הפקולטה להנדסת חשמל

פרויקט סיום מעבדה 1ח'

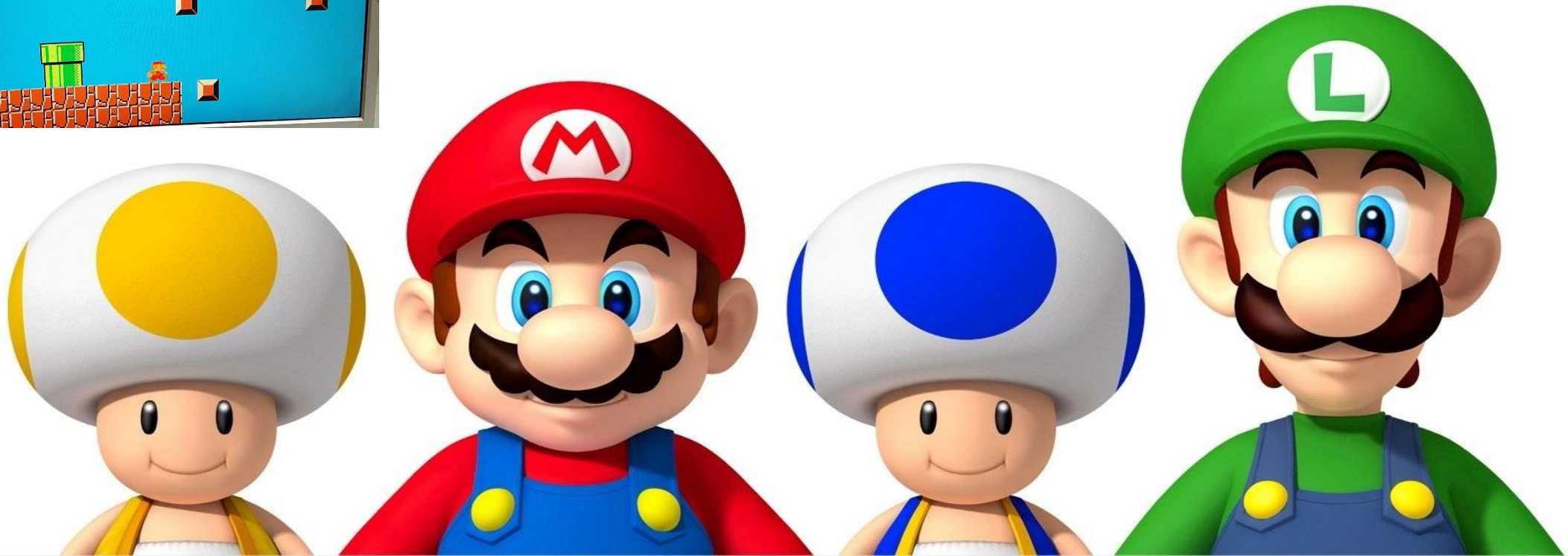
Super Mario Bros

קיץ 2017

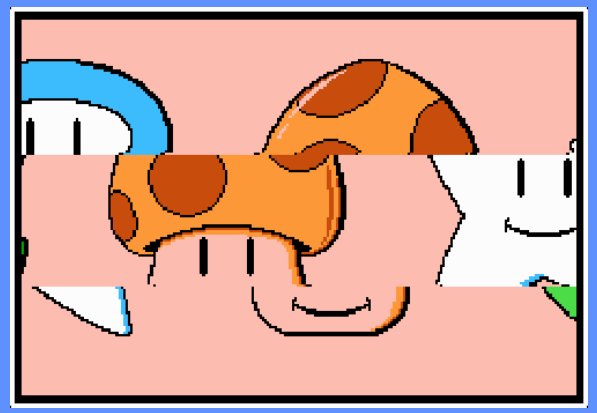
אמיר סאלח

ראיד מנדו

מדריכים: אברהם/עדי



אפיון הפרוייקט (1)



הדרישות המקוריות:

- שחקן הנע בציר X וקופץ (ישנם מכשולים עליהם יכול השחקן לעמוד).
- ישנם גושי זהב ופצצות המוגרלות אקראית ונעים לעבר השחקן במהירויות שונות.
- בכל פעם שפוגעים בגוש זהב מקבלים ניקוד ומופיע צליל זכיה.
- בכל פעם שפוגעים בפצצה מאבדים חיים ומופיע צליל פספוס.

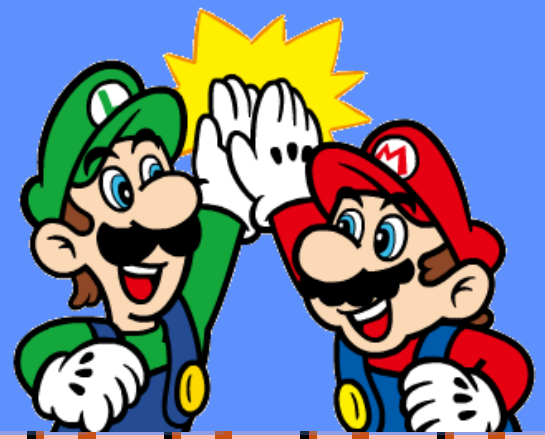
הנחות נוספות:

- בניית הגרפיקה באופן מודולרי וסקלבילי שיאפשר בניית מפות שונות ברמות שונות והוספה קלה של דמויות ואובייקטים חדשים.
- גישה כללית ובלתי תלויה לאלמנטים במשחק (מונחה התנגשויות), הורדת או הוספת חלקים לא תשפיע על חלקים אחרים במשחק.
- גישה של חיסכון בזיכרון, שימוש באותו זיכרון לייצג כמה אובייקטים מאותו סוג <u>ובאוריינטציות שונות</u>.
- שאיפה לדיוק מקסימלי בזיהוי התנגשויות (עד 2 פיקסילים, זיהוי צד פוגע).
- זיהוי לחיצה על שני מקשים במקביל והפרדת אותות make brake.
- ריסט למערכת יחזיר את המשחק למצב התחלתי כולל כל הרגיסטרים והיציאות.

