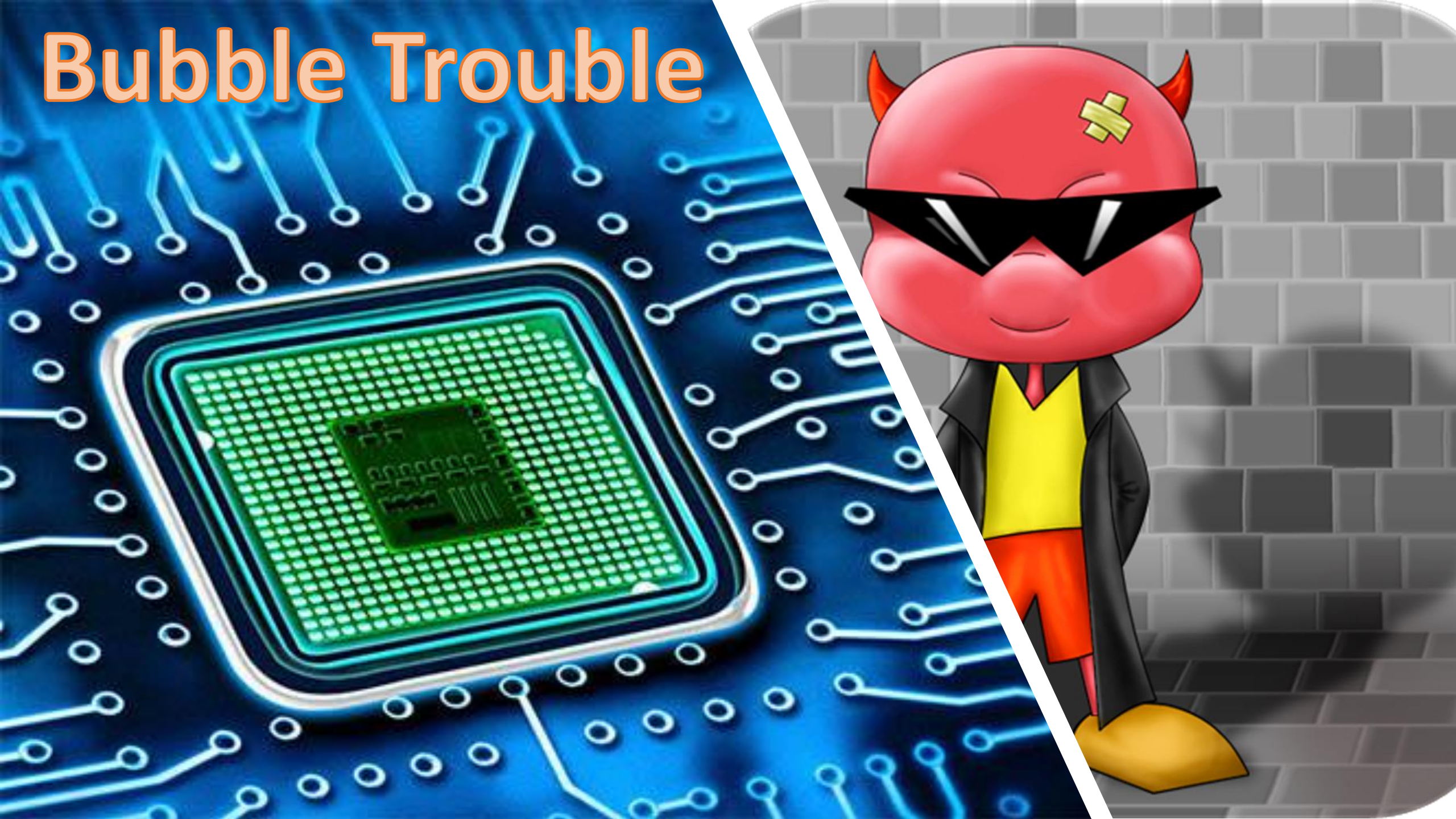




מעבדה 1א בהנדסת חשמל
פרויקט סופי

המגישים:
גמאל טנוס
עלי כיאל

Bubble Trouble



Bubble Trouble

אפיון הפרויקט

הדרישות המקוריות:

במסגרת תכנון הפרויקט היינו אמורים לתכנן את המשחק המפורסם Bubble Trouble.
לפי הדרישות המקוריות המשחק אמור לכלול שחקן אחד שיורה טילים לכיוון כדורים ולפצל כל כדור לשתיים.

