סיכום פגישה שנייה

* ארכיטקטורה כללית של ILA- שירטוט, דרישות כלליות, שמירת מידע. איך הדברים עובדים מה generic ומה registers.
* מסמך אפיון- (מצגת) להתחיל קוד, בלוק generic
* Generics- לכל צריבה (לא משתנה)
* Registers- תוכנה מעדכנת אותם (לא רק צריבה)
* בחלק הראשון של הפרויקט- טריגר אחד.
* Whishbone slave- כותב לרגיסטרים (יוצא ממנו לרגיסטרים) RX מקלף חבילת מידע שמגיעה על קו UART רוצים להעביר ל ILA דרך ממשק whishbone.
* Whishbone master- שולח את המידע המוקלט ל TX ומשם חזרה לתוכנה.
* יחידה שמפענחת את ה whishbone slave ושמה את המידע ברגיסטרים המתאימים. (כתיבה לרגיסטרים)
* Command registers- רגיסטרים במערכת: trigger type-rise,fall,'0','1' כל רגיסטר הוא 8 bit, כל bit הוא טריגר אחר (רק אחד פעיל בזמן נתון)

-רגיסטר שני- trigger position: מקליטים מתחילת הטריגר\ מקליטים כל הזמן וברגע שהגיע טריגר מקליטים עוד חצי וחצי לפני הטריגר\מקליטים מלפני הטריגר.

-רגיסטר שלישי- start- כל כתיבה של ערך אליו נזהה גישה לרגיסטר והמערכת מתחילה לעבוד.

* Read only registers סטטוסים של מצב מערכת, חיוויים מה VHDL – נצטרך לדבג (דוגמא) מזמן עליית טריגר ועד ירידת טריגר. (כלומר מונה)
* להוסיף עם הזמן עוד רגיסטרים כאלו, תלוי בנו- נותן אפשרות דיבוג יותר טובה.
* Generics:

-כמות הסיגנלים להקלטה- רוחב ה inputs

-כמה רוצים להקליט- משפיע על גודל (עומק) ה RAM.

-מספר הטריגרים (בחלק ראשון יהיה אחד)

* Reset polarity – 0 או 1 (פעיל)
* Data width of whishbone
* Adrass width of whishbone
* Core: בלוק שמטפל ברגיסטרים ה input נכנס אליו. הוא כותב ל RAM כמה לכתוב וכו. מסיים לכתוב מודיע ל read controller- קורא מה RAM ומעביר את המידע דרך ה WBM.

דיאגרמת בלוגים שורטטה בנוסף, לא נשרטט אותה כאן.