UVM VERIFICATION

קורס: שפת תכנון חומרה ווריפיקציה

מגישים: בר אליס , אורי כהן , חיים עוזר , שון פזרקר | מרצה: ד"ר אביחי אהרון

מבוא
UVM VERIFICATION

מה זה UVM?

UVM זה ראשי תיבות של Universal Verification Methodology

היא שיטה סטנדרטית בתעשיית המוליכים למחצה לאימות digital

designs, במיוחד מעגלים משולבים ומערכות-על-שבב - SoCs .

הוא בנוי על גבי שפת System-Verilog שהיא שפת תיאור ואימות של החומרה

וניתן לשלב רכיבים רבים בקלות בתהליך design verification .

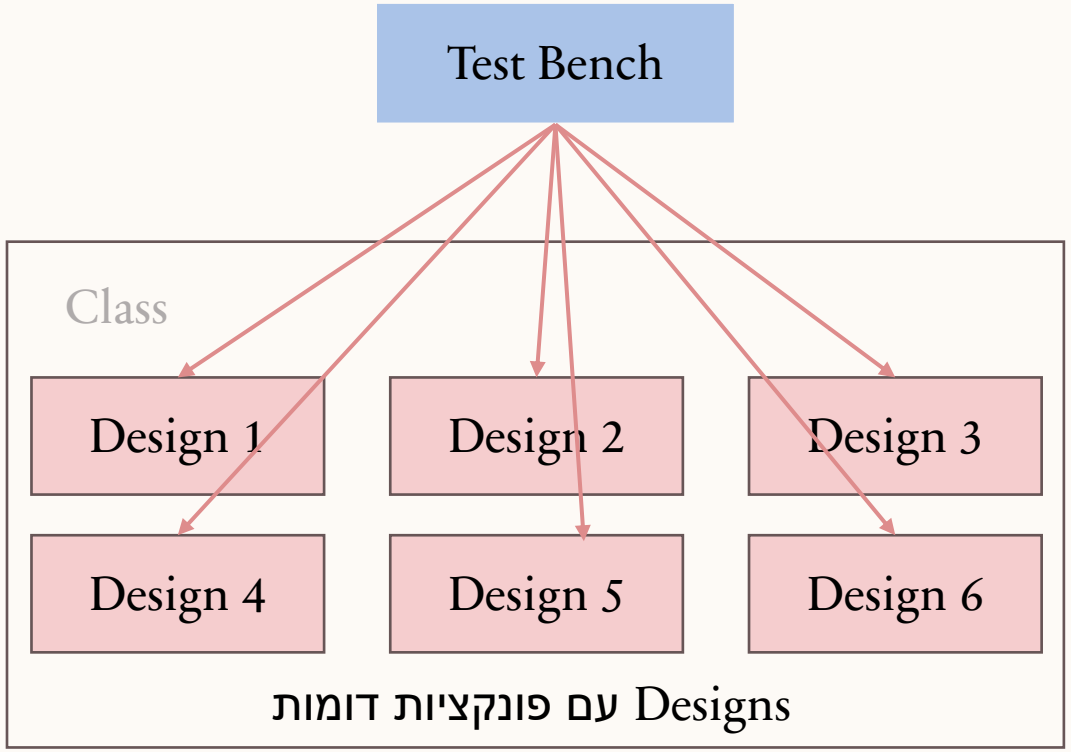
משמע UVM זה סט חוקים של "מה? מתי? ואיך?" לבדוק רכיבים שונים בעזרת

test bench ממוחזר.

