# הצלחה של 100%: וריפיקציה של מעבד בקוד פתוח ב-ASIC מסחרי

חוסר בשלות של התכנון, חוסר תמיכה, רישיון ואחריות הם אלו אשר גורמים ל־Open Code IP לא להיות בשימוש. ישנם מצבים, בהם יש פתרון אשר מאפשר שימוש של Open Code IP בתכנון ASIC מסחרי

**> דקלן סטאוטון ופול פורלינג Silicon Software Systems**
**תרגום: עדי כתב, כאל כתב אסוסיאטס**

התכנה בקוד פתוח (Open Source) מאד פופולרית במקרה של מערכת ההפעלה Linux. היום זה כבר די מבוסס ורבות החברות המסחריות אשר משתמשות בקוד זה. במקרה של תכנון רכיבים (ASIC) השימוש בקוד פתוח Open Core IP Verilog/VHDL אינו נפוץ וממריא באיטיות בשל סיבות טובות.

חוסר בשלות התכנון, חוסר תמיכה, ריישיון ואחריות הם אלו אשר גורמים ל־Open Code IP לא להיות בשימוש ואף לא לעמוד על הפרק, כאשר חברה מחליטה על סוג ה־IP בשימוש.

אף על פי כן, ישנם מצבים, כמו במקרה של המעבד LEON3, בהם יש פתרון אשר מאפשר שימוש של Open Code IP בתכנון ASIC מסחרי.

במאמר זה אנו נתאר את ניסיוננו בשימוש במעבד LEON3 ברכיב ASIC מסחרי בטכנולוגיות 0.13un, ובמיוחד את המדרג בו נקטנו לוודא ביצועים. היתרונות בשימוש ווריפיקציה של ה־Leon3 הסתברו כבחירה מוצלחת עבור התכנון שלנו.

**חוסר בשלות התכנון, חוסר תמיכה, רישיון ואחריות הם אלו אשר גורמים ל־Open Code IP לא להיות בשימוש ואף לא לעמוד על הפרק**

כהקדמה נאמר כי במאמר זה רצינו לתאר את ניסיוננו בשימוש ווריפיקציה של מעבד Leon3 ברכיב מסחרי. אנחנו נבחרנו כספק משנה של תכנון הרכיב מהגדרתו ועד Netlist שהועבר ליצרן הסיליקון (FAB). הווריפיקציה בוצעה על ידינו אך גם ללקוח היה חלק בווריפיקציה שכן כתיבת תוכנת C הייתה באחריותו. בחירת המעבד בוצעה לאחר אנליזה של מספר מעבדים על ידינו.

ישנם כמה פרמטרים שיש לקחת בחשבון כאשר רוצים לבחור מעבד עבור פיתוח System On Chip. ברבים מהמקרים ארכיטקטורת המעבד משפיעה מאד על בחירת המעבד. בפיתוח SOC הזה לא היו מגבלות מצד הארכיטקטורה ולא מצד פיתוח התוכנה. למרות זאת, ה־ASIC תוכנן להיות מיוצר בכמויות גדולות ולפיכך גודלו היה חשוב ומכאן גודל המעבד גם.

כפי שניתן לראות מדיאגרמה 1 (Figure 1), ארכיטקטורת רכיבים היא אופיינית לפיתוח SOC.

באופן כללי הדרישות מה־CPU היו:

ביצועים – בתחילה הדרישות מביצועי המעבד היו נמוכות אך הם עלו עם הזמן. היו כמה דרישות Real Time מחמירות.

תמלוגים (Royalty) – כמו בכל רכיב שמתוכנן לייצור בכמויות גדולות קיימת >

> חשיבות גדולה לעלות התמלוגים הנדרשת וכמובן רצוי מעבד שאינו מחייב בתמלוגים.

ניהול ומצב משתמש - המעבד חייב לתמוך בהפעלה של קוד אשר נכתב ע"י גורם שלישי עם גישה מינימלית לחומרה.

צריכת הספק לא הייתה פרמטר עיקרי בפרויקט זה, אבל צריכת ההספק של ה־Leon3 הייתה נמוכה ועומדת בציפיות.

ה־Leon3 פותח ב־VHDL בארכיטקטורת Sparc V8 (IEEE 1754). המעבד הוא קופי־גורבילי, ליבה 32BIT עם אפשרות בחירת גודל Cache, אפשרות של תוספת מאיץ לעיבוד בשיטת הנקודה הצפה, מוניטור לפיתוח תוכנה, ממשק AMBA AHB ות־מיכה במעבדים מקביליים נוספים. כמעט כל הפונקציות של ה־Leon3 ניתנות להג־דרה באמצעות מימשק גרפי (GUI) מבוסס VHDL - קובץ של קבועים המתאר את הגדרות המעבד (ראה Figure2).

