אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

מבוא ל-VLSI ומעגלים משולבים

361-1-3701

עבודת מספר 3

Verilog-A

ארד וזאני 207898180

עומר אורן 208948794

מספר משתמש – 34 (stu34)

רישיון קבוצה – 23

תאריך הגשה – 27.5.2025

פרמטר G = מספר הקבוצה = 34

בעבודה זו נתכנן רכיבים שונים בעזרת שפת תיאור החומרה Verilog-A, בהתבסס על המדריך שקיבלנו.

לאחר כתיבת הקוד ובדיקה שלו, ניצור סימבול של הרכיב ובכך נאכל לבדוק אותו בtb כמו שעשינו בעבודות קודמות, כלומר תחילה נתאר בעזרת קוד את הפורטים של הרכיב וכמובן את פונקציונליות שלו ואז נבדוק האם הוא עובד כראוי בעזרת בניית tb תחילה ע"י יצירת קובץ schematic המכיל את הרכיב ואת התכונות הנדרשות לפורטים השונים, משם נעבור ליצירת קובץ maestro לשם יצירת סימולציה ספציפית, הגדרת המשתנים המתאימים וצפייה בתוצאות.

בעבודה זו- VDD = 1.5 + 6*0.001*G = 1.704

1. רכיב Sample and Hold –

רכיב זה דוגם בעת עליית שעון את האות האנלוגי הנכנס, שומר על הערך למשך מחזור השעון עד לעליית השעון העוקבת שם ידגום שוב את הכניסה, יחזיק את הערך וחוזר חלילה.

1.1. להלן הקוד המתאר את הרכיב ואופן פעולתו –

```
// VerilogA for MyLIB, EX3p1_SH, veriloga

`include "constants.vams"
`include "disciplines.vams"

module sample_and_hold(Vin, Vout, clk);
  input Vin, clk;
  output Vout;

  electrical Vin, Vout, clk;
  parameter real vdd = 1.5 + 6*0.001*34;
  real sampled_val;

  analog begin
    @(initial_step) sampled_val = 0.0;

    @(cross(V(clk) - vdd, +1)) begin
         sampled_val = V(Vin);
    end

    V(Vout) <+ transition(sampled_val, 0, 10n);

  end
endmodule
```

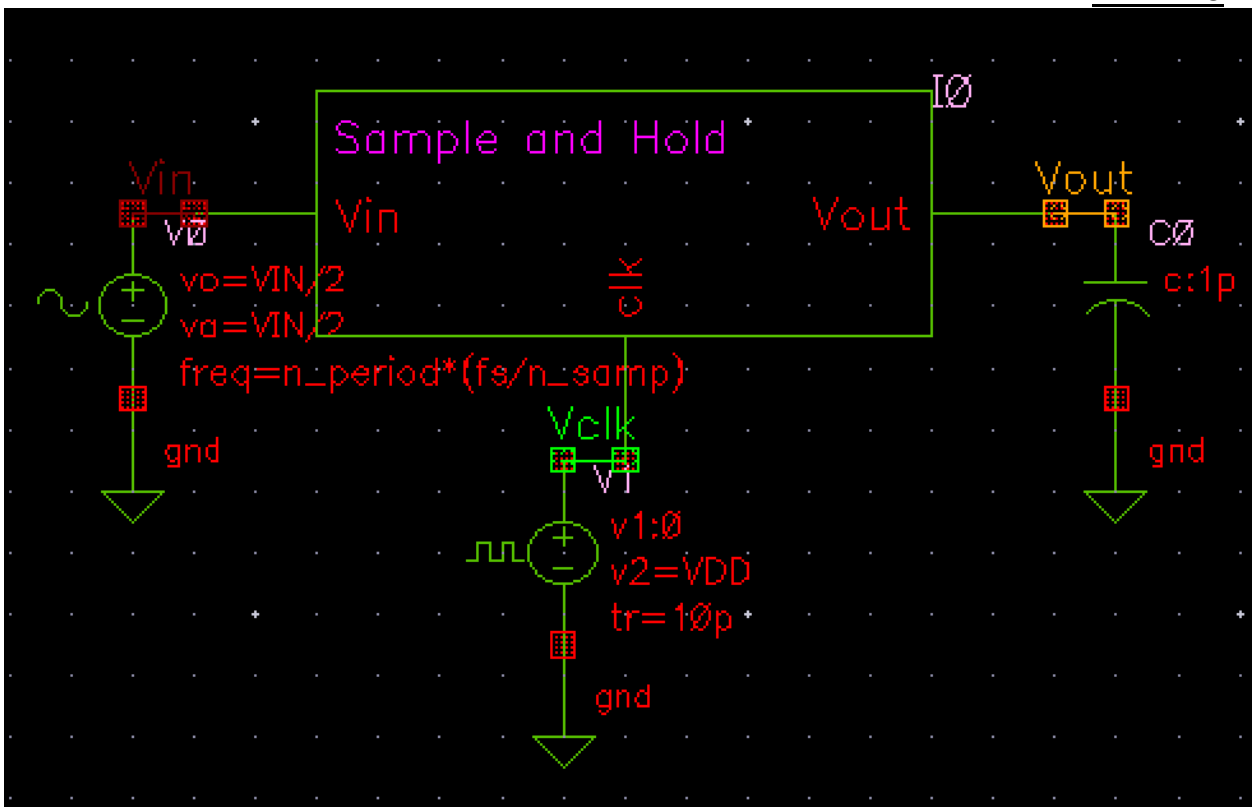
על פי המדריך, הוספנו את הספריות המתאימות לתיאור התנהגות אנלוגית בתוכנה שלנו והגדרנו את הרכיב עם כניסות של מתח הכניסה האנלוגי והשעון, היציאה היא האות שנדגם ומוחזר באותו ערך למשך מחזור שעון אחד, הגדרנו פרמטר של מתח הספק לשם בדיקה של עליית שעון.

כפי שרואים הבלוק האנלוגי מתחיל באתחול בעזרת הפקודת @initial_step שמבצעת את הפקודה שאחריה רק בתחילת סימולציה, ההתנהגות של הבלוק מוגדרת אחרי.

הגדרנו לרכיב את זמן עליית שעון בעזרת פקודת @cross שיוצרת בלוק אשר יתבצע רק כאשר הביטוי בפנים שווה ל0 מלמעלה (1+) כלומר רק בעליית שעון (שמתח הכניסה של השעון משתווה למתח הספק), נשמור את ערך הכניסה הרגעי בעת עליית השעון במשתנה זמני.

אבל כפי שראינו גם במדריך sample_val אינו רציף בצורה הזאת כיוון שבכל עליית שעון הערך שלו "קופץ" לערך אחר וזה לא התנהגות אנלוגית ולכן נצתרך למדל לסימולטור התנהגות אנלוגית ביציאה גדי לקבל תוצאות נכונות ומדויקות אז לשם הרציפות נחליק את ערך המשתנה הזמני בעזרת פונקציית transition שמעבירה את הערך למוצא במשך 10n שניות (השהייה).

