

פרויקט מסכם – 4 BIT - ALU

מבוא לתכנון מעגלי VLSI

מגישיים:

שני קרמר

עדי פינשטיידט

יעלי חיון

נדב מליכסון

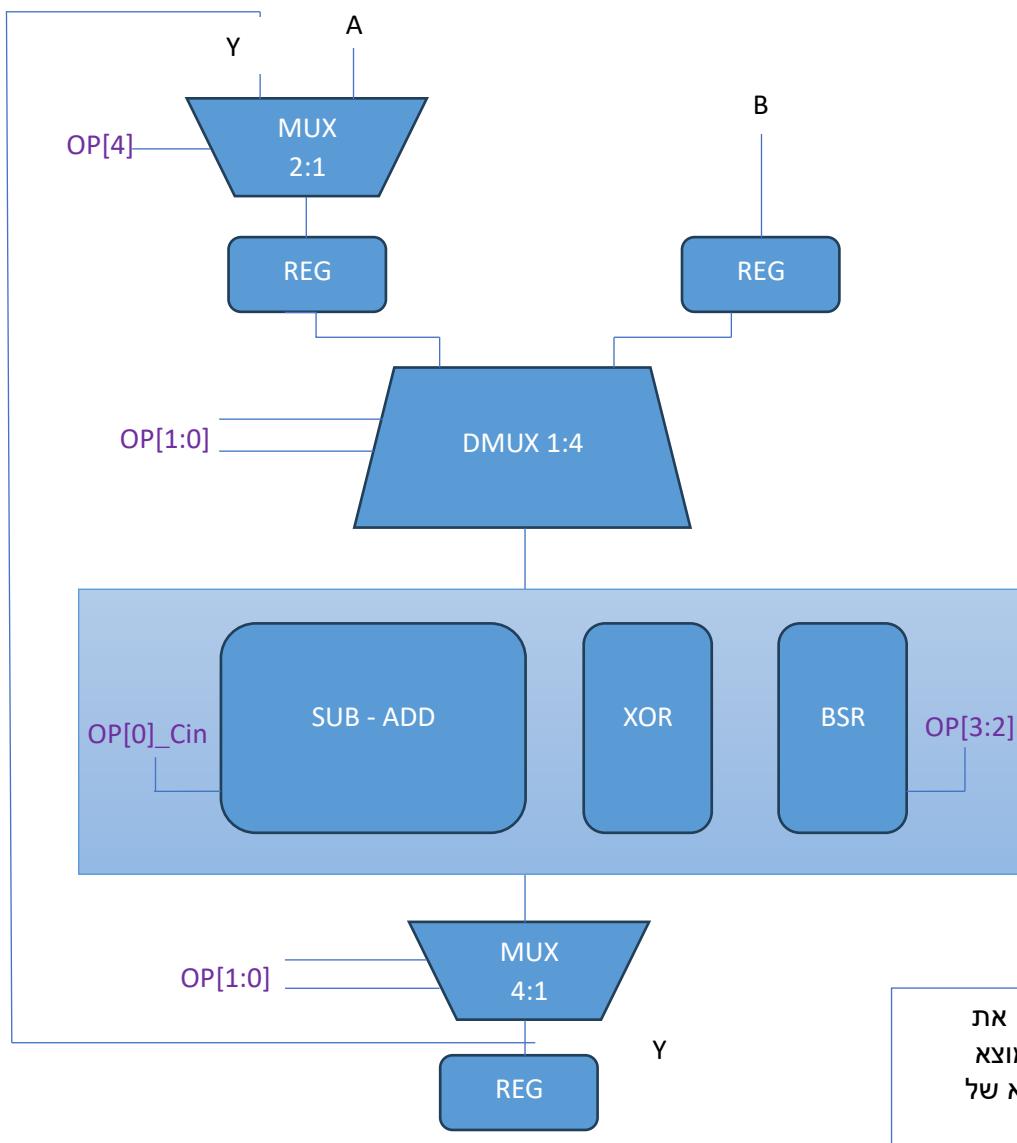
חלק א' - תכנון מוקדים

בפרויקט זה נבנה יחידת ALU של ארבעה ביטים במהלכו נשתמש בטכнологית gpdk045.

בחולק זה נציג את התכנון הראשוני של המעגל אשר ימשיך את יחידת ALU. יחידה זו נשלטת ע"י שדה ה-`opcode` המגדיר את הפעולה אותה צריך לבצע. נגידור את ה-`opcode` באופן הבא:

Block	Opcode	Function
SUB-ADD	Y0000	ADD
	Y0001	SUB
BSR	Y0110	BSR1
	Y1010	BSR2
	Y1110	BSR3
	Y0010	BSR4
XOR	Y0011	XOR

נציין כי כאשר $OP[0]=1$ נכניס קלט חדש לרגיסטר A ועבור $OP[1]=1$ נשתמש בפלט הקודם בעבר וגייסטר A.



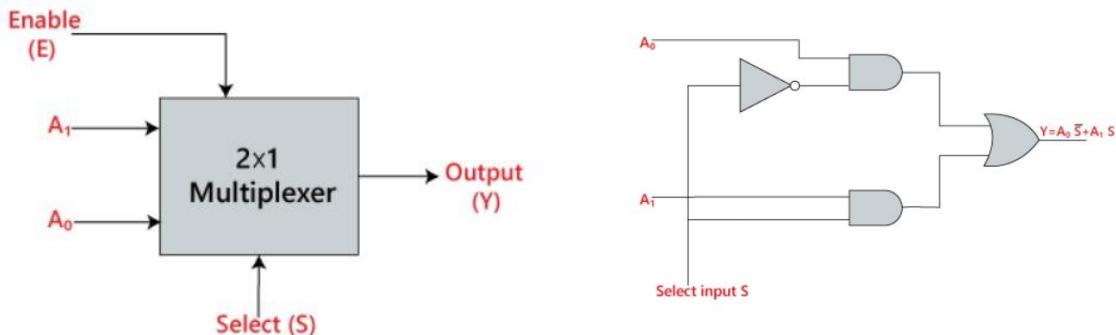
הערה- כאן שינוינו את המשוב להיות המוצא הקודם ולא המוצא של Register

בעת נסביר על כל מרכיב בדיאגרמת הבלוקים:

4 רכבי 2:1 MUX

סיבית ה- selector הינה $[4:0]$ והינה כבוייה כאשר הערך שנכנס לרגיסטר A הוא הקלט החדש, ודולקת כאשר נכנס הפלט מהモץ הקודם.

שבור כל רכיב נשתמש ב-2 שער AND, שער NOT ושער OR, כלומר בסה"כ עבור 4 רכיבים לפחות ל- 8 שער AND, 4 שער NOT ו-4 שער NOT.

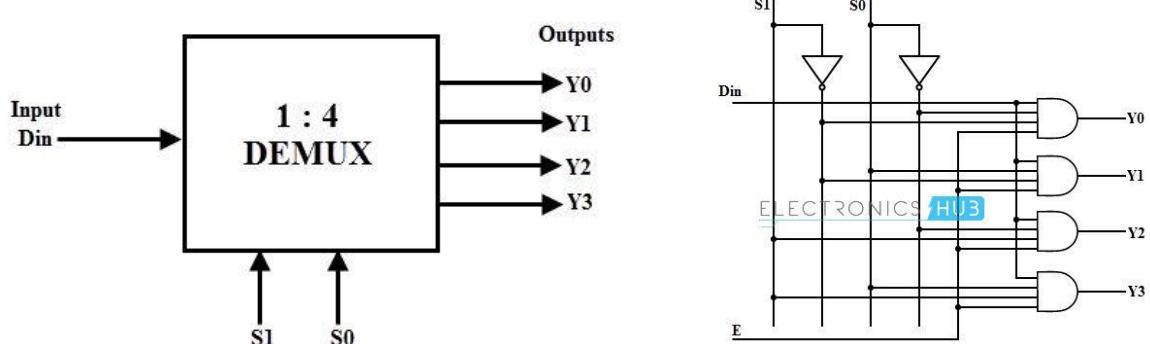


8 רכבי 1:4 DEMUX

4 רכיבים עבור הרגיסטר A ו-4 רכיבים עבור הרגיסטר B הנשלטים ע"י $[1:0]$ opcode אשר בוחר את המיקום אליו המידע יועבר. המוצא הכלול מרכיב מ-8 ביטים בך שלמיקום הייציאה הנבחרת הועבר ביט הבנינה, ובשאר המוצאים יש '0'. מיקום הייציאה הנבחרת מעיד על הבלוק אליו משתייכת הפעולה.

רכיב זה יוזר בהקטנת ההספק הסטטי המתבצע בمعال.

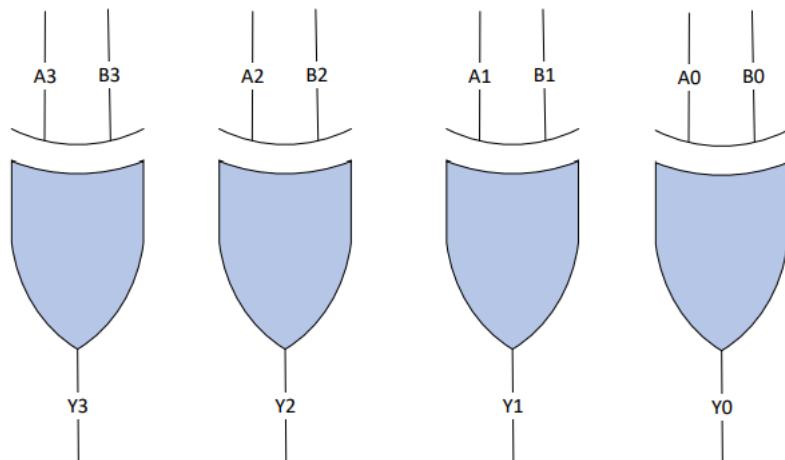
שבור כל רכיב נשתמש ב- 2 שער NOT ו-4 שער AND4:1, כלומר בסה"כ עבור 8 רכיבים לפחות ל- 16 שער NOT ו- 32 שער AND4:1.



Bitwise XOR

מקבל בקלט את תוכן הרגיסטרים A - B ומבצע bitwise XOR של 4 ביטים.

הIMPLEMENTATION יבוצע באמצעות שער 1:2:2:XOR בצורה הבאה:



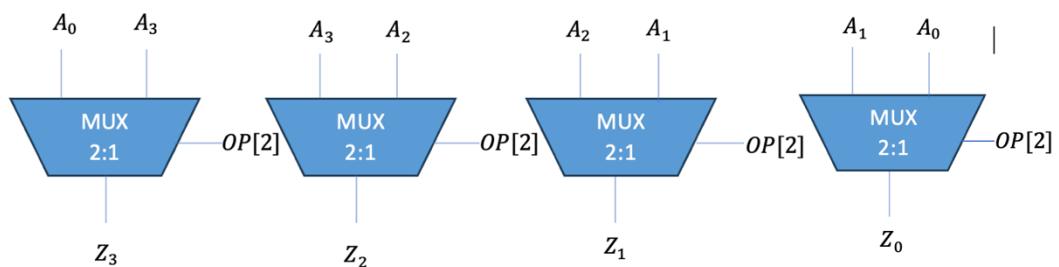
בזה"ב נזדקק ל- 4 שער 1:XOR.

BSR shifter

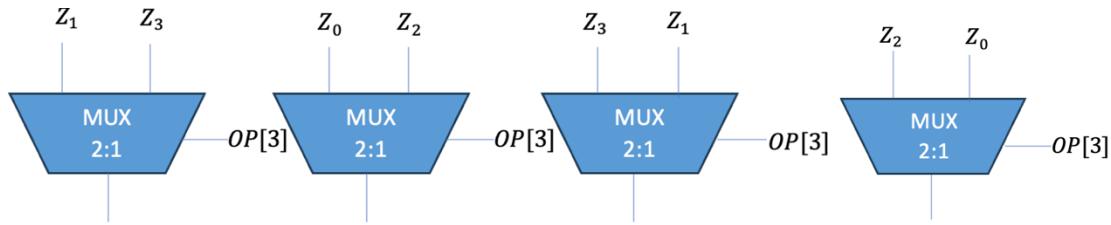
הBSR ימושש על ידי שתי דרגות הזרה כך שמעבר דרך שתי השכבות ייצור לנו 4 סוגי הזרזה הדורשים הנקבעים על פי [OP[3:2]] .

נציין כי BSR-4 היא למעשה מבצעת הזרה כלל שכן אוט הקבינה שלנו מורכב מ-4 ביטים בלבד כך שהזרזה ציקלית ב-4 מקומות שווים ערך ללא ההזרזה כלל.

shift 1

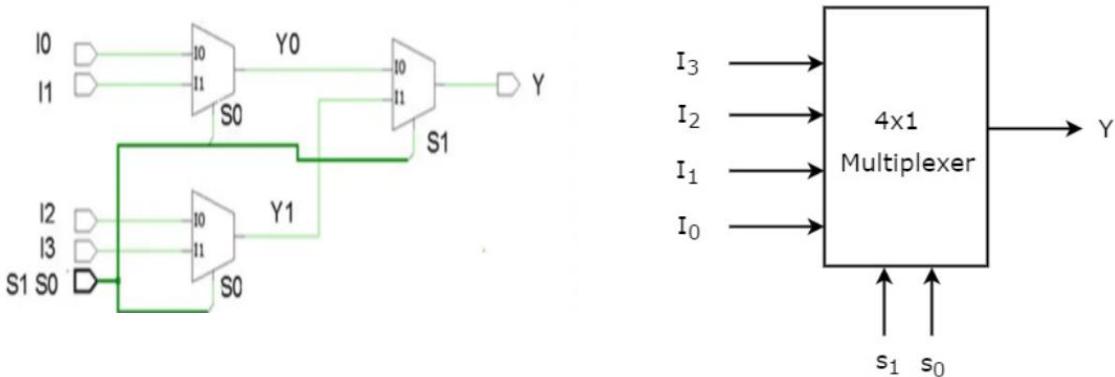


shift 2



4 רכיבי 1 MUX

רכיב זה מקבל בכניסה את מוצאי פעולות ה- ALU ובודר בעדרת $opcode[1:0]$ את המוצא הרצוי שיוכנס לרגיסטר Y . אחת מבניות ה-MUX תהיה D.C- כלומר הכניסות של הסיביות ה-0 וה-1 מקוצרות זו לזו.



רכיב זה מורכב משלושה 2:1 MUX ולבן נמשח אותו ע"י 3 שער NOT, 6 שער AND2:1 ו-3 שער OR2:1.
כלומר בסה"ב נשתמש ב 12 שער NOT, 24 שער AND2:1 ו-12 שער OR2:1.

Add-Sub

נמשח רכיב התומך ב-2 הפעולות המתבצע על רכיב ה- Adder שעלה מימושו נפרט בהמשך. הכניסות ל- Adder הן שני קלטים בעלי 4 ביטים כל אחד ובנוסף Cin.

ערכי הכניסות יקבעו לפי המימוש המצויר המתבצע על העקרונות הבאים:

- עבור פעולה SUB (כאשר $opcode[0] = 0$ שווה ל-1) ו- $C_{in} = 1$ עבור ADD.
- עבור פעולה SUB הקלט הנוסף יהיה \bar{B} , ועבור פעולה ADD יהיה B .

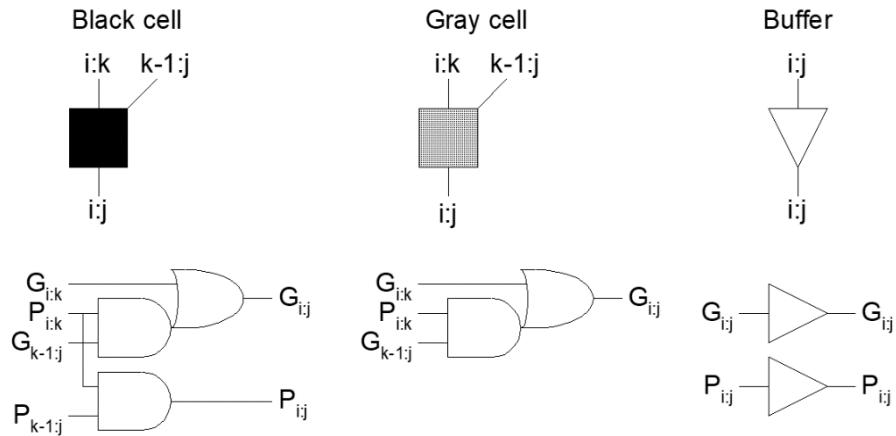
בעת נסbir על מימוש רכיב ה- Adder:

Adder-Knowles [2,1,1,1]

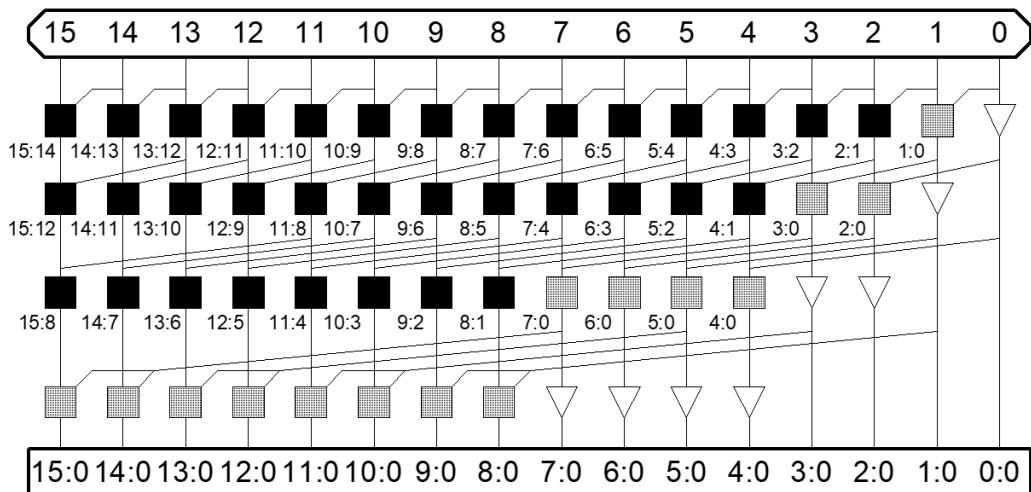
מימוש ה- Adder מחולק ל-3 שלבים:

1. Bitwise PG-Logic: בשלב זה ניצור עבור כל זוג ביטים מהקלטים A ו-B את הערכיהם ההתחלתיים המתאים עבור P ו- G. באשר נאותל את P_0 להיות 0 ואת G_0 להיות C_{in} שייקבע בהתאם לפעולה אותה נרצה לבצע: 1 עבור SUB ו-0 עבור ADD.
וחושבו בעדרת half-adders לפי המימוש שהוצג בהרצאה. $P_{1:4}$, $G_{1:4}$

.2 Group PG-Logic: בשלב זה נחלק את הסיגנלים G , P שהתקבלו מהשלב הקודם לקבוצות. בכל קבוצה מחושב ה- G וה- P המשויך אותה קבוצה בעזרת הגדרה של black cells ו- gray cells לפי המימוש הבא:

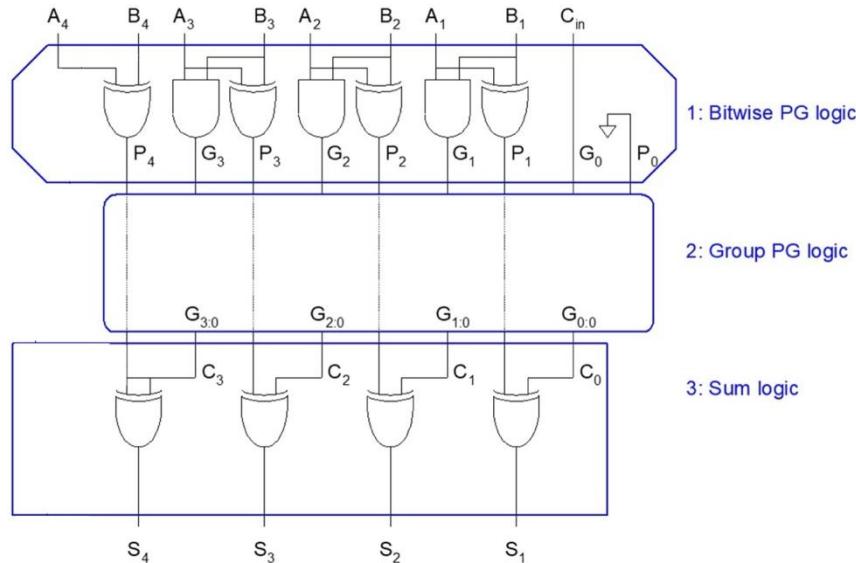


כאשר ה- G net מוממש בהתאם למימוש המתאים לו Knowles Adder בהתאם לשרטוט הבא:



.3 Sum Logic: מבצע עבור כל זוג ביטים XOR בין סיגנל ה- G שהתקבל בשלב השני לבין סיגナル P -הו שהתקבל בשלב הראשון המתאים לו. מוצא ה- Adder 4 ביטים

להלן סכמתה המסתמכת את השלבים:

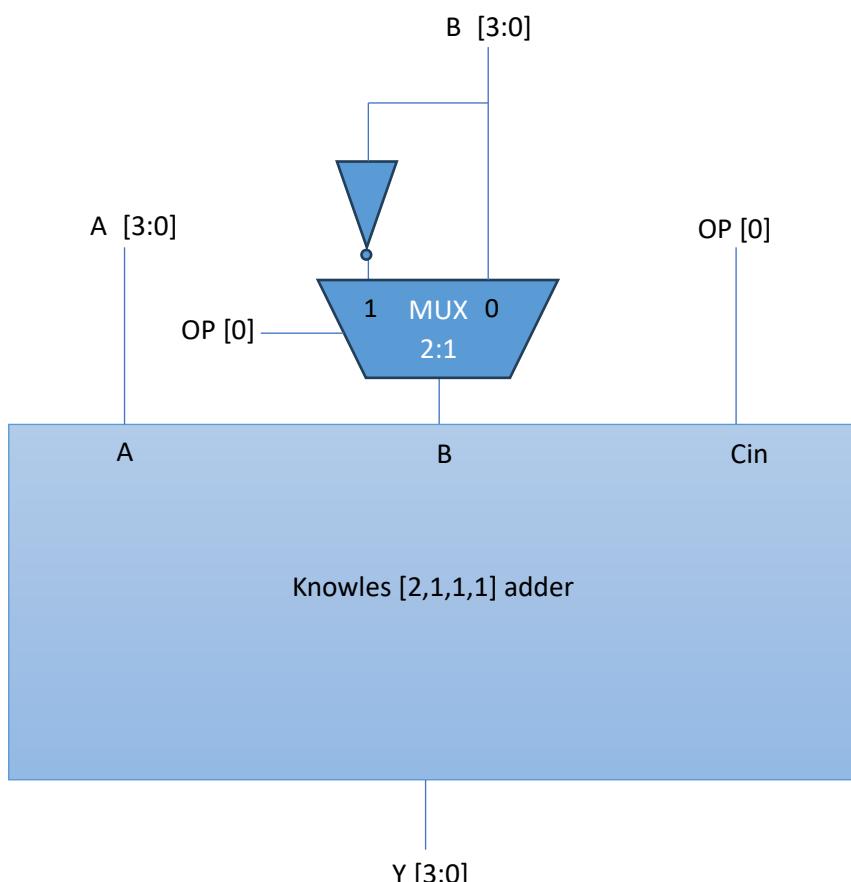


עבור רכיב ה- Adder נשתמש ב- 10 שערי AND2:2, 5 שערי OR2:1 ו- 8 שערי XOR2:1.

