מטרת הקורס ושיטת הלימוד

הקורס מקנה ידע רחב בהכרת שפת תיאור חומרה VHDL לסימולציה ולסינתזה עם רכיבים מתוכנתים (FPGAs). הקורס מיועד לבעלי ידע בסיסי או חסרי ידע בשפה.

משך הקורס המשוער לפי התכנית שתפורט בהמשך, עומד על כ 66 שעות הרצאה (ויכול לכלול תרגול מודרך קצר או הדגמה). ניתן להיות גמישים במשך הקורס, ובמידה ולא מספיקים את כל החומר בזמן המוקצב ניתן להוסיף פגישות נוספת. במידה ומסיימים את החומר מוקדם יותר ניתן להקדים את סיומו. הקצב תלוי במידה רבה במשתתפים. הזמן המוקצב בקורסים רבים שהועברו בעבר במסגרות שונות נע בין 50 ל - 80 שעות.

הקורס מועבר ברובו הגדול כהרצאה פרונטאלית. הפגישות תהיינה מלוות בשקפים מודפסים (כ - 1500) ומבוסס על קורסים דומים רבים שהועברו במסגרות שונות בתעשייה ובאקדמיה. ניתן לשלב בקורס גם הדגמה ותרגול מודרך כולל צריבה במידת הצורך. לקורס מתלווה ספר לימוד פופולארי שצמוד ומתאים לתכנים של הקורס. מספר המשתתפים אינו קריטי (פרט למקרה שנעשה תרגול מודרך משותף במעבדה, במידה ויתקיים).

מומלץ שהקורס יועבר בפגישות שהן בתדירות של כפעם בשבוע. הקורס שמועבר באופן כזה יכול להיות מבוסס במידה רבה גם על תרגול עצמי של המשתתפים שלא נעשה בשעות ההרצאה עצמן (שעורי בית). אוכל לסייע למשתתפי הקורס במידה מסוימת באמצעות E-mail.

הכלי העיקרי המומלץ לתרגול עצמי הוא כלי סימולציה Modelsim וכלי הסינתזה Quartus (שתיהן גרסאות Web חינמיות שניתנות גם להתקנה חופשית בבית וללא מגבלות זמן). כמובן שמשתתפים בקורס יוכלו להשתמש בידע שרכשו לשימוש גם בכלי פיתוח אחרים.

ראשי פרקים של נושאי הקורס

מס	הנושא	שעות
1	מבוא לשפה לכלים ולמהלך התכן	4
2	יסודות השפה, פעולותיה וסוגי המידע שלה	8
3	תיאור התנהגותי מקבילי וסדרתי ושימוש בתהליך	6
4	אבני בנייה לסינתזה צירופית	4
5	אבני בניה לסינתזה סינכרונית	8
6	אבני בניה לסימולציה	4
7	תיאורים מבניים בסיסיים	4
8	תיאורים מבניים מורכבים	4
9	מכונת מצבים	8
10	מערכים ורכיבי זיכרון	4
11	פונקציות ופרוצדורות כהרחבות לשפה	4
12	טיפול בקבצים	4
13	חידושים ב - VHDL-2008	4
	סכה"כ	66

בכבוד רב
מרצה הקורס
עמוס זסלבסקי
דואר אלקטרוני : amos.zaslavsky@gmail.com
אתר : http://www.aztech.co.il
פלאפון : 050-7270673

מצורפים בהמשך :
[1] תכנית לימודים מפורטת
[2] נספחים נוספים

תוכנית לימודים מפורטת

חלק 1 - מבוא לשפה לכלים ולמהלך התכן (4 שעות עד 6 שעות)

1.1	הכרת מקורות השפה ויעודה המקורי לתיעוד וסימולציה, והתפתחות כלי סינתזה שתומכים בחלק (Subset) מהשפה.
1.2	השוואה בין שפות.
1.3	קשיים והמלצות לאופן הלימוד של שפת VHDL.
1.4	השוואה בין Design-Entry (הכנסת תכן) גרפי ושפתי (Hardware Description Language) והיתרונות והחסרונות של כל שיטה.
1.5	הכרת Design Flow מודרני הכולל סימולציה פונקציונאלית בשפת המקור לפני הסינתזה ויתרונותיו והשוואה בין סימולציה ברמת המקור (Source-Level-Simulation) לעומת סימולציה ברמת שערים (Gate-Level-Simulation).
1.6	הכרת כמה מושגים ויכולות של השפה: תאור מקבילי\טורי, תאור התנהגותי\מבני ושימוש ברכיבים מובנים של יצרן הרכיב (Primitives, LPMs וכו..).
1.7	הדגמת כלים (ראה בהמשך) *
1.8	מבוא לסימולציה, הערה: בתחילת הקורס (עד פרק 7) נבצע סימולציה באמצעות שפת הסימולאטור ורק בהמשך נשתמש בהיררכיה וב - Test Bench.
1.9	מבוא לסינתזה: כלים פנימיים של הסינתזה, תיאור ברמת RTL לעומת Gate-Level, כתיבה לסינתזה, ודוגמאות לקוד שלא עובר סינתזה.
1.10	פעולות נוספות של כלי הסינתזה: STA, סימולציה פנימית ברמת שערים באמצעות סימולאטור פנימי של הכלי או לחילופין באמצעות יצירת קוד VHDL ברמת Gate-Level על ידי כלי הסינתזה.
1.11	הכרת סוגי כלים נוספים, נחיצותם של כלים שונים והקשר ביניהם.
1.12	מעבדים לעומת FPGAs, כמה דוגמאות ליישומים עם FPGAs. מילות "באז" וכמה מלים על ASIC.

