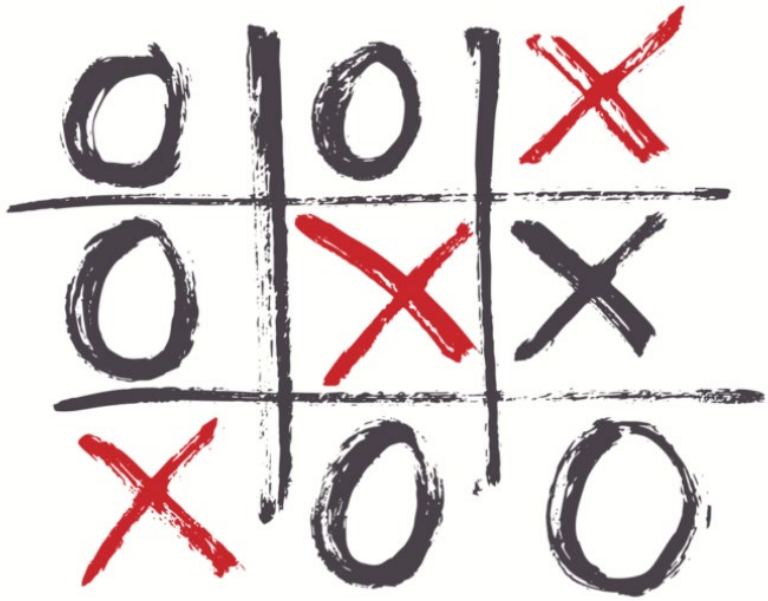
EE 277A – EMBEDDED SoC DESIGN

DEPARTMENT OF ELECTRICAL ENGINEERING

IMPLEMENTATION OF TIC-TAC-TOE GAME ON NEXYS A7 FPGA BOARD USING C LANGUAGE



Under the guidance of:

Dr. Shrikant Jadhav

Authors:

Kunal Sawant

Jayesh Pradhan

Shibendu Bhowmick

TABLE OF CONTENT

LIST OF FIGURES

1. INTRODUCTION

Tic Tac Toe, also known as Noughts and Crosses, is a classic two-player game that has been enjoyed by people of all ages for decades. The game is played on a 3x3 grid, where two players take turns marking their symbol on the board. The objective of the game is to be the first player to get three of their symbols in a row, either horizontally, vertically, or diagonally. The game's simplicity and ease of play have made it a popular choice for many people, and it can be played with just a pen and paper. However, with the advent of computers, the game has also been implemented in various programming languages, including C. In the game of Tic Tac Toe, each player is assigned a symbol, usually X or O. The game begins with an empty 3x3 grid, and players take turns marking their symbol on the board. The first player to get three of their symbols in a row wins the game.

We, in this project have implemented the Tic Tac Toe game using C programming language and assembly language which communicates with each other and with the help of Nexys A7 FPGA board we were able to display the working of the game on the monitor screen using VGA cable. This project is implemented on ARM cortex-M0 CPU. The crucial communication link between high-level software and low-level hardware will then be constructed with the aid of APIs. Additionally, we'll provide an API that makes use of CMSIS and software driver characteristics to allow more flexible and user-friendly application development services. To show off the Snake game on the SoC, we'll create the final application. The report provides a detailed explanation of the project implementation, required hardware and software, methodology used, challenges faced and learning outcomes.

