Soccer Penalty Game

Milwaukee School of Engineering
Electrical Engineering and Computer Science Department
EE-3921/011 Digital System Design
Zhaoming Qin and Ziwei Chen
11/11/2016

Table of Contents

Problem Description

Traditional soccer penalty game has a lot of limitations, such as: people need a soccer field, could be affected by the weather condition, time consuming and some people who like playing soccer do not actually have time to play. So the Soccer Penalty Game that was designed by us solve all the potential problems that may stop people from playing the game.

The Soccer Penalty Game was designed by creating a microcontroller using *Qsys* in *Quartus 13.0sp1* software. The software was developed and programmed using *Nios II Build Tools for Eclipse*. The microcontroller designed in *Qsys* consists of a CPU, which is a processor; a clock, which provide a clock signal and a reset signal; a VGA controller, a DE series board, a pixel buffer, a Dual Clock FIFO, a character buffer, an alpha blender, which allowed the pixel buffer and the character buffer to be used at the same time; two parallel I/O, one was made 8-bit wide and of type input and it was connected to top level design to SW[6..0]. Another was made 4-bit wide and of type input and it was connected to top level design to KEY[3..0]. In order to make the design fit in the hardware, SDRAM was used; a VGA monitor was used to display the game. The program was a Finite State Machine based program.

The game has a maximum of five rounds. All functions for the system were fully functional, and the project can be considered as a success.

Solution

Overview

This project involved designing a FPGA based video game using Altera DE1 FPGA platform. The objective of this project was to utilize the *Qsys* microcontroller build up tool in *Quartus 13.0sp1* software to customize a microcontroller, and utilize the VGA interface to display the game on a LCD display. The KEYs and SWs on the DE1 board should be able to take the input as player action, and the LCD display should be able to give instruction to the player and display animation for the soccer game. The project should be programed in C using *Nios II Build Tools for Eclipse,* a Finite State Machine was implemented as the structure of the software.

Two players were required to play this game. One player would choose a character as the goal keeper; the other player would choose a different character as the penalty taker. The game has a maximum of five rounds, and during each round one point would be add to the score if the player saves the goal as the keeper, or vice versa. The player to reach three points first will win the game. The game can be restarted after one of the players wins. Detailed instructions are provided to the user during the game.

Block Diagram

The block diagram for the system designed was shown as follow:

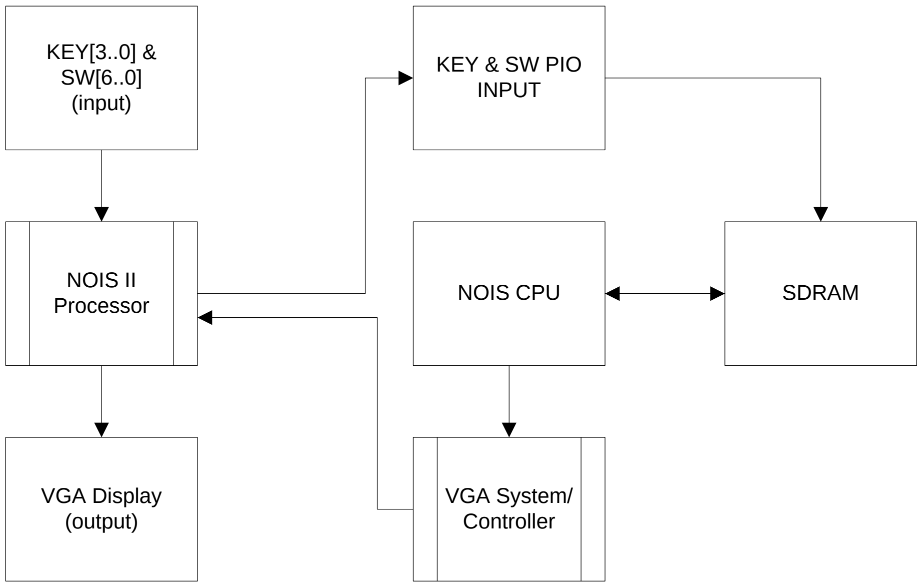


Figure 1: System Block Diagram

Flowchart (Systematic FSM)

Overall

Finite State Machine was used as the program structure. The logic for the finite state machine was shown in the following diagram:

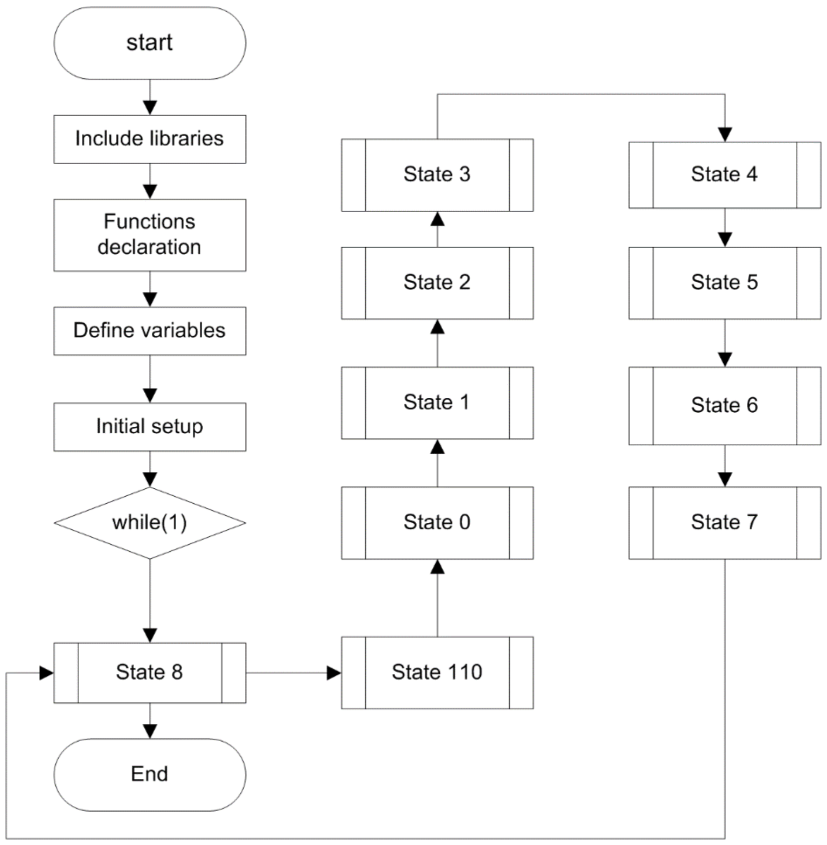


Figure 2: FSM Structure

State 0:

The state 0 is designed to reset all variables. The logic was shown in the following diagram:

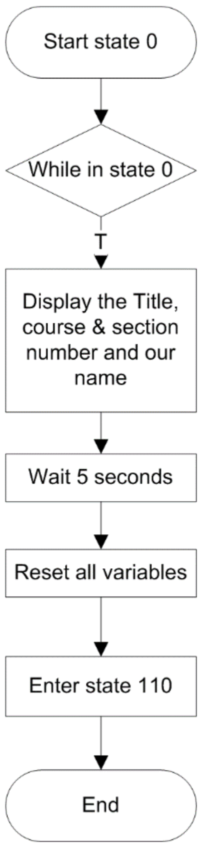


Figure 3: State 0 Structure

State 1:

The state 1 is designed to let the first player to choose a goal keeper from the list. The logic was shown in the following diagram:

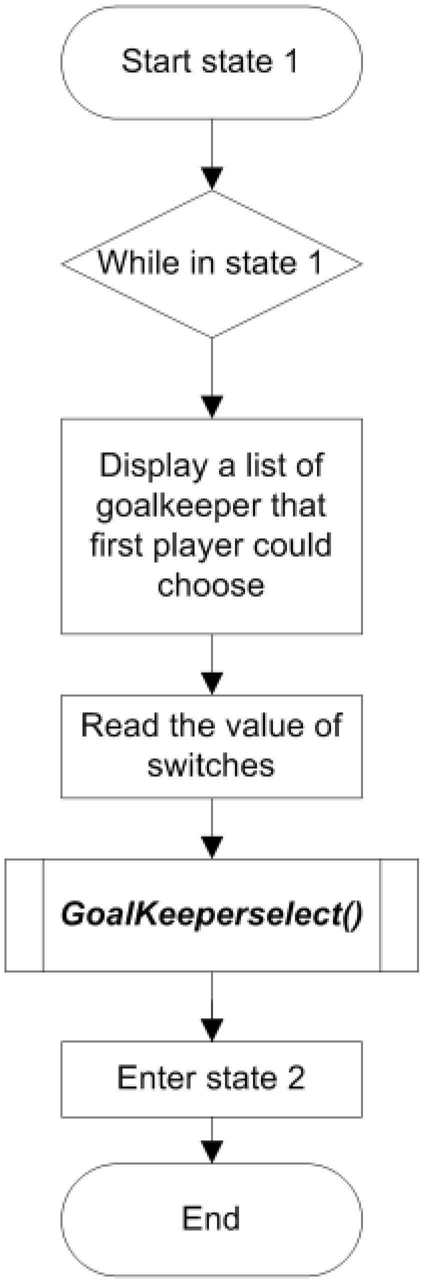

Start state 1
While in state 1
Display a list of goalkeeper that first player could choose
Read the value of switches
GoalKeeperselect()
Enter state 2
End

Figure 4: State 1 Structure

State 2:

The state 2 is designed to let the second player to choose a penalty taker from the list. The logic was shown in the following diagram:

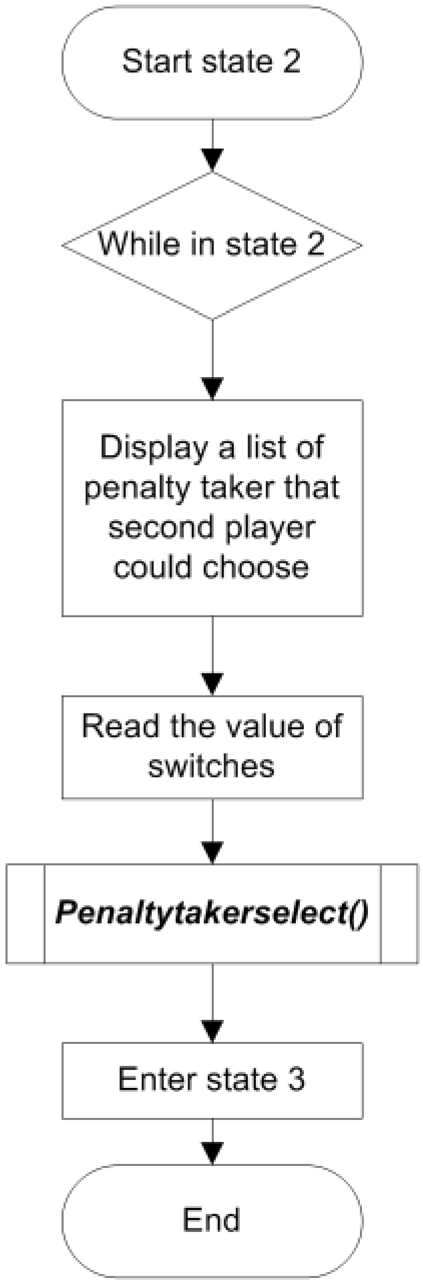

Start state 2
While in state 2
Display a list of penalty taker that second player could choose
Read the value of switches
Penaltytakerselect()
Enter state 3
End

Figure 5: State 2 Structure

State 3:

The state 3 is designed to initialize the game and let the penalty taker choose the direction. The logic was shown in the following diagram:

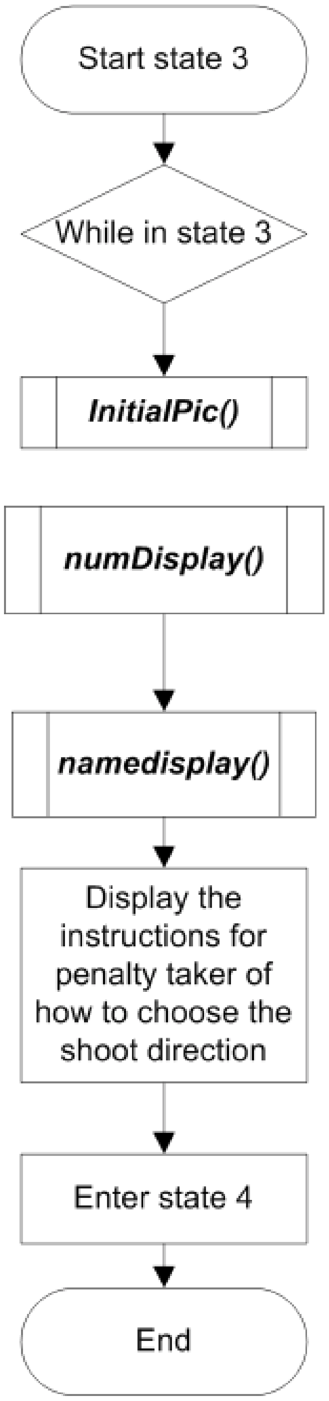


Figure 6: State 3 Structure

State 4:

The state 4 is designed to let the second player to choose the direction. The logic was shown in the following diagram:

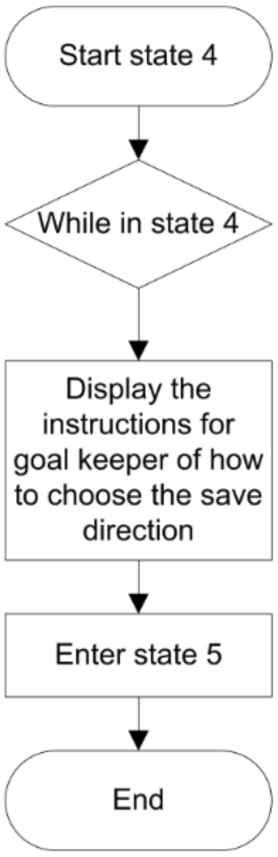

Start state 4
While in state 4
Display the instructions for goal keeper of how to choose the save direction
Enter state 5
End

Figure 7: State 4 Structure

State 5:

The state 5 is designed to calculate the input value for the function ***AnimPlay()***. The logic was shown in the following diagram:

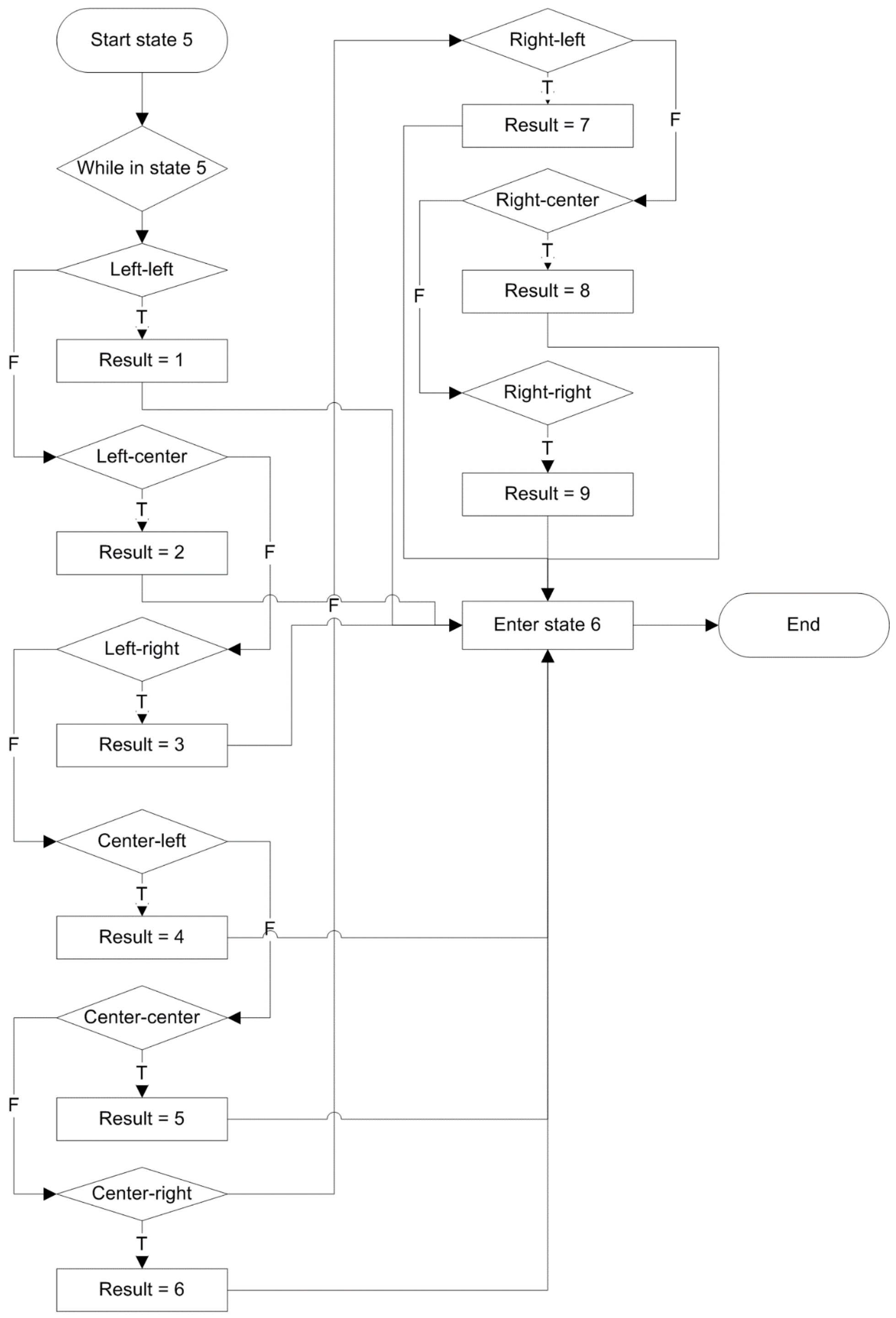


Figure 8: State 5 Structure

State 6:

The state 6 is designed to show the animation and update the score. The logic was shown in the following diagram:

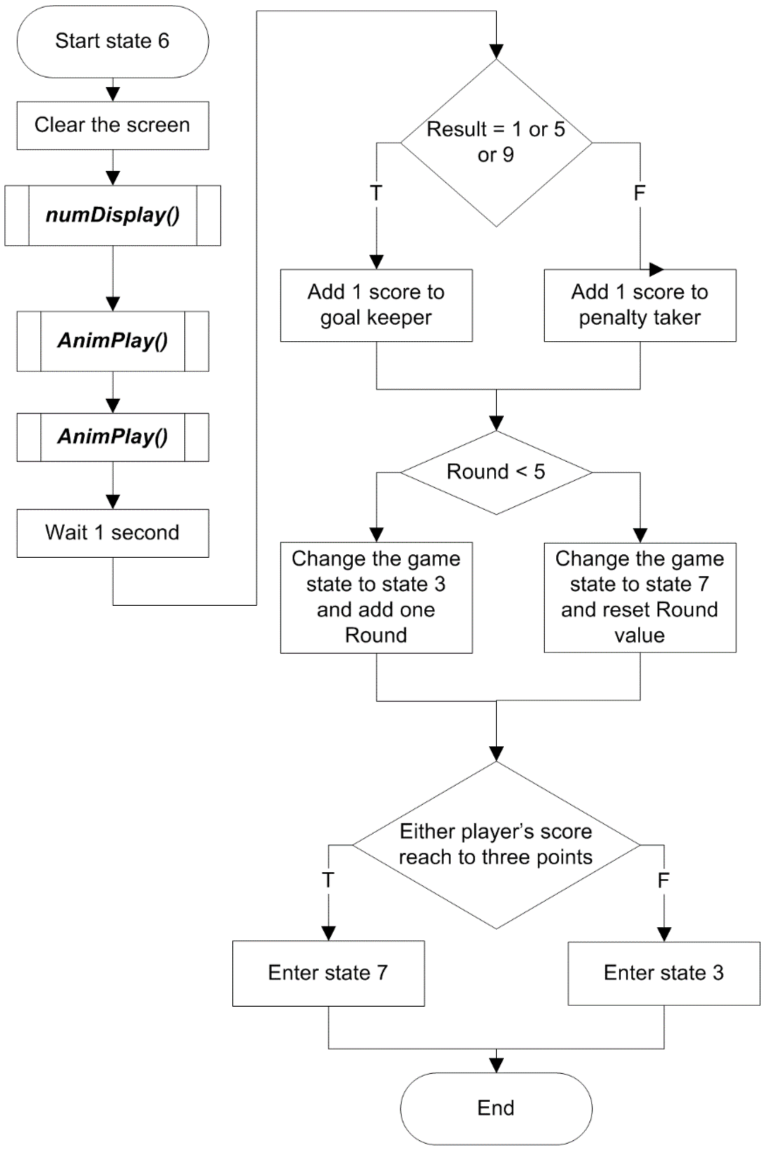

Start state 6
Clear the screen
numDisplay()
AnimPlay()
AnimPlay()
Wait 1 second
Result = 1 or 5 or 9
T
F
Add 1 score to goal keeper
Add 1 score to penalty taker
Round < 5
Change the game state to state 3 and add one Round
Change the game state to state 7 and reset Round value
Either player's score reach to three points
T
F
Enter state 7
Enter state 3
End

Figure 9: State 6 Structure

State 7:

The state 7 is designed to decide which player win the game. The logic was shown in the following diagram:

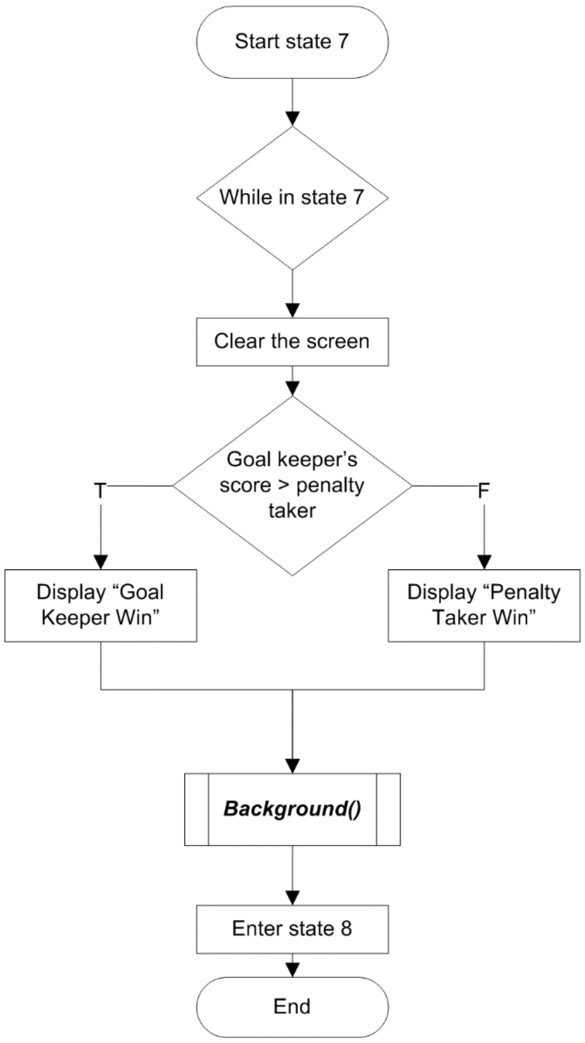

Start state 7
While in state 7
Clear the screen
Goal keeper's score > penalty taker
T
F
Display "Goal Keeper Win"
Display "Penalty Taker Win"
Background()
Enter state 8
End

Figure 10: State 7 Structure

State 8:

The state 8 is designed to be an idle state. The logic was shown in the following diagram:

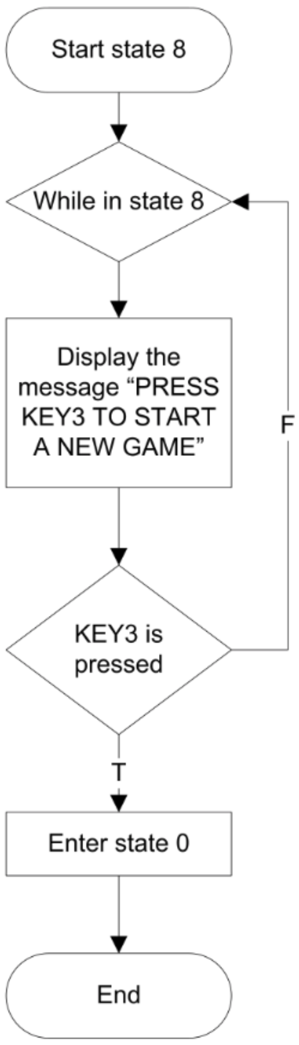


Figure 11: State 8 Structure

State 110:

The state 110 is designed to check if the switches were reset before the user select characters. The logic was shown in the following diagram:

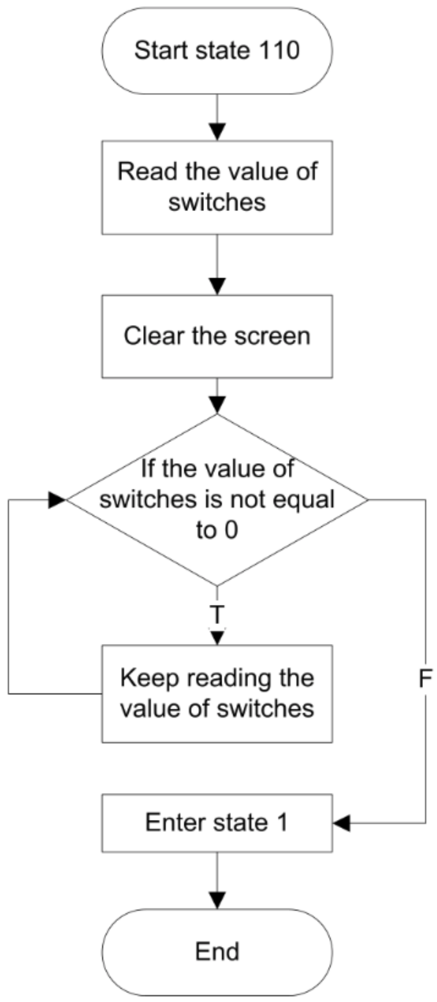

Start state 110
Read the value of switches
Clear the screen
If the value of switches is not equal to 0
T
Keep reading the value of switches
F
Enter state 1
End

Figure 12: State 110 Structure

Explanation of the functions used in the system

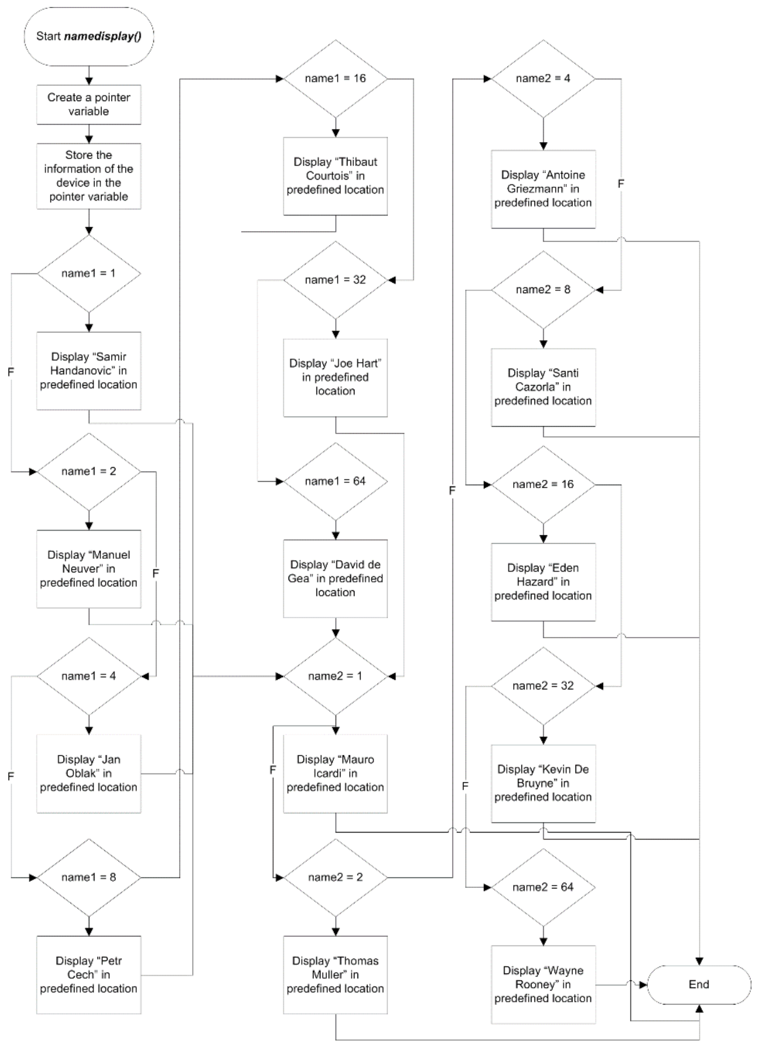
Multiple user defined functions were used in this project.

The function ***namedisplay()*** was designed to display the name of both goal keeper and penalty taker while the game is running.

The input for this function would be the integer value that represent the character, and the output would be displaying the character name on screen.

The character buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_char_buffer_open_dev*** function located in ***altera_up_avalon_video_character_buffer_with_dma.h*** library.

If statement was used as the structure of this function, to select the character base on the input integer value. The logic was shown in the following diagram:



Start namedisplay()
Create a pointer variable
Store the information of the device in the pointer variable
name1 = 1
Display "Samir Handanovic" in predefined location
name1 = 2
Display "Manuel Neuver" in predefined location
name1 = 4
Display "Jan Oblak" in predefined location
name1 = 8
Display "Petr Cech" in predefined location
name1 = 16
Display "Thibaut Courtois" in predefined location
name1 = 32
Display "Joe Hart" in predefined location
name1 = 64
Display "David de Gea" in predefined location
name2 = 1
Display "Mauro Icardi" in predefined location
name2 = 2
Display "Thomas Muller" in predefined location
name2 = 4
Display "Antoine Griezmann" in predefined location
name2 = 8
Display "Santi Cazorla" in predefined location
name2 = 16
Display "Eden Hazard" in predefined location
name2 = 32
Display "Kevin De Bruyne" in predefined location
name2 = 64
Display "Wayne Rooney" in predefined location
End
F

Figure 13: Function ***namedisplay()***

The function ***GoalKeeperselect()*** was designed to display the goal keeper that was chosen by the player at the beginning of the game.

The input for this function would be integer value that represent the character, and the output would be displaying the goal keeper's name on screen.

The character buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_char_buffer_open_dev*** function located in ***altera_up_avalon_video_character_buffer_with_dma.h*** library.

If statement was used as the structure of this function, to select the goal keeper base on the input integer value. The logic was shown in the following diagram:

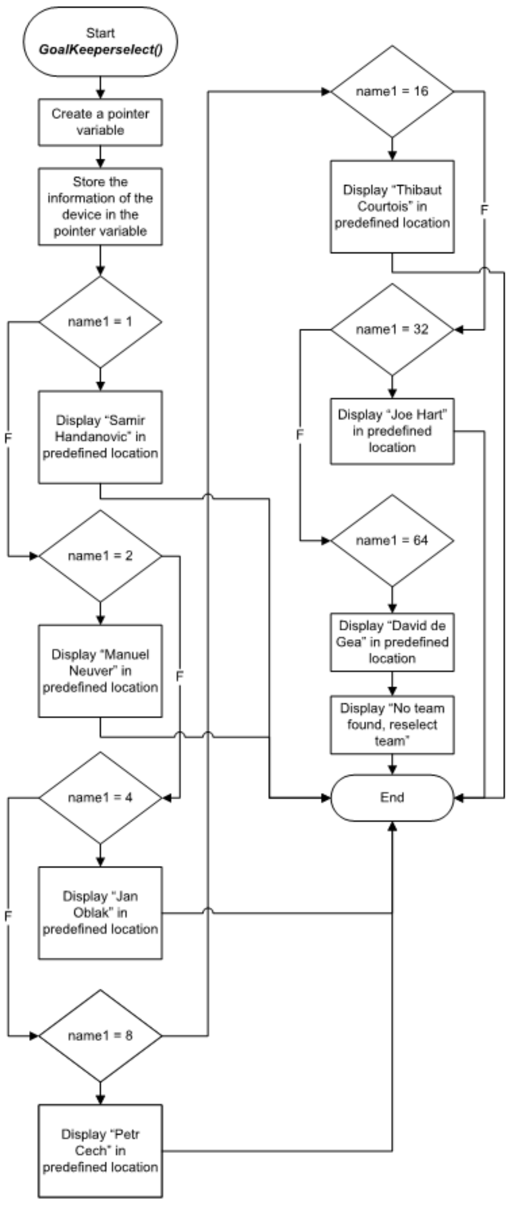

Start
GoalKeeperselect()
Create a pointer variable
Store the information of the device in the pointer variable
name1 = 1
Display "Samir Handanovic" in predefined location
F
name1 = 2
Display "Manuel Neuver" in predefined location
F
name1 = 4
Display "Jan Oblak" in predefined location
F
name1 = 8
Display "Petr Cech" in predefined location
name1 = 16
Display "Thibaut Courtois" in predefined location
F
name1 = 32
Display "Joe Hart" in predefined location
F
name1 = 64
Display "David de Gea" in predefined location
Display "No team found, reselect team"
End

Figure 14 Function ***GoalKeeperselect()***

The function ***Penaltytakerselect()*** was designed to display the penalty taker that was chosen by the player at the beginning of the game.

The input for this function would be integer value that represent the character, and the output would be displaying the penalty taker's name on screen.

The character buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_char_buffer_open_dev*** function located in ***altera_up_avalon_video_character_buffer_with_dma.h*** library.

If statement was used as the structure of this function, to select the penalty taker base on the input integer value. The logic was shown in the following diagram:

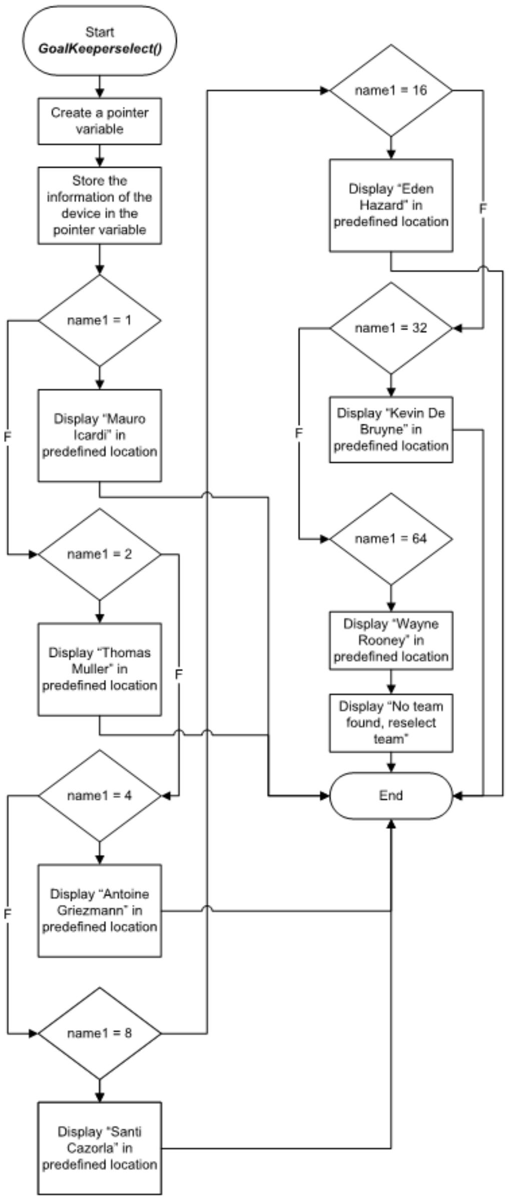

Start
GoalKeeperselect()
Create a pointer variable
Store the information of the device in the pointer variable
name1 = 1
Display "Mauro Icardi" in predefined location
F
name1 = 2
Display "Thomas Muller" in predefined location
F
name1 = 4
Display "Antoine Griezmann" in predefined location
F
name1 = 8
Display "Santi Cazorla" in predefined location
name1 = 16
Display "Eden Hazard" in predefined location
F
name1 = 32
Display "Kevin De Bruyne" in predefined location
F
name1 = 64
Display "Wayne Rooney" in predefined location
Display "No team found, reselect team"
End

Figure 15 Function ***Penaltytakerselect()***

The function ***Background()*** was designed to display an short animation after one player win the game.

This function has void input.

The pixel buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_pixel_buffer_dma_open_dev*** function located in ***altera_up_avalon_video_pixel_buffer_dma.h*** library; ***alt_up_pixel_buffer_dma_draw_box*** was used to show the animation.

