הטכניון – מכון טכנולוגי לישראל

הפקולטה להנדסת חשמל ומחשבים

המעבדה ל-VLSI

דו"ח סיכום פרוייקט א'

בנושא:

Hamming ECC in SSD RAID

תיקון שגיאות ב-SSD RAID בעזרת Hamming code

מבצעים:
אלירז קדוש 315675090
אלירם עמרוסי 319040325

מנחה:
ד"ר עמית ברמן

תוכן עניינים

מבוא

עם ההתפתחות המואצת בטכנולוגיות אחסון הנתונים, **RAID** - Redundant Array of Independent הפך לכלי מרכזי לשיפור ביצועים, יתירות ואמינות במערכות אחסון גדולות. מערכי RAID מאפשרים לא רק גישה מהירה יותר למידע, אלא גם הגנה מפני אובדן נתונים במקרה של תקלות חומרה. עם זאת, למרות היתרונות הברורים של מערכים אלו, הם עדיין רגישים לטעויות הנגרמות משגיאות ביטים הנפוצות במדיה אלקטרונית כמו כונני (Solid State Drives) **SSD**.

אחת מהטכניקות המרכזיות לתיקון שגיאות היא **קוד האמינג - Hamming Code** שפותח על ידי ריצ'רד המינג באמצע המאה ה-20. קוד זה מאפשר זיהוי ותיקון שגיאות בביטים תוך שימוש בנתונים יתירים המתווספים למידע המאוחסן. יישום של קוד המינג במערכי RAID יכול להעצים את אמינות המערכת ולצמצם סיכונים לאובדן מידע קריטי.

בפרויקט זה, אנו נבחן את היישום של מערכת המשלבת **קוד תיקון שגיאות המינג במערכת RAID** המבוססת על כונני SSD. מטרת הפרויקט היא להדגים כיצד ניתן להשתמש בקוד זה לזיהוי ותיקון טעויות ברמת הביט, תוך שמירה על יעילות האחסון וביצועים מיטביים. כמו כן, ננתח את היתרונות והחסרונות של השיטה, ונבחן את השפעתה על ביצועי המערכת הכוללת.

הבנת הממשק בין טכנולוגיות RAID, כונני SSD, וקוד תיקון שגיאות כמו המינג, היא צעד קריטי בדרך לפיתוח מערכות אחסון עמידות, מהירות ואמינות יותר.

תיאור האלגוריתם

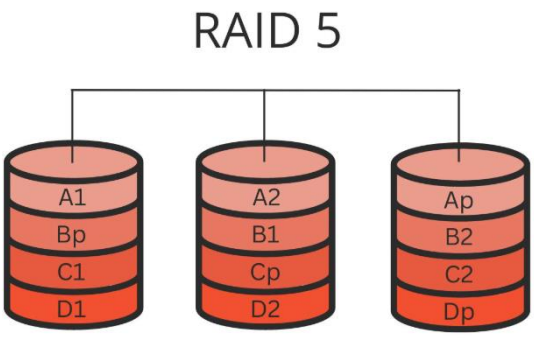
נסביר תחילה על RAID ועל קוד המינג, כל אחד בנפרד, ולאחר מכן נציג את היכולות של שילוב שני העקרונות הללו במערכת שבנינו ואת היתרונות והחסרונות שלה.

RAID

RAID (Redundant Array of Independent Disks) היא טכנולוגיה שמטרתה לשפר את האמינות והביצועים של מערכות אחסון באמצעות ארגון ואיזון נתונים בין מספר כוננים נפרדים. במקום לאחסן נתונים על כונן בודד, RAID מאפשר חלוקה ואחסון של נתונים על פני מספר כוננים, ובכך מספק יתירות ושחזור נתונים במקרה של כשל באחד הכוננים. קיימות רמות RAID שונות, כל אחת מציעה איזון אחר בין ביצועים, יתירות ועלות. בפרויקט זה, נתמקד ברמת RAID 5.

RAID 5

RAID 5 היא רמת RAID נפוצה המציעה אמינות וביצועים גבוהים יחסית על ידי שימוש בביטים של **Parity** (זוגיות) המאפשרים שחזור נתונים במקרה של כשל בכונן יחיד. מספר הכוננים המינימלי הדרוש עבור RAID 5 הוא 3 וזה גם המימוש בפרויקט שלנו.



עקרון הפעולה של RAID 5

1. **חישוב Parity** - בלוק ה-Parity מחושב באמצעות פעולת XOR (exclusive OR) על כל בלוקי הנתונים במחזור הכתיבה. לדוגמה, אם יש שני בלוקי נתונים (Data1, Data2) במחזור מסוים, בלוק ה-Parity יחושב כך: $Parity = Data1 \oplus Data2$.

2. **פיזור נתונים ו-Parity** - כל בלוק נתונים מפוזר בין N הכוננים במערך, בנוסף נכתב בלוק Parity על כונן אחד בכל מחזור כתיבה. מיקום בלוק ה-Parity משתנה בצורה מחזורית בין הכוננים כדי לפזר את עומס הכתיבה וכך לשפר Wear Leveling.

3. **שחזור נתונים במקרה של כשל** - אם כונן אחד כושל, ניתן לשחזר את הנתונים האבודים שהיו עליו באמצעות פעולת XOR של הנתונים הנותרים בכוננים התקינים יחד עם בלוק ה-Parity. מאחר ופעולת XOR היא הפיכה, ניתן לחשב את הבלוק החסר באמצעות XOR של בלוק ה-Parity ושאר בלוקי הנתונים הזמינים. לדוגמה, אם הכונן שמכיל את Data1 כשל, ניתן לשחזר אותו כך: $Data1 = Parity \oplus Data2$.

התמודדות עם שגיאות ב-RAID 5

- **Single Disk Failure - כשל דיסק בודד:** מסוגל לזהות כשל של דיסק בודד באמצעות מנגנון זיהוי כשלים של הבקר. לאחר זיהוי הכשל, ניתן לשחזר את המידע מהדיסק הכושל באמצעות חישוב ה-Parity והנתונים מהדיסקים הנותרים. תהליך השחזור מאפשר המשך גישה לנתונים (לרוב עם פגיעה קלה בביצועים) ובנייה מחדש של המערך לאחר החלפת הדיסק הפגום.

- **Multiple Disk Failure - כשל של יותר מדיסק אחד:** אם יותר מדיסק אחד כושל בו זמנית (במיוחד אם הכשלים מתרחשים באותם סטרייפים - רצועות נתונים מקבילות על פני הדיסקים), לא ניתן לשחזר את המידע האבוד באמצעות מנגנון ה-Parity הבודד של RAID 5. במקרה כזה, המידע הופך לבלתי נגיש.

- **Data Corruption - שגיאת נתונים:** אם מתרחשת שגיאה בבלוק נתונים בודד או בבלוק ה-Parity (למשל, עקב שינוי בלתי מכוון של ביט), ניתן לזהות זאת על ידי חישוב מחדש של ה-Parity והשוואתו ל-Parity הקיים. אם קיימת אי התאמה, הדבר מצביע על שגיאה. ניתן לשחזר בלוק נתונים פגום באמצעות הנתונים וה-Parity משאר הדיסקים.

סיכום סוגי השגיאות ויכולת ההתמודדות איתן:

Scenario	Detection	Recovery	Outcome
Single Disk Failure	☑ Yes	☑ Yes	Full disk recovery
Two Disk Failures	☑ Partial	✕ No	Data lost
Data Corruption (Single Block)	☑ Yes	☑ Yes	Block recovery

נראה כעת דוגמאות פשוטות לשגיאות הנ"ל וההתמודדות איתן על מערך RAID5 עם 3 דיסקים כאשר גודל כל דיסק הוא 4 בלוקים. כזכור בלוק הזוגיות נשמר בצורה ציקלית לאורך הדיסקים באופן הבא:

Stripe 1: Parity on Disk 2
Stripe 2: Parity on Disk 0
Stripe 3: Parity on Disk 1
Stripe 4: Parity on Disk 2

Initial State: Healthy Disks

Healthy State:

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	D0 = 1010	D1 = 1100	**P0 = 0110**
Stripe 2	**P1 = 1110**	D2 = 0101	D3 = 1011
Stripe 3	D4 = 1001	**P2 = 1111**	D5 = 0110
Stripe 4	D6 = 0011	D7 = 1101	**P3 = 1110**

Parity is calculated using XOR (exclusive OR).
Example Calculation:

P0 = D0 ⊕ D1 = 1010 ⊕ 1100 = **0110**

Scenario 1: Single Disk Failure (Recoverable)

<u>Disk 1 Fails</u>

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	**D0 = 1010**	**D1 = ✕**	**P0 = 0110**
Stripe 2	**P1 = 1110**	**D2 = ✕**	**D3 = 1011**
Stripe 3	**D4 = 1001**	**P2 = ✕**	**D5 = 0110**
Stripe 4	**D6 = 0011**	**D7 = ✕**	**P3 = 1110**

Detection Process:
The controller sends a Disk Failure signal on disk 1.

Recovery Process:
Stripe 1: D1 = D0 ⊕ P0 = 1010 ⊕ 0110 = 1100
Stripe 2: D2 = P1 ⊕ D3 = 1110 ⊕ 1011 = 0101
Stripe 3: P2 = P4 ⊕ D5 = 1001 ⊕ 0110 = 1111
Stripe 4: D7 = D6 ⊕ P3 = 0011 ⊕ 1110 = 1101

Recovered Data:

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	D0 = 1010	D1 = 1100	**P0 = 0110**
Stripe 2	**P1 = 1110**	D2 = 0101	D3 = 1011
Stripe 3	D4 = 1001	**P2 = 1111**	D5 = 0110
Stripe 4	D6 = 0011	D7 = 1101	**P3 = 1110**

Scenario 2: Two Disk Failures (Unrecoverable)

<u>Disk 1 and Disk 2 Fail</u>

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	D0 = 1010	D1 = ✕	**P0 = ✕**
Stripe 2	**P1 = 1110**	D2 = ✕	D3 = ✕
Stripe 3	D4 = 1001	**P2 = ✕**	D5 = ✕
Stripe 4	D6 = 0011	D7 = ✕	**P3 = ✕**

Detection Process:
The controller sends a Disk Failure signal on disk 1 and disk 2.

Recovery Process:
Recovery is not possible as two disks are missing in each stripe.

Scenario 3: Single Disk Data Corruption (Recoverable)

Disk 1, Stripe 1, Parity Block Corrupted

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	D0 = 1010	D1 = 0010 ✕	**P0 = 0110**
Stripe 2	**P1 = 1110**	D2 = 0101	D3 = 1011
Stripe 3	D4 = 1001	**P2 = 1111**	D5 = 0110
Stripe 4	D6 = 0011	D7 = 1101	**P3 = 1110**

Detection Process:

Stripe 1 Parity Check: D1 = D0 ⊕ P0 = 1010 ⊕ 0110 = 1100

Expected: 1100, Found: 0010

Recovery Process:

D1 = D0 ⊕ P0 = 1010 ⊕ 0110 = 1100

Recovered Data:

Stripe	Disk 0	Disk 1	Disk 2
Stripe 1	D0 = 1010	D1 = 1100	**P0 = 0110**
Stripe 2	**P1 = 1110**	D2 = 0101	D3 = 1011
Stripe 3	D4 = 1001	**P2 = 1111**	D5 = 0110
Stripe 4	D6 = 0011	D7 = 1101	**P3 = 1110**

קוד המינג

קוד המינג (Hamming Code) הוא קוד לגילוי ותיקון שגיאות שפותח על ידי ריצ'רד המינג בשנת 1950. הוא נמצא בשימוש נרחב במערכות תקשורת דיגיטליות, בזיכרון מחשב ובאחסון נתונים כדי להבטיח את שלמות הנתונים על ידי גילוי ותיקון שגיאות שידור. קיימים סוגים שונים של קוד המינג, ביניהם: Hamming (7,4), Hamming (12,8), SECDED, כל אחת מציעה איזון אחר בין תקורה ויעילות. בפרויקט זה, נתמקד ב-Hamming (12,8).

Hamming (12,8)

גרסה המשתמשת ב-12 ביטים בסך הכל, מתוכם 8 ביטי נתונים ו-4 ביטי זוגיות. תצורה זו מספקת יכולות משופרות של גילוי ותיקון שגיאות תוך שימוש יעיל בביטי הזוגיות. גרסה זו שימושית במיוחד כאשר משדרים או מאחסנים בלוקי נתונים גדולים יותר (8 ביטים = בייט אחד) תוך שמירה על הגנה חזקה מפני שגיאות עם 4 ביטי זוגיות. תצורה זו מסוגלת לגלות ולתקן שגיאות של ביט בודד, אך אינה מבטיחה גילוי מהימן של שגיאות של שני ביטים.

Bit Position	1	2	3	4	5	6	7	8	9	10	11	12
Encoded Bits	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity p1	✓		✓		✓		✓		✓		✓	
Parity p2		✓	✓			✓	✓			✓	✓	
Parity p4				✓	✓	✓	✓					✓
Parity p8								✓	✓	✓	✓	✓

Bits Arrangement in Hamming (12,8)

עקרון הפעולה של קוד המינג

קוד המינג פועל על ידי הוספת **ביטי זוגיות** (Parity Bits) לביטי הנתונים המקוריים. ביטי הזוגיות ממוקמים באופן אסטרטגי במיקומים התואמים לחזקות של 2 (למשל, 1, 2, 4, 8 וכו'). מטרתם היא לאפשר גם גילוי שגיאות וגם תיקון שגיאות על בסיס חישובים בינאריים.

מושגי מפתח בקוד המינג

1. **ביטי נתונים (D):** ביטים המקוריים של המידע שצריך לשדר.
2. **ביטי זוגיות (P):** ביטים נוספים שנוספים לצורך גילוי ותיקון שגיאות.
3. **מילת קוד (Codeword):** השילוב של ביטי הנתונים וביטי הזוגיות. במקרה של Hamming (12,8), מילת הקוד מכילה 12 ביטים.

תאוריית קוד המינג

1. **מיקום ביטי זוגיות:** ביטי הזוגיות ממוקמים במיקומים התואמים לחזקות של 2 (1, 2, 4, 8 וכו') בתוך מילת הקוד.
2. **חישוב ביטי זוגיות:** כל ביט זוגיות מבטיח שמספר הביטים בעלי הערך '1' בתת-קבוצה ספציפית של ביטים (כולל ביט הזוגיות עצמו) עוקב אחר כלל זוגיות מוגדר (זוגי או אי-זוגי).
3. **חישוב סינדרום:** הסינדרום (Syndrome), S, הוא מספר בינארי המשמש לזיהוי שגיאות במילת הקוד. כל ביט זוגיות בודק תת-קבוצה ספציפית של ביטים. אם כל בדיקות

הזוגיות עוברות, הסינדרום יהיה 0, מה שמצביע על היעדר שגיאות. אם בדיקת זוגיות כלשהי נכשלת, הסינדרום ייתן מספר בינארי המצביע על מיקום הביט השגוי. הנוסחה לחישוב הסינדרום במקרה של 4 ביטי זוגיות (Hamming 12,8) היא כדלהלן:

$$S = P_1 \cdot 2^0 + P_2 \cdot 2^1 + P_4 \cdot 2^2 + P_8 \cdot 2^3$$

כאשר:
P1: בודק תקינות של הביטים: 1, 3, 5, 7, 9, 11
P2: בודק תקינות של הביטים: 2, 3, 6, 7, 10, 11
P4: בודק תקינות של הביטים: 4, 5, 6, 7, 12
P8: בודק תקינות של הביטים: 8, 9, 10, 11, 12

4. **פירוש הסינדרום:**
   - **S = 0:** לא זוהתה שגיאה.
   - **S ≠ 0:** זוהתה שגיאה, והערך הבינארי של S מצביע על מיקום הביט השגוי. לדוגמה, אם הסינדרום הוא 5 (בינארי 101), השגיאה היא בביט מספר 5.
5. **תיקון שגיאה:** לאחר זיהוי מיקום השגיאה באמצעות הסינדרום, הביט במיקום זה משתנה (0 ← 1 או 1 ← 0) כדי לתקן את השגיאה.

משמעות מעשית של ערכי הסינדרום: במערכת Hamming (12,8), אורך מילת הקוד הוא 12 ביטים. ערכי סינדרום תקפים המצביעים על מיקום שגיאה הם בטווח 1–12. ערך סינדרום 0 מצביע על היעדר שגיאות. ערכי סינדרום 13–15 אינם יכולים להתרחש באופן טבעי במערכת Hamming (12,8) תקינה, ואם הם מופיעים, הם מרמזים על שגיאה בחישוב הסינדרום או על תקלה בלוגיקה של המערכת.

השוואה בין גילוי ותיקון שגיאה בביט נתונים (D) לביט זוגיות (P): אין הבדל מהותי בין גילוי ותיקון שגיאה בביט נתונים לבין שגיאה בביט זוגיות בקוד המינג. מנגנון חישוב הסינדרום פועל באותו אופן ללא קשר לסוג הביט בו התרחשה השגיאה. לאחר שהסינדרום מזהה את מיקום השגיאה, הביט במיקום זה פשוט משתנה, ללא התייחסות אם זהו ביט נתונים או ביט זוגיות.

סיכום סוגי השגיאות ויכולת ההתמודדות איתן:

Scenario	Error Type	Detection	Recovery	Action
No Errors	None	✅ Yes	✅ N/A	Use as-is
Single-Bit Error	Correctable	✅ Yes	✅ Yes	Correct bit
Double-Bit Error	Detectable Only	✅ Yes	❌ No	Unrecoverable

נראה כעת דוגמה מעשית לחישוב קוד המינג:

<u>Step 1:</u> Original 8-Bit Data

We will use the **8-bit data:** `1011 0011`

<u>Step 2:</u> Bit Arrangement in Hamming ECC (8→12)

Bit Position	1	2	3	4	5	6	7	8	9	10	11	12
Bit Type	P1	P2	D1	P4	D2	D3	D4	P8	D5	D6	D7	D8
Value	?	?	1	?	0	1	1	?	0	0	1	1

P1, P2, P4, P8: Parity bits (unknown at this stage).
D1–D8: Data bits.