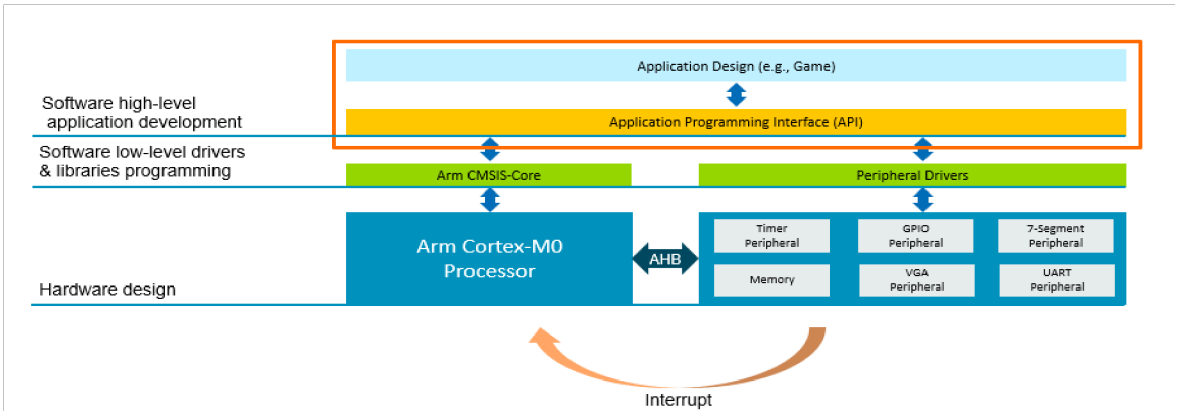


Figure 1. SoC Architecture

For all three of the levels shown in figure 1 above, drivers will be developed in this lab. First, we'll create a sophisticated snake game computer application. Then, in the application game, we will develop an API for interaction between the ARM CMSIS-core and peripheral drivers. In order to connect with our ARM Cortex M0 CPU-based hardware design and main application, we will also develop a low-level software driver. For the hardware development of the ARM Cortex M0 CPU, we'll make use of an IP core given by ARM. To access other SoC peripherals like VGA, memory access blocks, UART blocks, etc., we will also build hardware modules in Verilog. Finally, this lab will provide us with a full grasp of the whole application design process, from higher-level software development in C and assembly language programming to lower-level processor hardware development in Verilog.

2. GAME RULES

The rules of Tic Tac Toe are straightforward and easy to understand. Here are the rules of the game:

- The game is played on a 3x3 grid.
- Two players are required to play the game, and each player is assigned a symbol, usually X or O.
- The game begins with an empty grid.
- The first player places their symbol on the grid.
- The second player then places their symbol on the grid.
- Players continue to take turns placing their symbols on the grid until one of the following occurs: a. A player gets three of their symbols in a row, either horizontally, vertically, or diagonally. b. The grid is filled, and no player has three symbols in a row.
- If a player gets three of their symbols in a row, they win the game.
- If the grid is filled, and no player has three symbols in a row, the game ends in a draw.
- After the game ends, players can choose to play again.

In summary, the objective of Tic Tac Toe is to get three of your symbols in a row before your opponent does. The game is won by the player who achieves this objective, and if neither player succeeds, the game is a draw. The game is simple, but it requires players to think ahead and plan their moves carefully.

3. HARDWARE & SOFTWARE USED

1. **“Keil**: We have used Keil uVision 5 for C and assembly language i.e. for snake game application development”.
2. **“Tera term**: Used tera term software to send (UP, DOWN, RIGHT, LEFT) commands via UART to the SoC”.
3. **“Vivado**: We used Xilinx Vivado software for the hardware designing of the processor and other peripherals”.
4. **“Nexys A7:** we have used SoC board Nexys A7 manufactured by the Xilinx.”

3.1 “Verilog files used for hardware designing”.

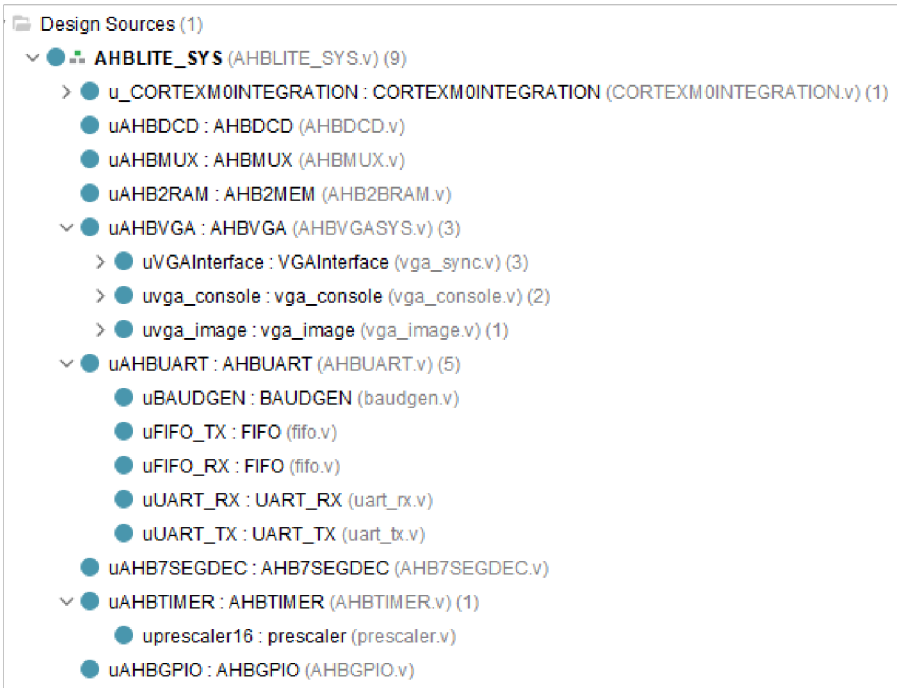


Figure 2. Verilog Files

3.2 “Software Files used in the application.