המשחק מתחיל עם כדור אחד ונגמר אחרי 15 פגיעות. אם כדור כלשהו פוגע בשחקן אז הוא מפסיד חיים.
סה"כ 3 פגיעות שונות יכולות להתרחש עד שהשחקן מפסיד.



Bubble Trouble

אפיון הפרויקט

תוספות:

הוספת מסך התחלתי לפני תחילת המשחק
המסך נעלם אחרי לחיצה על מקש Enter
והמשחק מתחיל מיד לאחר מכן...

הוספת רכיב Shield שמופיע על המסך אחרי 7
התנגשויות בין החץ לכדורים... במקרה של התנגשות
בין כדור ל-Shield אז הכדור שהתנגש יופיע במקום
אקראי... ולכן ה-Shield מסוגל לעזור או להזיק לשחקן
כתלות במקום האקראי.
הוספת תפוח שמוסיף לחיים של השחקן לב נוסף במקרה
והשחקן הספיק לאכול את התפוח...



Bubble Trouble

אפיון הפרויקט

תוספות:

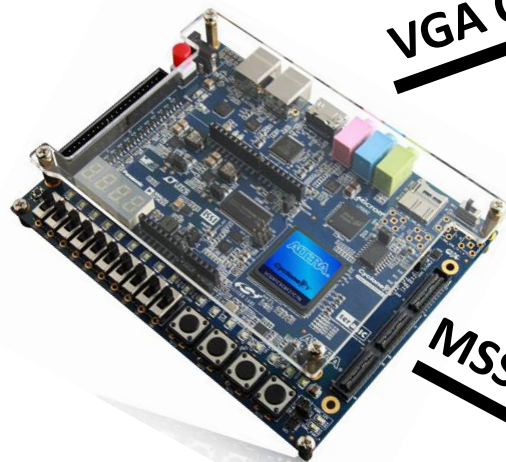
הוספת הודעות You Win\You Lose במקרה של
ניצחון או הפסד...
הוספת צלילים כשמשגרים את החץ



