תרגיל בית רטוב מספר 1

מערכות ספרתיות ומבנה מחשב

קראו היטב את הוראות ההגשה בסעיף 5.

נקודות יורדו למי שלא יבצע את ההוראות במדויק.

אחראי התרגיל: אייל טייטלר ataitler@technion.ac.il

שאלות בקשר לדרישות תרגיל 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 (חישובים תיאורטיים), 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.

בכל מקום בו נדרש לתכנן מימוש יעיל מבחינת מספר הרכיבים, אין צורך להגן מפני hazards.

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: יעקב, אריאל ומייקל 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 הזה נבנה יחידה אריתמטית לוגית (Arithmetic Logic Unit – ALU) פשוטה.

לתרגיל הזה מספר מטרות:

1. התנסות ראשונה עם שפת תיאור חומרה SystemVerilog ועם כלי סימולציית חומרה.
2. בניית מעגלים צירופיים מורכבים מאבני בניין בסיסיות.
3. סימולציה של מעגלים צירופיים עם השהיות.
4. הכרה מקרוב של מרכיב מרכזי בכל מעבד: ALU.

לתרגיל מצורף וידאו קצר המסייע בהבנת אופן השימוש בספריית השערים. צפו בוידאו הבא לפני תחילת העבודה על התרגיל:

https://www.youtube.com/watch?v=lEtfxCrT9Ho

בתרגיל הזה תשתמשו במודל המבני של שפת SystemVerilog, קרי עליכם לבנות את המעגלים באמצעות שערים לוגיים בסיסיים על ידי הצבתם כ-module instances. בתיקיית libcells תמצאו קבצים שמתארים את השערים הלוגיים הבאים:

1. שער NOT עם כניסה אחת
2. שער OR עם 2 כניסות
3. שער XOR עם 2 כניסות

עליכם להשתמש בשערים האלו בלבד למימוש המעגלים. אין לשנות את התכולה של הקבצים בספרייה. בקובץ libtest.sv נמצאת דוגמה לשימוש בתאי הספרייה. עברו היטב על הדוגמה, שימו לב לשיטה של הגדרת השהיות השערים. ההשהיות ניתנות לשערים ע"י שינוי של שני פרמטרים: Tpdhl ו-Tpdlh. לדוגמה, בשורת הקוד שלהלן ירידה (שינוי מ-1 ל-0) של אות a תגרור עלייה (שינוי מ-0 ל-1) של אות z בעוד 10 יחידות זמן, וכן עלייה של אות a תגרור ירידה של אות z בעוד 5 יחידות זמן:

```
OR2 #(.Tpdlh(10), .Tpdhl(5)) or2_inst (.Z(z), .A(a), .B(b));
```

2. חלק יבש

2.1. הציעו מימוש לבורר 1<-2 עם כניסות d0, d1 ו-sel (ללא כניסת enable) ויציאה z תוך שימוש בשערים שקיימים בספרייה בלבד (לא חובה להשתמש בכל השערים שבספרייה) וללא שימוש בקבועים (0,1).
הציגו טבלת אמת, כתבו ביטוי ל-z כפונקציה של הכניסות והציגו שרטוט של התכן עם השערים הלוגיים.
על-גבי השרטוט מספרו את כל השערים באופן הבא: g1, g2, g3… בסדר לבחירתכם.
יש לתכנן מימוש יעיל מבחינת כמות השערים. כל פתרון עם 8 שערים או פחות יקבל את מלוא הנקודות.
חשבו את כל ההשהיות של כל המסלולים מכל הכניסות למוצא z בטבלה הבאה:

Path	d0	d1	sel	Tpd
d0→g1→…→z	0→1	0	0	
…				

כלומר, עבור כל שינוי (עליה או ירידה) של כניסה מסוימת, ועבור כל המסלולים הקיימים מכניסה זו ליציאה, ועבור כל המצבים הקבועים האפשריים של שאר הכניסות, יש לחשב את ההשהיה מזמן שינוי הכניסה עד לזמן שינוי המוצא, בהנחה שהמוצא אכן משתנה ונותר יציב כתוצאה משינוי הכניסה הנ"ל.
אין צורך להוסיף לטבלה שורות עבור שינויים ומסלולים שאינם גורמים לשינוי ביציאה או שגורמים רק ל-hazard ללא שינוי יציב.

השתמשו בערכים הבאים עבור ההשהיות של השערים:

	t_{PDLH}	t_{PDHL}
NOT	B	C
OR2	D	E
XOR2	F	G

כאשר: ABCDEFGHI = ת.ז. של אחת/אחד הסטודנטיות/ים. אם אחת הספרות שווה ל-0, השתמשו בערך 10 במקומה.