Core Folder

- “core_cm0.h: CMSIS Cortex-M0 Core Peripheral Access Layer Header File”
- “core_cmFunc.h: CMSIS Cortex-M Core Function Access Header File core_cmInstr.h:”
- “CMSIS Cortex-M Core Instruction Access Header File”

Device Folder

- “cm0dsasm.s “
- “EDK_CM0.h: used to specify Interrupt Number Definition”
- “edk_driver.c: functions definition for VGA, timer, 7 segments, GPIO peripherals”
- “edk_driver.h: Peripheral driver header file”
- “edk_api.c: Application Programming Interface (API) functions edk_api.h: initialization of parameters and functions used for VGA, UART, rectangle etc. “
- “retarget.c: Retarget functions for ARM DS-5 Professional / Keil MDK, allows us to use print library functions “

Application Folder

- “main.c: tasks performed game initialization settings, boundary hit condition, target generation, UART, Timer “

4. BACKGROUND

4.1 Application Programming Interface (API)

"An API is a layer of software abstraction that enables usage of a common programming interface by application developers. For instance, the majority of operating systems have their own APIs to facilitate the development of apps by programmers. An API can be used to offer base services, visual interfaces, network services, and other interface services. commercial APIs like the Windows API and Java API."

"For creating tic tac toe game apps, we have built a simple API that will be applied in this project. The API may offer generic, user-friendly functionality for the end-user by combining CMSIS and peripheral driver operations. For instance, we used a SoC starting technique to restart the CPU as well as the peripherals."

4.2 CMSIS

"An independent of vendor hardware abstraction layer for the Cortex-M processor family is called the Cortex Microcontroller Software Interface Standard (CMSIS). In addition to library functions that make it simpler to operate the CPU, such as setting the nested vectored interrupt controller (NVIC), CMSIS offers a standardized software interface. The primary objective is to increase software portability across various Cortex-M serial processors and microcontrollers."

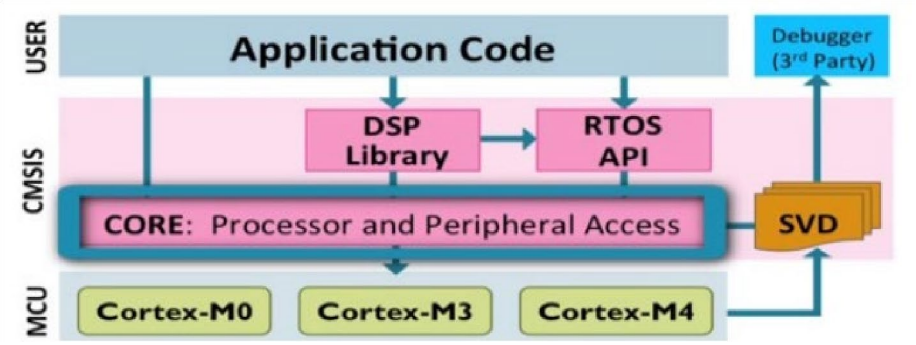


Figure 3. CMSIS Architecture

"CMSIS-CORE is a peripheral register and processor interface for the Cortex-M0, Cortex-M3, Cortex-M4, SC000, and SC300 processors. Over 60 functions in fixed point (fractional q7, q15, q31) and single precision floating-point (32-bit) implementation are included in CMSIS-DSP. CMSIS-RTOS API is a standardized programming interface for thread

control, resource management, and time management in real-time operating systems. CMSIS-SVD files (System View Description) contain the programmer's view of a whole microcontroller system, including peripherals. Visual design is provided by SVD. “

4.2.1 Programming of Cortex M0 using CMSIS

“There are several standardized functions included in the CMSIS; these mainly include access functions for peripherals, registers, and special instructions. The CMSIS includes functions for reading from or writing to core registers. We can also use CMSIS to execute special instructions. The table here lists some of the Cortex-M0 special instructions and their corresponding CMSIS intrinsic functions. CMSIS can also be used for system control and SysTick setup. “

“There are many standardized functions to access NVIC, system control block (SCB), and system tick timer (SysTick). For example: “

- “To enable an interrupt or exception use 'NVIC_EnableIRQ (IRQn_Type IRQn)'.”
- “To set pending status of interrupt us void 'NVIC_SetPendingIRQ (IRQn_Type IRQn)'.”

