# ECE 385

Spring 2024

Final Project

# FPGA-based Tetris Game

# Name:Jie Wang, Shitian Yang

Student ID: 3200112404, 3200112415

Prof. Chushan Li,

Prof. Zuofu Cheng

ZJU-UIUC Institute

May 24th , 2024, Friday D-225

TA: Jiebang Xia

**Demo Point: 25/30**

## Introduction

### 1.1 Project Overview

The objective of this project is to develop a real-time Tetris game on an FPGA using SystemVerilog and C. Based on the Avalon Bus for IP communication, the game features a VGA display for graphical output and a PS/2 keyboard for user input. The project includes the basic game logic, rendering graphics, handling user inputs, and managing game states. With our knowledge learned through *ECE385: Digital Systems Lab,* we provide a new illustration for this classical game.

图表

中度可信度描述已自动生成

**Fig 1:** Original Taris Game Background, drawn by DALL-E. This image is too big for the FPGA to render, thus we need to compress and redraw it in 160 \* 120 pixels size.

### 1.2 Motivation and Goals

This project was chosen because it provides a clear basis for the difficulty and development. With basis on 'Lab 8: SOC with USB and VGA Interface in S’, we keep improving our hardware and software design skills. The goals include gaining practical experience in digital system design, enhancing proficiency in SystemVerilog and C, and understanding the integration of hardware and software components in embedded systems.

We also participated in ***"Eaton Cup"*** *ECE 385 FPGA Platform Digital Design Competition,* winning ‘excellent design award’ and received beautiful gifts from Eaton Co.

 一群人坐在电脑前

中度可信度描述已自动生成

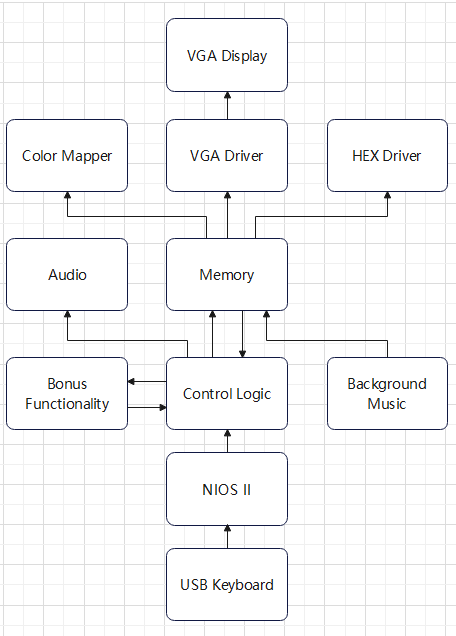
**Fig 2(left):** EATON, our course sponsor.

**Fig 3(right):** Demo to the Professors on our game.

## System Design

### 2.1 Block Diagram

Figure 4 is the block diagram of our system, with Keyboard Handler, Game Logic Controller, Audio Handler, VGA Display Handler, Memory Module, Timer Module and Score and Level System.



**Fig 4:** System Architecture with corresponding SV module and C code.

### 2.2 Description of Modules

* **Keyboard Handler**: Captures user inputs from the PS/2 keyboard to control the game. Including
* **Game Logic Controller**: Manages the game state, including piece generation, movement, rotation, and collision detection.
* **Audio Handler**: Manages background music and sound effects.
* **VGA Display Handler**: Renders the game graphics on the VGA display.
* **Memory Module**: Stores the game state, including the current position of pieces and cleared lines.
* **Timer Module**: Controls the game speed and progression by generating timing signals.
* **Score and Level System**: Tracks the player’s score and adjusts the game difficulty level.

### 2.3 Hardware Setting: IP Cores and Avalon Bus

The hardware component is the FPGA, which runs the SystemVerilog modules for game logic and display rendering. The software component is a C program running on an embedded processor within the FPGA, handling game states, score tracking, and user input processing.