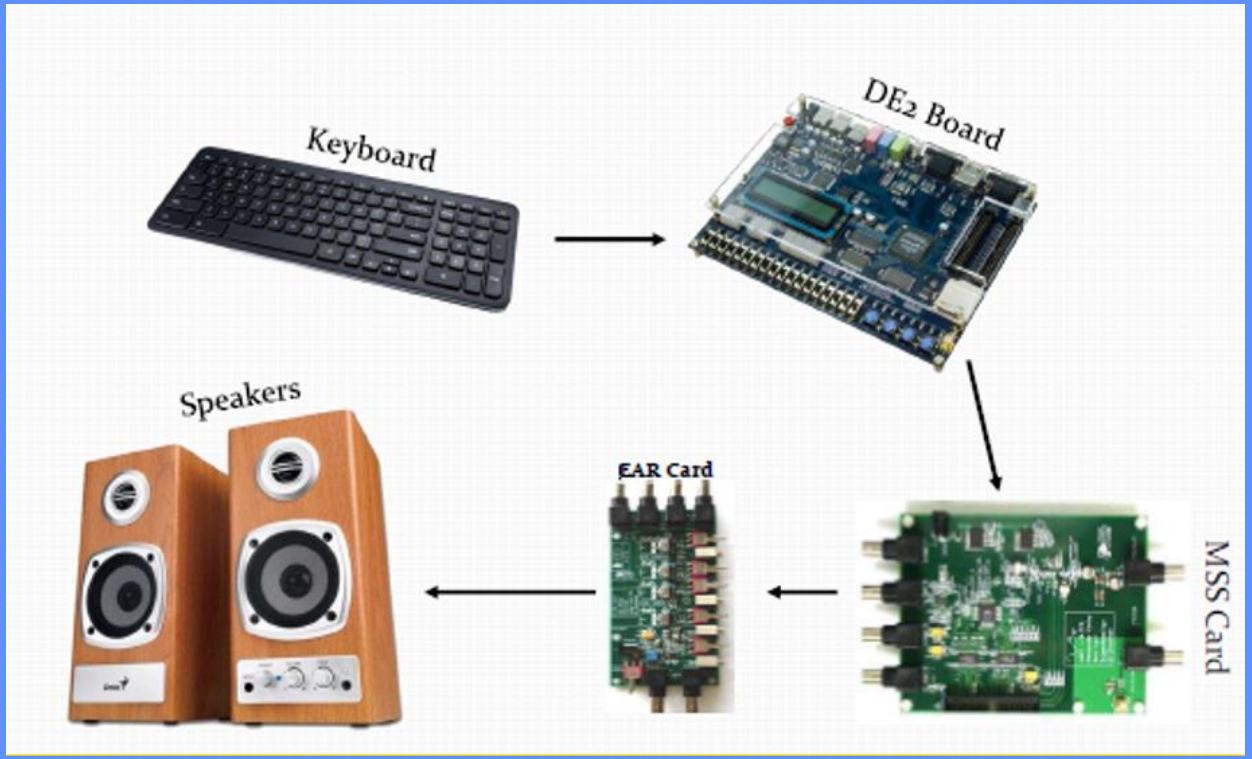
אפיון הפרוייקט (2)

<u>החלק היצירתי:</u>

- המפה נמתחת מעבר לגבולות המסך והמסך זז בהתאם למיקום השחקן במפה.
- השחקן משנה אוריינטציה כשהוא משנה כיוון תנועה או קופץ, וכשהוא נע הוא מחליף רגליים כדי ליצור אפקט של אנימציה.
- הוספנו פטריות שמקנות לשחקן עוד חיים ומגדילות אותו למריו גדול.
- יש בורות שהשחקן צריך לדלג מעליהם אחרת הוא נופל ומת.
- המפה מכילה שלב שבו יש בור גדול ובלוקים שמופיעים ונעלמים רנדומלית וצריך לעבור עליהם כדי לעבור את הבור.
- המערכת <u>יוצרת</u> מפה קטנה של כל השלב שתאפשר לראות את כל המפה בקטן.

