דנה ניפדוב	209381649
אווה פולוליאחוב	321882649

ספרתיות - רטוב 1

סעיף 2.1

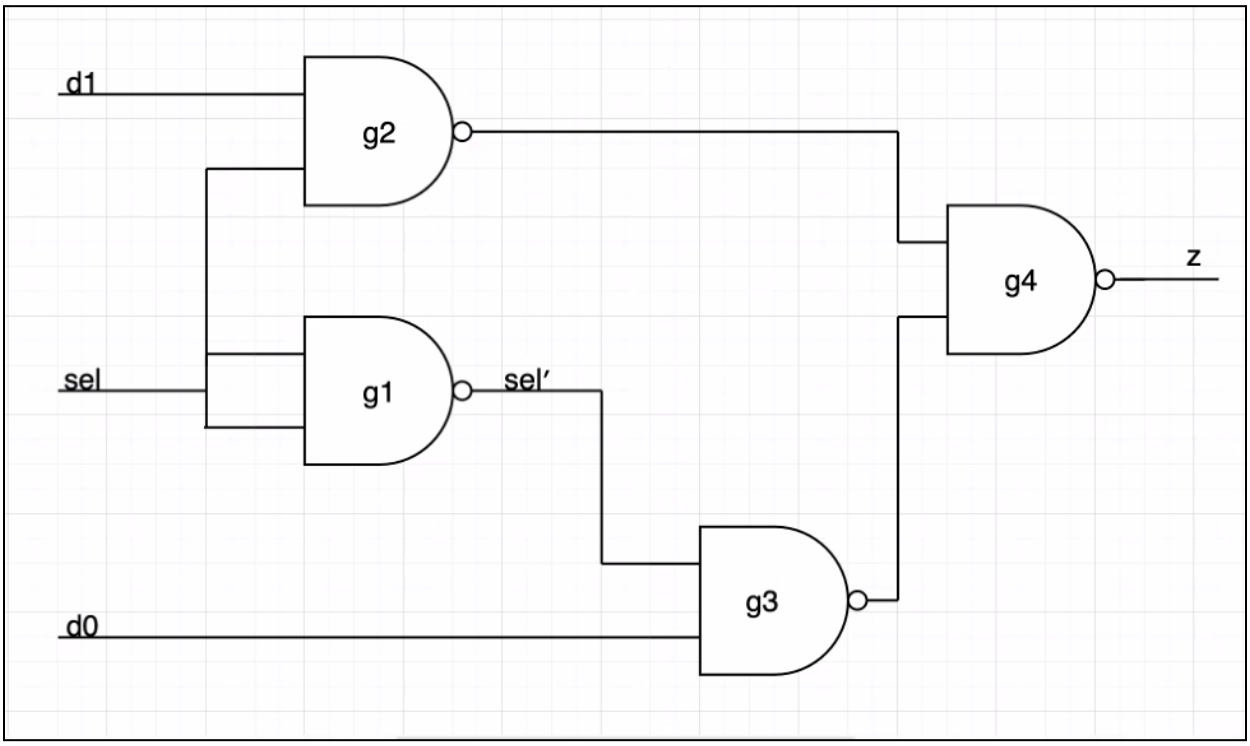
תחילה נציג טבלת אמת:

z	sel	d_1	d_0
0	0	0	0
0	1	0	0
0	0	1	0
1	1	1	0
1	0	0	1
0	1	0	1
1	0	1	1
1	1	1	1

כלומר, נקבל את הפונקציה הבאה:

$$z = d_0 \cdot sel' + d_1 \cdot sel$$

ולכן השרטוט שנקבל:



לפי ת.ז 321882649, נקבל את טבלת ה- Tpd הבאה:

Gate	Tpd_{LH}	Tpd_{HL}
NAND2	2	8
OR2	1	2
XNOR2	8	6

path	d_0	d_1	sel	Tpd
d0g3g4	0->1	0	0	10
d0g3g4	1->0	0	0	10
d0g3g4	0->1	0	1	Z doesn't change
d0g3g4	1->0	0	1	Z doesn't change

d0g3g4	0->1	1	0	10
d0g3g4	1->0	1	0	10
d0g3g4	1->0	1	1	Z doesn’t change
d0g3g4	0->1	1	1	Z doesn’t change
d1g2g4	0	0->1	0	Z doesn’t change
d1g2g4	0	1->0	0	Z doesn’t change
d1g2g4	0	0->1	1	10
d1g2g4	0	1->0	1	10
d1g2g4	1	0->1	0	Z doesn’t change
d1g2g4	1	1->0	0	Z doesn’t change
d1g2g4	1	0->1	1	10
d1g2g4	1	1->0	1	10
sel-g1g3g4 g2g4	0	0	0->1	Z doesn’t change
sel-g1g3g4 g2g4	0	0	1->0	Z doesn’t change
sel-g1g3g4 g2g4	0	1	0->1	Z doesn’t change
sel-g1g3g4 g2g4	0	1	1->0	Z doesn’t change
sel-g1g3g4 g2g4	1	0	0->1	10
sel-g1g3g4 g2g4	1	0	1->0	10
sel-g1g3g4 g2g4	1	1	0->1	Z doesn’t change
sel-g1g3g4 g2g4	1	1	1->0	Z doesn’t change

