ECE532H1S: Digital Hardware
Project Group Report

SUPER MARIO BROS.FPGA

Daniel Ng
990829181

Seung Woo Hur
990828994

Date: March 27, 2005

Overview

Goals:

The goal of this project is to recreate the Super Mario Bros. video game on the Xilinx Virtex-II Multimedia Board. The functionality of the project is split into three sections:

- Video output
- User input
- Audio output

The main focus of the project is on the development of the video output section. This involves the creation of a video core to display a graphical output from the SVGA interface on the multimedia board to a monitor. After the video core is developed, a core that accepts keyboard input from the UART is created. Finally, an audio core that plays MP3 files is used to generate audio output through speakers connected to the multimedia board.

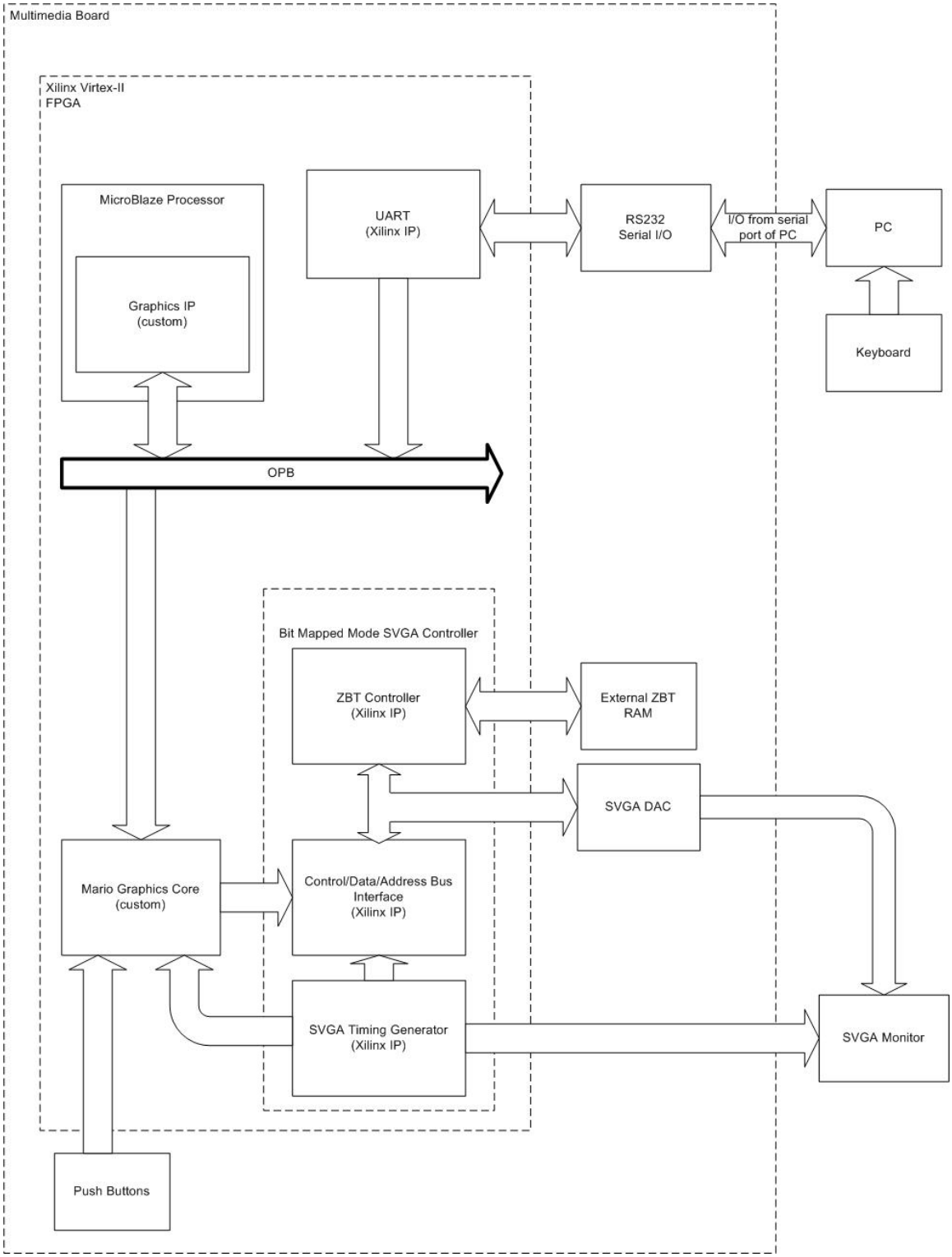

Figure 1: Block diagram of initialized planned project

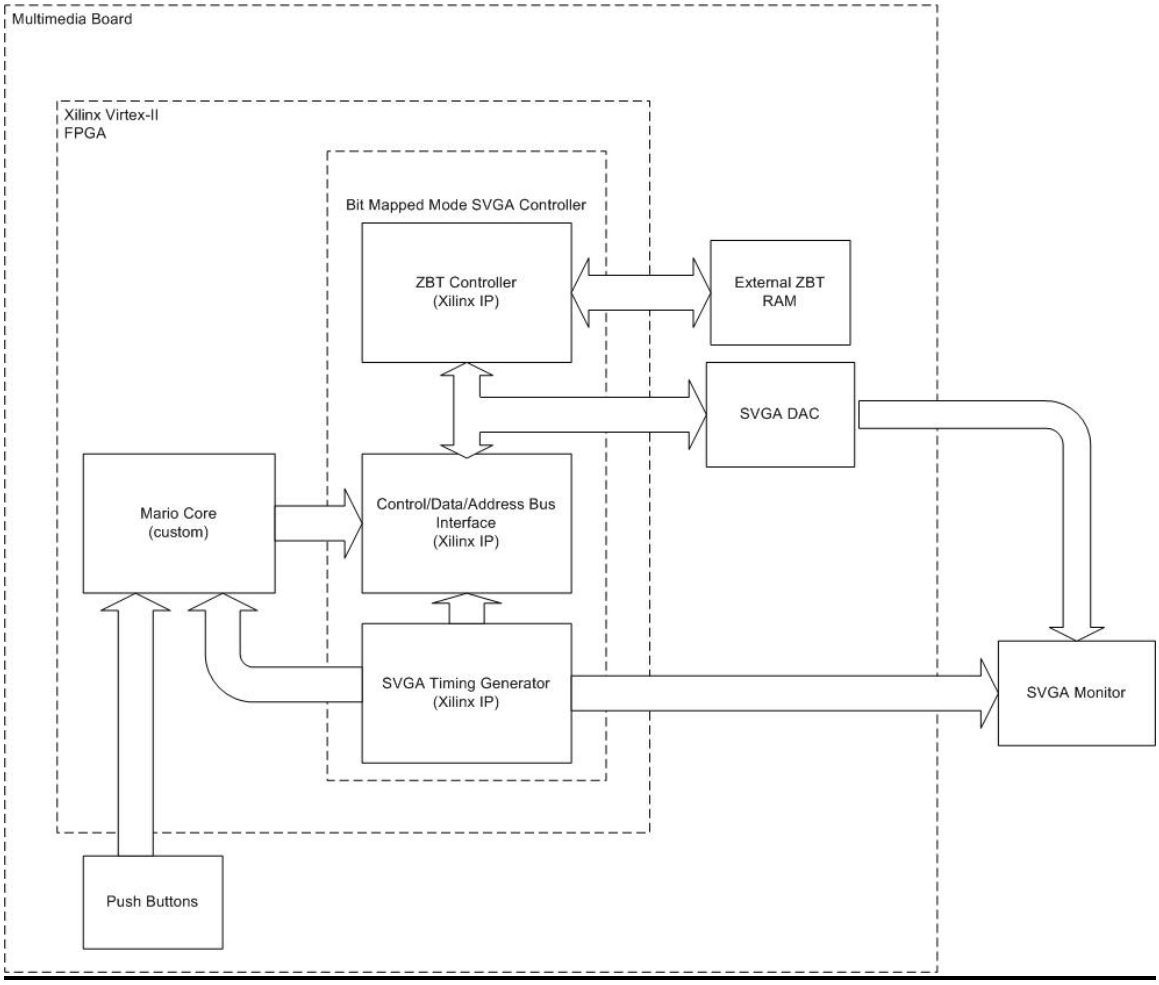

Figure 2: Block diagram of the final product

Outcome

The results of the development of each section are:

Video Output:

Result
Video output was successfully completed. We were able to display a moving Mario and a moving enemy (goomba) moving on brown ground with a sky blue background. These objects were implemented as hardware state machines in Verilog.

Suggested Improvement
A significant improvement to this core would be to implement only the display controller in Verilog, while providing the inputs to the controller in software. This would simplify the programming process and drastically decrease the size of the synthesized graphics core.