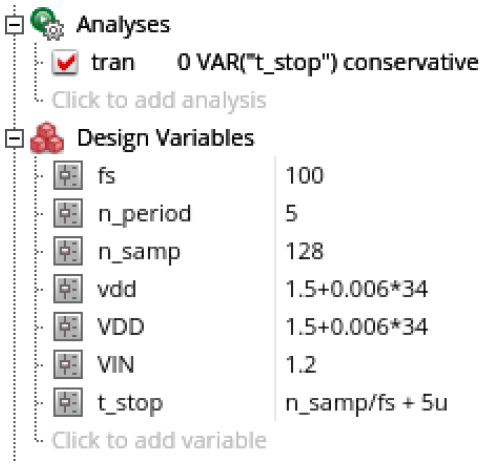
1.2. <u>סימבול –</u>



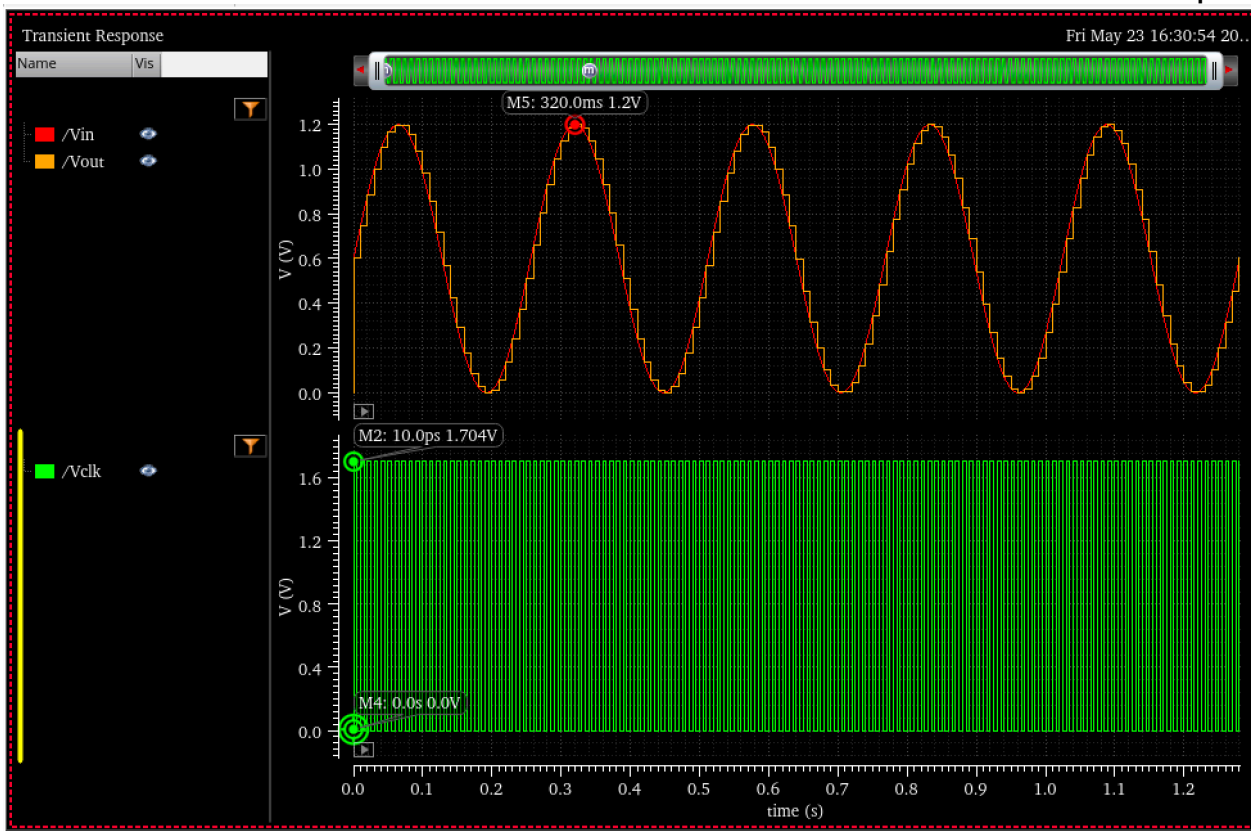
הסימבול של הרכיב הוא המלבן במרכז עם השם המתאים. התמונה מראה את סביבת הTB הכוללת.
כמו שתיארנו בקוד קיבלנו שתי כניסות אחד של השעון והשנייה היא המתח הנדגם, ביציאה נקבל את ערכי הדגימה מוחזקים למשך מחזור אחד.

1.3. <u>נעשה TB לרכיב –</u>

נגדיר את הסביבה של הרכיב על מנת לבדוק אותו כפי שהוגדר בתיאור השאלה- כניסת שעון מסוג אות ריבועי עם תדר 100hz ומתח כניסה סינוסי (אנלוגי) עם תדר כניסה שייתן 128/5 דגימות של אות הכניסה במחזור אחד שלו במוצא נשים קבל קטן למדידה.
כעת נעבור למאסטרו ונבצע סימולציית transient למשך קצת יותר ממחזור אחד של אות הכניסה –

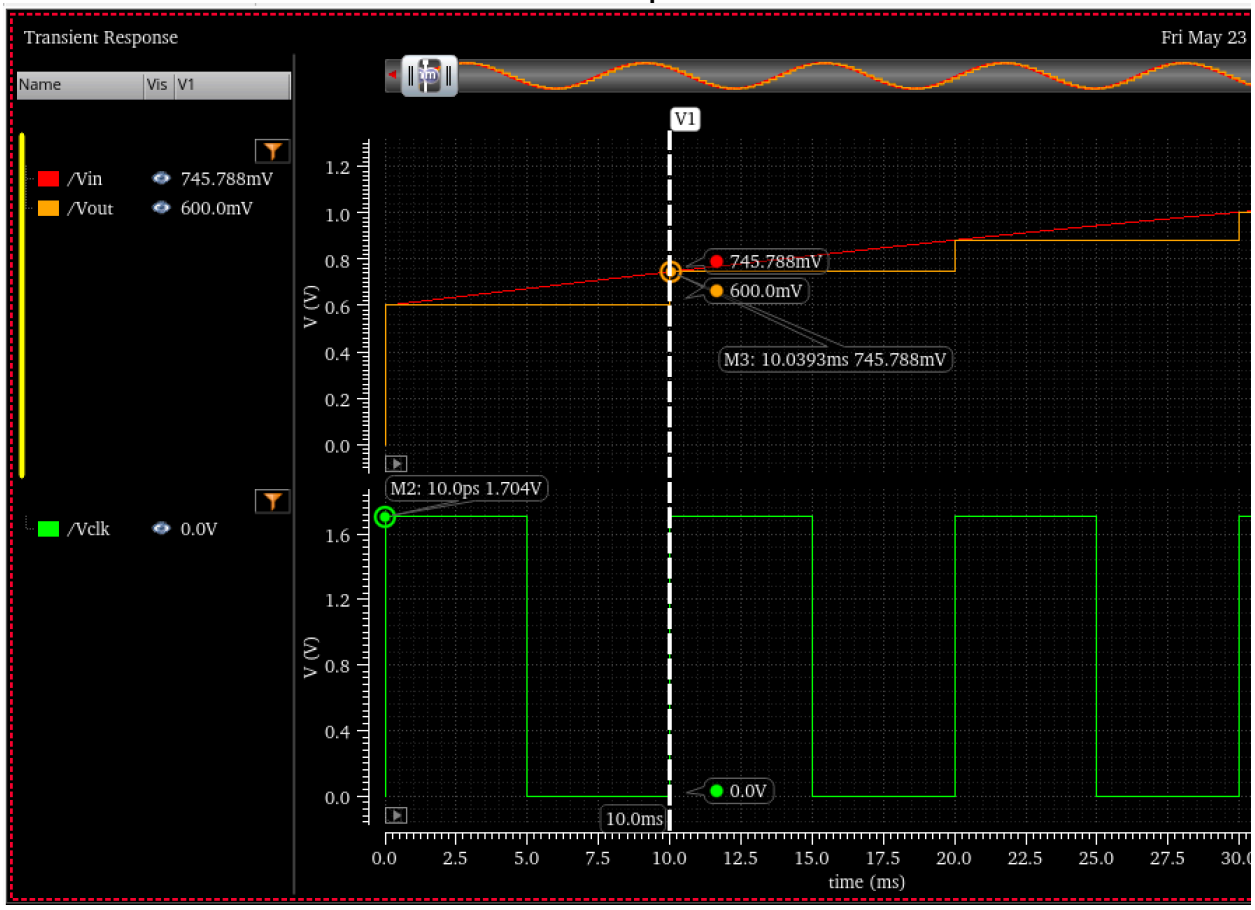


נגדיר כמו כן את המשתנים של הסביבה הכוללת וגם נשים לב שקיבלנו את פרמטר הספק שהוגדר בקוד – vdd .
נריץ ונסתכל על מתחי הפורטים של הרכיב.