הקשר בין הרכיבים



KBD Modules



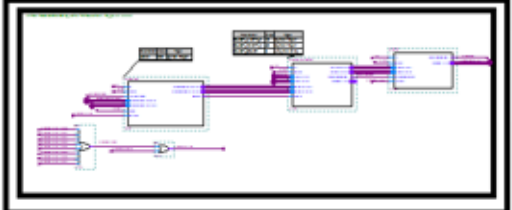
VGA Controller



MSS Modules



Arrow Modules



Apple Modules



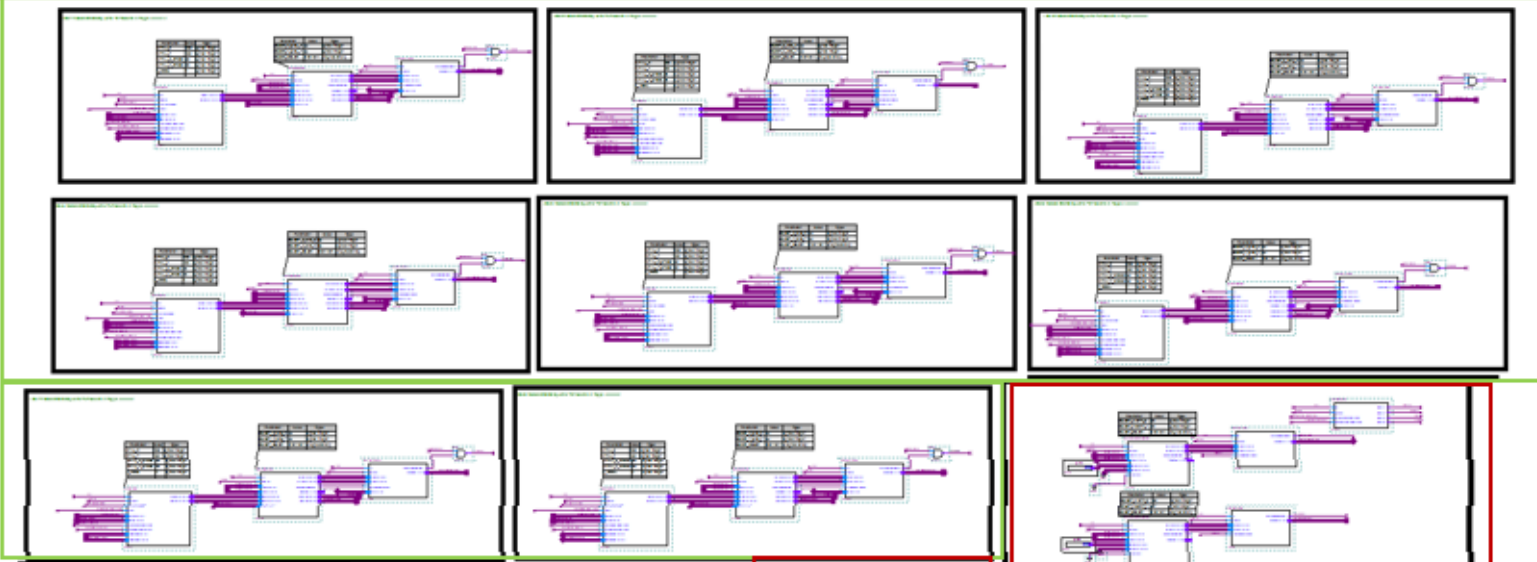
Player Modules



Shield Modules



Balls Modules



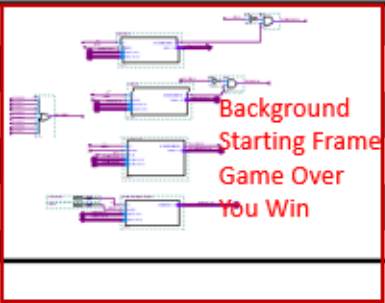
Random



Controller + Interpreter



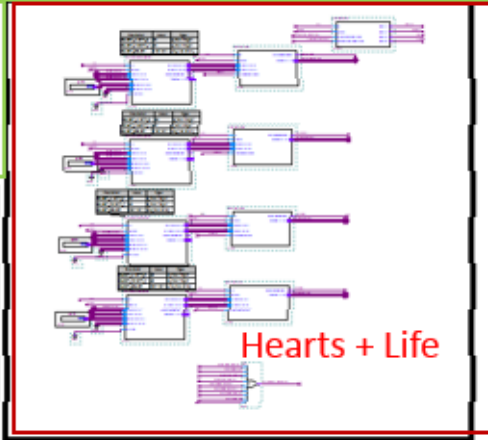
Background
Starting Frame
Game Over
You Win