If statement was used as the structure of this function, to decide when and which the row should be start to run. The logic was shown in the following diagram:

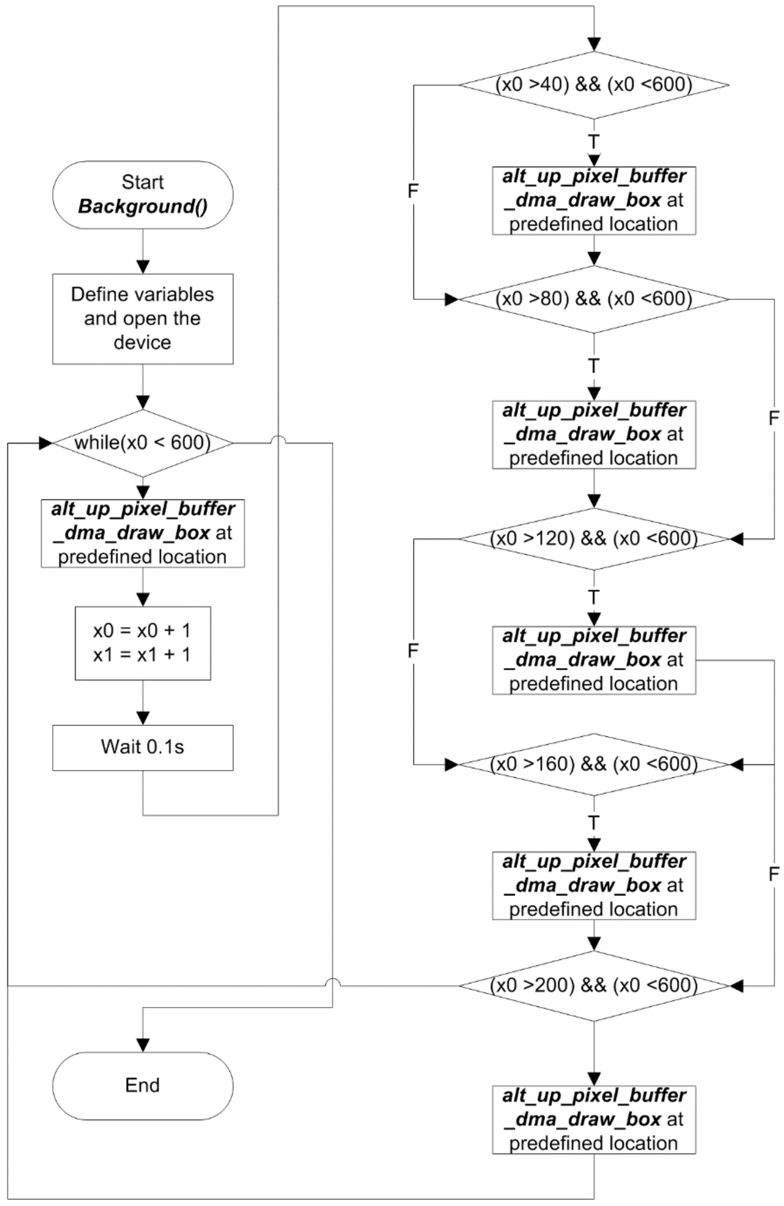


Figure 16 Function ***Background()***

The function ***numDisplay()*** was designed to display the scores of both goal keeper and penalty taker while the game is running.

There are two inputs for this function which both of them are integers. The output would be the scores displayed on the screen.

The character buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_char_buffer_open_dev*** function located in ***altera_up_avalon_video_character_buffer_with_dma.h*** library.

If statement was used as the structure of this function, to select the score based on the input integer value.

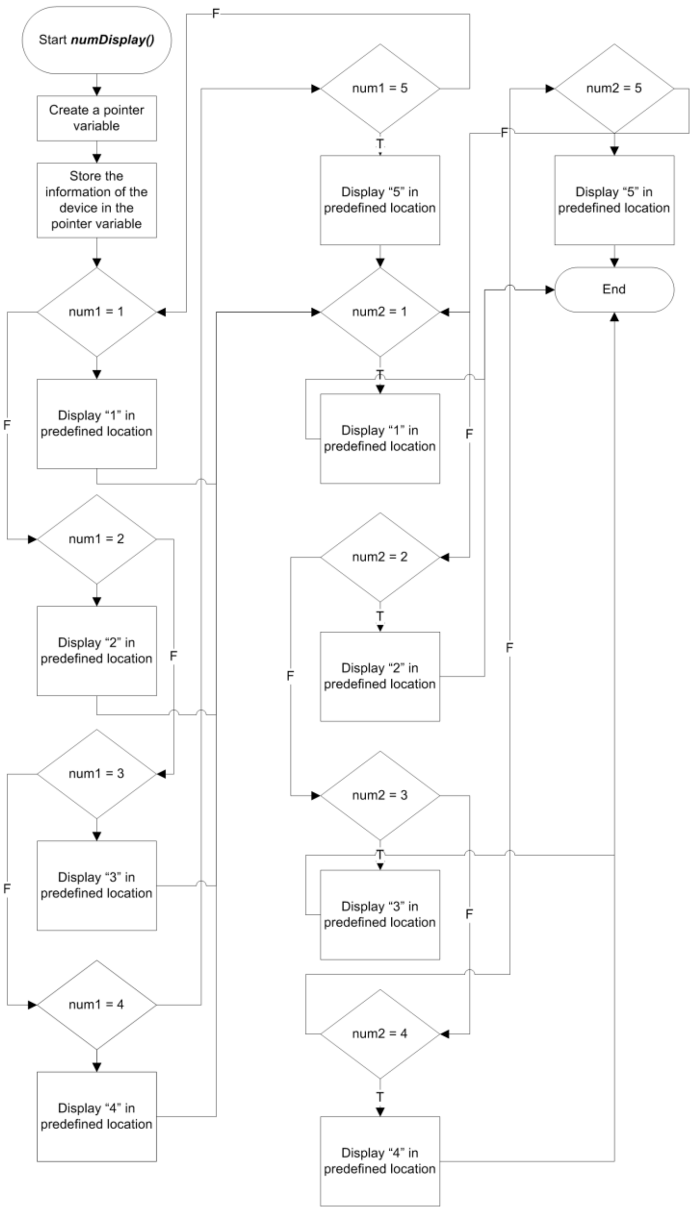

Start numDisplay()
Create a pointer variable
Store the information of the device in the pointer variable
num1 = 1
Display "1" in predefined location
num1 = 2
Display "2" in predefined location
num1 = 3
Display "3" in predefined location
num1 = 4
Display "4" in predefined location
num1 = 5
Display "5" in predefined location
num2 = 1
Display "1" in predefined location
num2 = 2
Display "2" in predefined location
num2 = 3
Display "3" in predefined location
num2 = 4
Display "4" in predefined location
num2 = 5
Display "5" in predefined location
End
T
F

Figure 17 Function ***numDisplay()***

The function ***initialPic()*** was designed to display the initial game frame after the users entering the game.

The function has void input.

The pixel buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_pixel_buffer_dma_open_dev*** function located in ***altera_up_avalon_video_pixel_buffer_dma.h*** library; ***alt_up_pixel_buffer_dma_draw_hline*** and ***alt_up_pixel_buffer_dma_draw_vline*** were used to draw the goal**,** ***alt_up_pixel_buffer_dma_draw_rectangle*** was used to draw the goal keeper, the penalty taker and the ball. The logic was shown in the following diagram:

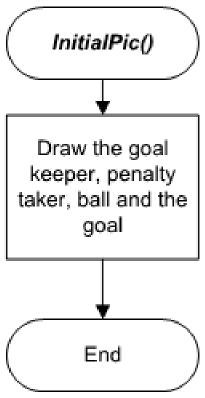


Figure 18 Function **initialPic()**

The function ***AnimPlay()*** was designed to show the goal and save animation after users chose the direction.

The input for this function would be integer value to choose which animation should be played based on the directions chosen by both players.

The pixel buffer was used; a pointer variable was created and store all the information of the device by using the ***alt_up_pixel_buffer_dma_open_dev*** function located in ***altera_up_avalon_video_pixel_buffer_dma.h*** library; ***alt_up_pixel_buffer_dma_draw_hline*** and ***alt_up_pixel_buffer_dma_draw_vline*** were used to draw the goal**,** ***alt_up_pixel_buffer_dma_draw_rectangle*** was used to draw the goal keeper, the penalty taker and the ball.

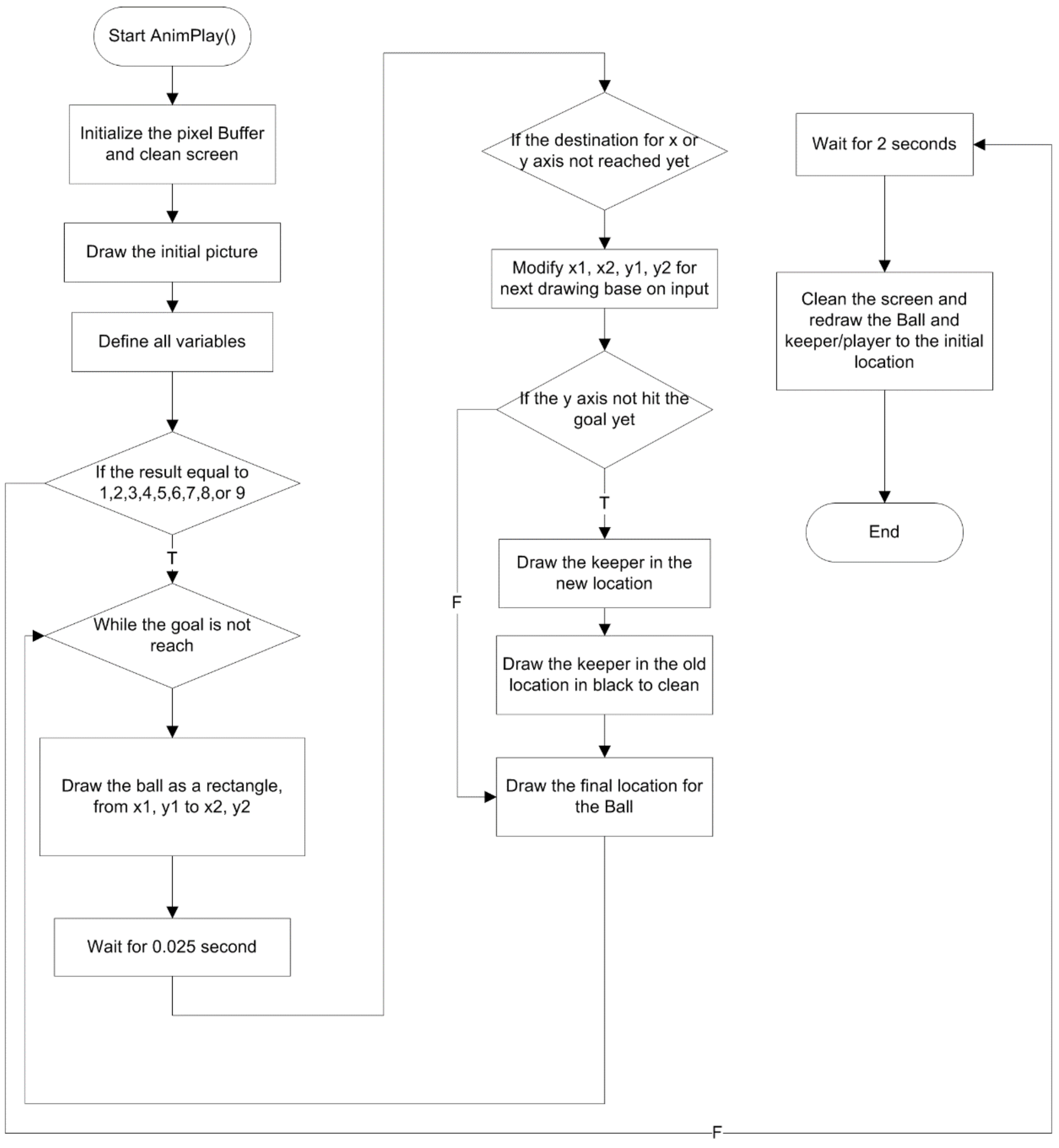

Start AnimPlay()
Initialize the pixel Buffer and clean screen
Draw the initial picture
Define all variables
If the result equal to 1,2,3,4,5,6,7,8,or 9
T
While the goal is not reach
Draw the ball as a rectangle, from x1, y1 to x2, y2
Wait for 0.025 second
If the destination for x or y axis not reached yet
Modify x1, x2, y1, y2 for next drawing base on input
If the y axis not hit the goal yet
T
F
Draw the keeper in the new location
Draw the keeper in the old location in black to clean
Draw the final location for the Ball
Wait for 2 seconds
Clean the screen and redraw the Ball and keeper/player to the initial location
End
F

Figure 19 Function ***AnimPlay()***

Test Description & Data

The functions that need to be tested are shown in the following table:

	Function name
1	***initialPic()***
2	***GoalKeeperselect()***
3	***Penaltytakerselect()***
4	***namedisplay()***
5	***numDisplay()***
6	***AnimPlay()***
7	***Background()***

Table 1: Function list

Test program/procedure design:

The test was performed by running the game for a considerable amount of times. The functions were all tested by simulating different inputs based on user's decisions. The following tables show the functions that was tested, and the test results from testing.

1. ***initialPic()***

Operation	***Expected results***	***Actual results***
Call *initialPic()*	Display the initial picture of the game.	The initial picture of the game was displayed

Table 2: Function Test: ***InitialPic()***

2. ***Background()***

Expected results	***Actual results***
Display the winning animation	Winning animation was displayed

Table 3: Function Test: ***Background()***

3. *GoalKeeperselect()*

Input value from switch	***Expected results***	***Actual results***
0000000	System keep reading the value of KEY.	The choice of you character would not be displayed
0000001	Display “you choose Samir Handanovic”	“you choose Samir Handanovic” was displayed
0000010	Display “you choose Manuel Neuver”	“you choose Manuel Neuver” was displayed
0000100	Display “you choose Jan Oblak”	“you choose Jan Oblak” was displayed
0001000	Display “you choose Petr Cech”	“you choose Petr Cech” was displayed”
0010000	Display “you choose Thibaut Courtois”	“you choose Thibaut Courtois” was displayed
0100000	Display “you choose Joe Hart”	“you choose Joe Hart” was displayed
1000000	Display “you choose David de Gea”	“you choose David de Gea” was displayed
Other values	We do not have a choice confirmation mechanism	It depends on which value would be read by the function first.