1.4. <u>האם הרכיב עובד כראוי?</u>

כן, נשים לב שאות המוצא מחזיק בערך אות הכניסה הנתון בזמן עליית השעון וכי ישנה השהייה מינימלית מרגע עליית השעון ועד הדגימה בפועל –



כמו שניתן לראות בתמונה שמנו מרקר ארוך (v1) שמראה ברגע עליית השעון את הערכים בשעון, בכניסה וביציאה. וניתן לראות כי מתח היציאה 600mv שנשמר עדיין מהמחזור הקודם בעוד הכניסה המשיכה לעלות.
במרקר (m3) לאחר 0.0393ms רואים את הערך החדש מגיע למוצא אשר ישמר למשך זמן המחזור שנותר.
כמו כן ההשהיות גם כתוצאה מפונקציית הtransiant שמדגימה התנהגות ריאלית וטבעית יותר לסימולטור. לכן הרכיב עובד כראוי.

2. <u>רכיב Comparator –</u>

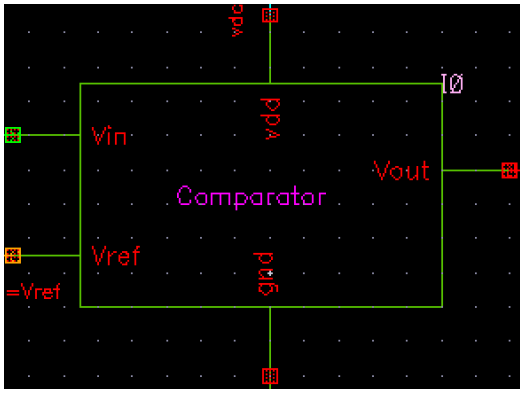
רכיב אסינכרוני המקבל ערך כניסה כלשהו ומשווה אותו למתח ייחוס, במוצא נקבל מתח גבוה (vdd) אם מתח הכניסה גבוה ממתח הייחוס ו0 אחרת.
מתח הייחוס (reference) הוא 34*10*0.001= 0.34v

2.1. קוד –

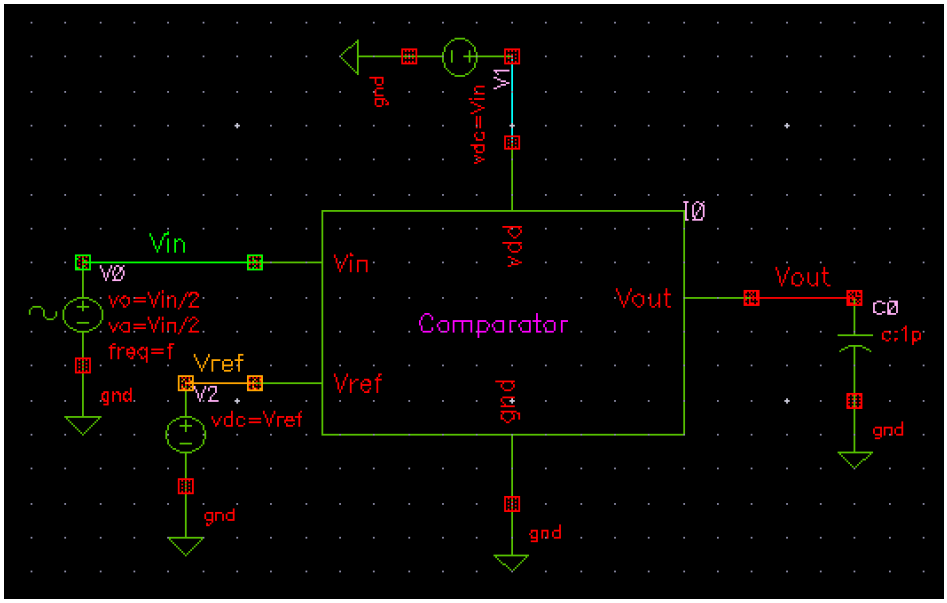
```
// VerilogA for MyLIB, EX3p2_comp, veriloga

`include "constants.vams"
`include "disciplines.vams"

module comparator(Vin, Vout, Vref, vdd, gnd);
  input Vin, Vref, vdd, gnd;
  output Vout;

  electrical Vin, Vref, Vout, vdd, gnd;
  analog begin
    V(Vout) <+ transition(
                  V(Vin) > V(Vref) ? V(vdd) : V(gnd),
                  0, 10n
              );
  end
endmodule

```

בכניסה כפי שמתואר בשאלה הגדרנו מתח כניסה (הנבדק) מתח יציאה שיציין את תוצאת ההשוואה, מתח ייחוס שהמתח הנבדק ישווה מולו. מתח ספק ואדמה שימותגו למוצא בהתאם לתוצאת ההשוואה.
גם בשאלה זו השתמשנו בפונקציה שתכניס השהייה הממדלת מצב ריאלי במוצא ותשמור על רציפות של האות האנלוגי.

2.2. סימבול –

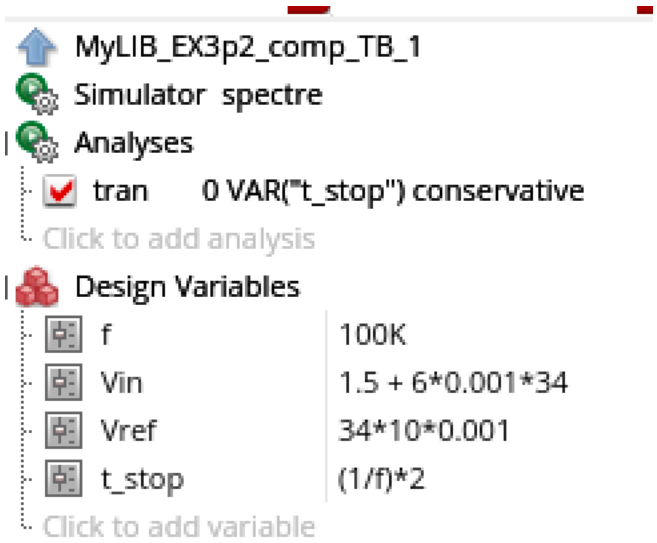


2.3. TB –



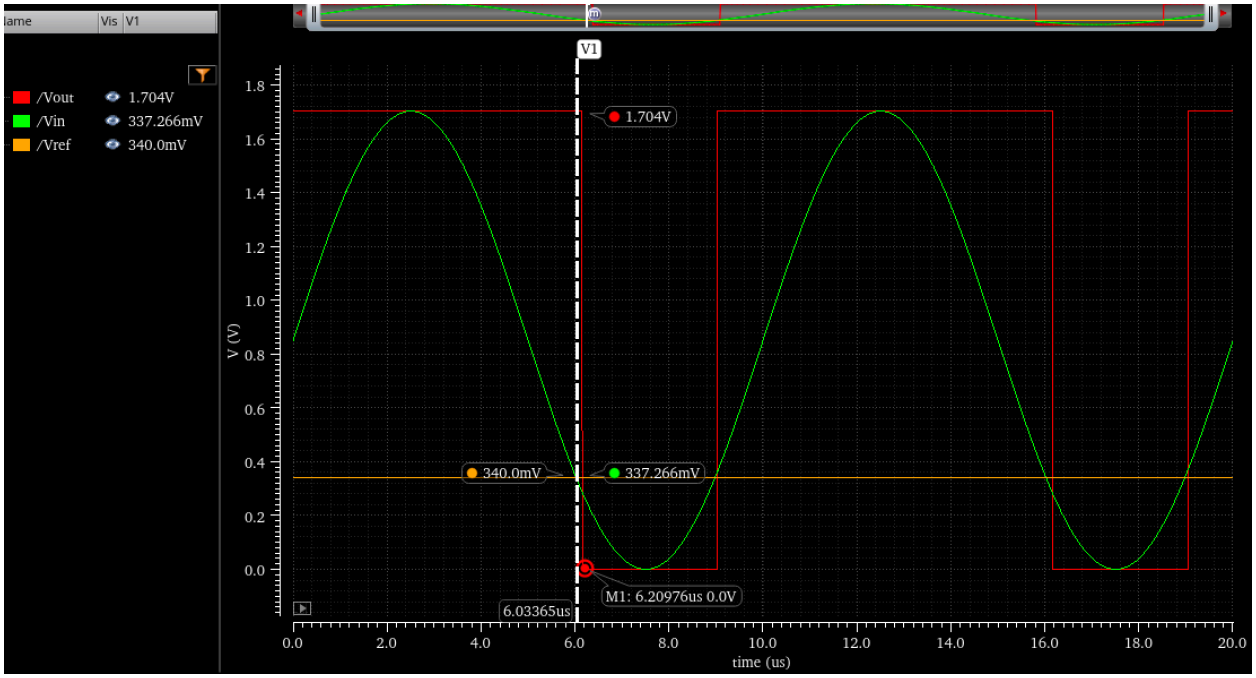
נבצע את הTB כפי שהוגדר בשאלה עם מתח כניסה סינוסי בין מח ספק למתח אדמה בתדר 100khz נחבר את האדמה והספק לרכיב ונקבע מתח רפרנס.

