תרגיל בית רטוב מספר 2

מערכות ספרתיות ומבנה מחשב

קראו היטב את הוראות ההגשה בסעיף 5.

נקודות יורדו למי שלא יבצע את ההוראות במדויק.

אחראי לתרגיל: סאמר כורזום ssamer15@campus.technion.ac.il

שאלות בקשר לדרישות התרגיל יש להפנות לאחראי התרגיל דרך הפורום במודל. בקשות מיוחדות יש לשלוח לאחראי התרגיל במייל.

בקשות לדחייה ללא סיבה מוצדקת ידחו על הסף (ראו נוהל להגשה באיחור).

אין להגיש שום חלק מודפס – את החלק היבש יש לכתוב במעבד תמלילים (למשל: Word) ולצרף לחלק הרטוב. יש להגיש כקובץ pdf בלבד.

1. הנחיות כלליות

מסמן שאלות שיש לענות עליהן במסמך (החלק היבש). ניתן לענות גם באנגלית.

במקרה של קושי בפתרון הסעיפים היבשים, יש לפנות למתרגלי הסדנאות בפורום SystemVerilog וסימולציות ב-Moodle, או בשעות הקבלה שלהם.

לצורך שרטוט מעגלים עם שערים לוגיים בתרגיל, ניתן לשרטט ידנית ולסרוק, או להשתמש בתוכנה (למשל: https://www.draw.io). את החישובים יש להקליד.

מסמן חלק שיש לבצע בסימולטור. את תוצאת הסימולציה (waveform) יש לצרף לחלק היבש ולהסביר את התוצאות בצורה איכותית. ניתן לשמור Waveform בעזרת צילום מסך. ב-waveform יש להכיל את האותות הרלוונטיים לתרגיל (כניסות ויציאות) ובצילום להראות את הקטעים הרלוונטיים בזמן. יש לדאוג שהתמונות תהיינה ברורות. נקודות יורדו על צילומי מסך שאינם ברורים.

יש לכתוב את קוד החומרה בשפת SystemVerilog בלבד וב-syntax שנלמד בסדנאות בלבד.

במקרה של קושי בפתרון הסעיפים הרטובים, יש לפנות למתרגלי הסדנאות בפורום SystemVerilog וסימולציות ב-Moodle, או בשעות הקבלה שלהם.

בכל שאלה על **דרישות התרגיל**, כלומר קשיים בהבנת דרישות הסעיפים השונים בתרגיל, יש לפנות אל **אחראי התרגיל,** בפורום SystemVerilog וסימולציות ב-Moodle, או בשעת הקבלה.

בנוסף, ניתן להיעזר במסמך בעיות ב-ModelSim וב-SystemVerilog הנמצא באתר הקורס (ModelSim_and_SystemVerilog_FAQ.pdf).

אין צורך לצרף את הקוד לחלק היבש אלא אם נאמר אחרת.

בתרגיל זה נעסוק במימושים שונים של מכפל, תחילה בחומרה הממומשת בעזרת מכונת מצבים סופית, ולבסוף בתוכנה על ידי קוד של RISC-V.

בתרגיל 1 נבנתה חומרה בשפת SystemVerilog בעזרת המודל המבני (structural), כלומר, יצירת שערים לוגיים בסיסיים וחיבור ביניהם. בתרגיל זה נעבור למימוש חומרה בעזרת המודל ההתנהגותי (behavioral). כתיבה במודל זה מאפשרת להשתמש בתיאור עילי יותר של הלוגיקה. לדוגמה, מחבר של שני מספרים ניתן למימוש באופן פשוט יותר בעזרת השורה הבאה:

```
assign result = operand1 + operand2;
```