* חלק זה (של הדגמה ואולי גם תרגול מעשי) יכול להיעשות קודם והוא יכלול Design-Flow מלא של דוגמה פשוטה עד לשלב הצריבה:

1.13	**סימולציה ברמת המקור:** כתיבת קוד VHDL של מערכת צירופית פשוטה באמצעות עורך טכסטים (Text Editor). תחת הסימולאטור ברמת המקור ניצור Library בשם work. נקמפל את קובץ המקור ונניפה שגיאות. נריץ את הסימולציה באמצעות פקודות של שפת הסימולציה. תוצאות הסימולציה תוצגנה בחלון wave או list.
1.14	**סינתזה:** יצירת פרויקט לסינתזה שמכיל את הקובץ שיצרנו בחלק הקודם. בחירת הרכיב ושליטה במיקום ההדקים שלו. ביצוע סינתזה וניפוי שגיאות והתבוננות בהודעות האזהרה ומשמעותן. התבוננות בכלי הדידקטי החשוב של ה RTL-Viewer שמציג את הפרשנות שכלי הסינתזה נתן לקוד שלנו. התבוננות ב – Report-File.
1.15	**סימולציה ברמת שערים:** נבצע גם סימולציה ברמת שערים. ניתן לבצע סימולציה כזו באמצעות הסימולציה הפנימית של כלי הסינתזה עצמו. לחילופין ניתן גם ליצור קובץ VHO באמצעות כלי הסינתזה ולבצע סימולציה בכלי החיצוני שבו השתמשנו בסימולציה הקודמת.
1.16	**בדיקה אופציונאלית בחומרה:** הורדת תכן לרכיב על לוח התרגול (תכנות\צריבה) ונבדוק את התכן.

חלק 2 - יסודות השפה, פעולותיה וסוגי המידע שלה (8 שעות)

2.1	כללי תחביר בסיסיים: הערות, כללים למתן שמות, רשימת מלים שמורות, שורה פיסית ולוגית.
2.2	יחידות קומפילציה: הישות (entity) והארכיטקטורה (architecture).
2.3	כיווני הדקים (modes) בישות.
2.4	אותות פנימיים (signals) שמוגדרים בחלק ההצהרתי של הארכיטקטורה.
2.5	נושא אופציונאלי: תיאור תזמונים ומודלים אפשריים שונים inertia ו transport.
2.6	נושא אופציונאלי: ריבוי ארכיטקטורות.
2.7	אותות, משתנים וקבועים.
2.8	סוגי מידע מספרי: (real, integer), casting, מספרים קבועים בבסיסים שונים, נגזרות של integer והמשמעות של תיחום integer עבור סימולציה וסינתזה ובעיות שיכולות להיווצר בהבדלים בין סימולציה וסינתזה של שלמים.
2.9	סוגי מידע enumerated שמובנים בשפה: boolean, character ו bit ומערכים מובנים שלהם: string ו bit_vector.
2.10	הכרת פעולות שרשור והשמה על וקטורים: השמות של וקטורים קבועים בבסיסים שונים, שימוש ב - Aggregates ושימוש ב positional association ו named association ו slices ושימוש ב others והדבקת תווים באמצעות &, נוקשות של סוגי מידע משני צידי ההשמה.
2.11	פעולות Shift, הבעייתיות של שלהן, ויצירת פעולות shift שונות באמצעות שרשור ו slice.
2.12	פעולות לוגיות (and,or,not,xor,nand,nor,xnor) והאופרנדים האפשריים שלהם. התגברות על בעיה של ביצוע פעולה משולבת בין ביט בודד ובין וקטור.
2.13	פעולות השוואה (רלציות) מהסוגים: (<,>,=,<=,>=,=!) והאופרנדים האפשריים שלהם וסוג המידע המוחזר.
2.14	המשמעות המספרית של השוואות ביו וקטורים ברוחב שווה והבעייתיות כששני האופרנדים לא באותו הרוחב.
2.15	פעולות חשבוניות (+,-,*,/,abs,rem,div,**) והאופרנדים האפשריים שלהם.
2.16	בעיות בסוגי המידע המובנים בשפה והצורך לבצע הרחבות באמצעות חבילות שמקומפלות לספריות שונות.
2.17	חבילות וספריות והפיכתן לנגישות.
2.18	**הרחבת השפה:** הרחבות של סוגי המידע bit ו bit_vector והשימוש ב std_logic ו std_logic_vector (בכדי להשיג עושר גדול יותר בערכים לוגיים). שימוש בספריה ieee ובחבילה std_logic_1164. הכרת כמה מהפונקציות המובנות של החבילה.
2.19	**הרחבת השפה:** שימוש בחבילות של: std_logic_unsigned או std_logic_signed בכדי לאפשר בנוחות ביצוע של פעולות חשבוניות על וקטורים ובכדי למנוע את הבעייתיות של שימוש ב integers והבעייתיות בשימוש בהשוואות וקטורים ברוחב שונה שהם חסרי משמעות חשבונית. הכרת הפעולות של החבילה כולל פעולות shift.
2.20	שימוש אופציונאלי בחבילה numeric_std של ieee והיכולת לבצע פעולות בעלות משמעות של unsigned או signed באותה ארכיטקטורה וחילוק של וקטורים.