Table 4: Function Test: ***GoalKeeperselect()***

4. Penaltytakerselect()

Input value from switch	***Expected results***	***Actual results***
0000000	System keep reading the value of KEY.	The choice of you character would not be displayed
0000001	Display “you choose Mauro Icardi”	“you choose Mauro Icardi” was displayed
0000010	Display “you choose Thomas Muller”	“you choose Thomas Muller” was displayed
0000100	Display “you choose Antoine Griezmann”	“you choose Antoine Griezmann” was displayed
0001000	Display “you choose Santi Cazorla”	“you choose Santi Cazorla” was displayed”
0010000	Display “you choose Eden Hazard”	“you choose Eden Hazard” was displayed
0100000	Display “you choose Kevin De Bruyne”	“you choose Kevin De Bruyne” was displayed
1000000	Display “you choose Wayne Rooney”	“you choose Wayne Rooney” was displayed
Other values	We do not have a choice confirmation mechanism	It depends on which value would be read by the function first.

Table 5: Function Test: ***Penaltytakerselect()***

5. ***namedisplay()***

Since there are 49 combinations of choice, so we only chose 5 combinations randomly to see if it gets the correct output.

Input value	***Expected result***	***Actual result***
1, 1	"Samir Handanovic" and "Mauro Icardi" were displayed on the correct position	"Samir Handanovic" and "Mauro Icardi" were displayed on the correct position
2, 64	"Manuel Neuver" and "Wayne Rooney" were displayed on the correct position	"Manuel Neuver" and "Wayne Rooney" were displayed on the correct position
32, 32	"Joe Hart" and "Kevin De Bruyne" were displayed on the correct position	"Joe Hart" and "Kevin De Bruyne" were displayed on the correct position
16, 8	"Thibaut Courtois" and "Santi Cazorla" were displayed on the correct position	"Thibaut Courtois" and "Santi Cazorla" were displayed on the correct position
8, 2	"Petr Cech" and "Thomas Muller" were displayed on the correct position	"Petr Cech" and "Thomas Muller" were displayed on the correct position
2, 8	"Manuel Neuver" and "Santi Cazorla" were displayed on the correct position	"Manuel Neuver" and "Santi Cazorla" were displayed on the correct position
64, 16	"David de Gea" and "Eden Hazard" were displayed on the correct position	"David de Gea" and "Eden Hazard" were displayed on the correct position
16, 1	"Thibaut Courtois" and "Mauro Icardi" were displayed on the correct position	"Thibaut Courtois" and "Mauro Icardi" were displayed on the correct position
32, 8	"Joe Hart" and "Santi Cazorla" were displayed on the correct position	"Joe Hart" and "Santi Cazorla" were displayed on the correct position

Table 6: Function Test: ***namedisplay()***

6. *numDisplay()*

Input value	*Expected results*	*Actual results*
0, 0	0, 0	0, 0
0, 1	0, 1	0, 1
0, 2	0, 2	0, 2
0, 3	0, 3	0, 3
1, 0	1, 0	1, 0
1, 1	1, 1	1, 1
1, 2	1, 2	1, 2
1, 3	1, 3	1, 3
2, 0	2, 0	2, 0
2, 1	2, 1	2, 1
2, 2	2, 2	2, 2
2, 3	2, 3	2, 3
3, 0	3, 0	3, 0
3, 1	3, 1	3, 1
3, 2	3, 2	3, 2

Table 7: Function Test: **numdisplay()**

7. *AnimPlay()*

Input value	*Goal Keeper's save direction*	*Penalty taker's shoot direction*	*Expected results*	*Actual results*
1	Left	Left	Goal keeper save the goal	Goal keeper save the goal
2	Left	Center	Penalty taker score the goal	Penalty taker score the goal
3	Left	Right	Penalty taker score the goal	Penalty taker score the goal
4	Center	Left	Penalty taker score the goal	Penalty taker score the goal
5	Center	Center	Goal keeper save the goal	Goal keeper save the goal
6	Center	Right	Penalty taker score the goal	Penalty taker score the goal
7	Right	Left	Penalty taker score the goal	Penalty taker score the goal
8	Right	Center	Penalty taker score the goal	Penalty taker score the goal
9	Right	Right	Goal keeper save the goal	Goal keeper save the goal

Table 8: Function Test: **AnimPlay()**

Analysis/Conclusion

This project was to design a soccer video game on a FPGA based digital system. Altera DE1 FPGA platform was chosen as the hardware. *Quartus 13.0sp1* software was used to build up the systemic top-level design by utilizing a block diagram. The Quartus built-in tool *Qsys* was used to build the customized microcontroller that contains SDRAM, PIO inputs, as well as an embedded CPU and the video system. A LCD display was connected as the output of the system using VGA interface. The *Nios II Build Tools for Eclipse* was used to develop the game logic using C language. A finite State Machine was used as the software structure. The game was designed to be played using KEYs and SWs on DE1 board.

The main problems encountered were memory issue, since the on-chip memory was used at the beginning of the project. SDRAM was the used to solve this problem; the problem of the coordinates in the character buffer function, since one character has the length of eight pixels, the largest coordinates should be the resolution divided by eight; the issue of the initial and reset value for KEY. When the KEY are not pressed, the value of KEY should be fifteen, not zero.

The game was tested to be fully functional. The knowledge of the development for a customized microcontroller was gained during this project, as well as the utilization of video system. The knowledge of SDRAM has also been learned.

Appendix

Task List:

The following table shows the task in this project and who was assigned.

	Task	***Performed by***
1	*Identify the project idea*	*C.Z & Q.Z*
2	*Identify the Top Level Design*	*C.Z & Q.Z*
3	*Identify the Qsys Component*	*C.Z & Q.Z*
4	*Identify the C Program Logic*	*C.Z & Q.Z*
5	*Implementation for Basic Qsys Video System*	*C.Z*
6	*Implementation for Pixel Buffer*	*C.Z*
7	*Implementation for Char Buffer*	*Q.Z*
8	*Implementation for SDRAM*	*Q.Z*
9	*Implementation for Top Level Design*	*C.Z*
10	*Implementation for C -- Main Structure*	*C.Z*
11	*Implementation for C – function* ***namedisplay()***	*Q.Z*
12	*Implementation for C – function* ***GoalKeeper()***	*Q.Z*
13	*Implementation for C – function* ***Penaltytakerselect()***	*Q.Z*
14	*Implementation for C – function* ***Background()***	*Q.Z*
15	*Implementation for C – function* ***numDisplay()***	*Q.Z*
16	*Implementation for C – function* ***initialPic()***	*C.Z*
17	*Implementation for C – function* ***AnimPlay()***	*C.Z*
18	*System Level Debugging*	*C.Z & Q.Z*
19	*System Test*	*C.Z & Q.Z*
20	*Presentation and Demo*	*C.Z & Q.Z*
21	*Final Report*	*C.Z & Q.Z*

Table 9: Task List

VHDL Design (Top Level Block Diagram)

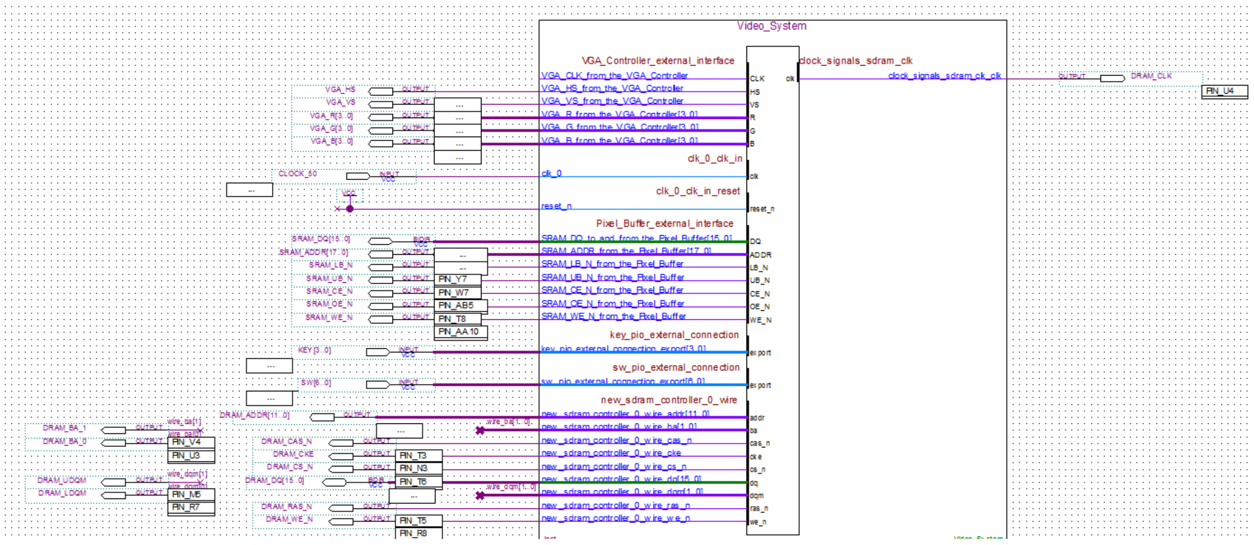
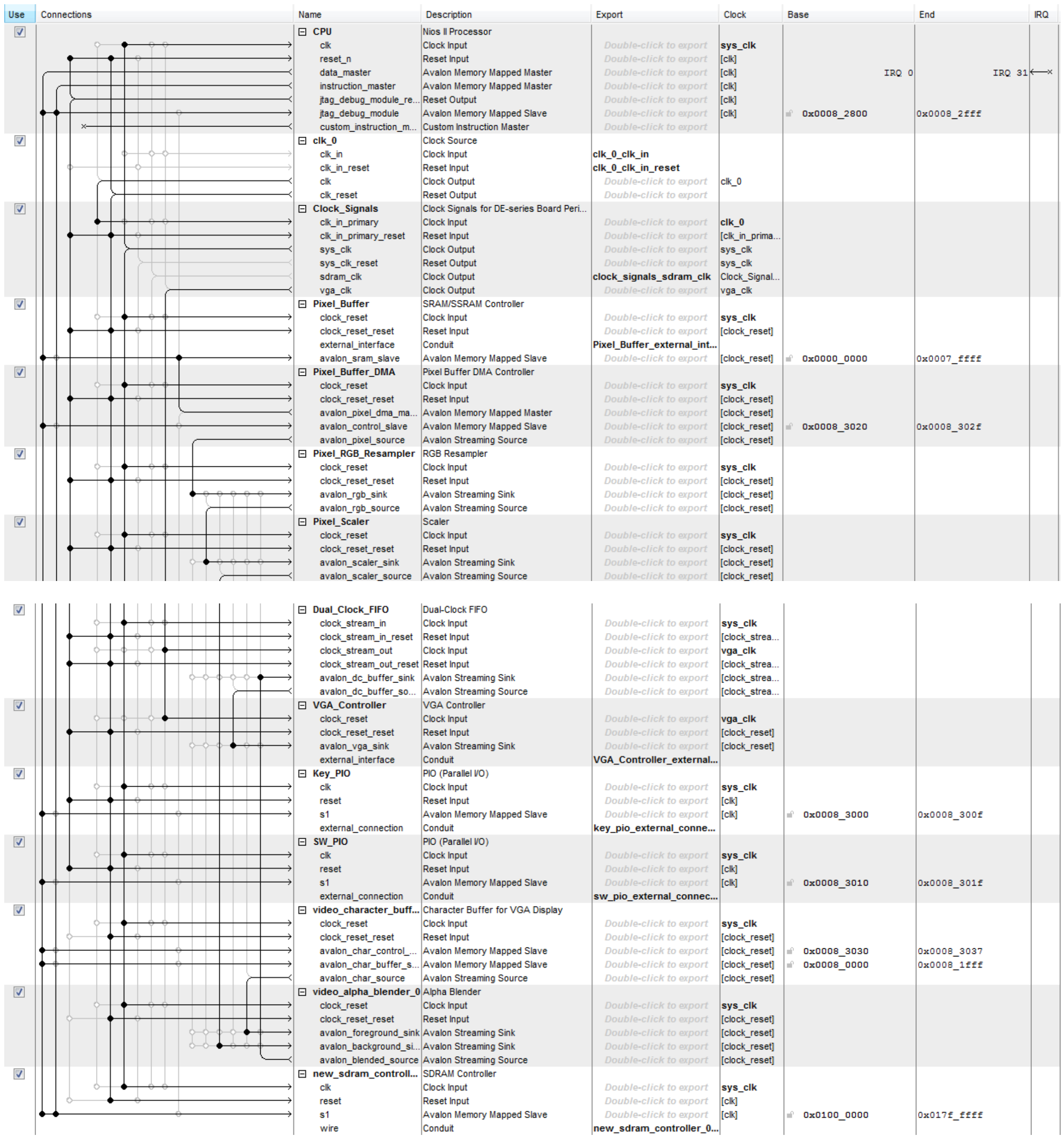