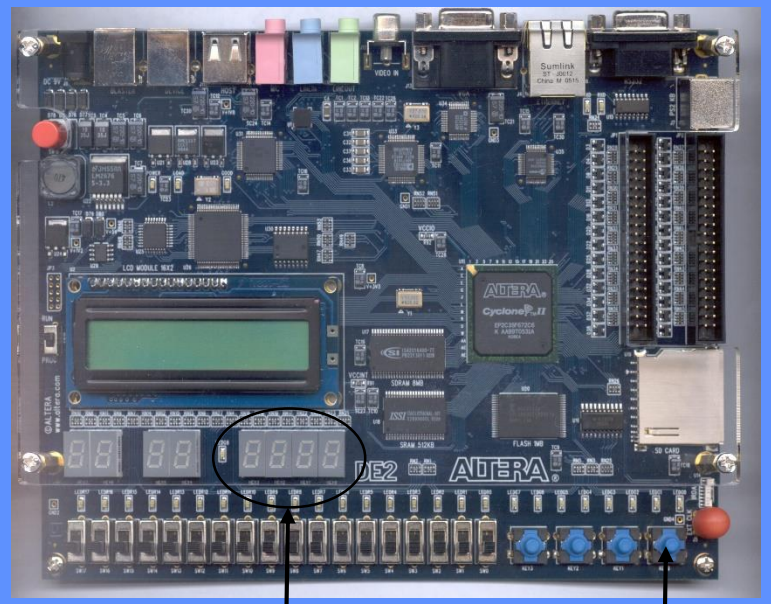


ארכיטקטורה וממשקים



VGA Screen

הוראות הפעלה



Score

Reset

Start Game

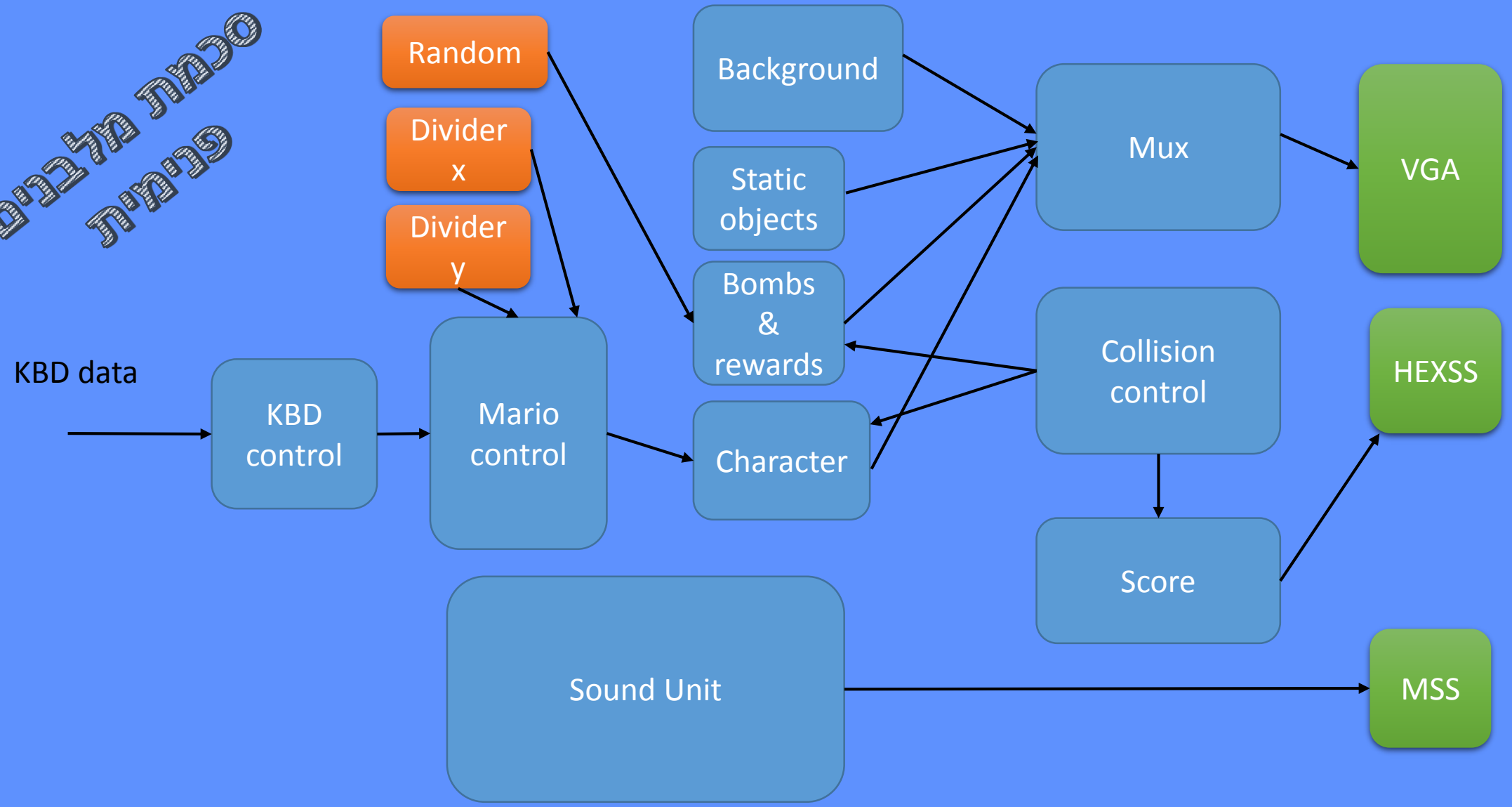
Show Map

Move Left

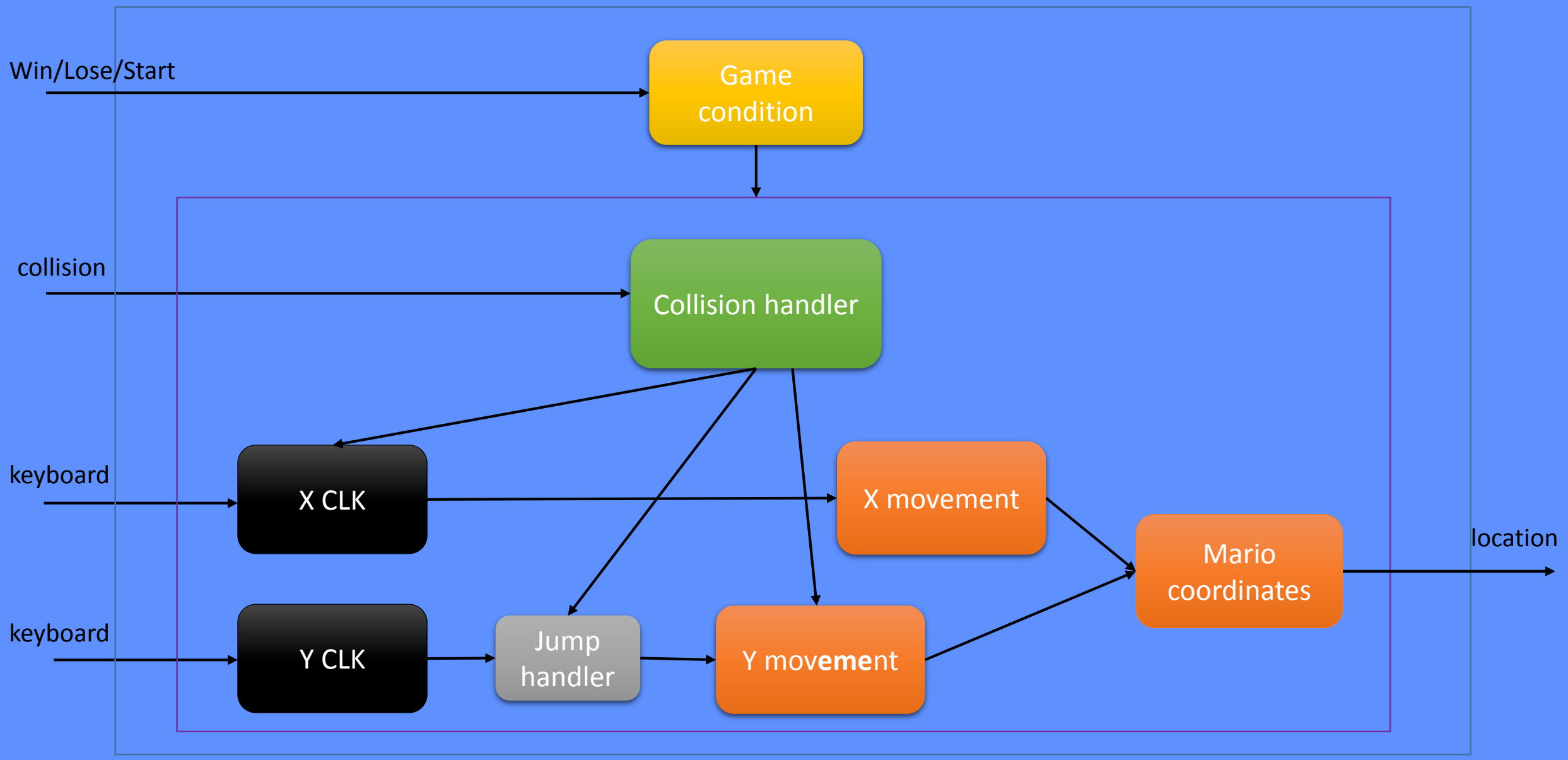
Move Right

Jump

