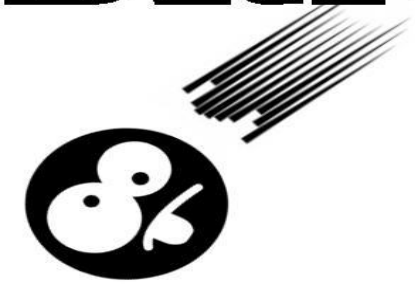
BRICK BREAKER

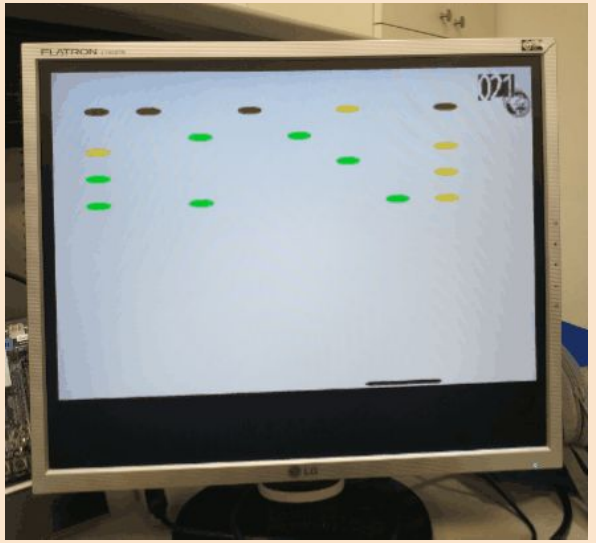


BY

HADAS LEVY AND OMER REUVENI

PRESS ENTER TO START

Brick Breaker



Lab 1A 044157
Technion
August 2019

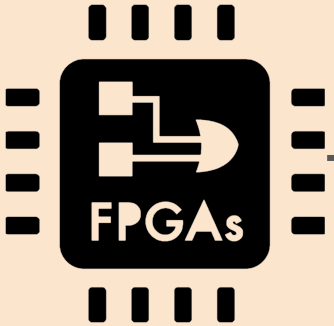
Project by
Hadas Levy && Omer Reuveni

Mentored by Mor Dahan

Project Requirements

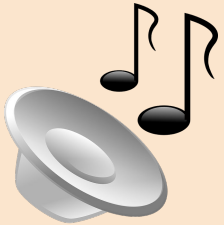
Done	Feature	
☑	שתי לבנים	
☑	לפחות שני צבעים	
☑	כדור נע בקווים ישרים על גבי המגרש וחוזר מהקירות	
☑	מחבט נע בציר X ומחזיר את הכדור כמו מראה	
☑	החזרה מכל צידי הלבנה	
☑	אם כדור נפל - מוגרל כדור חדש	
☑	לוח ניקוד	
☑	צלילי חיווי: כדור נפל, פגיעה במחבט, פגיעה בלבנה	60
☑	צבעים שונים עם "משקל" שונה (ניקוד שונה)	
☑	חיווי ניקוד לכל פגיעה	
☑	טיפול מיוחד בפגיעות מכיוונים שונים של הלבנים	80
☑	שילוב ציורים מיוחדים לכדור	
☑	Live scoreboard	
☑	לבנים מחליפות צבע לפי פגיעה ("דרגה")	
☑	מחבט מתכוון עם התקדמות השלבים	**יצירתיות**