נבצע אנליזת trans עבור שתי מחזורים של מתח הכניסה הסינוסי-



נצפה לראות מעיין גל ריבועי במוצא כאשר רוב הזמן הוא יהיה ב ״1״ כיוון שמתח הייחוס נמוך ממתח הכניסה ברוב הזמן מחזור.

2.4. <u>האם הרכיב מתנהג כפי שציפיתם -</u>



כן, כפי שרואים בצהוב זה מתח הייחוס ובירוק הכניסה, באדום רואים כפי שציפינו שבכל הזמן שהאות בירוק מעל האות בצהוב האות האדום בגבוה ואחרת הוא באפס.
ניתן לראות שבמרקר v1 מסומן כי הירוק קטן מהצהוב אבל האדום עדיין בגבוה אח לאחר 0.2 מיקרו שניות (m1) היציאה יורדת וזה מאחר ותיארנו לסימולטור התנהגות טבעית יותר ורציפה במעבר החד של היציאה בין 1 ל0.

3. רכיב (DAC) Digital to Analog Converter –

רכיב סינכרוני זה מקבל בכניסה וקטור בעל 8 ביטים ומוציא רמת מתח אנלוגי במוצא, הוא ממיר מספר בינארי. כלומר מקשר בין ערך דיגיטלי (בדיד) כלשהו לרמת ערך אנלוגי.
מספר הביטים בכניסה יגדירו את הערך שכל רמה תקבל –
יש לנו 8 ביטים שזה אומר 256 רמות דיגיטליות שונות לכן כל רמה מובדלת מהשנייה ב 1/256 mV ערך זה נקרא הרזולוציה של הממיר. לכן מתח המוצא מוגדר כך– Vout = Vin/256 כאשר מתח הכניסה זה הערך הדצימלי שמתקבל.

3.1. קוד –

```
`include "constants.vams"
`include "disciplines.vams"

module dac (clk, Vin[7:0], Vout_a);
  input clk;
  input [7:0] Vin;          // 8-bit digital input
  output Vout_a;

  electrical clk, Vout_a;
  electrical [7:0] Vin;

  parameter real vdd = 1.704;
  parameter real resolution = 1/256;         // LSB step size =0.0039

  real analog_out;
  integer digital_val;

  analog begin
    @(initial_step) begin
      analog_out = 0.0;
      digital_val = 0;
    end

    // On rising edge of clk, update analog_out
    @(cross(V(clk) - vdd/2, +1)) begin
      digital_val = 0;
      if (V(Vin[0]) > 0.5) digital_val = digital_val + 1;    // 2^0
      if (V(Vin[1]) > 0.5) digital_val = digital_val + 2;    // 2^1
      if (V(Vin[2]) > 0.5) digital_val = digital_val + 4;    // 2^2
      if (V(Vin[3]) > 0.5) digital_val = digital_val + 8;    // 2^3
      if (V(Vin[4]) > 0.5) digital_val = digital_val + 16;   // 2^4
      if (V(Vin[5]) > 0.5) digital_val = digital_val + 32;   // 2^5
      if (V(Vin[6]) > 0.5) digital_val = digital_val + 64;   // 2^6
      if (V(Vin[7]) > 0.5) digital_val = digital_val + 128;  // 2^7
      analog_out = digital_val * 0.0039;
    end
    // Output voltage with smoothing
    V(Vout_a) <+ transition(analog_out, 0, 10n);
  end
endmodule
```

הגדרנו שעון ווקטור כניסה בעל 8 ביטים ומוצא אחד המהווה את הערך האנלוגי.

גם פה נגדיר את מתח הספק כפרמטר לשם בדיקה עבור עליית שעון ובנוסף נגדיר משתנה רזולוציה שהוא כפי שהסברנו בתחילת החלק מהווה את גודל הצעד בין שני רמות סמוכות.

נגדיר משתנה המקבל ערכים שלמים והוא יהווה לנו את הערך הדצימלי של מערך הכניסה ומשתנה ממשי שיקבל את הערך האנלוגי לאחר ההמרה ויחובר למוצא.

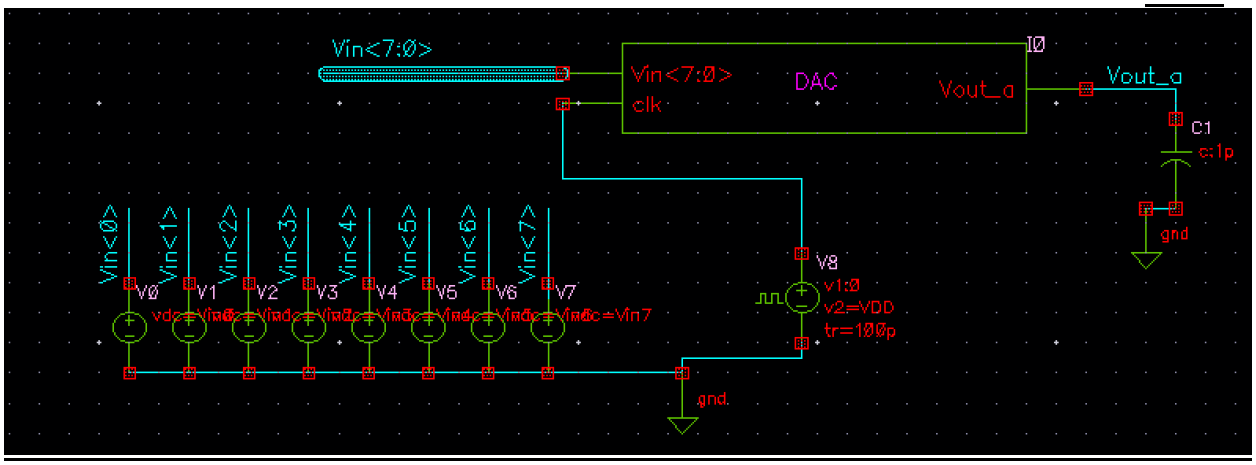
הרכיב בכל עליית שעון יחשב את הערך הדצימלי על ידי סכימה של הערך הדיגיטלי עבור כל אחד מהביטים לפי הערך העשרוני שלהם והאם הם דלוקים, כלומר יסכום את כל הערכים העשרונים של הביטים שדולקים. לאחר מכן נכפיל את הערך ברזולוציה ונקבל את הערך האנלוגי הרצוי.

כמובן שנוציא את הערך שהתקבל בצורה רציפה כמו שעשינו בשאר החלקים.

3.2. <u>סימבול –</u>

נשים לב שבכניסה קיבלנו פורט אחד על אף זה שהכניסה היא וקטור בעל 8 ביטים וסביר להניח כי נצטרך 8 פורטים לכל ביט אך למעשה בעזרת לייבלים מתאימים כפי שנראה בסעיף הבא ניתן לעזור לסימולטור לדעת איזה ערך נכנס לכל ביט מהספקים במקום לייצר 8 פורטים נפרדים.