User Input:

Result
User input was implemented but with the push buttons on the multimedia board instead of the keyboard. Pressing a button resulted in Mario flying in an up-right direction.

Suggested Improvement
Implement the keyboard as the user input instead of the push buttons.

Audio Output:

Result
Audio output was not successfully completed.

Suggested Improvement
Implement the MP3 tutorial module to play in-game music while the game was running.

Brief Descriptions of IP Blocks Used in Final Product

Our final product does not contain any software IPs, and thus it does not utilize the MicroBlaze processor. The project consists of the Bit Mapped Mode SVGA IP provided by Xilinx and our custom Mario IP written in Verilog.

Bit Mapped Mode SVGA IP:

The Verilog code for this IP was provided Xilinx's Multimedia Board Examples web page. The IP consists of two different cores: an SVGA timing generator and a ZBT memory controller.

SVGA Timing Generator Core

SVGA timing generator creates the timing signals for the SVGA output, controls the memory address provided to the DAC, and generates a user_access_ok signal that dictates when data can be written into the ZBT memory.

ZBT Memory Controller Core

ZBT memory controller accepts the control, address and data signal inputs and carries out write/read operations on the external ZBT memory. When user_access_ok is high, it

writes to the ZBT RAM, and when it is low, it reads from the ZBT and sends the read data to the DAC.