Architecture



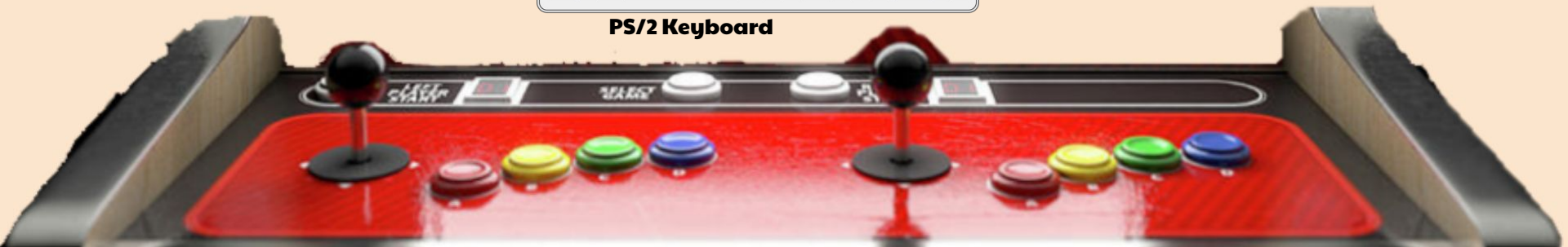
Intel Altera Cyclon

VGA Monitor

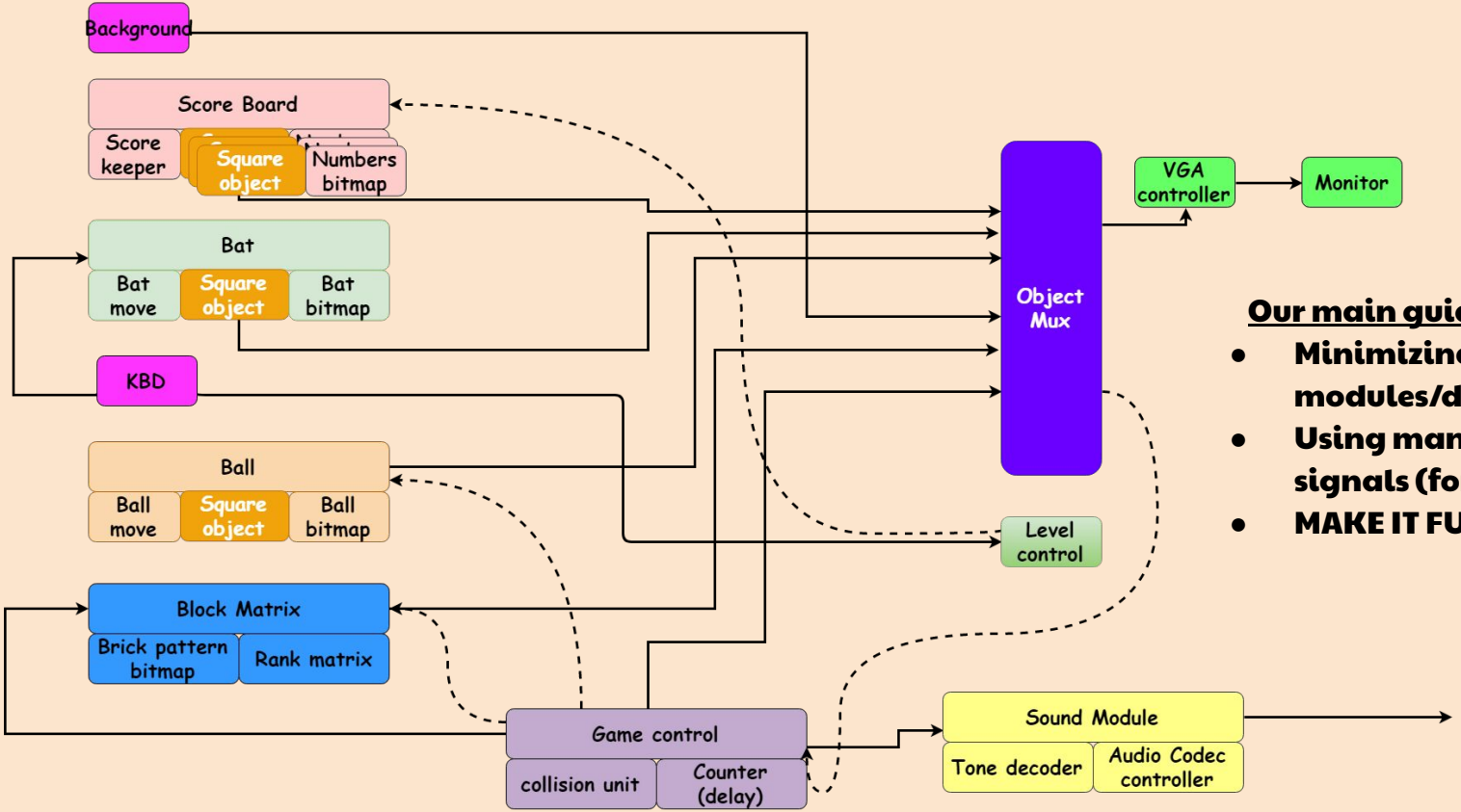


Sound Module

PS/2 Keyboard



Block Diagram



Our main guidelines:

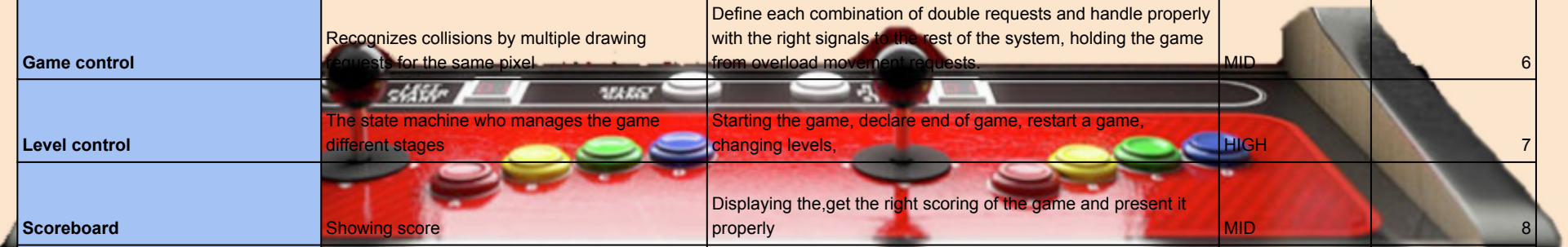
- **Minimizing costly modules/duplications**
- **Using many specific control signals (for debugging ease)**
- **MAKE IT FUN**

Modules Functions and priority

Module Name	Function	Details	complexity	Order of making
Background	The default background of the game	Default color for each pixel unless requested by objects	LOW	1
Ball	Bounce in straight lines “like a mirror” when hitting each other element (wall, brick, bat)	Displaying the ball,stop and change course on time, have a reasonable behavior.	MID	2
VGA controller	Executes the requests from the object mux[RGB] pixel by pixel to create the visible image	Needs to display properly according to the priority we planned	LOW	3
Object Mux	Has to prioritize the requests from the different objects who wants to display on a certain pixel.	Has to prioritize properly and ensure the correct forwarding of RGB_out.	MID	4

Modules Functions and priority

Module Name	Function	Details	complexity	Order of making
Background	The default background of the game	Default color for each pixel unless requested by objects	LOW	1
Ball	Bounce in straight lines “like a mirror” when hitting each other element (wall, brick, bat)	Displaying the ball,stop and change course on time, have a reasonable behavior.	MID	2
VGA controller	Executes the requests from the object mux[RGB] pixel by pixel to create the visible image	Needs to display properly according to the priority we planned	LOW	3
Object Mux	Has to prioritize the requests from the different objects who wants to display on a certain pixel.	Has to prioritize properly and ensure the correct forwarding of RGB_out.	MID	4
Keyboard unit	Keyboard interface recognizes the player’s moves	Arrows for bat move, Enter to start the game, ESC to reset a game.	MID	5
Game control	Recognizes collisions by multiple drawing requests for the same pixel	Define each combination of double requests and handle properly with the right signals to the rest of the system, holding the game from overload movement requests.	MID	6
Level control	The state machine who manages the game different stages	Starting the game, declare end of game, restart a game, changing levels,	HIGH	7
Scoreboard	Showing score	Displaying the,get the right scoring of the game and present it properly	MID	8