על מנת שהרכיב יוכל לתמוך גם בפעולות SUB נשתמש 4 שערי NOT ו- 4 רכיבי MUX2:1.

נציין כי אנו לא תומכים בסיגנל של Carry-out שבן מוצא המערכת הוא 4 ביט.

להלן סכמתה המתארת את הרכיב Add-Sub:



רכיב	רוחב [μm]	אורך [μm]	שטח [μm^2]
OR2:1	0.8	1.84	1.47
AND2:2	1	1.84	1.84
AND4:1	1.4	1.84	2.57
XOR2:1	1.6	1.84	2.94
MUX 2:1	1.4	1.84	2.57
MUX 4:1	4.4	1.84	8.09
NOT	0.4	1.84	0.73
D-FF	4.2	1.84	7.73

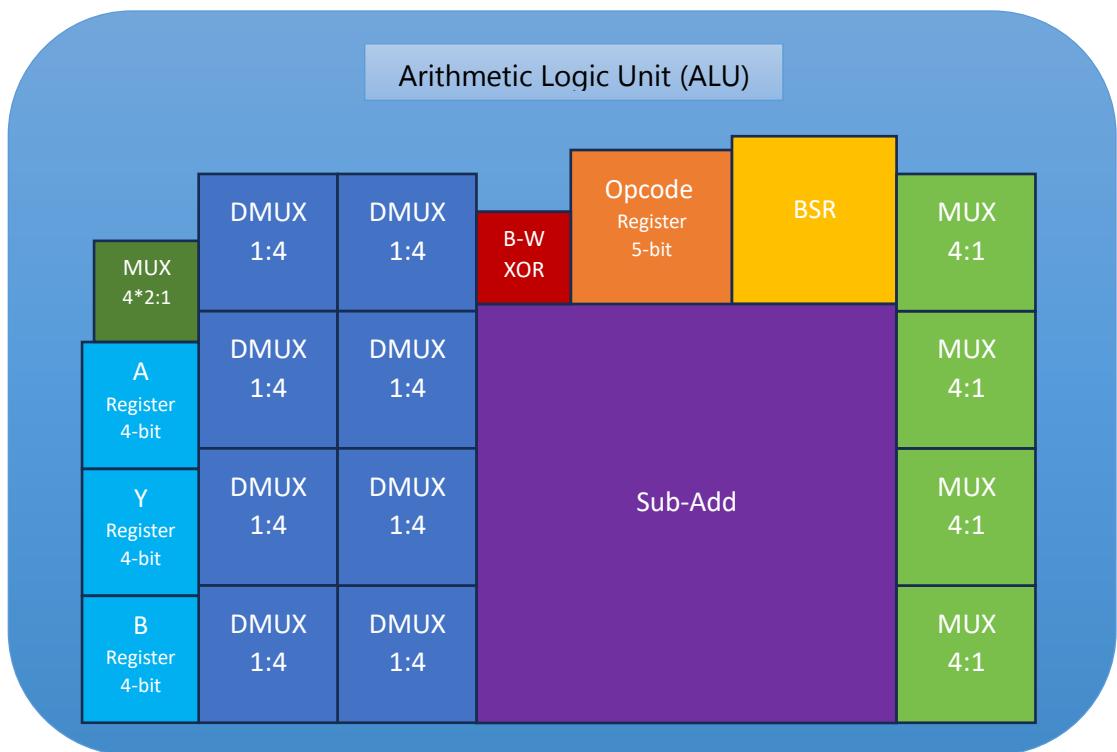
נסכם את שטחי הרכיבים הבסיסיים בטבלה:

רכיב	שטח [μm^2]
DEMUX 1:4	11.74
Bitwise XOR	11.76
BSR	20.56
Add-Sub	62.47
4 Bit Register	30.92
5 Bit Register	38.65

סה"כ השטח הכללי של יחידת ה- ALU $362.64 \mu m^2$.

Floor plan

להלן תרשים ה- Floor plan של יחידת ה- ALU:

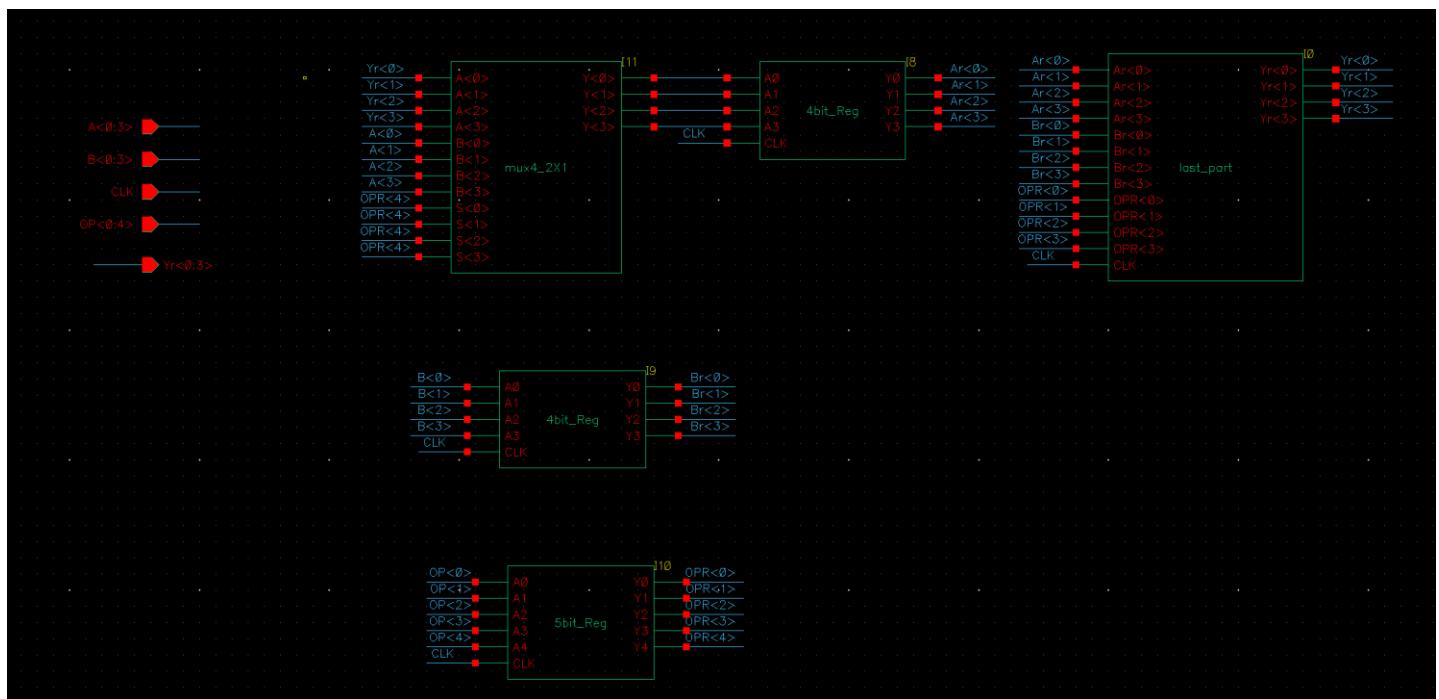


זו היא היררכיה הראשונית על סמך דיאגרמת הבלוקים ואינה מתחשב בחיבורים בין הבלוקים השונים ועל כן היררכיה השטחית תגדל במהלך המימוש.

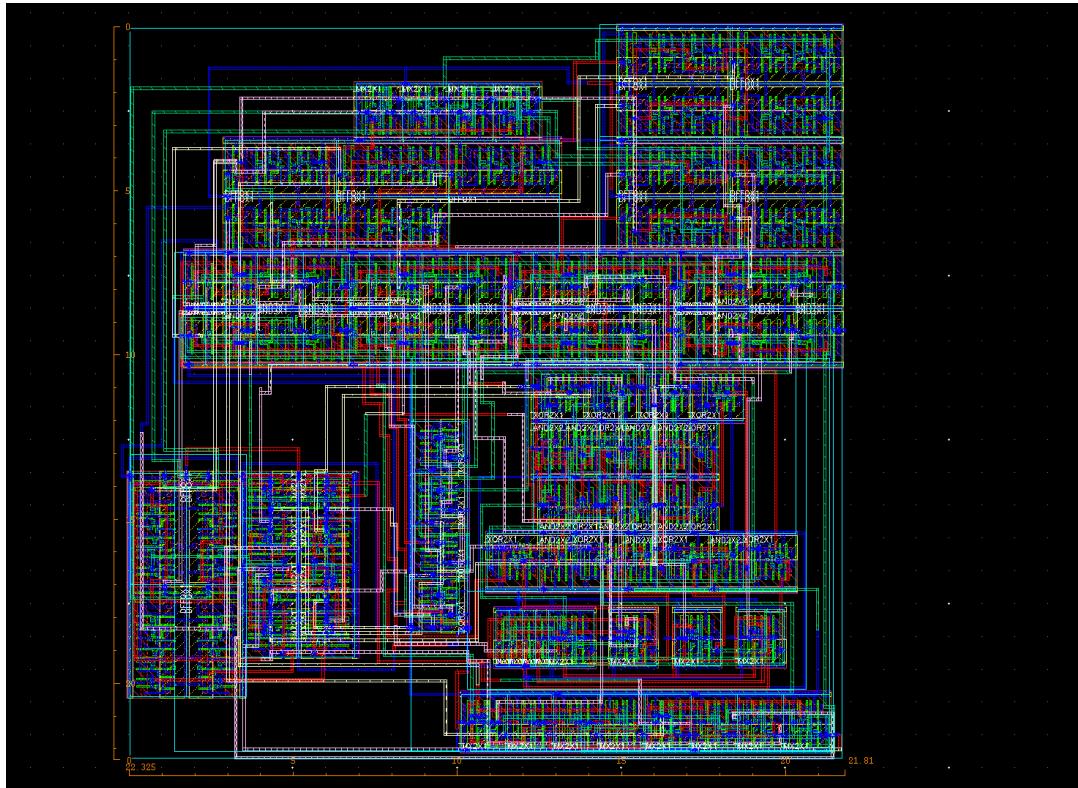
חלק ב' - מימוש ה ALU

עבור כל הרכיבים בפרויקט מומשו סכמה ו-Layout בהתאם לפירוט התכנון בחלק א'. צילומי מסך של אלו ושל תקינות הרצות DRC ו-LVS מצורפים להיל. את הסימולציות ניתן לראות בסוף. נציג כי במהלך שלב המימוש בוצעו מעט שינויים בתכנון הלוגי של רכיבי הפרויקט תוך שמרית הלוגיקה הקיימת. שינויים אלו נובעים מצרכי ייעול השטח של ה-ALU וכן העלות.

להלן הסכמה הכללת של המערכת -

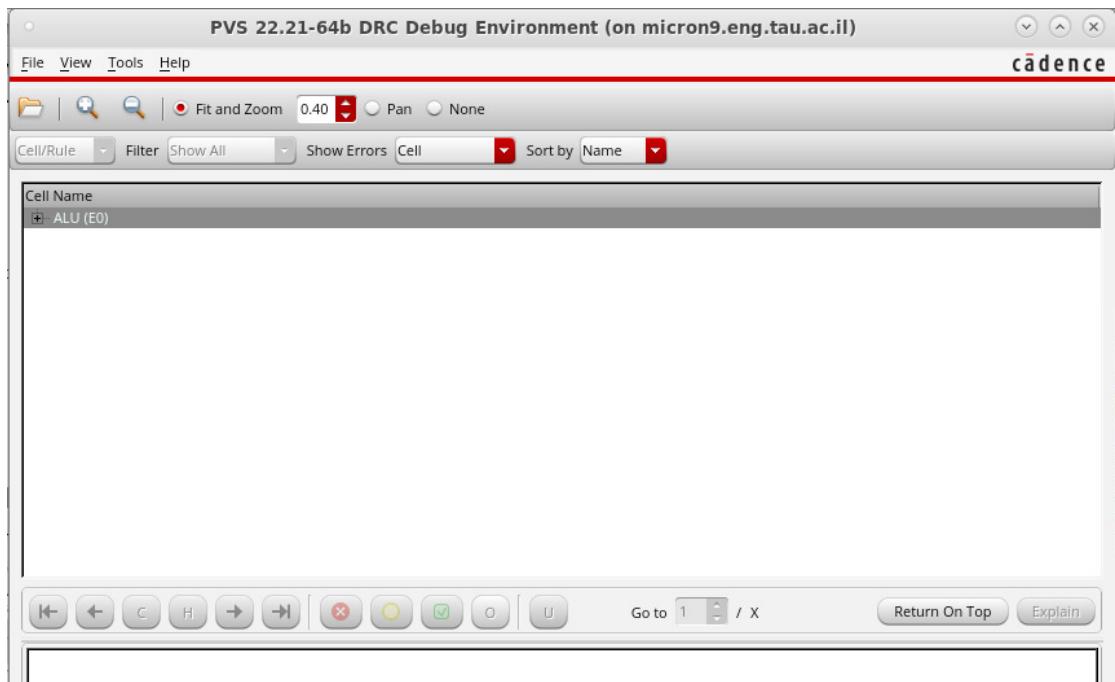


להלן ה-Layout של המערכת –



בשלב התכנון ראיינו כי השטח הכולל של יחידת ה- ALU הוא $362.64 \mu\text{m}^2$. כתע, ניתן לראות כי הגודל הכולל של ה-ALU הוא $486.91 \mu\text{m}^2$. מסרגל המדידה המצויר ל-Layout אורך - $22.325 \mu\text{m}$, רוחב $21.81 \mu\text{m}$. אכן המימוש עומד בדרישה.

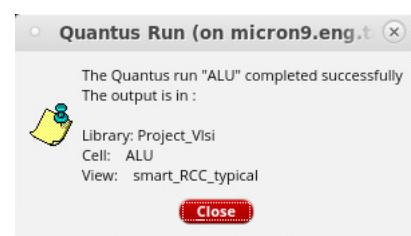
תקינות DRC של המערכת הכוללת –



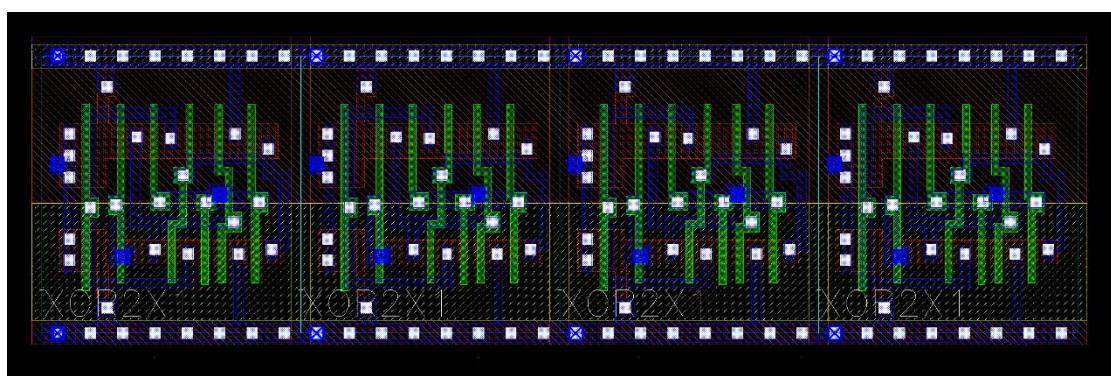
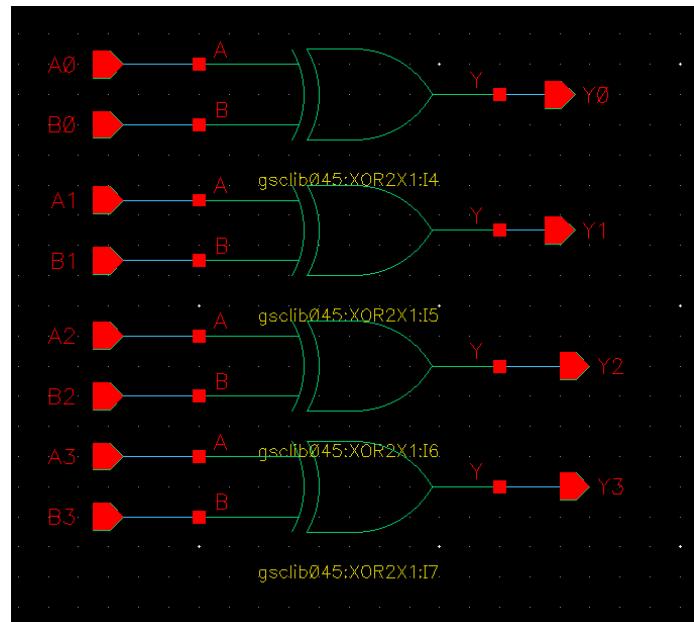
תקינות LVS של המערכת הכללת –