חלק 3 - תיאור התנהגותי מקבילי וסדרתי ושימוש בתהליך (8 שעות)

3.1	**התחביר הבסיסי של התהליך:** רשימת רגישות וחלק הצהרתי וחלק ביצועי.
3.2	**המצבים שבהם יכול להימצא התהליך:** פעיל (Active) או במנוחה (Suspended) והמעבר בניהם.
3.3	השמה בארכיטקטורה כמקרה פרטי של תהליך עם השמה.
3.4	**המכניזם של חישוב השמות לאותות בתהליך:** [1] השוואת אות ביניים בארכיטקטורה לאות ביניים בתהליך. [2] האם יש דבר כזה אותות פנימיים בתהליך שיכולים לבצע חישובי ביניים בארכיטקטורה (אין). [3] הכרת אופן הביצוע של השמות לאותות בפועל: [4] צד ימין מחושב תוך כדי הריצה כשהתהליך רץ (והערכים מאוחסנים בזיכרון זמני וההשמות מתבצעות בפועל בסוף הריצה של התהליך. [5] הכרת משתנים שהם האובייקטים המשמשים לחישובי ביניים תוך כדי הריצה של התהליך. אפשר להצהיר על משתנים בתהליך אך לא ניתן להצהיר עליהם בארכיטקטורה ומנגד ניתן להצהיר על אותות בארכיטקטורה ולא ניתן להצהיר עליהם בתהליך.
3.5	**סדר השמות בתיאור מקבילי וטורי:** [1] הסדר של השמות בארכיטקטורה (שאינו חשוב) לעומת הסדר של השמות בתהליך (שהוא כן חשוב). [2] כל השמה בארכיטקטורה היא בעצם תהליך (שרשימת הרגישות שלו הם האותות שרשומים בצד ימין של ההשמה) ולכן גם סדר התהליכים בארכיטקטורה אינו חשוב.
3.6	**השמות מרובות לאות או משתנה:** בארכיטקטורה נוצר Wired-Logic (כלומר חיבור בין יציאות) ולעומת זאת בתהליך ההשמה האחרונה קובעת.
3.7	**השהייה של אות מעבר ל delta ואי יכולת להשהות משתנה**
3.8	**אופיו הסיבובי של התהליך:** תהליך כחוג אינסופי, התפקיד של רשימת הרגישות כמכניזם סנכרון לעולם החיצוני. רשימת רגישות שוות ערך לפסוק wait on בסוף התהליך. מה קורה כשאין מנגנון סנכרון והגנות שונות וצעדים מעשיים כנגד היווצרות חוג אינסופי שלא מסתיים.
3.9	**התניה ובחירה בארכיטקטורה:** [1] השמה מותנית (conditional assignment), הבדלים בין VHDL-87 ו , VHDL-93, תזמונים [2] השמה נבחרת (Selected assignment), מנגנונים שונים של קיבוץ כמה אפשרויות כאופציה אחת, אפשרויות חסרות, אפשרויות כפולות [3] הצגת התחביר ואופן הפעולה של הפסוקים הנ"ל בדוגמאות שונות כגון: בורר (Mux), מפלג (De-Mux), מפענח (Decoder) ומקדד עדיפויות (Priority Encoder).
3.10	**התניה ובחירה בתהליך:** פסוקי if, פסוקי case, קיבוץ אפשרויות שונות לאופציה אחת, אפשרויות חסרות וכפולות, הבדלים בין פסוקי if של VHDL ו C וכן ההבדלים בין פסוקי case של VHDL ופסוקי switch של שפת C. ביצוע הזחות נכונות בשפת VHDL. הצגת התחביר ואופן הפעולה של הפסוקים הנ"ל בדוגמאות שונות כגון: בורר (Mux), מפלג (De-Mux), מפענח (Decoder) ומקדד עדיפויות (Priority Encoder).
3.11	**חוגי for:** הכרת חוג מסוג for. הכרת התחביר והצגת כמה דוגמאות. אזכור השילוב הנפוץ בין Loop מסוג for ביחד עם Variable שמשמש לחישובי תוצאות ביניים. יצירת חוג for סטטי שמתאים לסינתזה מחוג שאינו כזה.
3.12	**חוגים אחרים:** חוג while וכן חוג ללא מנגנון איטרציה או חוגים עם פסוק exit ו next. הבעייתיות של חוגים כאלה בכתיבה לסינתזה (הם אינם חוגים סטטיים).