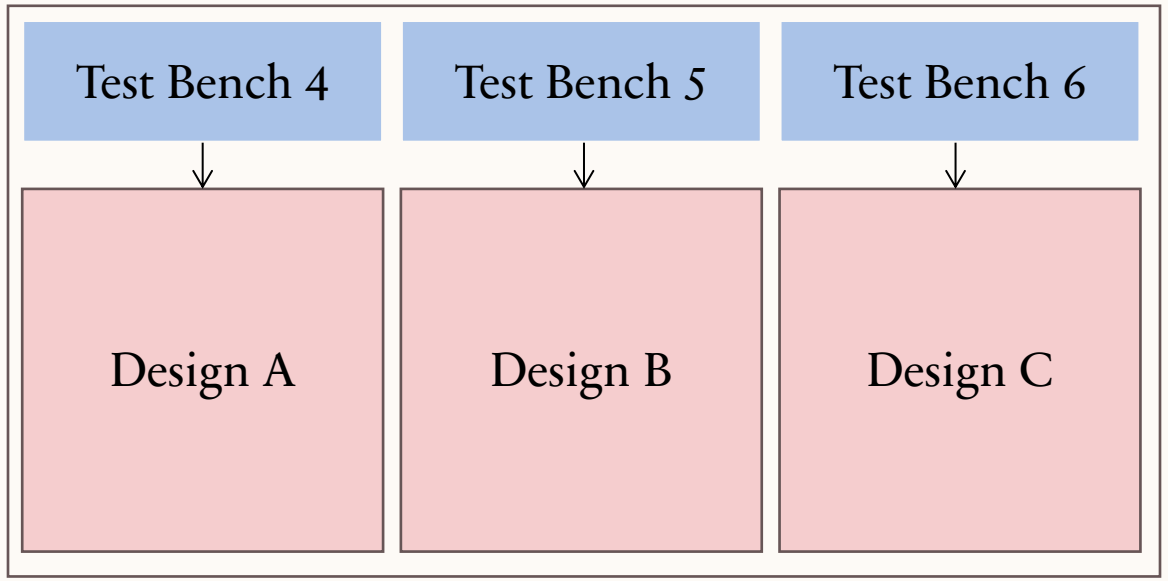
למה צריך UVM

ולא OVM או VMM או כל מתודולוגית VERILOG מבוססת TESTBENCH אחרת

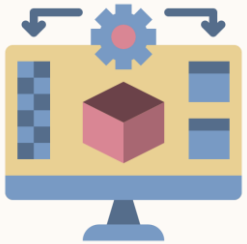
בUVM ניתן להשתמש ב testbench בודד למספר Desings
עם פונקציונליות דומה



כל מתודולוגיה אחרת עובדת עם סט חוקים זהה ל testbench
שונים על מנת ליצור אחדות בבדיקות



UVM ARCHITECTURE

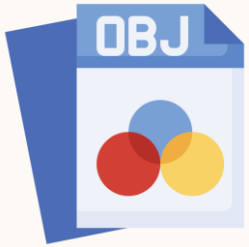


Objects

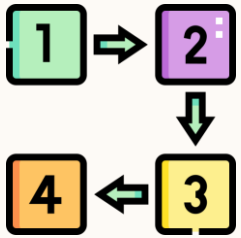
- Sequences
- Packet / Sequence Item

Components

- Top / Test bench
- Test
- Environment
- Agent
- Sequencer
- Driver
- Monitor



UVM- OBJECTS



Sequences - אובייקט רצף (Sequence) ב – UVM מייצגים רצף של טרנזקציות* או אירועים המוחלים על ה- DUT או נבדקים בEnvironment. הם שולטים ביצירת גירוי ובטיפול בתגובה, ומאפשרים תרחישי אימות מובנים וניתנים לשימוש חוזר.