**IP Cores Used:**

1. **Nios II Processor (nios2\_gen2\_0)**:
   * **Address**: 0x0000\_1000-0x0000\_17FF
   * **Function**: Controls the overall game logic and coordinates between hardware and software.
2. **On-Chip Memory (onchip\_memory2\_0)**:
   * **Address**: 0x0000\_0000 - 0x0000\_000F
   * **Description**: Stores the state of the game grid and active pieces.
3. **SDRAM Controller (sdram)**:
   * **Address**: 0x1000\_0000 - 0x17FF\_FFFF
   * **Description**: Provides additional memory for game data storage, ensuring smooth gameplay.
4. **SDRAM PLL (sdram\_pll)**:
   * **Address**: 0x0000\_0090 - 0x0000\_009F
   * **Description**: Generates the clock signals required by the SDRAM controller.
5. **System ID Peripheral (sysid\_qsys\_0)**:
   * **Address**: 0x0000\_00A8 - 0x0000\_00AF
   * **Description**: Provides a unique identifier for the system.
6. **JTAG UART (jtag\_uart\_0)**:
   * **Address**: 0x0000\_00B0 - 0x0000\_00B7
   * **Description**: Facilitates communication between the FPGA and a host computer for debugging purposes.
7. **PS/2 Keyboard Input (keycode)**:
   * **Address**: 0x0000\_0080 - 0x0000\_008F
   * **Description**: Handles inputs from the keyboard to control game pieces.
8. **Various OTG HPI Controllers**:
   * **Avalon Memory Mapped Slaves**
     + **Address**: otg\_hpi\_address\_s1: 0x0000\_0070 - 0x0000\_007F
     + **Address**: otg\_hpi\_data\_s1: 0x0000\_0060 - 0x0000\_006F
     + **Address**: otg\_hpi\_r\_s1: 0x0000\_0050 - 0x0000\_005F
     + **Address**: otg\_hpi\_w\_s1: 0x0000\_0040 - 0x0000\_004F
     + **Address**: otg\_hpi\_cs\_s1: 0x0000\_0030 - 0x0000\_003F
     + **Address**: otg\_hpi\_reset\_s1: 0x0000\_0020 - 0x0000\_002F
   * **External Connections**: Conduits for each OTG HPI module
   * **Description**: Interfaces for handling OTG communications and control signals.

**Function Description of Each IP**

* **VGA Controller**: Drives the VGA monitor, updating the display based on the current game state. It interprets the game grid data and renders the appropriate colors and shapes.
* **PS/2 Keyboard Interface**: Handles user inputs, translating key presses into game actions like moving or rotating tetrominoes.
* **Memory Controller**: Ensures efficient storage and retrieval of game state information, supporting real-time updates and smooth gameplay.

## 3. Features and Functionality

* **Piece Generation, Movement, and Rotation**: Random generation of tetrominoes, with movement and rotation controlled by the player.
* **Basic Scoring System**: Tracks and displays the player’s score.
* **Simple Color Graphics**: Different colors for different tetrominoes.
* **Audio Background Music (BGM)**: Plays background music during the game.

## 4. Implementation

### 4.1 SystemVerilog Implementation

For the detailed SystemVerilog module description, please refer appendix A.

#### 4.1.1 Game Logic Controller

**Implementation**

* **Module Name**: game\_logic.sv
* **Description**: Implements the core mechanics of the Tetris game, including piece generation, movement, collision detection, line clearing, and scoring.
* **Interface**:
  + **Inputs**:
    - clk: System clock.
    - reset: System reset signal.
    - key\_input: Signals from the PS/2 keyboard.
  + **Outputs**:
    - grid\_state: Current state of the game grid.
    - score: Current game score.

**FSM Design**

* **States**:
  1. **INIT**: Initializes game state.
  2. **IDLE**: Waits for user input.
  3. **MOVE**: Updates position of the active tetromino.
  4. **ROTATE**: Rotates the active tetromino.
  5. **COLLISION\_CHECK**: Checks for collisions.
  6. **LINE\_CLEAR**: Clears completed lines and updates the score.
  7. **GAME\_OVER**: Ends the game when conditions are met.

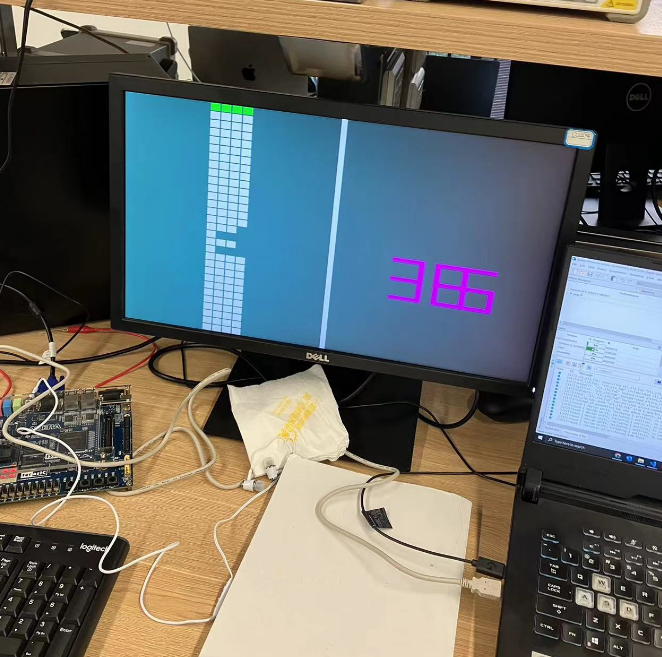


Fig:

#### 4.1.1 VGA Display Handler

**Implementation**

* **Module Name**: VGA controller.sv
* **Description**: Manages the rendering of the game state on a VGA display.

**Rendering Logic**

* Uses double buffering to avoid flickering.
* Converts game grid data into VGA-compatible signals.
* Supports different colors for each tetromino type.