חלק 4 - אבני בנייה לסינתזה צירופית (4 שעות)

4.1	סוגי מערכות חומרה אפשריות לסינתזה: מערכת צירופית, מערכת סינכרונית ו gated-Latch. צורות תיאור קוד אפשריות לתיאור מערכת צירופית עם תהליך וללא תהליך. מתי כדאי להשתמש בתהליך?
4.2	הקפדה על הימנעות ממשוב (כלומר הימנעות מרישום אות בשני צדי השמה).
4.3	מה קורה כשלא רושמים אות כניסה ברשימת רגישות? הקפדה על רישום כל אותות הכניסה ברשימת הרגישות (complete sensitivity list). אלו אותות נחשבים אותות כניסה?
4.4	מה קורה כאשר משתמשים בהשמות ליציאה שאינן מלאות? הקפדה על השמות מלאות (complete assignments) לכל היציאות והשיטה המועדפת של שימוש בהשמות ברירת מחדל.
4.5	הקפדה על שימוש נכון ב – Variables כאובייקטים לחישובי ביניים בלבד (תמיד מציבים אליהם ערך לפני שקוראים אותם ומצבים את הערך שנקרא חזרה לאות).
4.6	המלצה להימנעות מ – Multiple Conditioning. הבדלה בין מצב זה ואיסור להשתמש ב Multiple-Assignments (כשלא מדובר ברכיבים עם יציאות שיכולות להתנתק).
4.7	המלצה לשימוש בהשמות ברירת מחדל בכדי לכסות את כל אפשרויות הכניסה הנדרשים בסוג המידע std_logic_vector וגם כאשר משתמשים ב bit_vcetor מכיוון שכלי הסינתזה מצפים לסוג המידע std_logic_vector.
4.8	תיאור של רכיבי Tri-State Buffer ו Open-Drain Buffer באמצעות השמות מותנות. כמכנה משותף נמוך של כל כלי הסינתזה.
4.9	אי תמיכה של כלי סינתזה במצבים לוגיים חלשים מתוך הקוד ובצירופי ברירה.
4.10	צורת כתיבת קוד (שבלונה) לתיאור Gated Latch (הקפדה על רשימת רגישות מלאה והשמות לא מלאות).

חלק 5 - אבני בניה לסינתזה סינכרונית (8 שעות)