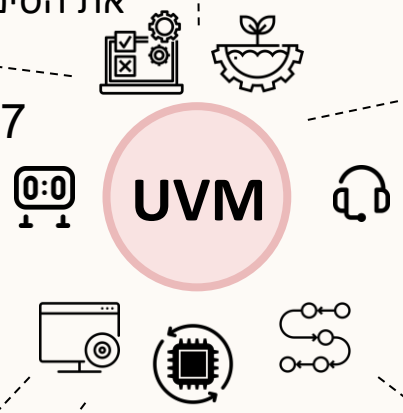
Packet - אובייקטים מסוג Packet, הידועים גם בשם Sequence Items, מקיפים טרנזקציות* בודדות או פאקטות של נתונים בתוך רצף. הם מכילים מידע כגון שדות נתונים, אותות בקרה ותכונות תזמון הרלוונטיות לטרנזקציה שנוצרת או נבדקת. הפאקטות מנוהלות על ידי ה- Sequencer.

**טרנזקציה ב- UVM היא מחלקה עם מאפיינים עבור הסיגנלים, כגון כתובת ונתונים, ומציג מידע נוסף כגון שגיאות או עיכובים.*

UVM- COMPONENTS

1. **Test-** רכיב ה- Test הוא הרכיב Top-Level השולט בביצוע הבדיקה. הוא מציג את הסביבה, הרצפים ורכיבים אחרים, ומגדיר את ה- Test Flow. רכיב הבדיקה גם מגדיר את התצורה של הבדיקות ומתחיל את הסימולציה.