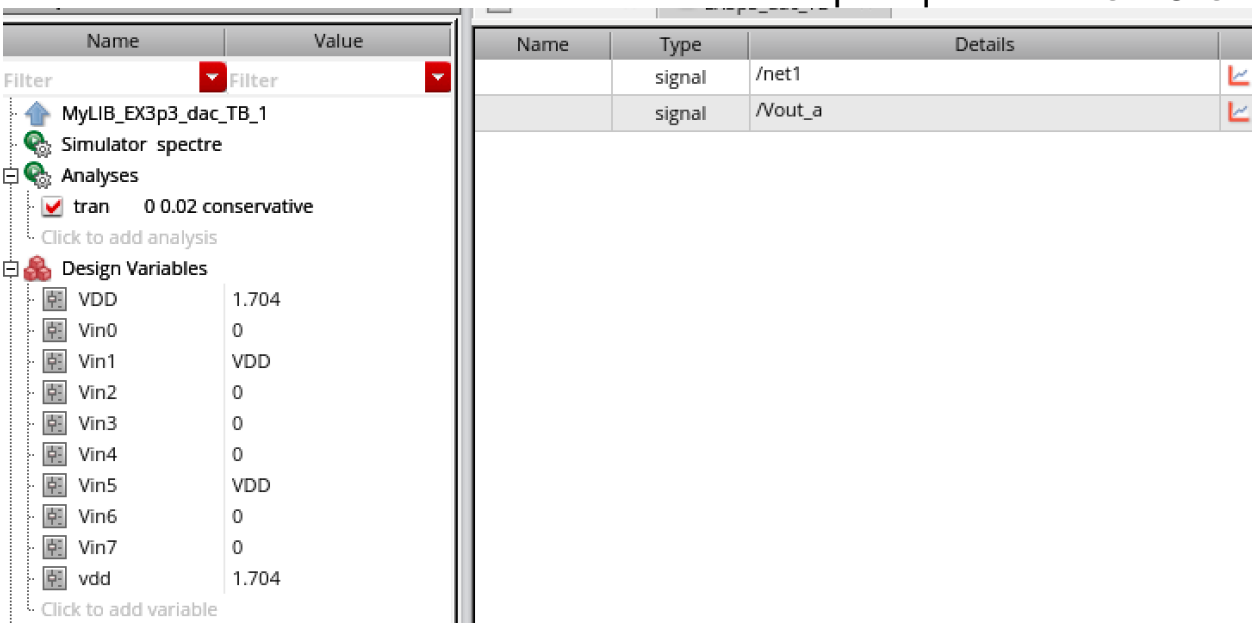
3.3. <u>TB –</u>



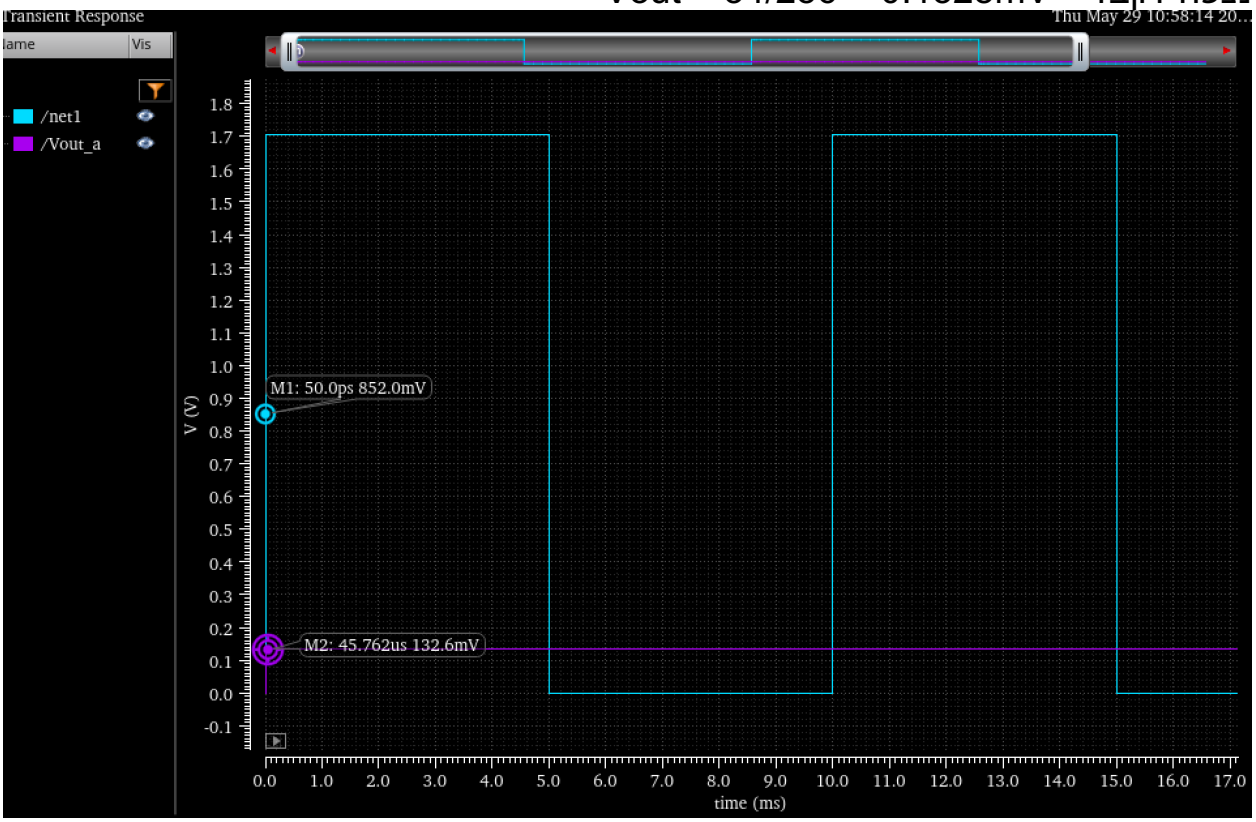
נחבר קבלים במוצאים כדי שיטענו במתח שהרכיב מוציא וניתן לכניסה לייבל שמתאר מערך בעל 8 איברים ואז לכל ספק ניתן את הלייבל עם השם של האיבר המתאים מאותו מערך, ככה התוכנה יודעת להתאים את החוט לביט המתאים בתוך הרכיב.
נחבר לשעון אות ריבוע בתדר 100hz.
כעת נעבור למאסטרו וניצור שתי סימולציות trans באורך שתי מחזורים של השעון.

- <u>Vin = G = 34</u>

ערך בינארי בכניסה הזו הוא – 00100010 בשמאל זה הMSB.
נסתכל על מתחי היציאה וניצור ביטוי שמחסר בין המוצאים במצב trans ויוצר הפרש דיפרנציאל לאורך הזמן .



נצפה לקבל – Vout = 34/256 = 0.1328mV

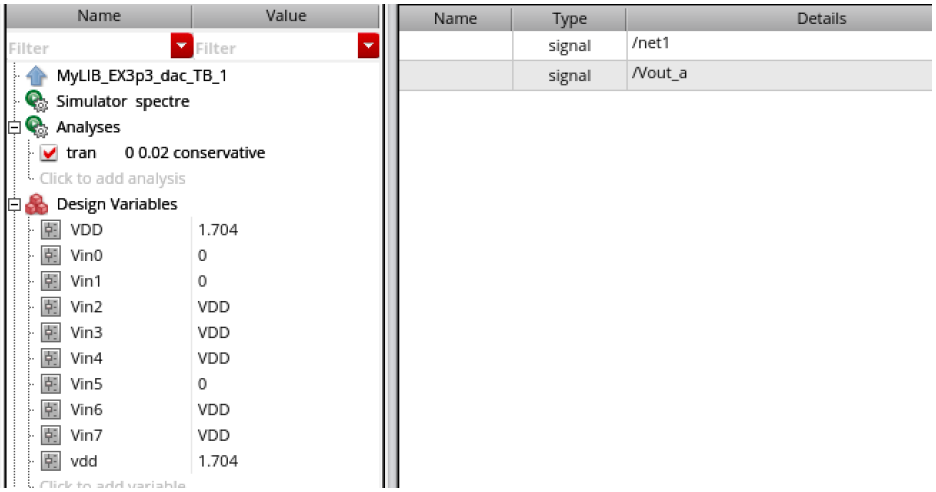


קיבלנו את הערך שציפינו לו במוצא וזה גם עם ההשהיה מרגע עליית השעון שממודל בעזרת פונקציית transient