<u>Step 3:</u> Parity Bit Calculations

P1: Checks bits **3, 5, 7, 9, 11**
Formula: $P1=D1\oplus D2\oplus D4\oplus D5\oplus D7$
Calculation: $1\oplus 0\oplus 1\oplus 0\oplus 1=1$

P2: Checks bits **3, 6, 7, 10, 11**
Formula: $P2=D1\oplus D3\oplus D4\oplus D6\oplus D7$
Calculation: $1\oplus 1\oplus 1\oplus 0\oplus 1=0$

P4: Checks bits **5, 6, 7, 12**
Formula: $P4=D2\oplus D3\oplus D4\oplus D8$
Calculation: $0\oplus 1\oplus 1\oplus 1=1$

P8: Checks bits **9, 10, 11, 12**
Formula: $P8=D5\oplus D6\oplus D7\oplus D8$
Calculation: $0\oplus 0\oplus 1\oplus 1=0$

Parity Bits: P1 = 1; P2 = 0; P4 = 1; P8 = 0

<u>Step 4:</u> Complete Table After Encoding

Bit Position	1	2	3	4	5	6	7	8	9	10	11	12
Bit Type	P1	P2	D1	P4	D2	D3	D4	P8	D5	D6	D7	D8
Value	1	0	1	1	0	1	1	0	0	0	1	1

Encoded Codeword: `1011 0110 0011`

נמשיך עם הדוגמה לקידוד הנ"ל ונראה כעת דוגמאות מעשיות להתמודדות עם זיהוי ותיקון שגיאות:

<u>Scenario 1: No Errors</u>

1. Original Encoding: `1011 0110 0011`

2. Received Codeword: `1011 0110 0011`

3. Parity Bit Calculations:

1. **P1 (1,3,5,7,9,11):**
   Formula: P1⊕D1⊕D2⊕D4⊕D5⊕D7
   Calculation: 1⊕1⊕0⊕1⊕0⊕1=0 ✅

2. **P2 (2,3,6,7,10,11):**
   Formula: P2⊕D1⊕D3⊕D4⊕D6⊕D7
   Calculation: 0⊕1⊕1⊕1⊕0⊕1=0 ✅

3. **P4 (4,5,6,7,12):**
   Formula: P4⊕D2⊕D3⊕D4⊕D8
   Calculation: 1⊕0⊕1⊕1⊕1=0 ✅

4. **P8 (8,9,10,11,12):**
   Formula: P8⊕D5⊕D6⊕D7⊕D8
   Calculation: 0⊕0⊕0⊕1⊕1=0 ✅

✅ **All parities match.**

4. Syndrome Calculation:

$$S = P_1 \cdot 2^0 + P_2 \cdot 2^1 + P_4 \cdot 2^2 + P_8 \cdot 2^3 = 0 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 = 0$$

✅ **Syndrome = 0 (No Error Detected).**

5. Syndrome Decoding: S = 0: No error detected.

6. Decoded Data: D1, D2, D3, D4, D5, D6, D7, D8: `1011 0011`

✅ **Result:** Data is valid.

Scenario 2: Single-Bit Error (Correctable)

1. Original Encoding: `1011 0110 0011`

2. Received Codeword (error at position 6): `1011 0010 0011`

3. Parity Bit Calculations:

1. **P1 (1,3,5,7,9,11):**
   Formula: P1⊕D1⊕D2⊕D4⊕D5⊕D7
   Calculation: 1⊕1⊕0⊕1⊕0⊕1=0 ✅

2. **P2 (2,3,6,7,10,11):**
   Formula: P2⊕D1⊕D3⊕D4⊕D6⊕D7
   Calculation: 0⊕1⊕0⊕1⊕0⊕1=1 ❌

3. **P4 (4,5,6,7,12):**
   Formula: P4⊕D2⊕D3⊕D4⊕D8
   Calculation: 1⊕0⊕0⊕1⊕1=1 ❌

4. **P8 (8,9,10,11,12):**
   Formula: P8⊕D5⊕D6⊕D7⊕D8
   Calculation: 0⊕0⊕0⊕1⊕1=0 ✅

✅ **Parity mismatches detected in P2 and P4.**

4. Syndrome Calculation:

$$S = P_1 \cdot 2^0 + P_2 \cdot 2^1 + P_4 \cdot 2^2 + P_8 \cdot 2^3 = 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 = 6$$

✅ **Syndrome = 6 (error detected at position 6).**

5. Syndrome Decoding: S = 6: Error detected at position 6 (D3). Flip the bit at Position 6: 0 → 1

6. Corrected Codeword: `1011 0110 0011`

✅ **Error corrected at position 6 (D3).**

7. Decoded Data: D1, D2, D3, D4, D5, D6, D7, D8: `1011 0011`

✅ **Result:** Data is valid.

Scenario 3: Double-Bit Error (Not Correctable)

1. Original Encoding: `1011 0110 0011`

2. Received Codeword (errors at positions 5 and 6): `1011 1010 0011`

3. Parity Bit Calculations:

1. **P1 (1,3,5,7,9,11):**
   Formula: P1⊕D1⊕D2⊕D4⊕D5⊕D7
   Calculation: 1⊕1⊕1⊕1⊕0⊕1=1 ✕

2. **P2 (2,3,6,7,10,11):**
   Formula: P2⊕D1⊕D3⊕D4⊕D6⊕D7
   Calculation: 0⊕1⊕0⊕1⊕0⊕1=1 ✕

3. **P4 (4,5,6,7,12):**
   Formula: P4⊕D2⊕D3⊕D4⊕D8
   Calculation: 1⊕1⊕0⊕1⊕1=0 ✅

4. **P8 (8,9,10,11,12):**
   Formula: P8⊕D5⊕D6⊕D7⊕D8
   Calculation: 0⊕0⊕0⊕1⊕1=0 ✅

✅ **Parity mismatches detected in P1 and P2.**

4. Syndrome Calculation:

$$S = P_1 \cdot 2^0 + P_2 \cdot 2^1 + P_4 \cdot 2^2 + P_8 \cdot 2^3 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 = 3$$

✅ **Syndrome = 3 (error detected at position 3).**

However, double-bit errors produce conflicting results because Hamming ECC can only correct single-bit errors.

5. Syndrome Decoding:

- **Syndrome result is unreliable:** Multiple errors detected.
- **Hamming ECC cannot reliably correct double-bit errors.**

6. Decoded Data:

- **Data extraction is unreliable.**

✕ **Result:** Data is corrupted and cannot be decoded.

מערכת המשלבת RAID 5 וקוד המינג

כאמור, בפרויקט זה שילבנו את שני הטכניקות, RAID 5 וקוד המינג, במערכת אחת.
המערכת הספציפית שלנו מכילה 3 דיסקים כאשר כל בלוק מכיל 12 ביטים.
נסקור כעת את היתרונות והחסרונות של המערכת.

	RAID 5 בלבד	Hamming (8,12) בלבד	RAID 5 + Hamming (8,12)	ללא RAID 5 וללא קוד המינג
תקורה	33%	33%	55.55%	0%
נצילות	66%	66%	44.44%	100%
יכולת שחזור	שחזור כשל דיסק יחיד	תיקון שגיאות של סיבית בודדת	שחזור כשל דיסק יחיד + תיקון שגיאות סיביות	אין יכולת שחזור
Wear Leveling	טובה	טובה	טובה מאוד	תלוי במערכת

תקורה:

- **RAID 5** - בלוק אחד מתוך השלושה מקוצה לטובת זוגיות - תקורה של 33%.
- **Hamming (8,12)** - 4 ביטים מתוך 12 בבלוק מוקצים לטובת זוגיות - תקורה של 33%.
- **RAID 5 + Hamming (8,12)** - בלוק אחד מתוך השלושה מקוצה לטובת זוגיות, בנוסף, בשני הבלוקים האחרים 4 ביטים מתוך 12 בבלוק מוקצים לטובת זוגיות – תקורה של 55.55%.
- **מערכת ללא RAID 5 וללא קוד המינג** - **אינה מוסיפה תקורה** כלל.

נצילות:

- **RAID 5** – שני בלוקים מתוך השלושה זמינים לטובת נתונים - נצילות של 66%.
- **Hamming (8,12)** - 8 ביטים מתוך 12 בבלוק זמינים לטובת נתונים - נצילות של 66%.
- **RAID 5 + Hamming (8,12)** – שני בלוקים מתוך השלושה זמינים לטובת נתונים, אבל בשני הבלוקים האחרים הללו רק 8 ביטים מתוך 12 זמינים לטובת נתונים – נצילות של 44.44%.
- **מערכת ללא RAID 5 וללא קוד המינג** - **אינה מוסיפה תקורה** כלל.

יכולת זיהוי שחזור:

- **RAID 5** – זיהוי ושחזור כשל של דיסק יחיד ובפרט של בלוק נתונים יחיד.
- **Hamming (8,12)** - זיהוי ושחזור שגיאה בביט אחד. זיהוי חלקי של שגיאה בשתי ביטים ללא יכולת שחזור.
- **RAID 5 + Hamming (8,12)** - זיהוי ושחזור כשל של דיסק יחיד ובפרט של בלוק נתונים יחיד. זיהוי ושחזור שגיאה בביט אחד. זיהוי חלקי של שגיאה בשתי ביטים ללא יכולת שחזור.
- **מערכת ללא RAID 5 וללא קוד המינג** - **אינה מספקת יכולת שחזור כלל.**

Wear Leveling:

- **RAID 5** - מפזר את הכתיבה בין הדיסקים ולכן מאזנת את הבלאי.
- **Hamming (8,12)** - לא משנה את רמת הבלאי במערכת.
- **RAID 5 + Hamming (8,12)** - **בנוסף למנגנון פיזור הנתונים הקיים ב-RAID 5, במערכת שלנו הוספנו מנגון זיהוי החוסך כתיבה מיותרת במקרה שאחד או שני הבלוקים המיועדים לכתיבה זהה לבלוק/ים שכבר שמור/ים בדיסק.**
- **מערכת ללא RAID 5 וללא קוד המינג** - תלויה במנגנון האחסון ובאיזון הכתיבות.

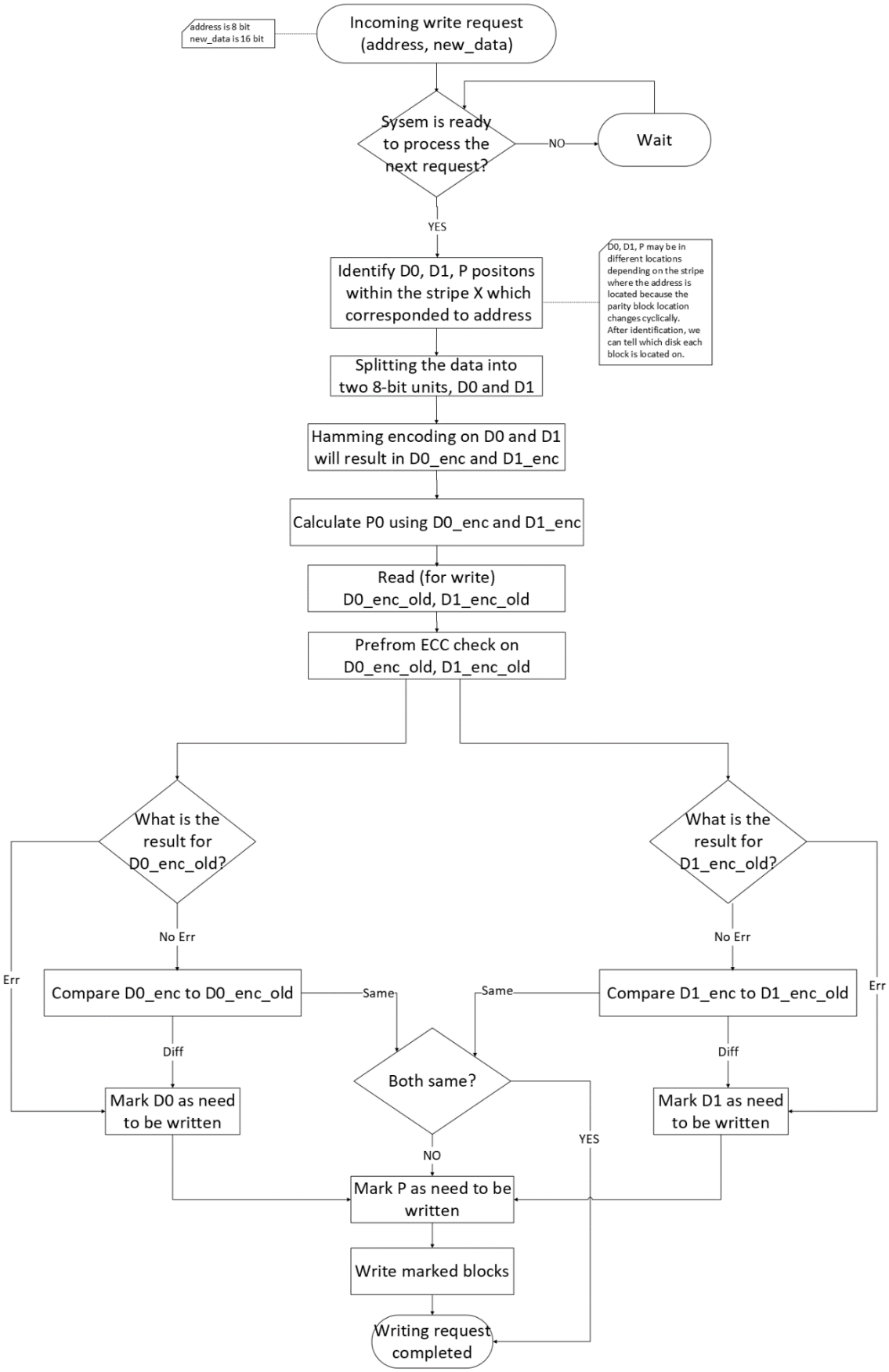
מסקנות:

- אם רוצים **הגנה מפני כשל דיסק**, RAID 5 הוא הפתרון המתאים.
- אם רוצים **תיקון שגיאות סיביות**, קוד המינג (8,12) הוא הפתרון המתאים.
- אם רוצים **הגנה מקסימלית**, עדיף **RAID 5 + Hamming (8,12)**, אך זה בא על חשבון תקורה גבוהה ונצילות נמוכה.

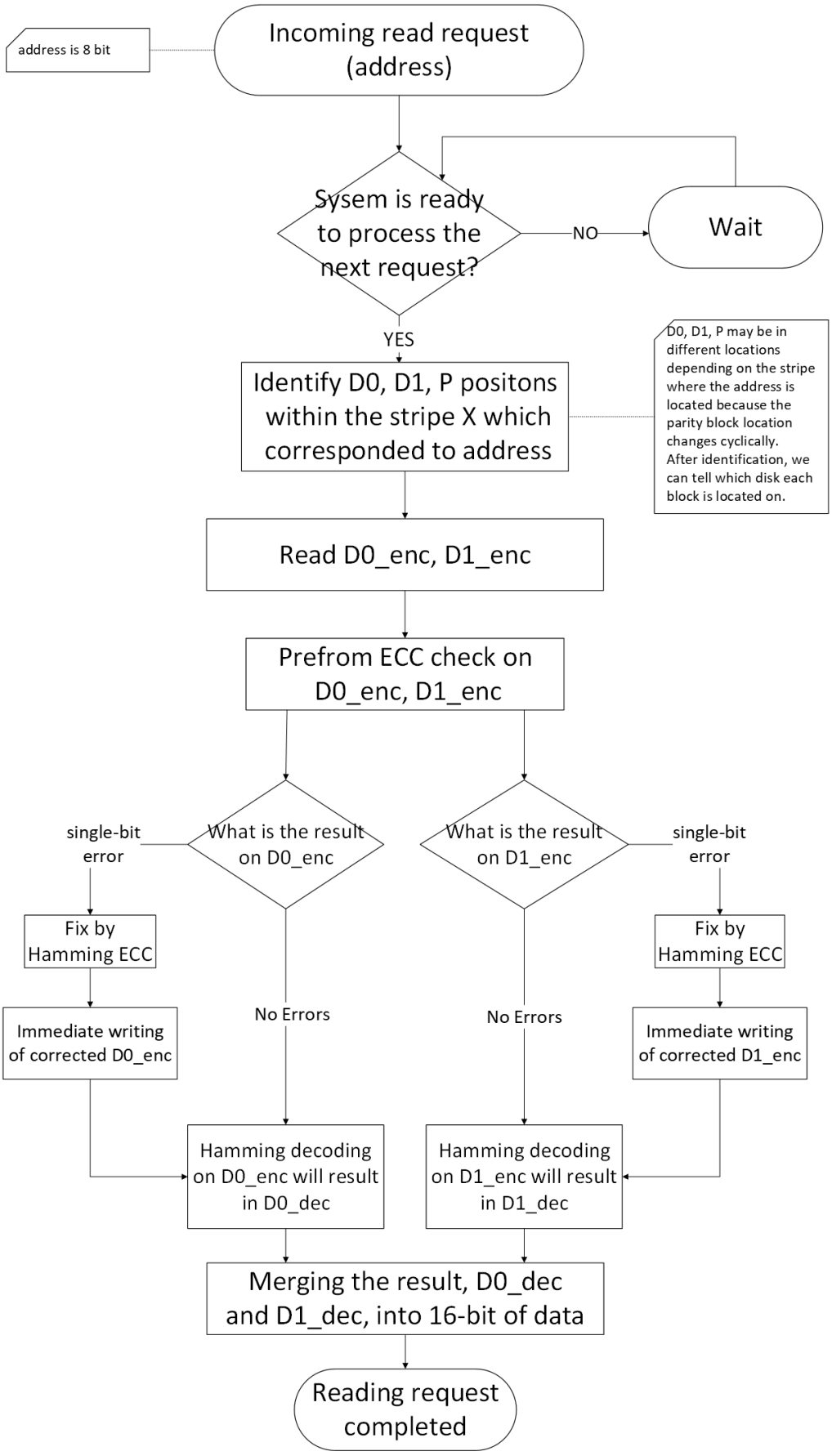
סיכום

מערכת אחסון המשלבת את יתרונות היתירות והשחזור של RAID 5 עם יכולות גילוי ותיקון השגיאות ברמת הביט של קוד המינג מציעה פתרון אמין, עמיד וגמיש במיוחד לשמירה על נתונים במערכות אחסון מודרניות. בהשוואה למערכת אחסון רגילה, המערכת המשולבת מספקת שכבות הגנה מרובות מפני כשלים מסוגים שונים, משפרת את שלמות הנתונים ומבטיחה המשכיות גם במצבי תקלה. השילוב בין שתי הטכנולוגיות החזקות הללו יוצר מערכת אחסון חזקה ויעילה יותר, המסוגלת להתמודד עם האתגרים של אחסון נתונים.