“For standardized access of special registers, the following functions can be used:”

- “Read PRIMASK register: uint32_t __get_PRIMASK (void) “
- “Set CONTROL register: void __set_CONTROL (uint32_t value)’

“For standardized functions to access special instructions following functions can be used:”

- “REV: uint32_t __REV(uint32_t int value) “
- “NOP: void __NOP(void)”

“For standardized name of system initialization functions following function can be used:”

- “System initialization: void SystemInit(void)”

4.3 Interrupt Handling

“The Cortex-M series processors include an interrupt controller called the Nested Vector Interrupt Controller for interrupt handling such as interrupt prioritization and interrupt masking. The NVIC contains programmable registers for interrupt management such as enable/disable, and priority levels. These registers are memory mapped. The priority levels are defined by 8-bit width registers, but only the MSB bits are implemented. In the Cortex-M0 and Cortex-M0+ processors, there are four programmable priority levels. The NVIC handles nested interrupts automatically. The CortexM0 Microprocessor Architecture is depicted in Figure 6. When an Interrupt Service Routine is operating, the NVIC handles interrupt prioritization and masks out same or lower priority interruptions after the priority levels of each interrupt have been set. If a higher priority interrupt occurs, the running ISR will be pre-empted so that the higher priority ISR can be executed as quickly as possible.”

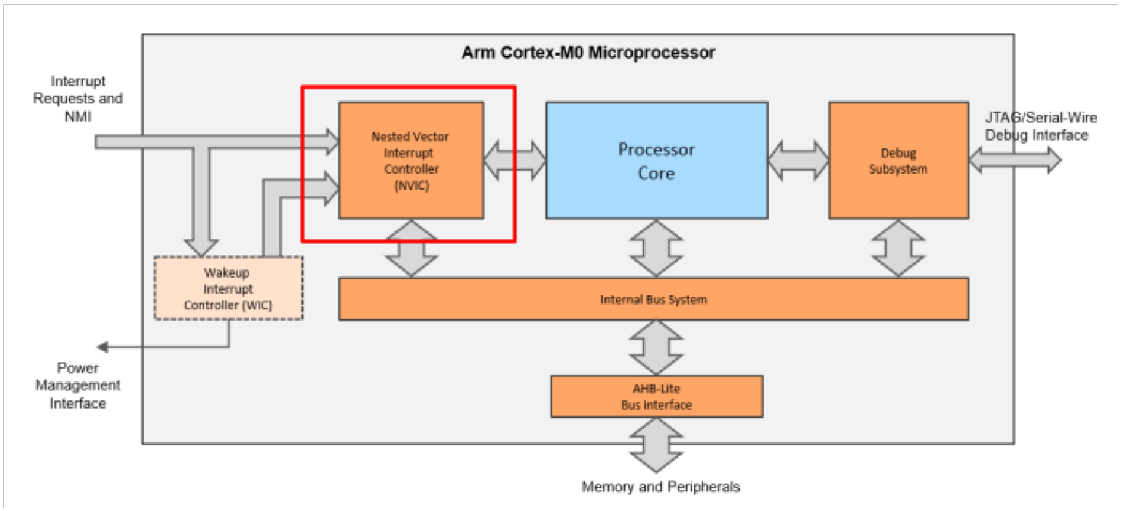


Figure 4. NVIC in Cortex-M0 Microprocessor

4.4 UART peripheral

“The Nexys A7 contains an FTDI FT2232HQ USB-UART bridge (connected to connector J6) that allows usage of Windows COM port commands to communicate with the board using PC software. USB packets are converted to UART/serial port data using free USB-COM port drivers, which can be found at www.ftdichip.com under the "Virtual Com Port" or VCP header. A two-wire serial interface (TXD/RXD) and optional hardware flow control (RTS/CTS) are used to communicate with the FPGA. Following the installation of the drivers, I/O commands from the PC can be directed to the COM port to generate serial data

traffic on the C4 and D4 FPGA pins. The transmit LED (LD20) and the receive LED (LD21) are two on-board status LEDs that provide visual feedback on traffic passing through the port (LD19). Signal designations that indicate direction are from the perspective of the DTE (Data Terminal Equipment), which in this case is the PC. To configure UART interrupt to send characters to a PC or laptop, mechanism to generate interrupt if the receiver FIFO is not empty can be implemented."

