מערכות ספרתיות ומבנה המחשב (044252)

סמסטר אביב תשפ"א

בחינה סופית – מועד א - פתרון

2 באוגוסט 2021

טור 1

מספר סטודנט

משך המבחן: 3 שעות (180 דקות). **תכננו את זמנכם היטב.**

חומר עזר: אין להשתמש בכל חומר עזר בכתב, מודפס או אלקטרוני, פרט לדפי העזר ולמחשבון.

הנחיות והוראות:

- הבחינה כתובה על גבי **21** עמודים כולל עמוד זה (לא רלוונטי עבור קובץ הפתרון) (**בדקו בתחילת הבחינה שלא חסרים לכם עמודים**). בסה"כ ישנן 15 שאלות: 10 שאלות אמריקאיות, ו- 5 שאלות פתוחות מרובות סעיפים.
- בתחילת הבחינה תקבלו חוברת בחינה, מחברת טיוטה, דפי עזר וטופס תשובות ממוחשב. בסיום הבחינה, החזירו את חוברת הבחינה וטופס התשובות הממוחשב בלבד.
- יש לענות על כל השאלות הפתוחות בגוף המבחן, **במלבנים המסומנים לכך בלבד.**
- אין **לתלוש או להפריד** דפים מחוברת הבחינה, ממחברות הטיוטה ומדפי העזר.
- יש לכתוב את התשובות באמצעות עט שחור או כחול בלבד. **אין לכתוב או לצייר בעט אדום**.
- רשמו את מספר הסטודנט שלכם על חוברת הבחינה (בראש עמוד זה). **ודאו כי על מחברת הבחינה ועל טופס התשובות האמריקאי מודבקת מדבקת הנבחן שלכם.**
- לא מורדות נקודות (אין "קנס") בגין תשובה שגויה. לכן, בשאלות האמריקאיות כדאי לסמן תשובה כלשהי לכל שאלה.
- ציון השאלות האמריקאיות ייקבע על סמך סריקה ממוחשבת של טופס התשובות בלבד. **לא לשכוח לסמן בטופס התשובות הממוחשב את מספר הטור שלכם (מופיע בראש עמוד זה).**
- אסור שימוש בכל חומר חיצוני מלבד מחשבון. אסורה העברת חומר כלשהו בין הנבחנים, ואסורה כל תקשורת עם אנשים אחרים או כל מקור מידע. האיסור חל על כל צורות התקשורת – מילולית, חזותית, כתובה, אלקטרונית, אלחוטית, טלפתית, או אחרת. בפרט, אין להחזיק בטלפון סלולארי.

בהצלחה!

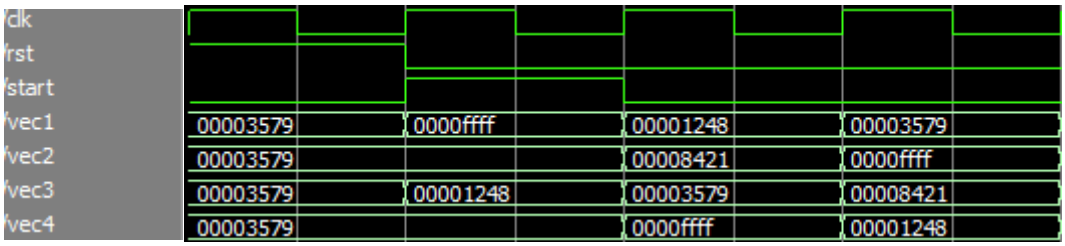
שאלה 1 (5 נקודות):

נתון קוד SytemVerilog הבא:

```
module my_mod(
    input logic clk,
    input logic rst,
    input logic start,
    output logic [31:0] vec
);

    logic [1:0] z;
    logic [1:0] b;
    logic x;
    logic y;

    fsm fsm_inst(
    .clk(clk),
    .rst(rst),
    .start(start),
    .z(z)
    );

    always_comb begin
        vec = 32'hABCD;

        x = z[1] ^ z[0];
        y = z[1] | z[0];

        case (b)
            2'b00: begin
                vec = 32'h3579;
            end
            2'b01: begin
                vec = 32'hFFFF;
            end
            2'b10: begin
                vec = 32'h1248;
            end
            2'b11: begin
                vec = 32'h8421;
            end
        endcase
    end


    assign b = {x,y};

endmodule
```

```
module fsm(
    input logic clk,
    input logic rst,
    input logic start,
    output logic [1:0] z
);

    typedef enum {
        idle_st,
        state_1,
        state_2
    } sm_type;

    sm_type current_state;
    sm_type next_state;


        // FSM synchronous procedural block.
    always_ff @(posedge clk, posedge rst) begin
        if (rst == 1'b1) begin
            current_state <= idle_st;
        end
        else begin
            current_state <= next_state;
        end
    end


    always_comb begin
        next_state = current_state;
        z = 2'b0;

        case (current_state)
            idle_st: begin
                if (start == 1'b1) begin
                    next_state = state_1;
                end
            end
            state_1: begin
                next_state = state_2;
                z = 2'b1;
            end
            state_2: begin
                next_state = idle_st;
                z = 2'd3;
            end
        endcase
    end

endmodule
```

בנוסף, נתון הכניסות ל- my_mod וארבע דיאגרמות אפשריות ליציאה vec.



סמן את התשובה הנכונה ביותר:

א. vec ייראה כמו דיאגרמה vec1 בסימולציה.

ב. vec ייראה כמו דיאגרמה vec2 בסימולציה.

ג. vec ייראה כמו דיאגרמה vec3 בסימולציה.

ד. vec ייראה כמו דיאגרמה vec4 בסימולציה.

ה. אף תשובה לא נכונה

פתרון:

תשובה: ב

In Idle state z=0. So b={x,y}=2'b00 and we output 'h3579. In state_1 b=2'b11 so we output 'h8421. In state_2 b=2'b01 so we output 'hFFFF.

שאלה 2 (5 נקודות):

נגדיר משתנה כניסה טרינארי של פונקציה כמשתנה שיכול לקבל את אחד מן הערכים $\{0,1,2\}$ (משתנה בבסיס 3).