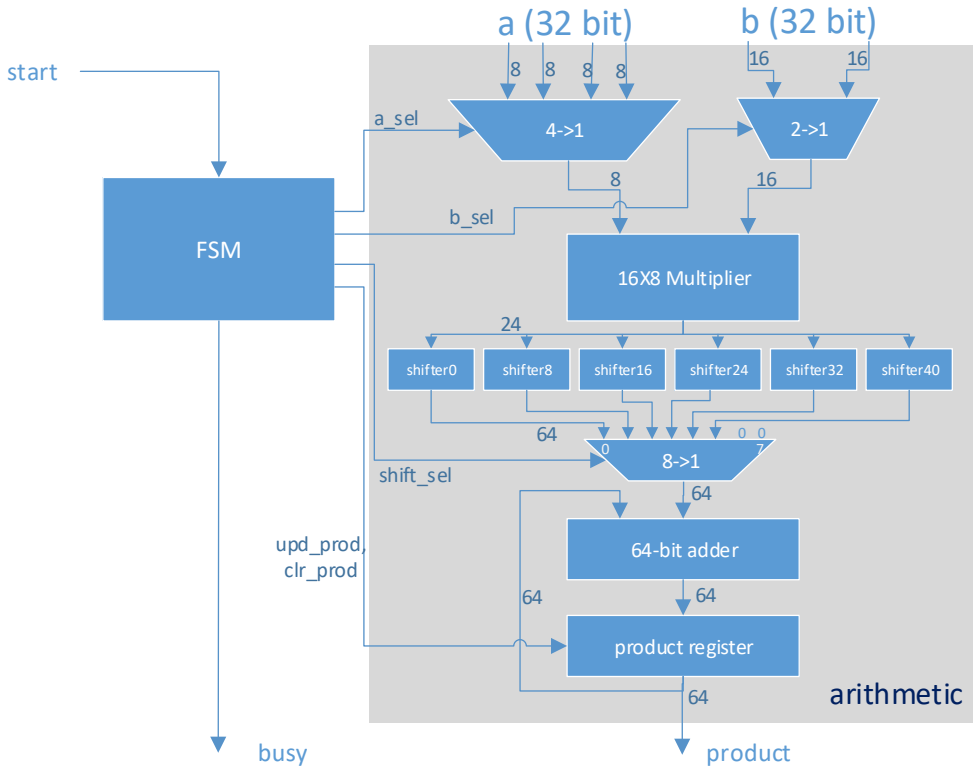
תהליך סינתזה מתרגם את השורה הזו למימוש דומה מאוד לזה שיצרתם באופן ידני עם שערים לוגיים בסיסיים בתרגיל 1. שימו לב שבניגוד למימוש בתרגיל 1, בתרגיל זה כל התכן ימומש ללא השהיות, לכן לאורך כל התרגיל **אין להשתמש ביחידות שמומשו בתרגיל 1**.

בנוסף, בתרגיל זה נתנסה לראשונה במימוש תכן סינכרוני (תלוי שעון).

2. חלק יבש

מימוש מכפל 32x32 באמצעות מכפל 16x8 ומכונת מצבים

ניתן לבנות מכפל עבור שני מספרים גדולים (לדוגמה, כל אחד בגודל 32 סיביות) ע"י שכפול של יחידות חישוב קטנות יותר (למשל, מכפלים של מספרים קטנים יותר, שכל אחד מהם יכול להיות מורכב מכמה מחברים, שבעצמם מורכבים משרשרת FA-ים וכו'...). אך למכפל כזה תהיה עלות גבוהה בשל מספר גדול **מאוד** של שערים במימוש. לחילופין, במקום לשכפל יחידות חישוב קטנות, ניתן לייצר אותן פעם אחת ולעשות בהן שימוש חוזר באופן סדרתי. במקרה שלנו, אפשר להשתמש במכפל קטן יותר (לדוגמה, 16x8 ביטים) בתוספת פעולות חיבור והזזה, וכן מכונת מצבים שמנהלת את כל היחידות הנ"ל. עקרון הפעולה של חישוב כזה דומה לפעולת כפל ארוך. להלן שרטוט של תכן שכופל שני מספרים ברוחב 32 סיביות, באמצעות מכפל של מספר ברוחב 16 סיביות במספר ברוחב 8 סיביות:



שימו לב שלצורך נראות בלבד לא צוינו בשרטוט הכניסות clk ו-reset, אך הן אכן קיימות ומחוברות לכל רכיבי הזיכרון בתכן, כמו בכל תכן סינכרוני. בנוסף, ה-product register הוא FF.