5.1	**השלמה וחזרה בנושא תכן סיכרוני (כולל סרטים לצפייה עצמית):** **כיצד נגרמים מרוצי יציאה:** [1] אי אחידות במהירות של פליפ-פלופים [2] חומרה צירופית שנמצאת בין הפליפ-פלופים והיציאה שמסוגלת לייצר Hazards. **הרעיון המרכזי של תכן סינכרוני:** אותות עם מרוצי יציאה מוזנים לכניסות של מערכות סינכרוניות נוספות שאות השעון שלהם הוא אותו אות השעון שיצר את מרוצי היציאה. **הכרת כללי עשה ואל-תעשה בתכן סינכרוני:** [1] השתמש בפליפ-פלופים שרגישים לעליה בלבד [2] הזנה ישירה של אותות השעון של כל הפליפ-פלופים לכניסת השעון [3] איסור הכנסה של השהייה כל שהיא על אותות השעון [4] שימוש בכניסות אסינכרוניות אך ורק לבצוע אתחול אחרי הדלקת המתח [5] חובת הפרדה בין האותות : אות השעון, אותות אסינכרוניים ואותות סינכרוניים [6] הימנעות משימוש במשוב צירופי (מכונה אסינכרונית) [7] הימנעות משימוש במונים אסינכרוניים. **חלק אופציונאלי:** טכניקות ניקיון יציאות מעשיות : [1] סנכרון יציאה רועשת (החיסרון הוא יציאת Latency) [2] שימוש ביציאות שמגיעות ממכונות מסוג Direct-Moore (הסבר מלא יינתן בהמשך בפרק שעוסק במכונת מצבים).
5.2	תיאור תהליכים שמסונכרנים לאות השעון : באמצעות שימוש ב - clk'event או rising_edge(clk) או falling_edge(clk).
5.3	**תיאור השבלונה הסינכרונית הבסיסית:** [1] רשימת רגישות שאינה מלאה הכוללת את אות השעון ואותות אסינכרוניים בלבד. [2] התניה if התחלתית שתלויה באותות אסינכרוניים (במידה וישנם כאלה) [3] התניה סינכרונית כהתניה ראשונה או לאחר התניה אסינכרונית. [3] איסור להכנסה של התניות נוספות בתנאי if הראשי לאחר התנאי הסינכרוני (Nested if מותר).
5.4	**על ההשמות בשבלונה הסינכרונית:** [1] כל השמה באזור סינכרוני יוצרת יציאה מסונכרת (צד שמאל של ההשמה). [2] האות בצד ימין של ההשמה הוא אות כניסה של פליפ-פלופ [3] במקרה של ביטוי בצד ימין של ההשמה, מדובר במערכת צירופית שמזינה את כניסות הפליפ-פלופים (לוגיקת עירור).
5.5	סוגיות שקשורות לכניסות אסינכרונות : שימוש ב - Not-Gate-Push-Back והמלצה שלא להשתמש בקוד בכמה כניסות א-סינכרוניות.
5.6	**הכרת שתי הדרכים ליצירה של מערכת צירופית שמזינה את הפליפ-פלופים:** [1] ביטוי בצד ימין של ההשמה [2] תנאי if מקונן (nested if).
5.7	הצגת כמה דוגמאות סינכרוניות : מונים שונים, רגיסטרים (מקבילי והזזה).
	הכרת כמה דוגמאות שמדגימות את הנוקשות של השבלונה הסינכרוני והתגובה של כלי סינתזה : למשל הכרת ההבדלים העדינים בין תיאור מערכת סינכרונית ללא reset ו Gated-Latch, הדרך הנכונה ליצירת אפשור סינכרוני ועוד...
5.8	הקפדה על שימוש נכון ב – Variables כאובייקטים לחישובי ביניים בלבד : (תמיד מציבים אליהם ערך לפני שקוראים אותם ומצבים את הערך שנקרא לאות).
5.9	איסור על ערבוב בין מערכות צירופיות וסינכרוניות באותו תהליך. החשיבות של פירוק המערכת לתהליכים צירופיים וסינכרוניים נפרדים (יצירת סקיצת RTL) לפני שכותבים את הקוד ולאחר מכן כתיבת קודים צירופיים וסינכרוניים נפרדים בארכיטקטורה לפי כללי הכתיבה ושבלונות שנלמדו בפרקים קודמים. ביצוע כמה תרגילי תכן שונים.
5.10	מניעת ערבוב בין מערכת סינכרונית שיש לא אתחול אסינכרוני ומערכת סינכרונית שאין לה אתחול אסינכרוני (למניעה של אותות אפשור בלתי צפויים).
5.11	כמה סוגיות שקשורות לאתחול משתנים ואותות.

חלק 6 - אבני בניה לסימולציה (4 שעות)

6.1	שימוש בפסוק – assert ובפסוק report.
6.2	תהליכים עם פסוקי wait מסוגים שונים (on, until, for) , ושילובים שלהם ודוגמאות לשימוש שלהם בתיאור מערכות מורכבות שמייצרות אותות לסימולציה.
6.3	מחוללי אותות (generators) שונים (אות שעון ואות איפוס א-סינכרוני), ויצירת מחוללי אותות מורכבים יותר.
6.4	הכרת Timing Attributes ושימוש בהם לבדיקת תזמונים כלומר כ - Timing-Checkers (למשל Setup Time ו Hold Time ורוחב פולס מינימאלי).

חלק 7 - תיאורים מבניים בסיסיים (4 שעות)

7.1	יתרונות וחסרונות של תיאור מבני לעומת תיאור התנהגותי.
7.2	יסודות יצירת ההיררכיה : [1] הצהרה על רכיב (component) [2] חיווט של רכיבים (component instantiation) ושימוש בפסוקי port map [4] קישור לפי מיקום (named association) [3] קישור הדקים לפי שם (named association).
7.3	הכרת עקרונות : עבודה Bottom-Up ועבודה Top-To-Bottom, מתי כדאי לפרק מערכת לרכיבים בארכיטקטורה לעומת פירוקה לתהליכים בארכיטקטורה.
7.4	חיווטים מיוחדים : [1] חיווט הדקי כניסה לאות קבוע [2] ניתוק הדקים (פסוק open) מסוג יציאה וגם כניסות.
7.5	חווטים מותרים בין הדקים בעלי כיוונים (Modes) שונים.
7.6	בדיקת מערכת באמצעות Test Bench לעומת שפת הסימולטור. כל הסימולציות מכאן ואילך יעשו בשיטת ה - Test-bench.

