



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: אלון ורטהיימר `zsoltalonw@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 אל המתרגלים: 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, אלון**, 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).

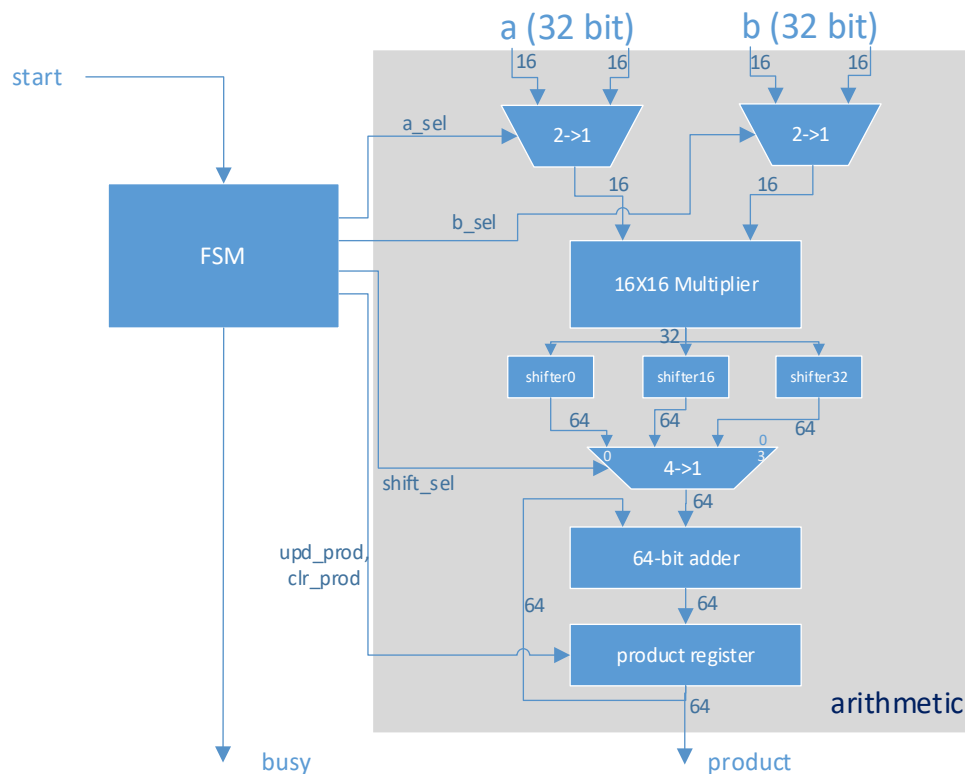


חלק א' - מימוש העלאה בחזקה באמצעות תכן לוגי ומכונת מצבים.

2. חלק יבש

מימוש מכפל 32x32 באמצעות מכפל 16x16 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, 16x16 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 16 סיביות:



הקופסאות הפנימיות בתוך היחידה האריתמטית (mux-ים, מכפל וכו') הם אינם modules בפני עצמם, אלא ממומשות בתוך ה-Module של היחידה האריתמטית. לא חובה להפריד כל קופסה ל- procedural block/assign/module instantiation נפרד.

ה-Shifter-ים מבצעים פעולת shift left, כפי שנלמדה בתרגולים ובסדנאות. למשל, הבלוק shifter16 מבצע shift לקלט 16 מקומות שמאלה.



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 (למעט ה-reset) וכן כי המספרים המוכפלים בייצוג unsigned.

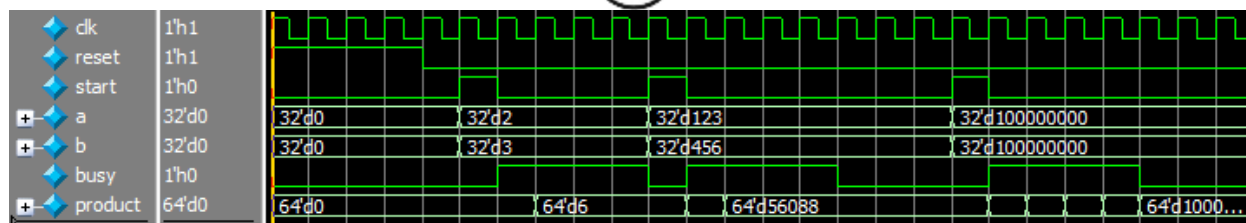
2.1. תכנון מכונת מצבים (FSM) מסוג Mealy השולטת על פעולת הכפל בתכן הנתון בשרטוט. שימו לב שכדי שמכונת מצבים תהיה מסוג Mealy, מספיק שלפחות במצב אחד, אחת היציאות תהיה תלויה צירופית באחת הכניסות. ה-ports של המכונה מתוארים בטבלה הבאה:



שם	כיוון	גודל [ביטים]	תאור
clk	כניסה	1	שעון המערכת בעל זמן מחזור של 2 יחידות זמן
reset	כניסה	1	Reset אסינכרוני, פעיל ב-'1'. מאפס את כל ה-FFs במערכת (גם באמצע חישוב).
start	כניסה	1	כאשר המכונה אינה עובדת וכניסה זו ב-'1', המכונה מתחילה לעבוד; כל עוד המכונה עובדת, כלומר כבר מתבצעת פעולת כפל, 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	1	Select לבורר הבוחר את אחת משתי המילים (באורך 16 ביט) של הכניסה a
b_sel	יציאה	1	Select לבורר הבוחר את אחת משתי המילים (באורך 16 ביט) של הכניסה b
shift_sel	יציאה	2	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 '1' clr_prod=, ה-product register מתאפס בלי תלות בערך של upd_prod

שימו לב שכל הסיגנלים הנ"ל הם סינכרוניים (כלומר, ניתן להניח כי הם משתנים רק בעליית השעון, ו-FF-ים מגיבים לשינויים בהם רק בעליית השעון העוקבת) למעט סיגנל reset שהוא אסינכרוני (יכול לעלות או לרדת בכל נקודת זמן בסימולציה, ויש להגיב לשינויים בו מיידית). כמו כן, ניתן להניח כי כל הכניסות יקבלו ערכים לוגיים (כלומר, רק '0' או '1', ולא 'x' או 'z'). לשימושכם, מצורפת דיאגרמת גלים בה ניתן לראות דוגמה לפעולת המכפל הנדרשת. בדוגמה זו המכפל מבצע שלוש פעולות כפל ברצף:

- 2×3
- 123×456
- $100,000,000 \times 100,000,000$



היעזרו בדיאגרמה על-מנת להבין את דרך פעולת המכונה (למשל, מתי busy צריך לעלות ביחס לכניסת start-וכן הלאה).

בתכנון מכונת המצבים יש לוודא שהמכפל תומך בביצוע מספר לא מוגבל של פעולות כפל ללא reset ביניהן. יתכן גם כי פעולות הכפל יהיו סמוכות, כמודגם בדיאגרמה בפעולת הכפל של 123×456 הסמוכה לפעולת הכפל הקודמת של 2×3 . כפי שהוסבר בטבלה, במידה וסיגנל ה-start עולה מוקדם יותר מהמודגם בדיאגרמה הנ"ל (כלומר, במידה ו: '1' start=באותו מחזור שעון שבו '1' busy), יש להתעלם ממנו ולהמשיך בחישוב.

שימו לב כי התוצאה הזמנית (המצטברת) של הכפל מופיעה במוצא product בעת החישוב. התוצאה הסופית של פעולת הכפל כולה צריכה להישמר במוצא product עד קבלת start נוסף. ניתן להניח כי בעת פעולת כפל מסוימת (החל מעליית הכניסה start ועל לירידת המוצא busy), הכניסות a ו-b נותרות קבועות.

יש לתכנן מכונת מצבים מצומצמת עם 5 מצבים.

ציירו את דיאגרמת המצבים של המכונה בעזרת תוכנת שרטוט, או ע"י כתיבה ידנית וסריקה (על השרטוט להיות ברור, על שרטוט שאינו ברור יורדו נקודות). אין צורך להציג טבלת מצבים או שרטוט של החומרה הנוצרת.

כמה מחזורי שעון לוקחת פעולת הכפל (כלומר, במשך כמה מחזורי שעון המוצא busy מקבל את הערך '1')?

2.2. כאשר מילה מסוימת (16 סיביות) של אחד הגורמים במכפלה שווה ל-0, תוצאת כפל של המילה הזו בכל מספר אחר היא גם 0. ניתן לנצל את התכונה הזו כדי לייעל את תהליך ההכפלה על ידי זיהוי של המילים האלו מראש ודילוג על שלבי החישוב הרלוונטיים במכונה.

הציעו שינוי במכונת המצבים המקורית כך שהפעולה תהיה מהירה יותר אם ה-Most Significant Halfword (סיביות 16 עד 31 כולל) בכניסה A ו/או ה-Most Significant Halfword (סיביות 16 עד 31 כולל) בכניסה B שווים ל-0. ההגבלה על מספר המצבים מהסעיף הקודם נותרת גם כאן. בנוסף ל-ports של מכונת המצבים המקורית, למכונת המצבים החדשה יש שתי כניסות נוספות:

שם	כיוון	גודל [ביטים]	תאור
a_msw_is_0	כניסה	1	'1' כאשר ה-Most Significant Halfword של A שווה ל-0, '0' אחרת
b_msw_is_0	כניסה	1	'1' כאשר ה-Most Significant Halfword 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. שימו לב שהסיגנלים a_msw_is_0, b_msw_is_0 הם כניסות, ניתן להניח כי ערכן נקבע לפי הסיגנלים a ו-b מחוץ ל-FSM ובסעיף זה לא מממשים אותם.