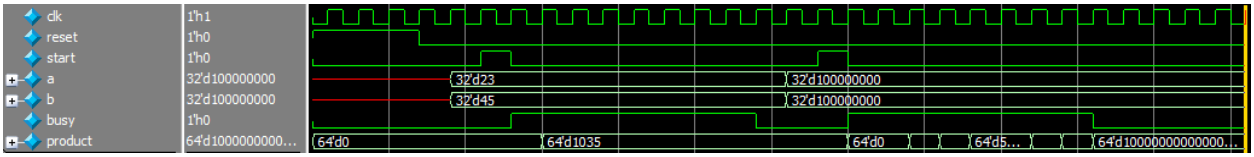
כמו-כן, לאורך כל התרגיל ניתן להניח כי הכניסות מסונכרנות לעליית השעון וכן כי המספרים המוכפלים בייצוג unsigned.

2.1. תכננו מכונת מצבים (FSM) מסוג Mealy השולטת על פעולת הכפל בתכן הנתון בשרטוט.
שימו לב שכדי שמכונת מצבים תהיה מסוג Mealy, מספיק שלפחות במצב אחד, אחת היציאות תהיה תלויה צירופית באחת הכניסות.
ה-port-ים של המכונה מתוארים בטבלה הבאה:

שם	כיוון	גודל [ביטים]	תאור
clk	כניסה	1	שעון המערכת בעל זמן מחזור של 10 יחידות זמן
reset	כניסה	1	Reset אסינכרוני, פעיל ב-'1'
start	כניסה	1	כאשר המכונה אינה עובדת וכניסה זו ב-'1', המכונה מתחילה לעבוד; כל עוד המכונה עובדת, כניסה זו לא משפיעה על פעולת המכונה
busy	יציאה	1	'1' כל עוד המכונה עובדת (במחזור השעון לאחר ש-start עולה), '0' אחרת
a_sel	יציאה	2	Select לבורר הבוחר את אחד מארבעת הבתים של הכניסה a
b_sel	יציאה	1	Select לבורר הבוחר את אחת מארבע המילים (באורך 16 ביט) של הכניסה b
shift_sel	יציאה	3	Select לבורר הבוחר את תוצאת אחד מה-shifter-ים
upd_prod	יציאה	1	'1' כאשר ה-product register צריך לדגום את הערך שבכניסה שלו, '0' אחרת
clr_prod	יציאה	1	'1' כאשר ה-product register צריך להתאפס, '0' אחרת; קיימת עדיפות לאות זה על-פני האות upd_prod, כלומר, כאשר clr_prod='1', ה-product register מתאפס בלי תלות בערך של upd_prod

לשימושכם, מצורפת דיאגרמת גלים בה ניתן לראות דוגמה לפעולת המכפל הנדרשת. בדוגמה זו המכפל מבצע שתי פעולות כפל:

- 23 x 45
- 100,000,000 x 100,000,000



היעזרו בדיאגרמה על-מנת להבין את דרך פעולת המכונה (למשל, מתי busy צריך לעלות וכן הלאה).
בתכנון מכונת המצבים יש לוודא שהמכפל תומך בביצוע מספר לא מוגבל של פעולות כפל אחת אחרי השנייה ללא reset ביניהן. תוצאת כל פעולת כפל צריכה להישמר במוצא המכפל עד קבלת start נוסף.
יש לתכנן מכונת מצבים מצומצמת עם 9 מצבים.
ציירו את דיאגרמת המצבים של המכונה. אין צורך להציג טבלת מצבים או שרטוט של החומרה הנוצרת.
כמה מחזורי שעון לוקחת פעולת הכפל?

2.2. כאשר בית מסוים (8 סיביות) של אחד הגורמים במכפלה שווה ל-0, תוצאת כפל של הבית הזה בכל מספר אחר היא גם 0. ניתן לנצל את התכונה הזו כדי לייעל את תהליך ההכפלה על ידי זיהוי של הבתים האלה מראש ודילוג על שלבי החישוב הרלוונטיים במכונה.