2. **Environment-** מכיל רכיבי ורפיקציה מרובים לשימוש חוזר. היא מגדירה את תצורת ברירת המחדל שלהם ואת האופן שבו הם מקיימים אינטראקציה זה עם זה.



3. **Agent-** מכיל את ה Sequencer, Monitor וה- Driver. הAgent יכול להיות אקטיבי (מקיים אינטראקציה עם הDesign) או פסיבי (דוגם סיגנלים מה DUT ע"י תרגום סיגנלים לטרנזקציות).

4. **Sequencer-** רכיב אקטיבי שולט בזרימת הטרנזקציות בין רצפים לדרייבר. הוא מייצר ושולח טרנזקציות לדרייבר.

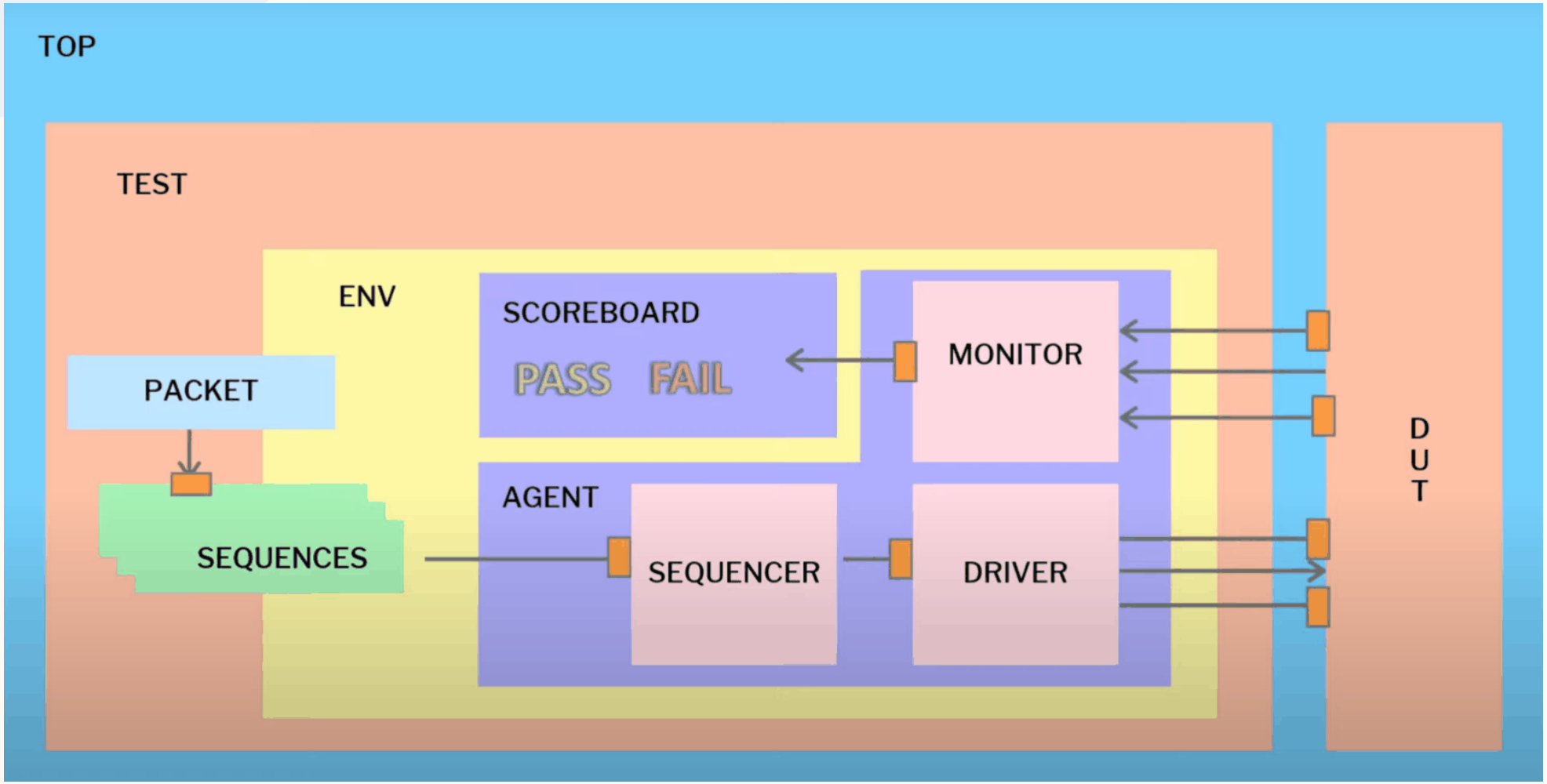
5. **Driver-** רכיב אקטיבי שעובד עם ה interface*, בסוף מתרגם טרנזקציות לסיגנלים.

