GEBZE TECHNICAL UNIVERSITY

ELECTRONIC ENGINEERING

ELM 457

FPGA SYSTEM DESİGN

PROJECT 2

Berk SARI	–	141024068
Sait Berk VARIMLI	–	141024054
Emre Mert ÖZCAN	–	141024037
Muhammed Taha DUYGU	–	141024055

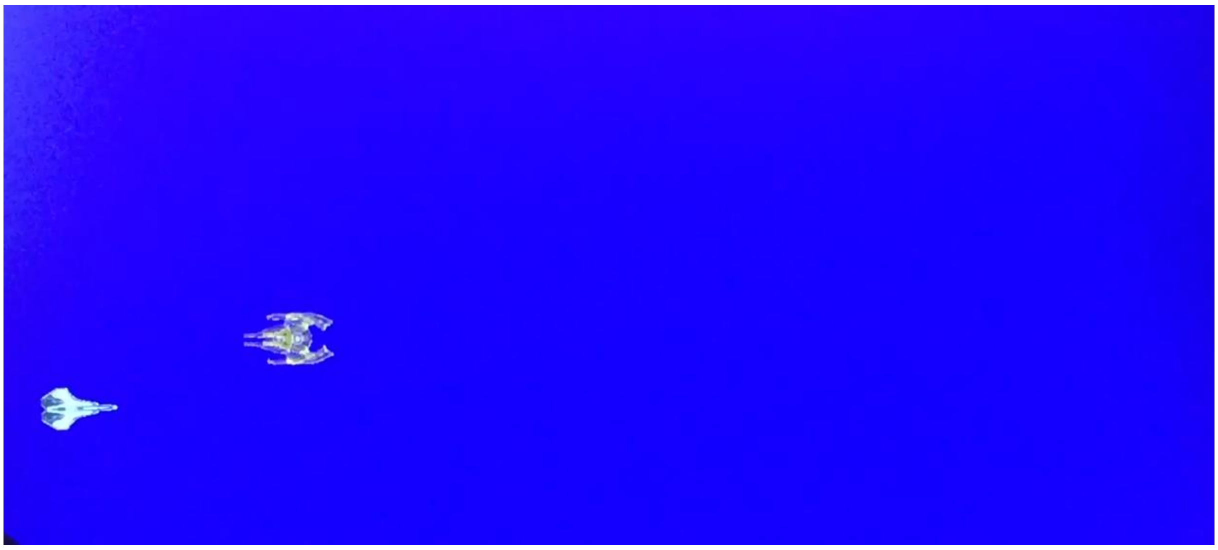
1 – Introduction to our Game

Space Impact is an adventure game that takes place in space. The aim of the game is to escape and survive from the rival spaceships that come to us quickly by making various maneuvers.

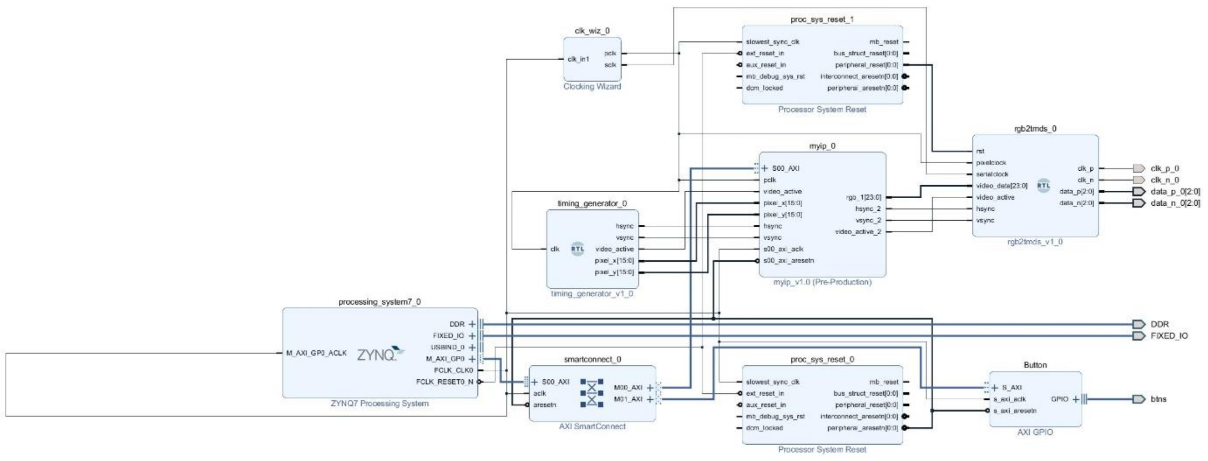
2 – Game Explanation

The ship that belongs to player is located on the left side of the screen. The player's ship can be moved up and down by all the screen with buttons(Button-1/Button-2) on the FPGA board. When the FPGA board programmed with our game and launched, it won't start unles reset/start(Button-3) button pressed on the FPGA board. Enemy ships will spawn at right side of the screen randomly and move to the left.

When the player crash into an enemy ship, the game will stop and it won't start unless reset/start(Button-3) button is "pressed.



<u>3 – Block Design of The Game</u>



AXI Smart Connect, connects button inputs, ‘myip_0’ and processor to each other via bus.

The textures of the game, background color and print order of the objects is in ‘myip_0’ block.

For needed clocks for HDMI and other blocks, 74.25 Mhz and 371.25 Mhz are provided from clock wizard.

Pixel coordinates, the signal that helps to syncranization of screen vertical-sync and horizontal-sync and video active signal is produced in ‘timing_generator_0’.

All the information to draw frames on screen is sent to ‘rgb2tmds’ block. In this block, the information is turned into a way that it can be read on hdmi port. This block is basically HDMI driver.

4 – Game Structure

Objects and textures of the game written in objectbuffer code which is under AXI ip hierarchy. The colors of textures, background color, size objects and draw order is located in this part.

The game logic is written in SDK with C programming language. How buttons work, how objects move, the speed of objects is written in here.

5 – Challenges and Problems

- In the start of the Project, our object's texture sized was too big. We saw huge timing problems. As soon as we lowered our texture sizes, the problem was solved.

- The registers and objects in the AXI ip code has different clocks. Because they have different clock, they won't work syncronized and this causes timing problems. We added clock and used pipelining to prevent timing issues.

- We saw that input part of the 'rgb2tmds' block, not all signal came syncronized. Some signals came from 'myip_0' (our object buffer) and some came from 'timing_generator_0'. This created clock difference and caused timing problem. To solve this, we ran all the signal through to 'myip_0' block and used pipelining in needed place.

6 – Workload Dividing

- Berk Sarı
  - Object Buffer
  - Block diagram
  - AXI ip
  - SDK(C)
  - Constrains

- Sait Berk Varımlı
  - Object Buffer
  - AXI ip
  - Block Diagram
  - Adjusting & Finging Textures

- Emre Mert ÖZCAN
  - Object Buffer
  - Block Diagram
  - Adjusting & Finging Textures

- Muhammed Taha Duygu
  - Object Buffer
  - SDK(C)
  - Adjusting & Finging Textures

Conclusion

With the help of this Project we learned how to create how to use AXI ip, block design, write a logic for the system in SDK application window and run that logic and design on FPGA board.