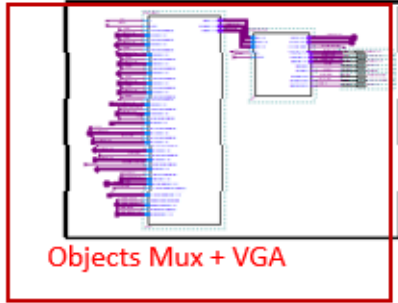
Keyboard



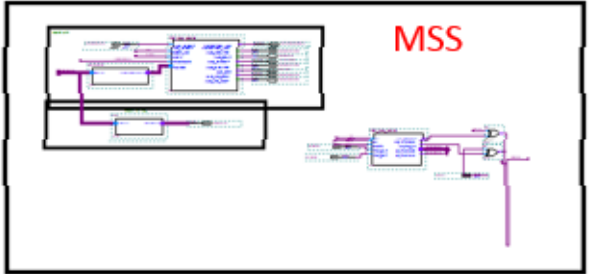
Hearts + Life



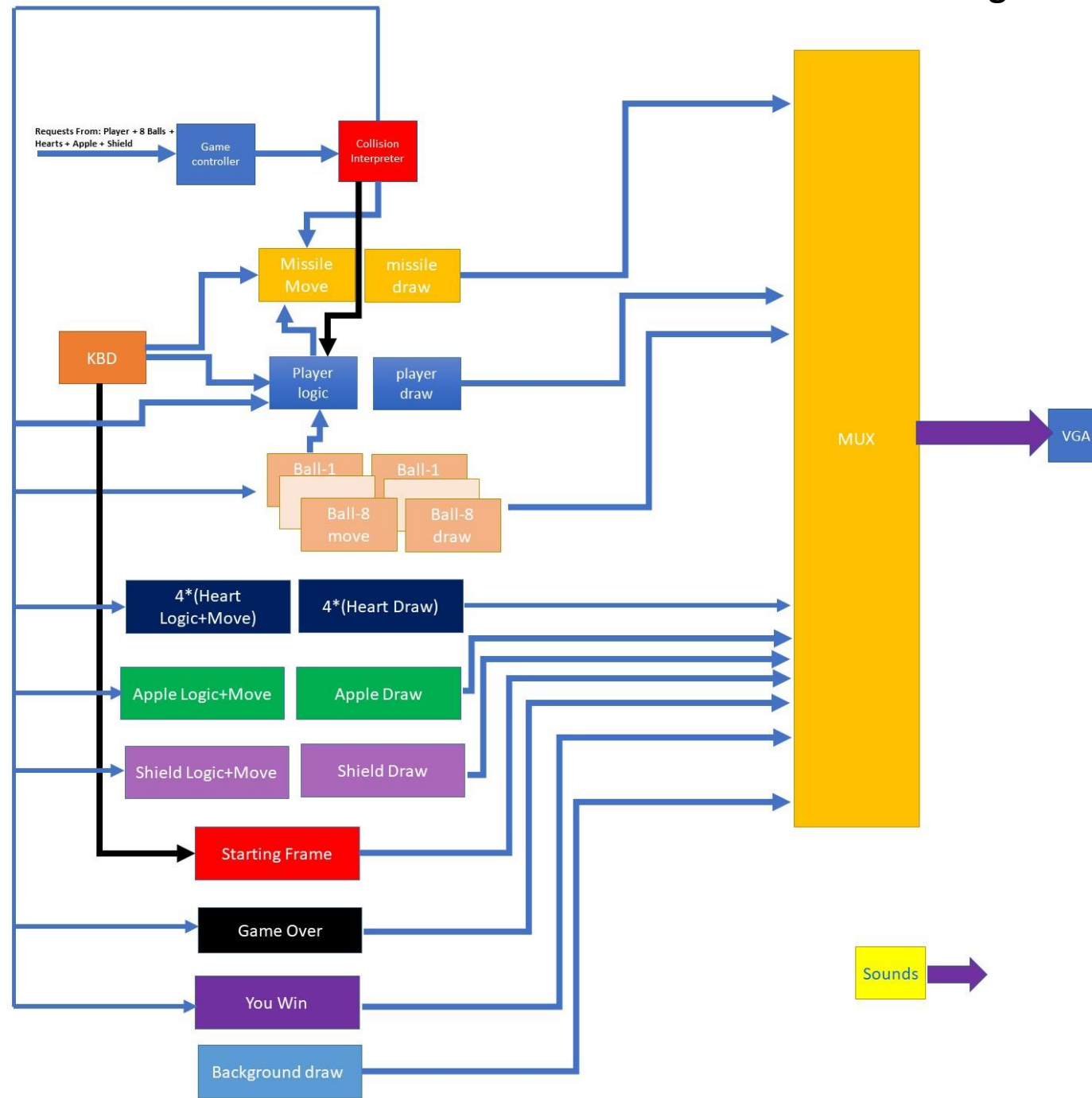
Objects Mux + VGA



MSS



Block Diagram



המקשים של המקלדת

Missile Trajectory

Start The Game



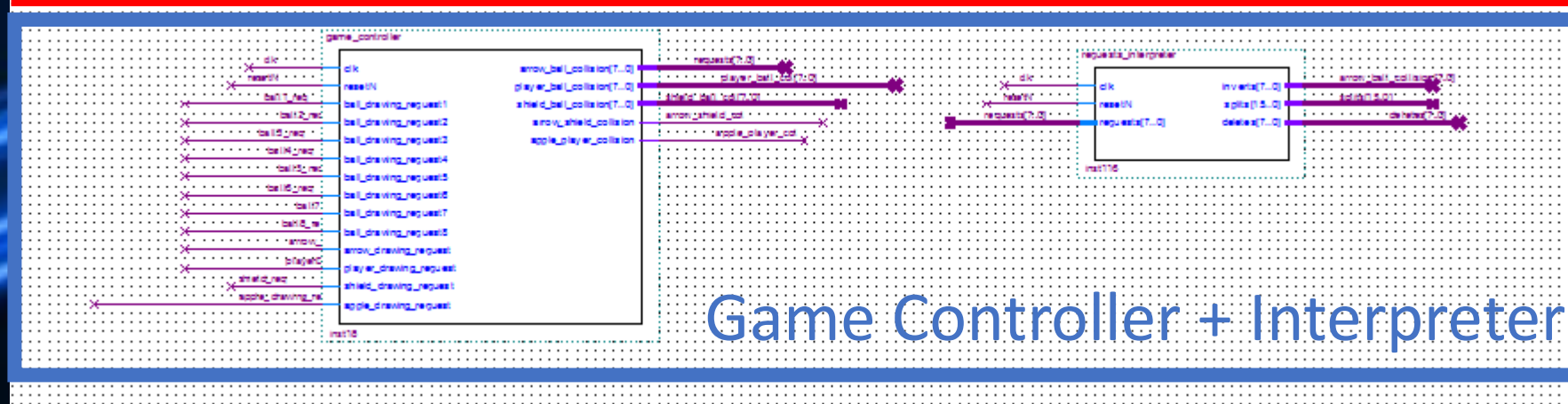
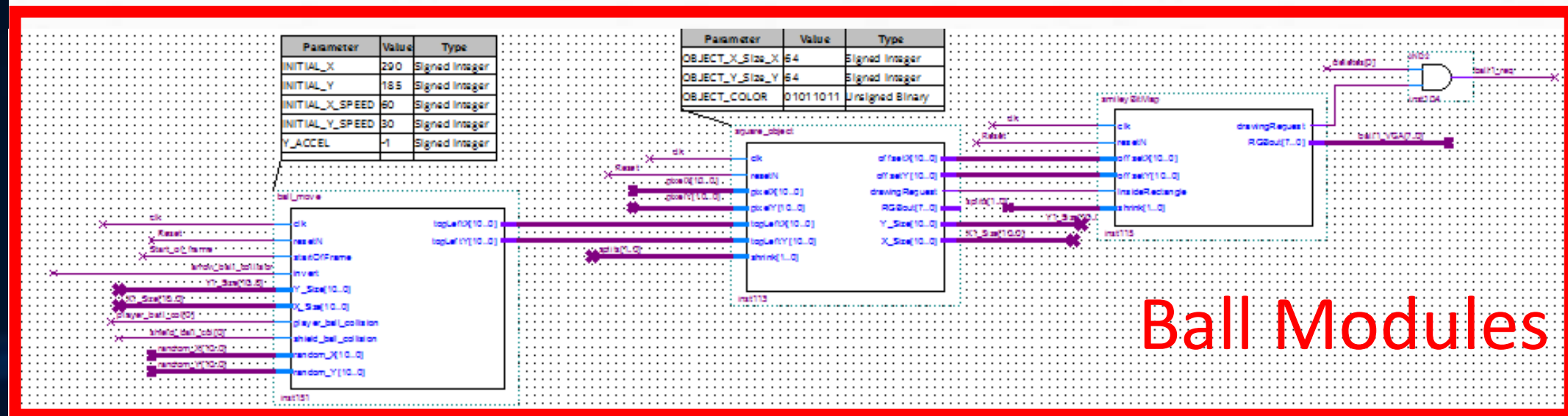
Move The Player To The Left

Move The Player To The Right



דוגמאות לרכיבים מעניינים שמימשנו

הכדורים



הכדורים

רכיב זה הוא בעצם מספר רכיבים המקושרים יחד... רכיבי הכדור מתארים את אופן תנועת הכדור, הגודל שלו ואיך הוא נראה על המסך...
לרכיב זה יש כניסות שמקורן מחיבור שני הרכיבים:

Game Controller + Collision Interpreter

Game Controller

רכיב זה בודק אם יש דרישות כתיבה ממודולים שונים שאמורים להתנגש ומוציא וקטור בגודל 8 סיביות בשם Requests לרכיב הסמוך לו (Requests Interpreter).

Requests Interpreter

מקבל את הוקטור Requests[7..0] ומוציא שני וקטורים :

Inverts[7..0]-Decides whether the ball should invert its moving direction or not.