הרישיון של Leon3 הוא מבוסס (GPL) GNU General Public License או על רישיון מסחרי. רבות האוניברסיטאות והמכונים אשר משתמשים במעבד לצרכי מחקר בשל הרישיון הפתוח.

סביבת פיתוח תוכנה מבוססת על קומפ־לייר GNC C/C++ זמינה עבור ה־Leon. סי־מולטור לפיתוח תוכנה קיים גם הוא אך אין אפשרות לבצע סימולציה עם ה־RTL. כלי פיתוח התוכנה וה־IP היו זמינים מ־Gaisler Research.

הפעולה הראשונה בהתאמה של Leon3 לאפליקציה שלנו הייתה זיהוי של הרכי־בים שאנו צריכים לתכנון שלנו מתוך המ־גוון המגיע יחד עם המעבד (כולל bridges, interface, peripheries ועוד).

הרכיבים החיוניים לנו היו ליבת המעבד עצמו, Integer Unit, cache Controller, AHB I/F. יצרנו Test Bench כדי לבדוק את תפקוד כל האלמנטים הללו ללא שאר הרכיבים המסופקים יחד עם ה־Leon.

השלב השני היה יצירת bridge בין Leon AHB I/P ו־Proprietary bus לזיכרון DRAM וב־Peripherals. למרות ש־AHB לא היה בשימוש, בתכנון היה יתרון גדול שכן הצוות הכיר את ה־BUS. בשלב מאוחר יותר הוספנו מכפיל ומחלק בחומרה ומערך של Register File (ה־Leon תומך במגוון של זיכרונות ומערכי Registers של מגוון יצרני סיליקון ו־FPGA).

ההכרות עם המעבד הייתה חלקה והקוד עצמו כתוב במבנה קבוע, דבר שהקל על הבנת ה־Source Code.

השלב השלישי היה שינויים ב־CPU, בכדי לעמוד בדרישות האפליקציות. הי־רכיית רכיבי ה־CPU, ראה Figure 3, הותא־מה במקביל לפיתוח הרכיב כולו.

ראשית, ה־Cache Controller מותאם לשורת 256bit wide על ידי קריאה של 32bit בכל פעם. בתכנון זה השתמשנו בקריאה בת 250bit בכל קריאה. ע"י מספר שינויים היה אפשרי לבצע את ההתאמה ל־256bit. כמו כן ביצענו שינויים נוספים ביחידת ה־Memory Management Unit.

השלב הרביעי היה בניית אסטרטגית ווריפיקציה לרכיב. האסטרטגיה שנבחרה מבוססת BOTTOM-UP כפי שמתואר ב־Figure 4 האסטרטגיה נבחרה בעצה אחת יחד עם קבוצת התכנון וקבוצת הווריפי־קציה, כאשר הגדרנו את הפונקציות אותם אנו רוצים לבדוק. נבחרה גישה משולבת להלן:

1) בלוקים באיור 1 כלומר TIMER, GPIO ייבדקו ע"י קוד HDL ברמת הבלוק. הדרישה כאן היא בדיקה ברמה של 100%. בדיקת מודולי ה־bus ייבדקו ע"י ה־CPU באמצעות כתיבה וקריאה.

2) קבוצה קטנה אצל הלקוח, כתבו test case בשפת C++. וקטורי I/O נקלטו מה־testcase ונבחנה יכולת ה־RTL. כל בלוק בנפרד נבדק מול הספציפיקציות.

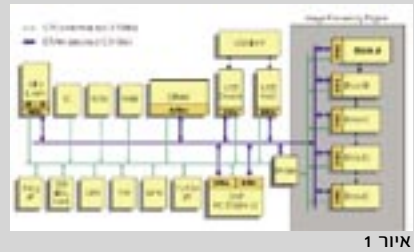
3) מגוון יחידות המשנה של ASIC, כמו LEON ויחידות המשנה שלו נבדקו באספ־קט הקישוריות ביניהם.

4) לבסוף, בוצעה אינטגרציה של יחידות הרכיב. כל הקישוריות שלא נבדקו עד כה הושלמו בשלב זה. נבדקה האינטרקציה בין כל יחידות הרכיב כדי להוכיח ביצועים ברמת הרכיב.

