**הטכניון - מכון טכנולוגי לישראל**

**הפקולטה להנדסת חשמל**



מעבדה 1א'

044157

ניסוי SV1

שאלות ודוח הכנה

גרסה 2.57

קיץ 2020

על פי חוברות של עמוס זסלבסקי, 2009

|  |  |
| --- | --- |
| תאריך כתיבת הדו"ח |  |
| שם המדריך |  |

|  |  |  |
| --- | --- | --- |
| סטודנט | שם פרטי | שם משפחה |
| 1 |  |  |
| 2 |  |  |

תוכן עניינים

[1 פתיחת ארכיב 3](#_Toc33538109)

[2 תרגיל תכנון MUX בשיטות שונות 3](#_Toc33538110)

[2.1 מימוש MUX באמצעות IF – קומבינטורי 3](#_Toc33538111)

[2.2 מימושMUX באמצעות CASE – קומבינטורי 6](#_Toc33538112)

[3 תרגיל תכנון MUX הירארכי 6](#_Toc33538113)

[4 מונה עולה סינכרוני 7](#_Toc33538114)

[5 מונה סינכרוני עם קפיצות 8](#_Toc33538115)

[6 תצוגת 7Segment עם הדלקה וכיבוי מלאים 9](#_Toc33538116)

[7 גיבוי העבודה 11](#_Toc33538117)

**הערות חשובות**:

1. בכל התרגילים הבאים השפה לכתיבת הקוד היא **System Verilog או בקיצור SV**
2. בכתיבת הקוד חובה להשתמש בשמות המודולים, הכניסות והיציאות המופיעים בהגדרת התרגילים
3. יש לתת **שמות קבצים ותיקיות באנגלית בלבד וללא רווחים** וללא מקף **-**
4. שם הקובץ צריך להיות כשם המודול
5. תמיד יש להגדיר את הקובץ שעליו עובדים **כהירארכיה עליונה או בקיצור כ- TOP**
6. יש להשלים את הקוד שלך לפי הדרישות בקבצי השלד הנתונים במקומות המסומנים ב- **// fill your code here**
7. בסוף התהליך יש להעתיק את הקוד בצורה **קריאה** דרך **ה-NOTEPAD++** לקובץ זה במקומות המסומנים לכך
8. בכמה מקבצי השלד הנתונים היה צורך לסגור חלק של הקוד כדי לעבור קומפילציה. לכן לפני שמתחילים לכתוב קוד יש להסיר את ההערות המסומנות ב-

**/\* $$$$$$ remove to fill**

ולהשלים את הקוד שלכם במקומות המסומנים ב-

**//fill your code here**

# פתיחת ארכיב

הורד מהמודל קובץ ארכיב של המעבדה ופתח אותו לפרויקט בדיסק שלך.

ודא תכולת קבצים כזו:



הפרויקט שתתחיל אותו כעת בעבודת ההכנה תמשיך אותו במעבדה.

# תרגיל תכנון MUX בשיטות שונות

שים לב! לכתיבת קוד בתרגיל זה העזר בקבצים הנתונים לך במודל.

## מימוש MUX באמצעות IF – קומבינטורי

השלם את הקוד במודול בשם mux\_4to1\_if. לשם כך פתח את הקובץ והפוך אותו ל- TOP. השלם את הקוד שלך לפי הדרישות להלן במקומות המסומנים:

// fill your code here.

כתוב את הקוד שמתאר את הרכיב באמצעות **התניית** **if** .

השתמש רק בהשמותBLOCKING . =

בצע אנליזה (**Analysis & Elaboration**) לתכן ותקן שגיאות סינטקס אם ישנן כאלה.

Module interface

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Direction | |
| **data\_in[3:0]** | logic | **input** |
| sel[1:0] | logic | **input** |
| outd | logic | **output** |

Truth table

|  |  |  |  |
| --- | --- | --- | --- |
| outd | sel[1:0] | **data\_in[3:0]** | |
| data\_in[0] | 00 |  |
| data\_in[1] | 01 |  |
| data\_in[2] | 10 |  |
| data\_in[3] | 11 |  |

**תזכורת לשלבי הקומפילציה השונים** בקוורטוס והשימושים שלהם. כפי שהוסבר כל שלב יותר מתקדם מקודמו ומכיל את השלבים הקודמים, לכן גם אורך יותר זמן.

* **בשלב ההרחבה (Elaboration) – אלבורציה**,התכן נבדק מבחינה סינטקטית וסמנטית, ספציפית לשפת התכנות בה כותבים.
* **בשלב האנליזה (Analysis)** נבדק התכנון ההירארכי של התכן ונבנית טבלת קשרים בין האלמנטים השונים, עדיין כבלוקים ב- RTL (Register-Transfer Level).
* **בשלב הסינטזה (Synthesis)** הקוד הנתון מתורגם למשאבי החומרה בתוך ה-FPGA ברמת השערים הלוגיים מהם מורכבת החומרה. בשלב זה גם נוצרים ה- netlists שמאפשרים לבצע סימולציה (פונקציונלית) **בקומפילציה מלאה (Compilation)** מתבצעים כל השלבים הנוספים, הכוללים בין היתר תכנון הצורה והחיבוריות של החומרה, ניתוח זמנים, וגם הפקת קובץ הצריבה, הדרוש לשם צריבת התכן לכרטיס.