נגדיר מוצא אוקטלי של פונקציה כמוצא שיכול לקבל אחד מתוך הערכים $\{0,1,2,3,4,5,6,7\}$ עבור קלט נתון (מוצא בבסיס 8).

נסמן ב-f פונקציה בעלת n משתני כניסה טרינארים ויציאה אוקטלית אחת:

$$f:\{0,1,2\}^n \to \{0,\dots,7\}$$

נגדיר פונקציה קבועה כפונקציה שמוצאה זהה עבור כל קלט.

כמה פונקציות שונות f שאינן קבועות קיימות?

א. 3^{3^n}

ב. $3^{8^n} - 3$

ג. $3^{8^n} - 8$

ד. $8^{3^n} - 3$

ה. $8^{3^n} - 8$

פתרון:

כיוון שמשתני הכניסה הם טרינארים, מספר אפשרויות הקלט השונות הוא:

$$\underbrace{3 \cdot 3 \cdot 3 \dots \cdot 3}_{n\ times} = 3^n$$

יציאה הפונקציה היא אוקטלית, כלומר היא יכולה לקבל 8 ערכים שונים עבור כל קלט. לכן מספר הפונקציות הכולל הוא:

$$\underbrace{8 \cdot 8 \cdot 8 \dots \cdot 8}_{3^n\ times} = 8^{3^n}$$

כיוון שקיימים 8 ערכי מוצא אפשריים, קיימות 8 פונקציות קבועות. לכן התשובה הסופית היא:

$$8^{3^n} - 8$$

תשובה ה'.

שאלה 3 (5 נקודות):

ממשו את הפונקציה $F(x,y,z)=xz'+yz'$ בעזרת שערי *NOR* עם 2 כניסות בלבד.

מהו מספר שערי *NOR* המינימלי שיש צורך להשתמש בהם?

א. 1

ב. 2

ג. 3

ד. 4

ה. 5

פתרון:

2 שערים

$xz'+yz'=(x+y)z'=(x+y)''z'=\left(nor(x,y)\right)'z'=nor((nor(x,y),z)$

שאלה 4 (5 נקודות):

נתונה הפונקציה הבאה:

$$f(a,b,c,d,e) = \sum(6,7,22,23,31) + \sum_{\phi}(18,30)$$

סטודנט בקורס מעוניין לממש את הפונקציה.

לרשות הסטודנט יש שער *OR* יחיד עם 32 כניסות, הקבועים '0' ו- '1', ובנוסף רכיבים כמתואר בטענות הבאות:

טענה 1: ניתן לממש את הפונקציה בעזרת מפענח *3->8* ושער *AND* יחיד בעל שתי כניסות בלבד.

טענה 2: ניתן לממש את הפונקציה בעזרת מפענח *4->16* יחיד, וללא שערים לוגים נוספים בלבד.

טענה 3: ניתן לממש את הפונקציה בעזרת מפענח *5->32* יחיד בלבד.

כאשר לכל המפענחים יש כניסת *Enable* כנלמד.

הערה: בתשובות להלן כאשר מצוין ששתי טענות נכונות, הכוונה היא שכל אחת מהן נכונה בנפרד.

סמנו את התשובה הנכונה ביותר:

א. כל הטענות **לא** נכונות.
ב. טענה 3 נכונה, וטענות 1 ו- 2 לא נכונות.
ג. טענות 2 ו- 3 נכונות, וטענה 1 לא נכונה.
ד. טעות 1 ו- 3 נכונות, וטענה 2 לא נכונה.
ה. כל הטענות נכונות.

פתרון:

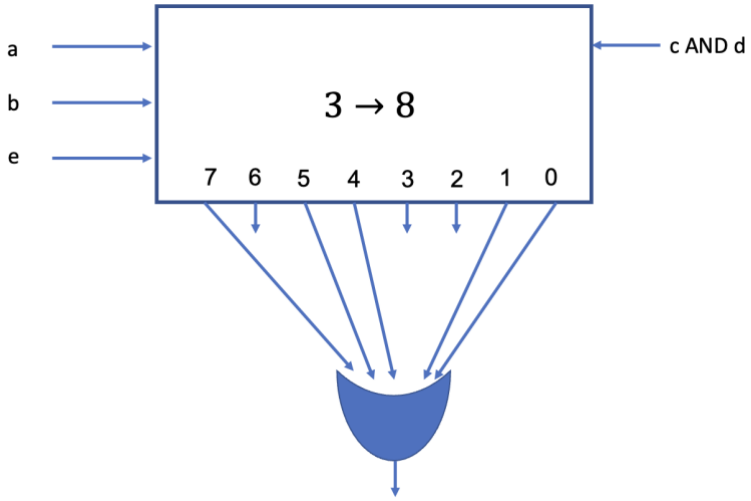
תשובה ה- כל הטענות נכונות.

טבלת אמת של הפונקציה רק במקומות בהם הפונקציה שווה ל- 1 או *DC*:

abcde	*f*
00110	*1*
00111	*1*

10010	dc
10110	1
10111	1
11100	dc
11111	1

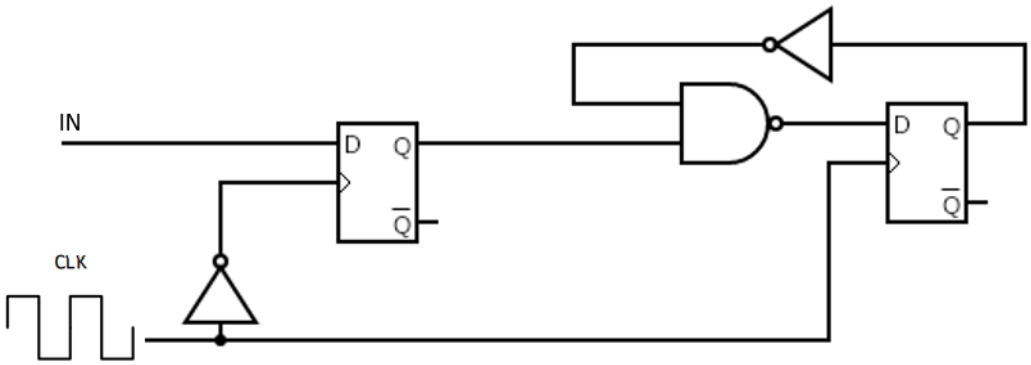
מכיוון שאנחנו מעוניינים להשתמש במספר קטן ככל האפשר של *Decoders*, ושיהיה קטנים ככל האפשר נשים לב שגם עבור הקלט *c* והקלט *d* שווים ל- '1' בכל מקום שבו הפונקציה מוציאה '1'. על מנת לממש את המעגל בעזרת דקודר קטן ככל האפשר נכניס בכניסת ה- *enable* שלו את הפעולה הלוגית *c AND d*, שמטרתה לבדוק האם *c* וגם *d* שווים ל- 1. במצב זה אנחנו מתייחסים ל- *DC* כאל 0.



