האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

בית הספר להנדסה ולמדעי המחשב
מעבדת אלקטרוניקה ספרתית

יום א׳ 10 למרץ 2019
ג׳ באדר התשע״ט

הכרת Quartus והכרת כרטיס DE1 FPGA
Lab #1

במעבדה זו

- נתחיל להכיר את תהליך הפיתוח בסביבה של תוכנת QUARTUS
- ואת כרטיס ה DE1 FPGA

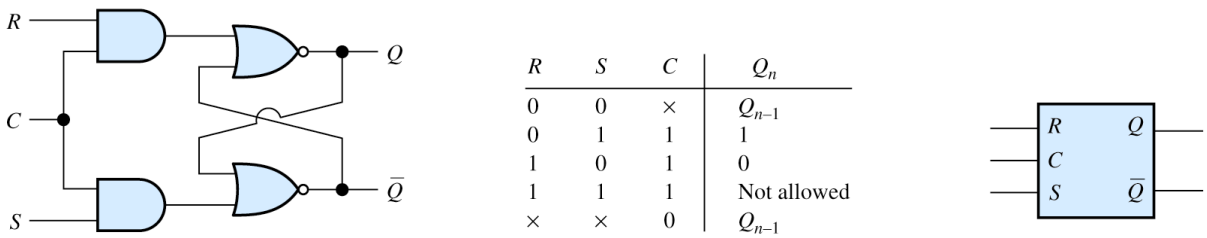
במעבדה זו נלמד איך פותחים פרויקט ואיך עוברים את כל התהליך :

- פתיחת פרויקט ב QUARTUS
- תכנון לוגי בכל אחת מהשיטות: שרטוט חשמלי או Verilog (ראה דרישות בכל סעיף)
- קומפילציה של התכנון
- סימולציה ע״י תוכנת הסימולציה של QUARTUS
- הורדת התכנון לכרטיס ובדיקה בפועל – צריבה לכרטיס
- כולל בחינת אופן המימוש של הפונקציה

נתחיל במעגלים פשוטים לפי הסעיפים הבאים :

1. <u>מימוש FLIP FLOP מסוג SR (ב Gate level)</u>

- בסעיף זה נכיר איך מיישמים Flip Flop מסוג SR
- תחילה, עליכם לממש את הרכיב בעזרת שערים לוגיים, בדקו שהוא מתאים להגדרה (טבלת האמת של SR Flip Flop)
- פתח פרויקט חדש בעזרת תוכנת QUARTUS, כאשר הקפד/י לבחור את ה-FPGA שבו נשתמש (ראו תיעוד של הכרטיס DE1)
- פתחו קובץ תכנון בעזרת שרטוט רגיל של QUARTUS
- יש לעשות את התכנון, לעבור קומפילציה לבצע סימולציה פונקציונלית. שימו לב שבסימולציה לא די לבדוק כי המצבים בטבלת האמת מתקיימים, אלא כי כל המעברים האפשריים מתבצעים באופן תקין.
- **בתרגיל זה אין צורך להוריד את התכנון לכרטיס**



R	S	C	Q_n
0	0	×	Q_{n-1}
0	1	1	1
1	0	1	0
1	1	1	Not allowed
×	×	0	Q_{n-1}

(a) Circuit diagram (b) Truth table (b) Circuit symbol

Figure 7.41 A clocked SR flip-flop.

נקודות שצריך לכלול בדו"ח:

- האם מדובר ברכיב סינכרוני או א-סינכרוני? מה ההשלכות של זה על התכנון?
- צרפו תמונה של התכנון באמצעות block diagram כפי שמימשתם ב-Quartus.
- בצעו סימולציה functional והציגו תוצאותיה (צילומי מסך של ה-wave form) הסבירו באופן מילולי מה רואים בסימולציה, כיצד ניתן להבין מכך את נכונות התכנון.

2. <u>סלקטור 8 ל 1 (8 to 1 MULTIPLEXER) באמצעות סלקטור 2 ל-1 (ב Gate level וגם ב Verilog)</u>

- בסעיף זה נבנה סלקטור מ-8 ל-1 במספר דרכים, על מנת להשוות בין דרכי המימוש השונות:

1. מימוש באמצעות דיאגרמת בלוקים.

א. יש לממש mux2:1 בדיאגרמת בלוקים. שמרו את הקובץ כבלוק נפרד (File →Create/Update→Create Symbol Files for Current File).