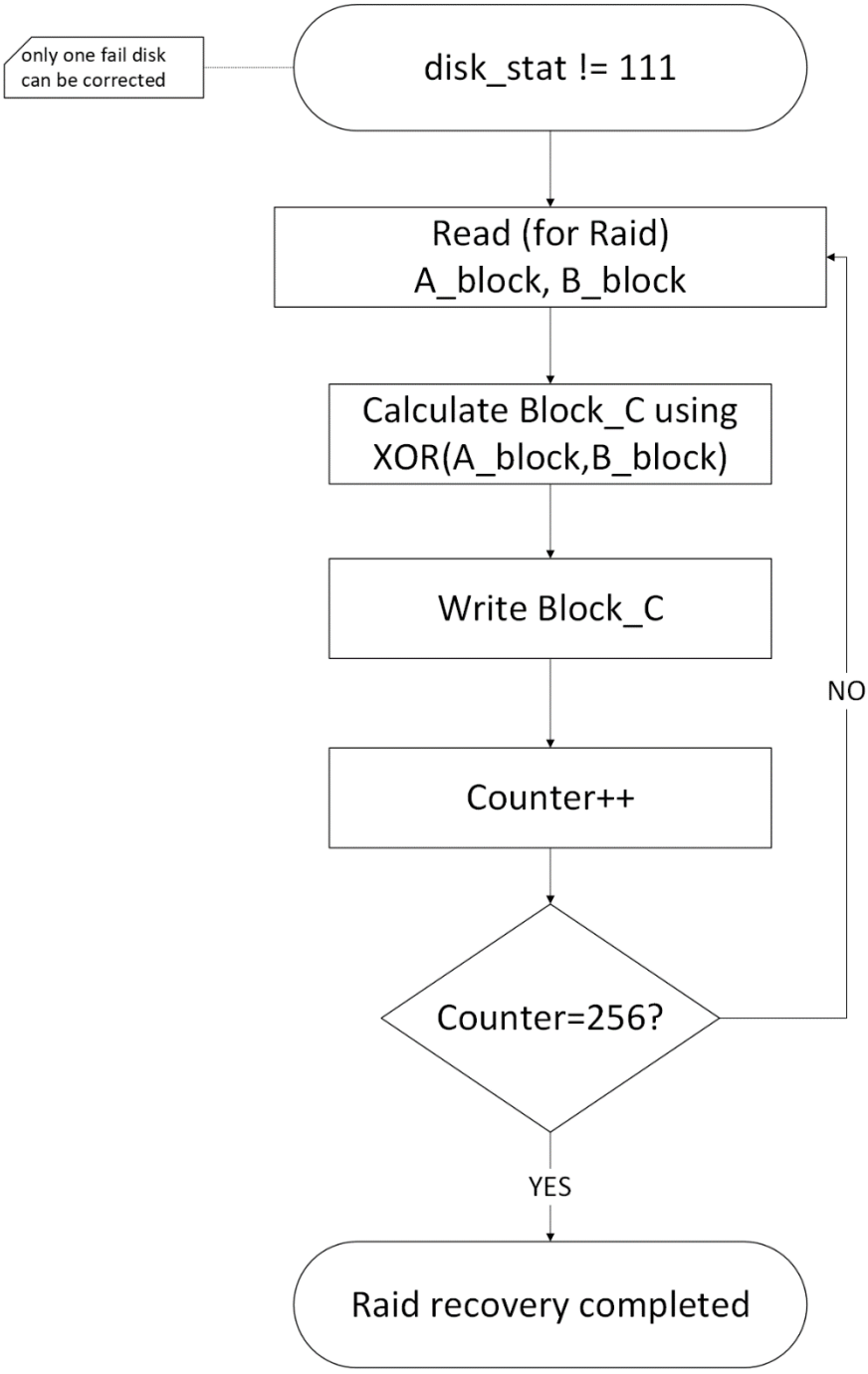
תרשים זרימה Write Request:



תרשים זרימה Read Request:



תרשים זרימה RAID Recovery:

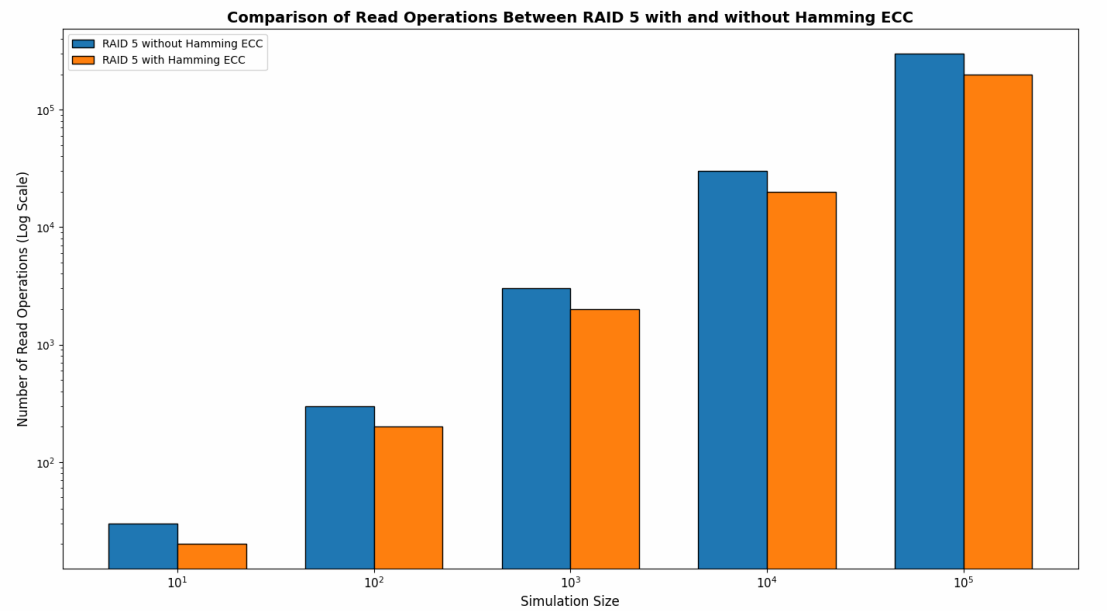


מימוש בתוכנה

עשינו שימוש ב-Python בכדי לדמות את פעולת ה-chip שתכננו.

השתמשנו בגדלים שונים של סימולציות כדי לבדוק את היעילות כתלות במספר בקשות הקריאה/הכתיבה לזיכרון.

בגרף הבא אנו רואים את כמות בקשות הקריאה מהזיכרון במערכת שלנו, RAID 5 עם Hamming(8,12), לעומת מערכת RAID 5 ללא Hamming(8,12).



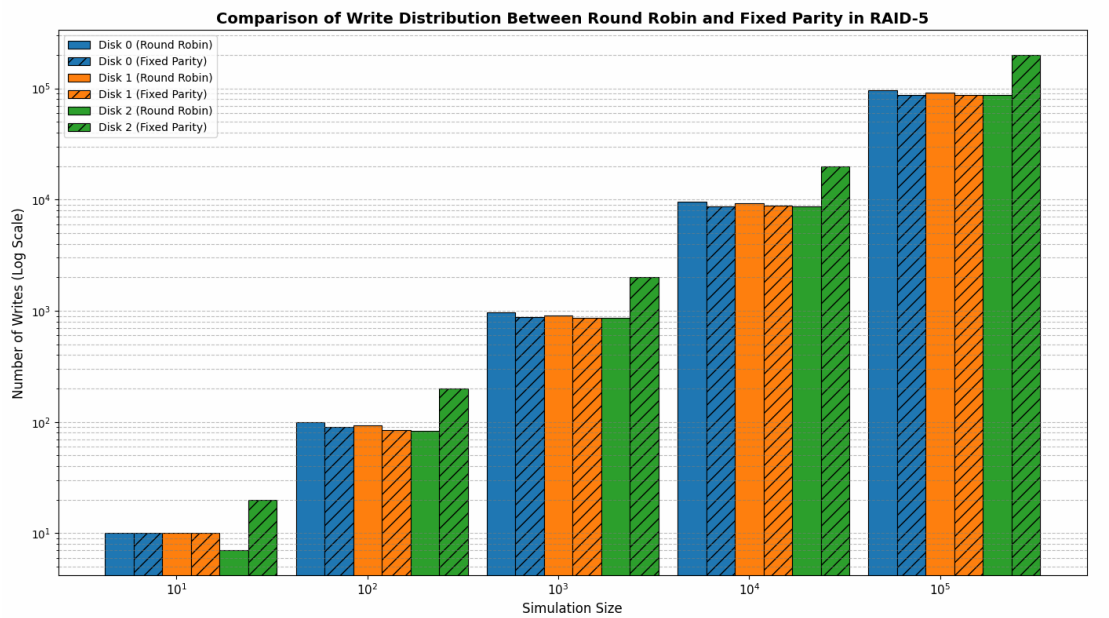
נסביר את תוצאות הגרף:

	מספר קריאות נדרשות כאשר המידע תקין	מספר קריאות במקרה של SBE	מספר קריאות במקרה של שגיאה ביותר מביט אחד
RAID 5 ללא Hamming	**3**	**3**	**3**
RAID 5 עם Hamming	**2** (D0, D1)	**2** (זיהוי ותיקון השגיאה מתבצע בבלוק עצמו)	**3** (שחזור RAID 5 רגיל)

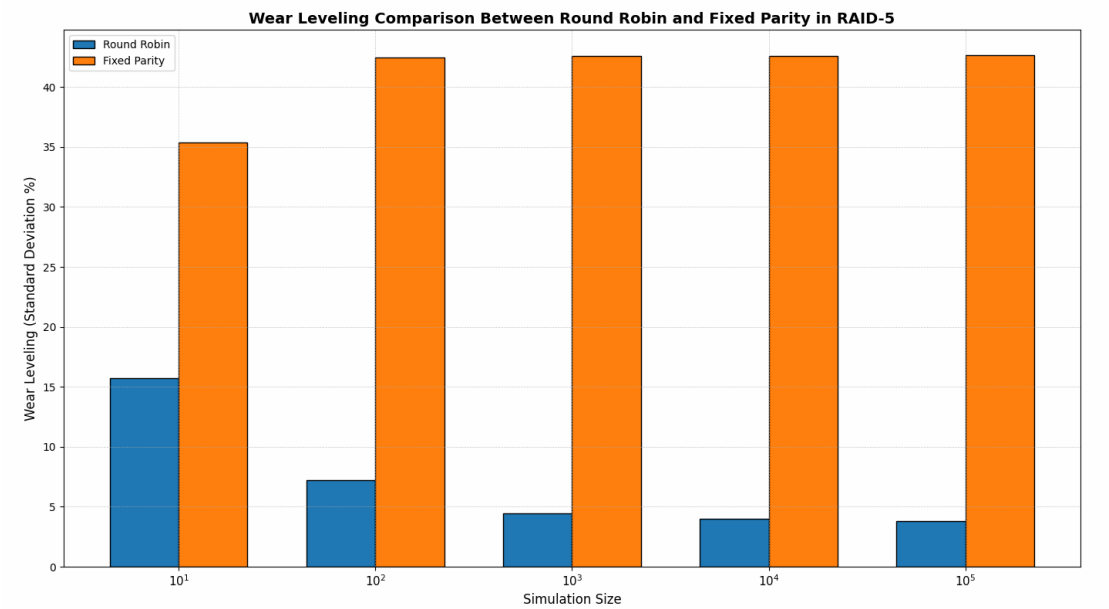
הסבר מפורט:

- **RAID 5 ללא Hamming ECC** חייב לקרוא **את כל שלושת הבלוקים** בכל קריאה כדי לוודא את תקינות הנתונים. אם יש תקלה, הוא משווה את ה-parity שנשמר בעבר לה-parity המחושב כדי לאתר ולתקן שגיאות.
- **RAID 5 עם Hamming ECC** מקודד את שני בלוקי המידע בנפרד, כך שהוא יכול **לגלות ולתקן שגיאות ביט בודדות (SBE) ללא קריאה ל-parity.** רק במקרה של **שגיאה חמורה** כמו **שני ביטים פגומים (DBE) או כשל דיסק**, יש צורך לקרוא גם את ה-parity לשחזור הנתונים.
- **היתרון של RAID 5 עם Hamming ECC** הוא **הפחתת כמות הקריאות** הדרושות ברוב המקרים, מה שמשפר ביצועים ומפחית עומס על מערכת האחסון.

הגרף שלפנינו מציג את כמות הכתיבות לכל דיסק. בסימולציות הכתיבה הנחנו כי 75% מהכתיבות הן מידע חדש ו-25% מהכתיבות הן כתיבות חוזרות בבלוק D0 או בבלוק D1 באופן רנדומלי. כזכור – בפרויקט שלנו אנו כותבים כתיבות חיוניות (חדשות) בלבד כדי לשפר Wear Leveling ולכן כאשר יש שינוי רק ב-D0 או רק ב-D1, אנחנו נכתוב רק אותו ואת ה-parity החדש וכך חסכנו כתיבה לאחד הדיסקים. ניתן לראות כי במערכת RAID 5, שבה בלוק ה-parity נכתב בשיטת Round Robin בין הדיסקים, חלוקת הכתיבה מאוזנת ביניהם. לעומת זאת, כאשר בלוק ה-parity נשמר על דיסק קבוע (Fixed Parity), אותו דיסק חווה שחיקה משמעותית גבוהה יותר בהשוואה לשאר הדיסקים.



באופן דומה בגרף הבא ניתן לראות את ההשפעה של מנגנון ה-parity על איזון השחיקה (**Wear Leveling**) במערכת כולה. כפי שניתן להבחין, כאשר בלוק ה-parity נכתב בשיטת Round Robin בין הדיסקים, השחיקה מתאזנת בצורה משמעותית יותר, בעוד שבמקרה של Fixed Parity, השחיקה גבוהה בהרבה ומתרכזת בדיסק ספציפי. השפעה זו נעשית בולטת יותר ככל שגודל הסימולציה גדל.
<u>בסימולציה שגודלה 100,000 כתיבות, ניתן לראות שיפור של כ-40% ב-Wear Leveling.</u>



מיקרו-ארכיטקטורה

Top Level

הרכיב בנוי ממספר חלקים עיקריים: רכיבים הקשורים למנגנון RAID 5, רכיבים האחראים על קידוד המינג, ורכיבים המטפלים בקריאה ובכתיבה מהזיכרון. בנוסף, בקר מרכזי מנהל את פעולתה הכוללת של המערכת.

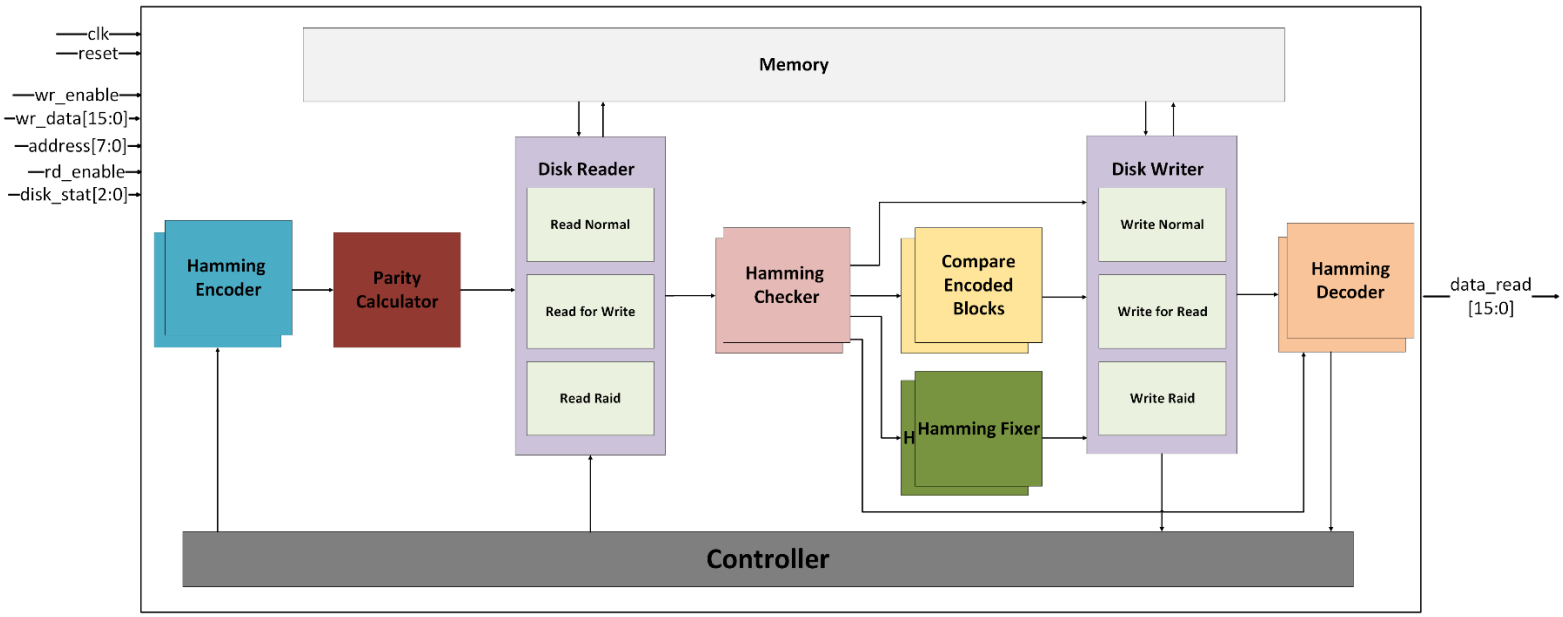
במהלך התכנון שמנו דגש על מהירות ולכן מיקבלנו רכיבים ופעולות מתמטיות בבלוקים היכן שניתן.