2.2. הציעו מימוש לבורר 1<-4 עם כניסות d0, d1, d2, d3 ו-sel[1:0] ויציאת z באמצעות רכיבי בוררי 1<-2 שבניתם בסעיף הקודם בלבד. יש לתכנן מימוש יעיל מבחינת כמות הרכיבים. כל פתרון עם 3 או פחות רכיבי בורר 1<-2 יקבל את מלוא הנקודות.
שימו לב ש-sel[0] הוא ה-LSB ו-sel[1] הוא ה-MSB בוקטור, כלומר לדוגמה, sel=01 (כאשר sel[0]=1, sel[1]=0) יבחר את d1.
בחרו את אחת הכניסות d0-d3 או sel[0] או sel[1], ושינוי כלשהו עבורה (1→0 או 0→1) וכן מצב קבוע עבור כל שאר הכניסות, אשר גורמים לשינוי יציב ביציאה. חשבו את ההשהיה המקסימלית עבור שינוי זה (כלומר, ההשהיה המקסימלית מבין כל המסלולים מהכניסה המשתנה ליציאה). חשבו גם את ההשהיה המקסימלית עבור השינוי ההפוך (אם בחרתם קודם 1→0, חשבו עבור 0→1, ולהפך).

2.3. הציעו מימוש ליחידת Full Adder/Subtractor עם כניסות a, b, cin, a_ns ויציאות s ו-cout. היחידה תבצע את הפעולה הבאה:

$$cout, s = \begin{cases} a + b + cin & , \quad a\_ns = 1 \\ a - b - cin & , \quad a\_ns = 0 \end{cases}$$

טבלת האמת של הרכיב הינה:

a	b	cin	a_ns	cout	s
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	1	1
1	1	0	0	0	0
0	0	1	0	1	1
1	0	1	0	0	0
0	1	1	0	1	0
1	1	1	0	1	1
0	0	0	1	0	0
1	0	0	1	0	1
0	1	0	1	0	1
1	1	0	1	1	0
0	0	1	1	0	1
1	0	1	1	1	0
0	1	1	1	1	0
1	1	1	1	1	1

כמו קודם, השתמשו ברכיבים מהספרייה בלבד ו<u>ללא</u> שימוש בקבועים (0,1), עם ההשהיות מסעיף 2.1. בסעיף זה אין להשתמש באף רכיב אחר (כולל רכיבים שמימשתם בסעיפים הקודמים) לצורך המימוש. יש לתכנן מימוש יעיל מבחינת כמות השערים. כל פתרון עם 25 שערים או פחות יקבל את מלוא הנקודות.

בחרו את אחת היציאות cout, s, את אחת הכניסות a, b, cin, a_ns, שינוי כלשהו עבורה (1→0 או 0→1) וכן מצב קבוע עבור כל שאר הכניסות, אשר גורמים לשינוי יציב ביציאה שבחרתם. חשבו את ההשהיה המקסימלית עבור שינוי זה (כלומר, ההשהיה המקסימלית מבין כל המסלולים מהכניסה המשתנה ליציאה). חשבו גם את ההשהיה המקסימלית עבור השינוי ההפוך (אם בחרתם קודם 1→0, חשבו עבור 0→1, ולהפך).

2.4. בסעיף זה נממש יחידה אריתמטית לוגית (ALU) עם כניסות data ברוחב סיבית בודדת. בנוסף, היחידה מקבלת בכניסה גם מילת בקרה op בעלת 2 סיביות שקובעת את פעולת ה-ALU לפי הטבלה הבאה:

פעולת ALU	Operation
NOR (בין A ל-B)	00
XOR (בין A ל-B)	01
חיבור (כמו ב-FAS)	10
חיסור (כמו ב-FAS)	11

הציעו מימוש ליחידת החישוב. ליחידה כניסות a, b, cin ו-op[1:0], ויציאות s ו-cout.

השתמשו ברכיבים הבאים: בורר אחד (1<-2 או 1<-4), Full Adder/Subtractor אחד (שבניתם בסעיפים הקודמים), וכן ברכיבי הספרייה, ללא שימוש בקבועים (0,1).

בפעולות NOR ו-XOR, ערך היציאה cout אינו משנה (don't care).

שימו לב ש-op[0] הוא ה-LSB ו-op[1] הוא ה-MSB. כלומר עבור op=01, כאשר op[0]=1 ו-op[1]=0, מקבלים פעולת XOR.

בחרו את אחת היציאות, את אחת הכניסות, שינוי כלשהו עבורה (1→0 או 0→1) וכן מצב קבוע עבור כל שאר הכניסות, אשר גורמים לשינוי יציב ביציאה שבחרתם. חשבו את ההשהיה המקסימלית עבור שינוי זה (כלומר, ההשהיה המקסימלית מבין כל המסלולים מהכניסה המשתנה ליציאה). חשבו גם את ההשהיה המקסימלית עבור השינוי ההפוך (אם בחרתם קודם 1→0, חשבו עבור 0→1, ולהפך).

2.5. כעת הציעו מימוש ל-ALU בעל כניסות data ברוחב 64 ביט. ליחידה כניסות: a[63:0], b[63:0], cin ו-op[1:0], ויציאות s[63:0] ו-cout.

השתמשו ביחידה מהסעיף הקודם ושערים מהספרייה במידת הצורך.