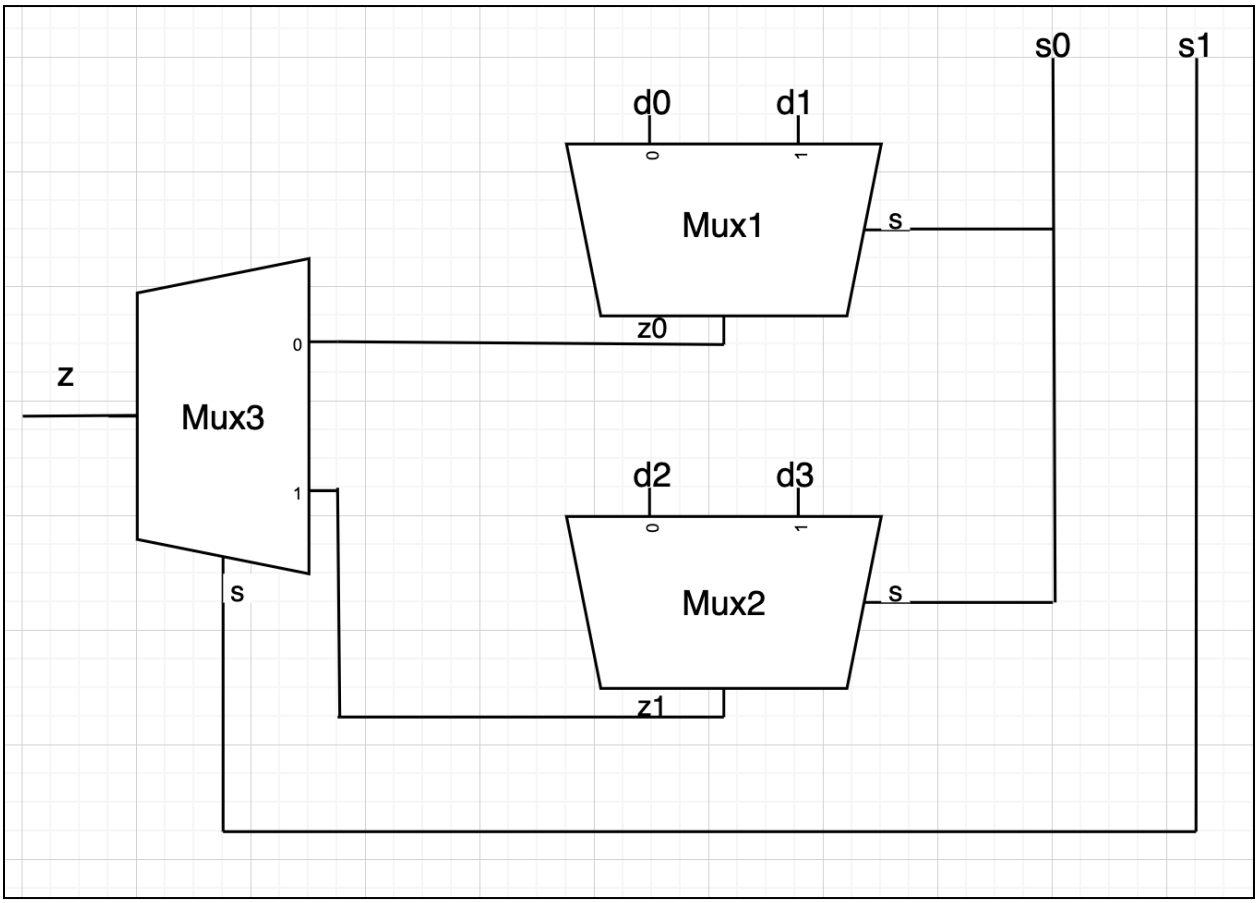
טבלת השהיות נוכחית:

Gate	Tpd_{LH}	Tpd_{HL}
NAND2	8	8
OR2	2	2
XNOR2	8	8

סעיף 2.2

נסמן תחת MUX את הלוגיקה שמימשנו בסעיף א'.
ביחס לנתוני ה Tpd העדכניים, מתקיים : $Tpd_{max}(Mux) = 16$

הדיאגרמה של Mux4:

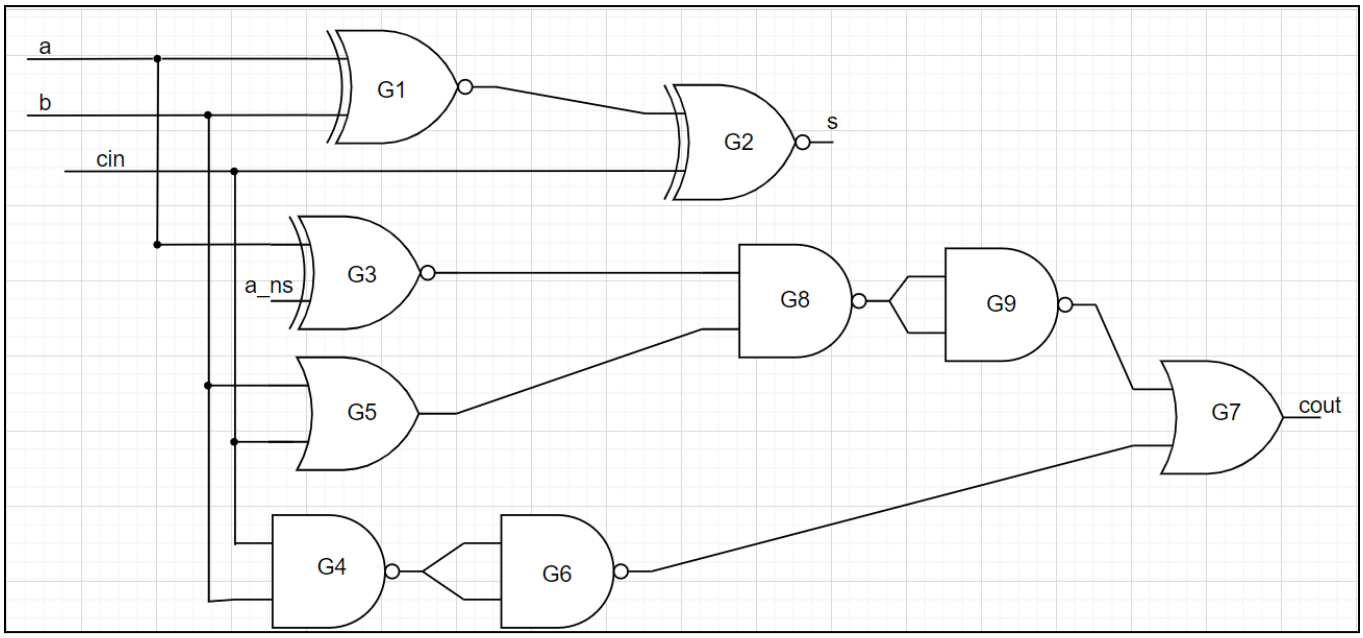


כעת נחשב זמן השהייה מקסימלי עבור שינוי יציב במערכת:

path	d_0	d_1	d_2	d_3	sel_0	sel_1	Tpd_Z
$sel_0 \rightarrow Mux2 \; -> \; Mux3$	0	0	0	1->0	1	1	32
$sel_0 \rightarrow Mux2 \; -> \; Mux3$	0	0	0	0->1	1	1	32

<u>סעיף 2.3</u>

<u>הדיאגרמה של FAS:</u>



נחשב את ההשהיות המקסימליות מכל כניסה לכל יציאה:

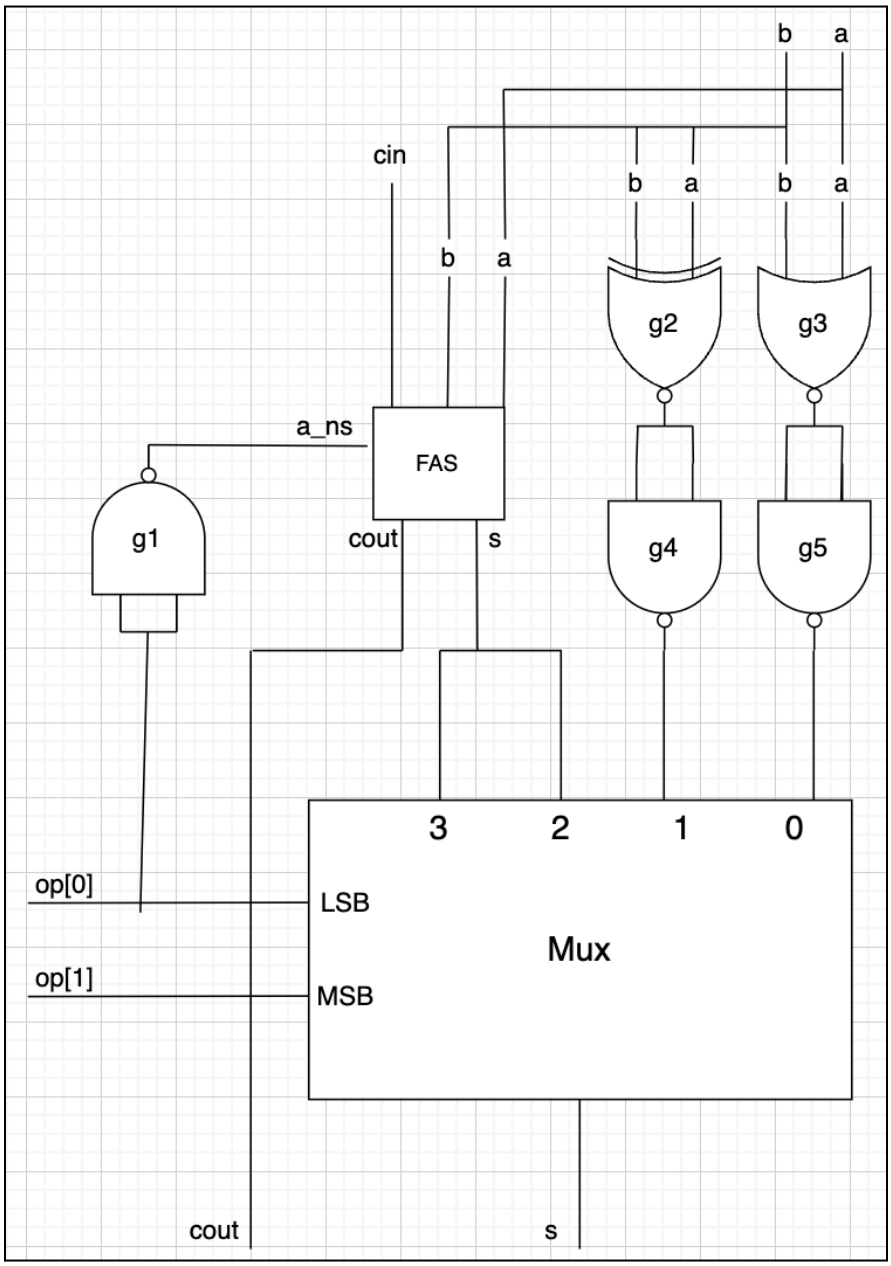
path	a	b	cin	a_ns	Tpd_{cout}	Tpd_{s}
a->G3->G8->G9->G7	1->0	1	0	1	26	16
a->G3->G8->G9->G7	0->1	1	0	1	26	16
b->G5->G8->G9->G7	0	1->0	0	0	20	-
b->G5->G8->G9->G7	0	0->1	0	0	20	-
cin->G5->G8->G9->G7	0	0	1->0	0	20	-
cin->G5->G8->G9->G7	0	0	0->1	0	20	-
a_ns->G3->G8->G9->G7	1	1	0	1->0	26	16
a_ns->G3->G8->G9->G7	1	1	0	0->1	26	16