סכמת מלבנים כללית

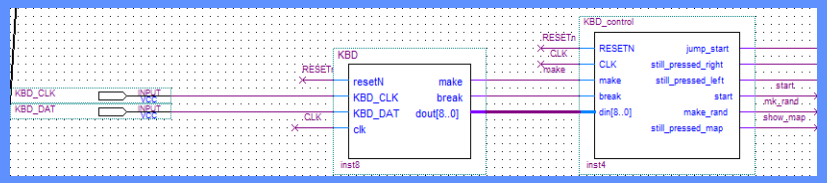


מודול Mario move דיאגרמת תהליכים

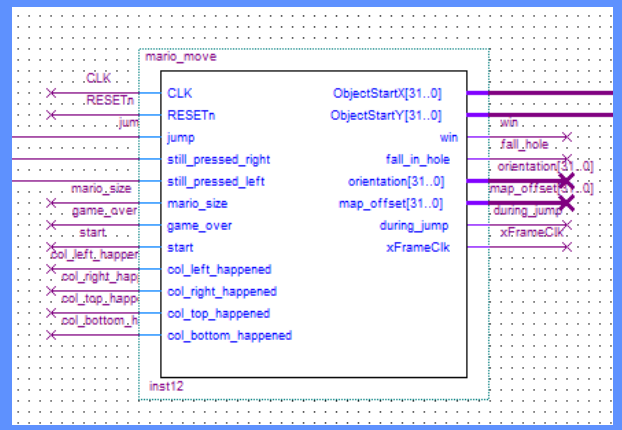


שרטוט היראכיה עליונה

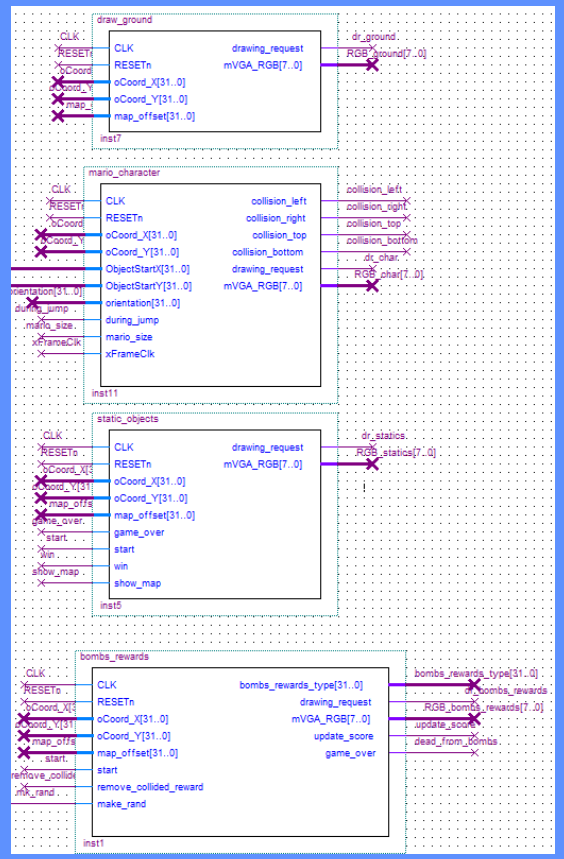
מנהל הקלט מהמקלדת



המוח של תנועת מריו (המוח של המשחק)