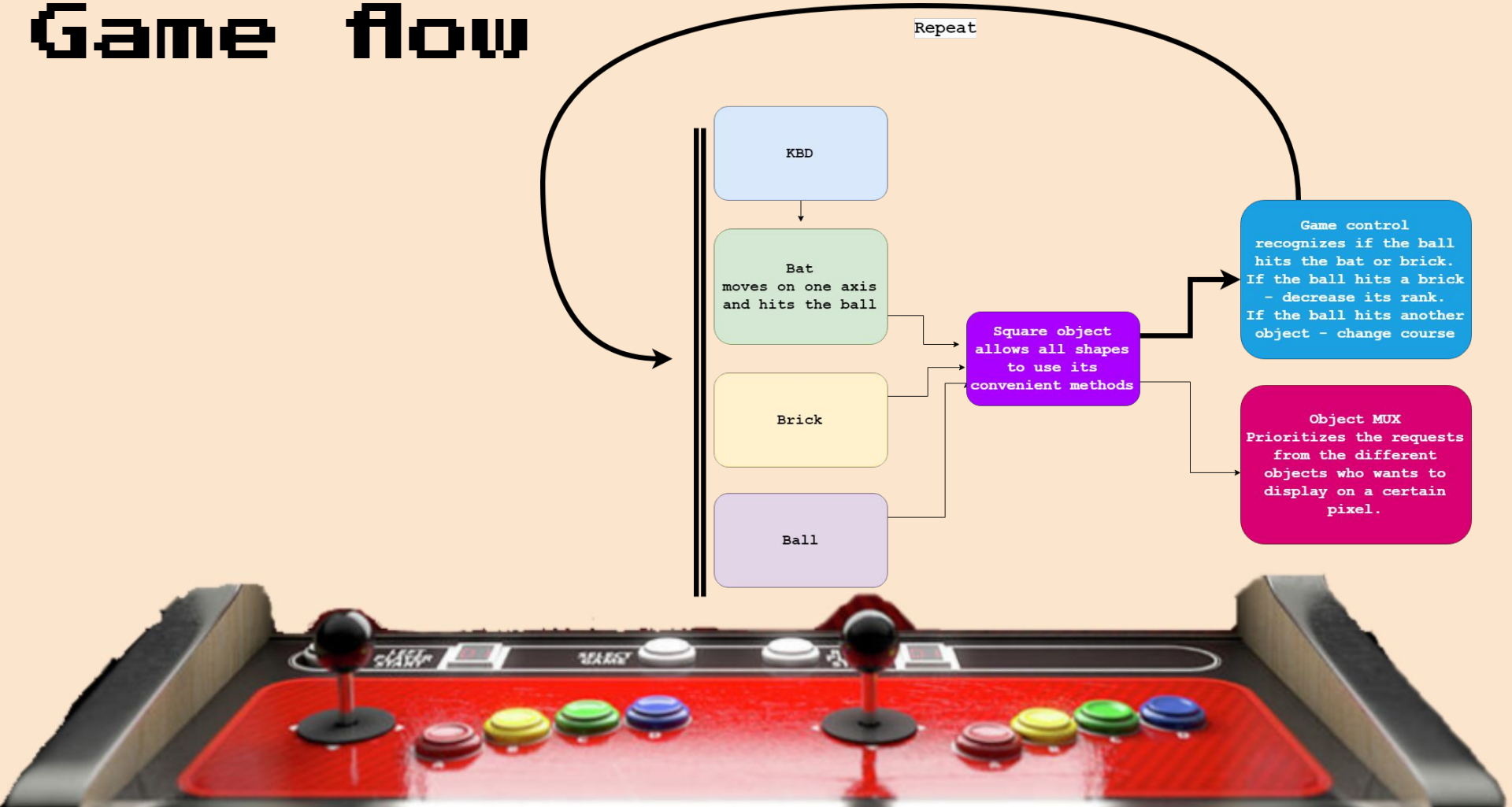
Modules Functions and priority

Module Name	Function	Details	complexity	Order of making
Keyboard unit	Keyboard interface recognizes the player's moves	Arrows for bat move, Enter to start the game, ESC to reset a game.	MID	5
Game control	Recognizes collisions by multiple drawing requests for the same pixel	Define each combination of double requests and handle properly with the right signals to the rest of the system, holding the game from overload movement requests.	MID	6
Level control	The state machine who manages the game different stages	Starting the game, declare end of game, restart a game, changing levels,	HIGH	7
Scoreboard	Showing score	Displaying the,get the right scoring of the game and present it properly	MID	8

Modules Functions and priority

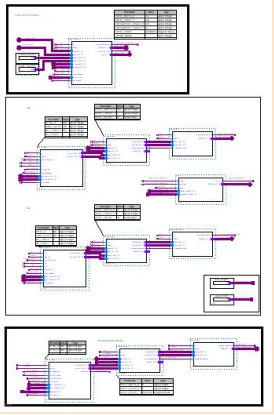
Module Name	Function	Details	complexity	Order of making
Brick	To appear on the right place, react to hit (decreasing rank, sound)	Displaying the,holds its rank, change color	MID	9
Bat	Moves on a single axis, launches the ball properly (required some intentional delay)	Displaying the bat and move properly	HIGH	10
Matrix	Display and manage the bricks, allows different patterns of bricks	Holds: 1 bitmap for the brick shape. 4 Rank matrices for different stages. When ball hits the brick - decreasing the specific brick's rank.	HIGH	11
Counter	We used the counter to create delay,This was our way to ensure we handle only as many requests as we can handle to avoid unwanted behaviors (ex. ball stuck in the corner)		LOW	12