באותו אופן, כפי שראינו בתרגול, ניתן לבחור את אחת הרגליים *c* או *d* ולהשתמש בה כרגל ה- *enable* של מפענח 4 ל- 16.

שאלה 5 (5 נקודות):

נתון המעגל הבא:



שימו לב שהשערים הלוגים בשרטוט הינם *NAND* ו- *NOT* בלבד.

זמני ההשהיה של הרכיבים מופיעה בטבלה הבאה (נתונים ב ns):

	$t_{pd}/t_{pC\to Q}$	t_{setup}
FF	10	***X***
NOT	2	-
NAND	5	-

נתון ש- $t_{cd}(NOT) = t_{pd}(NOT)$.

זמן המחזור של המעגל הינו $\boldsymbol{T = 50ns}$, כאשר הזמן שבו השעון שווה '1' שווה לזמן שבו השעון שווה '0' (*Duty Cycle = 50%*).

הניחו כי הכניסה *IN* עומדת בתנאי *setup* ו-*hold*, ותנאי *hold* במערכת מתקיימים.

מהו ערכו המקסימלי האפשרי של *X* (זמן ה*setup* של ה *FF*) כך שהמערכת עומדת בתנאי התזמון?

א. $5ns$

ב. $8ns$

ג. $17ns$

ד. $33ns$

ה. המערכת לא יכולה לעמוד בתנאי *setup* ללא תלות בערכו של *X*.

פתרון:

תשובה ב'.

נשים לב שיש שער *NOT* בכניסת השעון של ה *FF* השמאלי.

כלומר הוא יראה עליית שעון $25ns$ אחרי עליית השעון (בירידת השעון).

לכן זמן המחזור האפקטיבי יהיה מחצית מזמן המחזור

FF1->FF2:

$$T_{pd}(NOT) + T_{pd}(FF) + T_{pd}(NAND) + T_{su}(FF) \leq \frac{T}{2}$$

$$2 + 10 + 5 + X \leq 25$$

$$X \leq 8$$

FF2->FF2:

$$T_{pd}(FF) + T_{pd}(NOT) + T_{pd}(NAND) + X \leq T$$

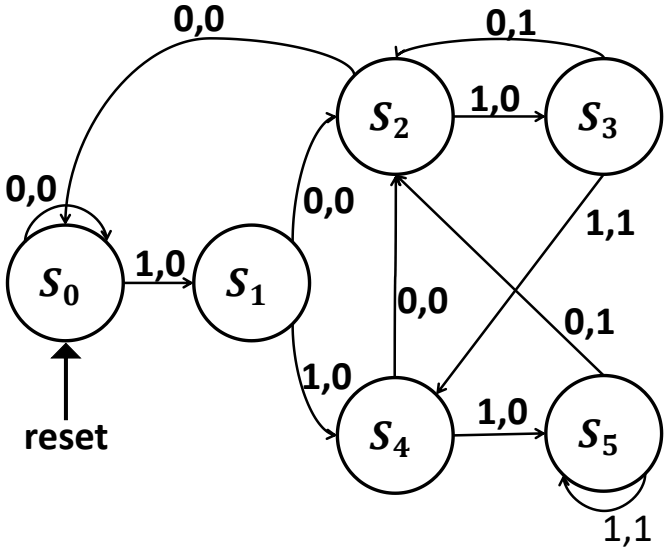
$$10 + 2 + 5 + X \leq 50$$

$$X \leq 33$$

שאלה 6 (5 נקודות):

נתונה דיאגרמת המצבים של מערכת *Mealy* הבאה.

הסימון על החיצים הינו: $input, output$.



קלט המערכת עבור כל מחזור הינו:

cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
input	1	0	1	1	1	1	1	0	1	0	1	0	1	1	1	0

בתחילת מחזור *0* מכונת המצבים נמצאת במצב S_0.

מה יהיה פלט המערכת (הסיבית השמאלית ביותר הינה המוצא במחזור מספר 0)?

א. 0001010100001000

ב. 0001001100010110

ג. 0001011101010101

ד. 0001010101001001

ה. 1001000101010101

פתרון:

תשובה ג'.

ישנם 2 דרכים להגיע לפתרון. הראשון הוא לשים לב כי המערכת פולט 1 אם ה-4 תווים האחרונים הם 1010,1011,1111,1110. נשים לב כי 6 הספרות האחרונות הם 101110, ולכן הסיומת חייבת להיות 101. בנוסף, התו הראשון של הפלט הוא 0, מה שמשאיר רק את תשובה ג'.

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
input	1	0	1	1	1	1	1	0	1	0	1	0	1	1	1	0
output	0	0	0	1	0	1	1	1	0	1	0	1	0	1	0	1

לחלופין, ניתן לעקוב אחריי המכונה:

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
input	1	0	1	1	1	1	1	0	1	0	1	0	1	1	1	0
State	1	2	3	4	5	5	5	2	3	2	3	2	3	4	5	2
output	0	0	0	1	0	1	1	1	0	1	0	1	0	1	0	1

ולכן התשובה ג'

<u>שאלה 7 (5 נקודות):</u>

מהנדס חשמל מעוניין לממש שתי מערכת עקיבה מסוג מילי בעלות סיבית כניסה אחת וסיבית מוצא אחת. שתי המערכות מתייחסות לרצף הסיביות שנקלטו עד כה כמספר בינארי X (למשל אם סדרת הקלט הינה 0101, אז $X=5$).

<u>מערכת א':</u> מתייחסת לסיבית <u>הנקלטת בכל מחזור</u> כסיבית ה- *<u>LSB</u>* של המספר X (כלומר הסיבית הראשונה שנקלטה הינה סיבית ה- *MSB* של המספר).