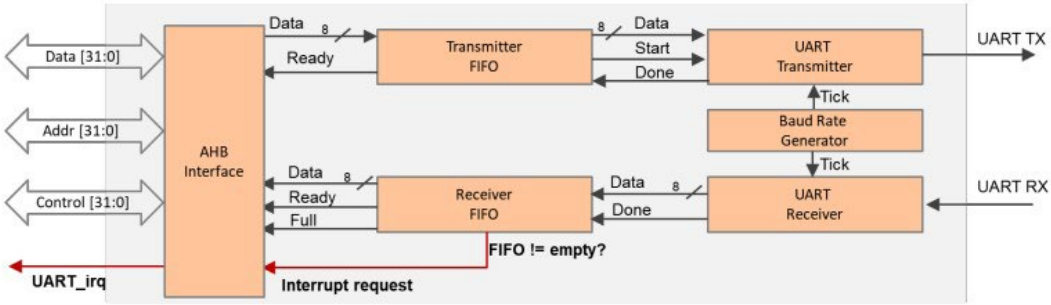


Figure 5 UART Interrupt Signal

5. IMPLEMENTATION

5.1 Tasks for Double Player Tic Tac Toe Game:

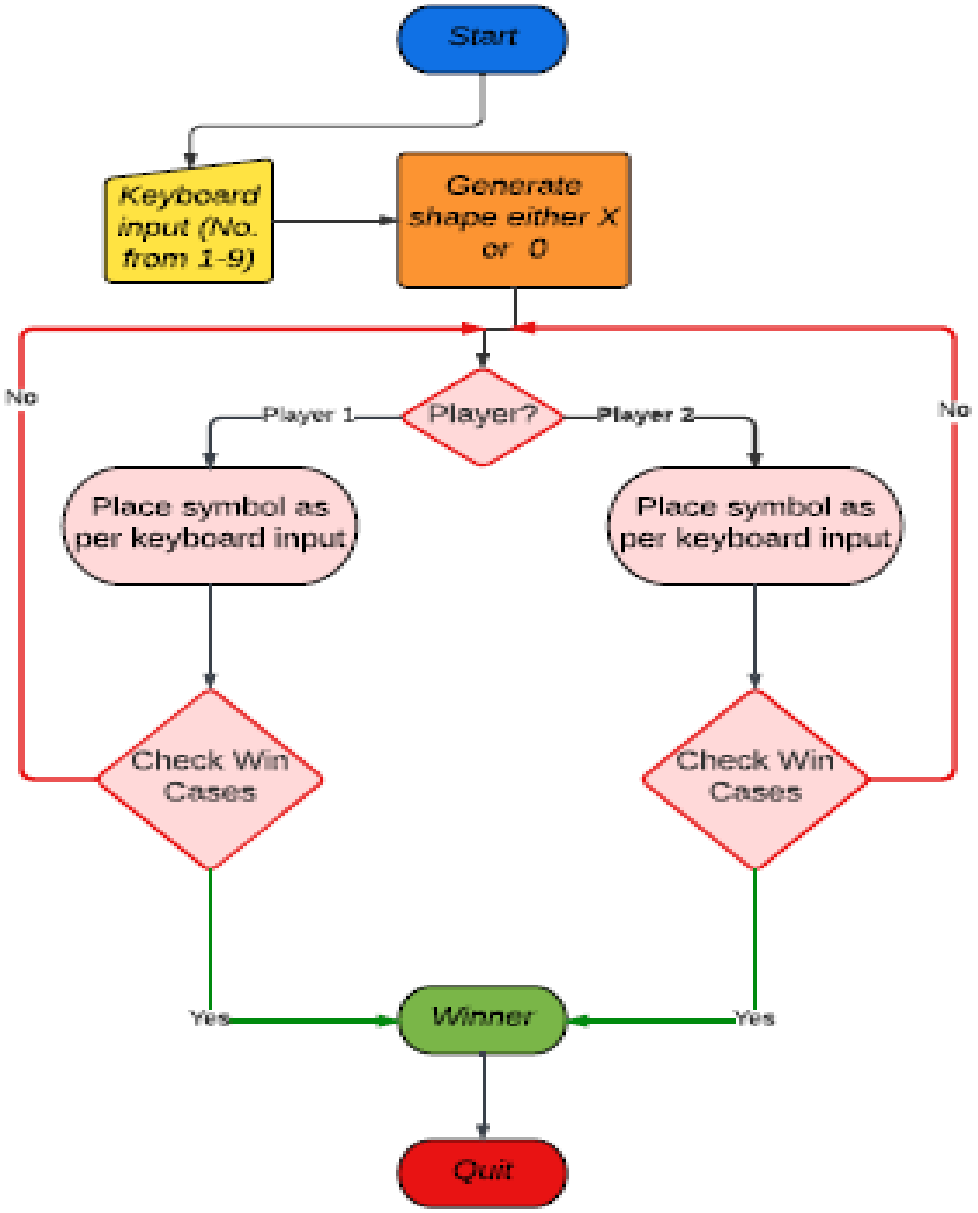


Figure 6 Flowchart of Double player Tic Tac Toe application

Above figure 13, flowchart explains basic working flow of the code for double player Tic Tac Toe game.