Game flow

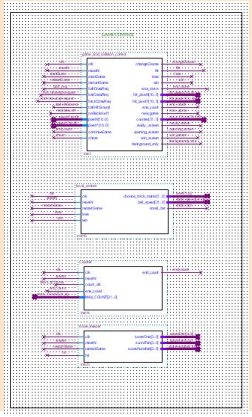

Repeat
KBD
Bat
moves on one axis
and hits the ball
Brick
Ball
Square object
allows all shapes
to use its
convenient methods
Game control
recognizes if the ball
hits the bat or brick.
If the ball hits a brick
- decrease its rank.
If the ball hits another
object - change course
Object MUX
Prioritizes the requests
from the different
objects who wants to
display on a certain
pixel.

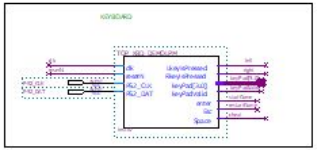
Top Hierarchy

Interactive Modules

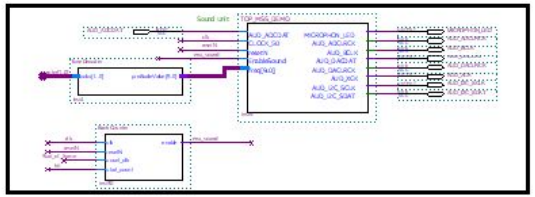


Control unit

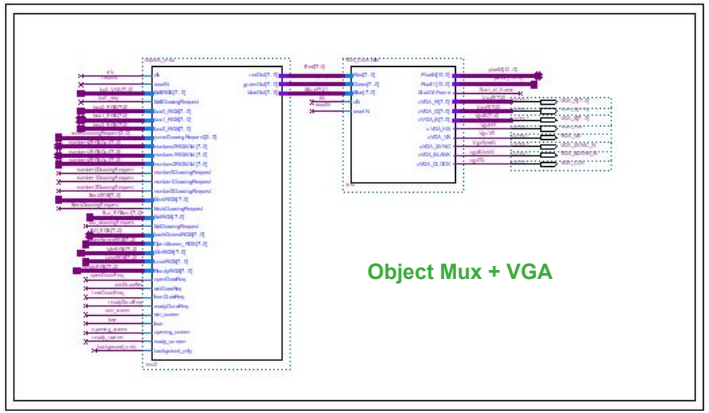




Keyboard



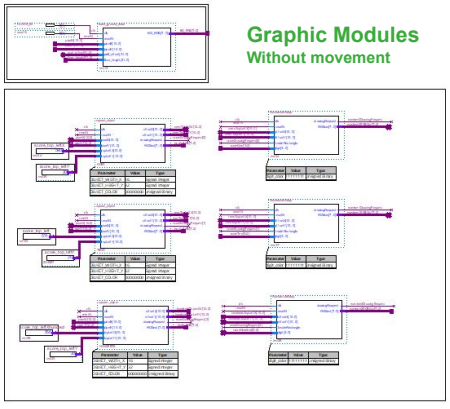
Sound



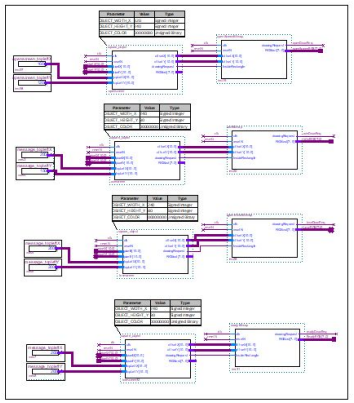
Object Mux + VGA

Graphic Modules

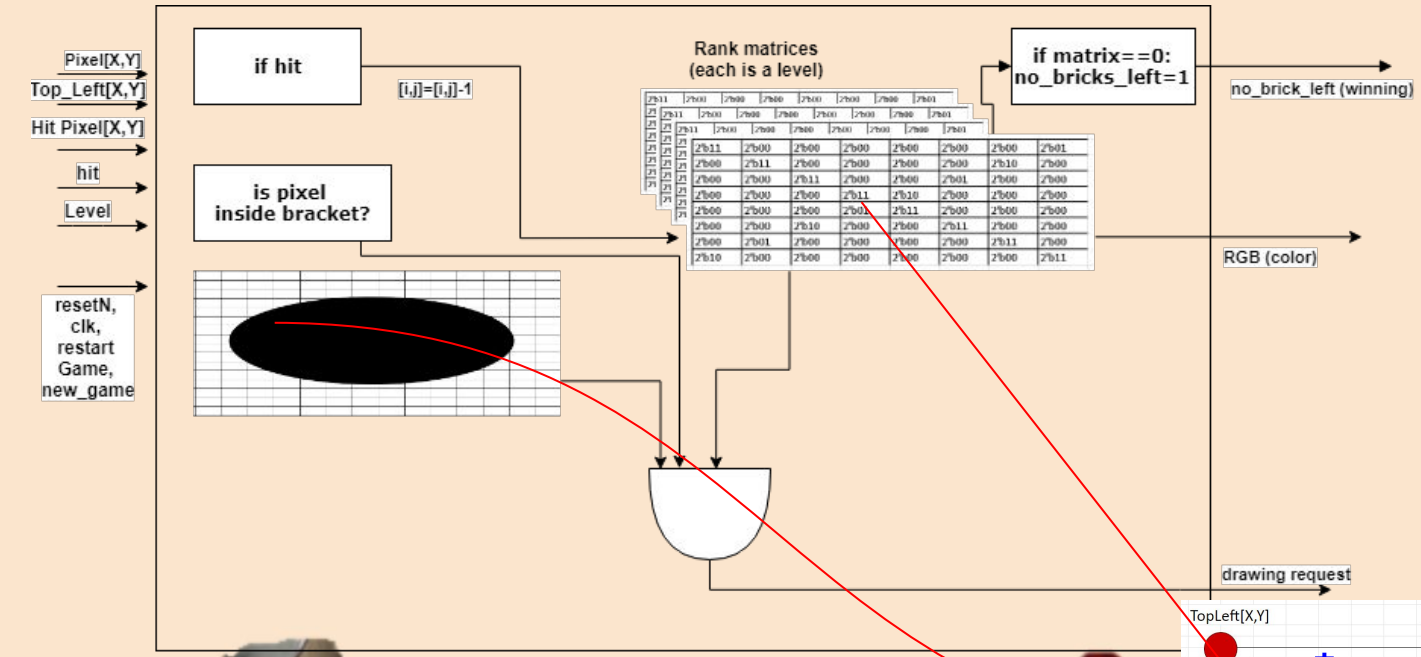
Without movement



Bitmap screens



Matrix Module



Define "happiness"?
Successful Simulation

Flow Summary

<<Filter>>

Flow Status	Successful - Wed Aug 28 10:56:58 2019
Quartus Prime Version	17.0.0 Build 595 04/25/2017 SJ Lite Edition
Revision Name	Lab1Demo
Top-level Entity Name	TOP_VGA_DEMO_WITH_MSS_ALL
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	8,254 / 41,910 (20 %)
Total registers	3699
Total pins	42 / 499 (8 %)
Total virtual pins	0
Total block memory bits	2,097,280 / 5,662,720 (37 %)
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 %)

Main Challenges

1. **Brick matrix**
   It took us more time than expected to understand the full logic of it and making it work properly and efficiently.
   Eventually - our current module allows highly efficient resourcing and is also highly adjustable for any NxM size, brick pattern, number of levels, number of ranks, etc.