הציעו שינוי במכונת המצבים המקורית כך שהפעולה תהיה מהירה יותר אם ה-Most Significant Byte (סיביות 24 עד 31 כולל) בכניסת A ו/או ה-Most Significant Word (סיביות 16 עד 31 כולל) בכניסה

B שווים ל-0.
בנוסף ל-port-ים של מכונת המצבים המקורית, למכונת המצבים החדשה יש שתי כניסות נוספות:

שם	כיוון	גודל [ביטים]	תאור
a_msb_is_0	כניסה	1	'1' כאשר ה-Most Significant Byte של A שווה ל-0, '0' אחרת
b_msw_is_0	כניסה	1	'1' כאשר ה-Most Significant Word של B שווה ל-0, '0' אחרת

ניתן להניח כי כניסות אלו מתעדכנות בהתאם לערכים של a ו-b בתחילת החישוב, ונותרות יציבות עד סוף החישוב.
ציירו את דיאגרמת המצבים החדשה. אין צורך להציג טבלת מצבים או שרטוט של החומרה הנוצרת.
כמה מחזורי שעון לוקחת פעולת הכפל כעת? תחת אלו תנאים המכונה תעבוד הכי מהר?

מימוש פעולת כפל 16x16 באמצעות פקודת כפל 16x8 בתוכנה

2.3. בדומה לנעשה בשאלות הקודמות, גם בתוכנה משתמשים במשאבים מוגבלים כדי לבצע משימות מורכבות.
לצורך תרגיל זה, הניחו שברשותכם מעבד שיודע לכפול מספר בגודל 8 סיביות במספר בגודל 16 סיביות ולהוציא תוצאה בגודל 24 סיביות.
עליכם לתכנן אלגוריתם תוכנה שכופל שני מספרים בגודל 8N סיביות, כאשר N מספר טבעי וזוגי. ניתן להניח כי המספרים בייצוג unsigned. כמו כן, ניתן להניח כי המשתנים שמשתמשים בהם באלגוריתם גדולים ככל שתרצו.
תארו בפירוט את האלגוריתם ואת אופן פעולתו. ניתן, אך לא חובה, להשתמש גם בתרשים זרימה ו/או pseudo code לצורך תיאור האלגוריתם. ציינו את הקשר בין N לבין זמן הביצוע של האלגוריתם (סיבוכיות זמן ריצה).

2.4. בקובץ mult16x16.s כתבו קוד אסמבלי של RISC-V שכופל שני מספרים בגודל 16 סיביות תוך שימוש בפעולת כפל של 16x8 סיביות ופעולות נוספות. ניתן לפתוח ולערוך את הקובץ עם notepad++.
שימו לב שבקובץ mult16x16.s שקיבלתם, מופיעים כבר שני קטעי קוד שמממשים פעולת כפל 16x8:

```
# Mask for 16x8=24 multiply
ori     t0, x0, 0xff
slli    t0, t0, 8
ori     t0, t0, 0xff
slli    t0, t0, 8
ori     t0, t0, 0xff
```

קטע הקוד הנ"ל מכין באוגר t0 מסכה של '1'-ים ב-24 הסיביות התחתונות של האוגר, ו-'0'-ים בכל שאר הסיביות. ביצוע bitwise AND בין האוגר הזה לאוגר אחר תגרום לאיפוס 8 הביטים העליונים של הערך השמור באוגר השני. ניתן להשתמש בתוצאת הקוד הנ"ל (המסכה המוכנה באוגר t0) בקוד שלכם. אין צורך לשכפל את קטע הקוד הנ"ל לתוך הקוד שלכם.

```
# Use the code below for 16x8 multiplication
#   mul           <PROD>, <FACTOR1>, <FACTOR2>
#   and           <PROD>, <PROD>, t0
```

הקוד בהערה הנ"ל מדגים את אופן השימוש במסכה לצורך מימוש כפל 16x8. הפקודה mul מבצעת כפל בין מספרים יותר גדולים מ-16x8. לכן, מטרת המסכה היא לקחת רק את הביטים התחתונים של המכפלה. תוצאת פעולת ה-mul (בגודל 32 סיביות) עוברת פעולת bitwise AND עם המסכה, מה שמותיר רק 24 הסיביות התחתונות של פעולת הכפל, כפי שנדרש בפעולת כפל 16x8. יש להשתמש

בשתי השורות הנ"ל לפי הדוגמה בתוך הקוד שלכם בכל מקום שבו תרצו לבצע פעולת כפל. חובה להשתמש בשתי השורות באופן המודגם הנ"ל על-מנת לבצע את הכפל.
בסוף ההרצה כתבו את תוצאת הכפל לאוגר t6.
הריצו את הקוד בסימולטור שבאתר: http://www.kvakil.me/venus
הוסיפו לחלק היבש צילום מסך של הסימולטור לאחר הרצת הסימולציה. ודאו שערך האוגר t6 מופיע בצילום.
בהנחה שזמן הביצוע של כל פקודה הוא מחזור שעון אחד (כולל כל אחת מהפקודות mul, and שמשתתפות בפעולת הכפל), כמה מחזורי שעון לוקחת פעולת הכפל? מדדו את פעולת הכפל בלבד, ללא פעולות האתחולים וה-finish (כלומר, רק קטע הקוד שבין Start of your code לבין End of your code).