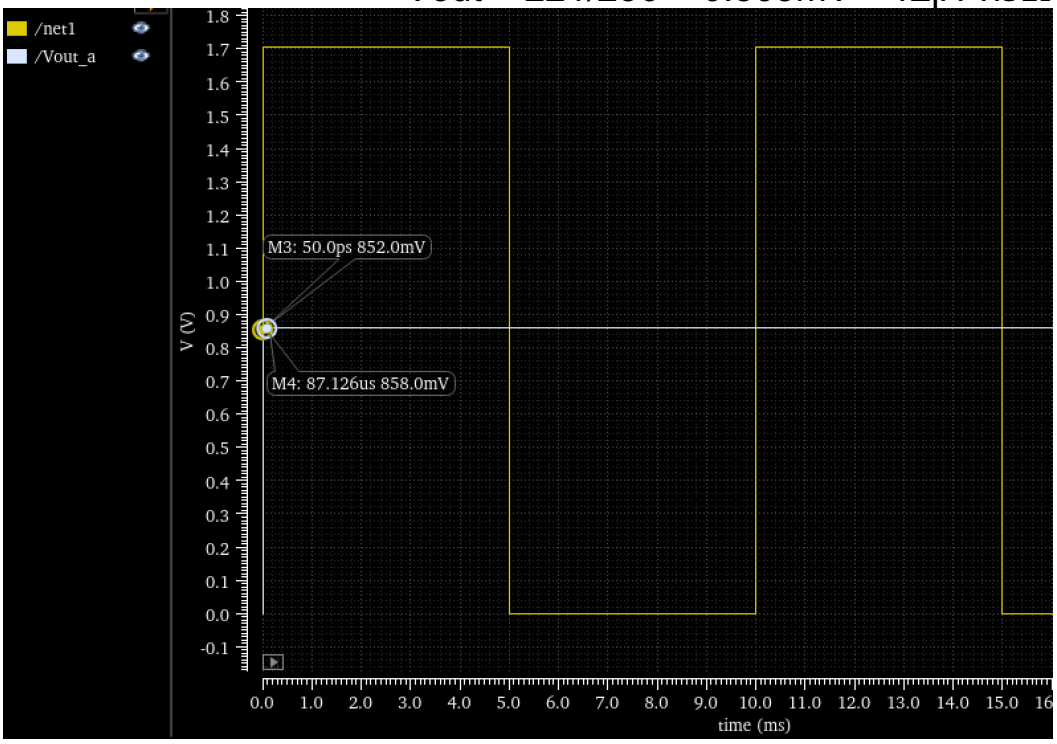
- Vin = $G = 221$

זה הערך הבינארי המשלים למספר הקבוצה שלנו – 11011101 בשמאל זה הMSB.

נסתכל שוב על מתחי היציאה וניצור ביטוי שמחסר בין המוצאים במצב trans ויוצר הפרש דיפרנציאל לאורך הזמן .



נצפה לקבל – Vout = 221/256 = 0.863mV



קיבלנו את הערך שציפינו לו במוצא וזה גם עם ההשהיה מרגע עליית השעון שממודל בעזרת פונקציית transient

3.4. האם קיבלתם את התוצאה לה ציפיתם ?

אכן קיבלנו את התוצאה לה ציפינו מאחר ורואים כי בשתי המקרים שבחנו התקבל ערך אנלוגי לו ציפינו מהחישוב האנלוגי בהתאם לעליית השעון ויחד עם השהייה ריאלית ומעבר רציף של המוצא.

4. רכיב Digital to Analog converter reference –

רכיב שממיר מידע דיגיטלי (בעצם תוצאה בינארית של אלגוריתם – SAR Successive Approximation Register) למתח אנלוגי שמייצג את התוצאה.
המטרה של המודול היא לדמות **DAC בתוך לולאת SAR-ADC**. כל פעם שמגיע פלט מהקומפרטור, הוא מחליט אם להעלות או להוריד את הקוד הדיגיטלי (DAC code), ומייצר מתח אנלוגי בהתאם לערך הזה. המתח האנלוגי הזה משמש להשוואה מול אות הקלט האנלוגי בתוך תהליך ה־SAR.

4.1. קוד –

```
`include "constants.vams"
`include "disciplines.vams"

module DACref(
    input clk,                  // Clock signal
    input comp,                 // Comparator output
    input [2:0] index,          // Current SAR step (0 to 7)
    input reset_b,              // Active-low reset
    output Vout_ref       // Analog DAC output
);
    electrical clk, comp, reset_b, Vout_ref, index[2:0];
    parameter real VDD = 1.704;            // Reference voltage

    integer code;
    integer last_index;
    integer i;
    real vout;

    analog begin
        @(initial_step) begin
            code = 128;
            last_index = -1;
            vout = code / 256.0;
        end

        // Falling edge of reset_b → reset code
        @cross(V(reset_b) - VDD/2, -1) begin
            code = 128;
            last_index = -1;
             vout = code / 256.0;
        end

        // Rising edge of clk
        @cross(V(clk) - VDD/2, +1) begin
         // Convert analog index[2:0] to integer
            i = 4 * (V(index[2]) < 0.5) + 2 * (V(index[1]) < 0.5) + (V(index[0]) < 0.5);

            if (i != last_index) begin
                code = code + ((2 * (V(comp) > 0.5 ? 1 : 0) - 1) * (1 << i));

                // Clamp code to [0, 255]
                if (code > 255) code = 255;
                if (code < 0)   code = 0;
                last_index = i;
            end
            vout = code / 256.0;
        end

        // Smooth analog output
        V(Vout_ref) <+ transition(vout, 0, 10n);
    end

endmodule
```

מתבצע בהתחלה או בהינתן reset– מאתחלים את הקוד ל־128 (אמצע טווח ה־DAC), מייצרים את ה־vout הראשוני.
בכל עליית שעון המודול:

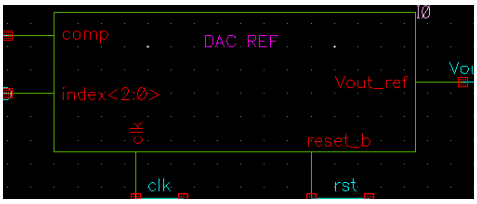
- קורא את הערך האנלוגי של index[2:0] ומחשב את שלב הסבב מ0 עד 7 בעזרת כניסות הindex.
- בודק אם השלב השתנה (i != last_index).
  אם כן, מעדכן את הקוד לפי הנוסחה –

$$code = \text{code} + (2 * \text{comp} - 1) * 2^{7-i}$$

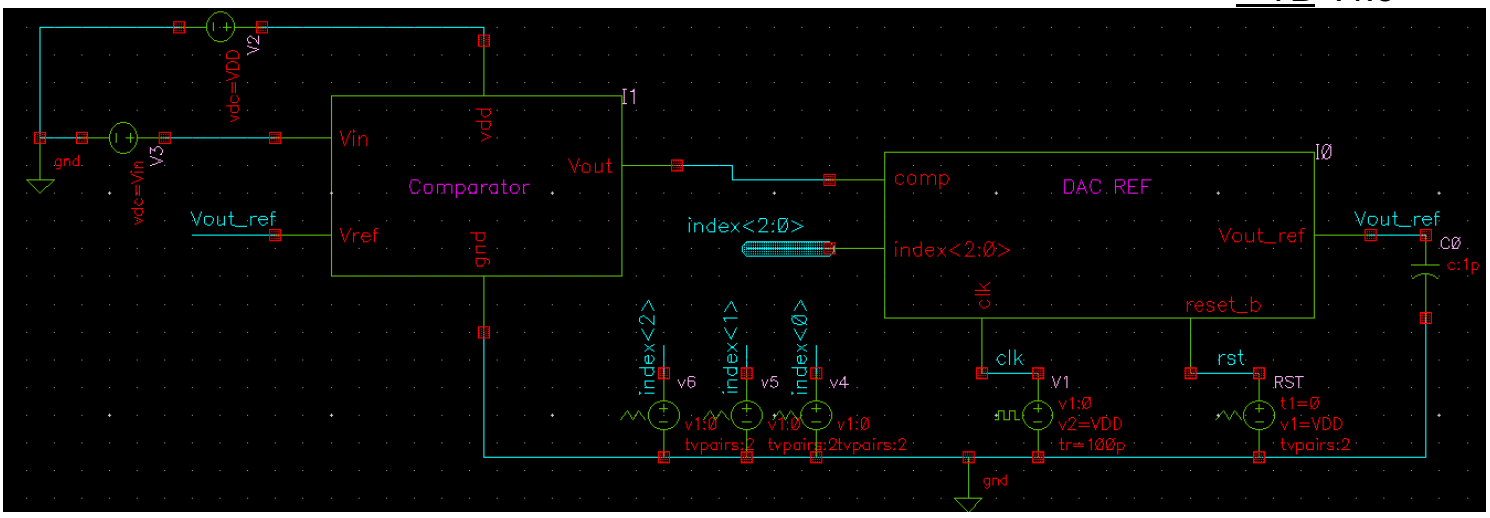
- אם הקומפרטור אמר שהמתח הנוכחי קטן מדי (comp==1), מעלים את הקוד.
  אחרת מורידים. כל שינוי תלוי בערך הביט המתאים ל־i.

