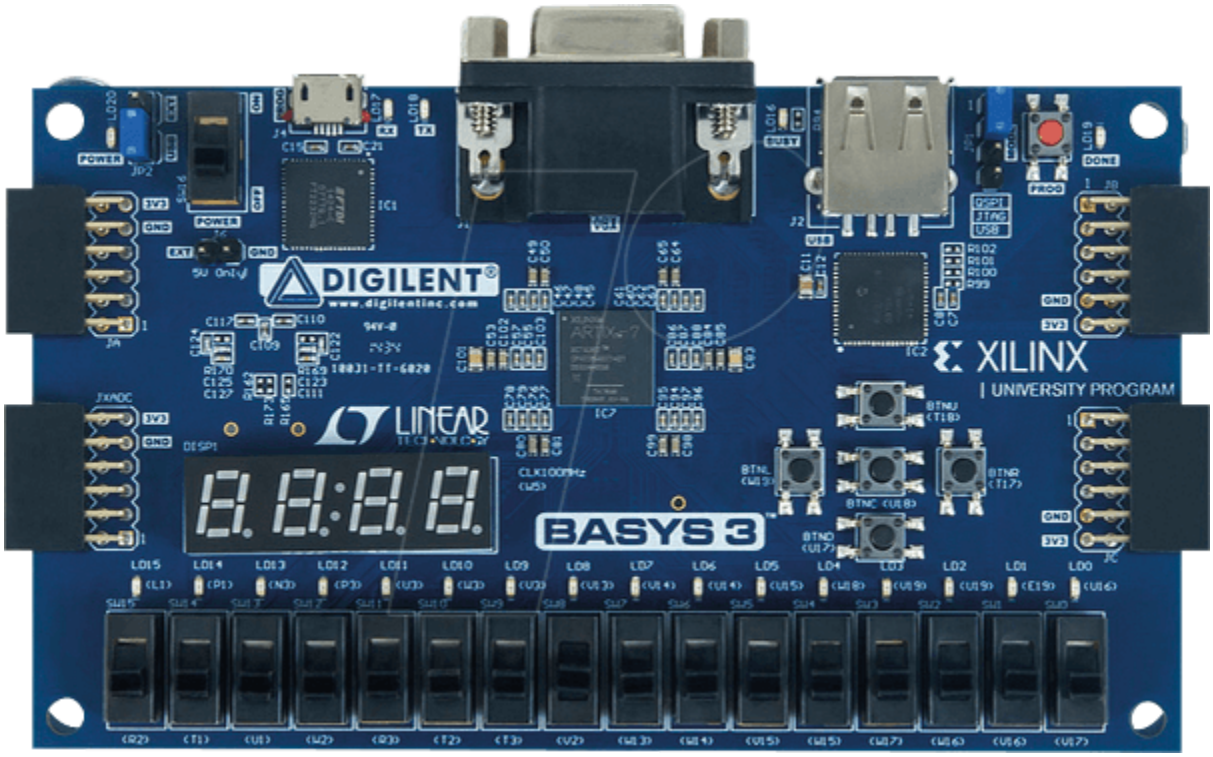
Space-Ship Game

Team Members :
1. *Pranavkumar Mallela : 2020CSB1112*
2. *Akshat Toolaj Sinha : 2020CSB1068*

Project: Space-Ship Game using FPGA

Github Link: https://github.com/akshat1712/CS203-Project_2020CSB1068_2020CSB1112

Description: In this project, we have implemented a spaceship game using Verilog and Vivado software on Digilent Basys3 Xilinx Artix-7 FPGA Board (shown below). The objective of the game is to score as many points as possible. We increase our score by shooting down an led and the score is shown using a 7-segment display. We can also change our game mode from sequential to random or vice versa depending on the user's choice. There is a reset button which resets the current score to 0, and the spaceship to its initial state (state is reset only in the sequential mode).



Inputs:

1. **Cannons** : 16 sliding switches
2. **Reset** : Middle push button
3. **Game Mode 1**: Top push button
4. **Game Mode 2**: Bottom push button

Outputs:

1. **Current score**: First and second seven-segment display from left
2. **Maximum score**: Third and fourth seven-segment display from left
3. **Spaceship** : 3 consecutive lighting LEDs

Functionality:

1. Spaceship:

- **Length:** 3 LEDs long
- **Speed:** Depends on game mode
- **Motion:** Type of motion depends on the game mode the user has selected. We characterize our led pattern with a parameter **state** , which holds the position of the left-most lit led(0-indexing is done from right to left).

Mode 1: In this mode,the spaceship moves 2 leds at a time. Our state machine goes through states 2,4,6,8,10,12,14,2… and so on. Pressing the reset button will make the state 2 instantaneously.

Mode 2: This game is a pseudo-random one. This game mode is implemented with the knowledge that prime numbers occur in random order. So, we add 23,11,17,41 in this order and then add 2 to state modulo 14 ((state%14) +2) to get a state that can be represented on the FPGA. Reset button does nothing in this game mode.

2. Score:

Display of Score: CS: MS (Current score: Maximum score)
Range of score : 0-99
Change: We increase/decrease the score depending on the action:

1. If user misses - decrease score if score is greater than 0
2. If user hit the led: Increase score if score is less than 99
3. User does nothing : Score remains the same

Pressing reset will make the current score 0. We only increase the maximum score if current score exceeds the present maximum score.

3. Game Mode:

We can change the game mode by pressing the respective button on the FPGA.

4. Reset :

We can reset our game by pressing the reset button.

