תרגיל בית רטוב מספר 1 מערכות ספרתיות ומבנה מחשב

קראו היטב את הוראות ההגשה בסעיף 6

נקודות יורדו למי שלא יבצע במדויק.

מתרגל אחראי לתרגיל: לאוניד עזריאל leonida@tx

שאלות בקשר לתרגיל יש להפנות ללאוניד דרך הפורום. בקשות מיוחדות יש לשלוח ללאוניד במייל

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

חשוב מאד:

אין להגיש שום חלק מודפס – את החלק היבש יש לכתוב במעבד תמלילים ולצרף לחלק הרטוב בפורמט pdf בלבד.

1. הנחיות כלליות

- מסמן שאלות שיש לענות עליהן במסמך (החלק היבש)
- מסמן חלק שיש לבצע בסימולטור. את תוצאת הסימולציה (waveform) יש לצרף לחלק היבש ולהסביר את התוצאות בצורה איכותית. waveform אפשר לקבל בעזרת צילום מסך. ב-waveform יש להכיל את האותות הרלוונטיים לתרגיל (כניסות ויציאות) ובצילום להראות את הקטעים הרלוונטיים בזמן. יש לדאוג שהתמונות תהינה ברורות.

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 Verilog ועם כלי סימולצית חומרה.
2. בניית מעגלים צירופיים מורכבים מאבני בניין בסיסיות.
3. סימולצייה של מעגלים צירופיים עם השהיות.
4. הכרה מקרוב של מרכיב מרכזי בכל מעבד: ALU.

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

1. שער NAND עם 2 כניסות
2. שער OR עם 2 כניסות
3. שער XNOR עם 2 כניסות

עלייכם להשתמש בשערים האלו בלבד למימוש המעגלים. אין לשנות את התכולה של הקבצים בספרייה. בקובץ libtest.v תמצאו דוגמא של הצבות של תאי הספרייה. תעברו היטב על הדוגמא, שימו לב לשיטה של הגדרת השהיות של השערים המוצבים. ההשהיות ניתנות לשערים ע"י שינוי של שני פרמטרים Tpdhl ו-Tpdlh. לדוגמא בשורת הקוד שלהלן ירידה (שינוי מ-1 ל-0) של אות a או b תגרור עלייה (שינוי מ-0 ל-1) של אות z בעוד 10 יחידות זמן, וכן עלייה של אות a או b תגרור ירידה של אות z בעוד 5 יחידות זמן:

```
NAND2 #(.Tpdlh(10), .Tpdhl(5)) nand2_inst ( .Z(z), .A(a), .B(b) );
```

2. מימוש בורר 1->4

2.1. הציעו מימוש לבורר 1->2 עם כניסות S, D0, D1 ויציאה Z תוך שימוש בשערים שקיימים בספרייה בלבד. (כדאי להציג טבלת אמת אפשרית ולכתוב את Z כפונקציה של הכניסות). חישבו את ההשהיות t_{PDHL} ו-t_{PDLH} מכל הכניסות למוצא Z. השתמשו בערכים להלן עבור השהיות של הרכיבים:

שער	t_{PDLH}	t_{PDHL}
NAND2	B	C
OR2	D	E
XNOR2	F	G

כאשר ABCDEFGHI = ת.ז. של אחת\ד מהסטודנטיות\ים בזוג. כאשר הספרה שווה ל-0, השתמשו בהשהיה של 10 יחידות.

2.2. הציעו מימוש לבורר 1->4 עם כניסות S1-0 ,D0, D1, D2, D3 ויציאה Z. עלייכם להשתמש בבורר עם 2 כניסות שבניתם בסעיף הקודם. חישבו את ההשהיות t_{PDHL} ו-t_{PDLH} מכל הכניסות למוצא Z .

2.3. בקובץ mux2.v ממשו בורר 1->2 ע"י שימוש ברכיבים מהספרייה שקיבלתם. השתמשו בהשהיות מסעיף 2.1.

2.4. בקובץ mux4.v ממשו בורר 1->4 ע"י שימוש ברכיבי mux2 מהסעיף הקודם בלבד.