לאחר מכן מנרמלים את ערך הקוד בערך המקסימלי האפשרי וזה הערך החדש במוצא יחד עם פונקציית transition שתתן מעבר חלק ללא קפיצות.

4.2. <u>סימבול –</u>



4.3. <u>TB –</u>



תחילה חיברנו את הקומפרטור שעשינו בחלק 2 כך שהמוצא שלו יהווה את הכניסה ברכיב החדש שמחליטה אם להחסיר או להוסיף, הכניסות של הקומפרטור הם מתח הכניסה Vin שאליו בעצם הרכיב החדש ישאף. לכניסת הרפרנס בקומפרטור נכניס את מתח המוצא של הרכיב החדש אותו מתח אליו רוצים לשאוף.
לrst נכניס מתח vpwl ונוריד אחרי חצי מחזור שעון, את השעון נכניס עם דיליי של חצי מחזור כדי שיסתדר לנו עם כניסות הindex.

-rst

Number of pairs of points	2
Time 1	0 s
Voltage 1	VDD V
Time 2	1/200 s
Voltage 2	0 V
Noise file name	

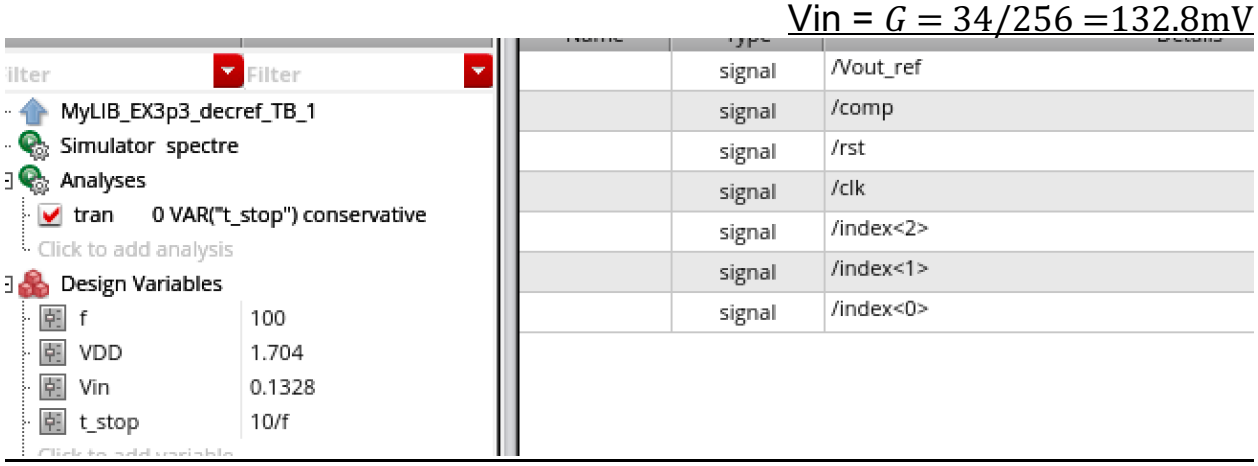
לכניסות האינדקס נכניס מתח ריבועי עם תדרים משתנים בחזקות 2 כדי לקבל ספירה עולה מ0 עד 7 כדי לדמות לולאת for בקוד כמו שהסברנו בתחילת החלק, להלן דוגמא עבור index<0> -

Voltage 1	0 V
Voltage 2	VDD V
Period	2/f s
Delay time	1/f s
Rise time	100p s
Fall time	100p s

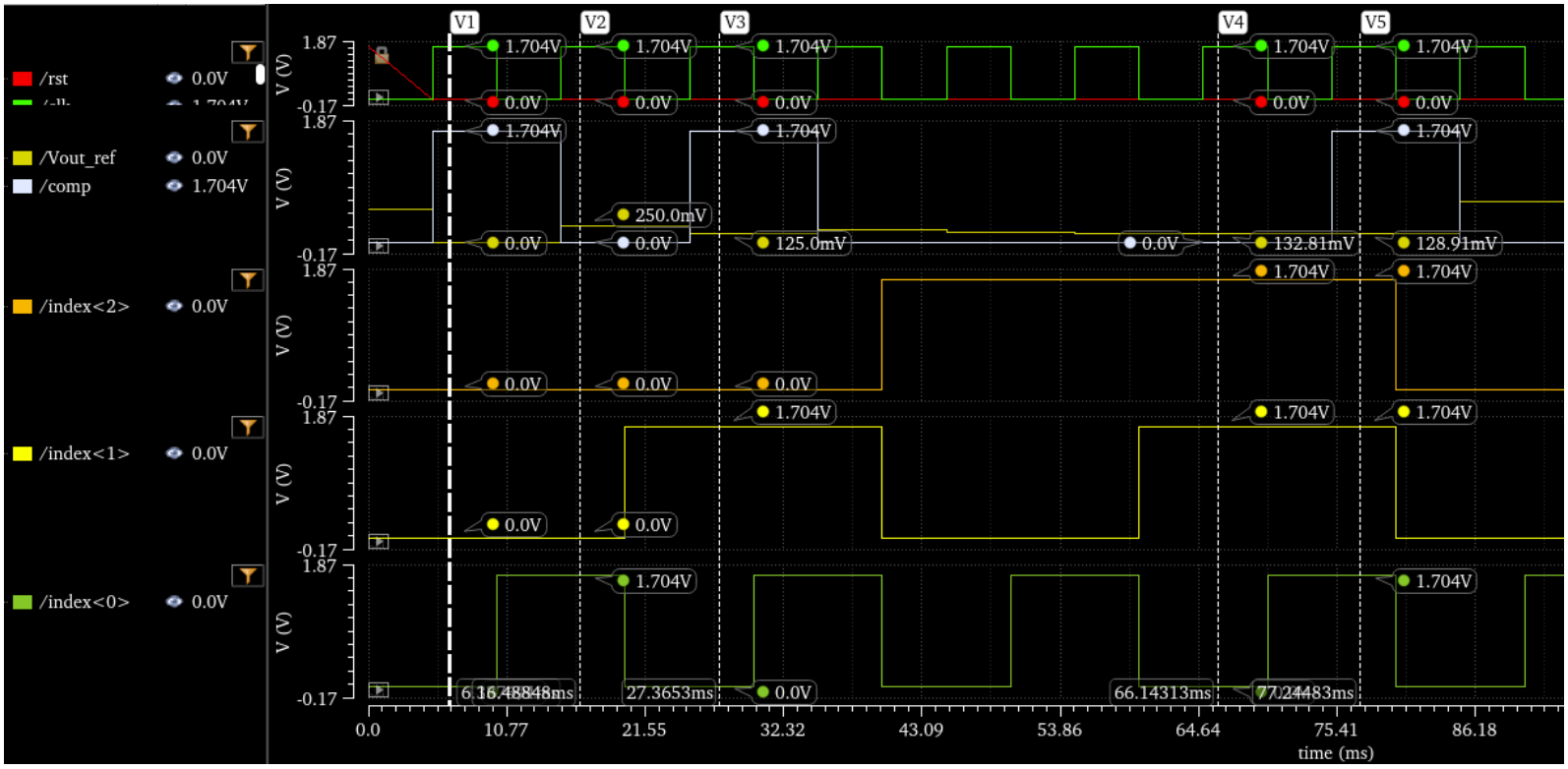
כך, בשביל לקבל ספירה עולה פשוט חיברנו לindex<1> גם מתח ריבועי רק שהדיליי וזמן המחזור מוכפלים ב-2, באותו אופן עבור index<2> נכפיל אותם ב-4.