2.5. תארו מהו השינוי הנדרש בקוד מסעיף 2.4 כדי לממש דילוגים על אפסים ב<u>בית</u> העליון של a ו/או b (בדומה לסעיף 2.2), כלומר כאשר כל הבית העליון של a מאופס, כל הבית העליון של b מאופס, או שניהם. ניתן לצרף את הקוד החדש או להסביר את השינויים בלבד. הבדיקה האם בתים אלו מאופסים צריכה להיות כלולה באלגוריתם. איך ישתנה זמן הריצה של התוכנה? האם השינוי הזה משתלם?

3. <u>חלק רטוב</u>

לפני תחילת העבודה על החלק הרטוב, מומלץ לצפות בוידאו הבא. הוידאו מדגים טעויות נפוצות בכתיבת קוד הממשמ FSM, ומסתמך על קוד שנכתב בסדנה 5. לכן יש לצפות בו לאחר הסדנה:

https://www.youtube.com/watch?v=eGfABYJcMSs

הנחיה כללית: שימו לב כי בכל קובץ קוד שקיבלתם יש לממש module אחד בלבד. אין לממש מספר modules בקובץ אחד, ואין להוסיף קבצי קוד חדשים בנוסף לאלו שקיבלתם.

3.1. בקובץ mult32x32_fsm.sv כתבו קוד SystemVerilog הממשמ את המכונה שתכננתם בסעיף 2.1.
3.2. בקובץ mult32x32_arith.sv כתבו קוד SystemVerilog הממשמ את היחידה האריתמטית המתוארת בשרטוט. שימו לב שמכונת המצבים <u>אינה</u> חלק מיחידה זו.
ה-port-ים של היחידה מתוארים בטבלה הבאה:

שם	כיוון	גודל [ביטים]	תאור
clk	כניסה	1	שעון המערכת בעל זמן מחזור של 10 יחידות זמן
reset	כניסה	1	Reset אסינכרוני, פעיל ב-'1'
a	כניסה	32	האופרנד A להכפלה
b	כניסה	32	האופרנד B להכפלה
a_sel	כניסה	2	Select לבורר הבוחר את אחד מארבעת הבתים של הכניסה a
b_sel	כניסה	1	Select לבורר הבוחר את אחת משתי המילים (באורך 16 ביט) של הכניסה b
shift_sel	כניסה	3	Select לבורר הבוחר את תוצאת אחד מה-shifter-ים
upd_prod	כניסה	1	'1' כאשר ה-product register צריך לדגום את הערך שבכניסה שלו, '0' אחרת
clr_prod	כניסה	1	'1' כאשר ה-product register צריך להתאפס, '0' אחרת; קיימת עדיפות לאות זה על-פני האות upd_prod, כלומר, כאשר clr_prod='1', ה-product register מתאפס בלי תלות בערך של upd_prod
product	יציאה	64	תוצאת הכפל

ניתן להשתמש בפעולת הכפל של SystemVerilog (האופרטור: *) על מנת לממש מכפל 16x8.

3.3. בקובץ mult32x32.sv כתבו קוד SystemVerilog המממש את ההיררכיה העליונה של המכפל ע"י יצירת שתי היחידות הקודמות שמימשתם וחיבור ביניהן.