<u>מערכת ב':</u> מתייחסת לסיבית <u>הנקלטת בכל מחזור</u> כסיבית ה- *<u>MSB</u>* של המספר X (כלומר הסיבית הראשונה שנקלטה הינה סיבית ה- *LSB* של המספר).

שתי המערכות מוציאות 1 אמ"מ המספר שהתקבל עד כה מתחלק ב- 16 ללא שארית.

<u>הערה</u>: שתי המערכות מאותחלות, ומקבלות במהלך המחזור הראשון את הקלט הראשון.

ממשו את מכונות המצבים כך שיכילו מספר מצבים מינימלי אפשרי, וסמנו את התשובה הנכונה ביותר:

א. אחת מהמכונות לא ניתנות למימוש כמכונת מצבים סופית.

ב. שתי המכונות לא ניתנות למימוש כמכונת מצבים סופית.

ג. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות ומספר המצבים של מכונה א' גדול ממספר המצבים של מכונה ב'.

ד. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות ומספר המצבים של מכונה א' שווה למספר המצבים של מכונה ב'.

ה. שתי המכונות ניתנות למימוש כמכונות מצבים סופיות ומספר המצבים של מכונה א' קטן ממספר המצבים של מכונה ב'.

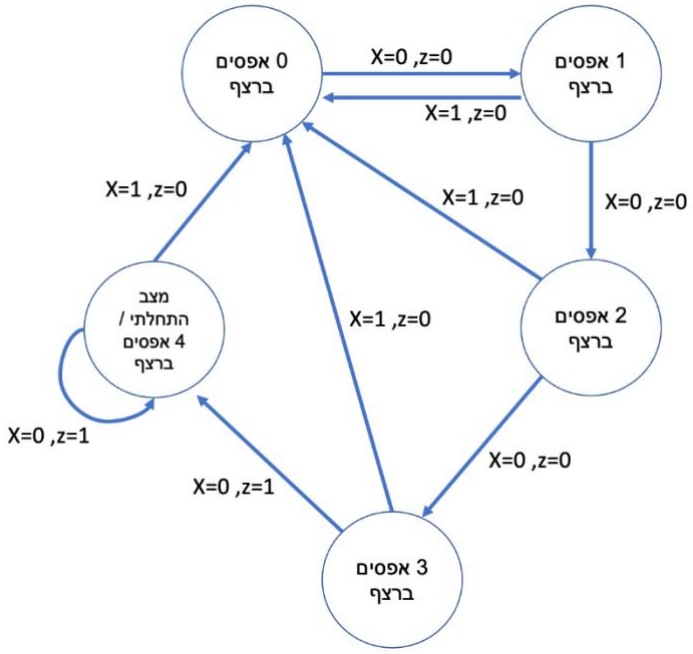
פתרון:

תשובה ה'.

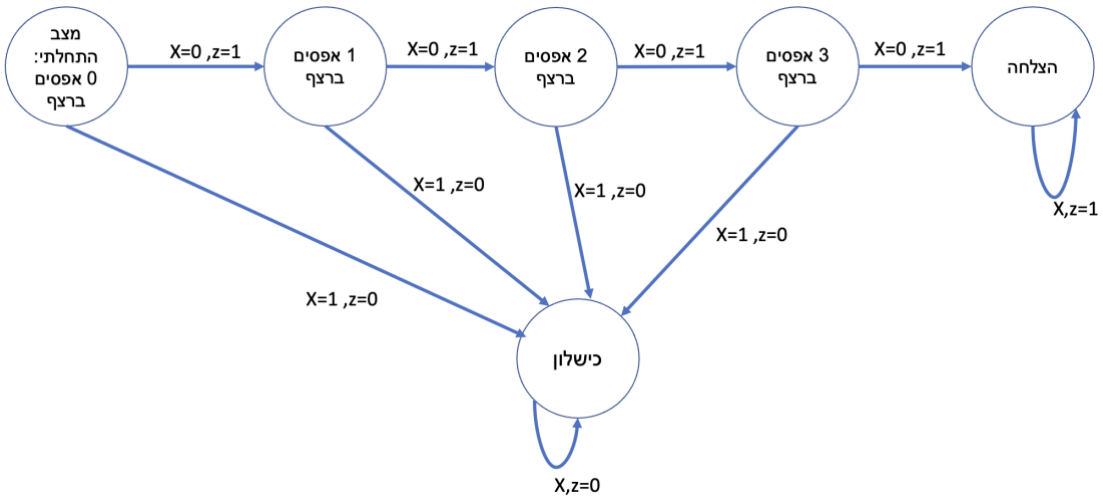
ראשית נשים לב שמספר בינארי שמסתיים ב- 4 סיביות רצופות שהן '0' מתחלק ב- 16.

מערכת א' צריכה לספור האם התקבלו לפחות ארבעה אפסים ברציפות ולהוציא 1 בהתאם, או להוציא 1 אם עד כה התקבלו רק אפסים ועדיין לא התקבל 1.

ניתן לבצע זאת באופן הבא:



עבור מערכת ב' נשים לב שאנחנו מעוניינים רק ב-4 מחזורים הראשונים. לאחר מחזורים אלו לא ניתן יהיה לשנות את מוצא המערכת:



<u>שאלה 8 (5 נקודות):</u>

נתון מעבד *Single Cycle RISC-V*.

מהנדסת מעוניינת להריץ את הפקודה *BeqMemReg* על המעבד כפקודה אמיתית (לא פסאודו פקודה).

פורמט הפקודה הינו:

BeqMemReg rs1, rs2, imm

הפקודה בודקת האם *Mem[Reg[rs1]+imm] = Reg[rs2]*. אם כן, אז הקפיצה תתבצע לפי *PC=Reg[rs2]*, ואם לא אז *PC=PC+4*.

בחרו את התשובה <u>המחמירה</u> ביותר שמאפשרת עבודה תקינה של המעגל עבור הפונקציה המוצעת, ולא פוגעת בביצוע שאר הפונקציות שנלמדו.

הערה: אם תשובה *X* מוכלת בתוך תשובה *Y*, אז תשובה *X* מחמירה יותר.