2.5. בקובץ mux4_test.v הציבו את המודול שבניתם והוסיפו סימולצייה שבודקת את ההשהיות הארוכות ביותר בכל המסלולים מכניסות שונות ליציאה (אין צורך לחזור על הסימולצייה עבור מסלולים זהים). ודאו שההשהיות שהתקבלו תואמות את החישוב בסעיף 2.2.
יש לצרף רק את תוצאות הסימולציה לחלק היבש.

3. מימוש רכיב Full Adder/Subtractor

3.1. בדומה לסעיף הקודם הציעו מימוש ליחידת Full Adder/Subtractor עם כניסות *A,B,Cin,A_nS* ויציאות S, Cout תוך שימוש בשערים שקיימים בספרייה בלבד. היחידה תבצע את הפעולה הבאה:

$$Cout, S = \begin{cases} A + B + Cin & \text{A\_}nS = 1 \\ A - B - Cin & \text{A\_}nS = 0 \end{cases}$$

כמו קודם, השתמשו ברכיבים מהספרייה עם ההשהיות מסעיף 2.1. תמצאו את המסלולים הארוכים ביותר מכל כניסה לכל יציאה ותראו את החישוב. מהו המסלול הארוך ביותר?

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

3.3. בקובץ fas_test.v הציבו את המודול שבניתם והוסיפו סימולצייה שבודקת את המסלול הארוך ביותר. ודאו שהתוצאה שהתקבלה תואמת את החישוב בסעיף 3.1.
יש לצרף רק את תוצאות הסימולציה לחלק היבש.

4. <u>מימוש רכיב ALU</u>

כאן נממש יחידה אריתמטית לוגית (ALU) עם כניסות ברוחב 64 סיביות. היחידה מקבלת בכניסה מילת בקרה בעלת 2 סיביות op שקובעת את פעולתה לפי הטבלה להלן:

פעולת ALU	Operation
XOR	00
XNOR	01
חיבור	10
חיסור	11

4.1. הציעו מימוש ליחידת סיבית בודדת של ה-ALU. ליחידה כניסות *A,B,Cin,op* ויציאות *Cout,S*.
השתמשו ברכיבים שבניתם בסעיפים הקודמים (עדיפות ראשונה) וכן ברכיבי הספרייה. חשבו השהיות מכל כניסה לכל יציאה. בונוס יינתן למי שימצא מימוש יעיל (מבחינת מספר שערים). נקודות יורדו למי שיציג מימוש מאד לא יעיל.

4.2. כעת הציעו מימוש ל-ALU עם 64 סיביות תוך שימוש ביחידה מהסעיף הקודם ושערים מהספרייה במידת הצורך. ציינו את **<u>כל</u>** המסלולים הקריטיים (הארוכים ביותר) האפשריים בהנחה שההשהיות לא ידועות. תקפידו לבחון את כל האפשרויות, נקודות יורדו למי שלא יציג את כולם. כעת מצאו את המסלול הקריטי בהתבסס על החישוב מהסעיף הקודם. הראו איזה שינוי בערכי הכניסות מקיים את המסלול הארוך ביותר.

4.3. בקובץ alu1bit.v ממשו את היחידה שהגדרתם בסעיף 4.1 ע"י שימוש ברכיבים מהספרייה שקיבלתם. השתמשו בהשהיות מסעיף 2.1.

4.4. בקובץ alu64bit.v ממשו את ה-ALU שהגדרתם בסעיף 4.2 ע"י שימוש ברכיבי alu1bit והרכיבים מהספרייה שקיבלתם. השתמשו בהשהיות מסעיף 2.1. מומלץ להשתמש בפקודת generate. ראו סעיף 7.6.

4.5. בקובץ alu64bit_test.v הציבו את המודול שבניתם. בצעו סימולצייה שבודקת את המסלול הארוך ביותר .
יש לצרף רק את תוצאות הסימולציה לחלק היבש.