כניסות הרכיב:

- clk – שעון מערכת.
- reset – איתחול מערכת.
- wr_enable – אות להפעלת כתיבה.
- wr_data [15:0] – נתונים לכתיבה.
- address [7:0] – כתובת הנתונים לקריאה או כתיבה (בהתאם למצב האותות wr_enable או rd_enable).
- rd_enable – אות להפעלת קריאה.
- disk_stat [2:0] – מציין אילו דיסקים תקינים (משמש לדימוי כשל דיסק).

יציאות הרכיב:

- data_read [15:0] – נתונים שנקראו.



Top Level micro-architecture

תיאור תתי הרכיבים

Hamming Encoder

תיאור הרכיב:

רכיב זה אחראי על קידוד 8 ביטים של נתונים לפורמט **Hamming code** המורכב מ-12 ביטים. הקידוד מאפשר זיהוי ותיקון של שגיאת ביט בודד (**Single Bit Error**).

Inputs:

- clk: System clock.
- reset: System reset.
- enable: Activate signal.
- data_in [7:0]: Data input to be encoded.
- address_in [7:0]: Address.

Outputs:

- enc_data [11:0]: Encoded 12-bit data.
- address_out [7:0]: Address.
- enc_data_valid: Indicates valid encoded data.

פעולת הרכיב:

מקבל 8 ביטים של נתונים כקלט, מחשב 4 ביטי פריטי (P1, P2, P3, P4) באמצעות לוגיקת XOR על פי מיקומים מוגדרים מראש, משלב את ביטי הפריטי עם הנתונים המקוריים ליצירת מילה מקודדת באורך 12 ביטים ושומר את התוצאה ברגיסטרים ליציאה במחזור שעון יחיד.

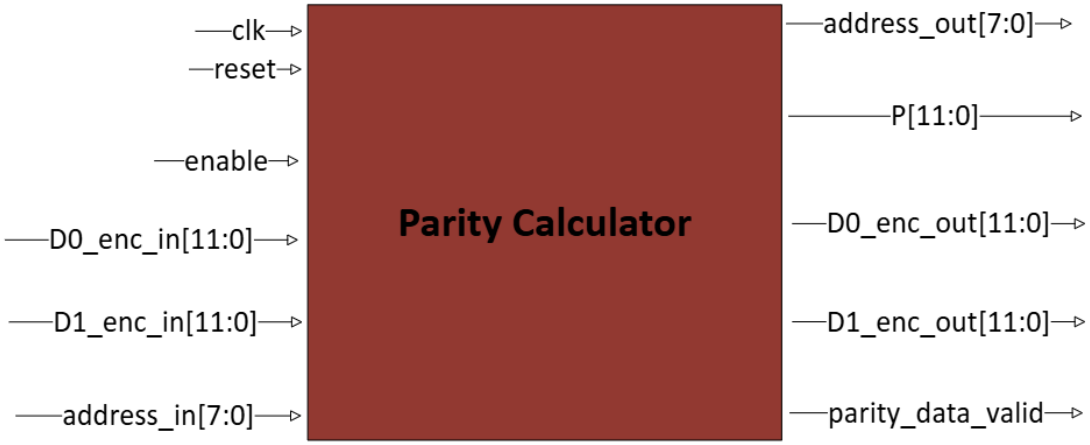
Bit Position	0	1	2	3	4	5	6	7	8	9	10	11
Encoded Bits	p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8
Parity p1	✓		✓		✓		✓		✓		✓	
Parity p2		✓	✓			✓	✓			✓	✓	
Parity p4				✓	✓	✓	✓					✓
Parity p8								✓	✓	✓	✓	✓

Bits Arrangement in Hamming (8, 12)

Parity Calculator

תיאור הרכיב:

הרכיב מחשב את בלוק הפריטי (**P**) באמצעות פעולת XOR בין שני בלוקים של נתונים מקודדים. בלוק הפריטי מאפשר תיקון של כשל דיסק לפי מנגון RAID 5.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Activate signal.
- D0_enc_in [11:0]: First encoded data block.
- D1_enc_in [11:0]: Second encoded data block.
- address_in [7:0]: Address.

Outputs:

- address_out [7:0]: Address.
- P [11:0]: First encoded data block.
- D0_enc_out [11:0]: First encoded data block.
- D1_enc_out [11:0]: Second encoded data block.
- parity_data_valid: Indicates parity is valid.

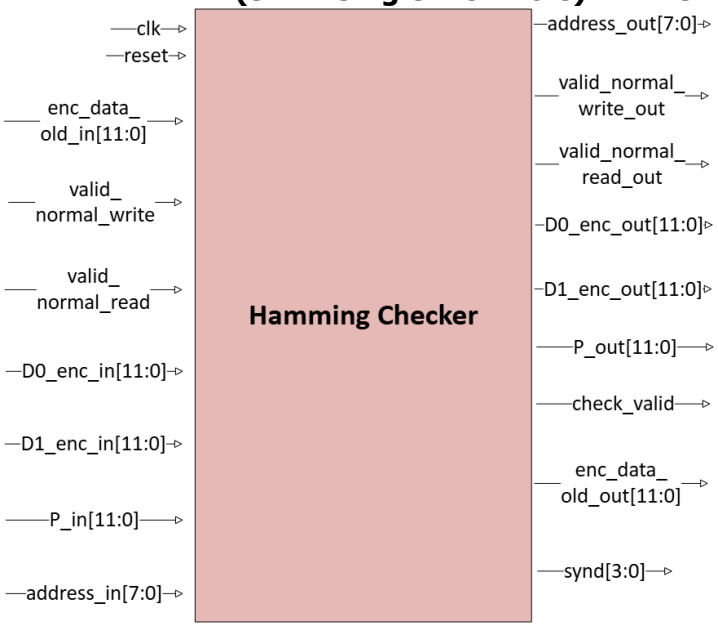
פעולת הרכיב:

מקבל שני בלוקים מקודדים ומבצע פעולת XOR לכל זוג ביטים תואם בין שני בלוקים ומייצר בלוק פריטי. שומר את התוצאה ברגיסטרים, ומוציא את בלוק הפריטי במחזור שעון יחיד.

Hamming Checker

תיאור הרכיב:

רכיב זה מיועד לבדוק נתונים מקודדים בפורמט **Hamming Code** באורך 12 ביטים לזיהוי **שגיאת ביט בודד (SBE - Single Bit Errors)**.



Inputs:

- clk: System clock.
- reset: System reset.
- enc_data_old_in [11:0]: Encoded data to check.
- valid_normal_write: Enable signal from read_for_write.
- valid_normal_read: Enable signal from read_normal.
- D0_enc_in [11:0]: Encoded 12-bit data.
- D1_enc_in [11:0]: Encoded 12-bit data.
- P_in [11:0]: Parity.
- address_in [7:0]: Address.

Outputs:

- address_out [7:0]: Address.
- valid_normal_write_out: Indicates how to progress, depends on read/write.
- valid_normal_read_out: Indicates how to progress, depends on read/write.
- D0_enc_out [11:0]: Encoded 12-bit data.
- D1_enc_out [11:0]: Encoded 12-bit data.
- P_out [11:0]: parity.
- check_valid: Indicates valid check.
- enc_data_old_out [11:0]: Encoded data to check.
- synd [3:0]: Syndrome pattern for error location.

פעולת הרכיב:

מקבל נתונים מקודדים בפורמט Hamming Code באורך 12 ביטים, מחשב באמצעות לוגיקת XOR את תבנית ה**סינדרום** המורכבת מ-4 ביטים, אשר מזהה ומסמנת את מיקום שגיאת הביט (אם קיימת), ושומר את תוצאת הסינדרום ברגיסטרים לשימוש במחזור הבא.

הסינדרום מחושב כך: $S = S_1 \cdot 2^0 + P_2 \cdot 2^1 + P_4 \cdot 2^2 + P_8 \cdot 2^3$

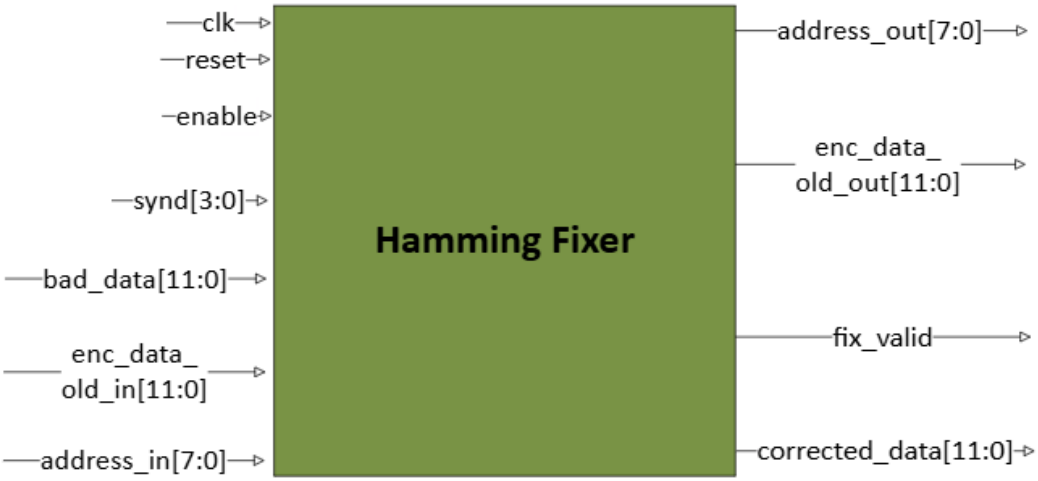
כאשר S=0: אין שגיאה.

כאשר S!=0: יש שגיאה בביט שבמיקום S.

Hamming Fixer

תיאור הרכיב:

רכיב זה מיועד לתקן **שגיאת ביט בודד (SBE - Single Bit Errors)** בהתבסס על תבנית ה**syndrome**.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Activate signal.
- synd [3:0]: Syndrome pattern for error location.
- bad_data [11:0]: Encoded data with an error.
- enc_data_old_in [11:0]: Encoded data to check.
- address_in [7:0]: Address.

Outputs:

- address_out [7:0]: Address.
- enc_data_old_out [11:0]: Encoded data to check.
- fix_valid: Indicates valid check.
- corrected_data [11:0]: Corrected 12-bit encoded data.

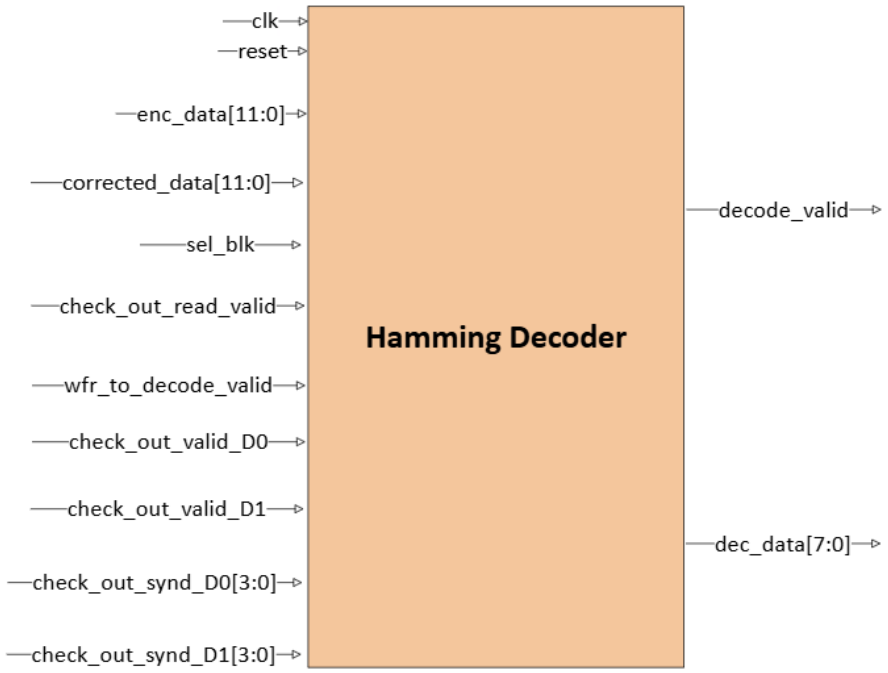
פעולת הרכיב:

הרכיב מקבל תבנית סינדרום, משתמש ב- Decoder 4-to-12 כדי לפענח את מיקום השגיאה, מעביר את כל 12 הביטים של הנתונים דרך שערי XOR לתיקון הביט השגוי (Flip), שומר את המידע המתוקן ברגיסטרים, ומוציא אותו כפלט שלם ומתוקן.

Hamming Decoder

תיאור הרכיב:

הרכיב מקבל דאטה **מקודד** לפי **קידוד המינג** באורך 12 ביטים ומוציא דאטה **מפוענח** באורך 8 ביטים.



Inputs:

- clk: System clock.
- reset: System reset.
- enc_data [11:0]: Encoded data from "Hamming Checker" or from write for read (in case only D0 or D1 had to be fixed).
- corrected_data [11:0]: Corrected data from "Hamming Fixer".
- sel_blk: Selects which input to retrieve the data from for decoding.
- check_out_read_valid: Indicates that reading is in progress.
- wfr_to_decode_valid: Indicates valid data from write-for-read operation.
- check_out_valid_D0: Validity flag for D0 syndrome check.
- check_out_valid_D1: Validity flag for D1 syndrome check.
- check_out_synd_D0 [3:0]: Syndrome bits for D0 indicating error position.
- check_out_synd_D1 [3:0]: Syndrome bits for D1 indicating error position.

Outputs:

- dec_data [7:0]: Decoded 8-bit data.
- decode_valid: Indicated dec_data is valid.

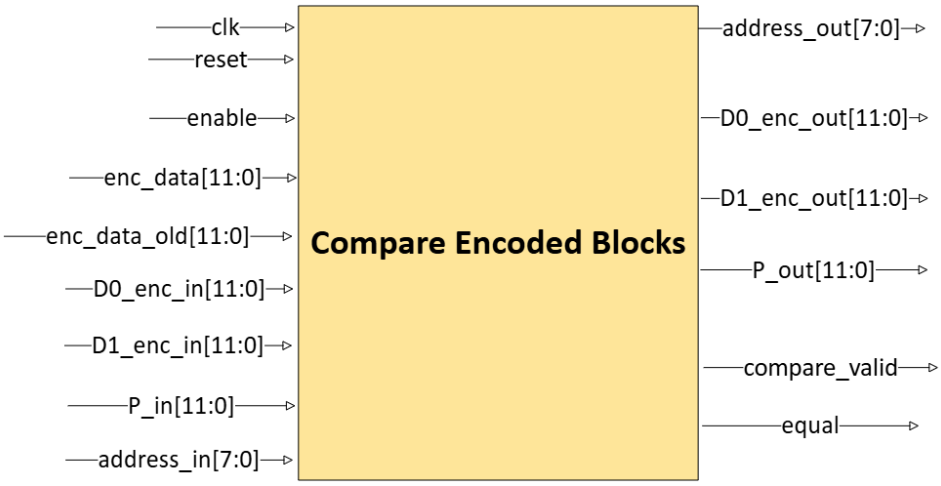
פעולת הרכיב:

במידה והתגלתה שגיאה בדאטה שנדרש לפיענוח, הכניסה שתיבחר היא corrected_data, אחרת תיבחר הכניסה enc_data.
לאחר מכן בוחרים להוציא בתור פלט את 8 הביטים שבהם שמור המידע.

Compare Encoded Blocks

תיאור הרכיב:

הרכיב **משווה** שני בלוקים מקודדים ומודיע האם הם זהים.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Activate signal.
- enc_data [11:0]: Encoded block from "Hamming Encoder".
- enc_data_old [11:0]: Encoded block from "Disk Reader".
- D0_enc_in [11:0]: Encoded 12-bit data.
- D1_enc_in [11:0]: Encoded 12-bit data.
- P_in [11:0]: Parity.
- address_in [7:0]: Address.

Outputs:

- address_out [7:0]: Address.
- D0_enc_out [11:0]: Encoded 12-bit data.
- D1_enc_out [11:0]: Encoded 12-bit data.
- P_out [11:0]: Parity.
- compare_valid: Indicates valid check for compare.
- equal: Indicates whether the two compared blocks are equal.