Modules and their Hierarchy:

The order in which they are defined are the order in which the top module / control module instantiates them.

Clk signal is input to each module and hence we have omitted its definition in each module. All operations performed in each module are done at the positive edge of Clk.

Module control :

Inputs:

rst_button (1-bit): stores the state of the push button assigned to reset.
game_mode1_button (1-bit): stores the state of the push button assigned to game mode 1.
game_mode2_button (1-bit): stores the state of the push button assigned to game mode 2.
buttons(16-bit): takes input from the 16 switches on the FPGA board.

Outputs:

led (16-bit): Outputs the state of the 16 LEDs (high or low)
seg(7-bit): denotes the High and low conditions of cathode chosen in 7-segment.
display using each bit as separately as ON/OFF.
an (4-bit): denotes the anode number which we want to access in one-hot encoding.

This module works like a top-level module for FPGA and interacts with FPGA with the help of I/O ports. This module also initializes other internal modules in a specific order and contains all the wires and registers that work with internal modules

Module reset:

Inputs:

button_rst_local (1- bit): tells if reset button is pressed or not

Outputs:

rst (1-bit): variable used to store state of reset button and to send to other modules.

Module game_mode :

Inputs:

game_mode1_button_local (1-bit): tells if game mode 1 button is pressed or not

game_mode2_button_local (1-bit): tells if game mode 2 button is pressed or not

Outputs:

game_mode1 (1-bit): variable to store state of game_mode1_button_local

game_mode2 (1-bit): variable to store state of game_mode1_button_local

Module spaceship_LED_state:

Inputs:

mode1: (1-bit): tells whether mode 1 has been selected

mode2: (1-bit): tells whether mode 2 has been selected

rst: (1-bit): indicates the state of the reset button

Outputs:

led_local: (16-bit): outputs the current state of the 16 LEDs, each bit representing one LED

out_state: (4-bit): indicates the next state of the spaceship, ranges from 0-15

Module check:

Inputs:

state: (4-bit): Indicates the position of the leftmost LED of the spaceship

buttons_local: (16-bit): Variable to read the 15 switch values from the FPGA..

Outputs:

action_type: (2-bit): Indicates whether the user has hit the spaceship or not

00 : indicates user missed the spaceship

01 : indicates user hit the spaceship

10,11 : Indicate don't care values

press: (16-bit): Each bit is a boolean value corresponding to a switch. [15:0] press ensures that no switch value is read 'high' more than once per push.

Module score:

Inputs:

action_type (2-bit) : depicts what type of action is performed (mapping is explained above)

rst (1-bit): indicates the state of the reset button

Outputs:

score_local (*7-bit*)*:* register containing the updated score

max_score_local (*7-bit*)*:* register containing the updated new score

Module display :

Inputs:

score (7-bit): current score of the game.

max_score (7-bit): maximum score of the game.

Outputs:

seg (7-bit): denotes the High and low conditions of cathode chosen in 7-segment display using each bit as separately as ON/OFF.

an (4-bit): denotes the anode number which we want to access in one-hot encoding.

Validation approach :

Definitions:

1. **Correct position:** Any of the three positions where the spaceship is present, i.e state, state-1, state-2.
2. **Wrong position:** Any position that is not a correct position is a wrong position.

Case 1: Firing a cannon at a correct position:
Result 1: If the score is not 99, it will increase by 1. Maximum score will change accordingly (see **Case 3**).

Case 2: Firing a cannon at a wrong position:
Result 2: If the score is not 0, it will decrease by 1. Maximum score will not change.

Case 3: Current score exceeds maximum score:
Result 3: Maximum score is now set to the new maximum score.

Case 4 : Toggling a switch to one and leave it there for some time
Result 4: Score will change by at most one unit, after which the switch takes a don't care value

Case 5 : Firing two cannons simultaneously
Result 5: This can result in three cases, depending on the state of the spaceship:

a. Score increases by 2 units (both switches are pressed at correct positions)
b. Score does not increase(one switch is pressed at a correct position, and the other at a wrong position)
c. Score decreases by 2 units(both switches are pressed at wrong positions)

Case 6: General case of Case 5 - firing multiple cannons simultaneously
Result 6: Each switch toggle is detected only once. This detection of which cannons were fired at correct and wrong positions changes the score accordingly. However, the score does not exceed the range of 0-99.

Case 7: Pressing Reset
Result 7:
Effect on score: Only Current Score is set to 0.
Effect on state:

1. Game mode 1 (Sequential): State is reset to 2, i.e spaceship is reset to rightmost position
2. Game mode 2 (Pseudo-random): No effect on state

Case 8: Pressing the button to change Game Mode:
Result 8: The game mode changes from sequential to pseudo-random or vice-versa, depending on which button is pressed.
Effect on state: The current state of the spaceship remains unchanged as the game mode is switched.

Future Directions:

This spaceship game can be improved further with

1. Speed of spaceship as a parameter in the game.
2. Making mode 2 more random with some more mathematical construct.
3. User can set the number of LEDs to represent the spaceship.

More improvement is left to understanding and creativity of the reader.

Acknowledgement:
We would like to heartily thank Dr. Neeraj Goel (Instructor), Mr. Lalit Sharma (Teaching assistant) and Mr. Badri Satyajaswanth (Teaching assistant) for their guidance and immense help throughout the making of this project without which we would not be able to complete the project within the given timeframe.

Results:
Spaceship game demonstrates the versatile use of FPGA. Our game works well with two game modes and displays the score on the 7-segment display. The user can also reset the game.