5. חלוקת הציון

Sect	Grade	Sect	Grade	Sect	Grade
2.1	5	3.1	5	4.1	10
2.2	5	3.2	10	4.2	5
2.3	5	3.3	10	4.3	10
2.4	5			4.4	10
2.5	10			4.5	10

Total	30		25		45

6. הוראות הגשה

6.1. ***ההגשה בזוגות בלבד***. ניתן לחפש בני זוג דרך פורום חיפוש שותפים. הגשה ללא בן זוג לא אישור מראש יגרור מאושרת מראש תגרור הורדה בציון של 10 נקודות

6.2. יש לארוז בקובץ zip בשם <id>.zip את כל קבצי הקוד וקובץ הטקסט של החלק היבש, כאשר id זהו מס' ת.ז. מלא של אחד מבני הזוג.

הערות:

- בתחילת כל תרגיל יש לכתוב שמות ו ת.ז. של כל אחד מהסטודנטים בטבלה כמו בדוגמה:

שם 1	123456789
שם 2	987654321

- יש ליצור קובץ zip בודד עבור כל הקבצים. (ולא קובץ נפרד לכל אחד **וישירות תחת ה zip** **וללא תתי תיקיות למעט תיקיית libcells**).
- Zip - זה לא rar, לא 7z ולא שום תוכנת כיווץ אחרת.
- אין להדפיס אף חלק בתרגיל. קובץ zip שיגיע ללא חלק יבש (קובץ טקסט) יגרור ציון 0 על התרגיל כולו.

6.3. הסימולציה תעבור בדיקה אוטומטית. אנו נריץ סימולציה על הקבצים שתספקו ולכן חשוב כי תשתמשו באותם שמות module המופיעים בתרגיל.

6.4. הכניסו את הקוד שלכם אך ורק במקום המסומן בתוך הקבצים בצורה הבאה:

```
// Put your code here
// ------------------


// End of your code

endmodule
```

אין לשנות את הקבצים במקומות אחרים!

יש להשאיר ללא שינוי את שמות הקבצים, את שמות ה modules -ואת שמות הפתחים (port - ים).

6.5. עליכם לעקוב אחרי הודעות אשר מתפרסמות באתר הקורס, הודעות אילו מחייבות.
כל שאלה על התרגיל אשר איננה בקשה אישית צריכה להישאל דרך הפורום באתר הקורס.

6.6. אנא בידקו היטב את הקבצים לפני ההגשה. טענות מסוג "אבל בבית זה עבד נכון" לא תתקבלנה.
קוד שלא מתקמפל יגרור הורדת נקודות מלאה של הסעיף

6.7. שימו לב ל-Warnings שמתקבלים כפלט של הסימולטור. נקודות יורדו על Warnings חמורים.

7. המלצות לתרגיל:

7.1. השתמשו ב -editor של modelsim או ב ++notepad או ב-vim

7.2. מומלץ להסתמך על קובץ הבדיקה שניתן לכם עבור שאר הסעיפים .לא מומלץ לעבוד עם force

7.3. קודם חישבו ותכננו את המערכת ורק אח״כ התחילו לכתוב קוד .ככל שתקדישו יותר זמן לתכנון מוקדם - שלב המימוש יהיה קל יותר .

7.4. אל תחכו לרגע האחרון ,שלב ה- debugging בד״כ ארוך ומסובך יותר משלב כתיבת הקוד.

7.5. לפני הגשה מומלץ ליצור תיקייה חדשה (נקייה) להעביר לשם את קבצי ה-Verilog שלכם, ליצור את הפרוייקט מחדש ולהריץ שוב. בצורה זו תוכלו לוודא כי אכן הרצתם את הקבצים הכי עדכניים שלכם (הקבצים אותם אתם מגישים) ולא גירסה ישנה שקומפלה מזמן.

7.6. מומלץ להשתמש בפקודת generate בשאלה 4. תוכלו למצוא את ההסבר לדוגמא בקישורים להלן:

https://www.sutherland-hdl.com/papers/2001-SNUG-paper_Verilog-2000_standard.pdf

http://1sutherland.com/papers/2001-SNUG-presentation_Verilog-2000_standard.pdf