Task 1: Define Game region and various boundaries for display region

- Define the game region using #define statements. Using this the left, right, top and bottom boundaries of the game is defined.
- Adjusting various boundaries in Display region
    - Ex: rectangle(5,top_boundary, 86,top_boundary+boundary_thick,BLUE);//draws top boundary

Task 2: Writing text for the game rules and player choices

- void Game_Init(void) function is used to write the rules of the game, how to quit and reset and player choices in the text region.

Task 3: Game Initialization and invalid inputs

- For 2 player game, two constructs are constructed namely x and y.
- initialise_score(void) function is used to initializes the score.
- GameOver (void) function is used to quit the game, reset the game and also it has the mechanism to identify if user has input some invalid input i.e any key other than 1-9 while playing the game.

Task 4: Drawing the game grid on the VGA drawing area

Board() function helps in drawing the tic tac toe sample game grid in the text region so as to help the users to understand which key amongst 1-9 is to be given as input for playing the game.

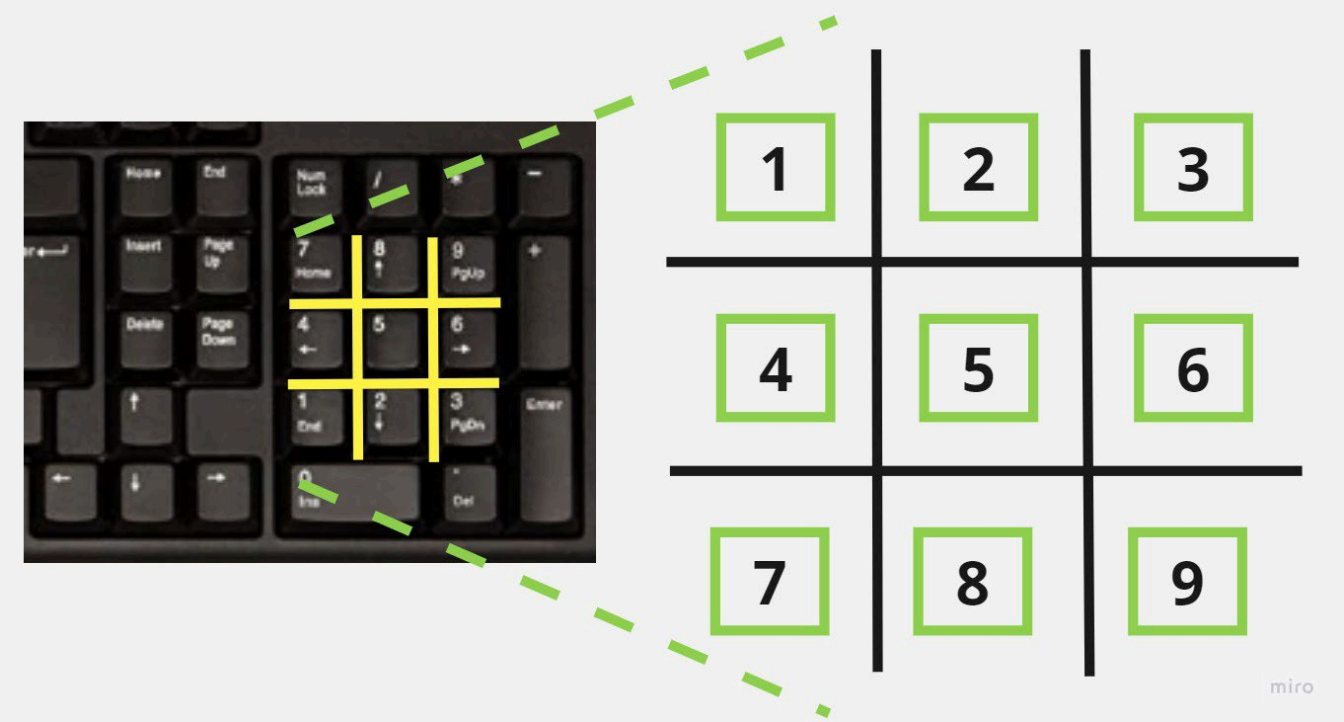


Figure7: Sample game grid in text region

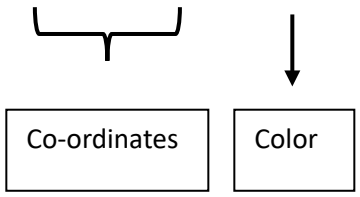
Task 5: Win check scenario after each pass

Via the Checkwin() function every time a pass is played the logic checks if there is a winner. The logic checks that if any of the 8 combinations(3 horizontal, 3 vertical and 2 diagonal) of winning is satisfied then the message is displayed saying which player has won.

