

FPGA-Based Tetris Game

Design Report for ECE385

"Eaton Cup" ECE 385 FPGA Platform Digital Design Competition 2024

Team:

- Jie Wang, Shitian Yang
- Student ID: 3200112404, 3200112415
- May 22rd, 2024

Advisors:

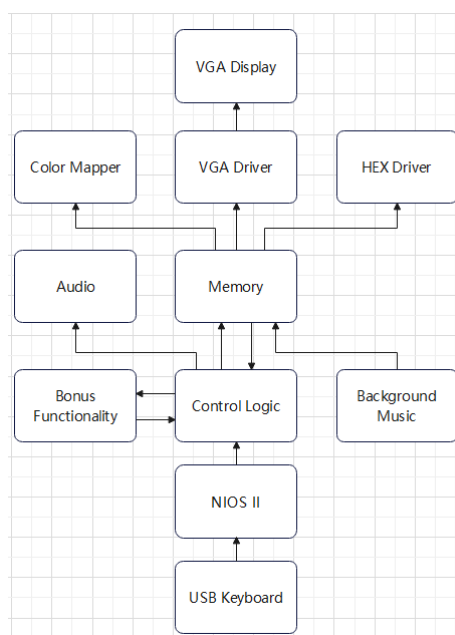
- Prof. Chushan Li, Prof. Zuofu Cheng
 - TA: Jiebang Xia
 - ZJU-UIUC Institute
-

1. Introduction

Overview

Our FPGA-based Tetris game leverages the Avalon Bus for IP communication, VGA for display, and PS/2 keyboard for user input. The system integrates multiple modules to manage game logic, display, audio, and user interaction.

Block Diagram Description



IP Cores and Avalon Bus

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.
-

2. Implementation and Interface Definition of Main Modules

2.1 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.

2.2 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.

2.3 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.
 - Spacebar for Bomb on the screen, destroy 5*5 blocks.
-

3. Implementation Process of the C Algorithm

Overview

The C algorithm manages the overall game state, score tracking, and user input processing. It ensures synchronization between hardware and software components.

Execution Cycle

1. **Initialization:** Sets up the game environment, initializes variables, and starts the game loop.
2. **Main Loop:**

- **Input Handling:** Reads input from the PS/2 keyboard.
- **Game Logic Update:** Updates the game state based on inputs and timer interrupts.
- **Display Update:** Sends the updated game state to the VGA controller.
- **Sound Update:** Triggers sound effects based on game events.

Data Synchronization

- **Registers Used:**
 - `grid_state_reg`: Stores the current state of the game grid.
 - `score_reg`: Stores the current score.
 - `input_reg`: Captures user inputs from the keyboard.
 - **Mechanism:**
 - The hardware modules write to these registers during each clock cycle.
 - The software reads from these registers to update the game state and display.
 - Interrupts are used to handle real-time updates and ensure smooth gameplay.
-

Conclusion

This report outlines the design, implementation, and interfacing of the FPGA-based Tetris game. By leveraging the Avalon Bus for IP communication and integrating SystemVerilog and C components, we achieve a real-time, interactive gaming experience using FPGA Development Board. Our project not only demonstrates technical understanding in ECE385: Digital Systems but also creativity and innovation, positioning us for the Eaton Competition.