פעולת הרכיב:

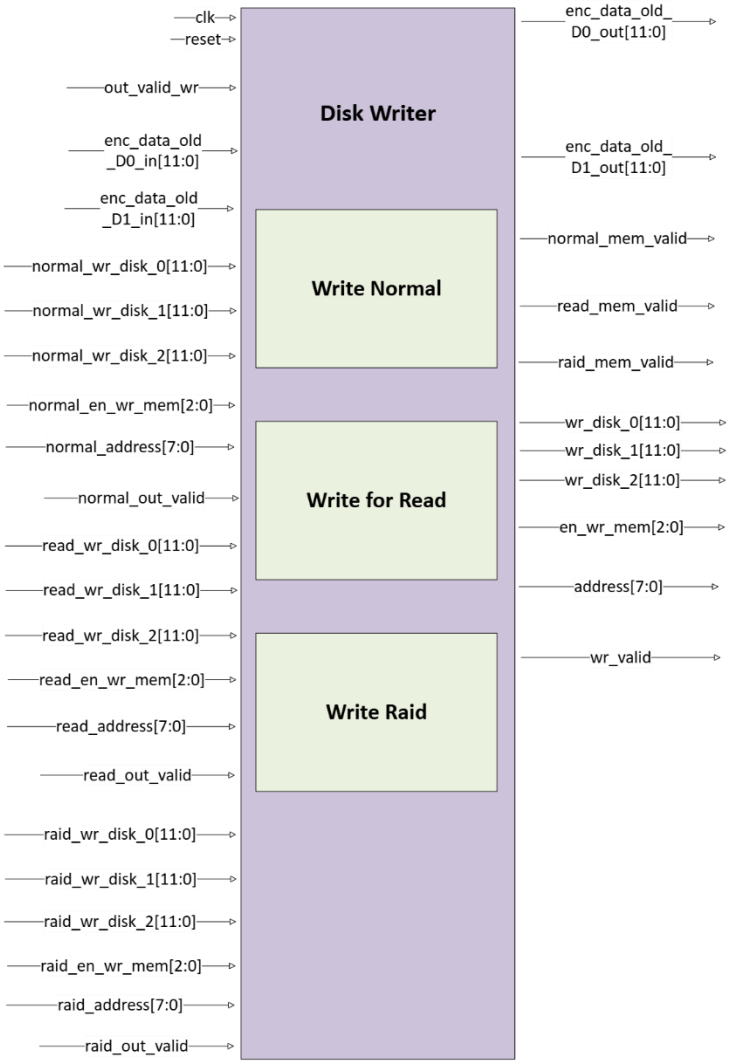
הדאטה הראשון להשוואה נכנס מהיחידה "Hamming Encoder". הדאטה השני להשוואה מגיע מהיחידה "Disk Reader".

במידה והם זהים אנו יודעים כי אין צורך לכתוב את הבלוק הנוכחי לזיכרון וכך נחסוך כתיבה מיותרת.

Disk Writer

תיאור הרכיב:

הרכיב משמש כ־MUX השולט בשלושת הרכיבים הפנימיים: Write Normal, Write for Read ו־Write RAID. מטרתו לפשט את בחירת המסלולים מהבקר, ולאפשר לוגיקה ברורה ונוחה יותר להבנה.



Inputs:

- clk: System clock.
- reset: System reset.
- out_valid_wr: Indicates completion of writing, signaling back to the originating block.
- enc_data_old_D0_in [11:0]: Previous encoded data for D0.
- enc_data_old_D1_in [11:0]: Previous encoded data for D1.
- normal_wr_disk_0 [11:0]: Data to write to disk 0 in normal mode.
- normal_wr_disk_1 [11:0]: Data to write to disk 1 in normal mode.
- normal_wr_disk_2 [11:0]: Data to write to disk 2 in normal mode.
- normal_en_wr_mem [2:0]: Enable signal for normal memory write.
- normal_address [7:0]: Address for normal write operation.
- normal_out_valid: Valid signal for normal write operation.
- read_wr_disk_0 [11:0]: Data to write to disk 0 from write-for-read.

- read_wr_disk_1 [11:0]: Data to write to disk 1 from write-for-read.
- read_wr_disk_2 [11:0]: Data to write to disk 2 from write-for-read.
- read_en_wr_mem [2:0]: Enable signal for write-for-read memory write.
- read_address [7:0]: Address for write-for-read operation.
- read_out_valid: Valid signal for write-for-read operation.
- raid_wr_disk_0 [11:0]: Data to write to disk 0 in RAID mode.
- raid_wr_disk_1 [11:0]: Data to write to disk 1 in RAID mode.
- raid_wr_disk_2 [11:0]: Data to write to disk 2 in RAID mode.
- raid_en_wr_mem [2:0]: Enable signal for RAID memory write.
- raid_address [7:0]: Address for RAID write operation.
- raid_out_valid: Valid signal for RAID write operation.

Outputs:

- wr_disk_0 [11:0]: Data to write to disk 0.
- wr_disk_1 [11:0]: Data to write to disk 1.
- wr_disk_2 [11:0]: Data to write to disk 2.
- en_wr_mem [2:0]: Enable write for the selected disks.
- address [7:0]: Address for the write operation.
- wr_valid: Write valid signal.

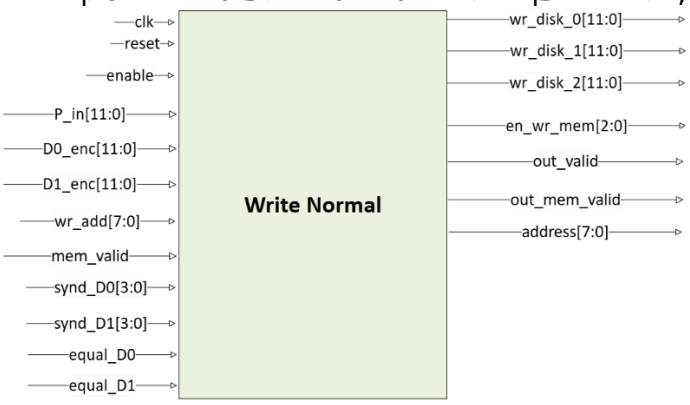
פעולת הרכיב:

1. קליטת נתונים - הרכיב מקבל נתונים מקודדים מיחידות שונות (כתיבה רגילה, כתיבה עבור קריאה, ושחזור RAID).
2. החלטה על מקור הנתונים - הרכיב בוחר את מקור הכתיבה הנוכחי על פי אותות הבקרה read_out_valid, normal_out_valid או raid_out_valid.
3. שליחת נתונים לזיכרון - לאחר בחירת מקור הנתונים, הרכיב מעביר את הנתונים לאחסון בדיסקים על פי הכתובת המתאימה ומפעיל את האותות הנדרשים.
4. אישור השלמת כתיבה - בסיום פעולת הכתיבה, הרכיב שולח אות wr_valid כדי לסמן לבלוק המקורי שהכתיבה הושלמה.

Write Normal

תיאור הרכיב:

הרכיב אחראי על **כתיבת מידע לדיסקים במצב עבודה רגיל**, תוך שימוש בקידוד המינג וזיהוי שגיאות על בסיס תבניות סינדרום.
הרכיב קובע אילו בלוקים יש לכתוב (D0, D1, P) בהתאם לאותות הסינדרום ולמצב הבלוקים בזיכרון, ולאחר מכן שולח את הנתונים לכתיבה בדיסקים המתאימים.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Indicates that comparison and error checks are completed, allowing normal write.
- P [11:0]: Parity block.
- D0_enc [11:0]: First encoded data from "Hamming Encoding."
- D1_enc [11:0]: Second encoded data from "Hamming Encoding."
- wr_add [7:0]: Address to which the data will be written.
- mem_valid: Indicates that memory has acknowledged the write request.
- synd_D0 [3:0]: Syndrome pattern indicating error location for D0.
- synd_D1 [3:0]: Syndrome pattern indicating error location for D1.
- equal_D0: Indicates whether the two D0 blocks are equal.
- equal_D1: Indicates whether the two D1 blocks are equal.

Outputs:

- wr_disk_0 [11:0]: Data to write to disk 0.
- wr_disk_1 [11:0]: Data to write to disk 1.
- wr_disk_2 [11:0]: Data to write to disk 2.
- en_wr_mem [2:0]: Enable write signals for the selected disks.
- out_valid: Valid output signal indicating data is ready for writing.
- out_mem_valid: Valid signal indicating that the write operation has been completed.
- address [7:0]: Address to be written.

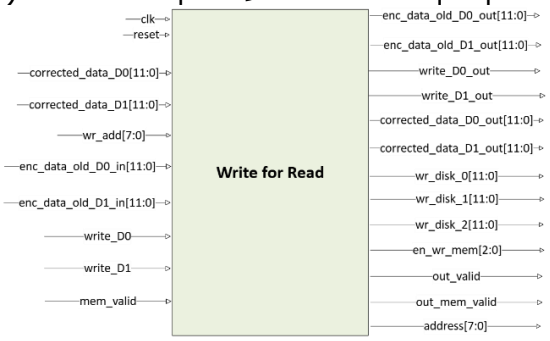
פעולת הרכיב:

1. <u>זיהוי בלוקים לכתיבה</u> - הרכיב מזהה אילו בלוקים יש לכתוב במידה ויש שגיאות ויודע לזהות מידע שכבר כתוב בזיכרון.
2. <u>קביעת יעד הכתיבה</u> - הרכיב מחליט לאילו דיסקים לכתוב את הנתונים, תוך התחשבות ב-parity.
3. <u>כתיבת הנתונים</u> - הנתונים נשלחים לכתיבה בדיסקים הרלוונטיים.
4. <u>אישור כתיבה</u> - בסיום פעולת הכתיבה, הרכיב מפעיל אות out_mem_valid לאישור השלמת הפעולה.

Write for Read

תיאור הרכיב:

הרכיב אחראי על **כתיבת נתונים מתוקנים** לאחר תיקון שגיאות באמצעות Hamming Fixer. הרכיב מזהה אילו בלוקים יש לכתוב (D0, D1) בהתאם למידע המתקבל ומבצע את פעולת הכתיבה לזיכרון תוך כדי שמירה על תקינות הפריטי (Parity).



Inputs:

- clk: System clock.
- reset: System reset.
- corrected_data_D0 [11:0]: First encoded data from "Hamming Fixer."
- corrected_data_D1 [11:0]: Second encoded data from "Hamming Fixer."
- wr_add [7:0]: Address to which the data will be written.
- enc_data_old_D0_in [11:0]: Previous encoded data for D0.
- enc_data_old_D1_in [11:0]: Previous encoded data for D1.
- write_D0: Indicates that corrected data for D0 needs to be written.
- write_D1: Indicates that corrected data for D1 needs to be written.
- mem_valid: Indicates that memory has acknowledged the write request.

Outputs:

- enc_data_old_D0_out [11:0]: Previous encoded data output for D0.
- enc_data_old_D1_out [11:0]: Previous encoded data output for D1.
- write_D0_out: Indicates a valid write operation for D0.
- write_D1_out: Indicates a valid write operation for D1.
- corrected_data_D0_out [11:0]: Corrected data output for D0.
- corrected_data_D1_out [11:0]: Corrected data output for D1.
- wr_disk_0 [11:0]: Data to write to disk 0.
- wr_disk_1 [11:0]: Data to write to disk 1.
- wr_disk_2 [11:0]: Data to write to disk 2.
- en_wr_mem [2:0]: Enable write signals for the selected disks.
- out_valid: Valid output signal indicating data is ready for writing.
- out_mem_valid: Valid signal indicating the write has been completed.
- address [7:0]: Address to be written.

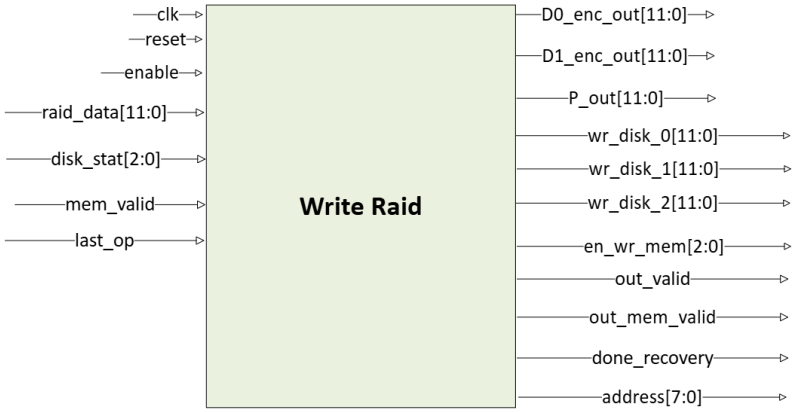
פעולת הרכיב:

1. <u>קליטת נתונים מתוקנים</u> - הרכיב מקבל את הנתונים המתוקנים מהמודול Hamming Fixer לאחר תיקון שגיאות.
2. <u>החלטה על דיסק היעד</u> - הנתונים מנותבים לדיסקים הרלוונטיים על פי כתובת הזיכרון והמבנה המחזורי של הפריטי.
3. <u>כתיבה לזיכרון</u> - לאחר קביעת היעדים, הנתונים נכתבים לדיסקים, והרכיב מסמן שהכתיבה הסתיימה על ידי הפעלת הסינגל out_mem_valid.

Write Raid

תיאור הרכיב:

הרכיב אחראי על **כתיבת נתונים משוחזרים** לדיסקים. תהליך זה מתבצע כאשר אחד הדיסקים נכשל, והרכיב כותב מחדש את הנתונים המשוחזרים לדיסק הפגום על בסיס שני הדיסקים שנותרו תקינים.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Indicates that RAID recovery is in progress.
- raid_data [11:0]: Recovered data from "RAID5 Recovery."
- disk_stat [2:0]: Indicates which disk needs to be recovered.
- mem_valid: Indicates that memory has acknowledged the write request.
- last_op: Indicates that the next operation will be the last in the recovery process.

Outputs:

- wr_disk_0 [11:0]: Data to write to disk 0.
- wr_disk_1 [11:0]: Data to write to disk 1.
- wr_disk_2 [11:0]: Data to write to disk 2.
- en_wr_mem [2:0]: Enable write signals for the selected disks.
- out_valid: Valid output signal indicating data is ready for writing.
- out_mem_valid: Valid signal indicating the write has been completed.
- done_recovery: Indicates to the controller that the disk recovery process is complete.
- address [7:0]: Address to be written.

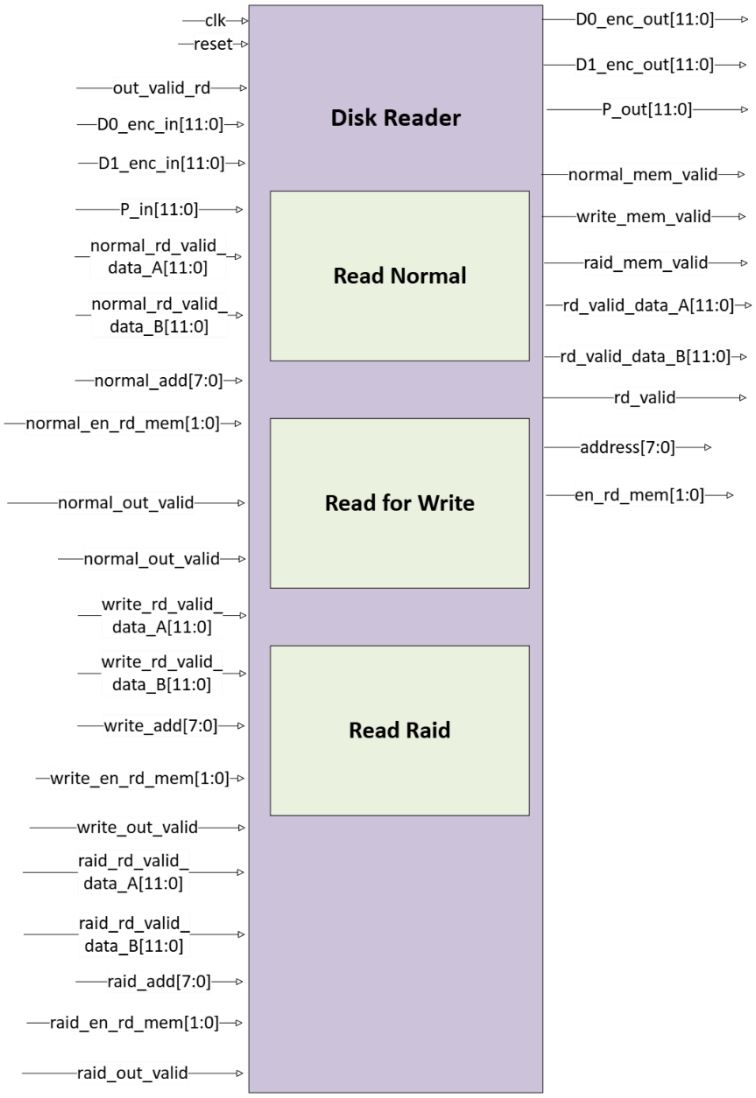
פעולת הרכיב:

1. קליטת נתונים משוחזרים - הרכיב מקבל את הנתונים ששוחזרו ממנגנון ה-RAID5.
2. זיהוי הדיסק שנכשל - מבצע בדיקה על פי disk_stat וקובע לאיזה דיסק יש לכתוב את הנתונים המשוחזרים.
3. כתיבה לדיסק הפגום - הנתונים נכתבים מחדש לכתובת הנדרשת בזיכרון.
4. אישור סיום השחזור - כאשר השחזור הושלם, הרכיב מפעיל את done_recovery כדי ליידע את בקר המערכת שהתהליך הסתיים.

Disk Reader

תיאור הרכיב:

הרכיב משמש כ־MUX השולט בשלושת הרכיבים הפנימיים: Read Normal, Read for Write ו־Read RAID. מטרתו לפשט את בחירת המסלולים מהבקר, ולאפשר לוגיקה ברורה ונוחה יותר להבנה.