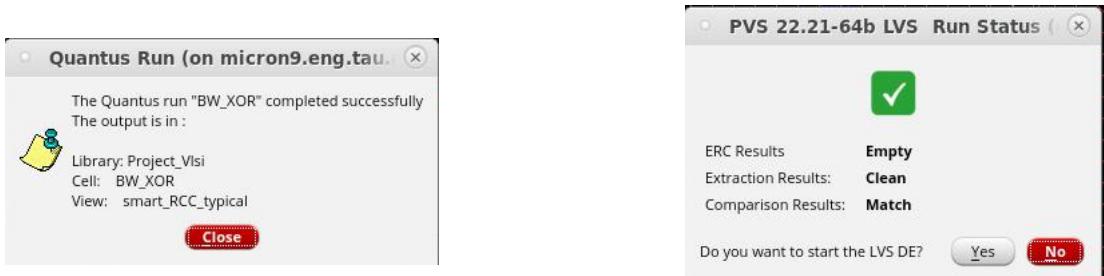


תקינות QRS של המערכת הכללת –

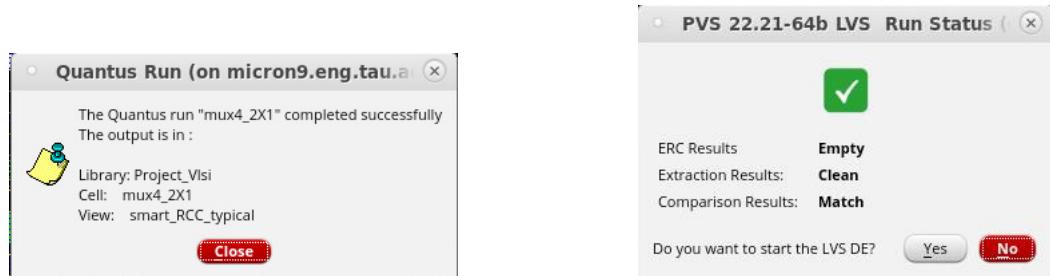
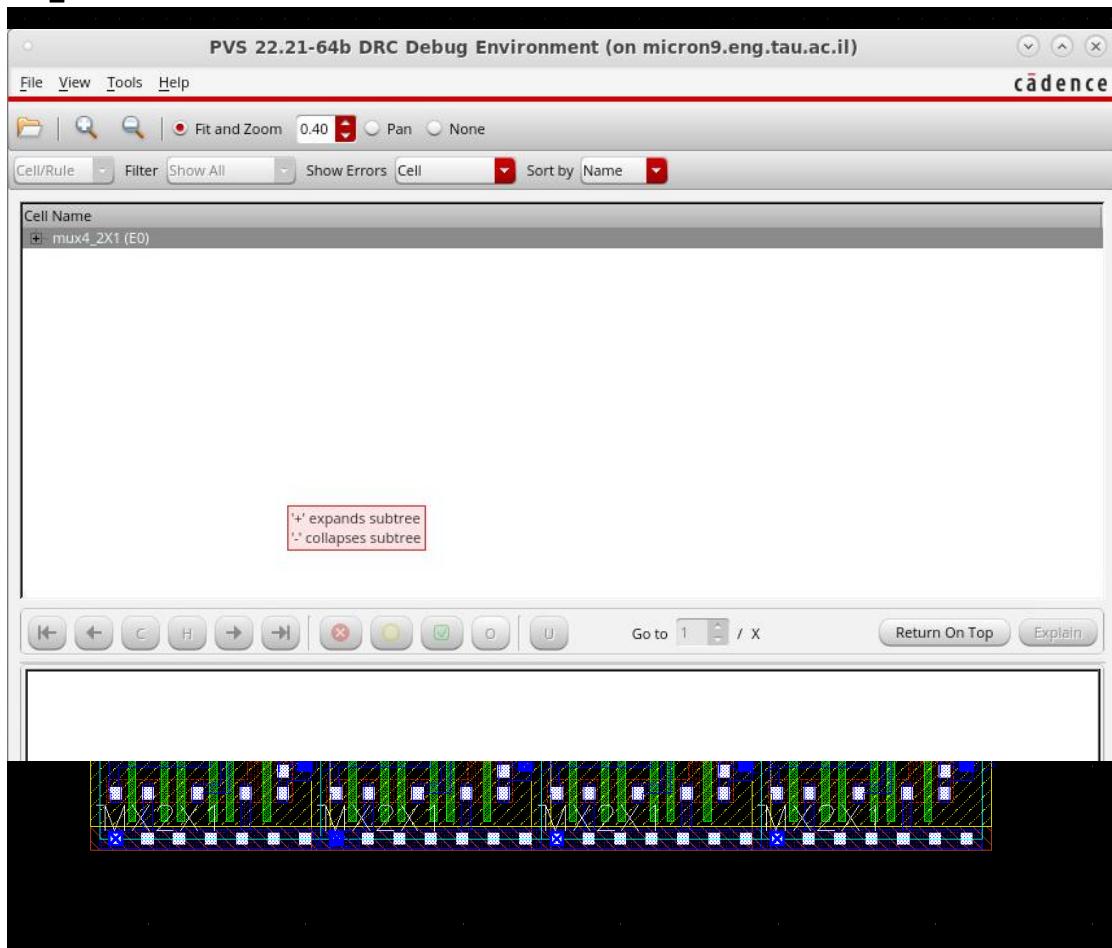


להלן מוצגים כל רכיבי המערכת, מוצגים: סכמה, מוצגים: סכמה, הוכחת תקינות DRC, LVS, QRS, Layout,

BW_XOR:

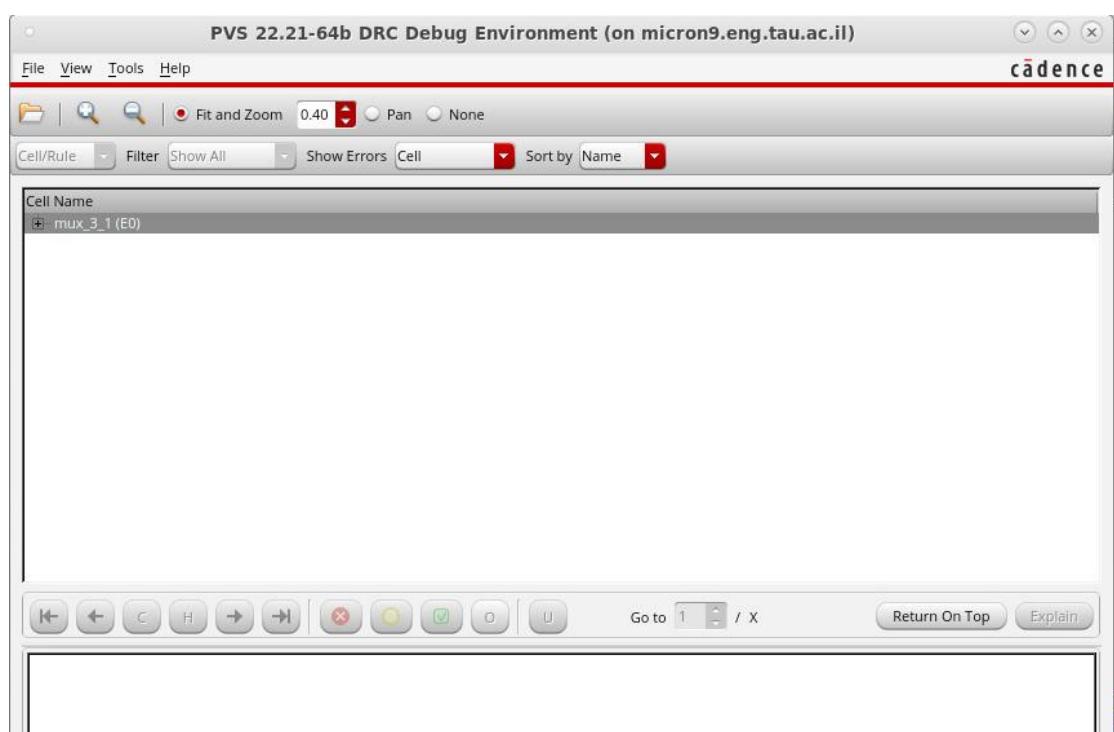
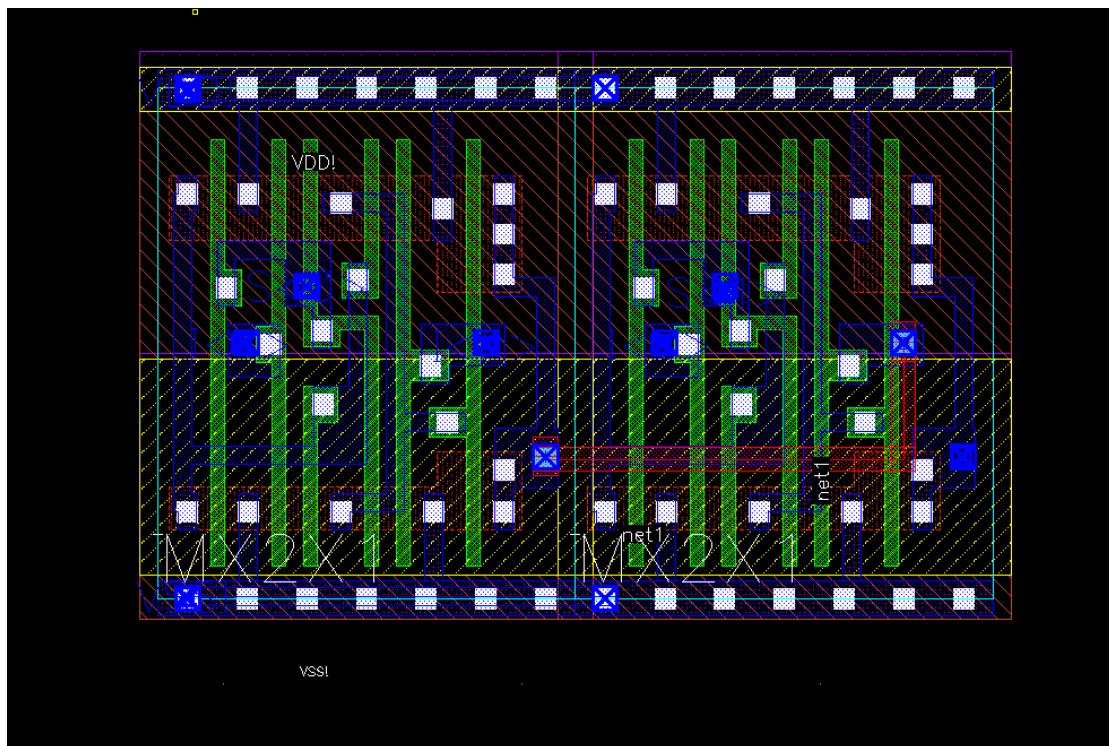


BW_MUX2x1:



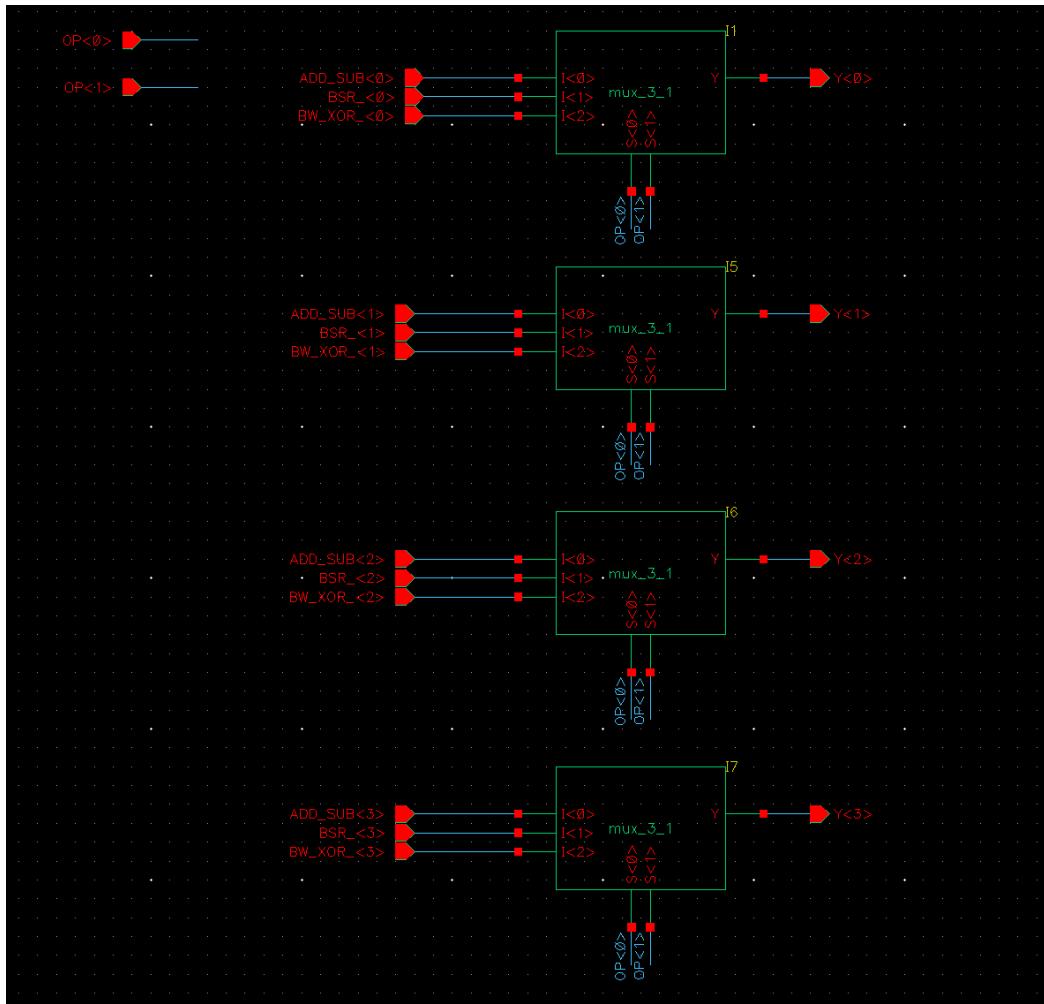
MUX3:1:

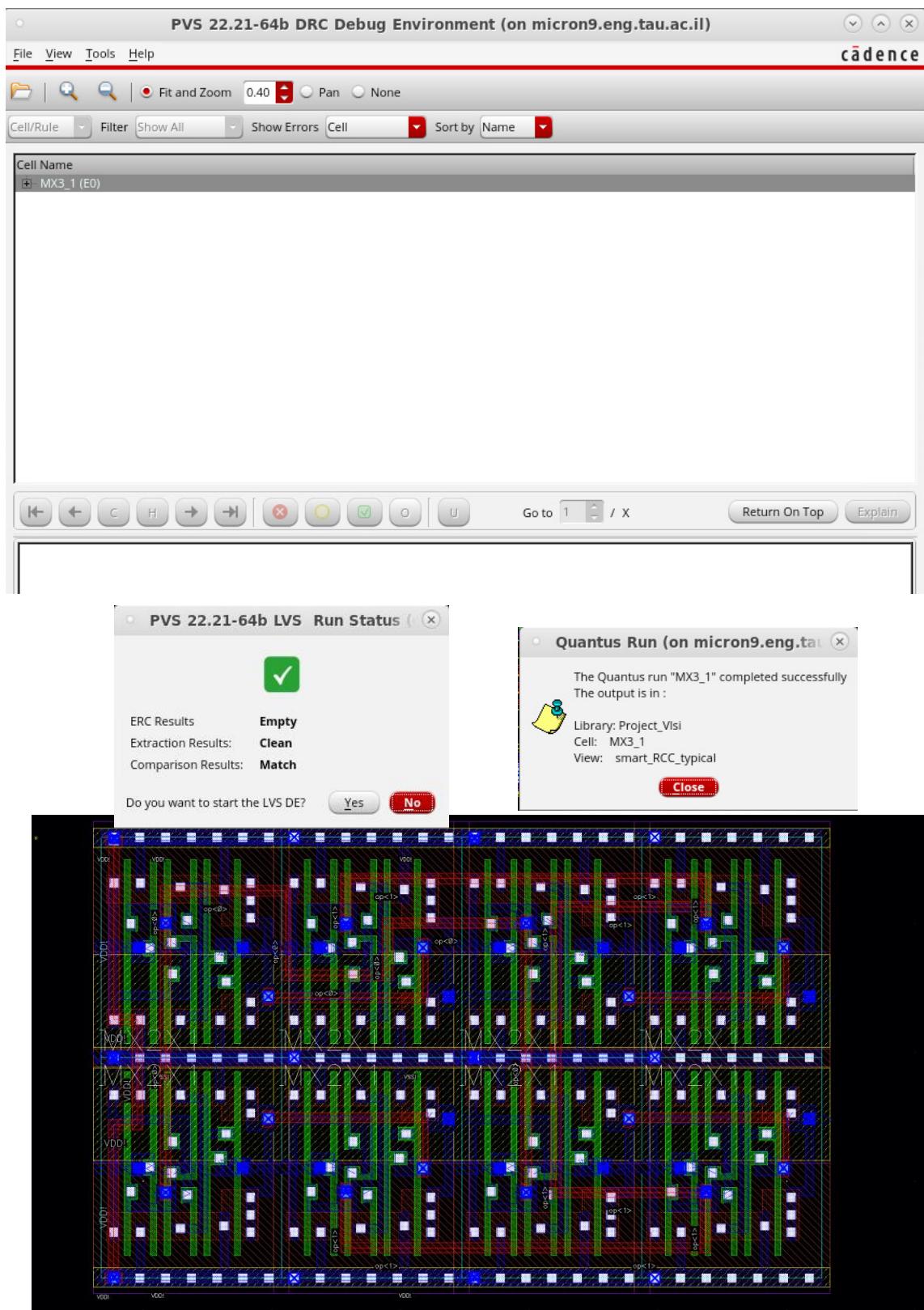
נשים לב כי רכיב זה שונה מהרכיב בשלב התכנון – חשבנו על דרך ליעול השטח מתוך מחשבה על קר שקיימים 3 בלוקים בלבד וכיון להקטין ולייעול את הצורך בMUX4:1. لكن בשלב זה השתמשנו במימוש החדש.



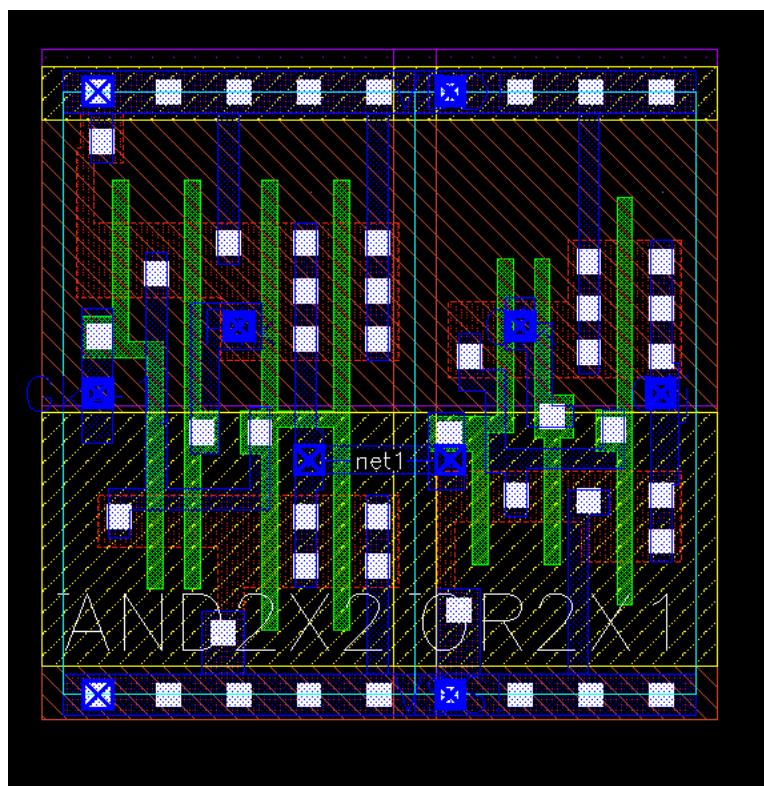
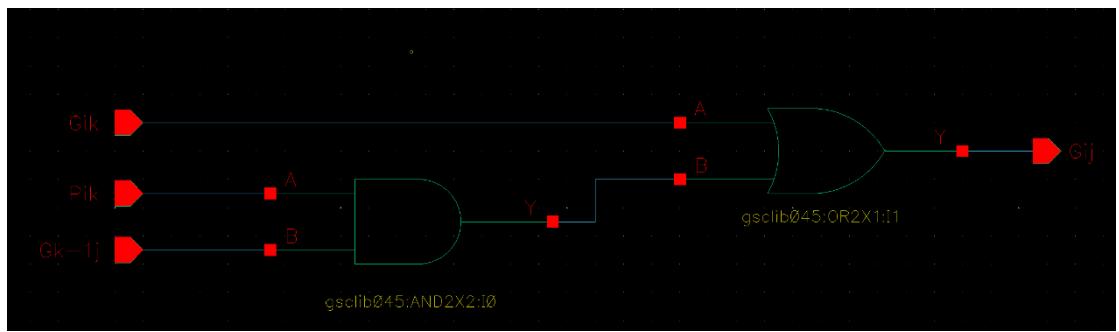


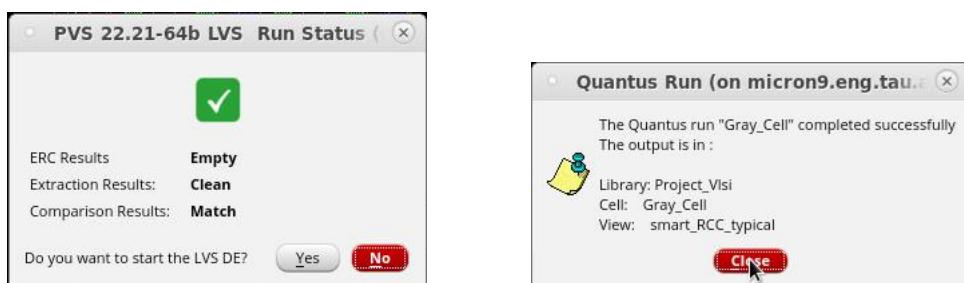
4bit_BW_3:1:



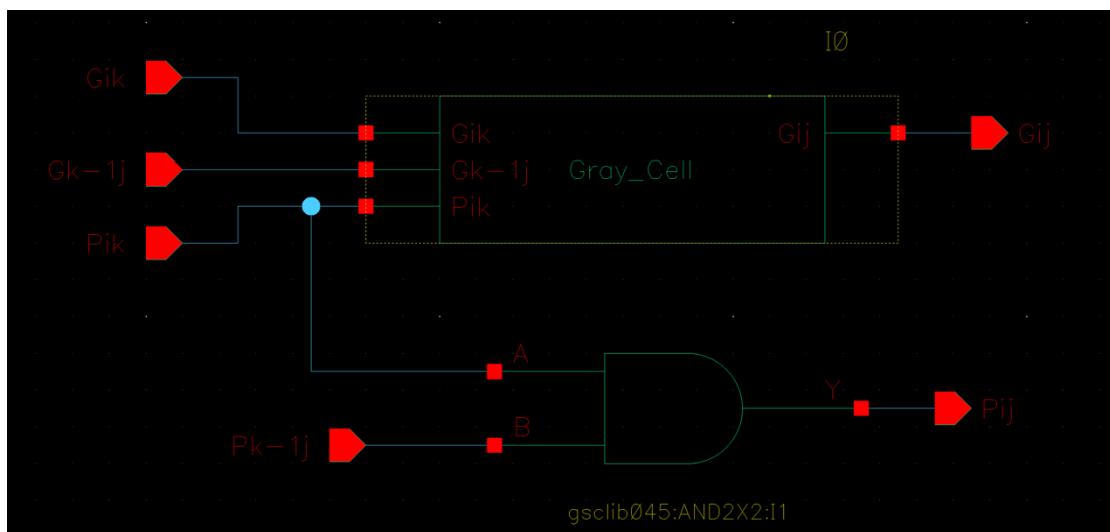


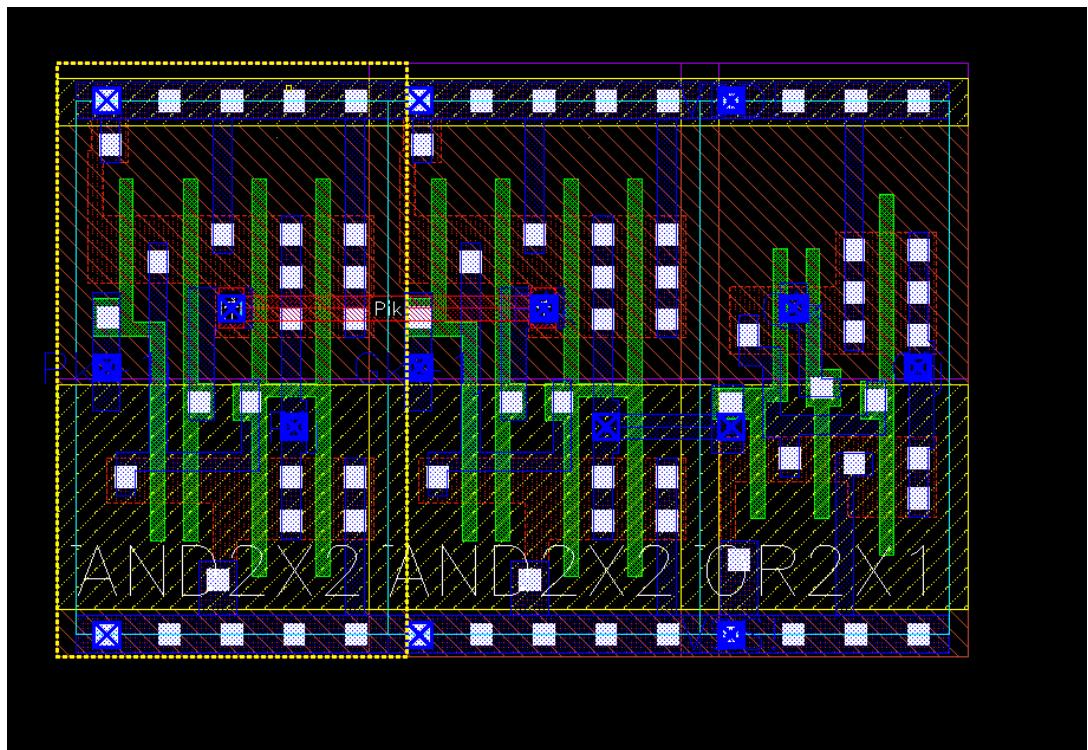
Gray_Cell:





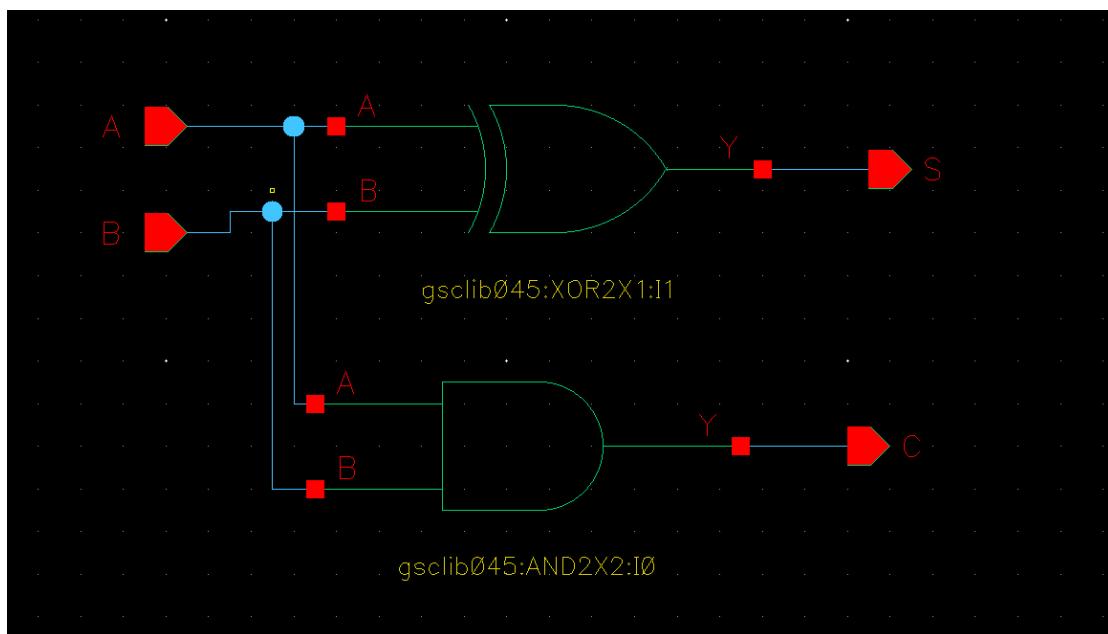
Black_Cell:

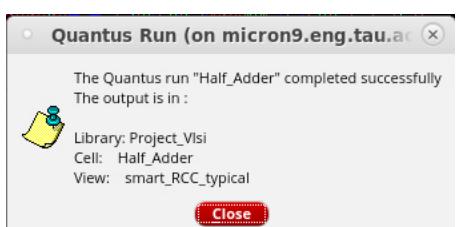
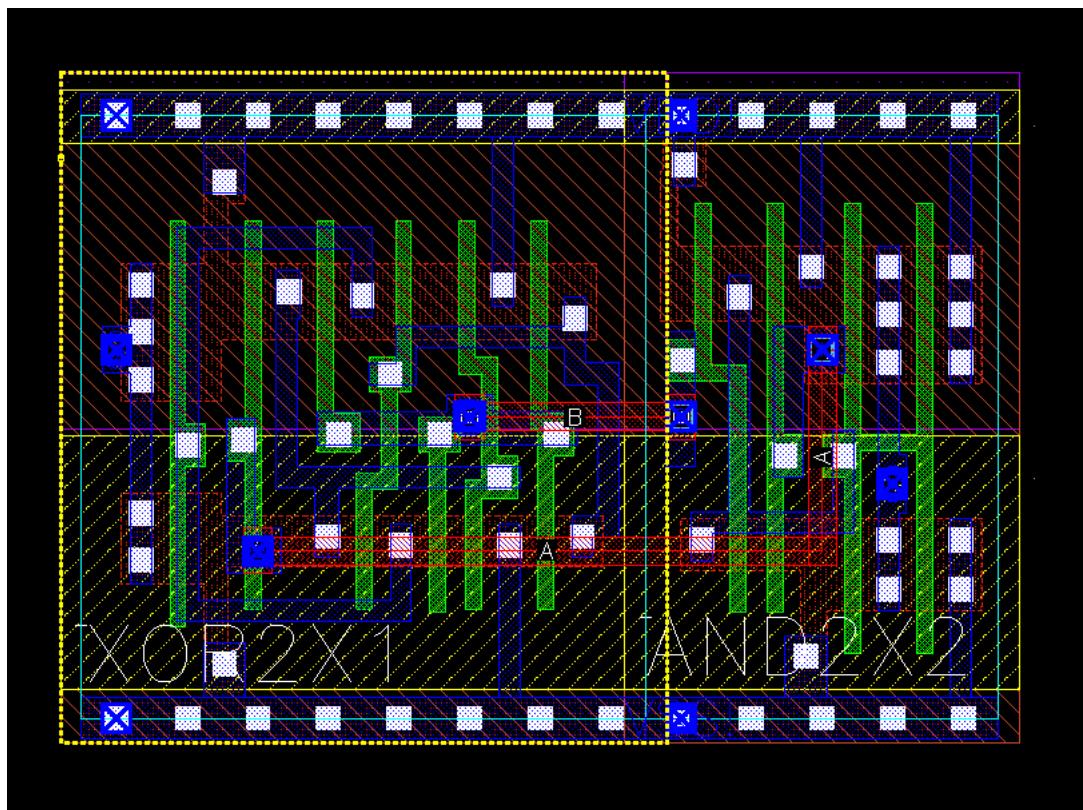




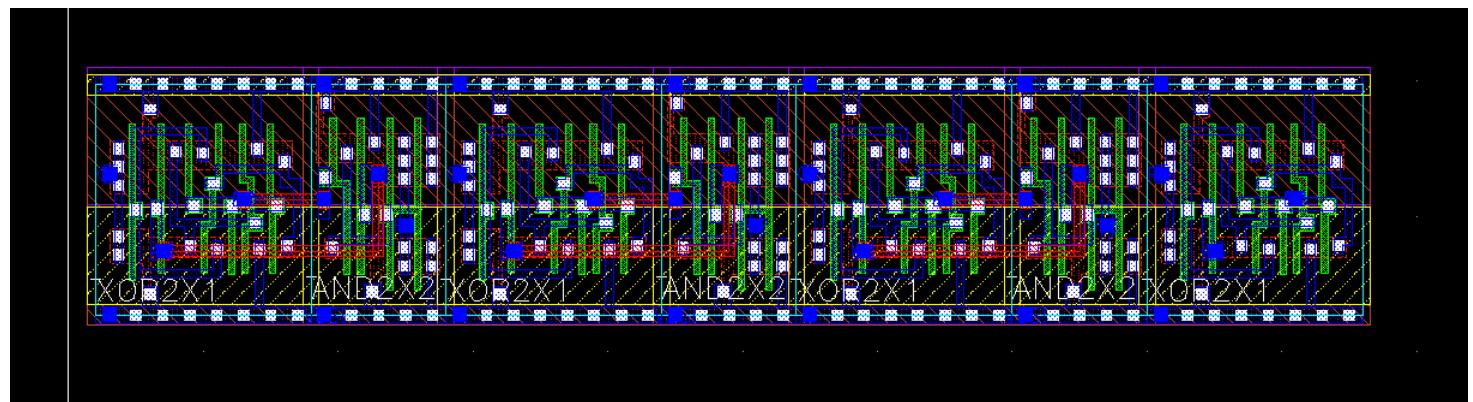
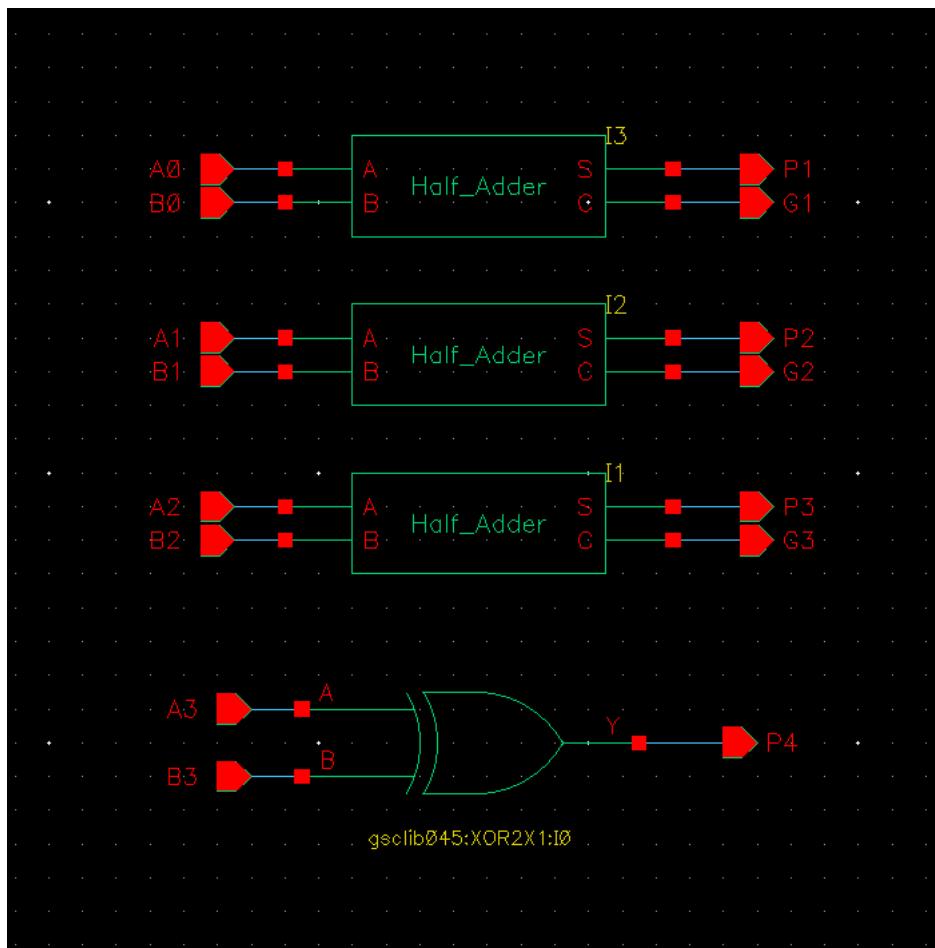


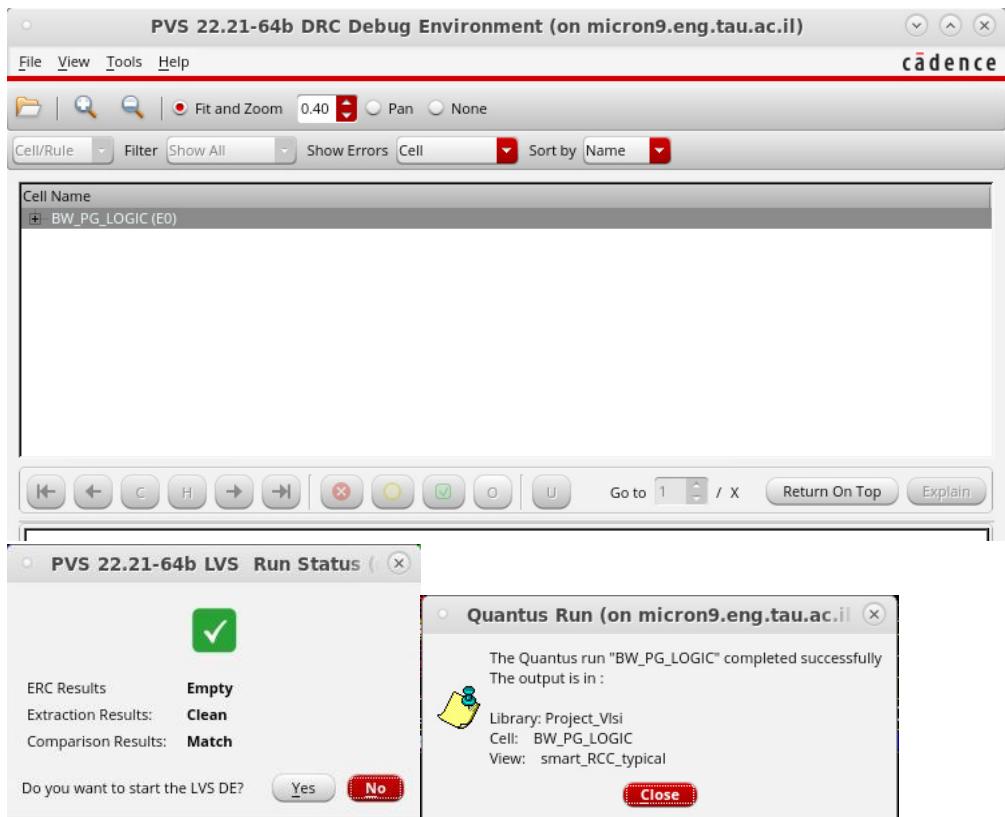
Half-Adder:



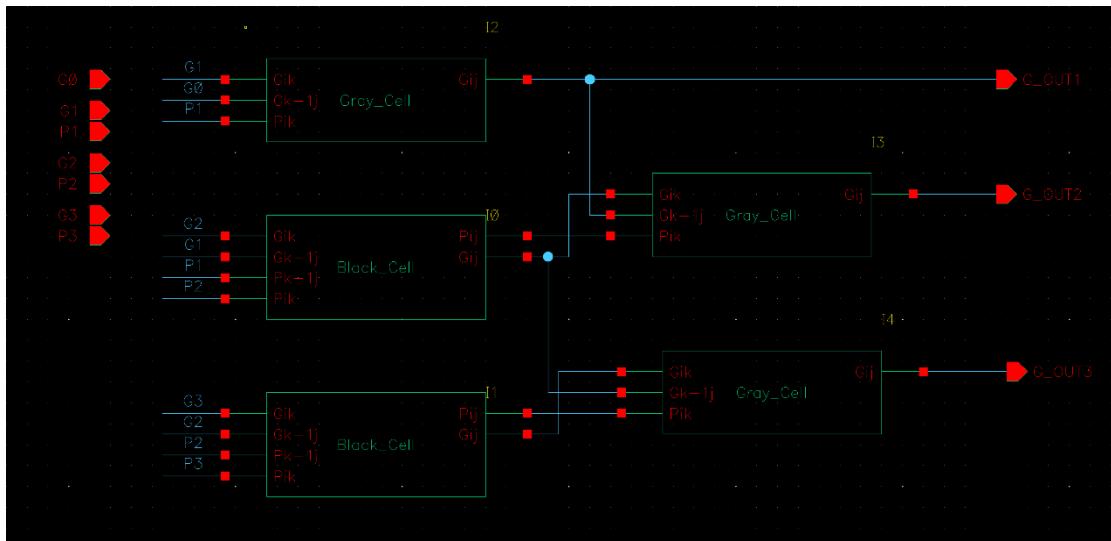


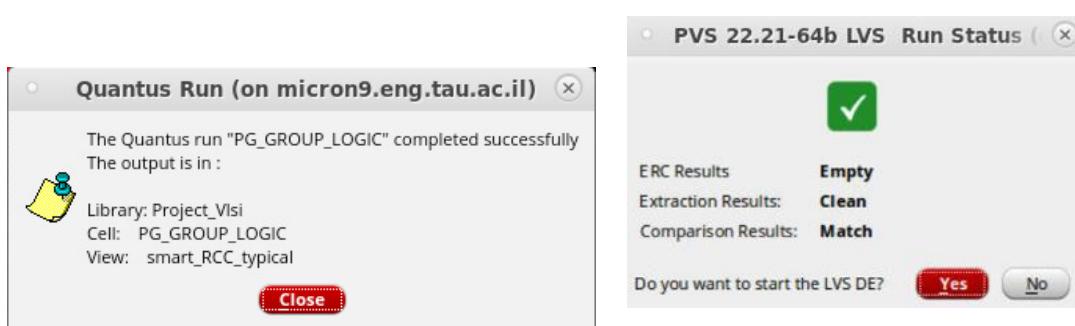
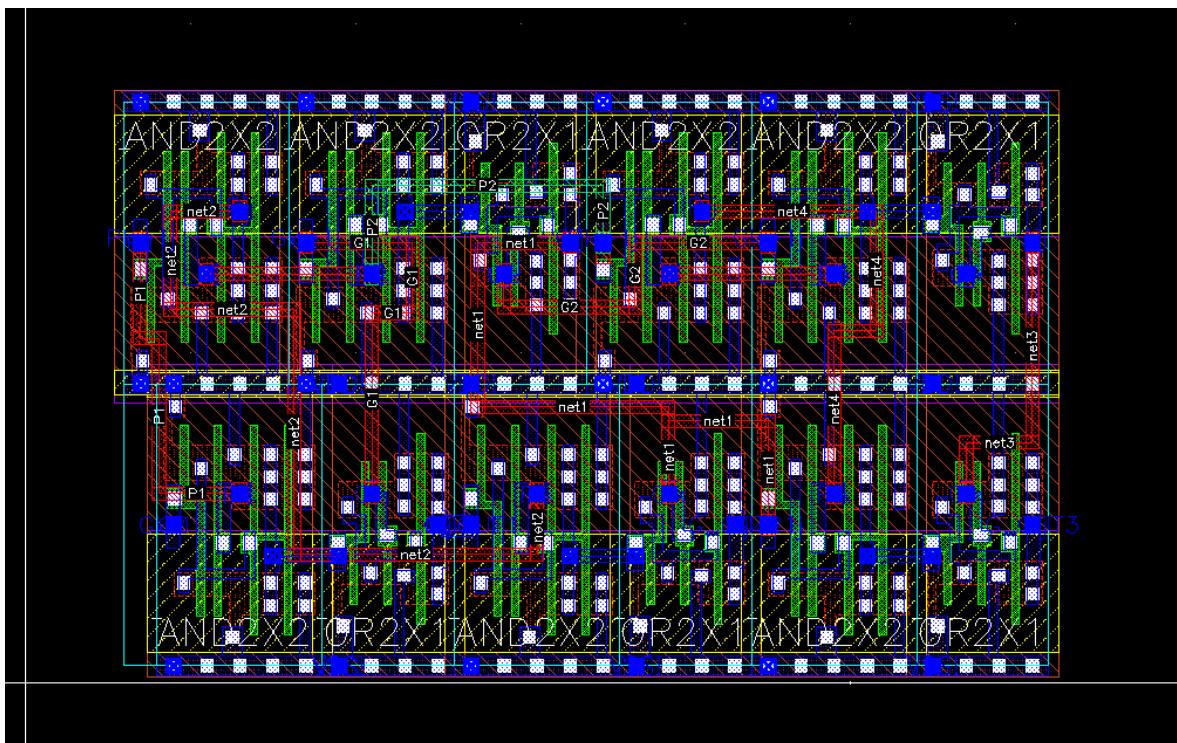
BW_PG_LOGIC:



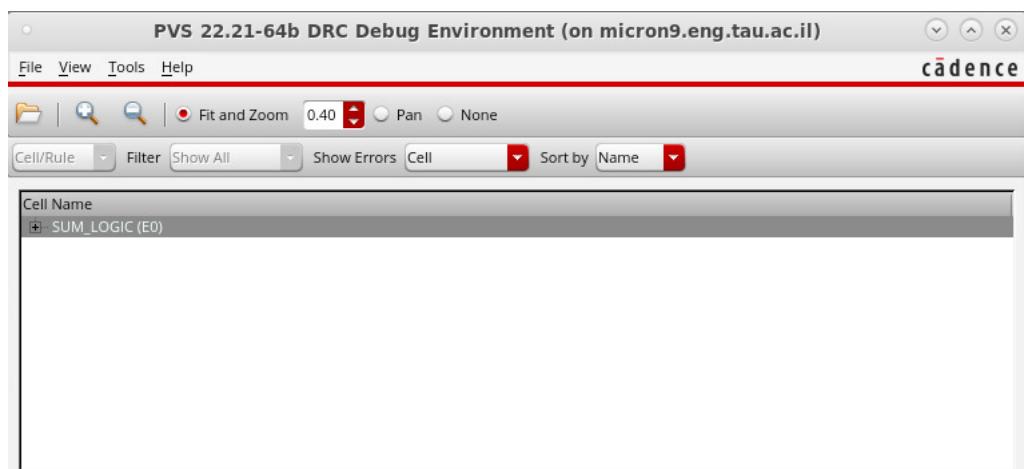
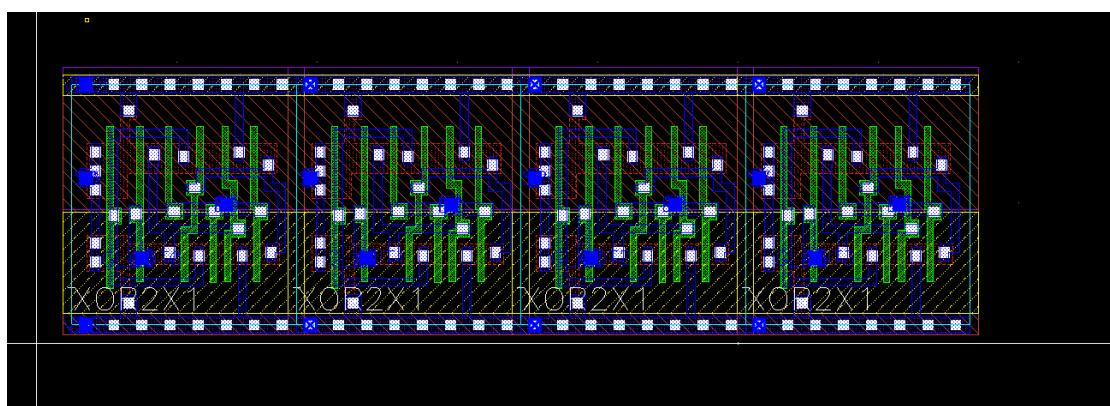
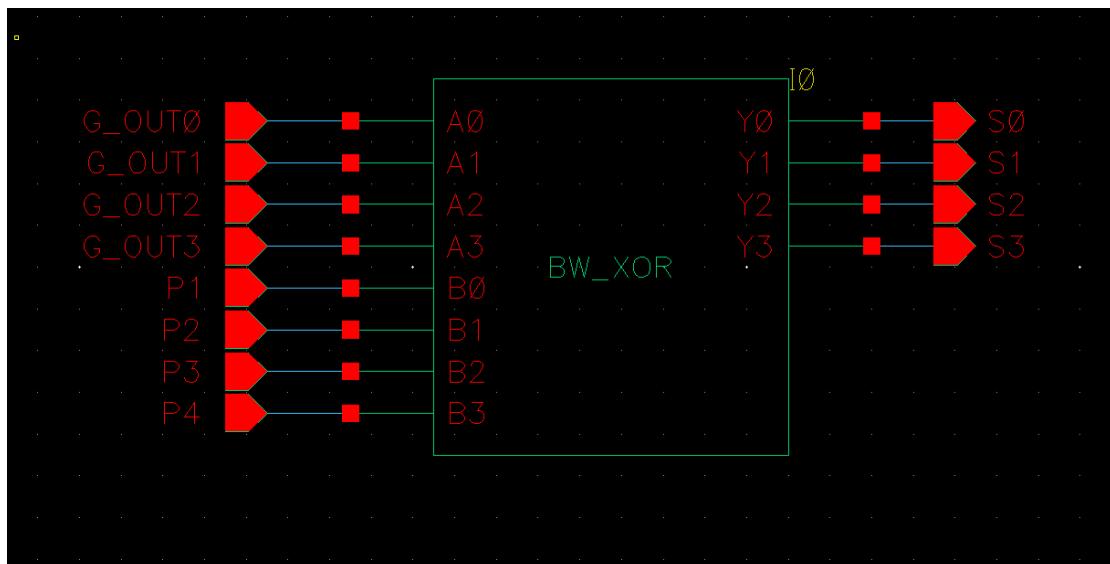


PG_GROUP_LOGIC:



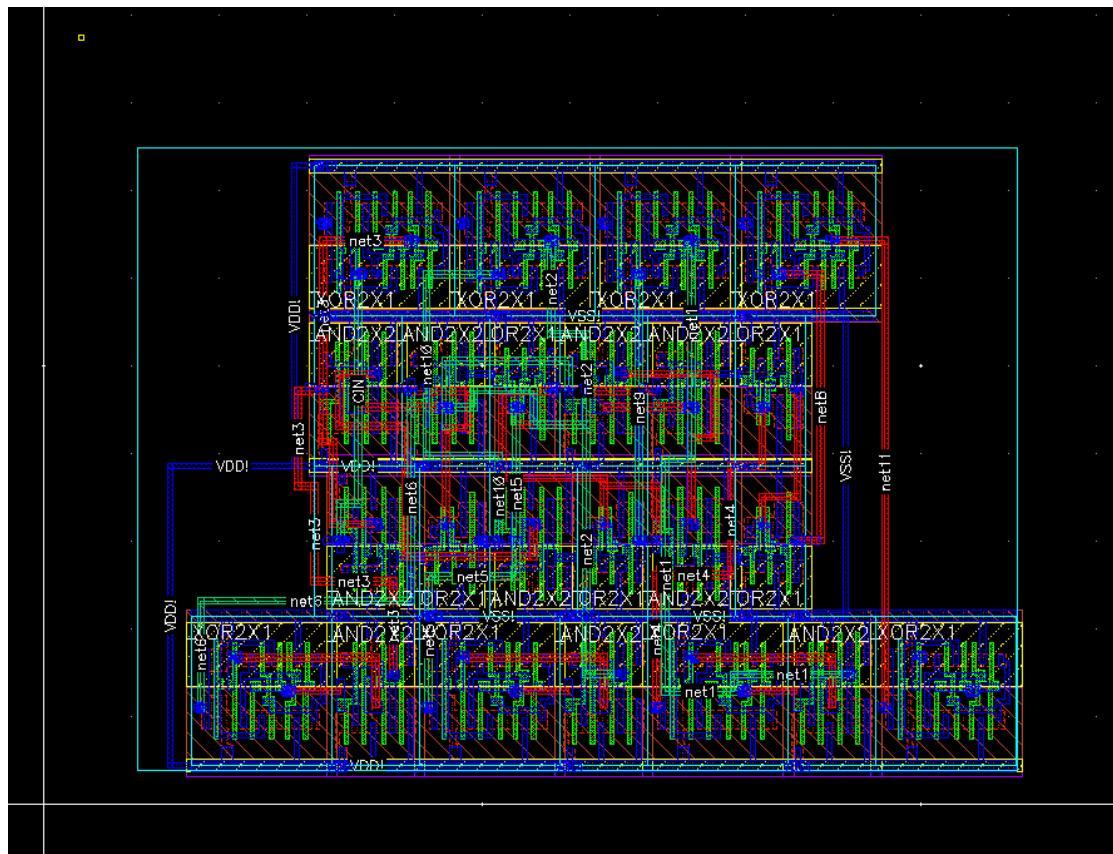
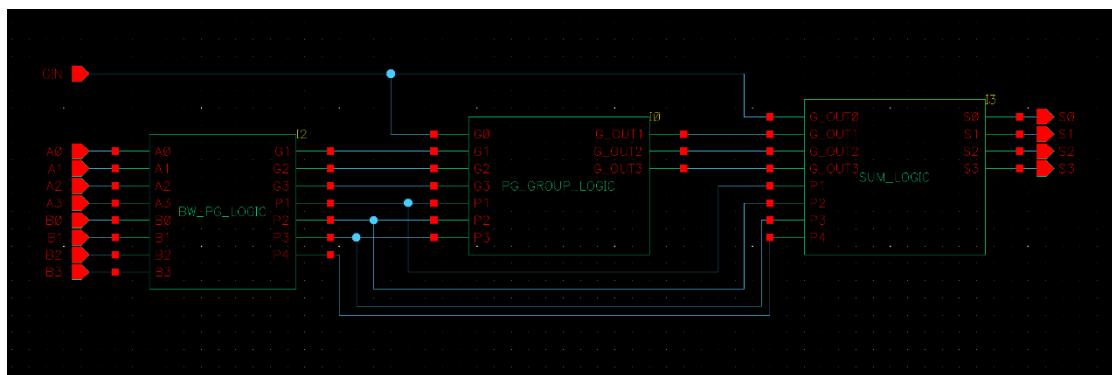


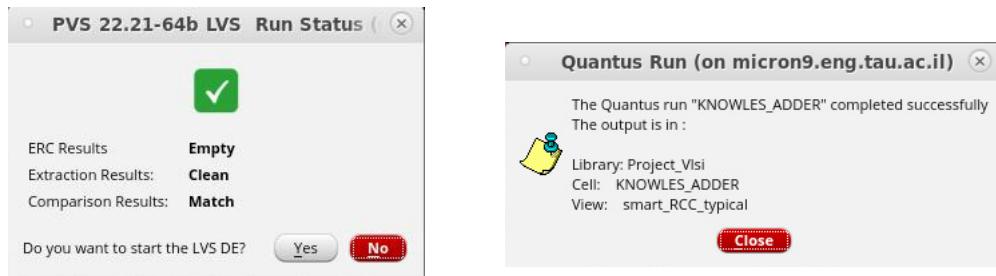
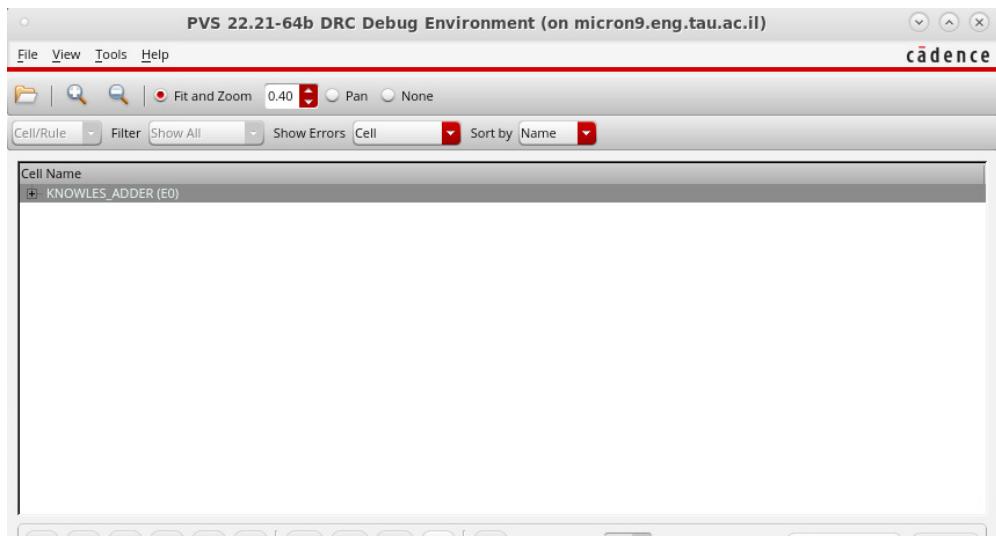
SUM_LOGIC:



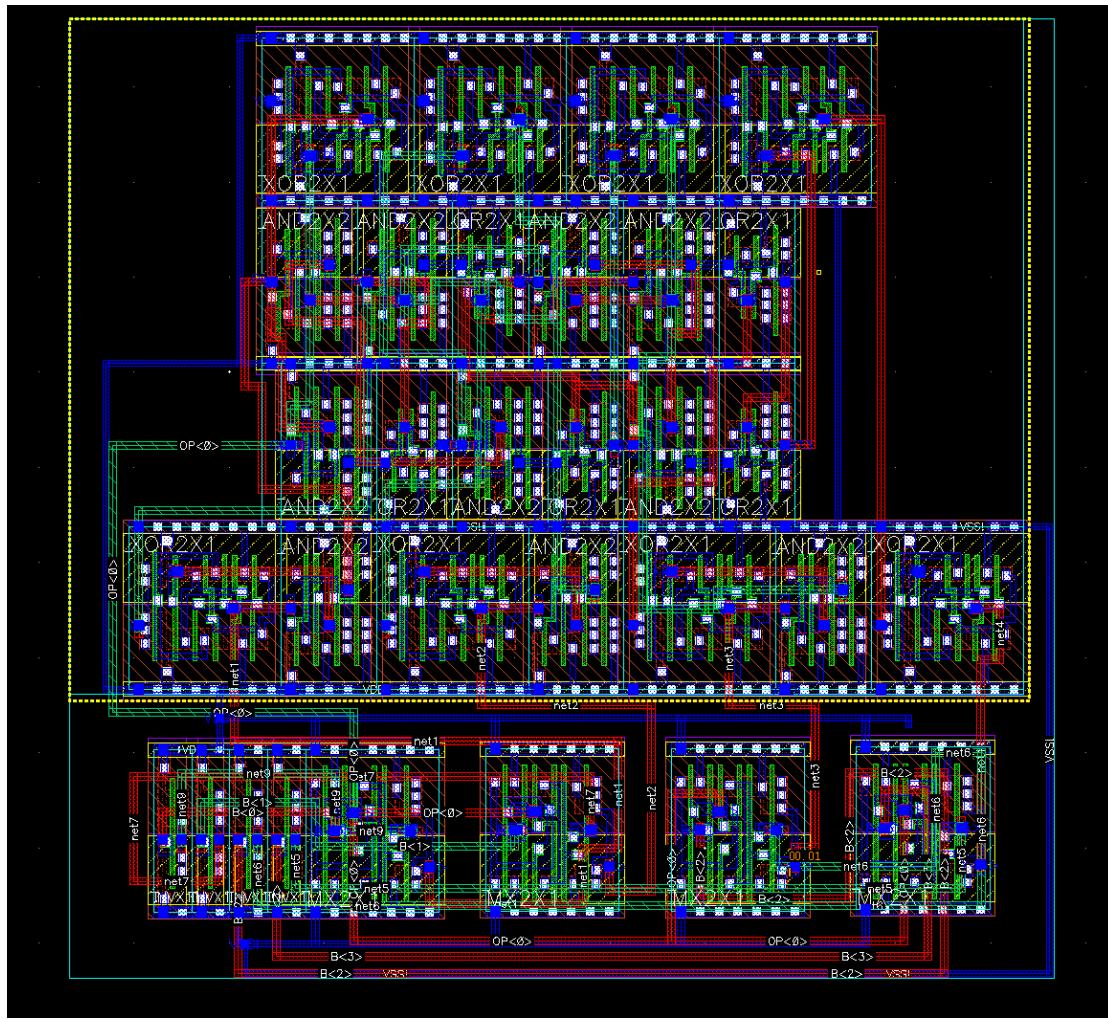
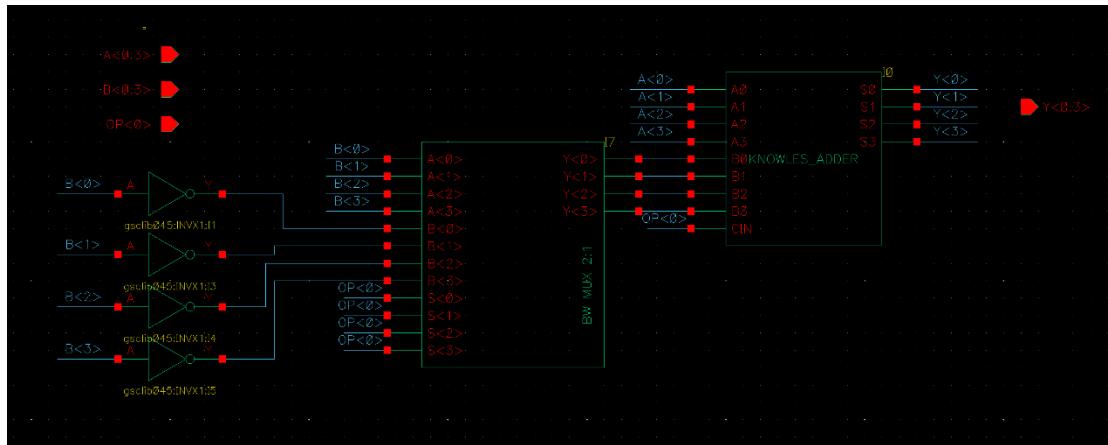


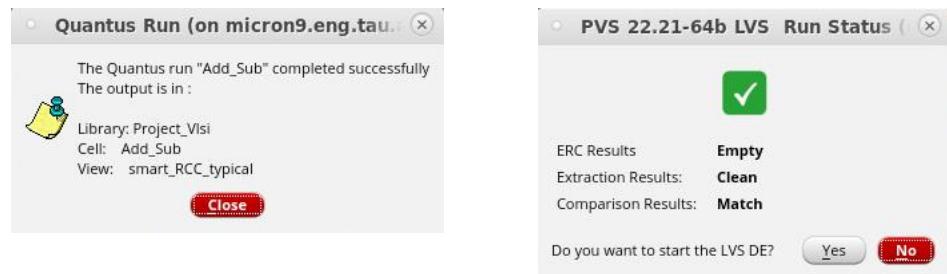
KNOWLES_ADDER:



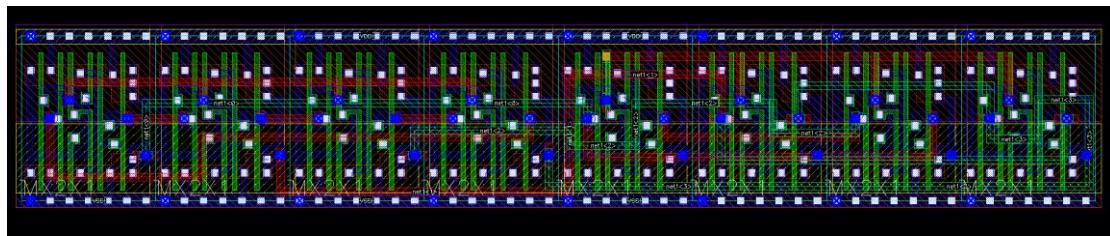
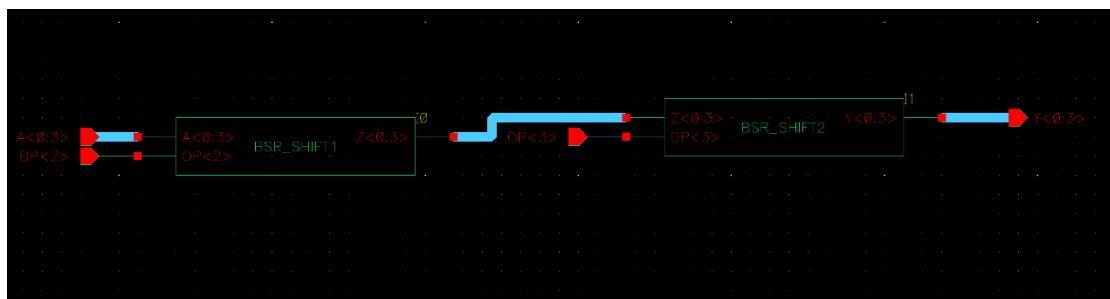


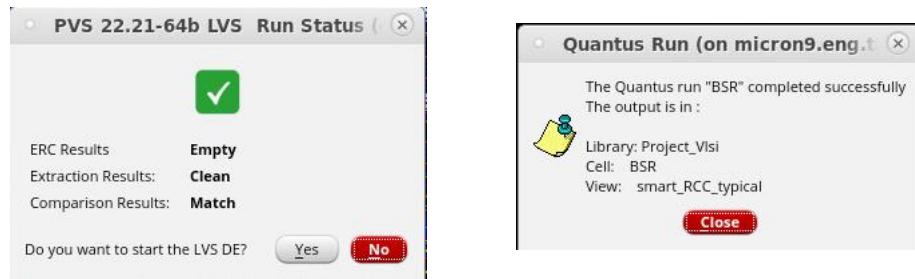
ADD_SUB:



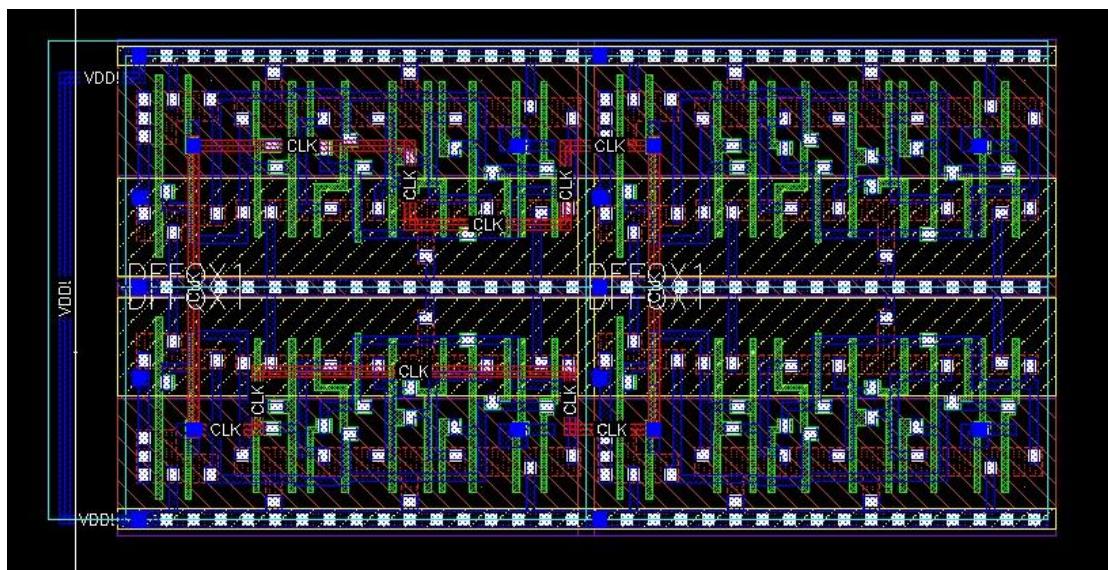
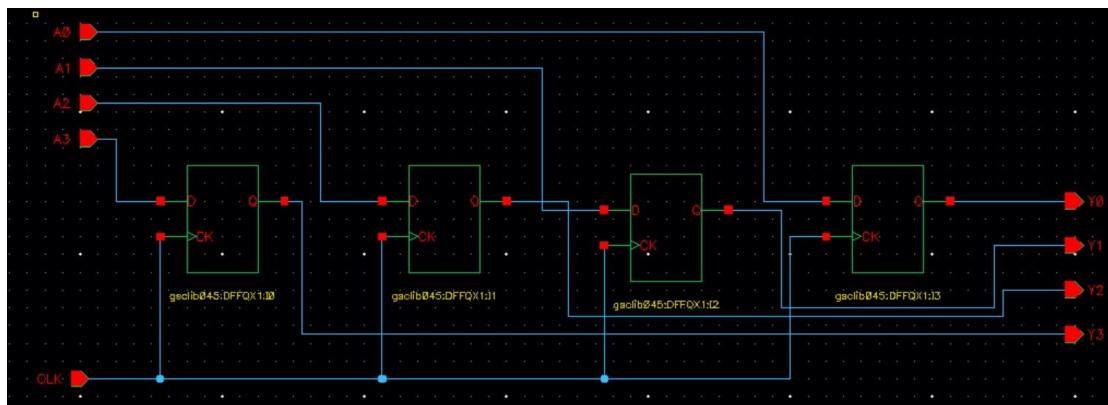


BSR:



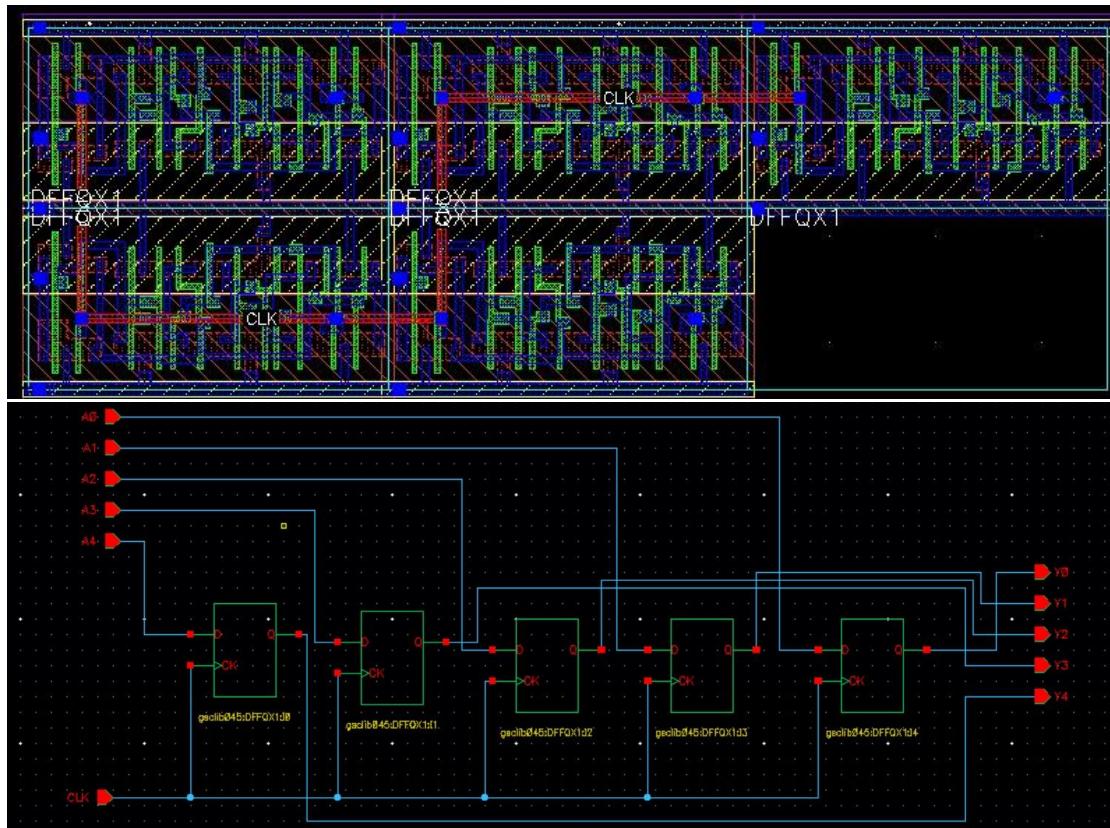


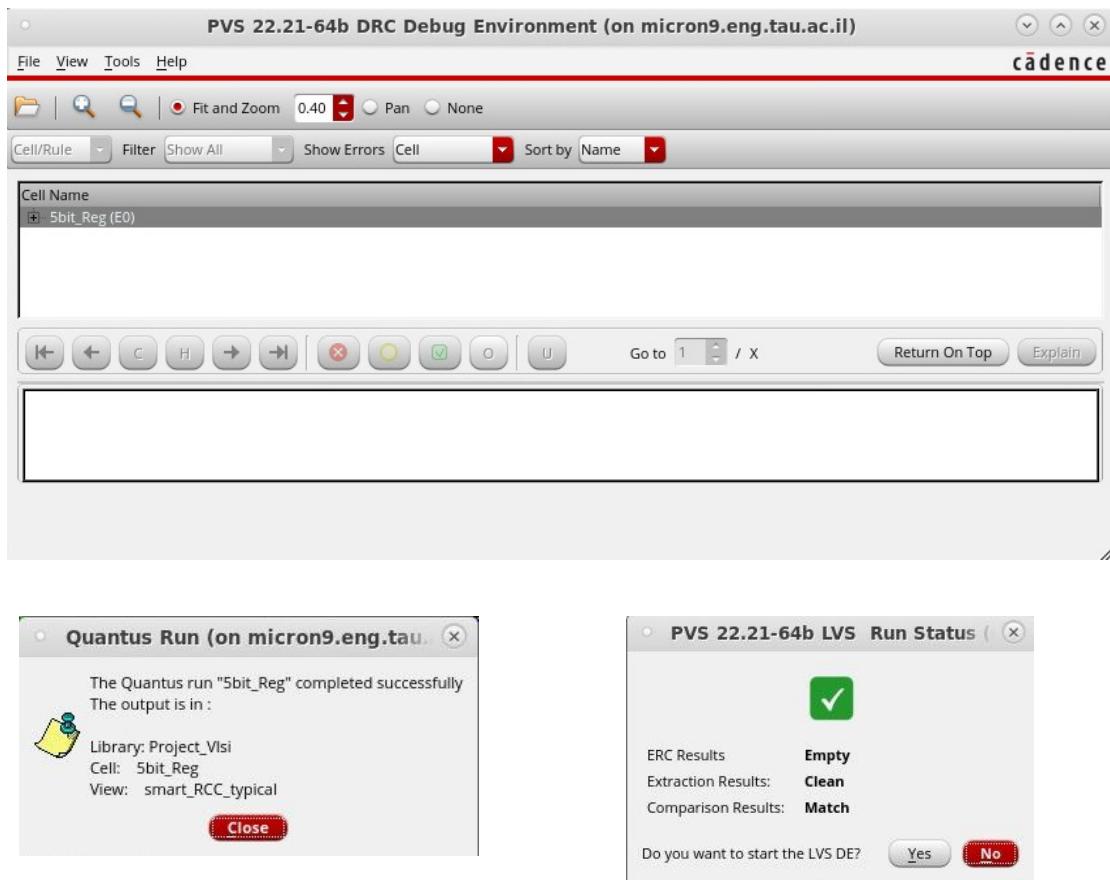
4 Bit Register:



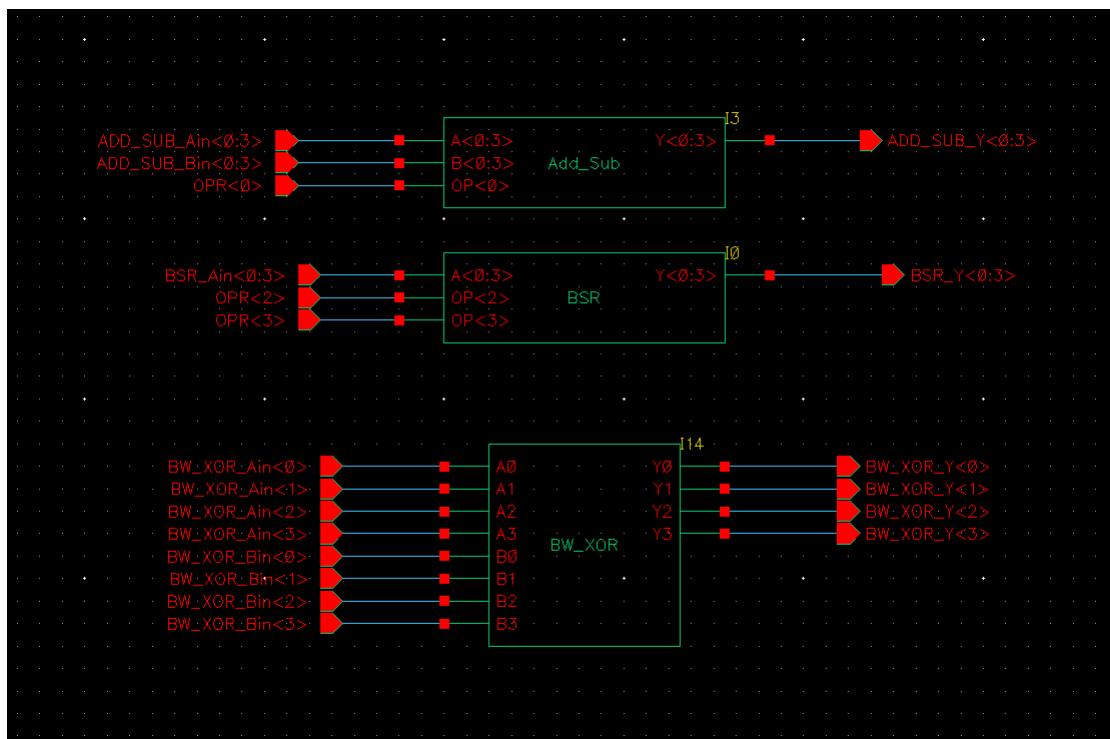


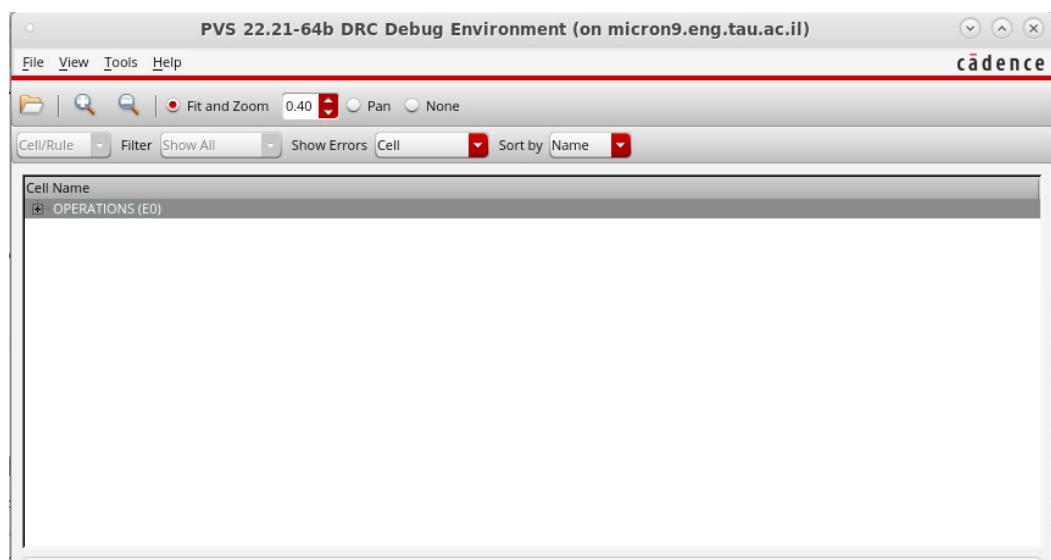
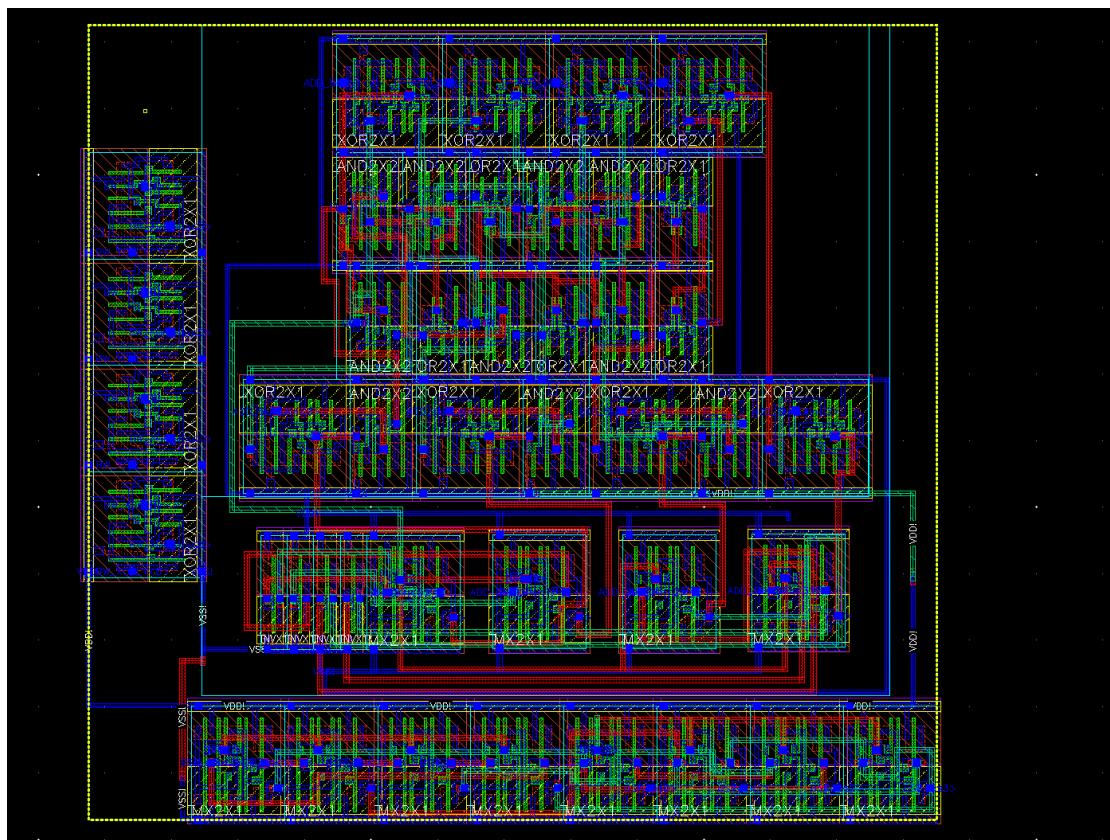
5 Bit Register:

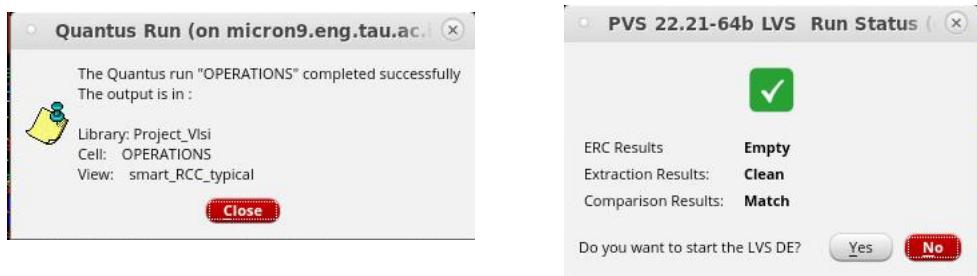




OPERATIONS:

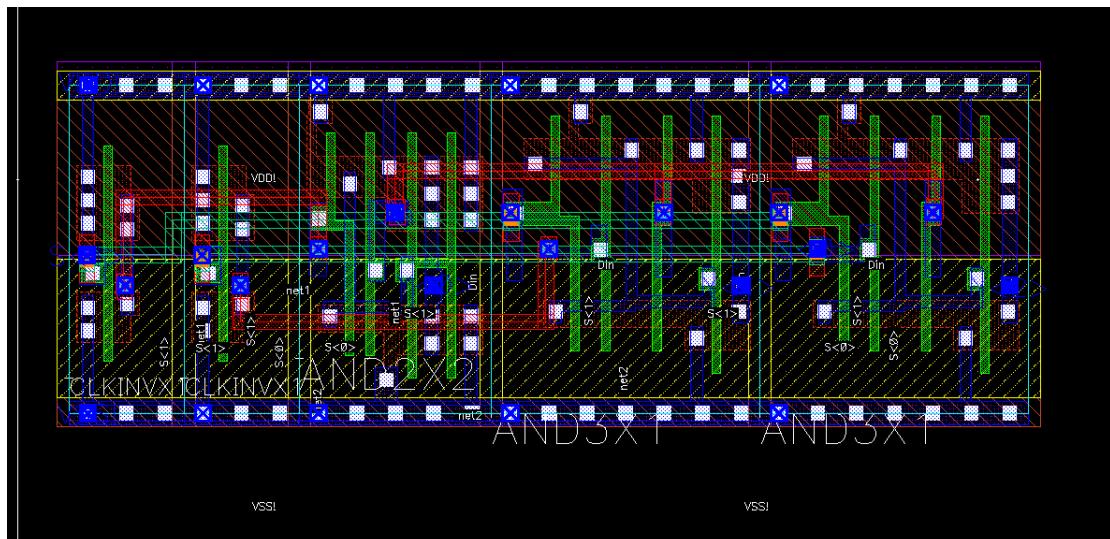
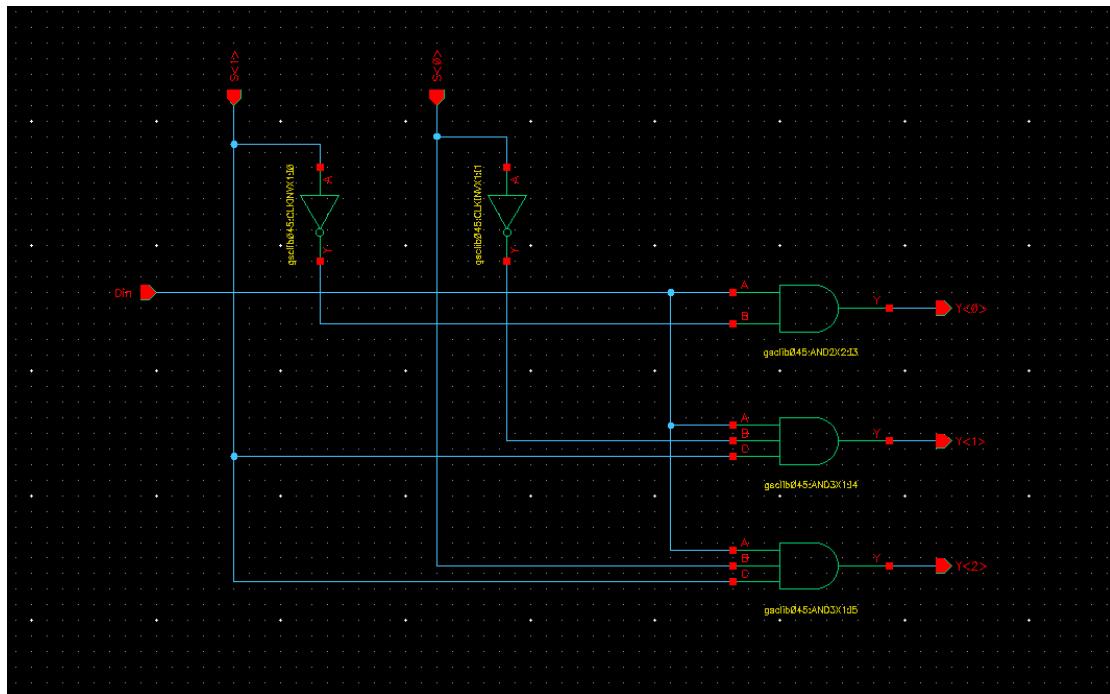