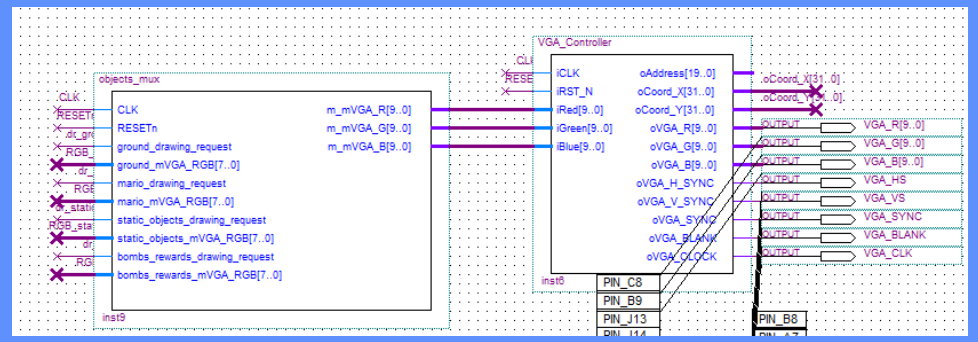


מנהלי אובייקטים לפי סוג

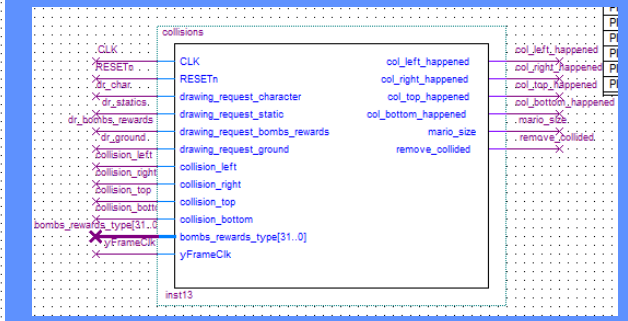


מכיל יחידת RAND

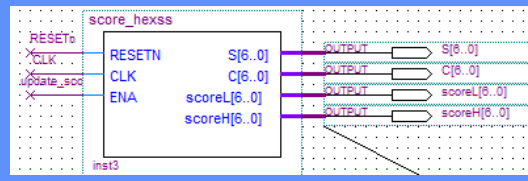
Mux + VGA ctrl



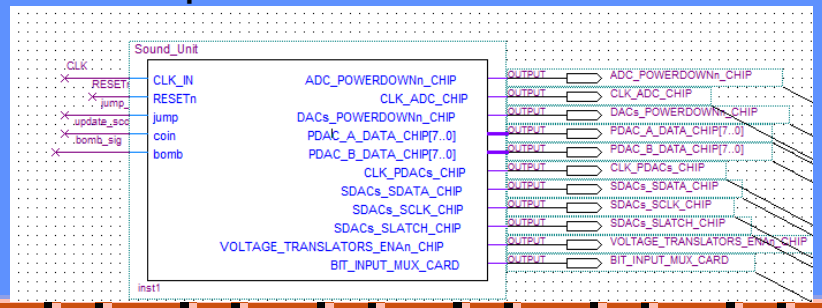
מנהל התנגשויות



לניקוד 7SEG

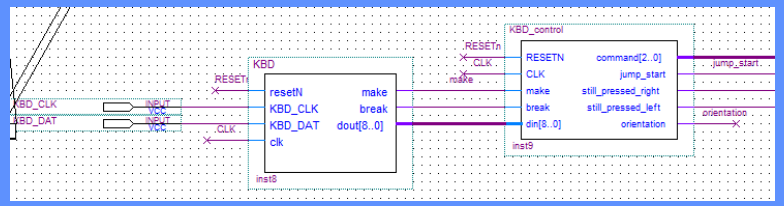


יחידת קול

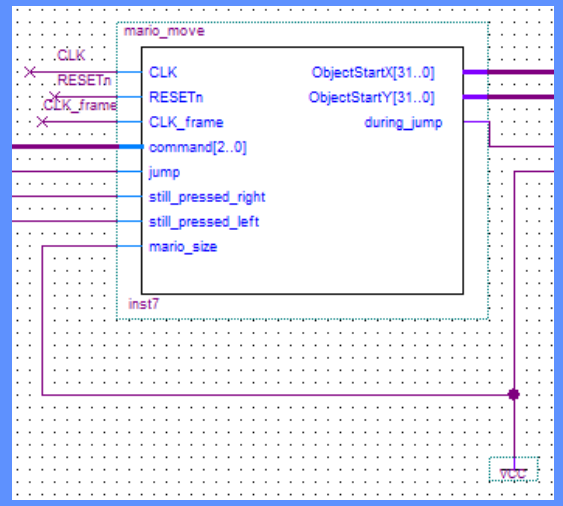


היראכיה עליונה PIPE

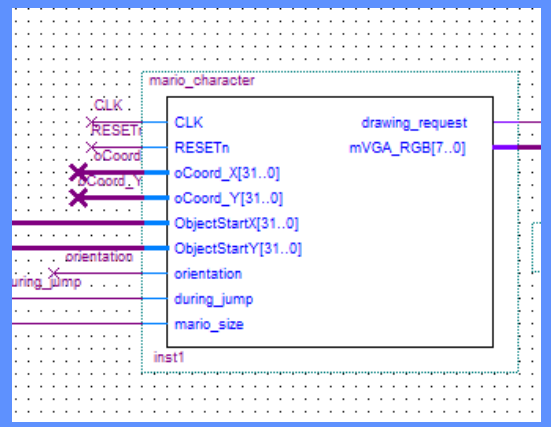
מנהל הקלט מהמקלדת(תנועות בסיסיות)



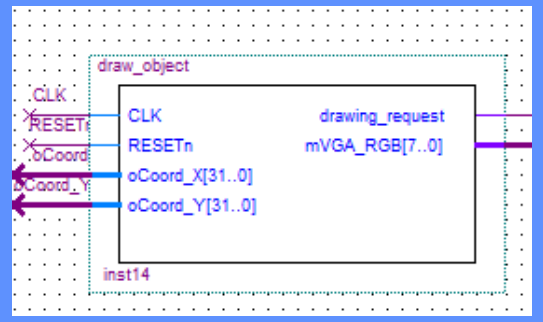
המוח של תנועת מריו בלי התנגשויות



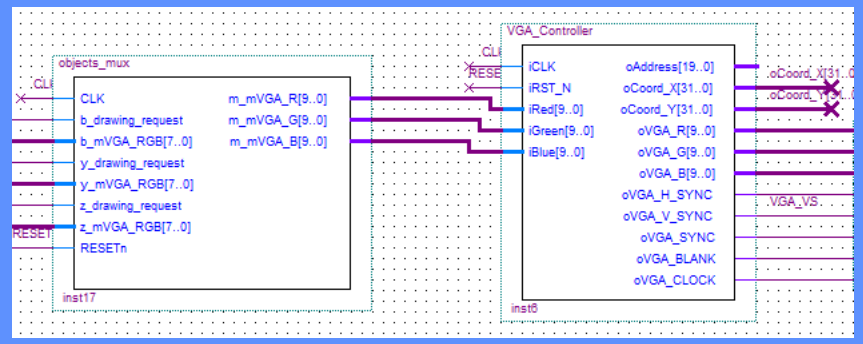
מנהלי אובייקט דמות



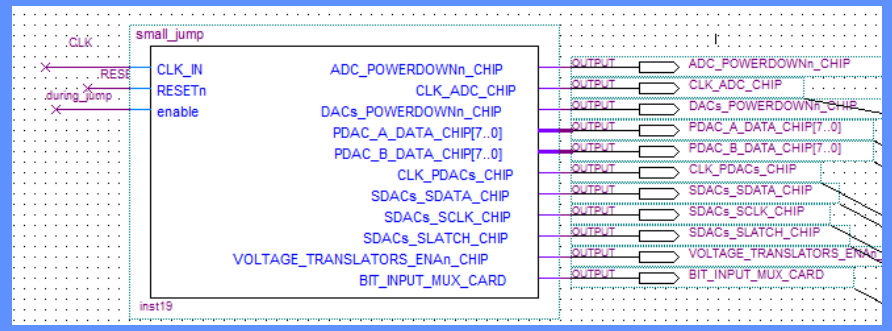
אובייקט מסך כללי



Mux + VGA ctrl



ניגון קול אחד



משימות לפי קדימויות

מס'	מודול	תפקיד	סיבוכיות
1	KBD_control	ניהול קלט מהמקלדת ולחיצות ארוכות/כפולות	קל
2	Mario Control	ינהל תנועת מריו בציר X ו Y, ישלח איתותים שינהלו את תזוזת המסך להתקדמות או החלפה בין התקדמות המסך או התקדמות השחקן, לפיו גם האנימציה מופעלת	קשה מאוד
3	Object_Control	ינהל את האובייקטים של המערכת, יצירתם, מיקומם, הצגתם והסתרתם, והתנועה שלהם.	קשה
4	MSS_Control	אחראי על השמעת קולות ע"פי סיגנלים שמתקבלים משאר המודולים במערכת, דואג להפסיק השמעת קול באמצע אם מתקבלת בקשה חדשה.	בינוני

הזזת המסך

- המפה של המשחק נפרסת על גבי כמה מסכים, השחקן נע עד שמגיע לאמצע המסך ואז המסך מתחיל לנוע בכיוון ההפוך והדמות נשארת סטטית.
- השתמשנו במשתנה אחד map_offset שמתוחזק ע"י בקר הזזת הדמות וגורם לכל האובייקטים סטטים, דינמים ואדמה לזוז בהתאם.
- הבלוקים של האדמה משוכפלים מרביעיית בלוקים יחידה וזזים בהתאם ל map_offset:

```
ground_block: pic_ground port map(address_rom,RGB_rom);

bCoord_X    <= ((oCoord_X mod object_X_size + map_offset) mod object_X_size);   -- cool formula
bCoord_Y    <= (oCoord_Y - ObjectStartY);
```

```
effectiveX <= ObjectStartX - map_offset;

objectEastXboundary <= object_X_size + effectiveX ;
objectSouthboundary <= object_Y_size + ObjectStartY ;

drawing_X   <= '1' when  (oCoord_X  >= effectiveX) and  (oCoord_X < objectEastXboundary) else '0';
drawing_Y   <= '1' when  (oCoord_Y  >= ObjectStartY) and  (oCoord_Y < objectSouthboundary) else '0';

bCoord_X    <= (oCoord_X - effectiveX) when ( drawing_X = '1' and  drawing_Y = '1'  ) else 0 ;
bCoord_Y    <= (oCoord_Y - ObjectStartY) when ( drawing_X = '1' and  drawing_Y = '1'  ) else 0 ;
```

יצירת אובייקטים חדשים

```
entity generic_object is
port    (
            CLK             : in std_logic;
            RESETn          : in std_logic;

            oCoord_X        : in integer;
            oCoord_Y        : in integer;
            map_offset      : in integer;

            ObjectStartX    : in integer;
            ObjectStartY    : in integer;
            object_X_size   : in integer;
            object_Y_size   : in integer;

            drawing_request : out std_logic ;
            address         : out integer
    );

end generic_object;
```

- כל האובייקטים במשחק (עננים, לבנים, צינורות, דמות, פצצות, מטבעות, טילים, מפה קטנה, תמונות התחלה הפסד וניצחון...) משתפים אותו מודול גינרי שמקבל קואורדנטות התחלה ומחשב אם יש drawing request בכל מחזור שעון ואם כן איפה במטריצה של האובייקט לשלוף את הצבע.