6. **Monitor-** המוניטור לוקח סיגנלים מה DUT באמצעות ה- interface ומתרגם אותם לפורמט של פאקטות.
הפאקטות מועברות לרכיבי UVM אחרים כמו ל Sequencer או ל Scoreboard.

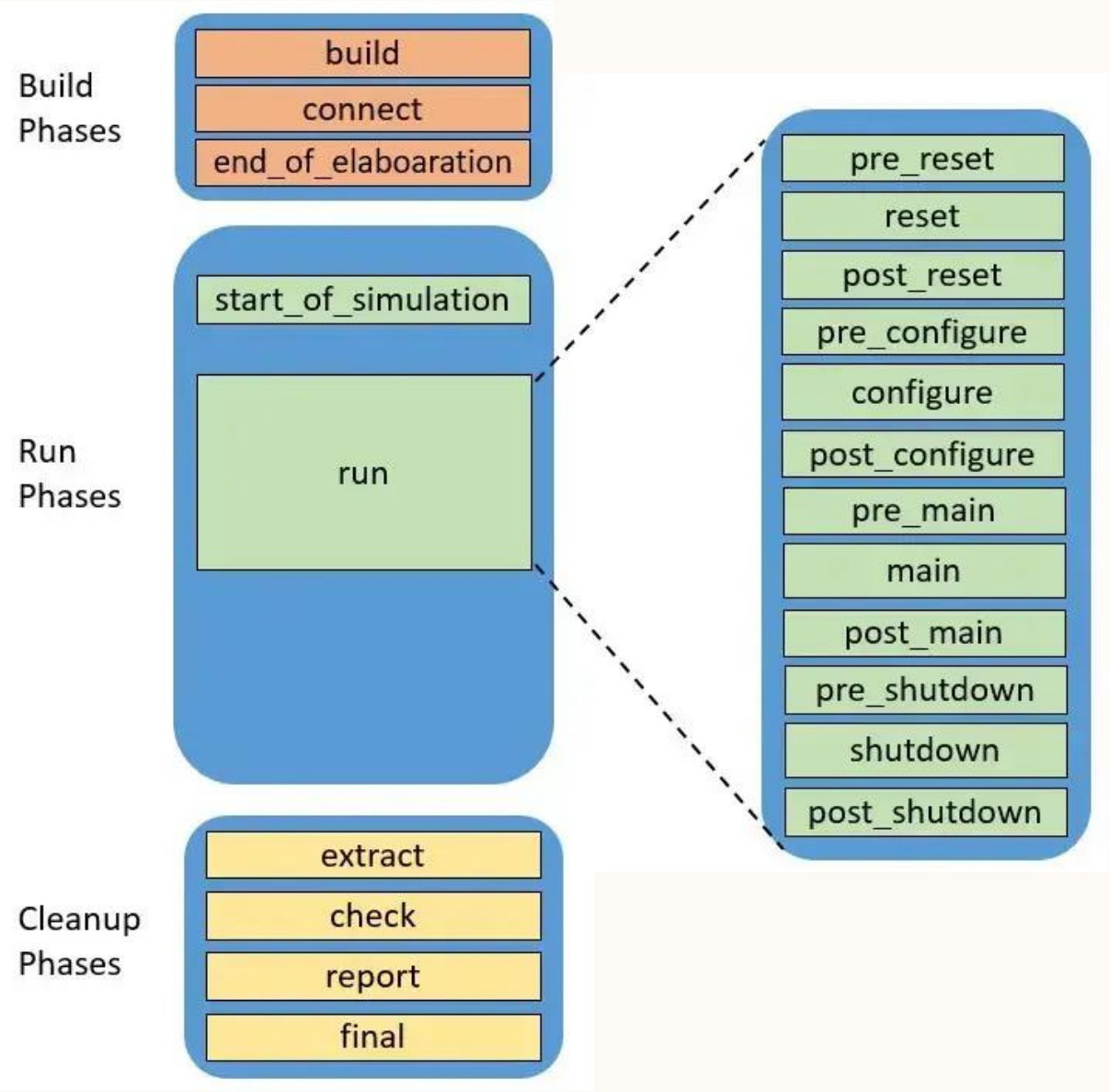
7. **Scoreboard-** רכיב ורפיקציה שבודק את הפונקציונליות של הDesign. הוא מקבל טרנזקציות מהמוניטור ובודק אם הפלט תואם לתוצאה הצפויה.

Interface - מתייחס לחיבור סטנדרטי או פרוטוקול תקשורת בין רכיבים או מודולים שונים בתוך מערכת.

UVM FLOW


TOP
TEST
ENV
SCOREBOARD
PASS
FAIL
MONITOR
PACKET
AGENT
SEQUENCES
SEQUENCER
DRIVER
DUT

מחזור החיים של הרכיבים בUVM



לרכיב UVM שלושה שלבים עיקריים:

1. שלבי הבנייה מגדירים את הסביבה, כוללים יצירה והגדרה של רכיבים.
2. שלבי הריצה מבצעים את תרחיש הבדיקה, מפעילים גירוי ובודקים תגובות.
3. שלבי הניקוי משחררים משאבים ומבצעים פעולות אחרונות לאחר סיום הבדיקה.

דוגמאות לשימושים ב UVM בתעשייה

חברת intel השתמשה ב- UVM בורפיקציה מעבדי - Intel Core מעבדים מרובי ליבות על ידי יצירת רכיבי ורפיקציה UVM הניתנים לשימוש חוזר כמו מנהלי התקנים, צגים ולוחות תוצאות, הם השיגו ורפיקציה יעיל של פונקציונליות מעבדים שונים, כולל פרוטוקולי קוהרנטיות מטמון וורפיקציה לארכיטקטורת ערכות הוראות.