חלק 8 - תיאורים מבניים מורכבים (4 שעות)

8.1	יצירת רכיבים עם פרמטרים גנריים (generic) - למשל רוחב של רכיב. [1] הצהרה על רכיב הכוללת פרמטר גנרי (generic) [2] חיווט של רכיבים ושימוש בפסוקי generic map וקישור על פי מקום (Named Association) וקישור על פי שם (Named Association). דוגמאות כרכיבי LPM.
8.2	יצירת רכיבים גמישים באמצעות שימוש הדקים שלא הוגבלו (Unconstrained ports).
8.3	יצירת תיאור איטרטיבי (חוג) של חמרה בארכיטקטורה באמצעות for-generate.
8.4	יצירת התניות חמרה באמצעות if-generate והשימוש בפרמטרים כתנאי להתניה.
8.5	הגדלת הנוחות בתיאורים מבניים וחיווט : באמצעות ריכוז הצהרות component בחבילה.
8.6	הגדלת הנוחות בתיאורים מבניים וחיווט באמצעות שימוש ב – direct instantiation. היתרונות והחסרונות של שימוש בהצהרת component.

חלק 9 - מכונת מצבים (8 שעות)

9.1	יצירת סוגי מידע חדשים מסוג enumerated data type וחשיבותם בייצוג סימבולי גבוה של שמות מצבים במכונה. הכרת attributes של enumerated data types.
9.2	**השלמות וחזרות בנושא מכונות מצבים סינכרונית (סרט לצפייה עצמית):** **תכן מפוצל של מערכת כמסלול נתונים ובקר:** הנוחות של תכן של מערכת באמצעות חלוקתה לשני חלקים: [1] מסלול נתונים (Data-Path) שמבצע את החישובים [2] והבקר (Controller) שמבקר ומתזמן את אלגוריתם החישוב. הצגת כמה דוגמאות. תכונות מרכזיות של מכונות שמשמשות כבקרים: [1] מצד אחד - ריבוי כניסות, ו [2] מצד שני - תלות של המעברים ממצב למצב במספר קטן של כניסות רלוונטיות. רוב מכונות המצבים בעולם המעשי הם בקרים. **תיאור אפקטיבי של דיאגראמת מצבים:** כתיבת דיאגראמת מצבים שבה תנאי המעבר ממצב למצב אינם ביטויים קנוניים, אלא ביטויים כל שהם. בדיקה שאין סתירות בין תנאי המעבר ממצב למצב (תנאי המעבר **זרים**). בדיקה שלא חסר מידע (תנאי מעבר **ממצים**).
9.3	**חזרה על חישובי תזמון של מכונת מצבים בהסתמך על פרמטרי התזמון של הפליפ-פלופים והשערים (אזכור בהרצאה כולל צפייה אופציונאלית בשני סרטים):** חישוב תדר שעון מכסימלי (fmax) של המכונה, בדיקת קיום זמן החזקה פנימי במכונה, חישוב זמן הכנה (ts) וזמן החזקה (th) וזמני tco ו tpd של מכונה.
9.4	**חזרה על סוגי מכונות:** מכונת Mealy, מכונת Moore, מכונה עצמאית (אוטונומית). ההבדלים במבנה והשלכות מעשיות: [1] סכנה של היווצרות מכונה אסינכרונית פרזיטית סביב המכונה (Mealy), [2] תדר מכסימלי ו Latency בשני סוגי המכונות [3] סינון Spikes בשני סוגי המכונות . מכונת Moore ישירה ויתרונותיה.
9.5	הצגת דוגמאות תכן של מכונה.
9.6	תיאור מכונת מצבים בצורה המקובלת: [1] אותות present_state ו next_state מסוג enumerated data type [2] תהליך סינכרוני (לפליפ-פלופים של המכונה) [3] תהליך צירופי לפונקציות העירור ולפונקציות היציאות [4] הקפדה על הכללים לכתיבה לסינתזה של שני התהליכים הנ"ל (מפרקים 4 ו 5).
9.7	סוגי מכונות שמבוטא בקוד: ההבדלים בתיאור קוד של מכונת מצבים מסוג Moore לעומת Mealy. ההבדל תלוי בהשמות ליציאה בתהליך הצירופי.
9.8	הקצאת מצבים אוטומטית של כלי הסינתזה: הקצאה בינארית הנהוגה ברכיבים קטנים (בארכיטקטורות PT כמו רכיבי MAX7K) ובשאר הארכיטקטורות (רוב הרכיבים) הבחירה היא One-Hot.
9.9	החשיבות הגדולה של הקצאה :One-Hot ביכולת הידנית לכתוב את משוואות העירור ישירות מתוך התבוננות בדיאגראמת המצבים של המכונה וללא הזדקקות לחישובים כל שהם. הכרת היתרונות של בחירה בהקצאת מצבים One-Hot בתיכון של מכונת מצבים שממומשת עם ארכיטקטורה שמבוססת על LUTs: [1] קבלת Fan In נמוך (ותדר שעון גבוה) ו [2] היכולת לחזות מראש מהם המעברים שיוצרים את צוואר הבקבוק במהירות של המכונה והיכולת הפשוטה להקטין Fan-In על ידי פיצול מצבים ותוספת מצבים על חשבון Latency.
9.10	מכונות ישירות והתערבות ידנית בהקצאת המצבים בצורות שונות. שיטות שונות לאי איבוד היכולת להתבונן בסימולציה בשמות המצבים באופן סימבולי.
9.11	תיאור מכונה עם כניסות אוניברסאליות, כלומר כניסות שמבצעות את אותה הפעולה באופן גלובלי בכל מצבי המכונה (בדרך כלל אפשור סינכרוני או איפוס סינכרוני). יצירת קוד של מכונה כזו באמצעות שילוב הכניסות בתהליך הסינכרוני.
9.12	תיאור אלטרנטיבי של מכונת מצבים באמצעות Present-State בלבד עם: [1] תהליך סינכרוני לתיאור הפליפ-פלופים יחד עם העירור שלהם ו [2] תהליך צירופי לתיאור היציאות. היתרון של שיטת תיאור זו הוא בעיקר בתיאור מכונות ישירות שבהן התהליך הצירופי נהפך לחוטים.