כל האובייקטים מאותו סוג מחשבים כתובת ושולפים את הצבע מאותה מטריצה

```
entity pic_star is
port(
      ADDR    : in integer range 0 to 895;
      Q       : out std_logic_vector(7 downto 0)
);
end pic_star;

architecture arch of pic_star is

type matrix is array(0 to 895) of std_logic_vector(7 downto 0);

constant star : matrix := ( ); --initialize your matrix here

begin

    Q <= star(ADDR);

end arch;
```

יצירת אובייקט חדש - דוגמה

```
-----------------------------------------------

constant bigCloudSizeX : integer := 96;
constant bigCloudSizeY : integer := 48;

COMPONENT pic_big_cloud IS
    PORT
    (
        ADDR        : in integer range 0 to (bigCloudSizeX*bigCloudSizeY - 1);
        q           : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
    );
END COMPONENT;


signal muxed_big_cloud          : integer range 0 to (bigCloudSizeX*bigCloudSizeY - 1) := 0;
signal big_cloud_rgb            : std_logic_vector (7 downto 0);
signal drawing_request_bCloud   : std_logic := '0';

-----------------------------------------------
```

הגדרה של המטריצה
שמכילה את האובייקט

הגדרה של המערכים
שמכילים את המיקומים של
האובייקטים

```
-----------------------------------------------------------------------------------------------------
constant big_clouds_locsX           : object_locations          := (150,1820,2000,1000, 1200,1450,2300,2800,3900,3600 );
constant big_clouds_locsY           : object_locations          := ( 52,52,52,52,52,52,52,52,52,52);
signal big_clouds_addresses         : addresses_array           := ( others => 0 );
signal big_clouds_dr                : drawing_requests_array    := ( others => '0' );

-----------------------------------------------------------------------------------------------------
big_cloud_rom       : pic_big_cloud         port map(muxed_big_cloud,big_cloud_rgb);

big_clouds :        for i in 0 to 9 generate
                        big_clouds :generic_object port map(CLK, RESETn, oCoord_X, oCoord_Y, map_offset, big_clouds_locsX(i)  , big_clouds_locsY(i), bigCloudSizeX, bigCloudSi
                    end generate big_clouds;
```

הגדרת זיכרון משותף

יצור מספר
אובייקטים רצוי
שמשתמשים בזיכרון
יחיד

שיפור זיהוי התנגשויות ומעברים חלקים

- המשחק מכיל מכשולים ואובייקטים שונים בגדלים שונים שנרצה למקם במקומות כרצוננו, לכן נרצה לזהות התנגשויות מוקדם ככל הניתן בצורה אחידה ולא מותאמת לאובייקט ספציפי.
- במצב אידיאלי נרצה להזיז את הדמות פיקסל אחד בכל ריענון מסך ולהספיק לקבל פידבק על התנגשות לפני הריענון הבא, מצב כזה יתרום גם לכך שהתנועה ותזוזת המסך ייראו חלקים.
- בפועל סריקת ה VGA לכל הפיקסלים במסך נעשית בתדר 87HZ, ולכן אנחנו מוגבלים בתדר הזה בחומרה והדיוק סופי.
- אבל אנחנו רוצים גם שהדמות תזוז מהר <u>ותאיץ/תאט</u>!

<u>פתרון:</u>

- השתמשנו במחלק תדר משתנה, במקום להזיז 5 פיקסלים בבת אחת נזיז פיקסל אחד כל פעם אבל בתדר גדול פי 5. בהאצה. האטה נשנה את תדר השעון.

```
entity dividerT is
  port (CLK          : in    std_logic;
        RESETN       : in    std_logic;
        HZ           : in    integer;
        speed        : in    integer;
        rate         : in    integer;
        slowClk      : out   std_logic
    );
end entity;
```

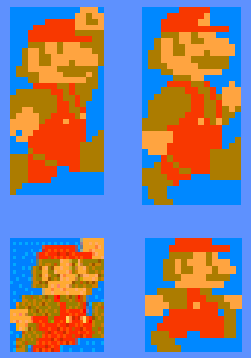
השלב המשוגע

- הלבנים נעלמים ומוצגים באופן רנדומלי כמעט כל שתי שניות, אם הלבנה עליה עומד מריו נעלמת הוא מתחיל ליפול, לשלב הזה אין ריצפה והמשחק יסתיים אם הוא לא יצליח לעבור לצד השני.

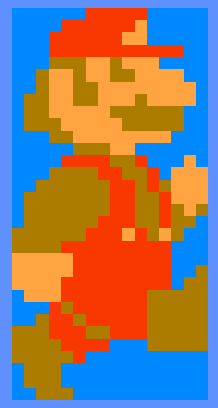
המפה הקטנה

- לחיצה על התו M תציג מפה קטנה של כל המשחק (יחס 1/16 בציר Y ו 1\32 בציר X).
- המפה מיוצרת אוטומטית, היא סורקת את המערכים של מיקומי האובייקטים הסטטיים ומעדכנת את המפה במחזור השעון הראשון.
- המפה מוגדרת כאובייקט סטטי רגיל (משתמשים באובייקט הגנרי) ולא מצריכה מנגנון מיוחד חוץ מהיצירה הראשונית שלה.