בהתאם לשלב הפיתוח יש להשתמש בפעולה המתאימה **זה חוסך המון זמן קומפילציה**!

* לפני הרצת **סימולציה ברמת השערים** יש לבצע  **Analysis & Synthesis.**
* לבדיקת **נכונות הקוד ויצירת יצוג RTL** השתמש ב- **Analysis & Elaboration**
* לפני **צריבת התכן לכרטיס** חייבים להריץ קומפילציה מלאה **Compilation**

**בסרגל הכלים בחר**: 

אחרי אנליזה מוצלחת (**Analysis & Elaboration**) הוסף את הקוד שלך לדו"ח.

**קוד SV – לאחר אנליזה מוצלחת**

על מנת לבדוק באופן חזותי/גרפי את תכנון המודול שכתבת הצג את המימוש שלך כ- RTL VIEW (Tools -> Netlist Viewers -> RTL Viewer) והוסף אותו לדו"ח (היעזר גם ב- Quartus Cook Book).

**הצג את המימוש כ- RTL VIEW**

## מימושMUX באמצעות CASE – קומבינטורי

פתח את הקובץ בשם mux\_4to1\_case וקבע אותו כ- TOP. המודול הינו Multiplexer

Module interface

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Direction | |
| **data\_in[3:0]** | logic | **input** |
| sel[1:0] | logic | **input** |
| outd | logic | **output** |

Truth table

|  |  |  |  |
| --- | --- | --- | --- |
| outd | sel[1:0] | **data\_in[3:0]** | |
| data\_in[0] | 00 |  |
| data\_in[1] | 01 |  |
| data\_in[2] | 10 |  |
| data\_in[3] | 11 |  |

השלם את הקוד שמתאר את הרכיב באמצעות **התניית** **CASE** **לא מסונכרנת שעון**.

השתמש רק בהשמות BLOCKING.

בצע אנליזה והוסף את הקוד ואת המימוש כ- RTL VIEW לדו"ח.

**קוד SV – לאחר אנליזה מוצלחת**

**הצג את המימוש כ- RTL VIEW**

# תרגיל תכנון MUX הירארכי

השלם את הקוד במודול בשם: mux\_16to1. המודול מממשMultiplexer בעל 16 כניסות מידע din (וקטרו באורך 16), 4 כניסות בחירה sel(וקטור באורך 4) ויציאת outd של ביט אחד.

המערכת מורכבת מ- 5 רכיבי 1 <= 4 Multiplexer מהתרגיל הקודם.

ממש תכן הירארכי ב- Verilog על ידי שימוש במימוש עם case ובצוע instantiation (הפעלת המודול).

העזר בחומר העזר Verilog workshop 1 .

בצע אנליזה והוסף את הקוד ואת המימוש כ- RTL VIEW לדו"ח.

**קוד SV – לאחר אנליזה מוצלחת**

**הצג את המימוש כ- RTL VIEW**

# מונה עולה סינכרוני

השלם את הקוד במודול של מונה בינארי סינכרוני עולה **simple\_up\_counter**.

Module interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name |  | Type | Direction | |
| **clk** |  | logic | **input** |
| resetN |  | logic | **input** |
| count | [3:0] | logic | **output** |

Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| count next | count[3:0] | resetN | | **CLK** |
| 4'b0000 | 4'b0000 | 0 | **x** | |
| count+1 | count | 1 | **↑** | |

היות ובתרגיל זה יש להריץ גם **סימולציה, כעת בצע סינתזה** (**Analysis & Synthesis**). הוסף את הקוד ואת את המימוש כ- RTL VIEW לדו"ח.

**קוד SV – לאחר סינתזה מוצלחת**

**הצג את המימוש כ RTL VIEW**

**צור קובץ WAVEFORM והרץ סימולציה של המעגל**.

הוסף את תוצאות הסימולציה לדו"ח.

**פלט סימולציה**

# מונה סינכרוני עם קפיצות

השלם את הקוד במודול של מונה בינארי סינכרוני עולה **jmp\_counter.sv** .

Module interface

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name |  | Type | Direction | |
| **clk** |  | logic | **input** |
| resetN |  | logic | **input** |
| count | [3:0] | logic | **output** |

Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| count next | count[3:0] | resetN | | **CLK** |
| 4'b0000 | 4'b0000 | 0 | **x** | |
| count+1 | count | 1 | **↑** | |
| 4'b1011 | 4'b0110 | 1 | **↑** | |
| 4'b0000 | 4'b1111 | 1 | **↑** | |

הסופר עד15 עם קפיצות: המונה מתחיל לספור מ- 0, סופר עד 6, קופץ ל- 11, ממשיך לספור עד 15, חוזר ל- 0, ושוב סופר מ- 0 עד 6, קופץ ל- 11, סופר עד 15, מתאפס וכן הלאה ממשיך בצורה מחזורית.

בצע סינתזה והעתק את הקוד שכתבת לכאן.

**קוד SV אחרי סינתזה מוצלחת**

צור קובץ WAVEFORM והרץ סימולציה של המעגל.

הוסף את תוצאות הסימולציה לדו"ח.

**פלט סימולציה**

# תצוגת 7Segment עם הדלקה וכיבוי מלאים

Module interface

|  |  |  |  |
| --- | --- | --- | --- |
|  | Type | Direction | |
| **darkN** | logic | **input** |
| **LampTest** | logic | **input** |
| **hexin[3:0]** | logic | **input** |
| ss[6:0] | logic | **output** |

Truth table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ss[6:0] | **hexin[3:0]** | **LampTest** | **darkN** | |
| 7'b1111111 | 4'hxx | **x** | **1'b0** |
| 7'b0000000 | 4'hxx | **1'b1** | **1'b1** |
| 7'b1000000 | 4'h0 | **1'b0** | **1'b1** |
| 7'b1111001 | 4'h1 | **1'b0** | **1'b1** |
| 7'b0100100 | 4'h2 | **1'b0** | **1'b1** |
| 7'b0110000 | 4'h3 | **1'b0** | **1'b1** |
| 7'b0011001 | 4'h4 | **1'b0** | **1'b1** |
| 7'b0010010 | 4'h5 | **1'b0** | **1'b1** |
| 7'b0000010 | 4'h6 | **1'b0** | **1'b1** |
| 7'b1111000 | 4'h7 | **1'b0** | **1'b1** |
| 7'b0000000 | 4'h8 | **1'b0** | **1'b1** |
| 7'b0010000 | 4'h9 | **1'b0** | **1'b1** |
| 7'b0001000 | 4'hA | **1'b0** | **1'b1** |
| 7'b0000011 | 4'hB | **1'b0** | **1'b1** |
| 7'b1000110 | 4'hC | **1'b0** | **1'b1** |
| 7'b0100001 | 4'hD | **1'b0** | **1'b1** |
| 7'b0000110 | 4'hE | **1'b0** | **1'b1** |
| 7'b0001110 | 4'hF | **1'b0** | **1'b1** |

בשאלה זו יהיה עליך לתכנן ממיר צירופי מקוד בינארי ברוחב 4 סיביות לתצוגת  
 Seven Segment עבור כל 16 הצירופים האפשריים של 4 הסיביות. לממיר עוד 2 כניסות בקרה של סיבית אחת כל אחת, **LampTest** ו **darkN** אשר תפקידן נתון בטבלת האמת.

נתון לך שלד של רכיב בשם **HEXSS**, בו תשלים את הקוד שלך. רכיב זה יהיה שימושי בהמשך בניסויים הבאים ובפרויקט הסופי.

תצוגות Seven Segment בעלת המבנה המרחבי הבא:



**הערה: מותר להיעזר בקוד HEXSS שקיבלת במעבדה סכמתי1 בתור דוגמה**

יש לממש את טבלת ההמרה בעזרת מערך דו ממדי, אותו אפשר להגדיר כך:

**ולא**  להשתמש בפקודת CASE .

**logic** **[**0**:**15**]** **[**6**:**0**]** SevenSeg **=**

'**{** 7'h40**,** //0

7'h79**,** //1

..........

};

סרטט **בעפרון** סמל גרפי של הרכיב (כניסות ויציאות)

החלף בשרטוט

בצע סינתזה והעתק את הקוד שכתבת לכאן.

**קוד SV אחרי סינתזה מוצלחת**

צור קובץ WAVEFORM והרץ סימולציה של המעגל.

**שים לב** בסימולציה ניתן לקבוע בקלות את ערכי הכניסה hexin העוקבים 0 עד F בעזרת **כלי המונה, ה- Count Value**  שנמצא על סרגל הכלים בחלון הסימולציה,   
ה- Simulation Waveform Editor.

הוסף את תוצאות הסימולציה לדו"ח.

**פלט סימולציה**

# גיבוי העבודה

שמור את הפרויקט רגיל וגם **כארכיב (באמצעות Project -> Archive Project)**.

פעולת הארכיב יוצרת קובץ עם סיומת \*.qar אותו תגבה, **העלה למודל והבא למעבדה** כי תצטרך אותו בניסוי.

**באופן כללי יש לשמור, לגבות ולהביא למעבדה את כל קבצי הקוד והפרויקטים שכתבתם כי תשתמשו בהם במהלך הניסויים ובפרויקט הסופי.**

***לאחר שסיימת - לחץ על ה LINK ומלא בבקשה את השאלון המצורף***

|  |
| --- |
|  |
| |  | | --- | | [**מלא את הטופס**](https://docs.google.com/forms/d/e/1FAIpQLScKIokZgowmcDuf0l79Qzn_sybx6sq9v_V_CBx9J30Exvg08w/viewform?c=0&w=1)  שמור דו"ח זה כ-PDF והעלה למודל | |