Figure 20: Top Level Block Dagram

Soccer Penalty Game

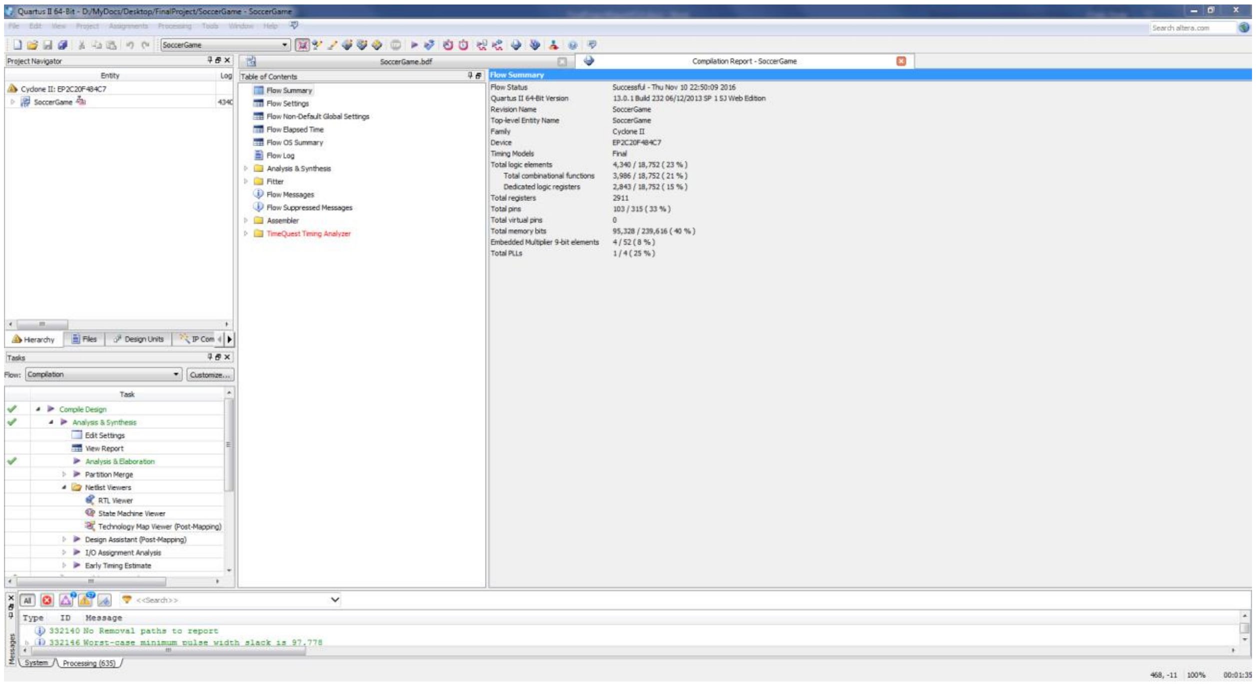
NIOS Design



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
✓		CPU	Nios II Processor					
		clk	Clock Input	Double-click to export	sys_clk			
		reset_n	Reset Input	Double-click to export	[clk]			
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	IRQ 0	IRQ 31	
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]			
		jtag_debug_module_re...	Reset Output	Double-click to export	[clk]			
		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0008_2800	0x0008_2fff	
		custom_instruction_m...	Custom Instruction Master	Double-click to export				
✓		clk_0	Clock Source					
		clk_in	Clock Input	clk_0_clk_in				
		clk_in_reset	Reset Input	clk_0_clk_in_reset				
		clk	Clock Output	Double-click to export	clk_0			
		clk_reset	Reset Output	Double-click to export				
✓		Clock_Signals	Clock Signals for DE-series Board Peri...					
		clk_in_primary	Clock Input	Double-click to export	clk_0			
		clk_in_primary_reset	Reset Input	Double-click to export	[clk_in_prima...			
		sys_clk	Clock Output	Double-click to export	sys_clk			
		sys_clk_reset	Reset Output	Double-click to export	sys_clk			
		sdram_clk	Clock Output	clock_signals_sdram_clk	Clock_Signal...			
		vga_clk	Clock Output	Double-click to export	vga_clk			
✓		Pixel_Buffer	SRAM/SSRAM Controller					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		external_interface	Conduit	Pixel_Buffer_external_int...				
		avalon_sram_slave	Avalon Memory Mapped Slave	Double-click to export	[clock_reset]	0x0000_0000	0x0007_ffff	
✓		Pixel_Buffer_DMA	Pixel Buffer DMA Controller					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_pixel_dma_ma...	Avalon Memory Mapped Master	Double-click to export	[clock_reset]			
		avalon_control_slave	Avalon Memory Mapped Slave	Double-click to export	[clock_reset]	0x0008_3020	0x0008_302f	
		avalon_pixel_source	Avalon Streaming Source	Double-click to export	[clock_reset]			
✓		Pixel_RGB_Resampler	RGB Resampler					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_rgb_sink	Avalon Streaming Sink	Double-click to export	[clock_reset]			
		avalon_rgb_source	Avalon Streaming Source	Double-click to export	[clock_reset]			
✓		Pixel_Scaler	Scaler					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_scaler_sink	Avalon Streaming Sink	Double-click to export	[clock_reset]			
		avalon_scaler_source	Avalon Streaming Source	Double-click to export	[clock_reset]			
✓		Dual_Clock_FIFO	Dual-Clock FIFO					
		clock_stream_in	Clock Input	Double-click to export	sys_clk			
		clock_stream_in_reset	Reset Input	Double-click to export	[clock_strea...			
		clock_stream_out	Clock Input	Double-click to export	vga_clk			
		clock_stream_out_reset	Reset Input	Double-click to export	[clock_strea...			
		avalon_dc_buffer_sink	Avalon Streaming Sink	Double-click to export	[clock_strea...			
		avalon_dc_buffer_so...	Avalon Streaming Source	Double-click to export	[clock_strea...			
✓		VGA_Controller	VGA Controller					
		clock_reset	Clock Input	Double-click to export	vga_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_vga_sink	Avalon Streaming Sink	Double-click to export	[clock_reset]			
		external_interface	Conduit	VGA_Controller_external...				
✓		Key_PIO	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	sys_clk			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0008_3000	0x0008_300f	
		external_connection	Conduit	key_pio_external_conne...				
✓		SW_PIO	PIO (Parallel I/O)					
		clk	Clock Input	Double-click to export	sys_clk			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0008_3010	0x0008_301f	
		external_connection	Conduit	sw_pio_external_connec...				
✓		video_character_buff...	Character Buffer for VGA Display					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_char_control_...	Avalon Memory Mapped Slave	Double-click to export	[clock_reset]	0x0008_3030	0x0008_3037	
		avalon_char_buffer_s...	Avalon Memory Mapped Slave	Double-click to export	[clock_reset]	0x0008_0000	0x0008_1fff	
		avalon_char_source	Avalon Streaming Source	Double-click to export	[clock_reset]			
✓		video_alpha_blender_0	Alpha Blender					
		clock_reset	Clock Input	Double-click to export	sys_clk			
		clock_reset_reset	Reset Input	Double-click to export	[clock_reset]			
		avalon_foreground_sink	Avalon Streaming Sink	Double-click to export	[clock_reset]			
		avalon_background_si...	Avalon Streaming Sink	Double-click to export	[clock_reset]			
		avalon_blended_source	Avalon Streaming Source	Double-click to export	[clock_reset]			
✓		new_sdram_controll...	SDRAM Controller					
		clk	Clock Input	Double-click to export	sys_clk			
		reset	Reset Input	Double-click to export	[clk]			
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0100_0000	0x017f_ffff	
		wire	Conduit	new_sdram_controller_0...				

Figure 21: Nios Design

Quartus Compile Summary Statistic



Flow Summary	
Flow Status	Successful - Thu Nov 10 22:50:09 2016
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	SoccerGame
Top-level Entity Name	SoccerGame
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	4,340 / 18,752 (23 %)
Total combinational functions	3,986 / 18,752 (21 %)
Dedicated logic registers	2,843 / 18,752 (15 %)
Total registers	2911
Total pins	103 / 315 (33 %)
Total virtual pins	0
Total memory bits	95,328 / 239,616 (40 %)
Embedded Multiplier 9-bit elements	4 / 52 (8 %)
Total PLLs	1 / 4 (25 %)

```
Type   ID     Message
       332140 No Removal paths to report
       332146 Worst-case minimum pulse width slack is 97.778
```