3.4. בקובץ mult32x32_test.sv כתבו testbench עבור יחידת ה-mult32x32 שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה.
יש לבדוק את התכן ע"י הכפלת שני מספרי הזהות (כמספרים עשרוניים) של שני הסטודנטים בזוג.
על ה-testbench לייצר אות reset שפעיל בתחילת הסימולציה ל-4 מחזורי שעון ולאחר מכן יורד עד סוף הסימולציה, וכן אות שעון תקין.
כמו-כן, על ה-testbench לבצע את רצף הפעולות הבא:
- אתחול start בערך '0'
- לאחר יציאה מ-reset, הצבת הכניסות a ו-b בערכי מספרי הזהות
- המתנה של מחזור שעון אחד
- הצבת '1' ב-start למשך מחזור שעון אחד
- המתנה להתייצבות היציאה (ירידה של האות busy)
הציגו בדיאגרמת הגלים את כל הכניסות והיציאות של יחידת ה-mult32x32. כמו-כן, הציגו את האותות current state ו-next state של מכונת המצבים.
צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. ודאו שניתן לראות בבירור את כל האותות הנדרשים, ובפרט את תוצאת המכפל ומצבי מכונת המצבים. במקרה הצורך, ניתן לבצע zoom-in ולצרף את הדיאגרמה במספר צילומי מסך.
שימו לב: התכן ייבדק בבדיקות אוטומטיות גם עבור מקרים נוספים, מעבר לאלו שאותם אתם נדרשים להגיש. יש לוודא שהתכן אכן עומד בדרישות גם במקרים נוספים (אך אין צורך להגיש בדיקות של מקרים נוספים). הקפידו שסיגנל busy יתנהג בדיוק כמו שניתן לראות בדיאגרמה שניתנה בסעיף 2.1, ושתוצאת המכפלה נשמרת במוצא עד עליית start. התנהגות שונה עלולה לגרור כשלון בכל הטסטים האוטומטיים הבודקים את התרגיל.
הערה: ניתן להגדיר את תצוגת המספרים בדיאגרמת הגלים ל"עשרונית ללא-סימן" ע"י קליק ימני על הסיגנל ואז לבחור Radix ו: Unsigned.

3.5. בקובץ mult32x32_fast_fsm.sv כתבו קוד SystemVerilog המממש את המכונה שתכננתם בסעיף 2.2.

3.6. בקובץ mult32x32_fast_arith.sv כתבו קוד SystemVerilog המממש יחידה אריתמטית חדשה.
היחידה זהה לזו שמימשתם קודם, בתוספת שתי יציאות חדשות: a_msb_is_0 ו-b_msw_is_0.
הוסיפו ליחידה האריתמטית לוגיקה צירופית אשר מממשת את שתי היציאות החדשות.
בקובץ mult32x32_fast.sv כתבו קוד SystemVerilog המממש את ההיררכיה העליונה של המכפל המהיר ע"י יצירת שתי היחידות mult32x32_fast_fsm ו-mult32x32_fast_arith וחיבור ביניהן.

3.7. בקובץ mult32x32_fast_test.sv כתבו testbench עבור יחידת ה-mult32x32_fast שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה.
יש לבדוק את התכן בשני מקרים: הכפלת שני מספרי הזהות (כמספרים עשרוניים) של שני הסטודנטים בזוג (בדומה לבדיקה ב-testbench הקודם), וכן הכפלת שני מספרי הזהות כאשר שני הבתים העליונים של תעודת הזהות הראשונה מאופסים וגם שני הבתים העליונים של תעודת הזהות השנייה מאופסים.
על ה-testbench לייצר אות reset שפעיל בתחילת הסימולציה ל-4 מחזורי שעון ולאחר מכן יורד עד

סוף הסימולציה, וכן אות שעון תקין.
כמו-כן, על ה-testbench לבצע את רצף הפעולות הבא:
- אתחול start בערך '0'
- לאחר יציאה מ-reset, הצבת הכניסות a ו-b בערכי מספרי הזהות
- המתנה של מחזור שעון אחד
- הצבת '1' ב-start למשך מחזור שעון אחד
- המתנה להתייצבות היציאה (ירידה של האות busy)
- המתנה של מחזור שעון אחד לפחות
- הצבת הכניסות a ו-b בערכי מספרי הזהות עם שני הבתים העליונים מאופסים
- המתנה של מחזור שעון אחד
- הצבת '1' ב-start למשך מחזור שעון אחד
- המתנה להתייצבות היציאה (ירידה של האות busy)
הציגו בדיאגרמת הגלים את כל הכניסות והיציאות של יחידת ה-mult32x32_fast. כמו-כן, הציגו את האותות current state ו-next state של מכונת המצבים.
צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. התייחסו בהסבר להבדל בזמן הריצה בין שני המקרים. ודאו שניתן לראות בבירור את כל האותות הנדרשים, ובפרט את תוצאת המכפל ומצבי מכונת המצבים. במקרה הצורך, ניתן לבצע zoom-in ולצרף את הדיאגרמה במספר צילומי מסך.
שימו לב: התכן ייבדק בבדיקות אוטומטיות גם עבור מקרים נוספים, מעבר לאלו שאותם אתם נדרשים להגיש. יש לוודא שהתכן אכן עומד בדרישות גם במקרים נוספים (אך אין צורך להגיש בדיקות של מקרים נוספים).