ה- Tpd המקסימלי הינו:

$$Tpd_{max}(FAS) = 26$$

סעיף 2.4

הדיאגרמה של ALU:



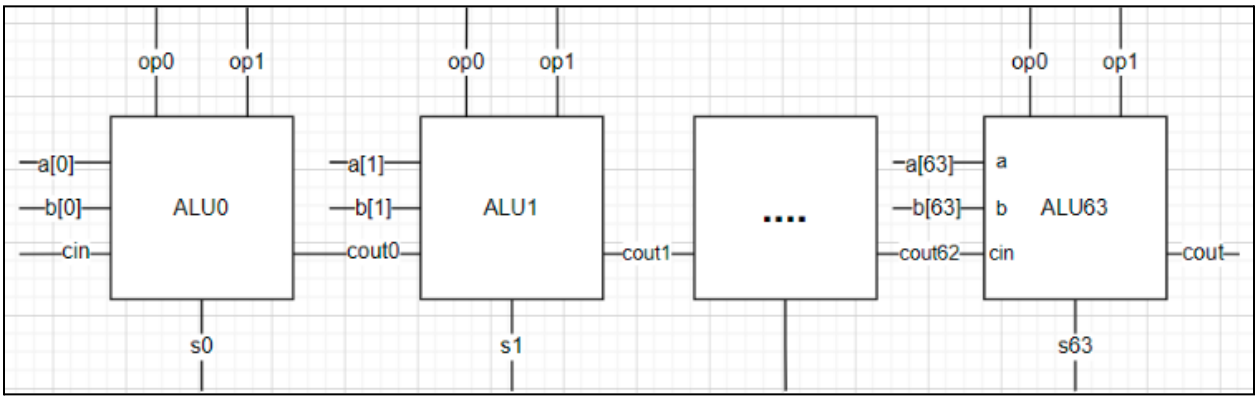
ניעזר ברכיב ה Mux וברכיב ה FAS שמימשנו בסעיפים 2.2 2.3.

path	a	b	cin	$op[0]$	$op[1]$	Tpd_s	Tpd_{cout}
a->FAS->Mux	1->0	1	0	1	0	58	32
a->FAS->Mux	0->1	1	0	1	0	58	32
b->FAS->Mux	1	1->0	0	1	0	58	32
b->FAS->Mux	1	0->1	0	1	0	58	32
cin->FAS->Mux	0	0	1->0	0	0	58	32
cin->FAS->Mux	0	0	0->1	0	0	58	32
op[0]->NAND->FAS	0	1	0	1->0	0	34	34
op[0]->NAND->FAS	0	1	0	0->1	0	34	34
op[1]->Mux	1	0	1	1	1->0	16	-
op[1]->Mux	1	0	1	1	0->1	16	-

ה- Tpd המקסימלי הינו:

$$Tpd_{max}(cout_{ALU}) = 34\,,\ Tpd_{max}(s_{ALU}) = 58$$

סעיף 2.5



ניעזר ב 64 רכיבי ALU מצורת רכיב ה ALU שמימשנו בסעיף 2.4.

<u>אלגוריתם החיבור:</u>

לכל $0 \le i \le 63$:

את הביט ה i של הוקטורים a , b נחבר בתור הכניסות של הקלטים בכל אחד מרכיבי ה ALU.

כעת, את ה $cout_i$ ה i נחבר ל cin_{i+1} .

בעבור ה cin הראשון (במקום ה 0) נאתחל ל1.

בכל רכיבי ה ALU נבחר $op = 11$.

כלומר בכל פעם, הרכיב בוחר לבצע פעולת חיסור בין שני הקלטים באותו הרכיב.

במצב ההתחלתי, נחסר 63 פעמים 0-0 ותוצאת החיסור, כלומר ה cin של הרכיב הבא תהיה 0 ולא תשפיע כך שהתוצאה הסופית תהיה 0.

נתבונן במקרה בו נשנה $a[0] : 0 \; -> \; 1$:

בהינתן השינוי, נקבל שינויים ב $cout$ וגם ב s של כל הרכיבים. מהצורה 0->1 .

נרצה לחשב את הפרש הזמנים בין קבלת שינוי ב $cout$ ועד להתייצבות השינוי האחרון של s .

מחישוב $Tpd_{cout_{ALU}}$, השינוי ב $cout$ יובחן ברכיב הבא ALU_1, לאחר 34_{ns} ולכן נתחיל את החישוב מרגע זה שיוסמן ב t_0.

כעבור 18_{ns} יתקבל שינוי ב s_{FAS} של ALU_1 ואז הכניסות 1,2 של Mux במשך של $tpd(Mux) = 16_{ns}$ כך שכעבור 34_{ns} סך הכל תשתנה היציאה s_1.

בנוסף נקבל שינוי ב $cout_{FAS}$, שזה שקול לזמן בו $cout_{ALU_1}$ משתנה ומתחיל השינוי ב cin ברכיב הבא.