א. ניתן לממש את הפקודה ללא שינוי של ה- *Datapath* כלל.

ב. ניתן לממש את הפקודה ע"י חיווט מחדש ב- *Datapath* בלבד.

ג. ניתן לממש את הפקודה ע"י חיווט מחדש ב- *Datapath*, הוספת בורר יחיד, והוספת סיגנלי בקרה בהתאם.

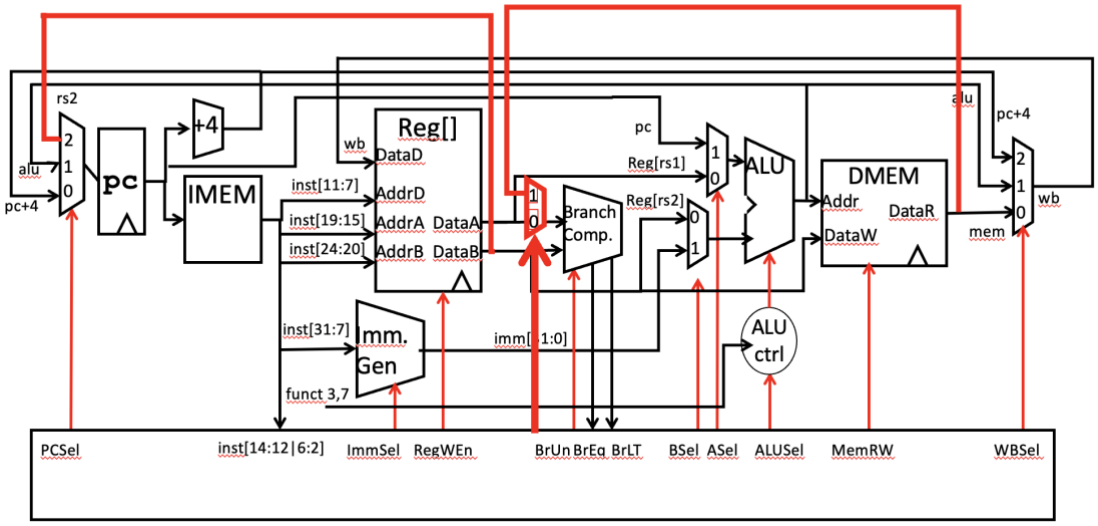
ד. ניתן לממש את הפקודה ע"י חיווט מחדש ב- *Datapath*, הוספת בורר יחיד, הרחבת בורר יחיד והוספת סיגנלי בקרה בהתאם.

ה. כל התשובות אינן נכונות.

פתרון:

תשובה ד'.

מימוש אפשרי הינו:



צריך להוסיף בורר יחיד, ולהרחיב בורר יחיד. ע"י חיווט מחדש והוספת סיגנלי בקרה בהתאם ניתן לממש את הפונקציה.

שאלה 9 (5 נקודות):

מעוניינים להוסיף מימוש של הפקודה *RMW* במעבד *multicycle RISC-V*.
פקודה זו קוראת מילה מהזכרון בכתובת ששמורה ברגיסטר *rs1* ושומרת אותה ברגיסטר *rd*. לאחר מכן כותבת מילה מרגיסטר *rs2* לזכרון לאותה הכתובת.
ניתן לבצע שינויים במעבד שכוללים הוספה של חיווטים והרחבה או הוספה של *MUXes*.

פקודה זו בעלת הפורמט:

rmw rd, rs1, rs2

המבצעת את הפעולות הבאות:

reg[rd] ← Mem[reg[rs1]]

Mem[reg[rs1]] ← reg[rs2]

הערה: רגיסטר *rd* שונה מרגיסטרים *rs1* ו- *rs2* (הכוונה למספר הרגיסטר ולא לתוכנו).

מה מספר המחזורים המינימלי הנדרש לביצוע פקודה זו?

א. 4

ב. 5

ג. 6

ד. 7

ה. 8

פתרון:

4 מחזורים.

מחזור 3 גישה לזכרון לקריאה.

מחזור 4 שמירה ברגיסטר *rd*, וכתיבה לזכרון.

הוחלט לקבל כאן גם את תשובה ב' – 5 מחזורים, משום שבשאלה נכתב "לאחר מכן" וחלק מהסטודנטים הבין שלא ניתן למקבל כתיבה לרגיסטר וכתיבה לזיכרון משום כך.

<u>שאלה 10 (5 נקודות):</u>

התוכנית הבאה מורצת במעבד RISC-V Pipelined בעל הנתונים הבאים:

- כולל יחידת Forwarding <u>מלא</u>, כלומר בשלב ה-EXE כל רכיב יכול לקבל את ערכו העדכני ביותר של כל רגיסטר. ה- *Forwarding* מתבצע בין השלבים: $WB \rightarrow ID, WB \rightarrow EXE, MEM \rightarrow EXE$.
- לא קיימת יחידת Load Hazard Detection.
- עבור פקודות branch, המעבד מניח שקפיצות אינן נלקחות, ובמידה שכן, **<u>לא</u>** מתבצע Flush ע"י המעבד.
- ההחלטה על ביצוע Branch מתקבלת בשלב ה-EXE.

עליכם לדאוג להוסיף פקודות nop במקומות הדרושים על מנת שהקוד ירוץ כהלכה על המעבד הנתון.

```
1.          add t1, x0, x0
2.          lw t2, 0(t1)        #initialized t2 to be 1
3.          lw t3, 4(t1)        #initialized t3 to be 3
4.          addi t5, x0, 1
5.  Loop:   sub t3, t3, t2
6.          slt  t4, x0, t3
7.          add t10, t4, x0
8.          beq t4, t2, loop
9.          add t5, t10, t5
```

מה מספר ה *nop*-ים <u>המינימלי</u> שצריך להוסיף כך שהקוד ירוץ כמתוכנן?

א. *nop* יחיד.

ב. 2 *nop*-ים.

ג. 3 *nop*-ים.

ד. 4 *nop*-ים.

ה. התוכנית רצה בצורה תקינה בלי *nops*.

<u>פתרון:</u>

התשובה הנכונה הינה 2 פקודות NOP בין פקודות 8 ו- 9.

בין פקודה 1 ל- 2 ו- 3, ה- hazard מטופל ע"י ה- forwarding unit.