4. חלוקת הציון

Sect	Grade	Sect	Grade
2.1	8	3.1	3
2.2	8	3.2	3
2.3	10	3.3	2
2.4	15	3.4	17
2.5	9	3.5	3
		3.6	4
		3.7	18

Total	50		50

5. הוראות הגשה

- ההגשה בזוגות בלבד. ניתן לחפש בני זוג דרך פורום חיפוש שותפים. הגשה ללא בן זוג ללא אישור מראש תגרור הורדה בציון של 10 נקודות.
- יש להגיש את הקבצים הבאים (**ואותם בלבד**):
  - mult16x16.s
  - mult32x32.sv
  - mult32x32_arith.sv
  - mult32x32_fast.sv
  - mult32x32_fast_arith.sv
  - mult32x32_fast_fsm.sv
  - mult32x32_fast_test.sv
  - mult32x32_fsm.sv
  - mult32x32_test.sv
  - Dry.pdf
- יש לארוז את כל הקבצים הנ"ל (קבצי הקוד וקובץ התשובות של החלק היבש בפורמט pdf בלבד) בקובץ zip אחד, בשם **<id>.zip**, כאשר <id> זהו מס' ת.ז. מלא של אחד מבני הזוג.
- בתחילת קובץ התשובות לחלק היבש יש לכתוב שמות ות.ז. של כל אחד מהסטודנטים בטבלה כמו בדוגמה הבאה:

שם 1	123456789
שם 2	987654321

- שימו לב להגיש קובץ בפורמט zip בלבד! (לא rar, לא 7z ולא שום תוכנת כיווץ אחרת)
- אין להדפיס אף חלק בתרגיל. קובץ zip שיגיע ללא חלק יבש (קובץ pdf) יגרור ציון 0 על התרגיל כולו.
- הסימולציה תעבור בדיקה אוטומטית. אנו נריץ סימולציה על הקבצים שתספקו ולכן חשוב להשתמש באותם module-ים (שמות ו-port-ים) המופיעים בתרגיל. אין לשנות את הקלטים והפלטים שלהם, ואין להוסיף להם פרמטרים.
- עליכם לעקוב אחרי הודעות אשר מתפרסמות באתר הקורס, הודעות אלו מחייבות.
- כל שאלה על התרגיל אשר איננה בקשה אישית צריכה להישאל דרך הפורום באתר הקורס.
- אנא בדקו היטב את הקבצים לפני ההגשה. טענות מסוג "אבל בבית זה עבד נכון" לא תתקבלנה. מומלץ לבדוק את תקינות הקוד על סביבה "נקייה" (למחוק את הספרייה work, ליצור אותה מחדש ולקמפל לתוכה מחדש את כל קבצי הקוד).

  קוד שלא מתקמפל יגרור הורדת נקודות מלאה של הסעיף.
- **שימו לב ל-Warnings שמתקבלים כפלט של הסימולטור. אזהרות יכולות להופיע גם בשלב ה-vlog וגם בשלב ה-vsim. נקודות יורדו על Warnings חמורים.**
- הגשה באיחור ללא אישור: על כל יום איחור יורדו 5 נקודות. הגשות באיחור מותרות עד שבוע מתאריך ההגשה.

6. המלצות לתרגיל:

6.1. מומלץ לערוך את קבצי הקוד בתוכנת Notepad++.

6.2. חישבו ותכננו כל module לפני שאתם ניגשים לכתוב את הקוד שלו. ככל שתקדישו יותר זמן לתכנון מוקדם, כך שלב המימוש יהיה קל יותר.

6.3. אל תחכו לרגע האחרון. בד"כ שלב ה-debug ארוך ומסובך יותר משלב כתיבת הקוד.