חלק 10 - מערכים ורכיבי זיכרון (4 שעות)

10.1	הגדרת סוגי מידע של מערכים, כולל מערכים בלתי מוגבלים.
10.2	בדיקת אפשרויות שונות לתיאור ROM ובסופו של דבר שימוש מערך חד ממדי של אות וקטורי ואתחולו באופנים שונים כקבוע שמאוחסן בחבילה.
10.3	שימוש ב – Attributes של מערכים (left, right, low, high, range, reverse_range length) והחשיבות שלהם ביצירת תיאורים כלליים (חשוב גם לפרק הבא)
10.4	מערכים בעלי אינדקסים שאינם integer ויצירת טבלאות אמת למצבים לוגיים מרובים (למשל mvl4 ו std_ulogic) והבנת התחביר בחבילות כמו std_logic_1164.
10.5	רשומות וביצוע השמות לשדות באופנים שונים.
10.6	כללי תיאור רכיבי RAM באופן התנהגותי ובאופן דומה לדרך שבה תיארנו את ה ROM [1] כמערך חד ממדי אות של וקטורי. [2] הדגשת החשיבות של אי שימוש במשטר עבודה א-סינכרוני והעדפת משטר עבודה סינכרוני. [3] הדגשת אי חווט משותף של הדקי כניסות המידע ויציאות מידע (כנהוג במיקרו-מעבדים) והעדפת שימוש בכניסות מידע ויציאות מידע נפרדים.
10.7	תיאור רכיבי RAM שונים.
10.8	הקפדה על כללים נוספים לכתיבה לסינתזה: [1] הקפדה על תיאור חלקים צירופיים וסינכרוניים לפי השבלונות מפרקים קודמים. [2] תיאור יחידת הזיכרון כרכיב [3] בדיקה שכלי הסינתזה עושה שימוש אוטומטי במשאבי זיכרון ייעודיים (כגון: M4K,M9K) על פני שימוש במשאבי החומרה הרגילים (כלומר שימוש Logic-Cells).
10.9	תיאור רכיב מסוג DPRAM (כלומר Dual-Port-RAM) שבו יש כניסות כתובת לקריאה וכניסת כתובת לכתיבה שהן נפרדות.
10.10	שבירת הכללים ויצירת רכיבי זיכרון מסוגים שונים ובעלי מבנים מיוחדים (ושאינם נתמכים במשאבי הזיכרון הפנימיים הגדולים של הרכיב המתוכנת).

חלק 11 - פונקציות ופרוצדורות כהרחבות לשפה (4 שעות)