Figure 22: Compile Summary Statistic

Software Design (C Code)

```
/*
 * Soccer Game
 * Name: Ziwei.C and Zhaoming.Q
 * EE 3921 011 Final Project
 * Final Rev: 11/9/2016
 */
#include "alt_types.h"
#include "altera_avalon_pio_regs.h"
#include "sys/alt_irq.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h>
#include "altera_up_avalon_video_pixel_buffer_dma.h"
#include "altera_up_avalon_video_character_buffer_with_dma.h"
#include "altera_avalon_pio_regs.h"
/*######################### FUNCTION CALL ################################*/
void initialPic();
void AnimPlay(int Result);
void numDisplay(int num1, int num2);
void Background();
void GoalKeeperselect(int GoalKeeper);
void Penaltytakerselect(int Penaltytaker);
void namedisplay(int name1, int name2);

int main(void) {
/*###################### INITIAL SETUP ################################*/
        /*
         * Define the state for gaming GameState.
         *   0 for START up the game and display the Title/Name
         *   1 for first player chose direction.
         *   2 for second player choose direction.
         *   3 for game result calculation.
         *   4 for Anim playing.
         *   5 for EXIT the game.
         *   6 for Idle, waiting for start the game.
         */
        int GameState = 8; /* Default to enter Idle state8 */
        int Round = 1; /* To count the total number of round played */
        int Score_1 = 0, Score_2 = 0; /* To count the score for both players.*/
        int Direction_1 = 0, Direction_2 = 0; /* To record the direction chosen for each 
player.*/
        int team_1 = 0, team_2 = 0; /* To record the team for each player*/
        /*
         *  Define the variable to store the result from game.
         *        Penalty taker - Goal keeper
         *  1 for left          - left
         *  2 for left          - center
         *  3 for left          - right
         *  4 for center        - left
         *  5 for center        - center
         *  6 for center        - right
         *  7 for right         - left
         *  8 for right         - center
         *  9 for right         - right
         */
        int GameResult = 0;
        alt_up_pixel_buffer_dma_dev* EnableDev;
        EnableDev = alt_up_pixel_buffer_dma_open_dev("/dev/Pixel_Buffer_DMA"); /* Enable the VGA 
Device */
        alt_up_pixel_buffer_dma_clear_screen(EnableDev, 0); /* Clean the Screen */

        alt_up_char_buffer_dev* EnableChar;
        EnableChar = alt_up_char_buffer_open_dev(
                        "/dev/video_character_buffer_with_dma_0");
        alt_up_char_buffer_clear(EnableChar); /* Clean the Screen */
        while (1) { /* Enter the main loop.*/
/*########################### STATE 110 ################################*/
               while (GameState == 110) {
                       int SWReading = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE); /* read the
value of the SW */
                       alt_up_char_buffer_clear(EnableChar);
                       while (SWReading != 0) {
                              SWReading = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE); /* read the
value of the SW */
                              usleep(200000);
                              alt_up_char_buffer_string(EnableChar,
                                             "PULL DOWN SW6 to SW0 TO START THE GAME", 23, 30);
                              usleep(200000);
                              alt_up_char_buffer_string(EnableChar,
                                             "PULL DOWN     to     TO START THE GAME", 23, 30);
                       }
                       GameState = 1; /* Enter State 1 to start the game8*/
               }
/*############################ STATE 0 #################################*/
               while (GameState == 0) { /* State for START game and display the Title/Name.*/
                       alt_up_char_buffer_clear(EnableChar);
                       alt_up_char_buffer_string(EnableChar, "SOCCER GAME", 35, 25); /* Display
the title 'Soccer Game', 'EE 3921 Final Project', 'Ziwei.C and Zhaoming.Q' */
                       alt_up_char_buffer_string(EnableChar, "EE 3921 FINAL PROJECT", 30,
                                      30);
                       alt_up_char_buffer_string(EnableChar, "ZIWEI.C & ZHAOMING.Q", 30,
                                      35);
                       usleep(5000000);
                       initialPic();          /*Display the initial picture for game.*/
                       /*
                        * Reset all variables to its initial values.
                        */
                       Round = 1;             /*To count the total number of round played.*/
                       Score_1 = 0, Score_2 = 0; /*To count the score for both players.*/
                       team_1 = 0, team_2 = 0;
                       GameState = 110;       /* Enter State 1, for the first player to choose
direction. */
               }
/*############################ STATE 1 #################################*/
               while (GameState == 1) { /* State for the first player to choose the team */
                       alt_up_char_buffer_clear(EnableChar);
                       alt_up_char_buffer_string(EnableChar,
                                      "First Player: Pull Up SW to Choose Goal Keeper", 5, 5);
/* Disp the char choose */
                       alt_up_char_buffer_string(EnableChar, "Samir Handanovic .. 0000001",
                                      5, 10);
                       alt_up_char_buffer_string(EnableChar, "Manuel Neuver ..... 0000010",
                                      5, 15);
                       alt_up_char_buffer_string(EnableChar, "Jan Oblak ......... 0000100",
                                      5, 20);
                       alt_up_char_buffer_string(EnableChar, "Petr Cech ......... 0001000",
                                      5, 25);
                       alt_up_char_buffer_string(EnableChar, "Thibaut Courtois .. 0010000",
                                      5, 30);
                       alt_up_char_buffer_string(EnableChar, "Joe Hart .......... 0100000",
                                      5, 35);
                       alt_up_char_buffer_string(EnableChar, "David de Gea ...... 1000000",
                                      5, 40);
                       int temp_team1; /*create a temp variable*/
                       team_1 = 0; /* set the initial value*/
                       temp_team1 = team_1;
                       while ((temp_team1 == team_1)
                                      || ((team_1 != 1) && (team_1 != 2) && (team_1 != 4)
                                                     && (team_1 != 8) && (team_1 != 16) &&
(team_1 != 32)
                                                     && (team_1 != 64))) {
                              team_1 = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE); /*read the value
of the SW*/
                       }
                       temp_team1 = team_1;                            /*store the value
of team_1 into temp variable*/
                       GoalKeeperselect(team_1);
                       while (temp_team1 != 0) {
                              temp_team1 = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE);
```

```
				alt_up_char_buffer_string(EnableChar,
						"PULL DOWN SWITCH TO SELECT NEXT CHARACTER", 5, 52);
				usleep(200000);
				alt_up_char_buffer_string(EnableChar,
						"PULL      SWITCH TO SELECT NEXT CHARACTER", 5, 52);
				usleep(200000);
			}
			GameState = 2;
		}
/*########################### STATE  #################################*/
		while (GameState == 2) { /*State for the second player to choose the team*/
			alt_up_char_buffer_clear(EnableChar);
			alt_up_char_buffer_string(EnableChar,
					"Second Player: Pull up SW to Choose Penalty Taker", 5, 5);
			alt_up_char_buffer_string(EnableChar, "Mauro Icardi ...... 0000001",
					5, 10);
			alt_up_char_buffer_string(EnableChar, "Thomas Muller ..... 0000010",
					5, 15);
			alt_up_char_buffer_string(EnableChar, "Antoine Griezmann . 0000100",
					5, 20);
			alt_up_char_buffer_string(EnableChar, "Santi Cazorla ..... 0001000",
					5, 25);
			alt_up_char_buffer_string(EnableChar, "Eden Hazard ....... 0010000",
					5, 30);
			alt_up_char_buffer_string(EnableChar, "Kevin De Bruyne ... 0100000",
					5, 35);
			alt_up_char_buffer_string(EnableChar, "Wayne Rooney ...... 1000000",
					5, 40);
			int temp_team2; //create a temp variable
			team_2 = 0; //set the initial value
			temp_team2 = team_2;
			while (temp_team2 == team_2) {
				team_2 = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE); //read the value
of SW
			}
			temp_team2 = team_2;
			Penaltytakerselect(team_2);
			while (temp_team2 != 0) {
				temp_team2 = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE);
				alt_up_char_buffer_string(EnableChar,
						"PULL DOWN SWITCH TO START GAME", 5, 52);
				usleep(200000);
				alt_up_char_buffer_string(EnableChar,
						"PULL      SWITCH TO START GAME", 5, 52);
				usleep(200000);
			}
			GameState = 3;
		}
/*########################### STATE 3 #################################*/
		while (GameState == 3) { /* State for the first player to choose direction.*/
			initialPic();
			numDisplay(Score_1, Score_2);
			namedisplay(team_1, team_2);
			alt_up_char_buffer_string(EnableChar,
					"Penalty Taker Choose Direction", 25, 30);
			alt_up_char_buffer_string(EnableChar,
					"Press KEY 0,1,2 To Choose Right, Center, or Left", 18,
35); /* Display: 'Penalty Taker choose direction', 'Press Key 0,1,2' to choose left, center, or
right*/
			int Dir_temp_1 = 16;
			Direction_1 = 15;
			Dir_temp_1 = Direction_1;
			while (Dir_temp_1 != Direction_1) {
				Dir_temp_1 = Direction_1;
			}
			while ((Dir_temp_1 == Direction_1)
					|| ((Direction_1 != 11) && (Direction_1 != 13)
							&& (Direction_1 != 14))) {
				Direction_1 = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE); /*Read the
Input from key, and store the value in Direction_1.*/
			}
```

```
                        Dir_temp_1 = Direction_1;
                        while (Dir_temp_1 != 15) {
                                Dir_temp_1 = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE); /*Read the
Input from key, and store the value in Direction_1.*/
                        }
                        GameState = 4; /*Enter State 2, for the second player to choose
Direction.*/
                }
/*############################ STATE 4 #################################*/
                while (GameState == 4) { /*State for the second player to choose direction.*/
                        alt_up_char_buffer_string(EnableChar,
                                        "Goal Keeper Choose Direction", 25, 30);
                        alt_up_char_buffer_string(EnableChar,
                                        "Press KEY 0,1,2 To Choose Right, Center, or Left", 18,
35); /* Display: 'Goal Keeper choose direction', 'Press Key 0,1,2' to choose left, center, or
right'.*/
                        int Dir_temp_2 = 16;
                        Direction_2 = 15;
                        Dir_temp_2 = Direction_2;
                        while (Dir_temp_2 != Direction_2) {
                                Dir_temp_2 = Direction_2;
                        }
                        while ((Dir_temp_2 == Direction_2)
                                        || ((Direction_2 != 11) && (Direction_2 != 13)
                                                        && (Direction_2 != 14))) {
                                Direction_2 = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE); /*Read the
Input from key, and store the value in Direction_2.*/
                        }
                        Dir_temp_2 = Direction_2;
                        while (Dir_temp_2 != 15) {
                                Dir_temp_2 = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE); /*Read the
Input from key, and store the value in Direction_2.*/
                        }
                        GameState = 5; /*Enter State 3, for result computation*/
                }
/*############################ STATE 5 #################################*/
                while (GameState == 5) { /*State for compute the result for game.*
                        /*
                         * Compare the value for direction to decide the game result, and Store
the result in GameResult.
                         */
                        if (Direction_1 == 11 && Direction_2 == 11) { //--------------------------
-left - left
                                GameResult = 1; //-----------------------------------------------
----Save the goal
                        } else if (Direction_1 == 11 && Direction_2 == 13) { //-------------------
-left - center
                                GameResult = 2;
                        } else if (Direction_1 == 11 && Direction_2 == 14) { //-------------------
-left - right
                                GameResult = 3;
                        } else if (Direction_1 == 13 && Direction_2 == 11) { //-------------------
-center - left
                                GameResult = 4;
                        } else if (Direction_1 == 13 && Direction_2 == 13) { //-------------------
-center - center
                                GameResult = 5; //-----------------------------------------------
----Save the goal
                        } else if (Direction_1 == 13 && Direction_2 == 14) { //-------------------
-center - right
                                GameResult = 6;
                        } else if (Direction_1 == 14 && Direction_2 == 11) { //-------------------
-right - left
                                GameResult = 7;
                        } else if (Direction_1 == 14 && Direction_2 == 13) { //-------------------
-right  - center
                                GameResult = 8;
                        } else if (Direction_1 == 14 && Direction_2 == 14) { //-------------------
-right - right
                                GameResult = 9; //-----------------------------------------------
----Save the goal
```

```
                        } else {
                                GameState = 3;
                        }
                        /*
                         * Play the Anim Base on the value for GameResult.
                         * when two values are equal, the goal keeper save the goal,
                         * else the penalty take the score.
                         */
                        GameState = 6; //------------------------------------Go to the state for
playing Anim.
                }
/*########################### STATE 6 #################################*/
                while (GameState == 6) { //--------------------------------State for Anim playing.
                        alt_up_char_buffer_clear(EnableChar);
                        numDisplay(Score_1, Score_2);
                        namedisplay(team_1, team_2);
                        AnimPlay(GameResult);
                        usleep(1000000);
                        if (GameResult == 1 || GameResult == 5 || GameResult == 9) { //Store the
score in either Score1 and Score2
                                Score_1++; //------------------------------------The keeper take
score
                        } else {
                                Score_2++; //------------------------------------The taker take
score
                        }
                        // Update the Score on screen
                        /* If Round value larger or equal than 5, enter the EXIT Game state.
                         * If Round value smaller then 5, then Enter the State for the first
player to choose direction.
                         */
                        numDisplay(Score_1, Score_2);
                        if (Round < 5) {
                                GameState = 3;
                                Round = Round + 1; //-----------------------------Update the Round
value
                        } else if (Round >= 5) {
                                GameState = 7;
                                Round = 1;
                                numDisplay(Score_1, Score_2);
                                namedisplay(team_1, team_2);
                                usleep(2000000);
                        }
                        if ((Score_1 == 3) || (Score_2 == 3)) {
                                GameState = 7;
                                numDisplay(Score_1, Score_2);
                                namedisplay(team_1, team_2);
                                usleep(2000000);
                        }
                }
/*########################### STATE 7 #################################*/
                while (GameState == 7) { //--------------------------------State for Exit the game
and reset all data.
                        alt_up_pixel_buffer_dma_clear_screen(EnableDev, 0);
                        alt_up_char_buffer_clear(EnableChar);
                        int WhoWin; //-----------------------------------------0 for keeper win, 1
for taker win.
                        if (Score_1 > Score_2) { //-----------------------------Compare the value
for Score_1 and Score_2, the larger one wins. Store value in WhoWin.
                                WhoWin = 1;
                        } else if (Score_1 < Score_2) {
                                WhoWin = 0;
                        }
                        if (WhoWin == 1) {
                                alt_up_char_buffer_string(EnableChar, "Goal Keeper Win!!", 32,
                                                25);  /* Display the message 'XXX WIN!!!', 'Game
