שאלות:

האם לממש RAM קטן יותר או להשתמש ביחידה הבסיסית אפילו אם ההקלטה קטנה יותר (ביחידה בסיסית 4096 ביטים להקלטה)

יש GENERIC-ים לטובת הגודל והרוחב של ה- RAM הבסיסי.

האם לקבוע גודל מקסימלי של הקלטה או שהמשתמש יכול להקליט כל מספר ביטים שהוא רוצה?

יש GENERIC, והכול צריך להיות תלוי ב- GENERICS-ים.

איך מייצגים את הבעיה כאשר צריך לשלוח את עומק ההקלטה בין היחידות במערכת (לדוגמא רוחב הפס בין READ-WRITE -> מצריך שליחת כתובת, תלוי ברוחב ההקלטה)

הכול צריך להיות תלוי ב- GENERIC-ים, ואם חסר GENERIC-ים שלא חשבנו עליהם אז יש להוסיף. ה- USER צריך לקבל גמישות מלאה.

האם לRAM יש כניסת שעון וRESET

כן.

* כרגע יש קצת בלאגן לגבי מי, איפה ואיך מחושבת כתובת הכתיבה לRAM (כאשר מגיעים האותות, הכתובת צריכה להתאפס לתא הראשון ואז פשוט להעלות את הכתובת בהתאם למספר האותות המוקלטים). כמובן שנתקן ונסנכרן הכל לאחר תשובתך, כרגע אצלנו יש גם קצת חוסר התאמה רצינו רק את חוות דעתך לפני שאנו משנים, חישוב הכתובת של המידע **הנכנס** כרגע מבוצע בWRITE CONTROLLER לפי השיטה שהצענו למעלה.
* ה- write controller  אחראי לשנות ולחשב את כתובת הכתיבה של ה- RAM (addr\_in), את ה- CONTROL שאומר שה- DATA לכניסה של ה- RAM חוקית (din\_valid), והוא מעביר גם את ה- DATA לכתיבה ל- RAM (שמורכבת מה- TRIGGER וה- INPUT DATA להקלטה לאחר שהוא מסדר אותם בהתאם ל-GENERICS).
*          האם לrdcntr תפקידים נוספים? כרגע הוא רק מקבל את המידע היוצא מהRAM ופשוט מעביר אותו החוצה דרך WBM (נראה כאילו אפשר להחליפו ברגיסטר פשוט) או שהוא בעצם מכין את המידע היוצא לפורמט ש WBM יכול לקרוא? אם נדרש שינוי בצורת המידע, כיצד אנו מבצעים זאת?
* ה- read controller צריך לקבל הוראה מתי להתחיל לעבוד, ואז בהתאם לקונפיגורציות, האן קורא מידע מה- RAM-ים, ואז מוציא אותו על גבי פרוטוקול ווישבון.
* אין לכם בכלל state machines בכל האיורים. שלחתי לכם מלא דוגמאות. לדוגמא כאן: נמצאים במצב IDLE, מחכים לטריגר להתחיל לעבוד, ואז עוברים למצב של קריאה מה- RAM-ים, מפעילים COUNTER כמה לקרוא, ואז מגיע מידע מה- RAM ומשם ממשיכים למצב של להוציא את המידע החוצה בפורמט הנכון בפרוטוקול ווישבון.
* הערות נוספות:
* 1.      ברגיסטרים אמור להיות יח' שמפענחת את הכתובת שמגיעה מהווישבון וכל ה- CONTROLLE-ים שלה ובהתאם כותבת לרגיסטר המתאים.
* 2.      ב- write controller המידע לא נשמר ב- shift register אלא נשמר כל הזמן ב- RAM גם כשלא מגיע הטריגר. ברגע שמגיע הטריגר אז מתחיל חישוב בהתאם לקונפיגורציות. לדוגמא: אם הטריגר הוא בסוף, אז סיימנו כי כבר רשמנו כל מה שצריך. אם הוא באמצע אז שמרנו כבר חצי מה- DATA ונותר לשמור עוד חצי. אם בהתחלה אז כל מה ששמרנו לא רלוונטי ומעכשיו יש לשמור את כל המידע (בהתאם לעומק הנדרש).
* 3.      ב- SVN בספריית CORE יש קבצים שונים של General Block Diagram. אמור להיות רק אחד. כך גם לגבי עוד קבצים.