Inputs:

- clk: System clock.
- reset: System reset.
- out_valid_rd: Indicates completion of reading, signaling back to the originating block.
- D0_enc_in [11:0]: Encoded 12-bit data.
- D1_enc_in [11:0]: Encoded 12-bit data.
- P_in [11:0]: Parity data.
- normal_rd_valid_data_A [11:0]: First data block read from memory.
- normal_rd_valid_data_B [11:0]: Second data block read from memory.
- normal_add [7:0]: Address for normal read operation.
- normal_en_rd_mem [1:0]: Enable signal for normal memory read.
- normal_out_valid: Valid signal for normal read operation.

- write_rd_valid_data_A [11:0]: First data block read from memory during write-for-read.
- write_rd_valid_data_B [11:0]: Second data block read from memory during write-for-read.
- write_add [7:0]: Address for write-for-read operation.
- write_en_rd_mem [1:0]: Enable signal for write-for-read memory read.
- write_out_valid: Valid signal for write-for-read operation.
- raid_rd_valid_data_A [11:0]: First data block read from memory in RAID mode.
- raid_rd_valid_data_B [11:0]: Second data block read from memory in RAID mode.
- raid_add [7:0]: Address for RAID read operation.
- raid_en_rd_mem [1:0]: Enable signal for RAID memory read.
- raid_out_valid: Valid signal for RAID read operation.

Outputs:

- D0_enc_out [11:0]: Encoded 12-bit data output.
- D1_enc_out [11:0]: Encoded 12-bit data output.
- P_out [11:0]: Parity output.
- normal_mem_valid: Indicates normal read operation is completed.
- write_mem_valid: Indicates write-for-read operation is completed.
- raid_mem_valid: Indicates RAID read operation is completed.
- rd_valid_data_A [11:0]: First read data block.
- rd_valid_data_B [11:0]: Second read data block.
- rd_valid: Read valid signal.
- address [7:0]: Address from which data is being read.
- en_rd_mem [1:0]: Enable signal for reading from the selected disks.

פעולת הרכיב:

1.קליטת בקשת קריאה - הרכיב מזהה האם הקריאה מגיעה מקריאה רגילה, קריאה עבור כתיבה, או קריאה לשחזור RAID.

2. שליחת בקשת קריאה לזיכרון - מתבצע על פי אותות הבקרה, normal_out_valid, write_out_valid או raid_out_valid.

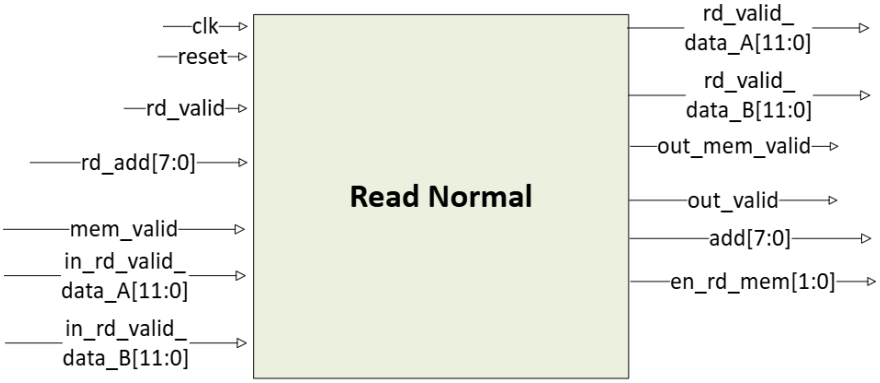
3. קבלת הנתונים מהזיכרון - לאחר הקריאה, הנתונים מועברים לפלט המתאים יחד עם rd_valid_data_A ו-rd_valid_data_B.

4. סימון סיום הקריאה - הרכיב מפעיל את rd_valid לסימון שהקריאה הושלמה, ולאחר מכן מסמן לבלוק המקורי שהמידע זמין לעיבוד.

Read Normal

תיאור הרכיב:

הרכיב אחראי על **קריאת נתונים מהזיכרון עבור קריאה רגילה** על פי כתובת.



Inputs:

- clk: System clock.
- reset: System reset.
- rd_valid: Indicates that a read request has started.
- rd_add [7:0]: Address from which data will be read.
- mem_valid: Indicates that memory has acknowledged the read request.
- in_rd_valid_data_A [11:0]: First read data block from memory.
- in_rd_valid_data_B [11:0]: Second read data block from memory.

Outputs:

- rd_valid_data_A [11:0]: First read data block.
- rd_valid_data_B [11:0]: Second read data block.
- out_mem_valid: Indicates that the read operation has completed.
- out_valid: Valid output signal indicating data is ready for reading.
- add [7:0]: Address from which data is being read.
- en_rd_mem [1:0]: Enable signal for reading from the selected disks.

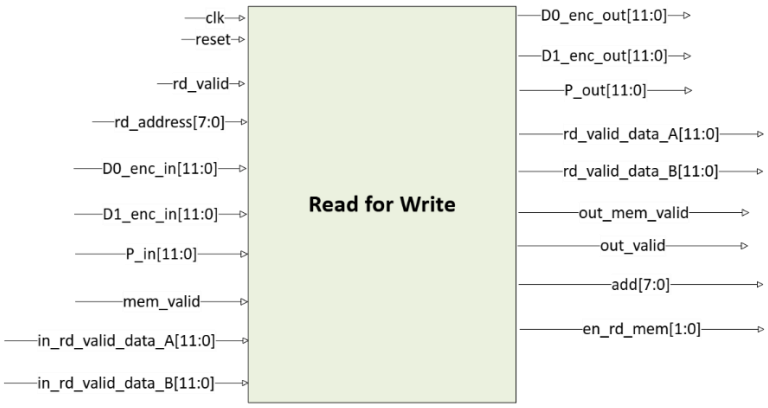
פעולת הרכיב:

1. קליטת בקשת קריאה - הרכיב מקבל אות rd_valid, המציין כי יש לבצע קריאה מהזיכרון.
2. חישוב דיסקים לקריאה - מתבצע חישוב באמצעות parity_disk כדי להגדיר אילו דיסקים נדרשים לקריאה.
3. שליחת בקשת קריאה לזיכרון - הכתובת והאתחול של אותות הקריאה מתבצעים.
4. קבלת הנתונים - כאשר הזיכרון מאשר את הקריאה באמצעות mem_valid, הנתונים מועברים לפלט.
5. סימון סיום הקריאה - אות out_mem_valid מסמן כי הנתונים זמינים לקריאה על ידי מודולים אחרים.

Read for Write

תיאור הרכיב:

הרכיב אחראי על **קריאת נתונים מהזיכרון לצורך כתיבה חדשה** לאחר חישוב פריטי. תהליך זה מתבצע כאשר יש צורך לבדוק את הנתונים הקיימים בזיכרון לפני ביצוע פעולת כתיבה.



Inputs:

- clk: System clock.
- reset: System reset.
- rd_valid: Indicates that a read request has started (from parity_calc).
- rd_add [7:0]: Address from which data will be read (provided by the user).
- D0_enc_in [11:0]: Encoded 12-bit data for D0.
- D1_enc_in [11:0]: Encoded 12-bit data for D1.
- P_in [11:0]: Parity data.
- mem_valid: Indicates that memory has acknowledged the read request (from disk_reader).
- in_rd_valid_data_A [11:0]: First read data block from memory.
- in_rd_valid_data_B [11:0]: Second read data block from memory.

Outputs:

- D0_enc_out [11:0]: Encoded 12-bit data output for D0.
- D1_enc_out [11:0]: Encoded 12-bit data output for D1.
- P_out [11:0]: Parity output.
- rd_valid_data_A [11:0]: First read data block.
- rd_valid_data_B [11:0]: Second read data block.
- out_mem_valid: Indicates that the read operation has completed.
- out_valid: Valid output signal indicating data is ready for reading.
- add [7:0]: Address from which data is being read.
- en_rd_mem [1:0]: Enable signal for reading from the selected disks.

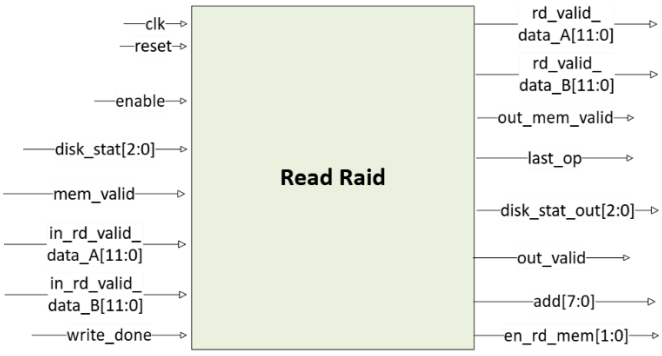
פעולת הרכיב:

1. קליטת בקשת קריאה - הרכיב מקבל אות rd_valid, המציין כי יש לבצע קריאה מהזיכרון לפני כתיבה.
2. חישוב דיסקים לקריאה - מבצע חישוב parity_disk כדי לקבוע אילו דיסקים מכילים את הנתונים הרלוונטיים.
3. שליחת בקשת קריאה לזיכרון - הקריאה מבוצעת בהתאם לאותות הבקרה והכתובת rd_add.
4. קבלת הנתונים - כאשר mem_valid פעיל, הנתונים נקלטים בפלטים המתאימים.
5. סימון סיום הקריאה - אות out_mem_valid מופעל כדי לאשר שהנתונים מוכנים לשימוש.

Read Raid

תיאור הרכיב:

הרכיב אחראי על **קריאת נתונים מהזיכרון כחלק מתהליך שחזור נתונים ב-RAID5** כאשר דיסק אחד נכשל. הוא מבצע קריאה מהדיסקים התקינים לצורך חישוב מחדש של הנתונים החסרים.



Inputs:

- clk: System clock.
- reset: System reset.
- enable: Indicates that the read operation should begin (triggered by memory after zeroing).
- disk_stat [2:0]: Indicates which disk(s) need to be read.
- mem_valid: Indicates that memory has acknowledged the read request.
- in_rd_valid_data_A [11:0]: First read data block from memory.
- in_rd_valid_data_B [11:0]: Second read data block from memory.
- write_done: Indicates that the current write operation is finished and the next read can begin.

Outputs:

- rd_valid_data_A [11:0]: First read data block.
- rd_valid_data_B [11:0]: Second read data block.
- out_mem_valid: Indicates that the read operation has completed.
- last_op: Indicates that the next operation will be the last in the RAID recovery process.
- disk_stat_out [2:0]: Indicates the disk status to be passed to the RAID write module.
- out_valid: Valid output signal indicating data is ready for reading.
- add [7:0]: Address from which data is being read.
- en_rd_mem [1:0]: Enable signal for reading from the selected disks.

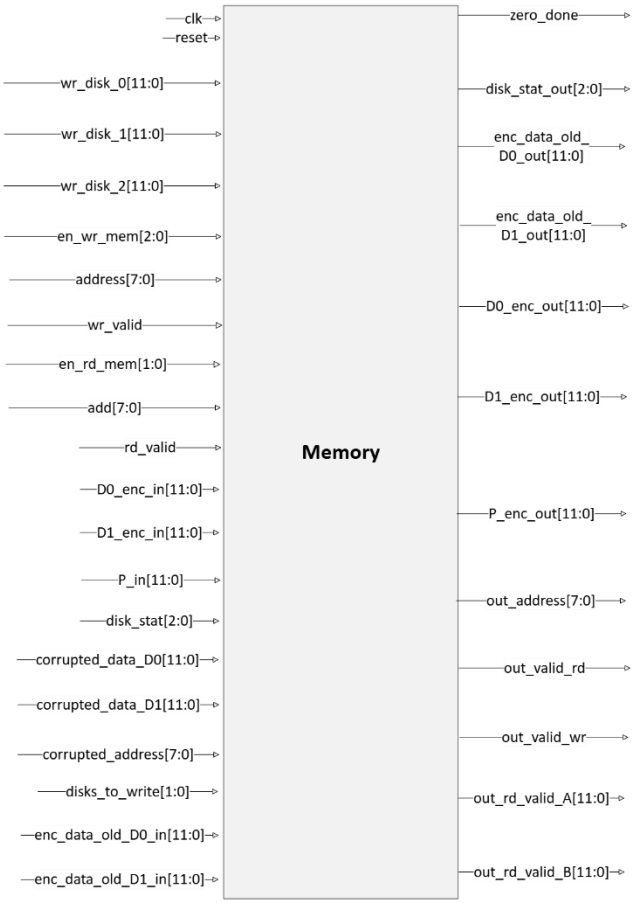
פעולת הרכיב:

1. <u>זיהוי הדיסקים התקינים</u> - הרכיב מקבל את disk_stat כדי לקבוע אילו דיסקים זמינים לקריאה.
2. <u>שליחת בקשת קריאה</u> - הרכיב מפעיל את אותות en_rd_mem כדי לאפשר קריאה מהדיסקים שנותרו.
3. <u>קבלת הנתונים</u> - כאשר mem_valid מאופשר, הנתונים נקלטים ומשמשים לשחזור הדיסק הפגום.
4. <u>מעבר לכתיבת הנתונים המשוחזרים</u> - כאשר מסתיימת קריאת כל הבלוקים, הרכיב שולח אות last_op כדי לציין שהשלב האחרון של הקריאה הסתיים.
5. <u>התקדמות לכתובות הבאות</u> - כתובת הקריאה מתעדכנת בכל מחזור שעון בהתאם לכתובות בזיכרון.

Memory

תיאור הרכיב:

הרכיב מדמה זיכרון חיצוני שמורכב ממערך של 3 כונני SSD עם 256 כתובות כל אחד, במערכת RAID5. תומך בקריאה וכתיבה.



Inputs:

- clk: System clock.
- reset: System reset (asynchronous).
- wr_disk0 [11:0]: Data to write to disk 0.
- wr_disk1 [11:0]: Data to write to disk 1.
- wr_disk2 [11:0]: Data to write to disk 2.
- en_wr_mem [2:0]: Enable signal indicating which disks to write.
- address [7:0]: Address to which data will be written.
- wr_valid: Indicates a valid write request from disk_writer.
- en_rd_mem [1:0]: Enable signal for reading from selected disks.
- add [7:0]: Address from which data will be read.
- rd_valid: Indicates a valid read request.
- D0_enc_in [11:0]: Encoded 12-bit data for D0.
- D1_enc_in [11:0]: Encoded 12-bit data for D1.
- P_in [11:0]: Parity data.
- disk_stat [2:0]: Indicates which disks are still valid (used for simulating a disk failure).
- corrupted_data_D0 [11:0]: Corrupted data to be written for D0.
- corrupted_data_D1 [11:0]: Corrupted data to be written for D1.
- corrupted_address [7:0]: Address where the corrupted data should be written.

- disks_to_write [1:0]: Indicates which disks should be written (00 - none, 01 - D0, 10 - D1, 11 - both).
- enc_data_old_D0_in [11:0]: Previous encoded data for checking.
- enc_data_old_D1_in [11:0]: Previous encoded data for checking.

Outputs:

- zero_done: Indicates that the disk has been zeroed and RAID recovery can start.
- disk_stat_out [2:0]: Indicates the updated disk status after zeroing a failed disk.
- enc_data_old_D0_out [11:0]: Previous encoded data output for checking.
- enc_data_old_D1_out [11:0]: Previous encoded data output for checking.
- D0_enc_out [11:0]: Encoded 12-bit data output for D0.
- D1_enc_out [11:0]: Encoded 12-bit data output for D1.
- P_out [11:0]: Parity output.
- out_address [7:0]: Address from which data was last accessed.
- out_valid_rd: Indicates that reading from memory has completed.
- out_valid_wr: Indicates that writing to memory has completed.
- out_rd_valid_A [11:0]: First read data block output.
- out_rd_valid_B [11:0]: Second read data block output.

פעולת הרכיב:

1. **כתיבת נתונים לזיכרון**:
   - מקבל נתונים מקודדים מהבקר (wr_disk0, wr_disk1, wr_disk2) וכותב אותם בהתאם לכתובת ולבקרת כתיבה (en_wr_mem, address).
   - מסמן שהכתיבה הושלמה דרך out_valid_wr.

2. **קריאת נתונים מהזיכרון**:
   - כאשר מתקבלת בקשת קריאה (rd_valid), הרכיב מחזיר את הנתונים המתאימים מהדיסקים דרך out_rd_valid_A ו-out_rd_valid_B.
   - מוודא שהנתונים נקראו בהצלחה עם out_valid_rd.

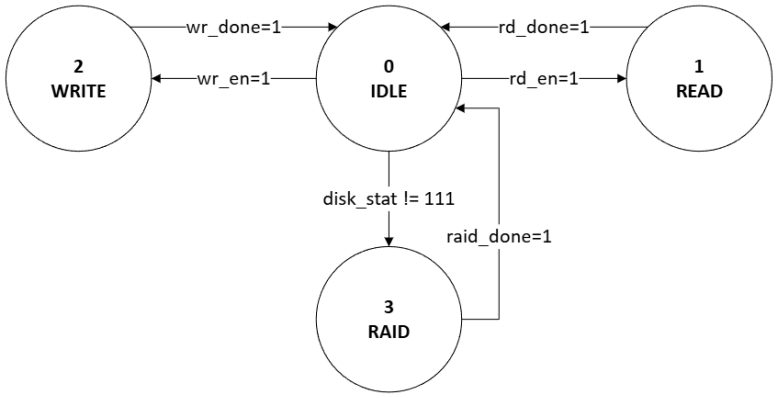
3. **שחזור נתונים במקרה של כשל**:
   - כאשר דיסק מסוים נכשל (disk_stat אינו 111), הרכיב מאפס את תוכן הדיסק המתאים כדי להכין אותו לשחזור.
   - שולח אות zero_done כדי לסמן שהדיסק אפס וניתן להתחיל בשחזור.

4. **כתיבת נתונים משוחזרים**:
   - כאשר מתקבלת בקשה לשחזור נתונים (disks_to_write), הנתונים המשוחזרים נכתבים חזרה לזיכרון בכתובת המתאימה.