END!!!' base on the value of WhoWin.*/
                        }
                        if (WhoWin == 0) {
                                alt_up_char_buffer_string(EnableChar, "Penalty taker Win!!", 32,
                                                25);
                        }
                        Background();
                        alt_up_pixel_buffer_dma_clear_screen(EnableDev, 0); //--Clean the Screen
                        GameState = 8;
                }
                /*########################### STATE 8 #################################*/
                while (GameState == 8) { //--------------------------------State for waiting.
Wither keep waiting or start the exit game.
                        alt_up_char_buffer_string(EnableChar,
                                        "PRESS KEY3 TO START A NEW GAME", 26, 30); // Display
'Press KeyX to start new game'
                        int Restart = 0;
                        int Restart_tmp = Restart; //---------------------------Read the value for
KeyX
                        while (Restart_tmp == Restart) {
                                Restart = IORD_ALTERA_AVALON_PIO_DATA(KEY_PIO_BASE);
                                alt_up_char_buffer_string(EnableChar,
                                        "PRESS      TO START A NEW GAME", 26, 30);
                                usleep(200000);
                                alt_up_char_buffer_string(EnableChar,
                                        "PRESS KEY3 TO START A NEW GAME", 26, 30);
                                usleep(200000);

                        }
                        if (Restart == 7) {
                                GameState = 0; //----------------------------------Enter State 0
for start new game, or stay in Idle.
                        }
                }
        }
}
/*#######################################################################*/
/*########################### Functions #################################*/
/*#######################################################################*/

void namedisplay(int name1, int name2) { /* team name display while playing the game */
        alt_up_char_buffer_dev* EnableChar;
        EnableChar = alt_up_char_buffer_open_dev(
                        "/dev/video_character_buffer_with_dma_0");
        if (name1 == 1) {
                alt_up_char_buffer_string(EnableChar, "Samir Handanovic", 14, 18);
        }
        if (name1 == 2) {
                alt_up_char_buffer_string(EnableChar, "Manuel Neuver", 14, 18);
        }
        if (name1 == 4) {
                alt_up_char_buffer_string(EnableChar, "Jan Oblak", 14, 18);
        }
        if (name1 == 8) {
                alt_up_char_buffer_string(EnableChar, "Petr Cech", 14, 18);
        }
        if (name1 == 16) {
                alt_up_char_buffer_string(EnableChar, "Thibaut Courtois", 14, 18);
        }
        if (name1 == 32) {
                alt_up_char_buffer_string(EnableChar, "Joe Hart", 14, 18);
        }
        if (name1 == 64) {
                alt_up_char_buffer_string(EnableChar, "David de Gea", 14, 18);
        }
        if (name2 == 1) {
                alt_up_char_buffer_string(EnableChar, "Mauro Icardi", 14, 43);
        }
        if (name2 == 2) {
                alt_up_char_buffer_string(EnableChar, "Thomas Muller", 14, 43);
        }
        if (name2 == 4) {
                alt_up_char_buffer_string(EnableChar, "Antoine Griezmann", 14, 43);
        }
        if (name2 == 8) {
                alt_up_char_buffer_string(EnableChar, "Santi Cazorla", 14, 43);
        }
        if (name2 == 16) {
                alt_up_char_buffer_string(EnableChar, "Eden Hazard", 14, 43);
        }
        if (name2 == 32) {
                alt_up_char_buffer_string(EnableChar, "Kevin De Bruyne", 14, 43);
        }
        if (name2 == 64) {
                alt_up_char_buffer_string(EnableChar, "Wayne Rooney", 14, 43);
        }
}
void GoalKeeperselect(int GoalKeeper) { /* Select the goal keeper */
        alt_up_char_buffer_dev* EnableChar;
        EnableChar = alt_up_char_buffer_open_dev(
                        "/dev/video_character_buffer_with_dma_0");
        if (GoalKeeper == 1) {
                alt_up_char_buffer_string(EnableChar, "You choose Samir Handanovic", 5,
                                50);
        } else if (GoalKeeper == 2) {
                alt_up_char_buffer_string(EnableChar, "You choose Manuel Neuver", 5,
                                50);
        } else if (GoalKeeper == 4) {
                alt_up_char_buffer_string(EnableChar, "You choose Jan Oblak", 5, 50);
        } else if (GoalKeeper == 8) {
                alt_up_char_buffer_string(EnableChar, "You choose Petr Cech", 5, 50);
        } else if (GoalKeeper == 16) {
                alt_up_char_buffer_string(EnableChar, "You choose Thibaut Courtois", 5,
                                50);
        } else if (GoalKeeper == 32) {
                alt_up_char_buffer_string(EnableChar, "You choose Joe Hart", 5, 50);
        } else if (GoalKeeper == 64) {
                alt_up_char_buffer_string(EnableChar, "You choose David de Gea", 5, 50);
        } else {
                alt_up_char_buffer_string(EnableChar, "No team found, reselect team", 5,
                                55);
        }

}

void Penaltytakerselect(int Penaltytaker) { /*Function to display the taker chosen*/
        alt_up_char_buffer_dev* EnableChar;
        EnableChar = alt_up_char_buffer_open_dev(
                        "/dev/video_character_buffer_with_dma_0");
        if (Penaltytaker == 1) {
                alt_up_char_buffer_string(EnableChar, "You choose Mauro Icardi", 5, 50);
        } else if (Penaltytaker == 2) {
                alt_up_char_buffer_string(EnableChar, "You choose Thomas Muller", 5,
                                50);
        } else if (Penaltytaker == 4) {
                alt_up_char_buffer_string(EnableChar, "You choose Antoine Griezmann", 5,
                                50);
        } else if (Penaltytaker == 8) {
                alt_up_char_buffer_string(EnableChar, "You choose Santi Cazorla", 5,
                                50);
        } else if (Penaltytaker == 16) {
                alt_up_char_buffer_string(EnableChar, "You choose Eden Hazard", 5, 50);
        } else if (Penaltytaker == 32) {
                alt_up_char_buffer_string(EnableChar, "You choose Kevin De Bruyne", 5,
                                50);
        } else if (Penaltytaker == 64) {
                alt_up_char_buffer_string(EnableChar, "You choose Wayne Rooney", 5, 50);
        } else {
                alt_up_char_buffer_string(EnableChar, "No team found, reselect team", 5,
                                55);
        }
}

void Background(void) { /*Function to draw the game end animation*/
        alt_up_pixel_buffer_dma_dev* EnableDev;
        EnableDev = alt_up_pixel_buffer_dma_open_dev("/dev/Pixel_Buffer_DMA"); //Enable the VGA
Device
```

```
        int x0 = 0;
        int x1 = 40;
        int y0 = 0;
        int y1 = 40;
        while (x0 < 600) {
                alt_up_pixel_buffer_dma_draw_box(EnableDev, x0, y0, x1, y1, 0X1111, 0); //Draw
first line
                x0++;
                x1++;
                usleep(10000);
                if ((x0 > 40) && (x0 < 600)) {
                        alt_up_pixel_buffer_dma_draw_box(EnableDev, x0 - 40, y0 + 40,
                                        x1 - 40, y1 + 40, 0X3333, 0); //Draw 2nd line
                        x0++;
                }
                if ((x0 > 80) && (x0 < 600)) {
                        alt_up_pixel_buffer_dma_draw_box(EnableDev, x0 - 80, y0 + 80,
                                        x1 - 80, y1 + 80, 0X6666, 0); //Draw 3rd line
                        x0++;
                }
                if ((x0 > 120) && (x0 < 600)) {
                        alt_up_pixel_buffer_dma_draw_box(EnableDev, x0 - 120, y0 + 120,
                                        x1 - 120, y1 + 120, 0X8888, 0); //Draw 4th line
                        x0++;
                }
                if ((x0 > 160) && (x0 < 600)) {
                        alt_up_pixel_buffer_dma_draw_box(EnableDev, x0 - 160, y0 + 160,
                                        x1 - 160, y1 + 160, 0XBBBB, 0); //Draw 5th line
                        x0++;
                }
                if ((x0 > 200) && (x0 < 600)) {
                        alt_up_pixel_buffer_dma_draw_box(EnableDev, x0 - 200, y0 + 200,
                                        x1 - 200, y1 + 200, 0XDDDD, 0); //Draw 6th line
                        x0++;
                }
        }
}

void numDisplay(int num1, int num2) { /*Function to display the numbers on screen*/
        int x1 = 20;
        int y1 = 20;
        int x2 = 20;
        int y2 = 45;
        alt_up_char_buffer_dev* EnableChar;
        EnableChar = alt_up_char_buffer_open_dev(
                        "/dev/video_character_buffer_with_dma_0");
        alt_up_char_buffer_clear(EnableChar);
        //alt_up_char_buffer_string(EnableChar, "Goal keeper", 18, 18);
        //alt_up_char_buffer_string(EnableChar, "Penalty taker", 18, 43);
        if (num1 == 0) {
                alt_up_char_buffer_string(EnableChar, "0", x1, y1);
        }
        if (num1 == 1) {
                alt_up_char_buffer_string(EnableChar, "1", x1, y1);
        }
        if (num1 == 2) {
                alt_up_char_buffer_string(EnableChar, "2", x1, y1);
        }
        if (num1 == 3) {
                alt_up_char_buffer_string(EnableChar, "3", x1, y1);
        }
        if (num1 == 4) {
                alt_up_char_buffer_string(EnableChar, "4", x1, y1);
        }
        if (num1 == 5) {
                alt_up_char_buffer_string(EnableChar, "5", x1, y1);
        }
        if (num2 == 0) {
                alt_up_char_buffer_string(EnableChar, "0", x2, y2);
        }
        if (num2 == 1) {
                alt_up_char_buffer_string(EnableChar, "1", x2, y2);
        }
        if (num2 == 2) {
                alt_up_char_buffer_string(EnableChar, "2", x2, y2);
        }
        if (num2 == 3) {
                alt_up_char_buffer_string(EnableChar, "3", x2, y2);
        }
        if (num2 == 4) {
                alt_up_char_buffer_string(EnableChar, "4", x2, y2);
        }
        if (num2 == 5) {
                alt_up_char_buffer_string(EnableChar, "5", x2, y2);
        }
}

void initialPic(void) { /*Fucntion to display the initial picture*/
        alt_up_pixel_buffer_dma_dev* EnableDev;
        EnableDev = alt_up_pixel_buffer_dma_open_dev("/dev/Pixel_Buffer_DMA"); //Enable the VGA
Device
        alt_up_pixel_buffer_dma_draw_hline(EnableDev, 120, 200, 50, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_vline(EnableDev, 120, 50, 70, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_vline(EnableDev, 200, 50, 70, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165, 67, 0XFFFF,
                        0); // keeper
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 175, 165, 185,
                        0XFFFF, 0); // taker
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 159, 170, 161, 172,
                        0XFFFF, 0); // Ball
}

void AnimPlay(int Result) { /*Play the soccer animation*/
        alt_up_pixel_buffer_dma_dev* EnableDev;
        EnableDev = alt_up_pixel_buffer_dma_open_dev("/dev/Pixel_Buffer_DMA"); //Enable the VGA
Device
        alt_up_pixel_buffer_dma_clear_screen(EnableDev, 0); //Clean the Screen
        alt_up_pixel_buffer_dma_draw_hline(EnableDev, 120, 200, 50, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_vline(EnableDev, 120, 50, 70, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_vline(EnableDev, 200, 50, 70, 0XFFFF, 0);
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165, 67, 0XFFFF,
                        0); //  draw keeper
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 175, 165, 185,
                        0XFFFF, 0); // draw taker
        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 159, 170, 161, 172,
                        0XFFFF, 0); //  draw Ball
        int y1 = 170;
        int y2 = 172;
        int x1 = 159;
        int x2 = 161;
        if (Result == 1) { //left - left
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 1;
                                x2 = x2 - 1;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
                        if (y1 <= 80) {
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
                                                67, 0X0000, 0); // keeper
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 125, 57, 135,
                                                67, 0XFFFF, 0); // keeper
                        }
                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 2) { //left - center
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 1;
                                x2 = x2 - 1;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;

                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 3) { //left - right
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 1;
                                x2 = x2 - 1;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
                        if (y1 <= 80) {
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
                                                67, 0X0000, 0); // keeper
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 195, 57, 185,
                                                67, 0XFFFF, 0); // keeper
                        }
                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 4) { //center - left
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 0;
                                x2 = x2 - 0;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
                        if (y1 <= 80) {
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
                                                67, 0X0000, 0); // keeper
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 125, 57, 135,
                                                67, 0XFFFF, 0); // keeper
                        }
                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 5) { //center - center
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 0;
                                x2 = x2 - 0;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;

                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 6) { //center - right
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 - 0;
                                x2 = x2 - 0;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
                        if (y1 <= 80) {
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
                                                67, 0X0000, 0); // keeper
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 195, 57, 185,
                                                67, 0XFFFF, 0); // keeper
                        }
                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 7) { //right -left
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 + 1;
                                x2 = x2 + 1;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
                        if (y1 <= 80) {
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
                                                67, 0X0000, 0); // keeper
                                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 125, 57, 135,
                                                67, 0XFFFF, 0); // keeper
                        }
                }
                alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                0XFFFF, 0); // Ball
        }
        if (Result == 8) { //right -center
                while (y1 >= 65 && y2 >= 67) {
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0XFFFF, 0); // Ball
                        usleep(25000);
                        alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
                                        0X0000, 0); // Ball
                        if (x1 >= 132 && x2 >= 135) {
                                x1 = x1 + 1;
                                x2 = x2 + 1;
                        }
                        y1 = y1 - 4;
                        y2 = y2 - 4;
```

```
		}
		alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
				0XFFFF, 0); // Ball
	}
	if (Result == 9) { //right -right
		while (y1 >= 65 && y2 >= 67) {
			alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
					0XFFFF, 0); // Ball
			usleep(25000);
			alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
					0X0000, 0); // Ball
			if (x1 >= 132 && x2 >= 135) {
				x1 = x1 + 1;
				x2 = x2 + 1;
			}
			y1 = y1 - 4;
			y2 = y2 - 4;
			if (y1 <= 80) {
				alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165,
						67, 0X0000, 0); // keeper
				alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 195, 57, 185,
						67, 0XFFFF, 0); // keeper
			}
		}
		alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, x1, y1, x2, y2,
				0XFFFF, 0); // Ball
	}
	usleep(2000000);
	alt_up_pixel_buffer_dma_clear_screen(EnableDev, 0); //Clean the Screen
	alt_up_pixel_buffer_dma_draw_hline(EnableDev, 120, 200, 50, 0XFFFF, 0);
	alt_up_pixel_buffer_dma_draw_vline(EnableDev, 120, 50, 70, 0XFFFF, 0);
	alt_up_pixel_buffer_dma_draw_vline(EnableDev, 200, 50, 70, 0XFFFF, 0);
	alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 57, 165, 67, 0XFFFF,
			0); // keeper
	alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 155, 175, 165, 185,
			0XFFFF, 0); // taker
	alt_up_pixel_buffer_dma_draw_rectangle(EnableDev, 159, 170, 161, 172,
			0XFFFF, 0); // Ball
}
```