאפסת. כאשר היא מתאפסת ספרת העשרות 2HEX מתחילה היא לגדול וכן הלאה. בצילומים שנצרף נראה את התחלת המונה ואת רגע איפוס המונה כאשר הוא מגיע ל-99.





כפי שניתן לראות, המונה אכן מתאפס כאשר מגיע ל-99 וזאת כאשר שתי היציאות HEX שוות שתיהן ל-10 ואז מקבלות את הערך 40 שזה כאמור 0.

כעת נסתכל על בדיקת TIMING.



קיבלנו תוצאות עם מקטעי מספרים בין המעברים ביציאות. אנו מסיקים כי היו שינויים הקרובים מדי לעליית שעון או מנגד התרחשה קפיצה של אחד הרכיבים שגרמה לשינוי וזה עשוי לנבוע מכך שבדיקת ה- TIMING מסתכלת על דברים שלא בהכרח אידאליים.

נרצה למצוא גם את תדר המקסימלי של המונה. זאת עשינו בעזרת

Time Quest Timing Analyzer

ושם בחרנו שהתוכנה תחזיר לנו את התדבר המקסימלי. קיבלנו:



מהו ובמה תלוי התדר המקסימלי בו המונה שלכם יכול לעבוד?

הסקתינו היא שהתדבר המקסימלי נקבע על פי פעולת המערכת, כלומר מימוש הקוד והשערים הלוגים שמתקבלים מכך. לכן, בעקבות השימוש ב-CASE בקוד שלנו המעגל שלנו דורש המון השוואות שעלולות לצרוך חומרה רבה ובכך להוריד את התדבר המקסימלי של המונה.