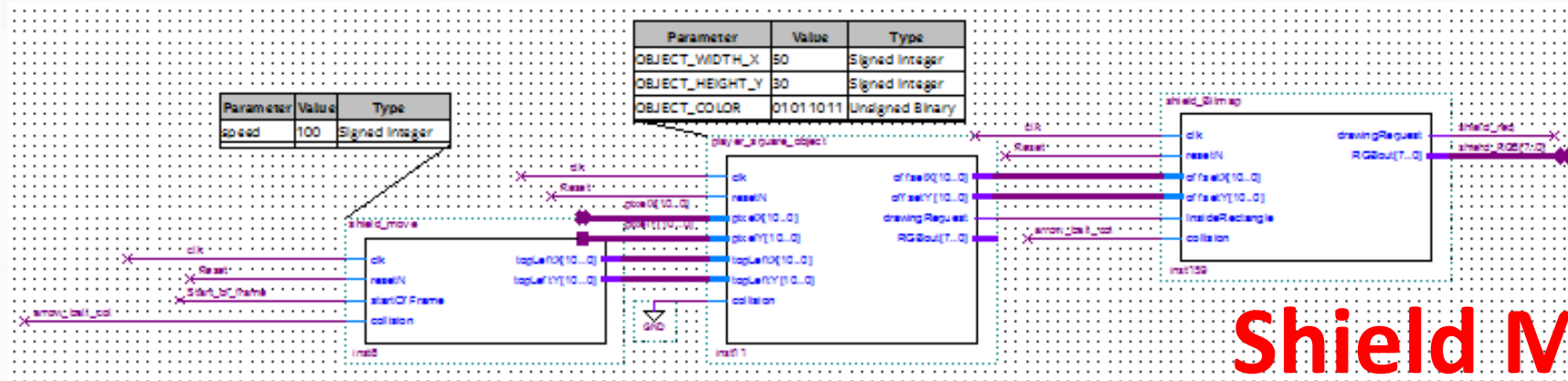
Splits[7..0]-Decides whether the ball's size should be divided by two or not.

הסיביות המתאימות משני הווקטורים האלו נקלטות בכניסות של רכיבי הכדור ולפי ערכם גודל וכיוון הכדור נקבעים.
למשל:

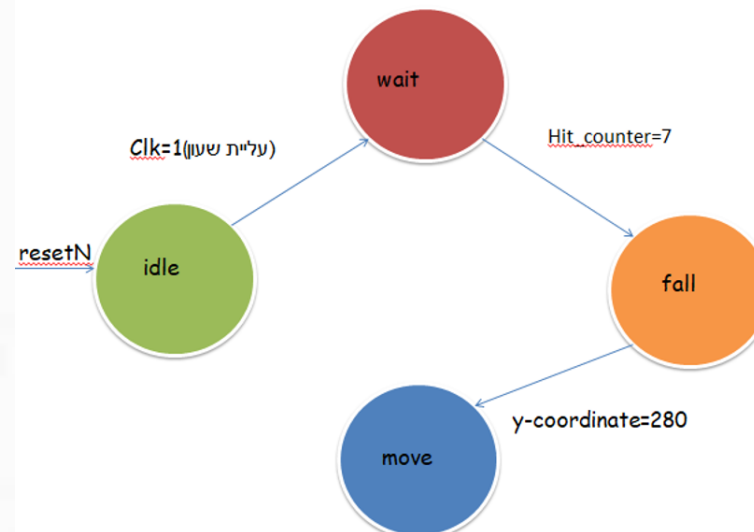
אם יש התנגשות בין החץ לכל הכדורים (ההתנגשות הראשונית):

Requests=8'b11111111 Splits=8'b11111111 Inverts=8'b00001111

Shield



Shield Modules



Shield

שם המצב	פעילות עיקרית	לאיזה מצב עוברים מהמצב הנוכחי ובאילו תנאים
Idle	המצב התחלתי בו טוענים את המיקום ההתחלתי של המכסה ומאפסים המונה	עוברים ל- wait עם עליית השעון
Wait	מחכים במצב זה עד שהשחקן מבצע 7 פגיעות בכדורים	עוברים ל- fall עם עליית השעון ובנוסף שהמונה שווה ל-7 פגיעות
Fall	המכסה מתחיל לרדת על שהוא מגיע למקום הרלוונטי שלו על ציר y	עוברים ל- move עם עליית השעון ובנוסף שהמכסה הגיע למיקום הרלוונטי על ציר Y
move	המכסה מתחיל לנוע ימינה ושמאלה על ציר X	עוברים ל- idle עם ירידת <u>resetN</u> בלבד, כלומר היא ממשיכה במצב זה עד סוף המשחק

כמובן שסיביות ההתנגשויות מגיעות כמוצאי הרכיבים:

Game Controller + Requests Interpreter



Signal Tap

Signal Tap

log: Trig @ 2019/05/18 12:45:56 (0:0:3.0 elapsed)			click to insert time bar				
Type	Alias	Name	-1	0	1	2	3
* 🔍		CLOCK_50					
* 🔍		..._counter:inst165 player_ball_collision					
* 🔍		lives_counter:inst165 counter					
* 🔍		lives_counter:inst165 heart1					
* 🔍		lives_counter:inst165 heart2					
* 🔍		lives_counter:inst165 heart3					
* 🔍		lives_counter:inst165 heart4					

הבאג: הבאג שהיה לנו הוא שבמהלך המשחק כתוצאה מפגיעת הכדור בשחקן הוא היה מפסיד שתי לבבות עבור כל פגיעה במקום להפסיד לב אחד.

אופן הזיהוי: ראינו את זה על המסך במהלך המשחק.

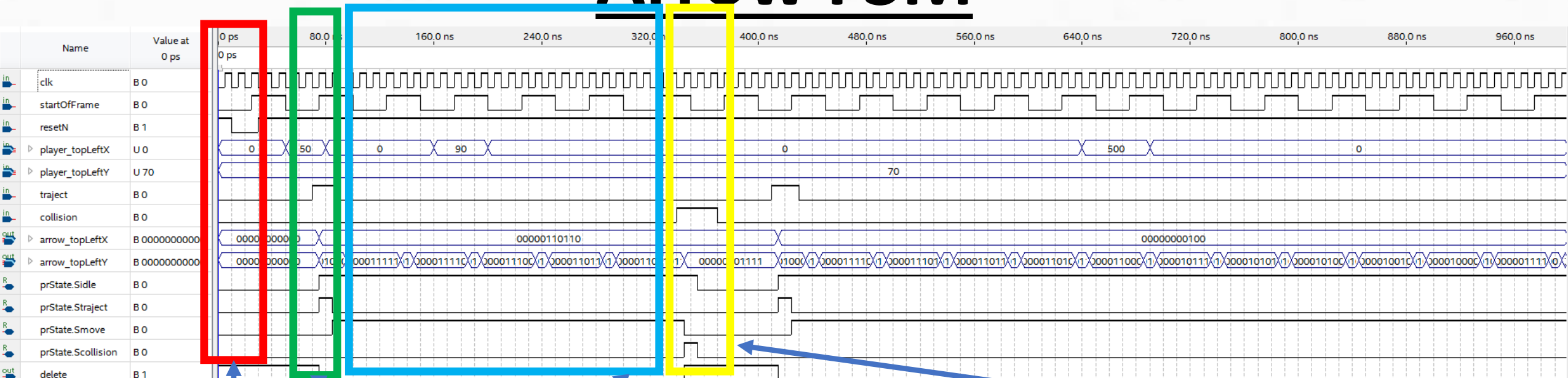
אופן תיקון: ראינו בעזרת ה-signal tap שה- player_ball_collision פעמיים במקום פעם

אחת. תיקנו את זה על ידי הורדת ה- drawing request של השחקן אחרי הפגיעה של הכדור בו וזה פתר את הבעיה.