ציירו את דיאגרמת המצבים החדשה בעזרת תכנת שרטוט, או ע"י כתיבה ידנית וסריקה (על השרטוט להיות ברור, על שרטוט שאינו ברור יורדו נקודות). אין צורך להציג טבלת מצבים או שרטוט של החומרה



הנוצרת.

כמה מחזורי שעון לוקחת פעולת הכפל כעת? תחת אילו 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 שיועד 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 שתוצא, ושהמעבד תומך בפעולות אריתמטיות בסיסיות כגון חיבור או shift. 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), כאשר ניתן להניח כי סיבוכיות כל פעולה אריתמטית (כולל כפל 16x8) היא $O(1)$.



נאסוף הכל ביחד

- 2.4. כעת נרצה לקחת את כל מה שעשינו ולממש רכיב המבצע העלאה בחזקה חיובית. לרכיב נקרא power ויהיו לו כניסות שעון reset כרגיל, בנוסף כניסת בסיס וכניסת חזקה. הבסיס שהוא המספר שאנחנו מעלים בחזקה יכול להיות מספר בין 0 ל-15.
- החזקה יכולה להיות מספר חיובי בין 1 ל-8.
 - לרכיב כניסת start הפועלת בצורה דומה לזו בסעיפים קודמים.
 - יציאת המערכת היא מספר ברוחב 64 סיביות.
 - יציאה נוספת בשם busy ברוחב ביט אחד, אשר תעלה ל-1 כל עוד המערכת פועלת ותורד ל-0 כאשר המערכת סיימה לחשב והיציאה תקינה.
- א. עבור מימוש הרכיב השתמשו במכפל המהיר. הסבירו למה ביצועיו יהיו משמעותית יותר טובים משל המכפל הרגיל (שימו לב שמותר להשתמש רק ברכיב מכפל אחד).
חשבו את ה-latency של פעולת החישוב עבור מכפל מהיר ועבור מכפל רגיל.
- ב. על מנת "לזכור" כמה פעמים עוד נשאר לנו להכפיל, נשתמש ברגיסטר שישמור לנו את מספר פעולות הכפל שנותרו. בנוסף, נשתמש ברגיסטר נוסף שישמור את התוצאה הזמנית שלנו.
הוסיפו שרטוט קוביות (כמו שניתן לכם בתחילת השאלה) של מימוש הרכיב והסבירו במילים את פעילותו.





3. חלק רטוב

לפני תחילת העבודה על החלק הרטוב, מומלץ לצפות בוידאו הבא. הוידאו מדגים טעויות נפוצות בכתיבת קוד הממש 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 המממש את המכונה שתכנתם בסעיף 2.1.
- 3.2. בקובץ mult32x32_arith.sv כתבו קוד SystemVerilog המממש את היחידה האריתמטית המתוארת בשרטוט. שימו לב שמכונת המצבים אינה חלק מיחידה זו. ה-ports 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'. מאפס את כל ה-FFs במערכת (גם באמצע חישוב).
a	כניסה	32	האופרנד A להכפלה
b	כניסה	32	האופרנד B להכפלה
a_sel	כניסה	1	Select לבורר הבוחר את אחת משתי המילים (באורך 16 ביט) של הכניסה a
b_sel	כניסה	1	Select לבורר הבוחר את אחת משתי המילים (באורך 16 ביט) של הכניסה b
shift_sel	כניסה	2	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 16x16, בפעולת החיבור (האופרטור: +) ובשאר האופרטורים שנלמדו בסדנאות.

- 3.3. בקובץ mult32x32.sv כתבו קוד SystemVerilog המממש את ההיררכיה העליונה של המכפל ע"י יצירת שתי היחידות הקודמות שמימשותם 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 הזרות (כמספרים עשרוניים, כלומר בסיס דצימלי) של שני הסטודנטים בזוג.