החל משינוי זה ב cin_{ALU_2} נקבל התייצבות של זמני התעדכנות היציאות, כך שהפרשי השינויים בין שינוי ב $cout_{ALU} = tpd(cout_{FAS})$ לבין התעדכנות היציאה $s_1 = tpd(ALU)$ הוא 20_{ns}.

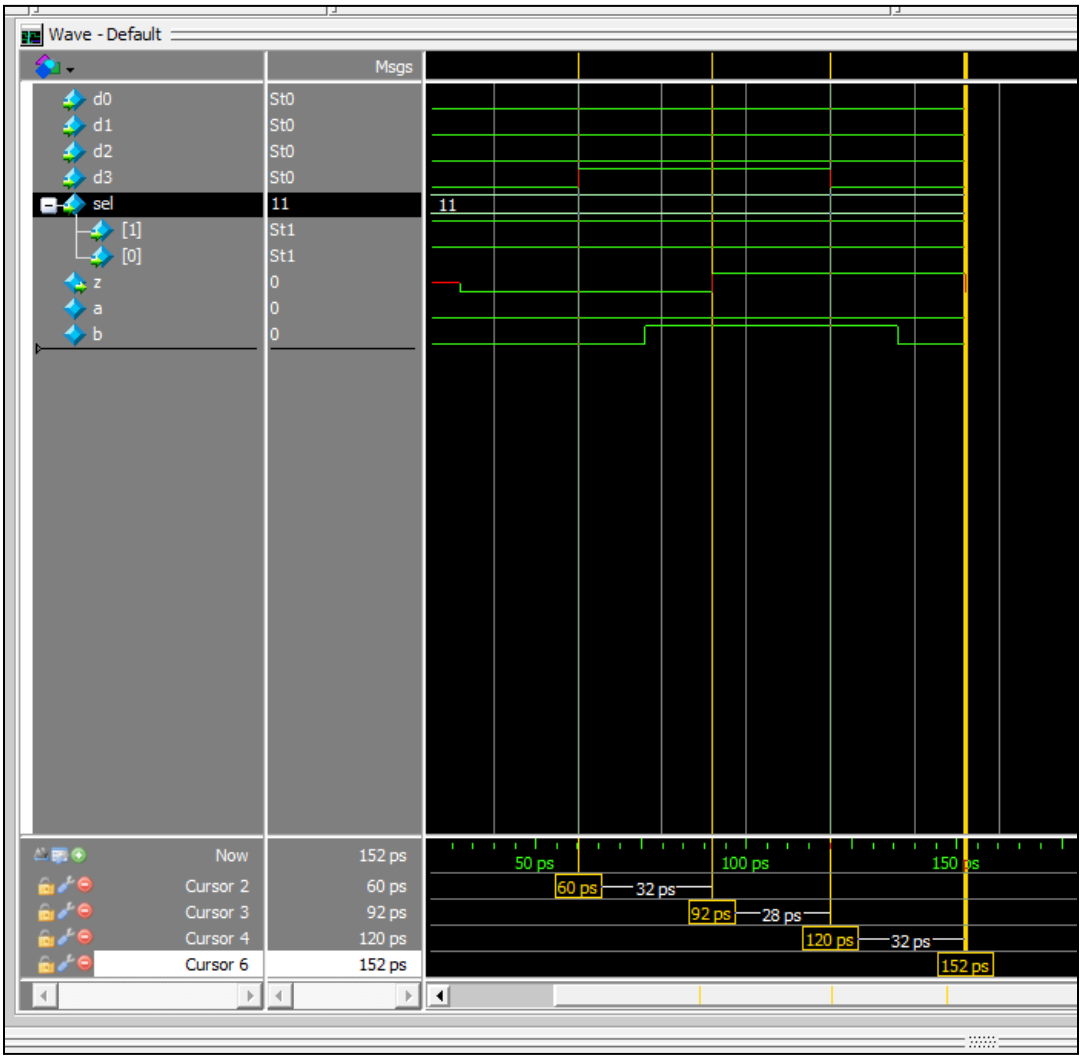
לכן סה"כ זמן :

$$updateS_0 + updateS_1FromCout_0update + updateTheRest = 48_{ns} + 18_{ns} + 62 \cdot 20_{ns}$$