ב. בקובץ תכנון חדש, ממשו mux8:1 תוך שימוש במספר בלוקים של mux2:1 שהכנתם בסעיף הקודם.

2. מימוש בלוק של mux2:1 בקוד Verilog ובנייה של mux8:1 בדיאגרמת בלוקים באמצעותו.

ג. יש לממש mux2:1 בקובץ Verilog. שמרו את הקובץ כסימבול (באותו אופן כמו בסעיף 1א).

א. פתחו קובץ תכנון בבלוקים, והרכיבו mux8:1. הפעם השתמשו לבנייתו בבלוקים של mux2:1 שמימשתם בקוד.

3. מימוש בלוק של Mux2:1 בקוד Verilog ובנייה של mux8:1 בVerilog באמצעותו.

פתחו קובץ תכנון בוירלוג. ממשו mux8:1 על ידי הגדרת מופעים (instances) של המודול שהגדרתם בסעיף 2א.

4. **מימוש בקוד Verilog של mux8:1 באופן ישיר.**

ממשו באופן ישיר בקובץ Verilog חדש mux8:1 באופן הפשוט ביותר.

- בצעו שתי סימולציות (פונקציונליות) ובדקו בהן את נכונות התכנון של mux8:1 מסעיפים 1 ו-4 בלבד.
- צרבו לכרטיס ה-FPGA את התכנון מסעיף 4 ובדקו את פעולתו. השתמשו ב(SW(0:2 עבור הסלקטורים, וביתר המתגים עבור הכניסות. היות ואין מספיק מתגים בלוח, את הכניסות הנותרות חברו ל-KEY. את התוצאה הציגו ב(LEDG(0.

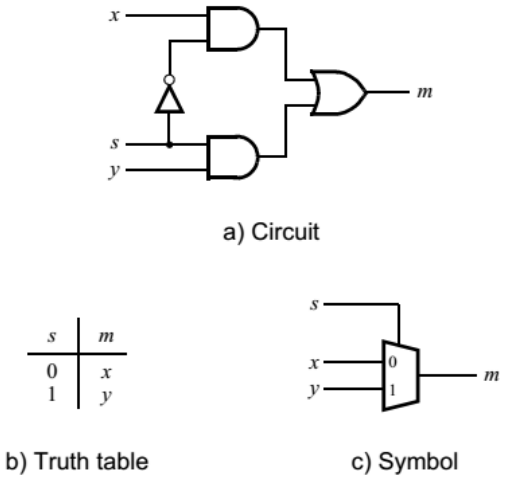
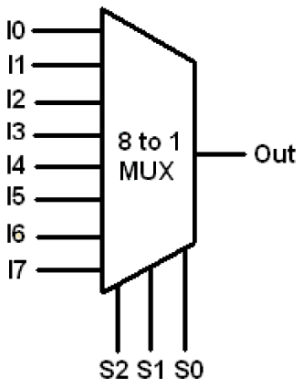


Figure 2: A 2-to-1 multiplexer.



יש לכלול בדו״ח:

- את ארבעת המימושים השונים.
- סימולציה פונקציונלית למימוש ב-Verilog מסעיף 4, לצד הסבר כיצד ניתן להסיק ממנה את נכונות התכנון.
- סימולציה פונקציונלית למימוש בסכמה מסעיף 1, לצד הסבר כיצד ניתן להסיק ממנה את נכונות התכנון.
- מנו שני יתרונות לשימוש בשפת HDL ושני יתרונות לשימוש בסכמת בלוקים.

3. בדיקת מכונת המצבים מתרגיל הבית

- בסעיף זה נצרוב את מכונת המצבים שבניתם בתרגיל הבית השני (סעיף 1), המזהה את הרצף "1010".
- לצורך הצריבה של מכונת המצבים, השתמשו ב-KEY0 לרסט את מכונת המצבים (זכרו כי ריסט הוא reset low) וב-SW0 כדי להכניס את הסדרה In ולבסוף בעזרת KEY1 כדי לספק שעון כניסה למערכת.
- השתמש ב LEDG0 כדי להציג את מצב היציאה Out, בנוסף חברו את כל אחד מחמשת המצבים של המכונה ל LEDR0 –LEDR4 (ניתן גם להציג בקידוד בינארי ולהשתמש רק בשלושה לדים)
- בצעו את ה-pin assignment באופן הבא:

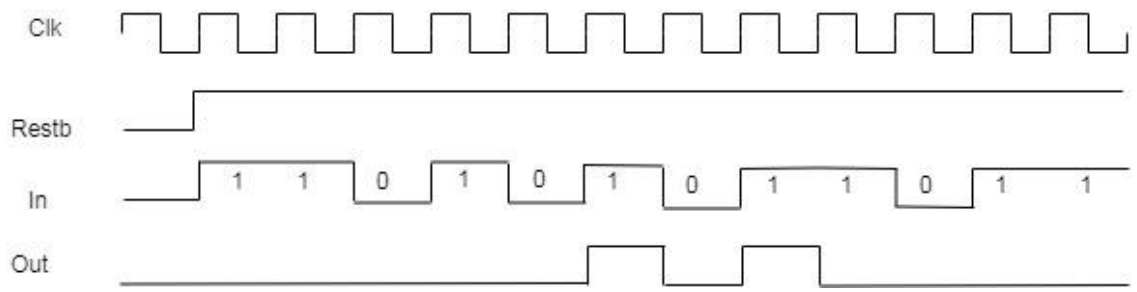
במקום להשתמש ב-pin planner (שבו יש להזין הכל באופן ידני, ויש לחזור על הפעולה הזאת בכל פעם שמתחילים קובץ חדש וכיו"ב), ניתן לייבא קובץ שמתאים לכל כניסה ויציאה את ה-pin המתאים עבורה.
פתחו קובץ אקסל, ורשמו בעמודה הראשונה את שמות הפורטים שמופיעים בקוד שלכם ובעמודה השנייה את שם ה-pin המתאים. בראש העמודות רשמו את המלים To ו-Location בהתאמה, בפורמט הבא –

	A	B
1	To	Location
2	clk	PIN_A6
3	enable	PIN_G21
4		
5		

שמרו את הקובץ כ-CSV. לאחר מכן, בתוכנת Quartus ייבאו את הקובץ באמצעות:

Assignments->Import assignments…

- הראו את נכונות המכונה על ידי הרצה של הרצף שניתן.
- הוסיפו שורה לדיאגרמת הגלים שסופקה לכם בתרגיל הבית השני שתציין את המצב בו נמצאת מכונת המצבים.



4. <u>מונים (בעזרת פונקציות מערכת LPM וגם ב Verilog)</u>

- בסעיף זה נממש מונה שסופר שניות על ידי שימוש בפונקציית מערכת מוכנה מספריית LPM וגם בעזרת VERILOG, אפשר להשתמש באחד משעוני המערכת של 27Mhz או 50Mhz.
- המונה צריך לספור עד 99 שניות, ולאחר מכן להתאפס ולהמשיך לספור.
- יש לתכנן את המונה, לכתוב את הקוד באמצעות שפת ורילוג, לקמפל ולבצע סימולציה פונקציונלית ובנוסף סימולציית timing.
- מה התדר המקסימלי בו המונה שמימשתם יכול לעבוד? (קראו את הפלט בקומפילציה של quartus).
- השתמש ב HEX1 ו HEX2 כדי להציג את מצב המונה.
- בדוח יש לתעד את התכנון והסימולציה שמוכיחה שהתכנון מבצע את הפונקציה המוגדרת בתרגיל.

בדו"ח עליכם לכלול:

- הסבירו מה מכיל הפרויקט שלכם ומדוע בחרתם לחלק את הרכיבים באופן זה.
- הציגו את כל חלקי התכנון
- סימולציות – גם functional וגם timing. מה בודקים בכל אחת מהסימולציות?
- מהו ובמה תלוי התדר המקסימלי בו המונה שלכם יכול לעבוד?

<u>כתיבת דו"ח המעבדה</u>

בדו"ח שעליכם להגיש אתם מתבקשים לדווח על מה שביצעתם במעבדה, ולהראות כי מעבר לכך שהצלחתם לבצע את המטלה הנדרשת, הבנתם היטב את שלבי התהליך. כמו כן, אתם נדרשים להסביר מה היו השיקולים בתכנון או המימוש שבחרתם, וכיצד בדקתם את נכונות המימוש.
בכל קטעי הקוד, יש לצרף הערות כנהוג בכתיבת קוד תקינה, המבהירות את תפקיד אותו קטע קוד, **הקפידו על קוד קריא ומסודר!**
בהצלחה!