Controller

תיאור הרכיב:

ניהול מכונת המצבים (FSM) של תהליכי המערכת: כתיבה, קריאה, שחזור RAID.



Inputs:

- clk: System clock.
- reset: System reset.
- wr_en: Write enable signal.
- write_data [15:0]: Data to be written.
- address [7:0]: Address to write data to.
- wr_done: Indicates that the write operation is completed.
- rd_en: Read enable signal.
- rd_done: Indicates that the read operation is completed.
- disk_stat [2:0]: Indicates the status of the disks.
- raid_done: Indicates that the RAID recovery process is completed.

Outputs:

- wr_en_out: Write enable output signal.
- write_data_out [15:0]: Data to be written output.
- address_out [7:0]: Address output.
- rd_en_out: Read enable output signal.
- disk_stat_out [2:0]: Disk status output.

פעולת הרכיב:

1. **מעבר בין מצבים (State Machine)**:
   - IDLE – המערכת במצב המתנה, מחכה לפקודת קריאה/כתיבה או לזיהוי כשל דיסק.
   - WRITE – מתבצע תהליך כתיבה כאשר מתקבלת בקשת כתיבה.
   - READ – מתבצע תהליך קריאה כאשר מתקבלת בקשת קריאה.
   - RAID – מתבצע תהליך שחזור במקרה של כשל דיסק.

2. **כתיבת נתונים**:
   - כאשר מתקבלת בקשת כתיבה (wr_en), הבקר עובר למצב WRITE ומעביר את הנתונים המתאימים לכתובת המבוקשת.
   - לאחר השלמת הכתיבה (wr_done), חוזר למצב IDLE.

3. **קריאת נתונים**:
   - כאשר מתקבלת בקשת קריאה (rd_en), הבקר עובר למצב READ ושולח את הכתובת הרלוונטית.
   - בסיום הקריאה (rd_done), חוזר למצב IDLE.

4. **שחזור נתונים (RAID Recovery)**:
   - במקרה של כשל דיסק (disk_stat אינו 111), הבקר עובר למצב RAID.
   - כאשר תהליך השחזור מסתיים (raid_done), הבקר חוזר למצב IDLE.

סימולציה

מטרת הסימולציה היא לדמות סביבת עבודה מציאותית, לבדוק את תפקוד הרכיב, לוודא את תקינות הפלט, ולבחון את יעילות השבב.

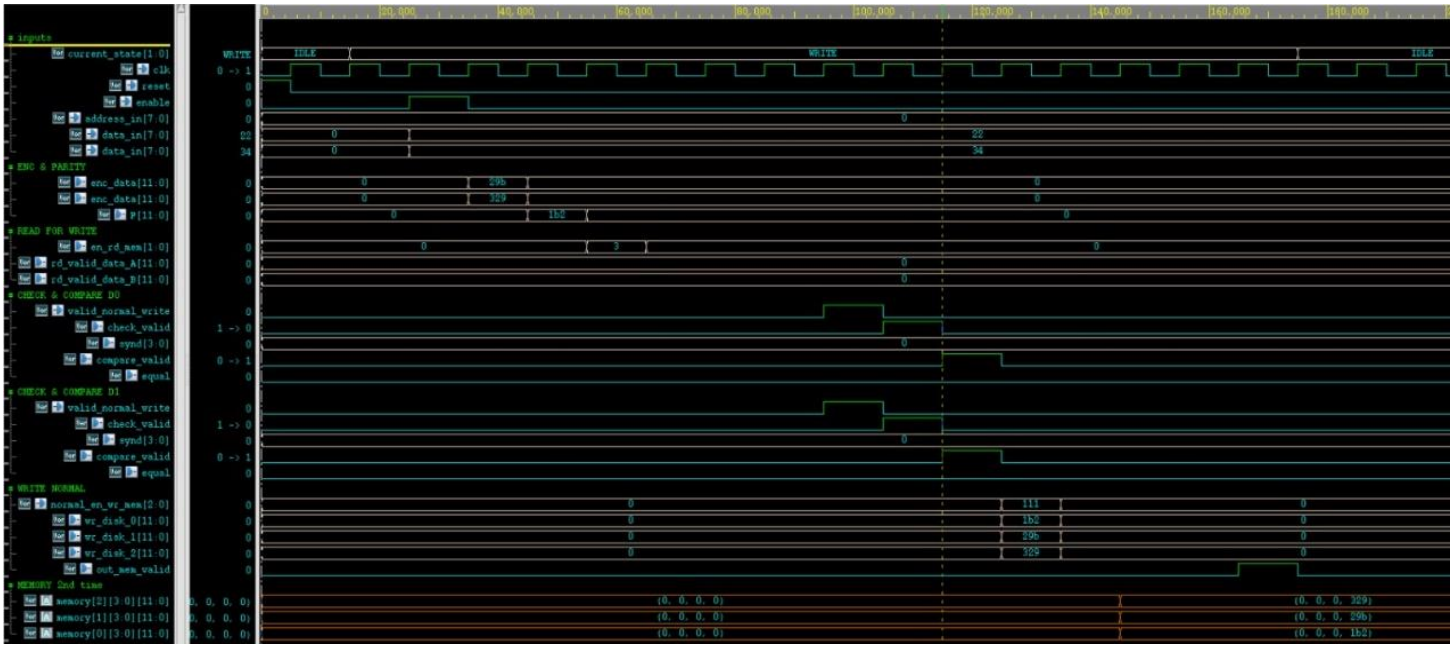
שלבי הסימולציה:

- תחילה, נכתב testbench לכל רכיב בנפרד באופן עצמאי, במטרה לוודא את פעולתו התקינה.
- לאחר מכן, נכתב testbench ייעודי לבדיקת פעולתו הכוללת של הרכיב.
- במהלך תהליך הבדיקה, בכל שלב נוסף לריצת הבדיקה הרכיב הבא בשרשרת, ורק לאחר שנבדקה תקינות האינטגרציה החלקית ואומתה כנכונה, הוסף הרכיב הבא בתור. תהליך זה נמשך עד להשלמת הבדיקה עבור המערכת כולה.

כעת נציג סימולציות שונות שעשינו כדי לוודא את תקינות הרכיב.

1. <u>מטרת הסימולציה</u>: כתיבה רגילה לזיכרון (write normal) מסלול מלא, ללא שגיאות בבלוקים הכתובים בזיכרון.

ניתן לראות כי מדלגים על ההשוואה בין הבלוקים וכותבים מיד לזיכרון.



ניתן לראות את קידוד המינג המופעל על הבלוקים:
D0: 0x22 (b <u>0010</u> <u>0010</u>) -> 0x29b (b <u>0010</u> **1**<u>001</u> **1**<u>0</u>**11**)
D1: 0x34 (b <u>0011</u> <u>0100</u>) -> 0x329 (b <u>0011</u> **0**<u>010</u> **1**<u>0</u>**01**)

כמו כן, ניתן לראות את בלוק הפריטי שערכו מחושב לפי XOR בין הבלוקים המקודדים:
P: **XOR**(0x29b,0x329) =
XOR(b <u>0010</u> **1**<u>001</u> **1**<u>0</u>**11** , b <u>0011</u> **0**<u>010</u> **1**<u>0</u>**01**) = 0x1b2 (b 0001 1011 0010)

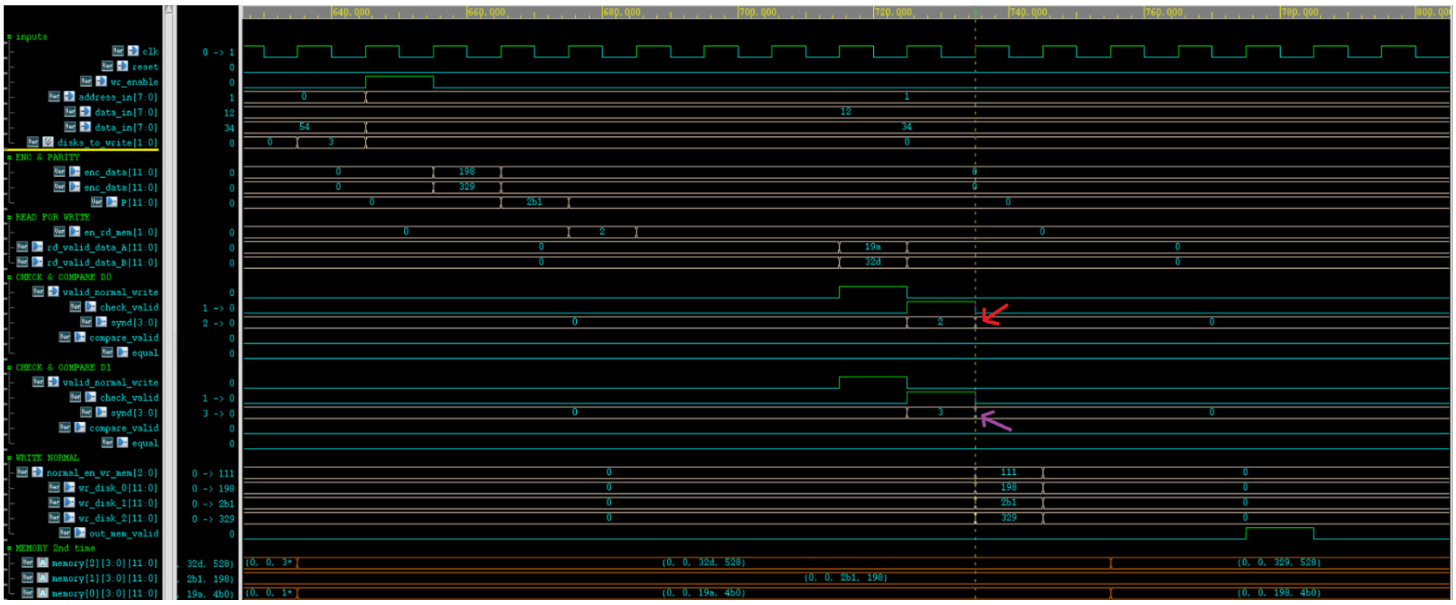
<u>ביטים מודגשים</u> – ביטים של פריטי שנוספו מהקידוד.

<u>ביטים עם קו תחתון</u> – ביטים של דאטה.

2. מטרת הסימולציה: כתיבה רגילה לזיכרון (write normal) עם 2 שגיאות, כלומר דילוג על שלב compare וכתיבה ישר לזיכרון.

חץ אדום – הסינדרום שונה מ-0 ולכן יש שגיאה בבלוק D0 (שגיאה בביט מספר 2). לכן, ניתן לראות כי מדלגים על ההשוואה בין הבלוקים וכותבים מיד לזיכרון.

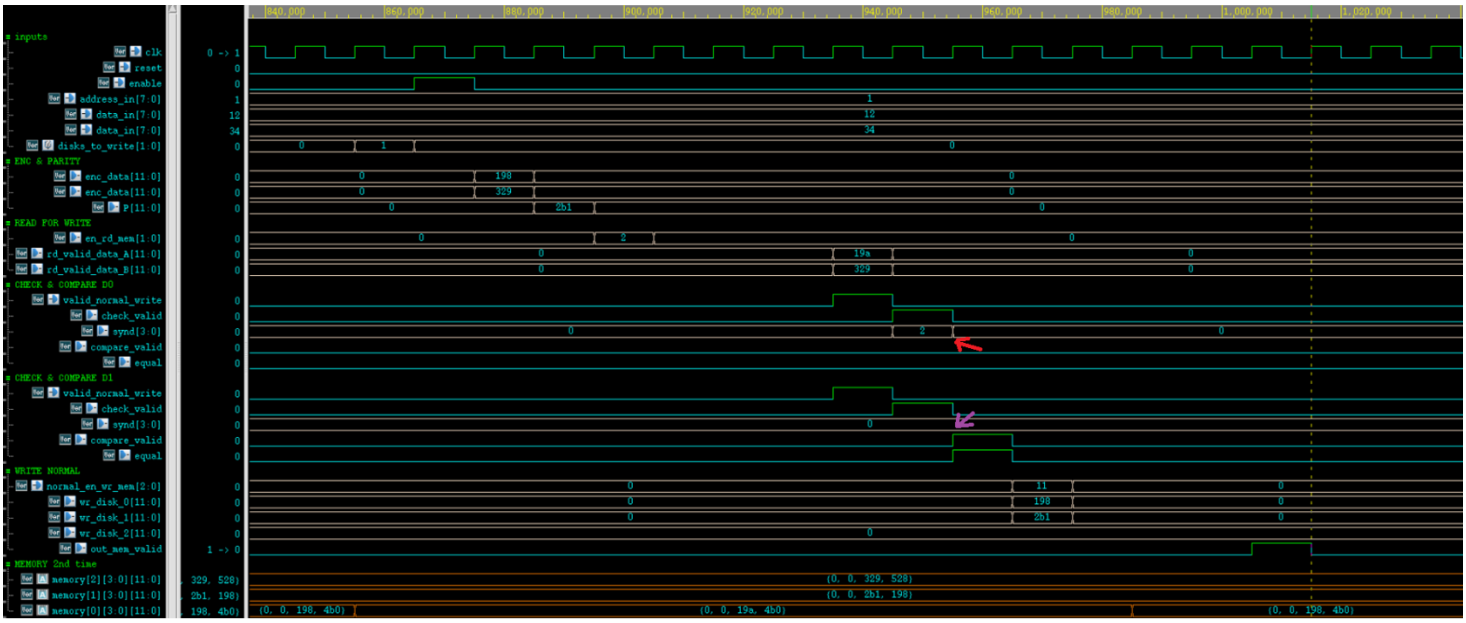
חץ סגול– הסינדרום שונה מ-0 ולכן יש שגיאה בבלוק D1 (שגיאה בביט מספר 3). לכן, ניתן לראות כי מדלגים על ההשוואה בין הבלוקים וכותבים מיד לזיכרון.



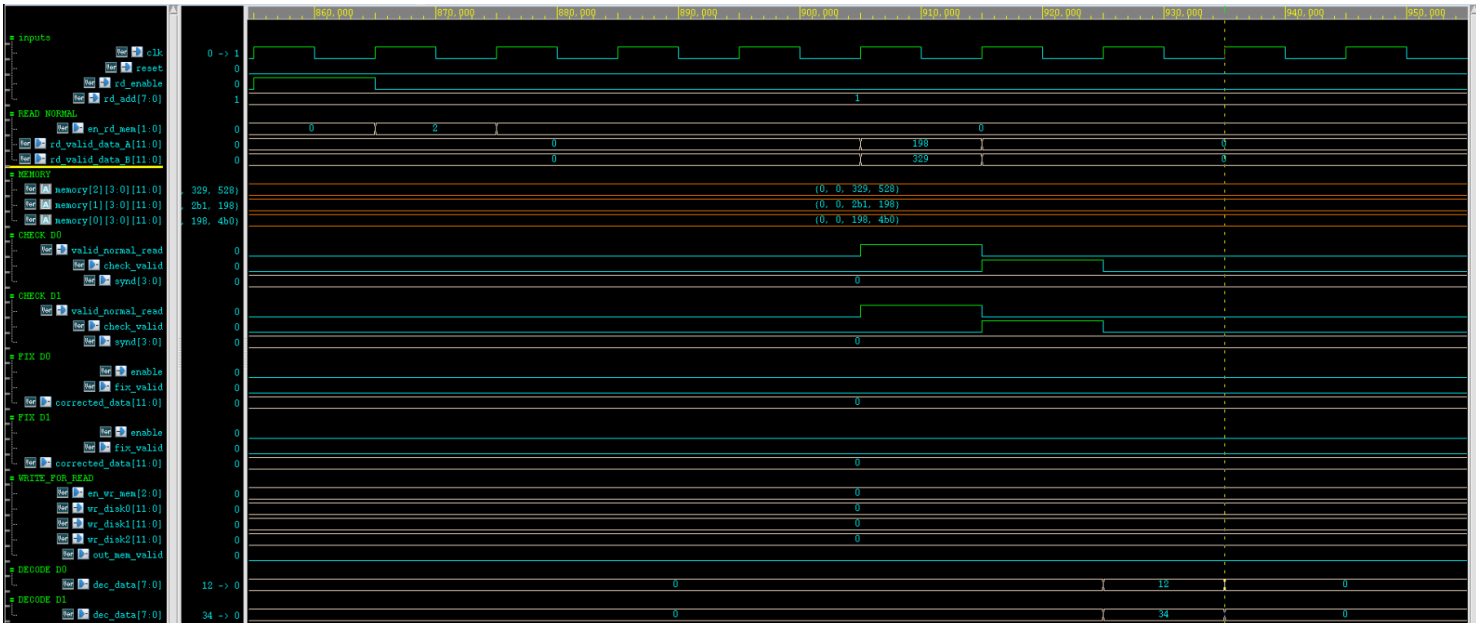
3. מטרת הסימולציה: כתיבה רגילה לזיכרון (write normal) עם שגיאה יחידה, רואים כי רק השוואה אחת קורית בתהליך בזמן שהשני ממתין.

חץ אדום – הסינדרום שונה מ-0 ולכן יש שגיאה בבלוק D0 (שגיאה בביט מספר 2). לכן, ניתן לראות כי מדלגים על ההשוואה בין הבלוקים וכותבים מיד.

חץ סגול – הסינדרום שונה מ-0 ולכן אין שגיאה בבלוק D1. ניתן לראות כי במחזור שעון לאחר מכן קורית פעולת השוואה בין הבלוקים. במחזור שעון שאחריו, הסיגנל equal=1 ולכן אנחנו לא כותבים את הבלוק לזיכרון וכך חוסכים כתיבה מיותרת.



4. מטרת הסימולציה: קריאה רגילה מהזיכרון (write normal), ללא שגיאות.