חברת NVIDIA השתמשה ב-UVM בורפיקציה עיצובי הממשק המהיר שלה, כגון אלו המשמשים ביחידות עיבוד גרפיות (GPU) ופתרונות רשת של מרכזי נתונים, מה שמבטיח ורפיקציה לפרוטוקול ונכונות תפקודית.

חברת .Analog Devices Inc (ADI).
מיישמת UVM בורפיקציה של מעגלים משולבים של אותות מעורבים, כולל ממירים אנלוגיים לדיגיטליים ולוגיקת בקרה דיגיטלית, תוך מינוף רכיבי ורפיקציה הניתנים לשימוש חוזר לצורך ורפיקציה מקיף.

חברת Qualcomm השתמשה ב-UVM בורפיקציה של תכונות ניהול צריכת חשמל עבור עיצובי System-on-Chip (SoC) שלה, תוך שילוב UVM עם טכניקות סימולציה מודעת לצריכת חשמל לורפיקציה מצבי הספק דינמיים ומעברי תחום הספק.

דוגמה לבדיקת רכיב ALU באמצעות UVM

בדוגמה הבאה אנחנו נציג שימוש של UVM Verification על רכיב 8bit ALU.
ל- ALU הבא הגדרנו את התנאים הבאים:

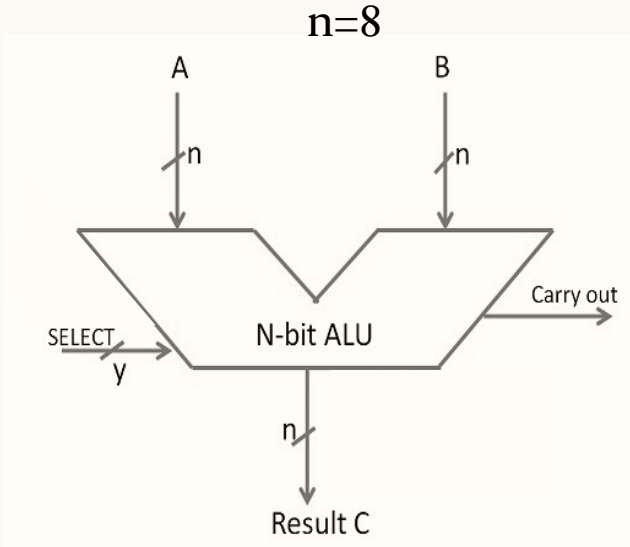
- הALU פועל בעליית שעון והמוצא משתנה בעת פעימת השעון הבאה.
- הALU בעל כניסת איפוס אסינכרונית הפעילה בגבוה (reset=1).
- לALU ארבעה מצבי הפעולה הבאים: חיבור, חיסור, כפל וחילוק. בעל אפשרות הוספת פעולות אריתמטיות נוספות - Reserved. (OP Code4 bit כלומר עד 16 פעולות בסה"כ).

Port Name	Input/Output	Size
Clock	Input	1 bit
Reset	Input	1 bit
A	Input	8 bits
B	Input	8 bits
ALU_Sel	Input	4 bits
ALU_Out	Output	8 bits
CarryOut	Output	1 bit

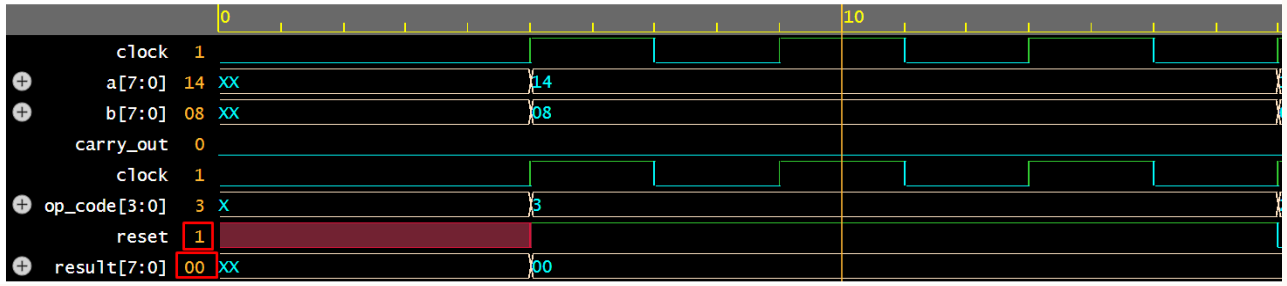
ALU_Sel	Operation
4’b0000	A + B
4’b0001	A - B
4’b0010	A * B
4’b0011	A / B
4’b0100 – 4’b1111	Reserved