בין פקודות 2 ו-3 ל- 5 יש forwarding.

בין 5-6 , 6-7 ו- 6-8 ה forwarding unit מטפלת ב- hazard שנוצר.

בין 8,9 יש control hazard ואם לא שמים nops בין שורות אלו, בשורה 9 ערכו של t5 משתנה בין איטרציות הלולאה ולכן בסוף ביצוע הקוד התוכן שלו לא יהיה לפי כוונת המתכנת בגלל שהוא השתנה תוך הלולאה ואנחנו משתמשים בו בחישוב הסופי.

הוחלט לקבל כאן גם את תשובה ד' – 4 NOPS, משום שלא נכתב באופן מפורש כמה פקודות NOP צריך להוסיף לקוד, ולכן חלק מהסטודנטים הבין שהדרישה בשאלה היא לכמה פקודות NOP ירוצו בסה"כ.

<u>החל מהעמוד הבא מתחיל החלק של השאלות פתוחות</u>

<u>(שאלות 11 – 15)</u>

שאלה 11 (10 נקודות):

מהנדס בונה פונקציה עבור משחק. הקלט לפונקציה הינו מספר דו-ספרתי בבסיס 4 בין $(1)_4$ ל- $(33)_4$, כולל (כלומר הקלט $(0)_4$ לא יכול להתקבל).

הפונקציה מוציאה 1 אם המספר מתחלק ב-3 או מכיל את הספרה 3.

לדוגמא, אם הקלט הוא $(12)_4$ הפלט הוא 1 (כי $(12)_4 = (6)_{10}$ מתחלק ב3).
אם הקלט הוא $(13)_4$ הפלט הוא 1 (כי אחת הספרות היא 3).

המספר מתקבל כקלט בינארי $(a_3a_2a_1a_0)_2$, כאשר a_1a_0 היא מייצגת את הספרה הימנית של המספר בבסיס 4, וa_3a_2 הינה הספרה השמאלית.

סעיף א' – מלאו את מפת הקרנו הבא בהתאם לפונקציה הנדרשת:

a_1a_0 \ a_3a_2	00	01	11	10
00				
01				
11				
10				

סעיף ב' – צמצמו את הפונקציה כסכום מכפלות. מה הביטוי המינימלי?

$f(a_0, a_1, a_2, a_3) =$

פתרון:

נשים לב כי 0 זהו קלט לא חוקי למערכת, ולכן נשים את הפלט של הכניסה המתאימה כ-*don't care*.

a_1a_0 \ a_3a_2	00	01	11	10
00	ϕ	0	1	0
01	0	0	1	1
11	1	1	1	1
10	0	1	1	0

ולכן הפונקציה המינימלית כסכום מכפלות הינה:

$$f(a_0, a_1, a_2, a_3) = a_3a_2 + a_1a_0 + a_3a_0 + a_2a_1$$

שאלה *12* (*10* נקודות):

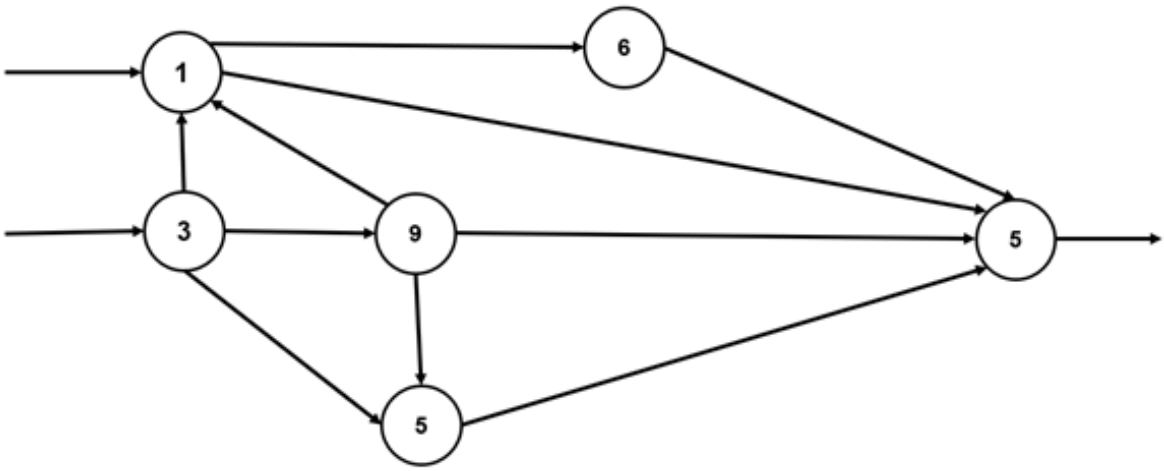
נתונה המערכת הצירופית הבאה. המספרים מציינים את השהית הרכיב בננו שניות.

באפשרותכם להשתמש בשני סוגים של רגיסטרים:

- רגיסטרים אידאלים, $T_{PCQ} = 0ns, T_{setup} = 0ns$, אשר עלותם גבוהה.
- רגיסטרים בעלי $T_{PCQ} = 1ns, T_{setup} = 1ns$, ללא עלות.

הניחו שתנאי *HOLD* מתקיימים במעגל.

עליכם לצנר את המערכת הנתונה כדי לקבל תפוקה מקסימלית בעדיפות ראשונה, השהייה מינימלית בעדיפות שניה ועלות מינימלית בעדיפות שלישית.