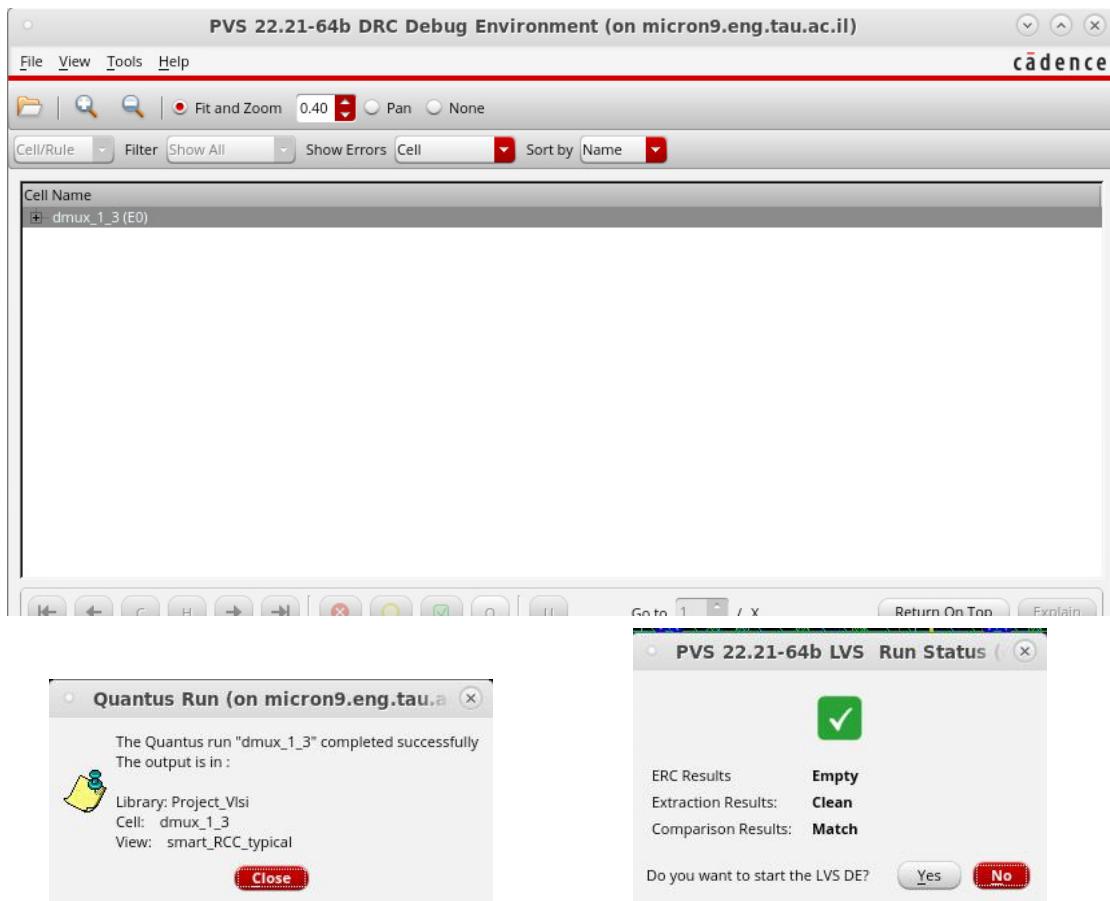




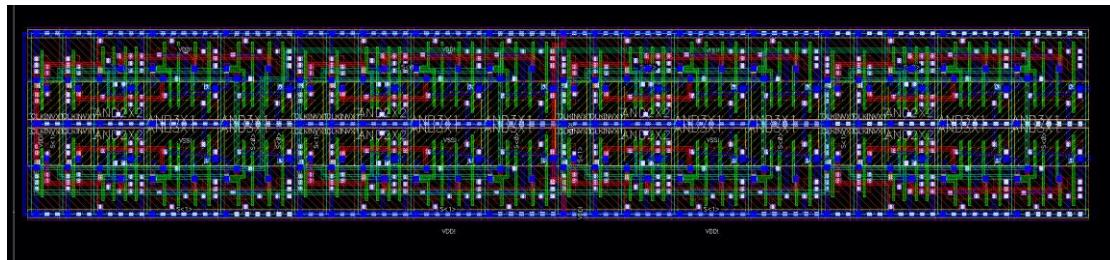
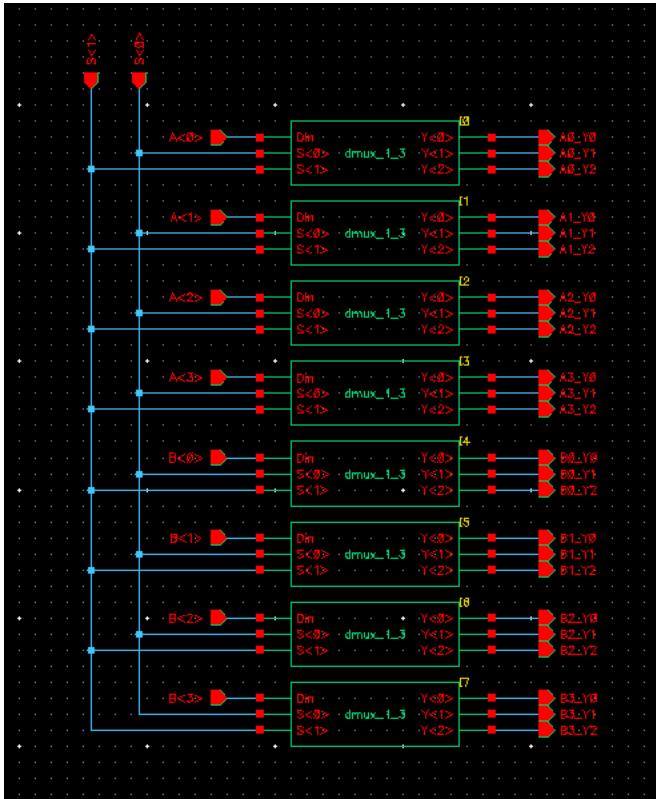
DMUX_1:3:

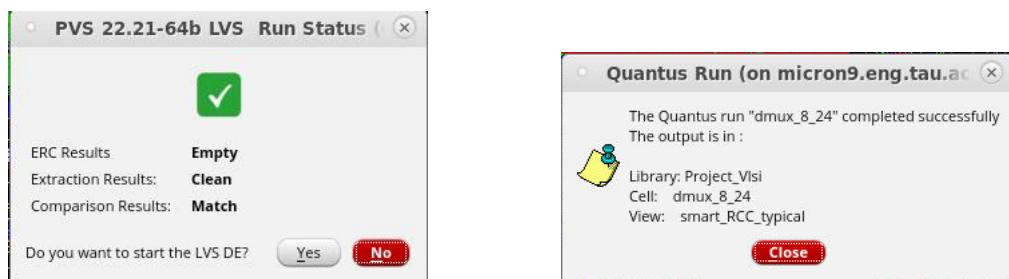
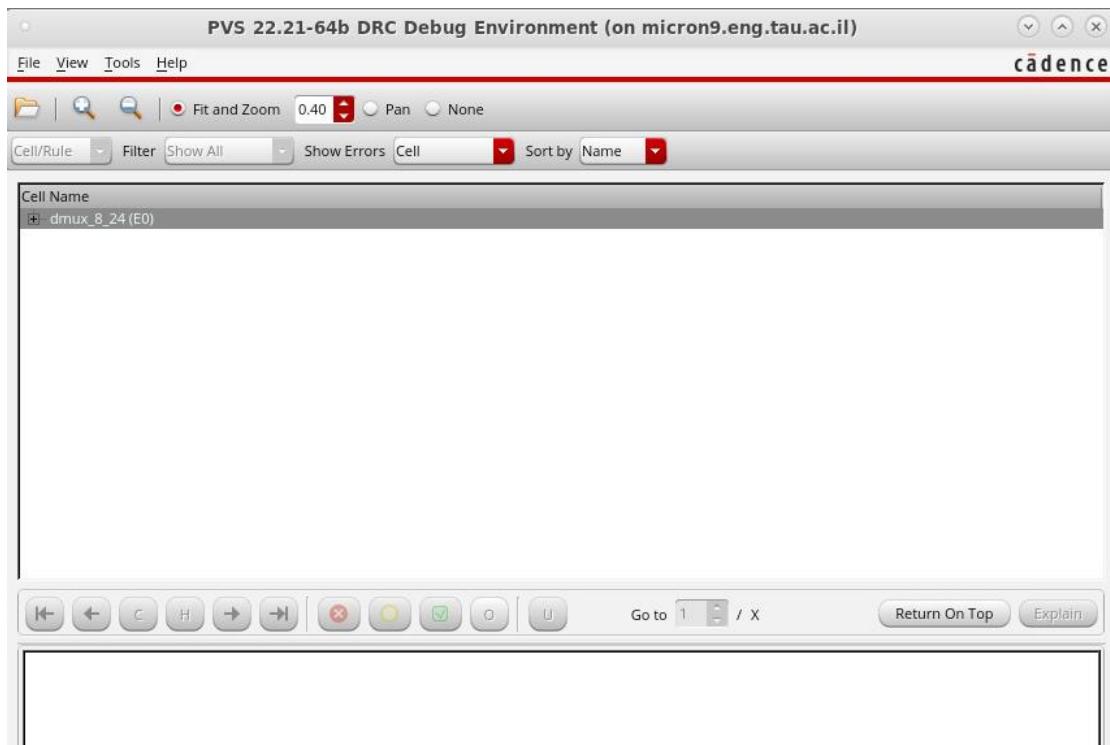
נשים לב כי רכיב זה שונה מהרכיב בשלב התכנון – חשבנו על דרך ליעול השטח מתוך מחשבה על כך שקיימים 3 בלוקים בלבד וניתן להקטין וליעול את הצורך ב-DMUX. لكن בשלב זה השתמשנו במימוש החדש.





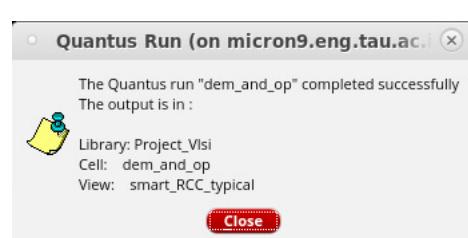
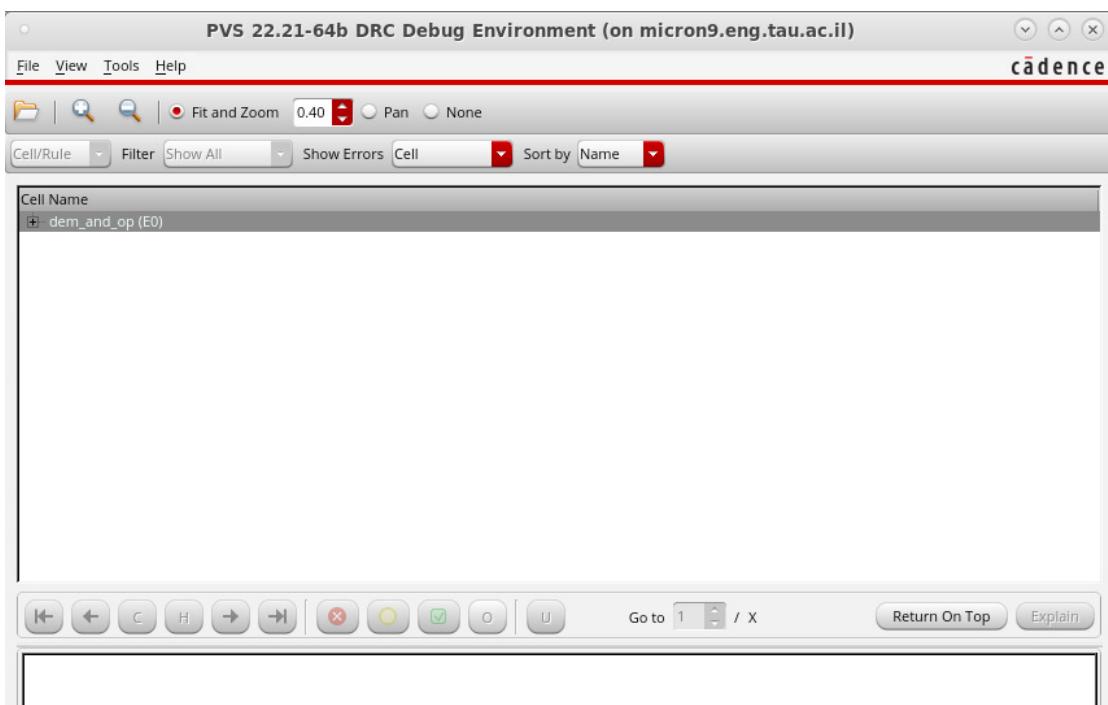
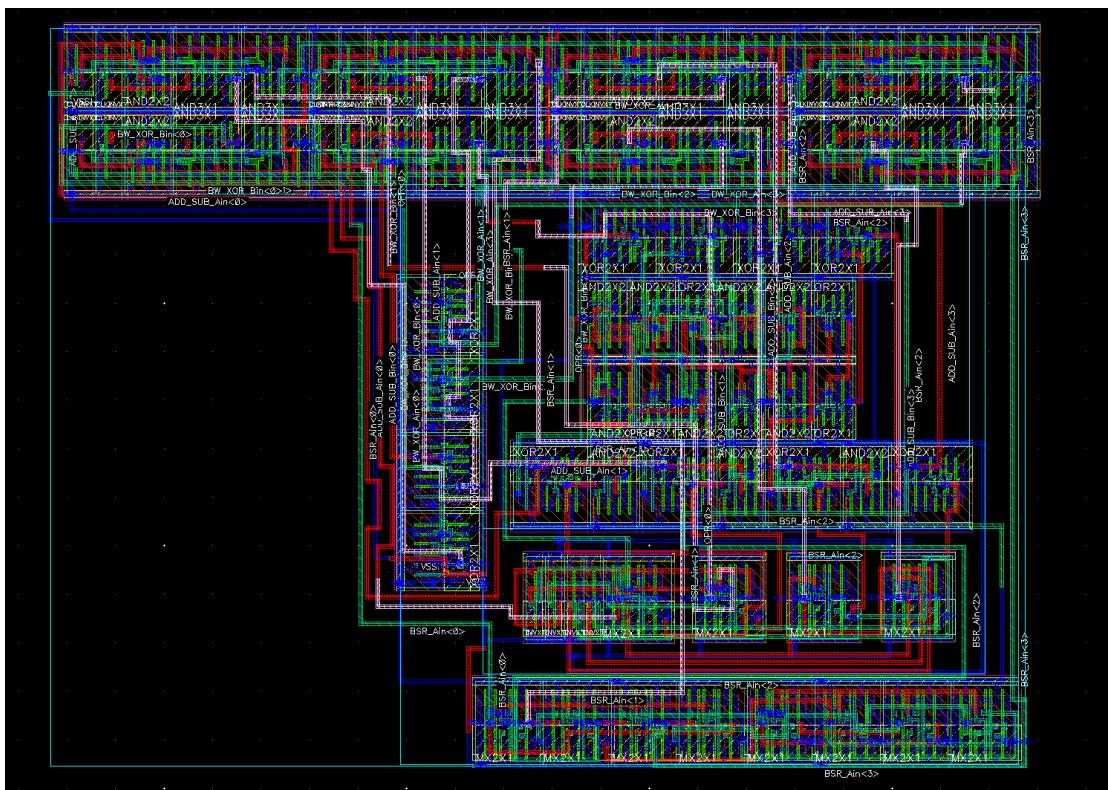
DMUX_8:24:



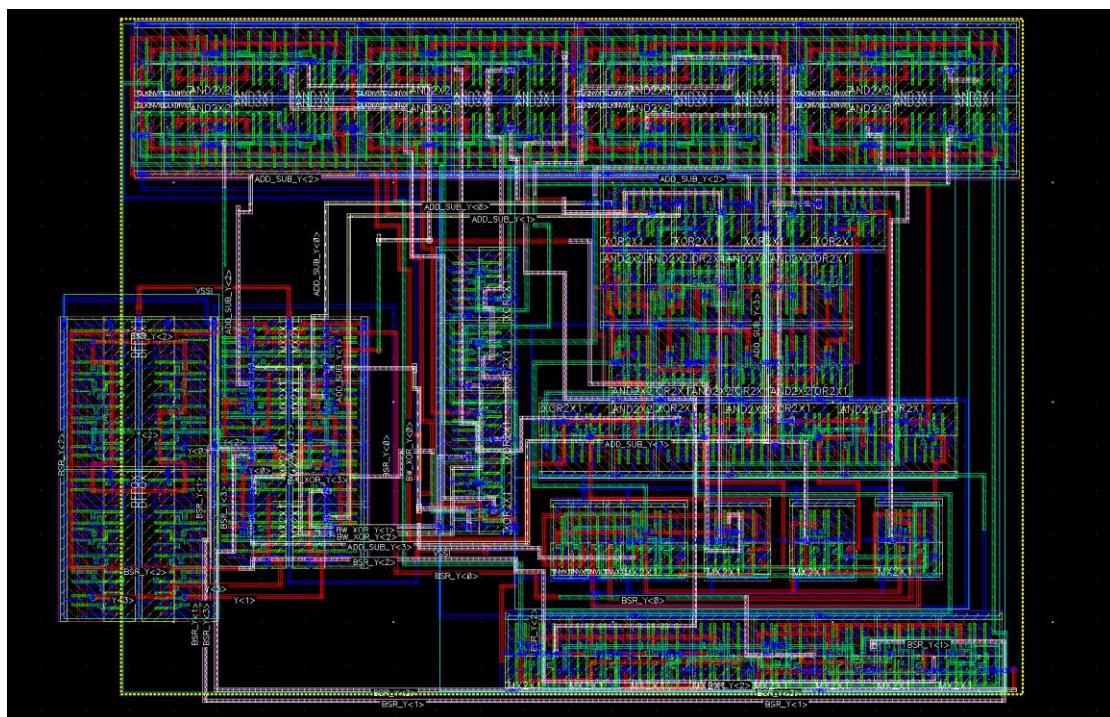


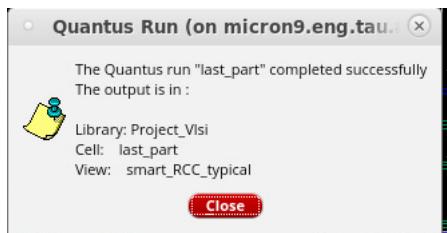
DMUX+OPERATIONS:





LAST_PART:





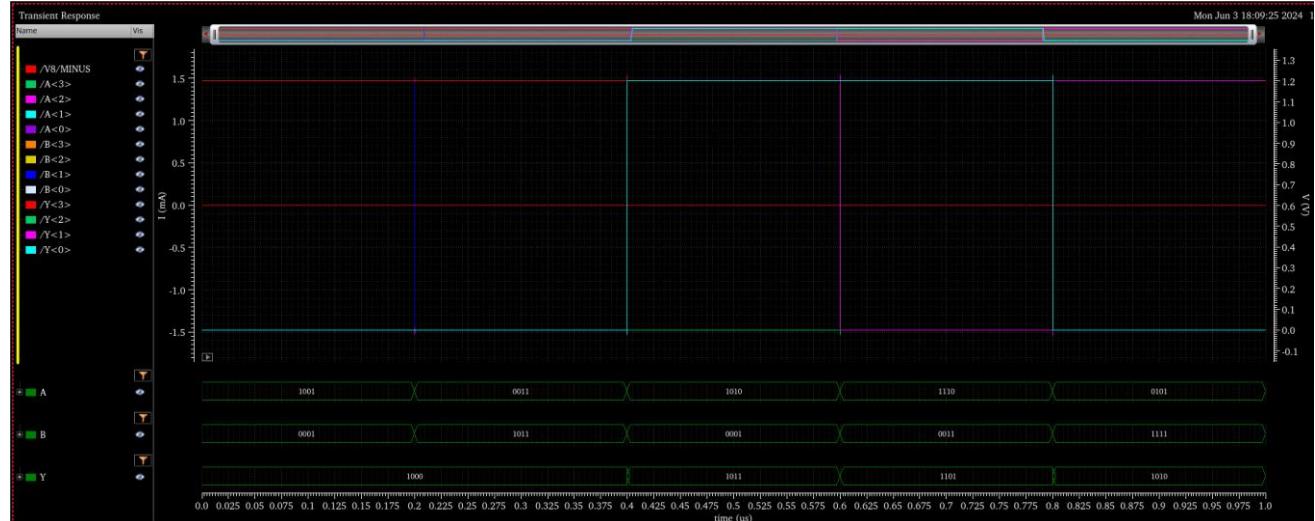
סימולציות

להלן הצגת הסימולציות של פעולות ה-ALU. עבור כל פעולה נוכיח את נוכנות הרכיב עבור קלט ופלט מתאימים.