Mario IP:

The Mario IP core generates the graphics information that is sent to the ZBT memory and accepts user input from the push buttons of the multimedia board. The graphics are created by state machines that generate the data, address, and control signals for the bus interface to produce the proper graphics. The state machines are controlled by the user_access_ok signal and the pixel clock, which is the main system clock (27MHz). The code to provide push button input was modified from lab m06 and is used to control the state machine that controls how Mario is displayed on the screen.

Detailed Descriptions of the Blocks

Bit Mapped Mode SVGA IP:

The Bit Mapped Mode SVGA IP provided by Xilinx consists of 9 files that are structured in the following hierarchy:

BM_MODE_SVGA_CTRL.v
- SVGA_TIMING_GENERATION.v
- DRIVE_DAC_DATA.v
- MEMORY_CTRL.v
  - ZBT_CONTROL.v
    - PIPELINES.v
    - DATA_BUS_INTERFACE.v
    - ADDR_BUS_INTERFACE.v
    - CTRL_BUS_INTERFACE.v

BM_MODE_SVGA_CTRL.v

This file is the top level module for the bit mapped mode SVGA display. It contains SVGA output ports, ZBT memory input/output ports, user input ports, a reset input port, and a pixel clock input port. The SVGA output, ZBT memory, reset, and pixel clock ports are connected to pins on the multimedia board. The user input ports define the information written into the ZBT memory when the user_access_ok signal received from the SVGA_TIMING_GENERATION module is HIGH. This module instantiates the SVGA_TIMING_GENERATION, DRIVE_DAC_DATA, and MEMORY_CTRL modules.

SVGA_TIMING_GENERATION.v

This module creates the horizontal synch, vertical synch, composite synch, and composite blank timing signals for the SVGA output, controls the memory address input to the DAC, and generates a user_access_ok signal that dictates when data can be written into the ZBT memory.

The module increments through each individual pixel from the top-left corner to the bottom-right corner on each rising edge of the pixel clock. For each pixel that is to be displayed on the screen, the user_access_ok output is asserted LOW and a memory address for that specific pixel of the screen is outputted to the DAC. The DAC reads the memory location and outputs the specific color for that monitor pixel. When the module increments past the last pixel on a line or the screen, the screen is temporarily blanked according to SVGA specifications. During this blanking, user_access_ok is asserted HIGH and data can be written into the ZBT memory.

DRIVE_DAC_DATA.v

This module generates the color output for a certain pixel and its inputs are controlled by the SVGA_TIMING_GENERATION and BM_MODE_SVGA_CTRL modules. The DAC input data is controlled by the SVGA_TIMING_GENERATION module, and the BM_MODE_SVGA_CTRL module generates the control input signals.

MEMORY_CTRL.v

This module and its submodules perform the read and write operations to the ZBT memory. Its inputs are controlled by the SVGA_TIMING_GENERATION module, the BM_MODE_SVGA_CTRL module, and the user inputs. When the user_access_ok signal is asserted LOW, the SVGA_TIMING_GENERATION module controls the memory. If user_access_ok is asserted HIGH, the BM_MODE_SVGA_CTRL module and the user inputs control the memory. This module instantiates the ZBT_CONTROL module, which instantiates the PIPELINES, DATA_BUS_INTERFACE, ADDR_BUS_INTERFACE, and CTRL_BUS_INTERFACE modules.

Mario IP: Mario.v (custom core)

The Mario core module instantiates and provides the inputs for the BM_MODE_SVGA_CTRL module to generate the required graphics. When the user_access_ok signal is asserted LOW, the BM_MODE_SVGA_CTRL module accepts the control, address, and data input signals for the memory from the Mario module. While user_access_ok is LOW, the address is incremented. Several state machines determine what data is written for a certain address location. To display Mario and the enemy, the state machine accepted a starting address input for both objects. The state machine draws these objects relative to the starting address provided. This starting address input is used to allow for object movement. The rest of the address locations that

do not display either Mario or the enemy are set to display either the blue sky or the brown floor.

This module also counts each displayed frame during a period of one second. The count resets to zero after each second. For this project, the monitor runs at 60 Hz, so 60 frames are counted each second. This frame count is used to implement movement on the screen. For example, to make Mario's legs move, our state machine is set to draw his legs at a certain position for the first 30 frames of the second. For the second 30 frames, the legs are redrawn at a different position to simulate leg movement. This idea is also used to make Mario and the enemy move across the screen. For every 10 frames of the second, the starting address for Mario and the enemy are incremented a certain amount.

Design Tree Description:

The zip file sent contains a folder that should be put into the pcores directory of an XPS project, the UCF file, and this group report.

The folder contains the folders and files necessary to add an instance of the Super Mario core to an XPS project. The files include the Verilog files for our core, the netlist file generated from synthesis in ISE, mpd file, pao file, and bbd file.

Any XPS project that does not use the ZBT memory bank0 and the SVGA DAC will run Super Mario properly when its bit stream is generated and downloaded.