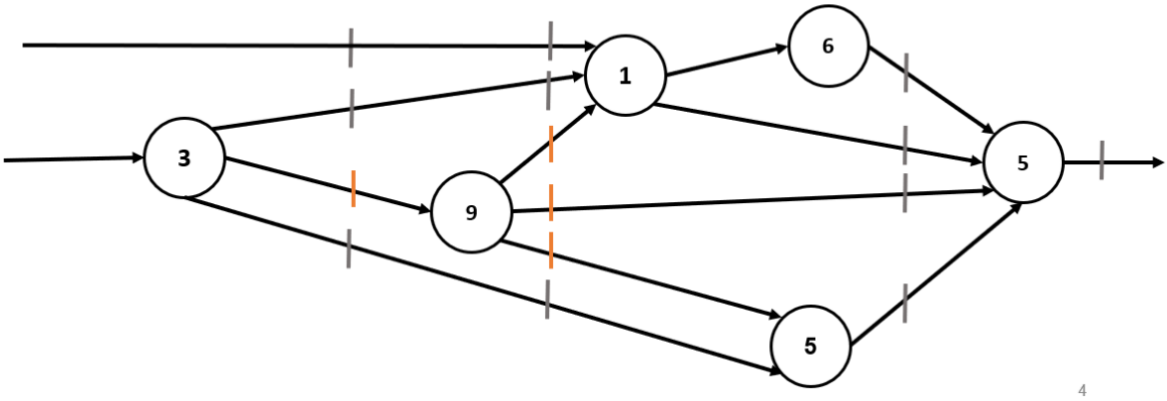


א. מהי התפוקה המקסימלית?

ב. בכמה רגיסטרים אידאליים תשתמשו?

ג. כעת אין אפשרות להשתמש ברגיסטרים אידאליים, אלא רק באלו עם השהייה. מהי התפוקה המקסימלית כעת?

פתרון:



א. 1/9

ב. 4

ג. 1/11

<u>שאלה *13* (10 נקודות):</u>

משדר ומקלט עובדים עפ"י פרוטוקול תקשורת טורית הדומה לזה שנלמד בקורס: תחילה משודרת סיבית התחלה (*Start Bit*), לאחר מכן משודרות X סיביות נתון, ולבסוף סיבית סיום (*Stop Bit*).

ידוע כי זמן מחזור השעון של המשדר הוא 4.8ns, וזמן מחזור השעון של המקלט הוא 5ns. מבחינת המשדר והמקלט, זמן הסיבית (Tbit) הוא חמישים מחזורי שעון פנימיים ($N_T = N_R = 50$).

הניחו כי המקלט מזהה באופן מידי את סיבית ההתחלה (Start Bit), כמו כן נדרש שגם <u>סיבית הסיום תידגם בצורה תקינה.</u>

מהו ה- X (כמות סיביות המידע) המקסימאלי? פרטו את הדרך **בקצרה**.

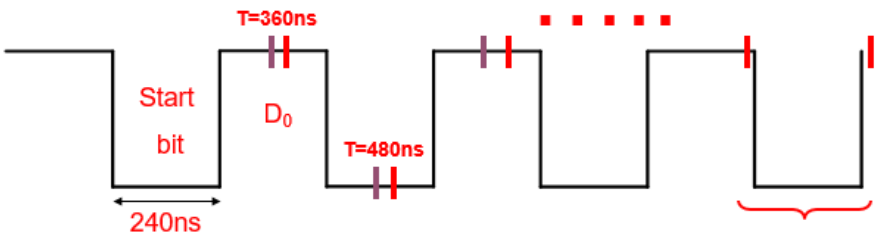
דרך:

תשובה סופית:

פתרון:

באופן דומה לנלמד בתרגול:

ניתוח ראשוני

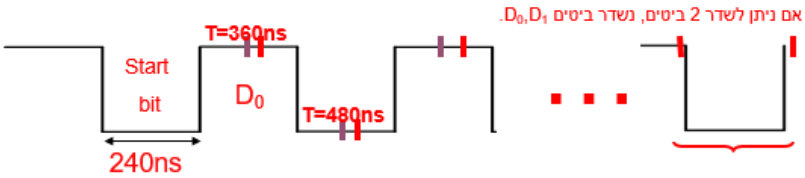


- ישנה סטייה הולכת וגדלה של דגימת המקלט ביחס לאמצע הביט האמיתי.
- החסם על מספר הביטים הניתנים לשידור, ניתן ע"י אותו הביט שבשל הסטייה המצטברת לא ידגם כלל.

אמצע הביט על פי הנחת המשדר. אמצע הביט על פי הנחת המקלט.

X = מספר ביטים הניתנים לשידור ללא תקלות ("דילוג" על ביט), לא כולל start bit. נמספר את הביטים $D_0, \ldots, D_{X-1}$.

לדוגמא: אם ניתן לשדר 2 ביטים, נשדר ביטים D_0,D_1.



The Edges of the last bit (D_{x-1}): [240x,240(x+1)]

T_{bit}(recv)=250ns

Sampling points: 1.5 Tbit(recv)=375ns; 2.5 Tbit(recv)=625ns; .....

The Sampling of the last bit (D_{x-1}):

[1.5 +(x-1)]*250

הדגימה של הביט האחרון צריכה להיות בתוך תחום השידור שלו ולכן:

$$240x < [1.5 + (x-1)]*250 < 240(x+1)$$
$$250x+125<240x+240 \Rightarrow x<11.5$$

תשובה : x=11 כאשר אם נדרש לדגום גם את ה-stop bit הרי שניתן לשדר סה"כ 10 סיביות מידע.

<u>שאלה *14* (10 נקודות):</u>

נתונים מעבדי *single-cycle RISC-V* ו-*multicycle RISC-V* כפי שנלמדו בכיתה.

בנוסף נתון מעבד RISC-V Pipelined בעל הנתונים הבאים:

- כולל יחידת Forwarding <u>מלא</u>, כלומר בשלב ה-EXE כל רכיב יכול לקבל את ערכו העדכני ביותר של כל רגיסטר. ה- *Forwarding* מתבצע בין השלבים: $WB \rightarrow ID, WB \rightarrow EXE, MEM \rightarrow EXE$.
- קיימת יחידת Load Hazard Detection.
- עבור פקודות branch, המעבד מניח שקפיצות אינן נלקחות, ומבצע Flush במידה וכן.
- ההחלטה על ביצוע Branch מתקבלת בשלב ה-EXE.

מעבד ה-*single-cycle RISC-V* עובד עם מחזור שעון T_{single}.

מעבד ה-*multicycle RISC-V* עובד עם מחזור שעון T_{multi}.

מעבד ה-*pipelined RISC-V* עובד עם מחזור שעון $T_{pipelined}$.