11.1	תיאור פונקציות (function specification) בחלק הצהרתי של הארכיטקטורה או בחלק ההצהרתי של התהליך או בישות.
11.2	שימוש במשתנים (שהם דינמיים) ופסוקי if, case בתיאור הפונקציה.
11.3	תיאור פונקציה בגוף החבילה והצהרה על הפונקציה (function declaration) בחבילה. והפיכת החבילה לברת שימוש באמצעות פסוק use.
11.4	שימוש בחבילה באופנים שונים ושימוש בפרמטרים פורמאליים ואמיתיים בשיטת Positional Association ובשיטת Named Association וכן השמטה של פרמטרים או שימוש ב open.
11.5	יצירת פונקציות עזר לוקליות בגוף החבילה.
11.6	דוגמאות של כתיבה של פונקציות בצורה כללית ושימוש ב attributes של מערכים.
11.7	פונקציות עם פרמטרים שהם אותות. קווי דמיון והבדלים בין פונקציה ותהליך.
11.8	פונקציה רקורסיבית ואי התאמתה לסינתזה.
11.9	העמסה על פונקציות ואופרטורים.
11.10	הבנת האופן שבו מועמסות הפעולות בחבילה std_logic_1164 כולל הגדרת resolution function.
11.11	פרוצדורות שמחזירות משתנה וקריאה להן מתהליך ופרוצדורות שמחזירות אותות (כתיאורי חומרה) וקריאה להן מהארכיטקטורה.
11.12	פרוצדורות שיש בהן פסוקי wait (בניגוד לפונקציות) ויכולת להיות impure.
11.13	הבדלים בין פרוצדורות ותהליכים.
11.14	שימוש בפרוצדורות ליצירת BFM (כלומר Bus Functional Models).
11.15	התבוננות בתכולה של חבילות שונות כמו std_logic_1164 או std_logic_unsigned ו std_logic_signed וחבילות אריתמטיות לוקטורים של IEEE ו Synopsys וחבילות אריתמטיות של IEEE למספרים ממשיים וקומפלקסים וחבילות אחרות והבנת האופן שבו נעשות הרחבות לשפה באמצעות חבילות.
11.16	דוגמה לתיאור אות סינוסואידלי או אות אקראי עבור Test Bench.

חלק 12 - טיפול בקבצים (4 שעות)

12.1	הסתייעות בחבילה textio לקובצי טקסט והכרת הפרוצדורות של החבילה.
	שימוש בגנרטורים שונים שקוראים טכסט ומפיקים ממנו אותות ותזמונים כולל גם בדיקה של תוצאות מצופות.
	כתיבה של תוצאות סימולציה לקובץ טכסט.
12.2	שימוש בחבילה std_logic_textio של Synopsys לקבלת תמיכה בסוגי מידע נוספים וכן עבור קריאה וכתיבה של קבצים הקסדצימליים ואוקטליים.
12.3	**שימוש בקבצים שאינם קבצי טכסט** : האופן שבו הסימולאטור מאחסן קבצים מסוגי מידע שונים. והכרת דוגמאות שונות שבהן קוראים וכותבים קבצים באופן בינארי ב VHDL גרסה 87 ו 93.
12.4	אתחול רכיב זיכרון מקובץ.
12.5	שימוש ב stdin ו stdout.

חלק 13 - חידושים ב - VHDL-2008 (4 שעות)

13.1	הצבעה על הכיוונים המרכזיים שעברו שיפור ומגמות לעתיד.
13.2	תחביר בסיסי ששופר : הערות, קבועים וקטוריים.
13.3	פעולות לוגיות מעורבות הכוללות ביטים בודדים וכן וקטורים.
13.4	אופרטורים לוגיים reduced והצבעה על יישומים אפשריים.
13.5	מערכים נוספים שהתווספו בשפה והשמות בשיטת named association ובשיטת positional association.
13.6	חבילות חדשות לפעולות חשבוניות על וקטורים.
13.7	רלציות משופרות שיש להן משמעות חומרתית.
13.8	פונקציות maximum ו minimum.
13.9	חבילות חשדות לסוגי מידע fixed (כלומר ufixed או sfixed) וביצעו פעולות בדיוק מלא (Full-Precision).
13.10	סוגי מידע וקטוריים של Floating-Point שתואמים לתקנים של IEEE בממדים שונים והשימוש בהם.
13.11	חבילות לתמיכה חלקית של VHDL-2008 ב VHDL-93
13.12	הכנסת אוסף הצהרות ביחידת context.
13.13	שימוש ב - conditional operator ויתרונות ברישום מקוצר ונוח של קודים.
13.14	פסוקי case משופרים.
13.15	השמות מותנות ונבחרות בתהליך.
13.16	שימוש ברשימת רגישות עם all.
13.17	שימוש נוח יותר בפונקציה rising_edge לסוגי מידע נוספים.
13.18	הכרת פונקציות to_string(_) לסוגיהם כולל גרסאות Hex ואוקטלי.
13.19	ערוב ביטויים (expressions) בחיווט.
13.20	שינוי ב modes שיוצרים נוחות.
13.21	שיפורים בפסוק generate.
13.22	גישה ל - External-Names וגישה באמצעותם לאותות פנימיים בתכן וכן שימוש בהם לאילוץ מצבים פנימיים (force ו release) והדגמה למשל של אילוץ מכונת מצבים להיכנס למצב שגוי ובדיקת התאוששות שלה.
13.23	הפרדת מערכות באמצעות שימוש ב driving value ו effective value.