Simulations

Arrow FSM



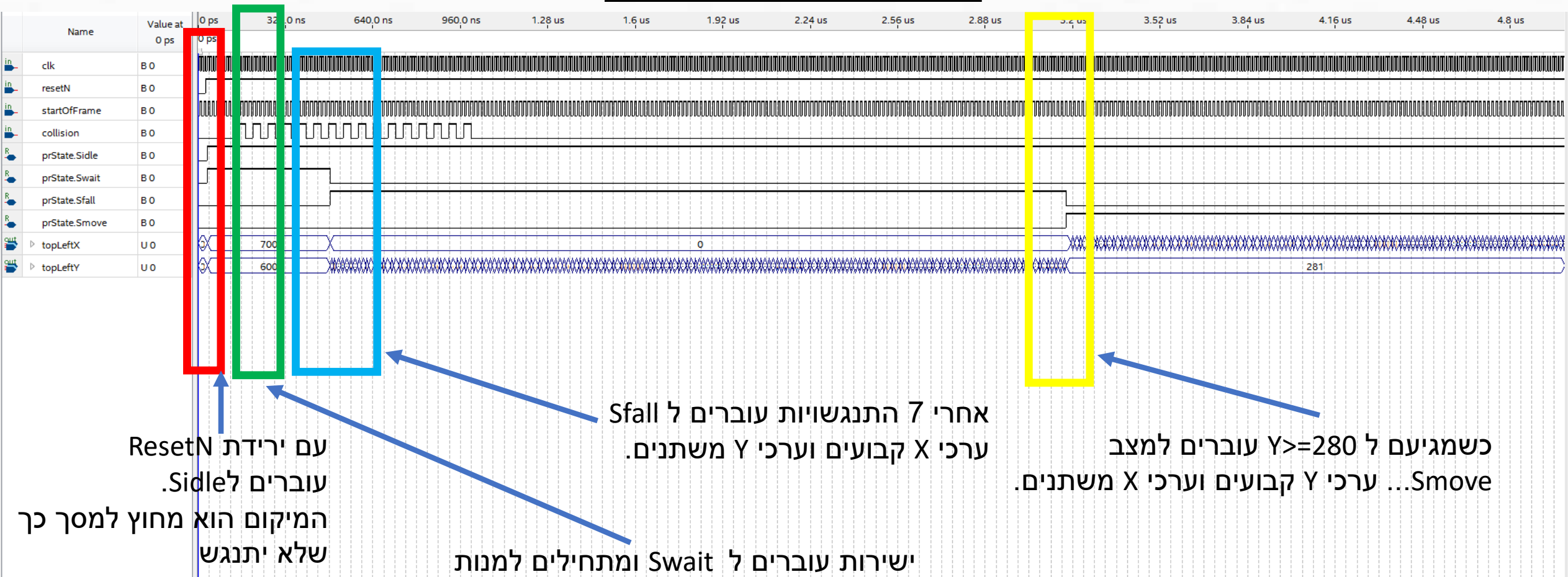
עם ירידת ResetN
עוברים ל'Sidle.
המיקום הוא מיקום השחקן

עם עליית traject עוברים ל'Straject...
המיקום ההתחלתי הוא מיקום השחקן


אחרי Straject עוברים ל'Smove... ערכי X
יציבים וערכי Y משתנים עם כל StartOfFrame

כש- collision עולה עוברים ל Scollision עם
ירידת collision חוזרים ל Sidle | Delete=0

Shield FSM



Compilation

Flow Summary	
 <<Filter>>	
Flow Status	Successful - Mon May 20 21:57:45 2019
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Standard Edition
Revision Name	Lab1Demo
Top-level Entity Name	TOP_VGA_DEMO_WITH_MSS
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	10,841 / 41,910 (26 %)
Total registers	3470
Total pins	50 / 499 (10 %)
Total virtual pins	0
Total block memory bits	20,608 / 5,662,720 (< 1 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

[illegible]



מסקנות

**הצלחנו לעמוד בדרישות כפי שתכננו ובנוסף הוספנו כמה דברים מעניינים בחלק היצירתי.
במהלך העבודה נתקלנו בהמון קשיים, רוב הקשיים היו סבב הקוד שלא עבד כפי שציפינו רוב הפעמים
ולקמפל את הקוד כל פעם לקח המון זמן שהיינו צריכים לחכות.
על מנת לפתור את הבעיות ניסינו להיעזר בכלים שניתנו לנו במהלך הסמסטר כגון: סיגנל טאפ או ביצוע
סימוליה על מנת להבין את הבעיה ולפתור אותה ביעילות וגם לחסוך קומפילציות מיותרות. בפתרון
הבעיות המסובכות שנתקלנו בהם ניסינו לחשב ביחד ולהעלות כמה שאשפר רעיונות על מנת ליעל הקוד
והפתרון לבעיה.**

מסקנות

מסקנות חשובות:

למדנו המון במהלך העבודה שלנו הסמסטר במעבדה:

-נהנינו מאד מלראות ולעבוד במשהו דומה לעבודת מהנדס, כלומר לתכנן לבנות ולחבר רכיבים ביחד על מנת לבנות משהו שיכל לשמש אותנו.

הצלחנו לפתח יכולת לכתוב בשפת ורילוג בצורה טובה מאד, דבר שהתקשינו אתו בעבר ובקורס קודם, כישרון זה ישמש אותנו כמובן בעתיד.

-למדנו איך לפתור בעיות במהלך העבודה שלנו על הפרויקט בצורה חכמה ויעילה



**Thanks For Listening
Hope You've Enjoyed 😊**