חלק רטוב

סעיף 3.3

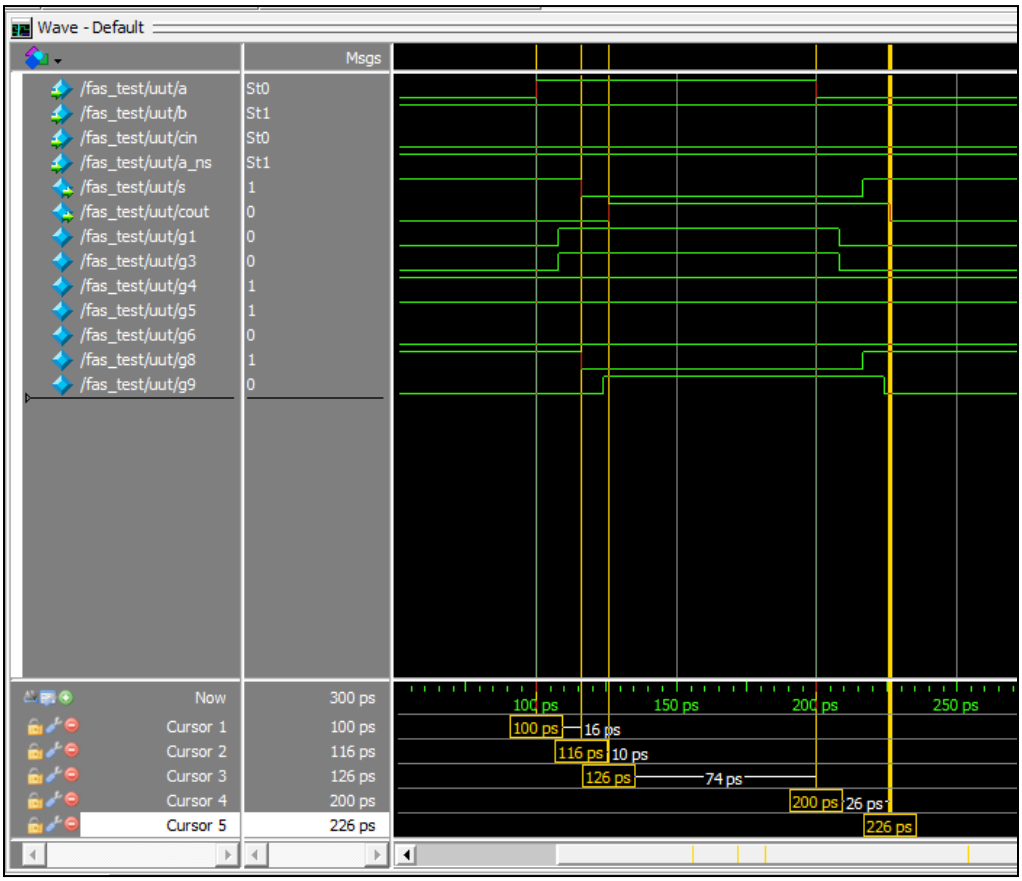
3.3. בקובץ mux4_test.sv כתבו testbench עבור המודול mux4 שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן בשני המקרים שנבחרו בסעיף 2.2.



נבחין כי בתחילת הסימולציה הצבנו את הערכים $d_3 = 0$. בזמן $t = 60_{ns}$ עשינו הצבה d_3: $0 -> 1$
ואכן בזמן $t = 92_{ns}$ נבחין בשינוי בפלט z: $0 -> 1$. ולכן אכן מתקיים $Tpd_{LH} = 92 - 60 = 32_{ns}$.
מאידך, נבחין כי בזמן $t = 120_{ns}$ עשינו הצבה d_3: $1 -> 0$ ואכן בזמן $t = 152_{ns}$ נבחין בשינוי בפלט
z: $1 -> 0$. ולכן אכן מתקיים $Tpd_{HL} = 152 - 120 = 32_{ns}$.

סעיף 3.5

3.5. בקובץ fas_test.sv כתבו testbench עבור המודול fas שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן בשנים מהמקרים שנבחרו בסעיף 2.3.