אנימציה



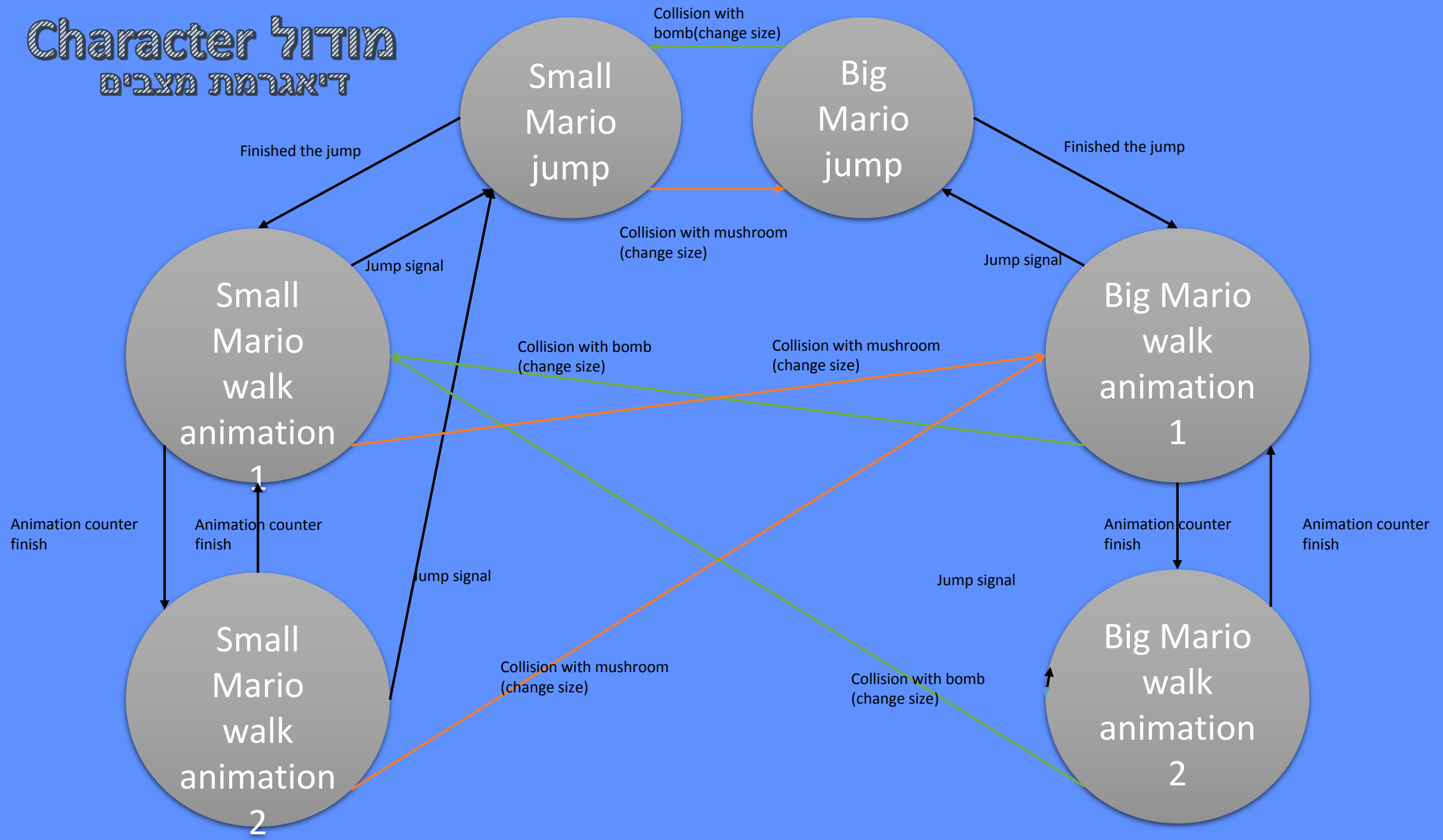
```
address_rom_small <= bCoord_Y * object_X_size + bCoord_X;
```



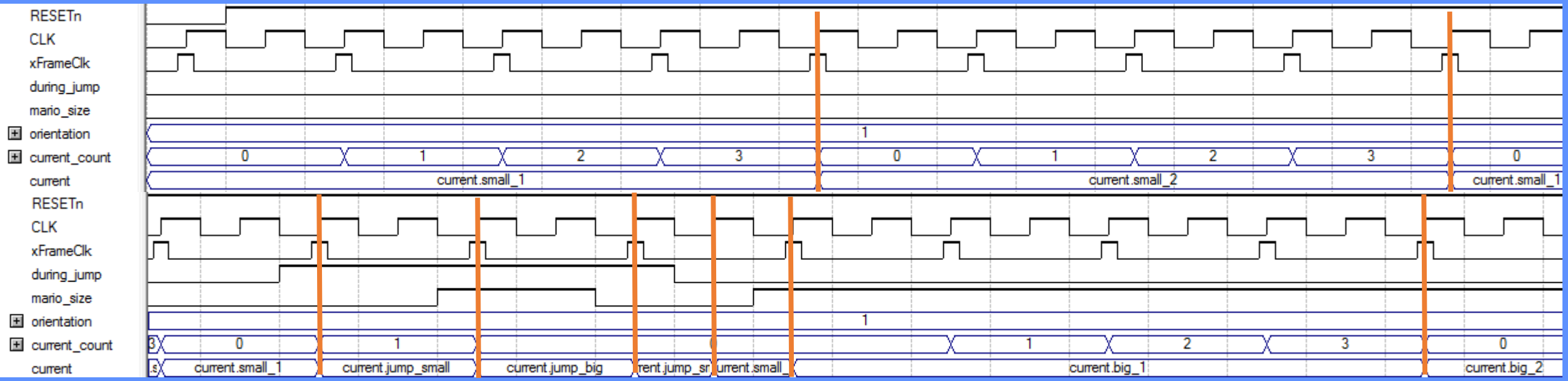
```
address_rom_small <= bCoord_Y * object_X_size + object_X_size - bCoord_X; --reading the matrix right to left
```

- הדמות של מריו מתחלפת לפי מכונת מצבים ויוצרת אפקט של אנימציה כשהדמות הולכת בכיוון מסוים או קופצת בכיוון מסוים.
- אנחנו מחזיקים תמונה אחת מכל פוזיציה של הדמות, כשנרצה להציג דמות שהולכת ימינה, נתרגם את הפיקסל הנוכחי לפיקסל בתמונה כמו שעושים בדרך כלל.
- כשנרצה להציג דמות שהולכת שמאלה, נתרגם את הפיקסל הנוכחי יחסית לפינה ימנית עליונה ונקבל תמונה שהיא מראה של התמונה המקורית בלי להשתמש ביותר זיכרון.