- על ה-testbench לייצר אות שעון תקין, המתחיל בערך '1' בתחילת הסימולציה ומשנה את מצבו כל יחידת זמן אחת.
- כמו-כן, על ה-testbench לבצע את רצף הפעולות הבא:
- אתחול reset בערך '1', אתחול start בערך '0', אתחול a ו-b באפסים
 - המתנה של 4 מחזורי שעון
 - הורדת reset לערך '0'
 - המתנה של מחזור שעון אחד
 - הצבת הכניסות a ו-b בערכי מספרי הזהות. בנוסף, 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 מכונית 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 מכונית המצבים. במקרה הצורך, ניתן לבצע zoom-in ולצרף את הדיאגרמה במספר צילומי מסך.
 - שימו לב: התכן יבדק 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 בבדיקות 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 לגרום 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 : Unsigned.
- 3.5. בקובץ mult32x32_fast_fsm.sv כתבו קוד SystemVerilog המממש את המכונה שתכנתם 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_msw_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 שעון תקין, המתחיל בערך '1' בתחילת הסימולציה ומשנה את מצבו כל יחידת זמן אחת.
- כמו-כן, על ה-testbench לבצע את רצף הפעולות הבא:
- אתחול reset בערך '1', אתחול start בערך '0', אתחול a ו-b באפסים
 - המתנה של 4 מחזורי שעון
 - הורדת reset לערך '0'
 - המתנה של מחזור שעון אחד
 - הצבת הכניסות a ו-b בערכי מספרי הזהות. בנוסף, הצבת '1' ב-start למשך מחזור שעון אחד
 - המתנה להתייצבות היציאה (ירידה של האות busy)



- המתנה של מחזור שעון אחד
- הצבת הכניסות a ו-b בערכי מספרי הזהות עם שני הבתים העליונים מאופסים. בנוסף, 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 מכוונת 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 מכוונת 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 בבדיקות של מקרים
נוספים).

3.8. בקובץ power_sv ממשו את הרכיב power המבצע העלאה בחזקה. יש לממש את היחידה
אותה כתבתם בסעיף 2.4. שימו לב שחלק מהשורות בקוד ניתנו לכם.



3.9. בקובץ power_tb_sv כתבו testbench עבור יחידת ה-power שבניתם. מטרת ה-testbench היא
לוודא נכונות לוגית של התכן שנבנה.
יש לבדוק את התכן בשני מקרים: בסיס וחזקה בעלי ספרה אחת, בסיס בעל שתי ספרות וחזקה בעלת
ספרה אחת.
על ה-testbench לייצר את שעון תקין, המתחיל בערך '1' בתחילת הסימולציה ומשנה את מצבו כל
יחידת זמן אחת.
כמו-כן, על ה-testbench לבצע את רצף הפעולות הבא:
- אתחול reset בערך '1', אתחול start בערך '0', אתחול a ו-b באפסים
- המתנה של 4 מחזורי שעון
- הורדת reset לערך '0'
- המתנה של מחזור שעון אחד
- הצבת הכניסות a ו-b בערכים אותם בחרתם בהתאם לתרחיש. בנוסף, הצבת '1' ב-start למשך
מחזור שעון אחד
- המתנה להתייצבות היציאה (ירידה של האות busy)
- המתנה של מחזור שעון אחד
- הצבת הכניסות a ו-b בערכים הבאים. בנוסף, 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-power.
צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת).
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 בבדיקות של מקרים
נוספים).





4. חלק ב' – מימוש אלגוריתם מיון באמצעות קוד אסמבלי.

בחלק זה עליכם לממש את אלגוריתם המיון bubble sort באמצעות קוד אסמבלי.

לנוחיותכם, אלגוריתם המיון מצורף:

```
void bubbleSort(int arr[], int N)
{
    int i, j;
    for (i = 0; i < N-1; i++)

        // Last i elements are already in place
        for (j = 0; j < N-i-1; j++)
            if (arr[j] > arr[j+1])
                swap(&arr[j], &arr[j+1]);
}
```

בקובץ bubblesort.s כתבו קוד אסמבלי של RISC-V המממש מיון עבור מערך בגודל N כלשהו. ניתן לפתוח ולערוך את הקובץ עם notepad++.

בקובץ כבר ממומש אתחול מערך בגודל 10 המתחיל בכתובת 0x10000000 בזיכרון. בסוף ריצת החלק הנתון, ברגיסטר s10 יהיה מוכל כתובת תחילת המערך ורגיסטר s1 יכיל את גודל המערך (10). בנוסף, חלק משורות הקוד נתונות על מנת להקל עליכם. אין חובה להשתמש בקוד הקיים עבור מימוש האלגוריתם.

הריצו את הקוד בסימולטור שבאתר: <http://www.kvakil.me/venus> הוסיפו לחלק היבש צילום מסך של הסימולטור לפני ואחרי הרצת הסימולציה. יש לשים לב שרואים את ערכי הזכרון.



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	10
		3.5	3
		3.6	4
		3.7	10
		3.8	4
		3.9	10
		4	10

Total	41		59
--------------	----	--	----

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):
 - bubblesort.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
 - power.sv
 - power_tb.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.
 - על בני הזוג להירשם לקבוצת הגשה ב-Moodle. אחד מבני הזוג צריך להגיש את הקבצים.
 - 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 יגורר 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.