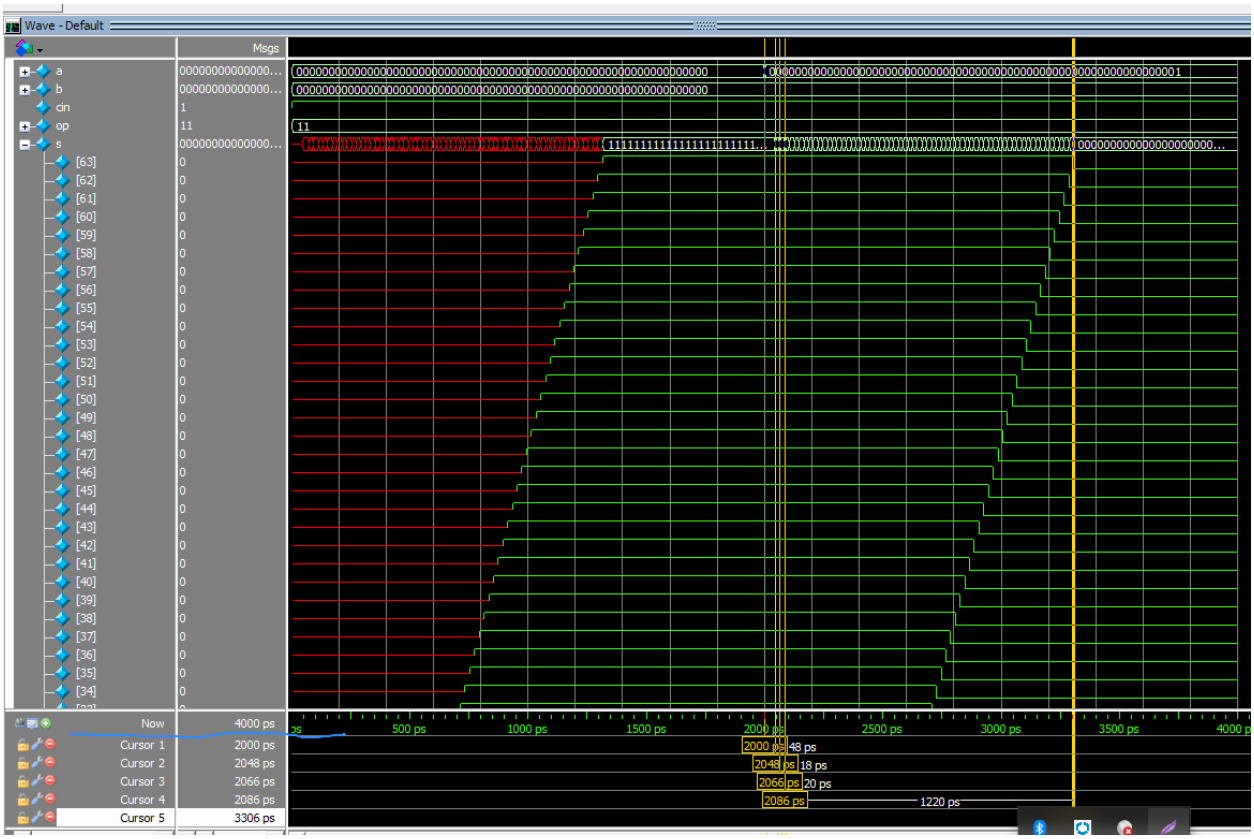


ניתן לראות כי ה- Tpd_{cout} הינו 26 וה- Tpd_s הינו 16 כפי שציפינו החלק התיאורטי.

סעיף 3.8

3.8. בקובץ alu64bit_test.sv כתבו testbench עבור המודול alu64bit שבניתם. מטרת ה-testbench היא לוודא נכונות לוגית של התכן שנבנה, וכן תאימות של ההשהיות בסימולציה לחישובים התאורטיים.
יש לבדוק את התכן עבור שינוי הכניסות שגורם להשהיה הארוכה ביותר עד למוצא, שחושב בסעיף 2.5.
על ה-testbench לבצע את רצף הפעולות הבא:
- הצבת כל הכניסות בערכים המתאימים לפני השינוי הגורם להשהיה הארוכה ביותר
- המתנה להתייצבות היציאה
- שינוי הכניסה הגורם להשהיה הארוכה ביותר
- המתנה להתייצבות היציאה

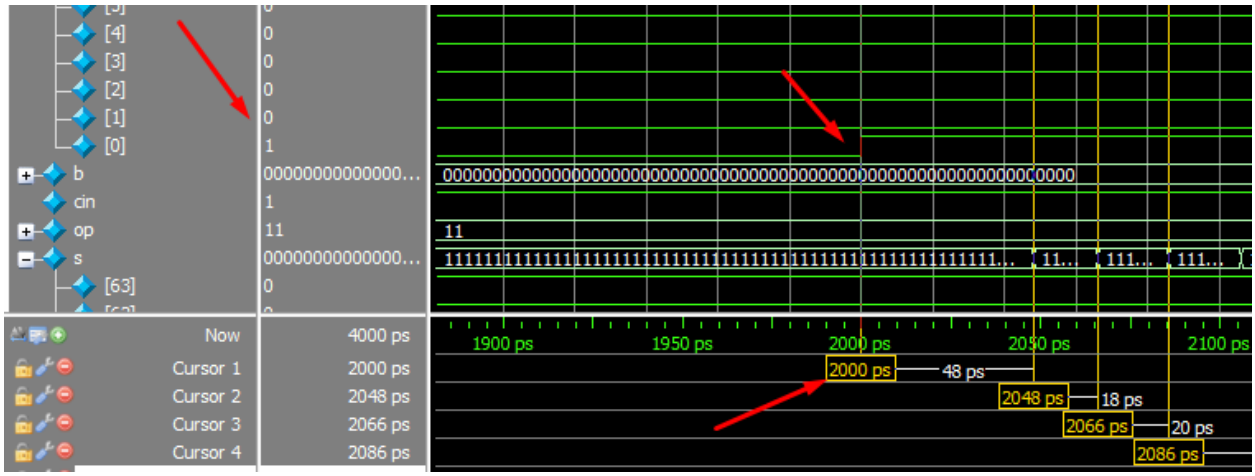
צרפו לחלק היבש רק את תוצאות הסימולציה (צילום מסך של דיאגרמת הגלים המתקבלת) והסבירו את התוצאות המתקבלות בדיאגרמה. הראו שההשהיות אכן מתאימות לחישוב התאורטי.
שימו לב: התכן ייבדק בבדיקות אוטומטיות גם עבור מקרים נוספים, מעבר לאלו שאותם אתם נדרשים להגיש. יש לוודא שהתכן אכן עומד בדרישות גם במקרים נוספים (אך אין צורך להגיש בדיקות של מקרים נוספים).



נשים לב שהערכים ב-s מתייצבים ב-1,500ns. נשנה את הערך של a[0] ב- 2,000ns באופן הבא:

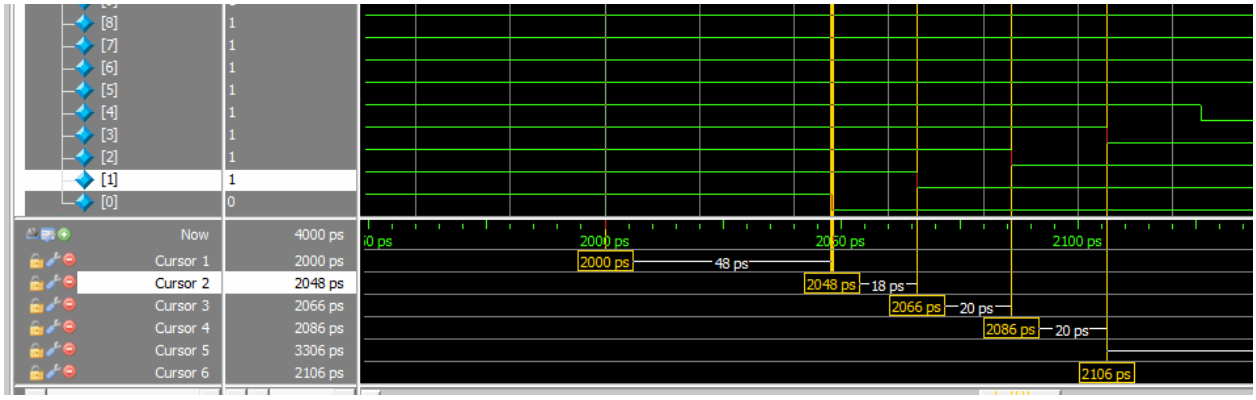
$a[0]: \ 0 \ -> \ 1$

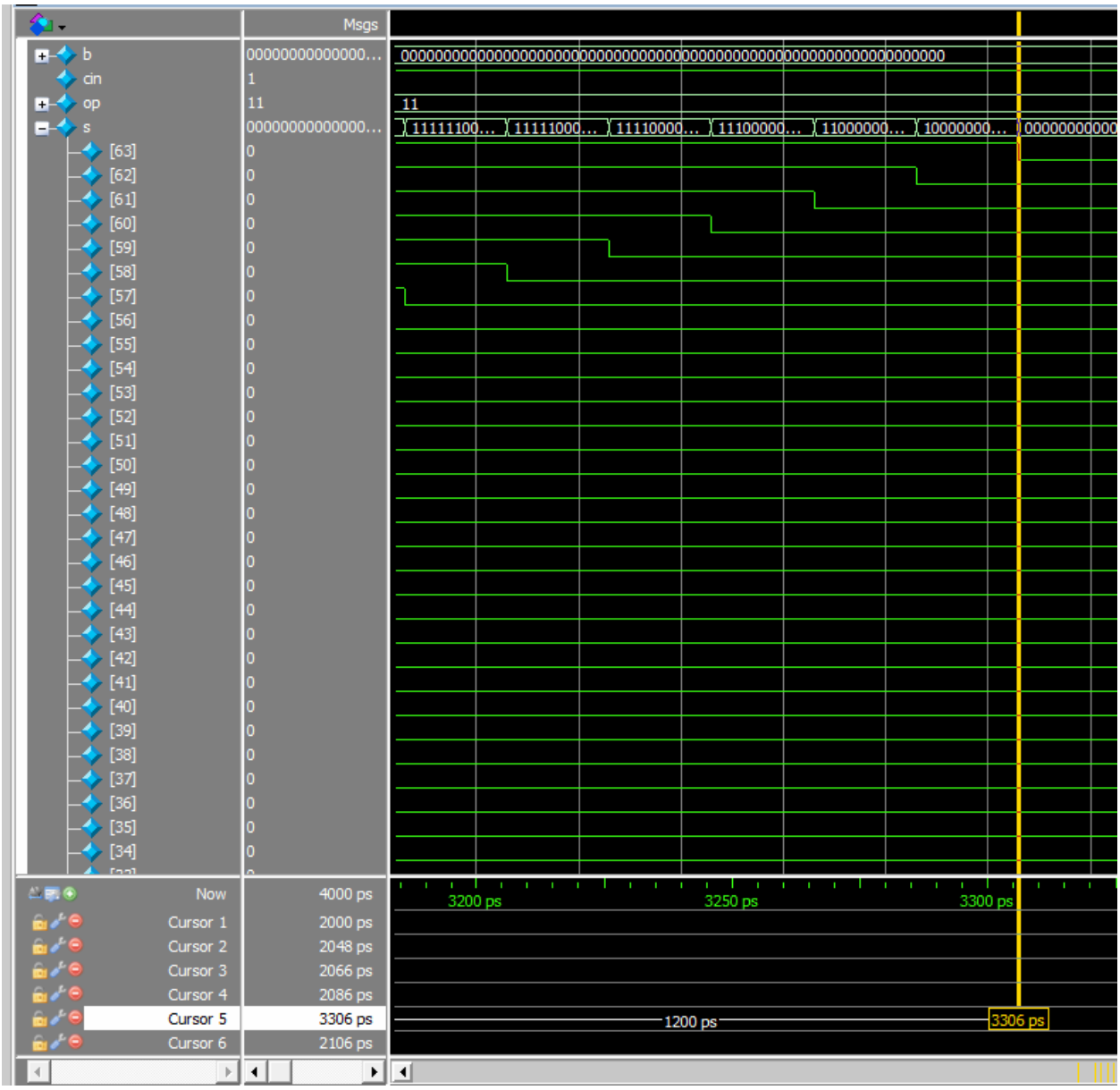
להלן השינוי של a[0] בסימולטור:



כעת נשים לב לשינוי בפעם הראשונה ב- s0.

השינוי מתרחש לאחר 48ps לאחר ששינינו את a[0], ואז אנחנו רואים שינוי ב- s[1] לאחר 18ps ולאחר מכן משך השינוי מתייצב ל-20ps עד סיום ההתייצבות שמסתיימת ב- 3306ps.




Msgs
b
cin
op
s
[63]
[62]
[61]
[60]
[59]
[58]
[57]
[56]
[55]
[54]
[53]
[52]
[51]
[50]
[49]
[48]
[47]
[46]
[45]
[44]
[43]
[42]
[41]
[40]
[39]
[38]
[37]
[36]
[35]
[34]
Now
Cursor 1
Cursor 2
Cursor 3
Cursor 4
Cursor 5
Cursor 6
4000 ps
2000 ps
2048 ps
2066 ps
2086 ps
3306 ps
2106 ps
3200 ps
3250 ps
3300 ps
1200 ps
3306 ps