Fig: Background image test and Block debug mode, score display test.

#### 4.1.1 Keyboard Handler

**Implementation**

* **Module Name**: keyboard\_handler.sv
* **Description**: Interprets PS/2 keyboard inputs and converts them into control signals for the game logic.
* **Interface**:
  + **Inputs**:
    - clk: System clock.
    - ps2\_data: Data from the PS/2 keyboard.
  + **Outputs**:
    - key\_input: Control signals for game actions (A- left, D- right,W- rotate,S- drop).

**Key Mapping**

* WASD keys for movement.

### 4.2 C Code Implementation

The C program handles the overall game state, user input processing, and score tracking. It communicates with the SystemVerilog modules via memory-mapped I/O.

## 5. Testing and Debugging

### 5.1 Testing Strategy

The system was tested using unit tests for individual modules, integration tests for combined modules, and system tests for the entire game. Both simulation and on-hardware testing were performed.

### 5.2 Debugging Process

Debugging tools included simulation software for SystemVerilog and printf debugging for C. The process involved identifying and fixing issues related to timing, synchronization, and data transfer between modules.

### 5.3 Test Results

The test results confirmed the correct functionality of the game logic, input handling, and VGA display. Issues such as timing mismatches and synchronization errors were resolved during testing.

电脑显示屏

描述已自动生成

**Fig:** Final Demo image, with colorful background image, sprite icon, dynamic score, and game display. It is sad that we didn’t took more photos / videos for our game, because we are hurry to write senior design thesis and take final exams (And we have to return the FPGA board).

## 6. Challenges and Solutions

### 6.1 Technical Challenges

* **Timing and Synchronization**: Ensuring proper timing and synchronization between hardware and software components.
* **Data Transfer**: Efficiently transferring large amounts of data between the C program and SystemVerilog modules.

### 6.2 Project Management Challenges

* **Time Constraints**: Managing time effectively to meet project deadlines.
* **Workload Management**: Balancing the workload between team members and ensuring consistent progress.

## 7. Future Work

### 7.1 Planned Enhancements

* **Increasing Levels of Difficulty**: Game speed increases with each level.
* **High Score Table**: Stores and displays high scores using non-volatile memory.
* **Enhanced Graphical Effects**: Includes animations for line clears and other events.
* **Game Loading from a CD**: Inspired by console game loading mechanisms.
* **Multiplayer Mode**: Allows multiple players to play simultaneously.
* **Improved Graphics**: Adding more detailed and visually appealing graphics.
* **Network Multiplayer**: Implementing a networked multiplayer mode.

### 7.2 Potential Improvements

* **Performance Optimization**: Enhancing the performance of the game by optimizing the hardware and software design.
* **User Interface**: Improving the user interface for a better gaming experience.

## 8. Conclusion

### 8.1 Summary of Achievements

Although having time limitation, our project successfully implemented 80% functionality of a real-time Tetris game on an FPGA, including features such as piece generation, movement, rotation, collision detection, background image, background music and a scoring system. This enables us to win the second prize in final competition.

### 8.2 Lessons Learned

Key lessons include the importance of timing and synchronization in FPGA design, effective debugging techniques, and the value of teamwork in managing complex projects.

### 8.3 Final Thoughts

The project provided a comprehensive learning experience in digital system design, combining hardware and software development. It has potential for further enhancements and serves as a solid foundation for future projects.

## 9. References

[1] OpenAI, "DALL·E 3," OpenAI, 2024. [Online]. Available: <https://openai.com/index/dall-e-3/>.

[2] TinyVGA, "VGA Signal Timing for 640x480 @ 60Hz," TinyVGA, 2024. [Online]. Available: http://tinyvga.com/vga-timing/640x480@60Hz.

[3] weixin\_44830487, "基于fpga的俄罗斯方块," CSDN, 2023. [Online]. Available: <https://blog.csdn.net/weixin_44830487/article/details/115972133>.

[4] FPGA4Fun, "Pong Game," FPGA4Fun, 2024. [Online]. Available: <https://www.fpga4fun.com/PongGame.html>.

[5] Terasic Inc., "Altera DE2-115 Development and Education Board," Terasic, 2024. [Online]. Available: [https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502&PartNo=4#](https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=502&PartNo=4).

[6] Lan Tian, "Cyclone IV FPGA Development Bugs Resolve," Lan Tian @ Blog, 2024. [Online]. Available: <https://lantian.pub/article/modify-computer/cyclone-iv-fpga-development-bugs-resolve.lantian/>.

[7] H. Ye, "FPGA Tetris Game," GitHub, 2016. [Online]. Available: https://github.com/hanchenye/FPGA-tetris?tab=readme-ov-file.

## 10. Appendices

* **Block Diagrams**: Detailed diagrams of the system architecture.
* **Code Snippets**: Key sections of the SystemVerilog and C code.
* **Test Logs**: Detailed logs of the testing process and results.

窗体顶端