בחרו את המסלול שגורם להשהיה המקסימלית מכניסה ליציאה (מבין כל האפשרויות הקיימות). חשבו את ההשהיה המקסימלית. שימו לב שכאשר מדברים על מסלול מכניסה ליציאה, הכוונה היא מ-bit בודד של כניסה אל bit בודד של יציאה (כלומר, הכניסה או היציאה הרלוונטיות יכולות להיות גם bit מסוים מתוך vector כניסה או יציאה, אך לא ה-vector כולו). ציינו את ערכי שאר הכניסות בזמן שינוי הכניסה שגורם להשהיות הנ"ל.

3. <u>חלק רטוב</u>

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 mux2.sv ממשו בורר 1<-2 ע"י שימוש ברכיבים מהספרייה שקיבלתם. השתמשו בהשהיות מסעיף 2.1.

3.2. בקובץ mux4.sv ממשו בורר 1<-4 ע"י שימוש ברכיבי mux2 מהסעיף הקודם בלבד. שימו לב ש-sel[0] הוא ה-LSB ו-sel[1] הוא ה-MSB.

3.3. בקובץ mux4_test.sv כתבו testbench עבור המודול mux4 שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן בשני המקרים שנבחרו בסעיף 2.2.
על ה-testbench לבצע את רצף הפעולות הבא:
- הצבת כל הכניסות בערכים המתאימים ו-0 בכניסה הנבחרת
- המתנה להתייצבות היציאה
- שינוי הכניסה מ-0 ל-1
- המתנה להתייצבות היציאה
- שינוי הכניסה מ-1 ל-0
- המתנה להתייצבות היציאה
צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. הראו שההשהיות אכן מתאימות לחישוב התאורטי.
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).

<u>תזכורת – קימפול והרצה ב-ModelSim:</u>

1. נווטו ב-ModelSim לתוך תיקיית התרגיל באמצעות הפקודה cd (ודאו שהנתיב אינו כולל רווחים או תווים בעברית).
2. צרו ספריית work באמצעות הפקודה vlib work. <u>אסור לשנות את שם הספרייה מ-work לשם אחר.</u>
3. קמפלו את כל הקבצים הרלוונטיים באמצעות הפקודה vlog. לדוגמה, על-מנת לבדוק את mux4 יש להריץ:
vlog libcells/NAND2.sv
…
vlog mux2.sv
vlog mux4.sv
vlog mux4_test.sv
שימו לב שאין צורך ב-import/include וכו', כל שצריך לעשות הוא לקמפל את כל הקבצים.
4. הריצו את הסימולציה באמצעות הפקודות: vsim, add wave, run וכו'...

פרטים נוספים על כל הפקודות ניתן למצוא בהוראות של סימולציה 0.

3.4. בקובץ fas.sv ממשו את יחידת ה-Full Adder/Subtractor ע"י שימוש ברכיבים מהספרייה שקיבלתם. השתמשו בהשהיות מסעיף 2.1.

3.5. בקובץ fas_test.sv כתבו testbench עבור המודול fas שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן בשני המקרים שנבחרו בסעיף 2.3.
על ה-testbench לבצע את רצף הפעולות הבא:
- הצבת כל הכניסות בערכים המתאימים ו-0 בכניסה הנבחרת

- המתנה להתייצבות היציאה
- שינוי הכניסה מ-0 ל-1
- המתנה להתייצבות היציאה
- שינוי הכניסה מ-1 ל-0
- המתנה להתייצבות היציאה

צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. הראו שההשהיות אכן מתאימות לחישוב התאורטי.

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).

3.6. בקובץ alu1bit.sv ממשו את היחידה שהגדרתם בסעיף 2.4.
השתמשו בהשהיות מסעיף 2.1.
שימו לב ש-op[0] הוא ה-LSB ו-op[1] הוא ה-MSB.

3.7. בקובץ alu64bit.sv ממשו את ה-ALU שהגדרתם בסעיף 2.5.
השתמשו בהשהיות מסעיף 2.1.
מומלץ להשתמש בפקודת for generate שנלמדה בסדנה.

3.8. בקובץ alu64bit_test.sv כתבו testbench עבור המודול alu64bit שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן עבור שינוי הכניסות שגורם להשהיה הארוכה ביותר עד למוצא, שחושב בסעיף 2.5.
על ה-testbench לבצע את רצף הפעולות הבא:
- הצבת כל הכניסות בערכים המתאימים לפני השינוי הגורם להשהיה הארוכה ביותר
- המתנה להתייצבות היציאה
- שינוי הכניסה הגורם להשהיה הארוכה ביותר
- המתנה להתייצבות היציאה

צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. הראו שההשהיות אכן מתאימות לחישוב התאורטי.

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	5	3.1	5
2.2	5	3.2	5
2.3	5	3.3	10
2.4	5	3.4	10
2.5	5	3.5	10
		3.6	10
		3.7	10
		3.8	15

Total	25		75

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**):
  - alu1bit.sv
  - alu64bit.sv
  - alu64bit_test.sv
  - fas.sv
  - fas_test.sv
  - mux2.sv
  - mux4.sv
  - mux4_test.sv
  - Dry.pdf

  אין להגיש את תיקיית libcells או את קבצי הספרייה הנמצאים בה.
- 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.