נתונה התוכנית הבאה:

0x1AA0 000C		*addi s0, x0, 4*
0x1AA0 0010	*Loop:*	*add s1, s2, s2*
0x1AA0 0014		*lw s3, 4(s4)*
0x1AA0 0018		*add s2, s3, s4*
0x1AA0 001C		*addi s0, s0, -1*
0x1AA0 0020		*bne s0, x0, Loop*
0x1AA0 0024	*Exit*	

ניתן להניח שהתוכנית רצה ללא תקלות וללא חריגות.

א. מה התחום שבו צריך להיות היחס $\frac{T_{single}}{T_{multi}}$ כדי שמעבד ה-*multicycle* יסיים את ריצת התוכנית מהר יותר. רשמו את תשובתכם במרובעים, כאשר המרובע מכיל **מספר** והמרובע השמאלי מכיל **רק** סימן יחס (אחד מן הסימנים הבאים: $<, >, \leq, \geq, =, \neq$).

$\frac{T_{single}}{T_{multi}}$ $\square$ $\square$

ב. מה התחום שבו צריך להיות היחס $\frac{T_{pipelined}}{T_{multi}}$ כדי שמעבד ה-*multicycle* יסיים את ריצת התוכנית מהר יותר. רשמו את תשובתכם במרובעים, כאשר המרובע מכיל **מספר** והמרובע השמאלי מכיל **רק** סימן יחס (אחד מן הסימנים הבאים: $<, >, \leq, \geq, =, \neq$).

$\frac{T_{pipelined}}{T_{multi}}$ $\square$ $\square$

<u>פתרון:</u>

עבור מעבד *single-cycle*, מספר מחזורי השעון הדרושים להרצת התוכנית שווה למספר הפקודות. בתוכנית ישנן 5 פקודות בתוך הלולאה ופקודה נוספת מחוצה לה. לכן מספר הפקודות הכולל יהיה:

$$1 + 4 \cdot 5 = 21\ cycels$$

עבור מעבר ה-*multicycle* מספר מחזורי השעון של המעבד אשר דרושים בכדי להריץ את התוכנית תלוי בהרכב הפקודות. עבור פקודות *r-type* דרושים 4 מחזורים, עבור *lw* 5 ועבור *bne* דרושים 3. סה"כ נקבל:

$$4 + 4 \cdot (4 + 5 + 4 + 4 + 3) = 84\ cycles$$

עבור מעבד ה-*pipeline* נשים לב כי ישנה הכנסה של *nop* בגלל *lw hazard* (קיים *fw* ולכן אין בעיות אחרות. התוכנית החדשה תיראה כך:

0x1AA0 000C		*addi s0, x0, 4*
0x1AA0 0010	*Loop:*	*add s1, s2, s2*
0x1AA0 0014		*lw s3, 4(s4)*
0x1AA0 0018		*nop*
0x1AA0 001C		*add s2, s3, s4*
0x1AA0 0020		*addi s0, s0, -1*
0x1AA0 0024		*bne s0, x0, Loop*
0x1AA0 0028	*Exit*	

הביצוע של כל פקודה לוקח 5 מחזורי שעון, כאשר ישנה חפיפה בין הפקודות לאחר מילוי המעבד. עלינו לזכור כי בכל איטרציה מלבד האחרונה אנחנו מבצעים *flush* ל-2 פקודות. לכן, בסה"כ:

$$4 + 1 + 4 \cdot 6 + 3 \cdot 2 = 35\ cycles$$

עבור סעיף א' דרוש:

$$T_{multi} \cdot 84 < T_{single} \cdot 21$$

$$\frac{T_{single}}{T_{multi}} > \frac{84}{21} = 4$$

עבור סעיף ב' דרוש:

$$T_{multi} \cdot 84 < T_{pipeline} \cdot 35$$

$$\frac{T_{pipeline}}{T_{multi}} > \frac{84}{35} = 2.4$$

שאלה *15* (10 נקודות):

נתון מעבד *pipelined RISC-V* **ללא** מנגנון *forwarding* וללא *hazard detection unit*.

כמו כן נתון הקוד הבא:

1	0*x1AA0 0000*	*main* :	*lw t2, 0(s1)*
2	0*x1AA0 0004*		*sw t2, 4(s1)*
3	0*x1AA0 0008*		*addi t2, a0, -1*
4	0*x1AA0 000C*		*addi t1, t2, 0*
5	0*x1AA0 0010*		*add a1, t2, t2*

סעיף א' -

עליכם למלא בטבלה הבאה את כמות פקודות ה- NOP **המינימאלית** הנדרשת לריצה תקינה של הקוד על המעבד הנתון.

<u>לדוגמה</u>: אם לדעתכם צריך להוסיף 3 פקודות NOP בין פקודות 5 ל- 6 אז תמלאו באופן הבא:

כמה NOP להוסיף	*אחרי פקודה מספר*	*לפני פקודה מספר*
3	*5*	*6*

מלאו את הטבלה הבאה (אין הכרח למלא את כולה):

כמה NOP להוסיף	*אחרי פקודה מספר*	*לפני פקודה מספר*

סעיף ב' -

כעת הוחלט להוסיף forwarding מסוג MEM->EXE.

מלאו את הטבלה הבאה איך תשתנה כמות פקודות ה- NOP המינימלית הנדרש לריצה תקינה של המעבד הנתון לאחר השינויים שנוספו.

כמה NOP להוסיף	*אחרי פקודה מספר*	*לפני פקודה מספר*

<u>פתרון:</u>

סעיף א':

```
0x1AA0 0000   main :   lw t2, 0(s1)
                       NOP
                       NOP
                       NOP
0x1AA0 0004            sw t2, 4(s1)
0x1AA0 0008            addi t2, a0, -1
                       NOP
                       NOP
                       NOP
0x1AA0 000C            addi t1, t2, 0
0x1AA0 0010            add a1, t2, t2
```

כלומר יש להוסיף 6 פקודות NOP בכדי לאפשר ריצה תקינה של התוכנית.

<u>סעיף ב':</u>

0x1AA0 0000	main :	lw t2, 0(s1)
		NOP
		NOP
		NOP
0x1AA0 0004		sw t2, 0(s1)
0x1AA0 0008		addi t2, a0, -1
0x1AA0 000C		addi t1, t2, 0
		NOP
		NOP
0x1AA0 0010		add a1, t2, t2

כלומר חסכנו פקודת *NOP* אחת באמצעות הוספת המנגנון.