דוגמה לבדיקת רכיב ALU באמצעות UVM

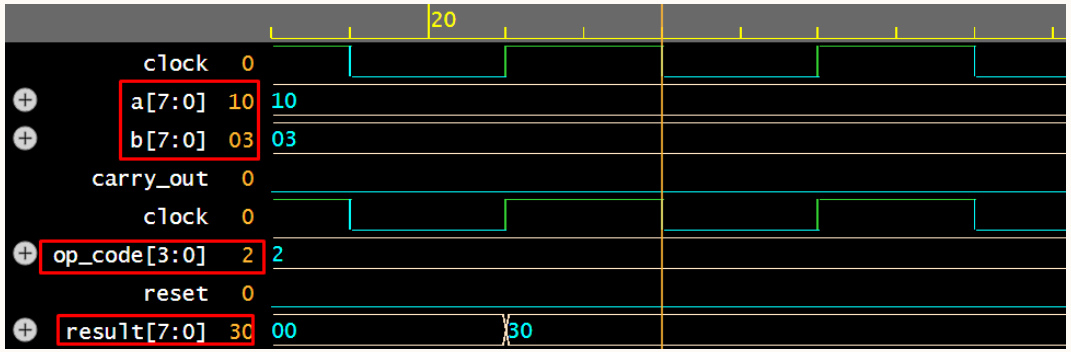
קישור לקוד



דיאגרמת זמנים WAVE FORM

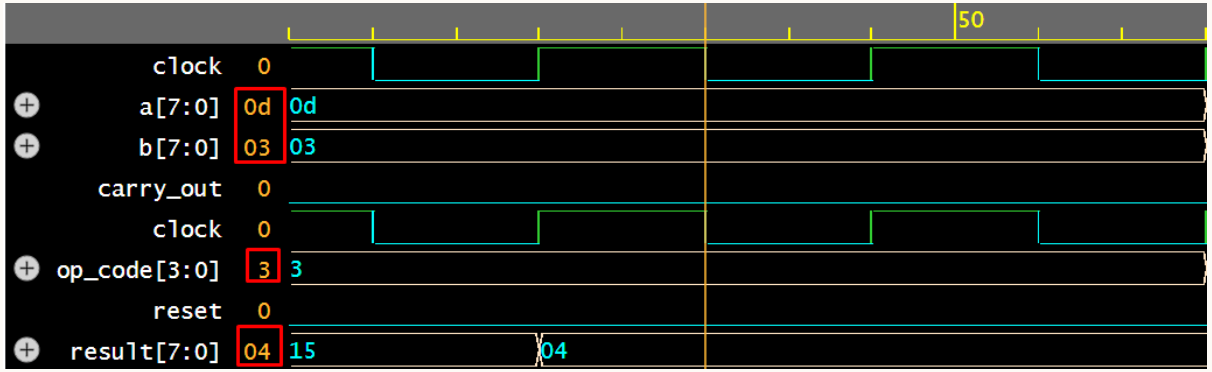


בדיקה עבור הReset, התוצאה תהיה 00 hex.