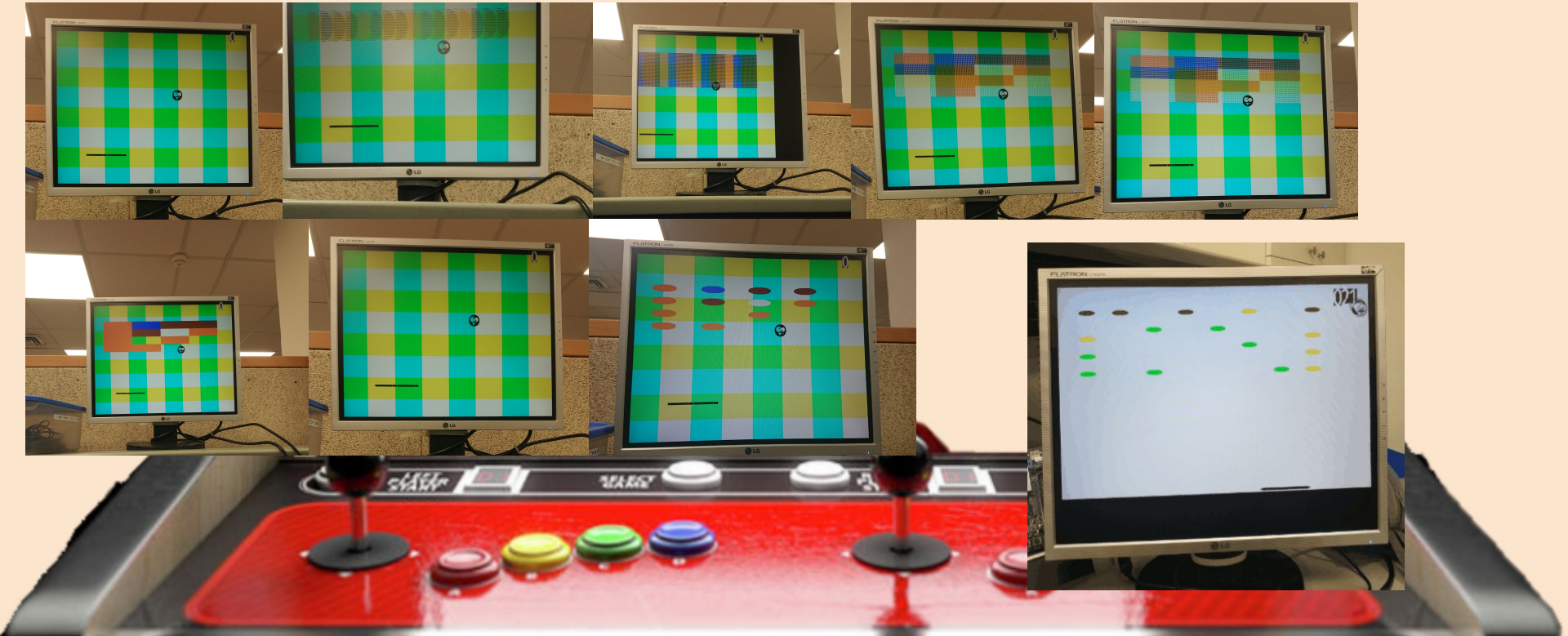
2. **Game controller (Ball)**
   Another challenge we encountered was making the ball hit each object only one time. As the collision detected by the pixel drawing request, it still takes the system another few cycles to react. Over that time there's a detection of hits by the number of pixels the ball got deep in the object.
   We solved it by creating a delay (using counter)and a toggle that allows the system to react to only one event at a time.

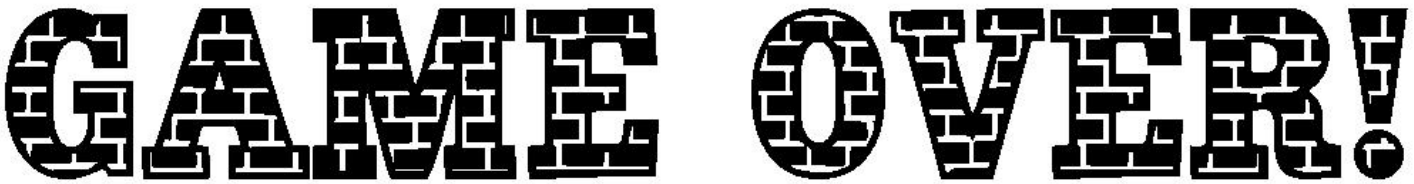
Conclusion/Special Case Studies

We have met all of the requirements for the project (plus extras),
However, we made a lot of unnecessary work due to lack of experience.
Later along the project we spotted some places where we were able to make significant improvements by planning them differently - and so we did.

- **Ex. Replacing multiplications and divisions by shift-reg with powers of 2.**

Evolution of an evening at the lab...





Press ESC to restart