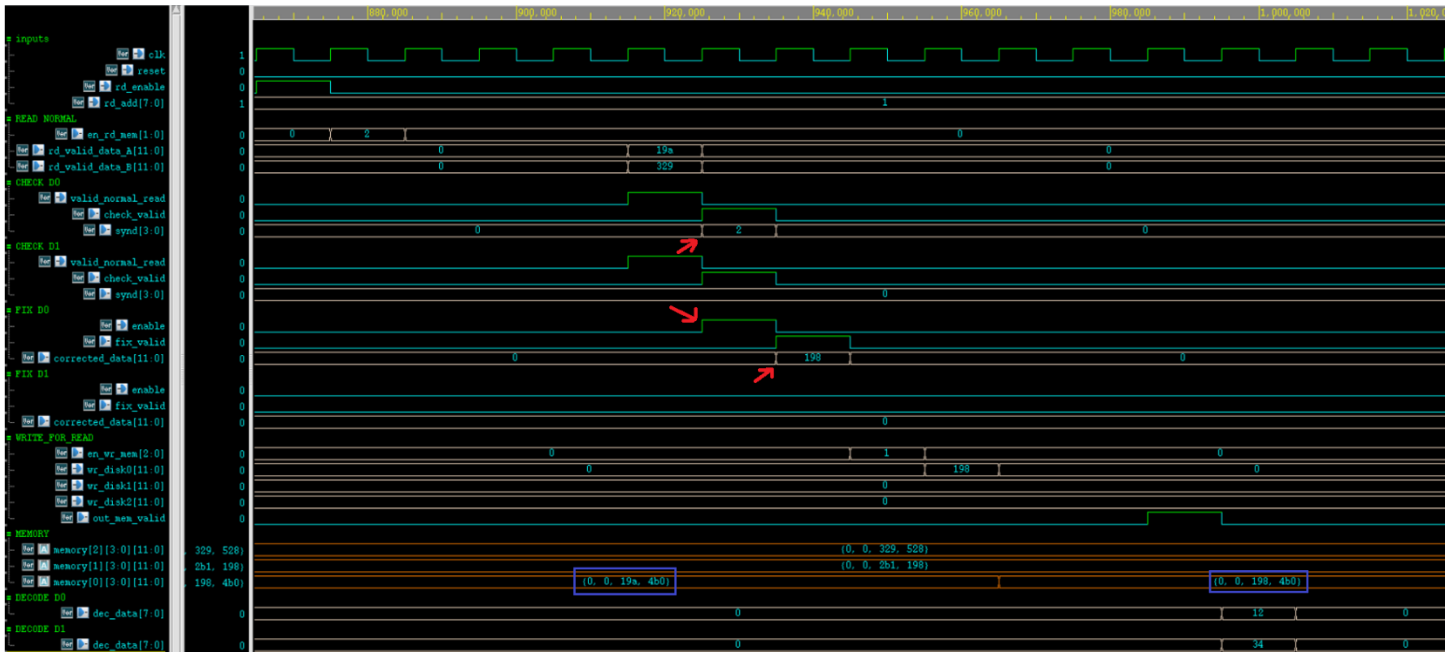
5. מטרת הסימולציה: קריאה רגילה מהזיכרון (read normal), עם SBE בבלוק D0.

חיצים אדומים – ניתן לראות כי זוהתה שגיאה בבלוק D0 בביט מספר 2 ולכן Fix D0 נכנס לפעולה והופך את הביט השגוי מ-1 ל-0. בסוף ניתן לראות כי המידע המפוענח יוצא נכון, 0x1234.

0x19a (b 0001 1001 1010) -> 0x198 (b 0001 1001 1000)

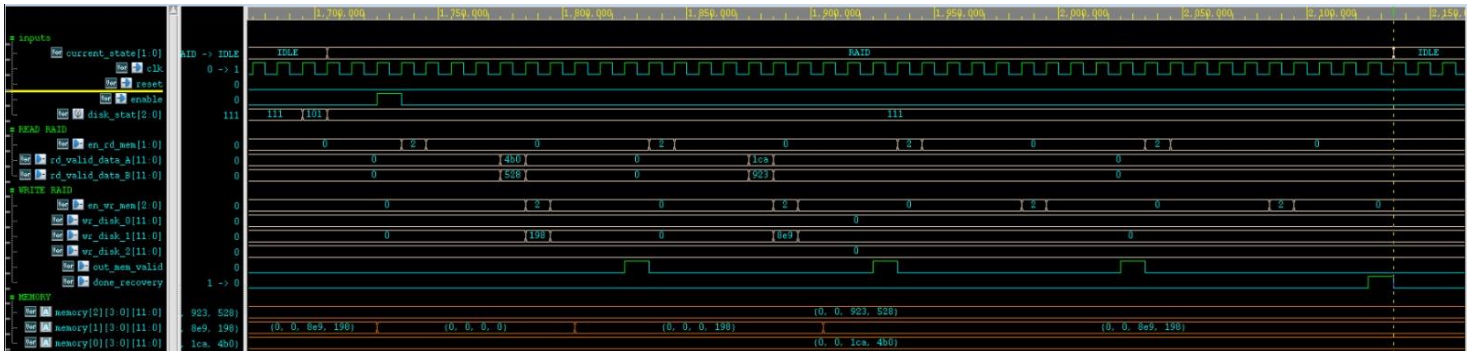
מלבנים סגולים – צד שמאל: מציג את המצב בזיכרון עם הבלוק השגוי.

צד ימין: מציג את המצב בזיכרון עם הבלוק המתוקן.



6. <u>מטרת הסימולציה</u>: איפוס + שחזור RAID

ניתן לראות כי disk_stat=101, ולכן ניתן להבין שדיסק מספר 2 במצב כשל. לטובת דימוי של כשל בדיסק, יצרנו מנגנון שכותב אפסים לכל הבלוקים בדיסק. במחזור שעון לאחר מכן דימינו החלפה של הדיסק הכושל בדיסק חדש שלתוכו ייכתבו הנתונים המשוחזרים ונכנסו במכונת מצבים למצב "raid". ניתן לראות כי מצב הזיכרון לפני הכשל זהה למצב הזיכרון לאחר הכשל מה שמעיד על שחזור מוצלח.



סינתזה

שלב הסינתזה בוצע באמצעות הכלי Design Vision מבית Synopsys. בשלב זה, המודולים שפותחו בשפת החומרה SystemVerilog הומרו לרכיבים חומרתיים, כגון שערים לוגיים, רגיסטרים ותאי ספרייה הקיימים בכלי. בסיום התהליך הופקו שלושה דוחות מרכזיים: מהירות, שטח, וצריכת הספק, המספקים תובנות חשובות לגבי ביצועי המערכת והאופטימיזציה שלה.

מהירות

```
data arrival time                                    4.20

clock CLK (rise edge)                       10.00    10.00
clock network delay (ideal)                  0.00    10.00
mem/memory_reg_1__2__0_/CP (dfcrq1)          0.00    10.00 r
library setup time                          -0.10     9.90
data required time                                    9.90
-------------------------------------------------------------
data required time                                    9.90
data arrival time                                    -4.20
-------------------------------------------------------------
slack (MET)                                           5.70
```

חישוב ה-slack (מרווח תזמון) מראה ערך חיובי של [ns] 5.70, המעיד כי הנתונים מגיעים מוקדם מהנדרש ולכן המסלול עומד בדרישות התזמון (MET).

כדי לחשב את תדר השעון המקסימלי שבו ניתן לעבוד במסלול זה, משתמשים בזמן הדרישה לנתונים (data required time) שהוא למעשה ה־clock period המינימלי הדרוש.

זמן מחזור מקסימלי מותר: [ns] 9.90.

תדר שעון מקסימלי:

$$f_{max} = \frac{1}{T_{min}} = \frac{1}{9.90[ns]} = 101.01\ MHz$$

שטח

```
****************************************
Report : area
Design : chip
Version: U-2022.12
Date   : Thu Feb 27 18:31:01 2025
****************************************

Library(s) Used:

    tsl18fs120_typ (File: /tools/kits/tower/PD

Number of ports:                          2289
Number of nets:                           6321
Number of cells:                          4129
Number of combinational cells:            2898
Number of sequential cells:               1208
Number of macros/black boxes:                0
Number of buf/inv:                         663
Number of references:                       33

Combinational area:                4035.250000
Buf/Inv area:                       483.750000
Noncombinational area:             6347.250000
Macro/Black Box area:                 0.000000
Net Interconnect area:             3100.510606

Total cell area:                  10382.500000
Total area:                       13483.010606

***** End Of Report *****
```

שטח הרכיב הינו 13,483 ביחידות של שערי NAND.

הספק

```
Power-specific unit information :
    Voltage Units = 1V
    Capacitance Units = 1.000000pf
    Time Units = 1ns
    Dynamic Power Units = 1mW    (derived from V,C,T units)
    Leakage Power Units = 1pW


Attributes
----------
i - Including register clock pin internal power


  Cell Internal Power  =  10.1168 mW   (95%)
  Net Switching Power  = 488.4557 uW    (5%)
                         ---------
Total Dynamic Power    =  10.6053 mW  (100%)

Cell Leakage Power     = 233.5946 nW


                 Internal         Switching           Leakage            Total
Power Group      Power            Power               Power              Power   (   %    )  Attrs
--------------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network      9.8733           0.0000            0.0000            9.8733  (  93.10%)  i
register           0.1210        2.5324e-02        1.5170e+05           0.1465  (   1.38%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational      0.1225           0.4631        8.1895e+04           0.5857  (   5.52%)
--------------------------------------------------------------------------------------------------
Total             10.1168 mW        0.4885 mW     2.3359e+05 pW       10.6055 mW

***** End Of Report *****
```

ההספק במעגל מורכב משני רכיבים עיקריים:

הספק סטטי: [nW] 233.59.

הספק דינמי: [mW] 10.60.

ההספק הסטטי (leakage power) בטכנולוגיית CMOS נובע ממספר מקורות זליגה, ובהם:

- זרם תת-הולכה (subthreshold leakage) הזורם בטרנזיסטורים במצב ניתוק.
- זרם מנהור דרך שער הטרנזיסטור (gate tunneling leakage).
- זרמי זליגה מדיודות בהטענה הפוכה (reverse-bias junction leakage).

ההספק הדינמי מקורו במעברי מצב (מיתוגים) של רכיבי המעגל במהלך פעילותו.

בדרך כלל, ההספק הסטטי נמוך משמעותית מההספק הדינמי, ולכן רוב האנרגיה במעגל נצרכת בפעולה רגילה ולא אובדת בזליגה.

ניתן גם לבחון את פיזור ההספק הסטטי בין רכיבי המעגל השונים – לוגיקה צירופית, לוגיקה סינכרונית, רגיסטרים וקווי שעון – כדי לזהות מוקדי זליגה ולבצע אופטימיזציה להפחתת ההספק הכולל.

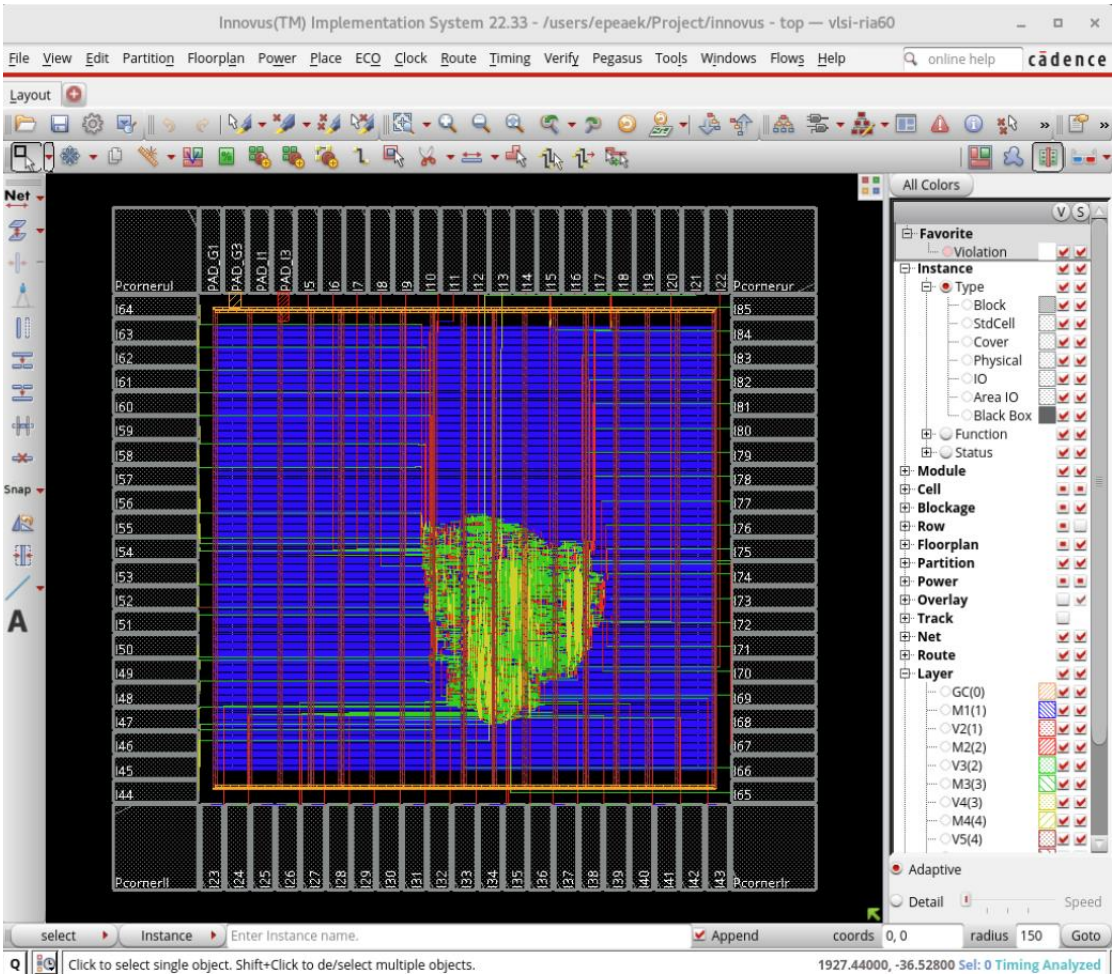
Layout

לאחר יצירת הקובץ שעבר סינתזה, יצרנו את ה-layout באמצעות כלי Innovus של חברת Cadence, בטכנולוגיית Tower CMOS 0.18µm.

בשלב זה, תאי המעגל שנוצרו במהלך הסינתזה הומרו לצורות גיאומטריות המתאימות לייצור.

הפעולות שבוצעו במהלך שלב זה:

- הוספת pads.
- הגדרת רשת קווי האספקה (VDD, VSS).
- מיקום התאים (floorplan).
- איזון עץ השעון – (CTS - Clock Tree Synthesis) למניעת הפרשי השהיה בין המסלולים השונים.
- מילוי הרווחים ב- grid באמצעות הוספת filler cells.
- חיווט קווי האספקה, חיווט פנימי של הרכיב, והוספת הכניסות והיציאות.
- שטח הרכיב: 2072 × 2006 [µm²].



סיכום ומסקנות

תהליך הפיתוח

תחילת הפרויקט כללה ביצוע **סקר ספרות** בתחום Error Correction Codes and RAID Accelerators for Solid-State Drives. לאחר קריאת מאמרים אקדמיים רלוונטיים, בחרנו להתמקד בפתרון המשלב את עקרונות **RAID 5** יחד עם **Hamming ECC**, וביססנו את האלגוריתם המתאים למימוש.

בהמשך, בנינו **מודל תוכנה ב-Python**, אשר מדמה את פעולות הקריאה והכתיבה של מערכת הזיכרון באופן המשקף את ההתנהגות החומרתית. המודל שימש להדגמת היתכנות הרכיב, ולשם כך הצגנו גרפים המאשרים את ביצועיו.

לאחר מכן, עברנו **לתכנון המיקרו-ארכיטקטורה**, שבו הפרדנו את פעולות המערכת לבלוקים עצמאיים, מתוך מטרה לשפר את יעילות המימוש החומרתי ולהקל על האינטגרציה. עם סיום שלב זה, **פיתחנו את המודולים בשפת SystemVerilog**, תוך התמקדות בתקשורת תקינה בין המודולים ובתאימות ביניהם.

כל מודול שנבנה לווה **בכתיבת testbench** ייעודי, אשר נועד לוודא שהפלטים תואמים לציפיות. בהמשך, ביצענו **אינטגרציה הדרגתית** בין המודולים, תוך הרחבת testbench באופן שיטתי, עד להשלמת האינטגרציה של המערכת כולה.

בסיום תהליך הפיתוח, ביצענו **סימולציה מקיפה על כל הרכיב**, שבה נבדקו כל התרחישים האפשריים, כולל מעבר בין כל המצבים במכונת המצבים של הבקר. התוצאות החומרתיות תאמו את המודל התוכנתי, מה שאישר את תקינות התכנון.

בהמשך, באמצעות Design Vision ביצענו **סינתזה**, שהמירה את מודולי ה-RTL לרכיבים אלקטרוניים. לאחר מכן, בעזרת Innovus, יצרנו **layout** פיזי, המתאים לתכנון החומרתי של הרכיב.

סיכום ולמידה

במהלך הפרויקט למדנו רבות על שלבי הפיתוח החומרתי – החל מהמחקר הראשוני, דרך מימוש בתוכנה ובחומרה, פיתוח RTL, סינתזה, ועד ליצירת layout תוך התחשבות בפרמטרים קריטיים כגון יעילות, מקביליות, תזמון (timing), שטח ועוד, במטרה לתכנן רכיב אופטימלי.

מקורות

1. A Performance Evaluation of RAID Architectures

Shenze Chen, D. Towsley.

https://ieeexplore.ieee.org/document/543706

2. Multiple Upset Tolerant Memory Using Modified Hamming Code
Babitha Antony, Divya S

https://www.iosrjournals.org/iosr-jece/papers/ICETEM/Vol.%201%20Issue%201/ECE%2005-31-39.pdf