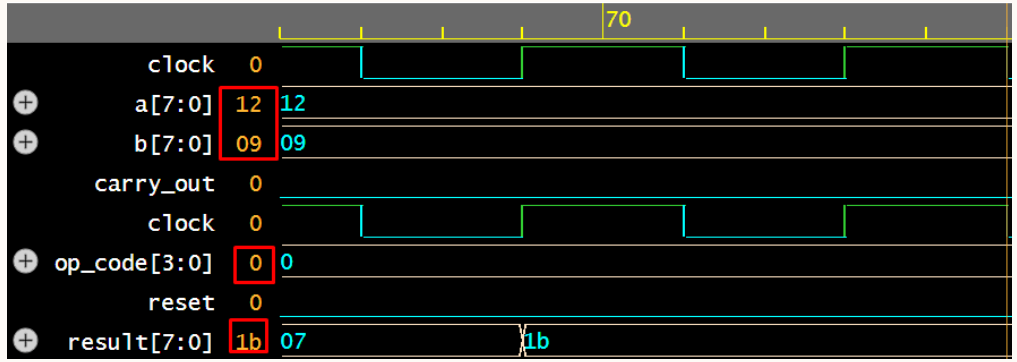


כאשר אנו ב- OP Code 0010 המגדיר פעולת כפל, לכן עבור A=10hex, B=03hex נקבל Result = 30hex. כפי שהגדרנו, את התוצאה נקבל רק בפעימת השעון הבאה.

דיאגרמת זמנים WAVE FORM



כאשר אנו ב- OP Code 0011 המגדיר פעולת חילוק, לכן עבור A=0dhex B=03hex, נקבל Result =4. כפי שהגדרנו, את התוצאה נקבל רק בפעימת השעון הבאה.



כאשר אנו ב- OP Code 0000 המגדיר פעולת חיבור, לכן עבור A=12hex, B=09hex נקבל Result =18+9=27dec = 1Bhex

סיכום ומסקנות

UVM היא שיטת עבודה סטנדרטית לdesign verification ומערכות-על-שבב (SoCs) בתעשיית המוליכים למחצה. הוא מספק מסגרת ליצירת רכיבי Testbench מודולריים הניתנים לשימוש חוזר שניתן לשלב בקלות בתהליך verification design. חלק ממרכיבי המפתח של UVM כוללים רכיבי Testbench שהם דרייבר, מוניטור, לוחות תוצאות וAgent.

מסקנות מUVM:

- **שימוש חוזר:** הדגש של UVM על מודולריות ושימוש חוזר מאפשר למהנדסי ורפיקציה לפתח ספרייה של רכיבים גנריים כמו רצפים, Agents, דרייברים ורכיבים אחרים.
- **Scaleability:** שיטת UVM משפרת את הסקיילאביליות, ומאפשרת התאמה קלה לדרישות הפרויקט המשתנות.
- **יעילות:** UVM מייעלת את תהליך הוריפיקציה, מקדמת פרודוקטיביות ומבטיחה Testbench ניתנים להתאמה.
- **מודולריות:** המתודולוגיה מתוכננת כרכיבים מודולריים (river, Sequencer, Agents, Env וכו') וזה מאפשר שימוש חוזר ברכיבים פשוטים (ALU, Counterm, Full Adder) ומורכבים (SoC/Chip).
- **הפרדה של ה- Test מה- Testbenches:** טסטים של רצפים נשמרים בנפרד מההיררכיה של ה Testbench בפועל, ומכאן שניתן לעשות שימוש חוזר ב Stimulus על פני פרויקטים. לדוגמה, יהיו כמה קלאסים של Test אך כולם ירוצו באותו ה Testbench.
- **Design Pattern (Factory):** זה מפשט את השינוי של רכיבים בקלות. יצירת כל רכיב באמצעות קונסטרקטור מאפשרת לעקוף אותם בבדיקות או בסביבות שונות מבלי לגעת בקוד. (כיוון שיש הורשה של הבדיקות ממחלקת האב)

THANK YOU