בין הסימולציות שונים שתלויים בopcode הנבחר, הקובע איזו פעולה מתבקש לבצע. בכל אחת מהסימולציות A מייצג את הקלטים, ו-Y מייצג את הפלט של הפעולה.

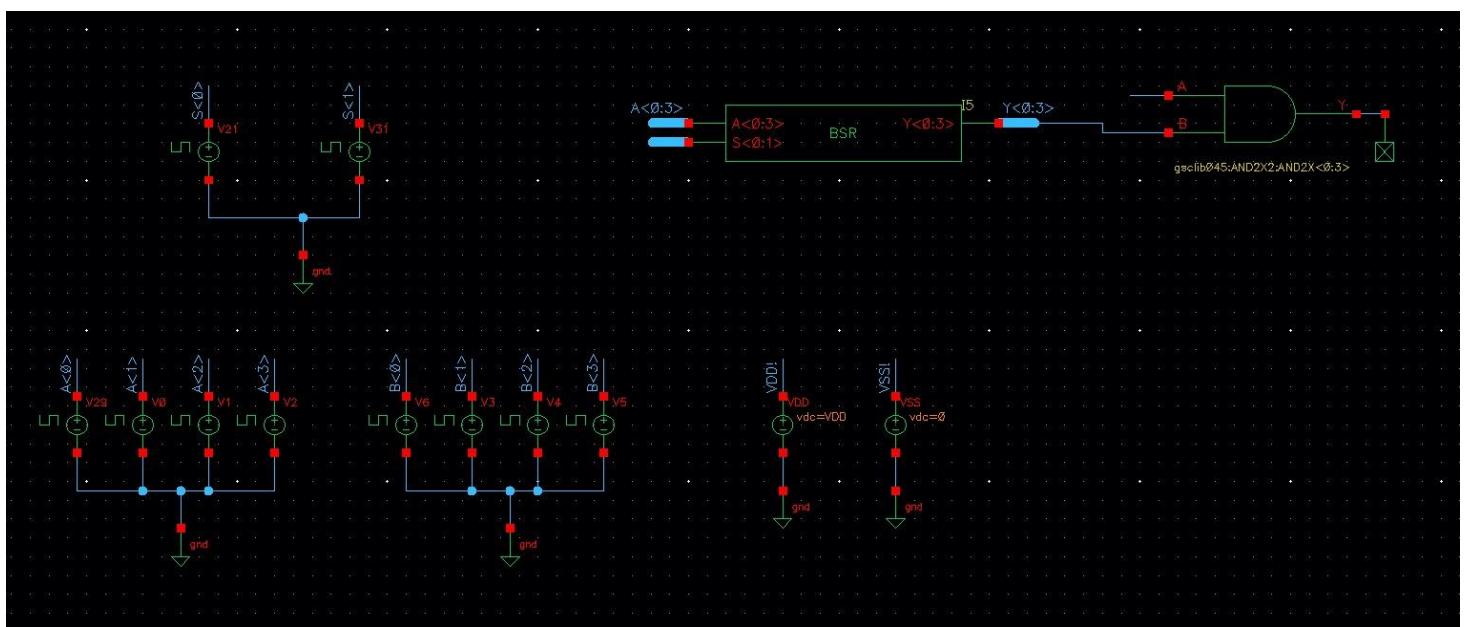
:B.W XOR

סימולציה –



:BSR

סימולציה –

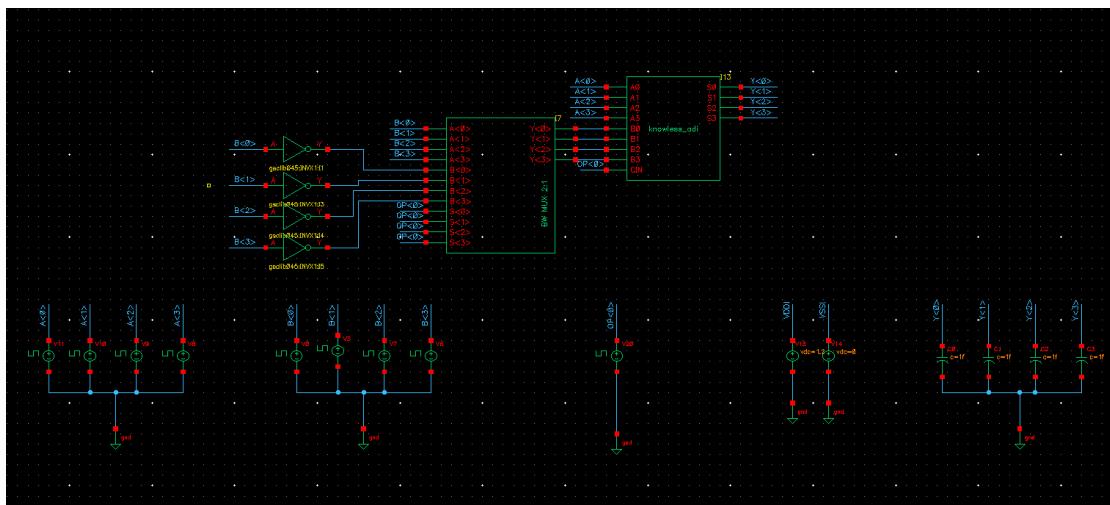


תוצאות הסימולציה –

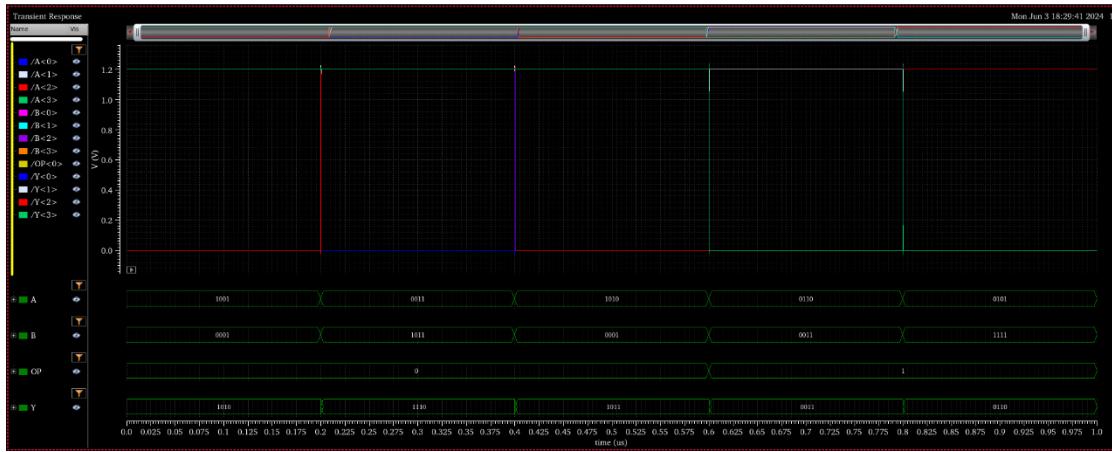


:Add, Sub

סימולציה –



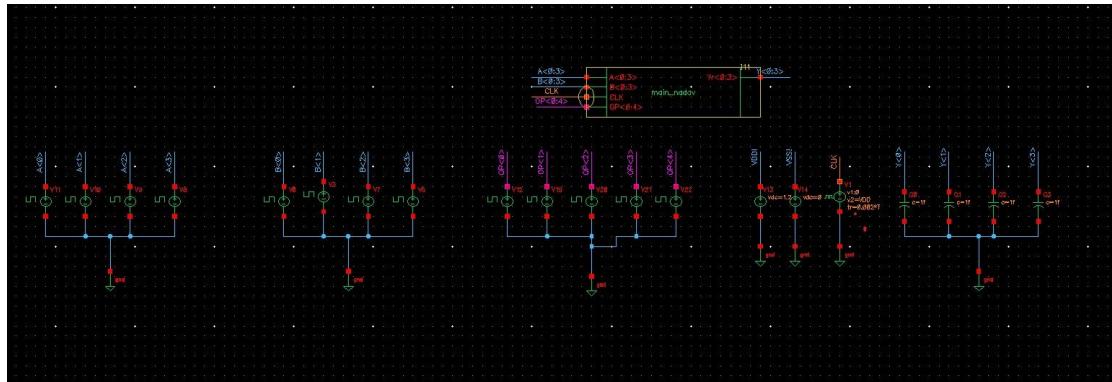
תוצאות הסימולציה –



הערה- כאן הוספנו סימולציה המרצה מספר פעולות שונות של ה ALU תוך התחשבות במשוב (אשר המשוב נלקח מהמוץא הקודם, ולא מתווך Register_Y בתוכנו המקורי שהוא שגוי)

פעולות ה- ALU הכוללת

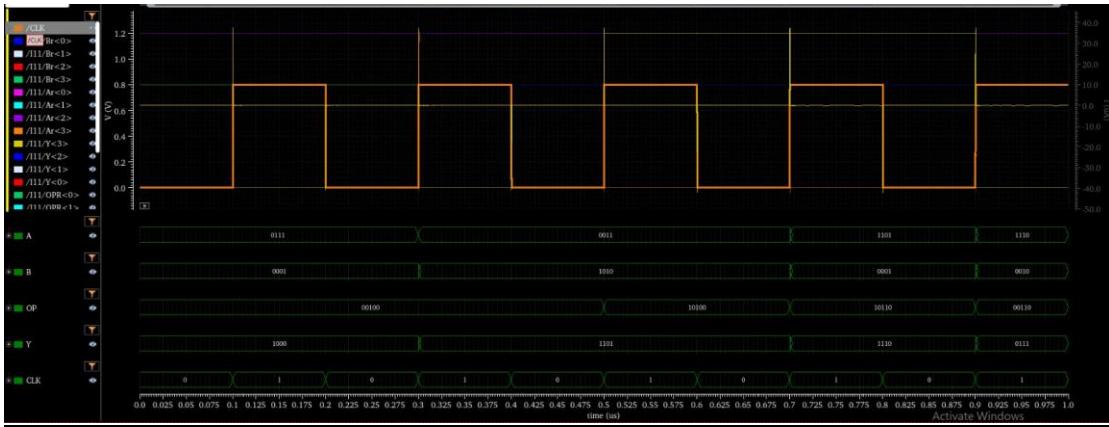
סימולציה –



הבלוק עליו מתבצעת הסימולציה. לאחר התקיקון הלוגי הנדרש



תוצאות הסימולציה –



מציאת המסלול הクリטי

בחלק זה של הפROYKT התבקשו למצוא את המסלול בעל ההשניה הגדולה ביותר, אשר נקרא המסלול הクリטי. צפינו לקבל כי המסלול הクリטי הוא עבר פועלות-h-SUB, כיוון שהוא כולל את המספר הרב ביותר של רמות לוגיות, וכן שימוש מרכיבי ברכיבי המערכת. לאחר בדיקה, התקבל כי SUB היא אכן בעל ההשניה הגדולה ביותר - המסלול הクリטי.

чисוב ההשניה הטענו באמצעות חישוב הזמן שלוקח לאוות לעבור את הלוגיקה עד לערך שהינו חצי מערך-h-DDD בבייט המוצא.

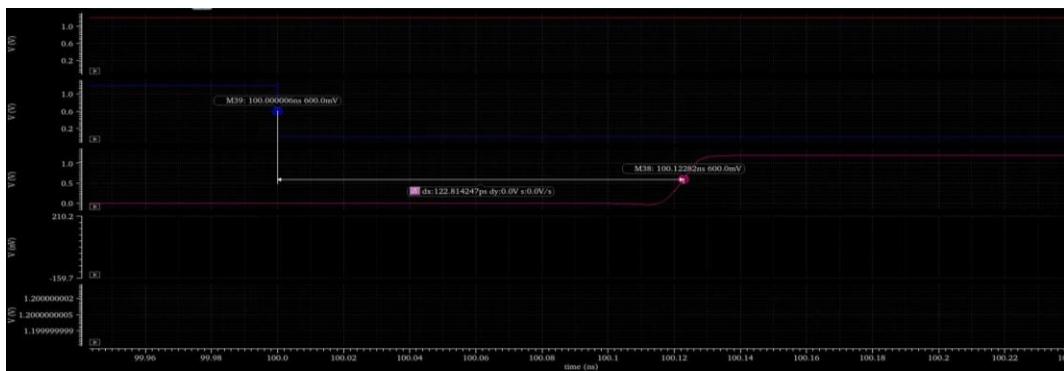
כעת נציג את חישוב זמני-h-tpL ו-h-Tpl של המוצא עבר פועלות-h-SUB במתח אספקה

.1.2V, 0.7V

הערה - כאן הוספנו התחשבות בקבלים פרזיטיים ותיקנו את הנוסחה למציאת t_p

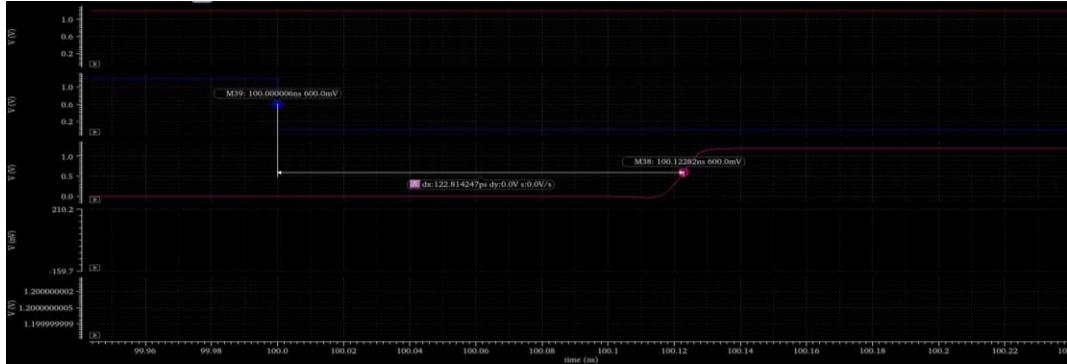
סימולציות זמניות ללא קיבולים פרזיטיים:

:Vdd=1.2V ,TpIh



מהסימולציה לעיל מקבל $TpIh=122.81[\mu s]$

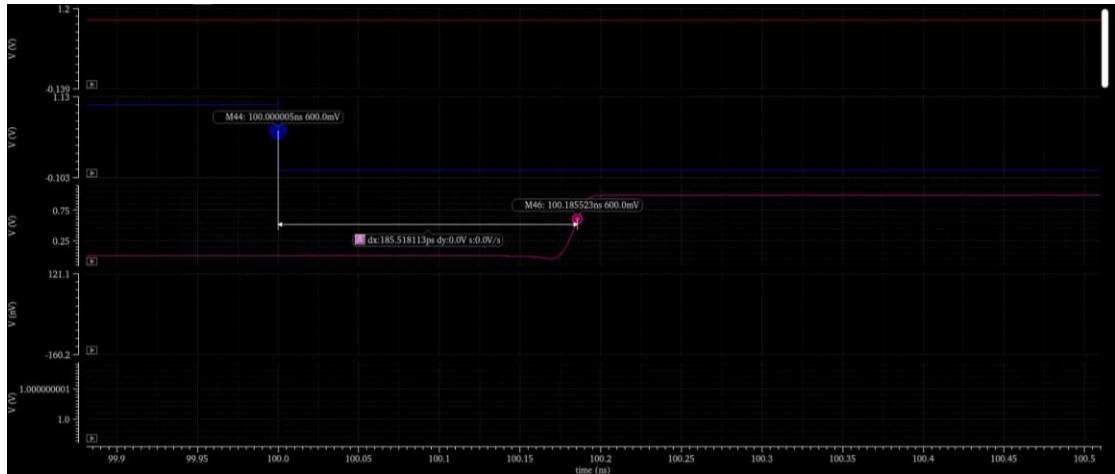
:Vdd=1.2V ,Tphl



ההסימולציה לעיל מתקבל [ps] **Tphl=98.87[ps]**

מנתונים אלו ניתן לחשב את $T_p = \max \{T_{phl}, T_{phh}\}$. לכן המסלול הקרייטי עברו מתח אספקה 1.2V **122.81[ps]** מתקבל

cut נשנה את מתח האספקה, מתקבל **T_p עבור 0.7V**:

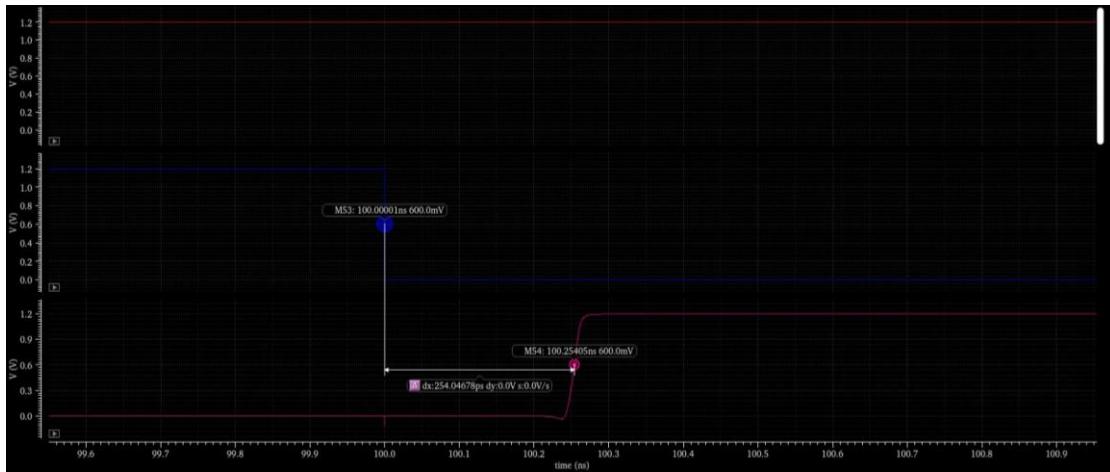


ההסימולציה לעיל מתקבל [ps] **Tp=185.52[ps]**

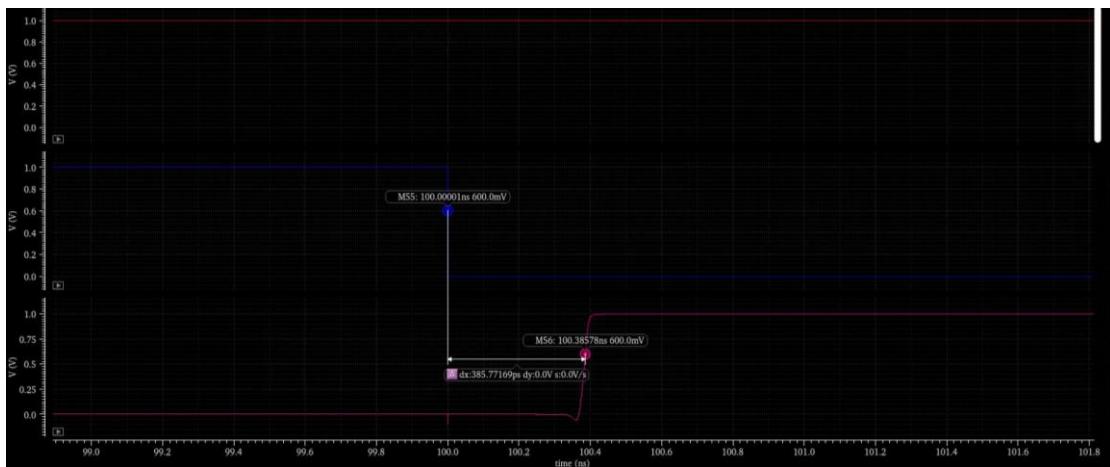
מסימולציות אלו עולה כי ככל שמתוך האספקה קטן, כך הערך של T_p גדול.

cut נוסף התחשבות ב-QRC ונבצע את הבדיקות בשנית –

קד עבור 1.2V כולל קיבולים פרזיטיים:



קד עבור 0.7V עם קיבולים פרזיטיים:



ניתן לראות שעבור הסימולציות עם הקיבולים הפרזיטיים התקבל ערך קד גדול יותר. בנויסט, נשמרת המגמה שצוינה לעיל, לפיה ככל שמתוח האספקה קטן כך קד גדול.

בהתאם לתוכנו המקבדים, המסלול הקריטי יורכב מרכיבים הבאים –

$$t_p = T_{cq1} + t_{MUX2:1} + t_{DEMUX1:3} + t_{ADD-SUB} + t_{MUX4:1} + T_{setup2}$$

ראינו כי עבור הרכיב ADD-SUB, SUB הינו בעל ההשאה הגדולה ביותר, לכן בчисוב הקט נתחשב בו.

לכן זמן המחזור המינימלי והתדר המקסימלי בו ניתן לעבוד כדי לעמוד ב- *TIME SETUP* הינו:

$$T \geq t_{ADD-SUB} + t_{DEMUX1:3} + 2 * t_{MUX2:1} \rightarrow f \leq \frac{1}{T}$$

עבור הרכיב אותו מדדנו מתקבל:

מתח אספקה [V]	פריזיטים	[ק[ס]ps] קט כולל קיבולים	תדר מקסימלי f[GHz]
1.2	254.05	122.81	8.14
0.7	385.77	185.52	5.39