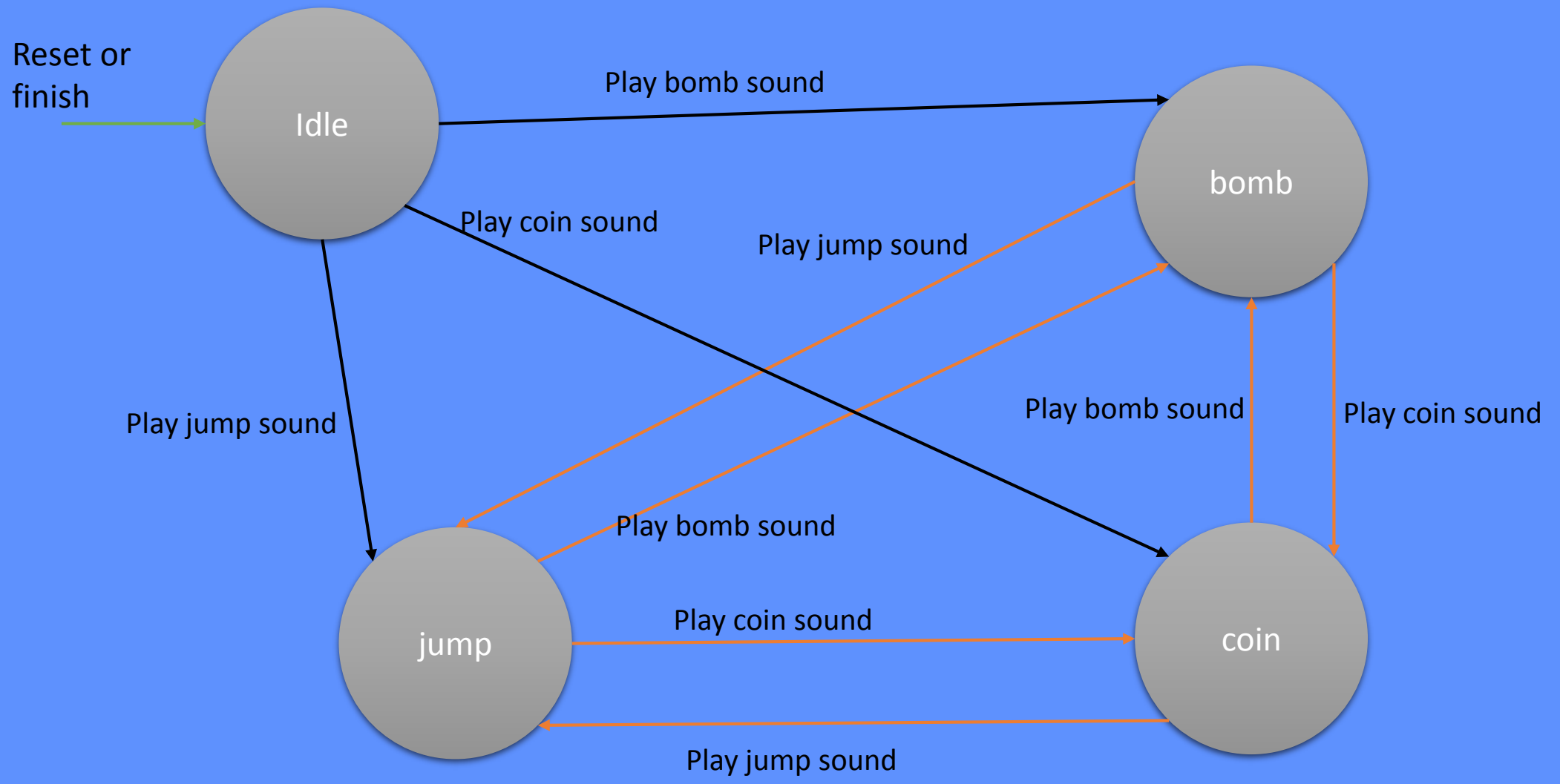
Character מודול
דיאגרמת מצבים



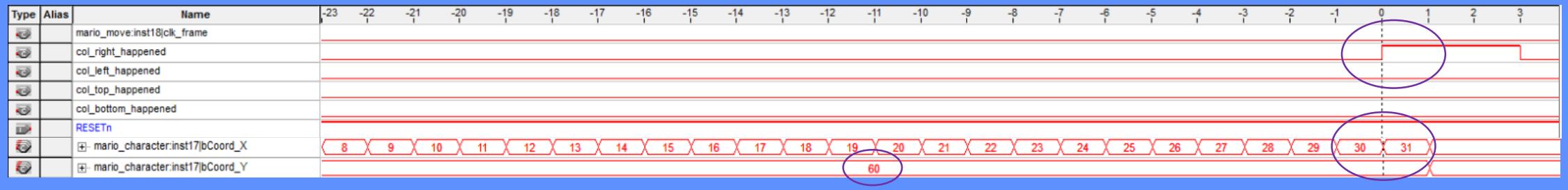
מודול Character
סימולציה



מודול Sound
דיאגרמת מצבים



זיהוי תקלת חומרה עם Signal Tap



- דיבוג מצב שבו הדמות מתחילה לטבוע לתוך האדמה כשמתקרבים לצינור וקופצים.
- אפשר לראות בST כי ברגע הפגיעה יש התנגשות מימין בגלל שפיקסל מהעמודה האחרונה התנגש עם הצינור אבל אין התנגשות מלמטה כי הדמות שקעה 4 פיקסלים לתוך האדמה לפני שהיה אפשר לזהות התנגשות, ומשם היא תמשיך לרדת ולחפש אדמה לעמוד עליה.
- הבעיה הראשונה שפתרנו הייתה לעשות את הבדיקה של צד הפגיעה באופן בלתי תלוי (אפשר להתנגש מימן ומלמטה בבת אחת).
- הבעיה השנייה הייתה חוסר הדיוק בהזזת הדמות בכמה פיקסלים בבת אחת ומשם התחלנו לחשוב ולנסות דברים עד שמימשנו את התנועה עם מחלק תדר משתנה.

סיכום ומסקנות

- **ביצוע מול אפיון:**
  - בוצעו כל המשימות שהוגדרו בתחילת הפרוייקט, ואף נוספו תכונות נוספות.
- **מה למדתי:**
  - לבנות מערכת מורכבת בחומרה בעזרת מודולים שמטפלים בהתקני חומרה שונים.
  - המון VHDL, להכיר את היכולות של השפה.
  - הצגת אובייקטים וניהול צג בעזרת פרוטוקול VGA.
  - עבודה עם שעונים ומחלקי תדר שונים.
- **מסקנות:**
  - להבין את המגבלות של התקני החומרה איתם עובדים.
  - ניהול סדר עדיפויות יותר טוב.
  - לשאוף לתכנן רכבים מודולריים וסקלבילים.
- **המלצות לעתיד:**
  - לשלב את המודול של ה VGA באחת המעבדות (החצי השני של מעבדת A2D ?), הכרה של VGA_CONTROLLER מבחינת תכן ולא רק כקופסה שחורה לדעתנו חשובה מאוד להתחלת העבודה עם הפרוייקט.


If the food isn’t pasta, it doesn’t count!