כעת למתח הכניסה נחבר מתח ישר על פי הסימולציה אותה נבצע –

- Vin = $G = 34/256 = 132.8\text{mV}$

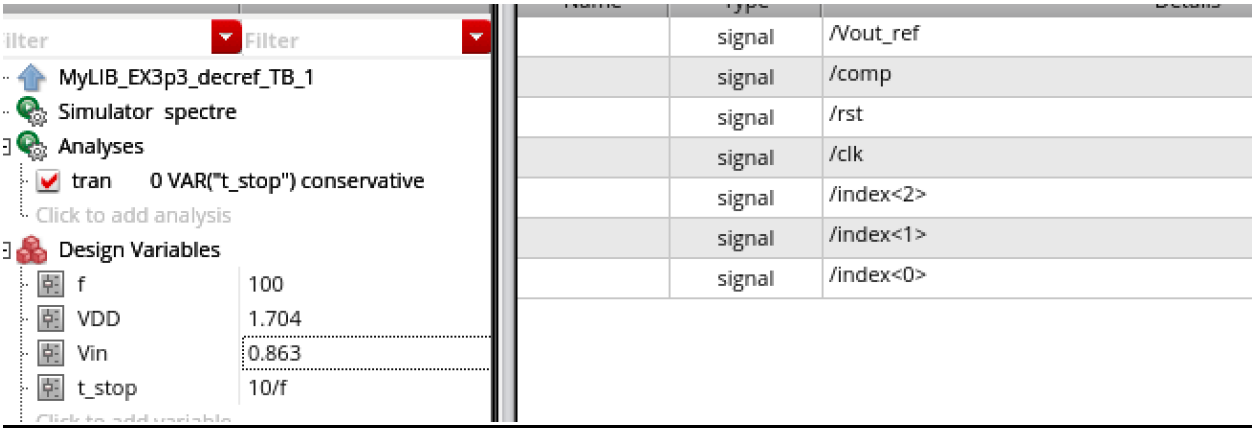


תחילה נכניס כמתח את מספר הקבוצה מנורמל ב256 וזה המתח אליו נרצה לשאוף במוצא.

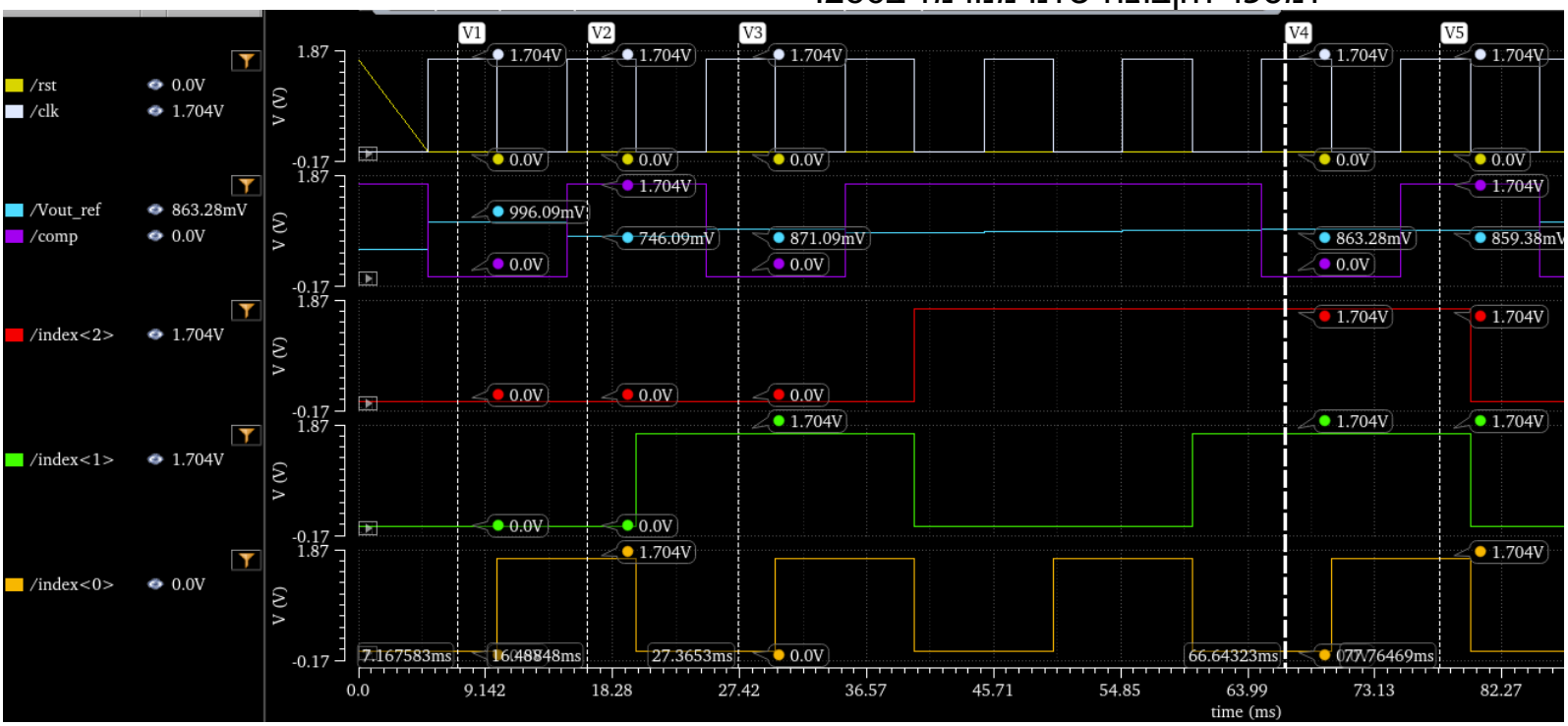


תחילה, רואים כי rst מביא את מתח המוצא ל0.5v והקומפ ב-0 כי מתח הכניסה נמוך כעת ממתח הרפרנס (המוצא), בv1 אנו רואים שהאינדקס הוא 0 והקומפ הוא 0 לכן **יורדים** ב128 ומגיעים ל0 במוצא, כעת מיד הקומפ עולה כי עכשיו מתח הכניסה גבוה ממתח הרפרנס.
בv2 האינדקס הוא 1 והקומפ גם גבוה לכן **נעלה** ב64 ונגיע ל- 64/256=250mV .
וככה ממשיכים ונשים לב שככל שהזמן עובר כך האינדקס עולה ומאפשר רוזולוציה נמוכה יותר עבור דיוק גבוה יותר. בv4 **נגיע למתח הרצוי** אך מאחר והמערכת ממשיכה לרוץ נגיע לאינדקס הגבוה ביותר והוא יוריד אותנו עוד קצת ונתרחק מהתוצאה הרצויה (v5), לכן רכיב זה **שואף** תמיד להיות בקרבת ערך הכניסה (לקח 6 מחסורי שעון כדי להגיע לערך זה)

- $\underline{\text{Vin} = G = 221/256 = 863.281\text{mV}}$



כעת נכניס במתח הכניסה את הערך האנלוגי המתקבל עבור הכניסה ההופכית למספר הקבוצה שלנו מנורמל ב256.



כמו מקודם הrst מביא אותנו לנק׳ ההתחלה וכבר שם מתח הכניסה גבוה ממתח המוצא ולכן נוסיף עוד 128 לערך הדיגיטלי (code) ונגיע 256 כלומר 1v במוצא (יש שגיאה קטנה ככל הנראה מדחיפת עומסים).
נמשיך כך עד שנגיע לערך הרצוי(v4) ונשאר קרובים אליו גם בהמשך ריצת הTB (לקח 6 מחזורים להגיע לערך הרצוי).

4.4. <u>האם קיבלתם את התוצאה לה ציפיתם ?</u>

אכן קיבלנו בשתי הסימולציות את הערך הרצוי וזה בהינתן כך שהרכיב לא יכול להימצא תמיד בערך המדויק ולהיעצר שם מאחר והוא תלוי בכניסות האינדקס שלו ובסביבה שבה בודקים אותו.