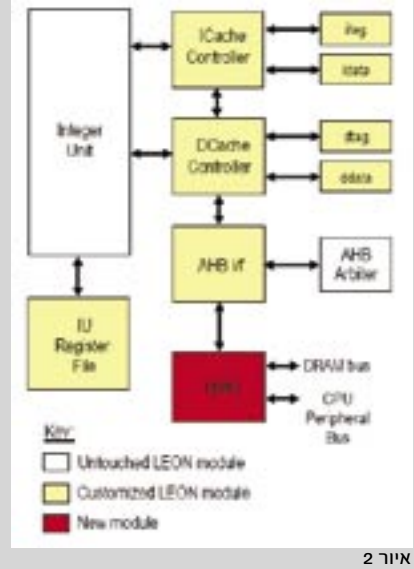
5) ווריפיקציה של LEON בוצעה לפי גישת bottom-up תוך כדי האינטגרציה. הווריפיקציה של ה־LEON נחלקת לשני שלבים: ווריפיקציה של רכיבי ה־Integer Unit CPU MMU ושאר הרכיבים השונים. הווריפיקציה של הרכיבים הללו בוצעה ע"י תוכנת "C" שהופעלה על ידי ה־LEON. ביצוע של כמות גדולה של Test Case על גבי ה־RTL מאפשר רמת בטחון גבוהה מאד של ווריפיקציה. בנוסף ובמקביל, מימוש על גבי FPGA מאפשר כתיבת ווריפיקציה של דרייברים ואפליקציות תוכנה נוספות. מימוש ואמולציה על גבי FPGA היה דרוש לבחינת משאבי הזיכרון. אמנם קיים סי־מולטור ל־LEON המאפשר כתיבת תוכנה ללא חומרה, אך המימוש על גבי FPGA איפשר הפעלת חלקים נוספים ברכיב עצמו ומחוצה לו.

מלבד החיסכון בכסף רב, השימוש ב־LEON3 היה נוח וקל תוך אפשרות להגדלת הביצועים לפי הצרכים העתידיים.

כמו בכל תכנון, דרישות הביצועים גדלות עם התכנון והיה צורך לאפשר גמי־שות בתדר (כדי להשיג יותר כח חישוב), וזיכרון והאצת ביצועים. מאחר וכל >

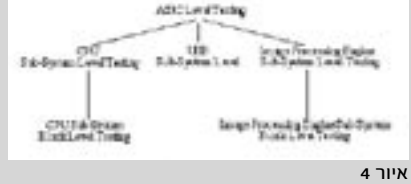


איור 1




איור 2

איור 3



איור 4

ה־Source Code היה זמין עבורנו, יהיה באפשרותנו להוסיף ולשנות תכונות למעבד עצמו וליחידות הסמך מהשלב הראשוני ללא דיאלוג או משא ומתן עם הספק; סביבת ה־GUI המסופקת יחד עם המעבד עזרה מאוד והקלה את המאמץ הזה.

הגישה ל־Source Code והחופש לשנותו מהווה כלי מוצלח מאוד להתאמת הביצועים ולא פחות חשוב, מתאפשרת גישה לפרטים בזמן הבדיקות והווריפיקציה. ללא החופש הזה היה קשה יותר לבצע שינויים והסיכוי לטעויות היה גדול.

ה־LEON תומך במגוון טכנולוגיות סיליקון (הכולל FPGA). הוספת טכנולוגיה חדשה לא היה קשה לנו כלל. הקוד עבר סינטזה ללא בעיות מיוחדות וכמו־כן, התכנון הפיזי (Layout) עבר גם הוא.

התמיכה המסחרית שניתנה על ידי Gaisler Research הייתה מעולה. היה הממשק הישיר למהנדסים שפיתחו את

**כמו בכל תכנון, דרישות הביצועים גדלות עם התכנון והיה צורך לאפשר גמישות בתדר (כדי להשיג יותר כח חישוב), וזיכרון והאצת ביצועים**

הליבה. השימוש בקבוצת ווריפיקציה חיצונית לארגון שלנו הגבירה את איכות התכנון והגבירה את הווריפיקציה שכן התכנון נבחן על ידי יותר ממהנדס אחד.

מעבד ה־LEON שהוא חדש יחסית, עבר את כל שלבי הווריפיקציה והמימוש בפעם הראשונה, מה שהוסיף מידת בטחון רבה מאוד.

בחירת מעבד הוא אחת ההחלטות הקשות בפיתוח של רכיב SOC. כאשר רוצים פיתוח חלק ללא בעיות, השימוש ב־LEON יהווה שיקול משמעותי, כשהוא ניתן ללא כל שינוי או התאמה. השימוש ב־Soucre Code והאפשרות לשינויים מאפשר התאמה לדרישות ללא הוספת לוגיקה חיצונית שבדרך כלל היא מסובכת. החשיבות של Open Source IP יגדל קרוב לוודאי לאחר הודעת SUN על כוונה לאפשר גישה לקוד שלה.

ווריפיקציה בגישת bottom-up יחד עם קבוצת ווריפיקציה חיצונית לארגון הניבה תוצאות טובות מאד. תוספת הבדיקות על גבי ה־CPU היוותה בדיקה של פקודות המעבד, בדיקת יחידות הסמך של המעבד והגביר את הביטחון לפני יציאה ליצור סיליקון.

KAL מייצגת את Gaisler Research בישראל. ניתן לקבל את המאמר השלם בפניה אל info@kaltech.co.il