Task 6: Function for X- and O-pixel plot

The call_draw_X(int t , int u) and call_draw_O(int t , int u) function is required to generate the pixel plot of the the actual X and O drawn in the game in the display region. VGA_plot_pixel is used to define the coordinates and the color of the structure of X and O.

Ex : VGA_plot_pixel(n.a[0], n.b[0], WHITE)



Task 7: To decode the character X and O to draw in the grid and identifying invalid inputs

The draw_X (int t) and draw_O (int t)function is used to identify the character drawn is X / O from the logic of the function call_draw_X (int t , int u) and call_draw_O (int t, int u) respectively. So , if player chooses 1st square to insert X/O then according to the call_draw_X / call_draw_O function to insert X/O in the 1st square. This function sets the coordinates of where to insert the X/O inside the square that is in exact middle/ right/left/top depending on coordinates. Additionally, if any other square apart from 1-9 is chosen then a error message is shown as invalid input is entered.

Task 8: To identify which square amongst the grid is selected by player.

The draw() function is kind of a top module function under which all the other functions related to drawing X/O (call_draw_X and call_draw_O), selecting exactly where to place X/0 inside the square(draw_X and draw_O). When this function is activated all the mentioned functions under draw() function are called and the tasks are executed.

Task 9: Logic for UART peripheral to be able to inputs from keyboard.

The UART_ISR(void) function is the function which is used to invoke the UART peripheral so that players can enter 1-9 numbers from keyboard to be able to play the game.

6. CONCLUSION

In this project, higher level software drivers and CMSIS drivers were used to program the SoC and peripherals to build a TIC TAC TOE game application. The game's boundary impacts, as well as the score increment procedure, were detected using a timed interrupt, and the results were shown on the VGA monitor. A UART interrupt was created to adjust the game's orientation by sending different characters to the SoC through a PC or laptop. The driver functions of CMSIS are used to implement both interruptions. The tasks are carried out by invoking embedded C code ISRs from assembly code.

7. RESULT

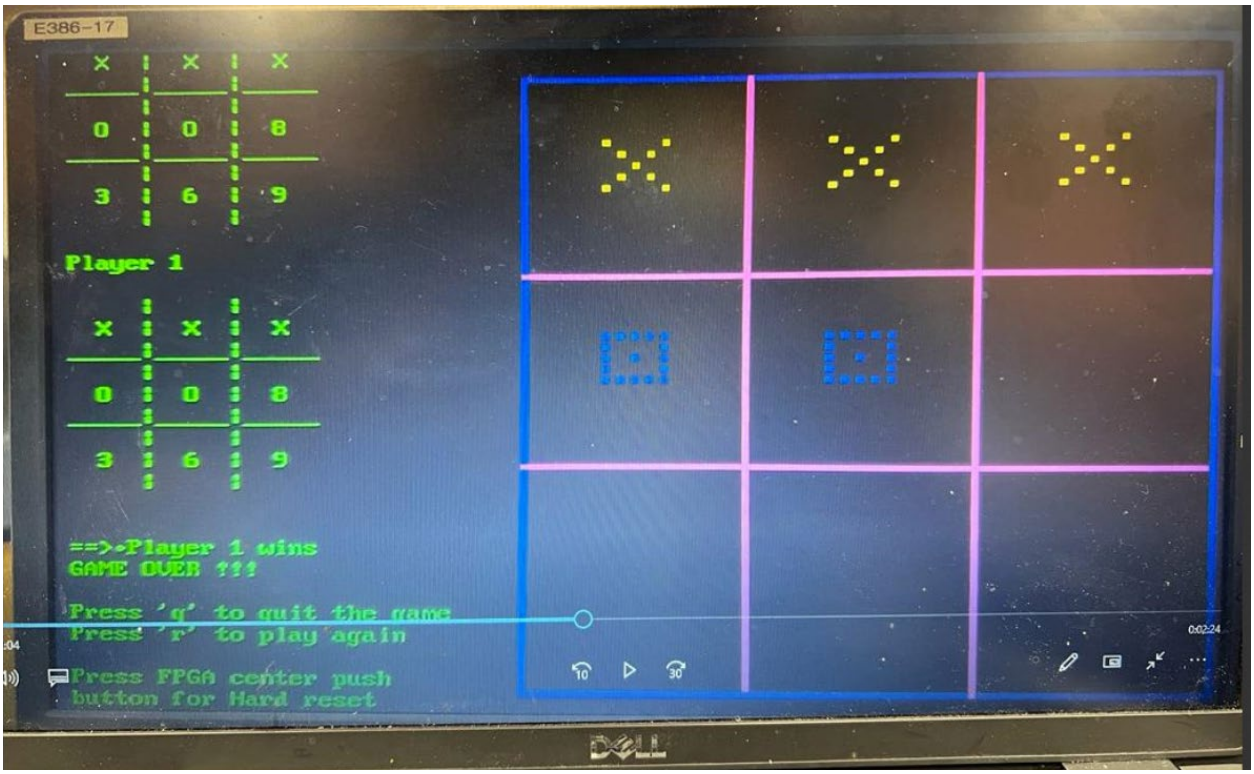
The video links for the working of the project and code explanation are below:

Code Explanation: https://drive.google.com/file/d/13TJ-zEITAS6kk-gAVMZNx9A-wbpLlTvU/view?usp=sharing

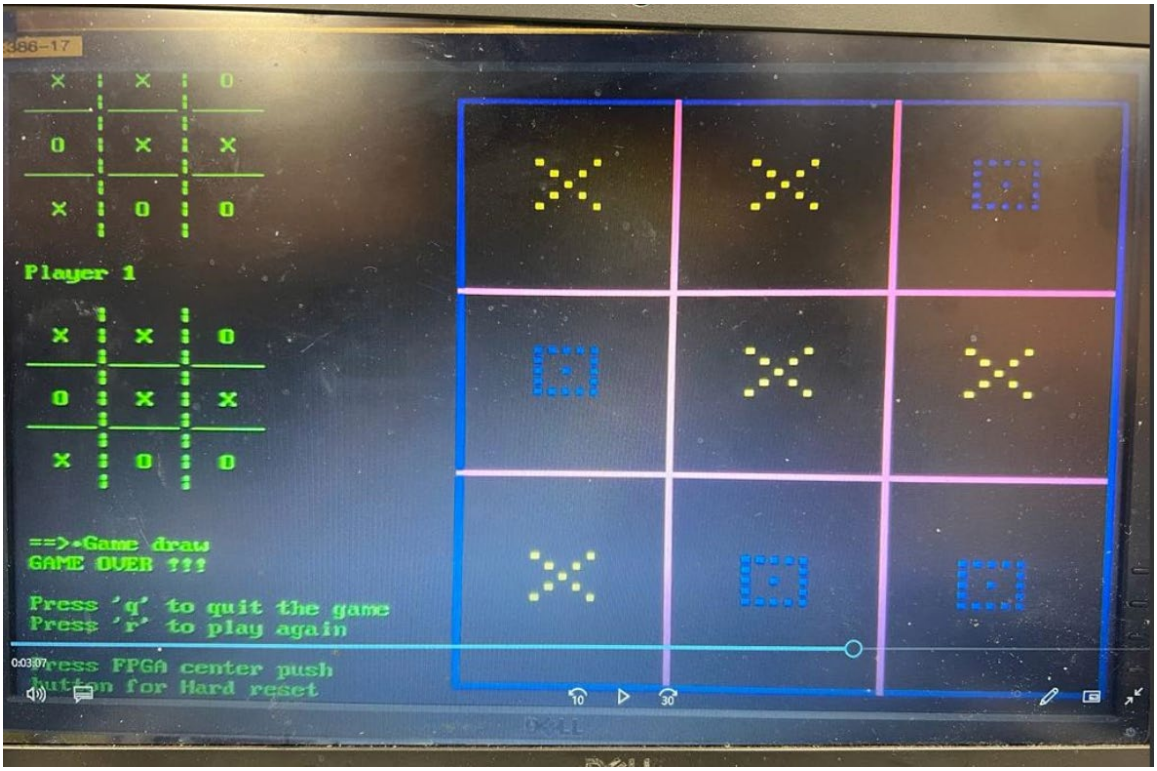
Project Working: https://drive.google.com/file/d/1TjJ0MxSgXiB6TeaJU78Vdp-UGkodvpBg/view?usp=sharing.

Following images show the result obtained while playing the tic tac toe game:

Win Scenario:



Draw Scenario:



Invalid Output Scenario: