Simple System-on-Chip Interface Using MicroBlaze Processor

1. <u>Introduction:</u>

The MicroBlaze processor is a 32-bit CPU designed for System-on-Chip (SoC) projects, acting as a system controller for tasks like user interface management and data I/O. Programmable in C, it interfaces with peripherals—such as LEDs and switches—and enables more complex functionalities via the AXI bus, making it well-suited for embedded applications on FPGAs.

The week 2 design extends the Microblaze-based System-on-Chip (SoC) design by integrating a USB host controller (MAX3421E) for USB keyboard communication and a VGA controller for graphical display. The Microblaze processor controls the system, processing input from the USB keyboard to allow user interaction, specifically to move a ball displayed on a VGA/HDMI monitor. The ball starts in the center of the screen and responds to directional key presses (W, A, S, D) for movement, bouncing off screen edges until a new direction command is issued. The design utilizes SPI for communication with the MAX3421E and incorporates an IP block to convert VGA signals to HDMI output.

2. <u>Description & Diagrams of Microblaze System:</u>

I/O Description for Lab 6.1

Input operations primarily utilize AXI GPIO modules, which are configured to read signals from devices such as buttons and switches for lab 6.1. The MicroBlaze processor accesses the GPIO input registers through memory-mapped I/O to determine the current state of these inputs. Additionally, the AXI Uartlite module allows the processor to receive data serially from external devices. The processor continuously monitors the UART input buffer for incoming data, enabling it to react promptly to user commands or external signals.

Output operations also leverage the AXI GPIO and AXI Uartlite modules. The GPIO can be set up to control external components, such as LEDs to display the binary sum for lab 6.1, by writing to the output registers that dictate the state of the pins. This allows the MicroBlaze to send visual feedback to users. Simultaneously, the AXI Uartlite module facilitates sending data back to external devices, which can be useful for debugging or providing status updates. Moreover, the I/O operations are synchronized with the processor's clock, and the system can utilize interrupts to ensure timely responses to user inputs, such as button presses.

MicroBlaze Interaction with MAX3421E and Ball Motion

The MicroBlaze processor interacts with the MAX3421E USB chip through a Serial Peripheral Interface (SPI) communication protocol, specifically using the AXI Quad SPI module. This setup allows the MicroBlaze to send commands and receive data from the MAX3421E, enabling USB functionality in the system. By writing to and reading from specific registers in the MAX3421E, the MicroBlaze can manage USB devices, handle data transfers, and respond to USB events. The SPI clock and chip select signals are crucial for synchronizing this communication, ensuring that commands are executed in the correct sequence.

Simultaneously, the MicroBlaze coordinates with the ball motion components by processing user input through the GPIO modules. The keycodes received from the user (via buttons) are interpreted by the MicroBlaze to determine the desired direction of the ball. The processor updates the ball's position accordingly by computing the next coordinates based on user commands and predefined motion rules. This updated position is then communicated to the VGA controller, which displays the ball's movement on the screen. This interaction creates a seamless loop where user inputs influence ball motion while the system maintains USB connectivity for additional functionality.

VGA Operation and Interaction with Ball and Color Mapper

The VGA operation in the system involves generating the necessary synchronization signals and pixel data required to display graphics on a screen. The VGA controller module is responsible for producing horizontal and vertical sync pulses, along with a composite sync signal, which are required for maintaining the correct timing for the VGA display. It generates coordinates for each pixel drawn on the screen, specifically managing a resolution of 640x480 pixels. The VGA controller operates at a pixel clock of 25 MHz and uses horizontal and vertical counters to track the current pixel and line being rendered. During the active display period, the VGA controller indicates when pixels can be drawn and provides the current pixel coordinates (drawX and drawY) to other modules in the system.

The Ball module is responsible for updating the position of a ball on the VGA display based on user inputs and defined motion constraints. It utilizes the frame clock to update its position and direction, which is determined by key presses. The ball's position is continuously checked against the display boundaries, and it bounces back when it reaches the edges of the display area. The Ball module outputs the updated coordinates (BallX and BallY) and its size (BallS) to the Color Mapper module.

The Color Mapper module takes the pixel coordinates from the VGA controller (drawX and drawY) and the ball's coordinates from the Ball module to determine if a pixel falls within the boundaries of the ball. It employs a mathematical formula to check whether the current pixel is inside a defined circular area based on the ball's position and size. If the pixel lies within the ball, the Color Mapper sets the appropriate RGB color values for that pixel, representing the ball’s color, which is orange in this case. If the pixel is outside the ball's area, the Color Mapper assigns a background color to that pixel, which is the gradient of the white color. The output RGB values from the Color Mapper are then used by the VGA controller to render the final image on the screen. This interaction creates a dynamic visual display where the ball's movement is continuously updated based on user input and rendered accurately on the VGA output.

VGA-HDMI IP Description

The VGA-HDMI IP serves as a critical component for converting analog VGA signals into HDMI, a digital format. This conversion allows devices with VGA output to connect seamlessly to HDMI input devices, facilitating compatibility between older and newer display technologies. The HDMI format not only supports higher resolutions and refresh rates but also integrates audio transmission alongside video in a single cable, making it a more versatile choice for modern applications. In contrast, VGA's limitations in resolution and the need for separate audio connections highlight the evolution of video transmission technologies.

Despite these differences, VGA and HDMI share commonalities in their fundamental purpose of delivering video signals from a source to a display. Both formats have specific connector types—VGA featuring a larger D-sub connector and HDMI offering more compact variants—ensuring standardization across various devices. Their widespread adoption across consumer electronics signifies their importance in enabling visual representation, although HDMI has largely supplanted VGA in high-definition applications due to its superior capabilities.

3. Top Level Block Diagram:

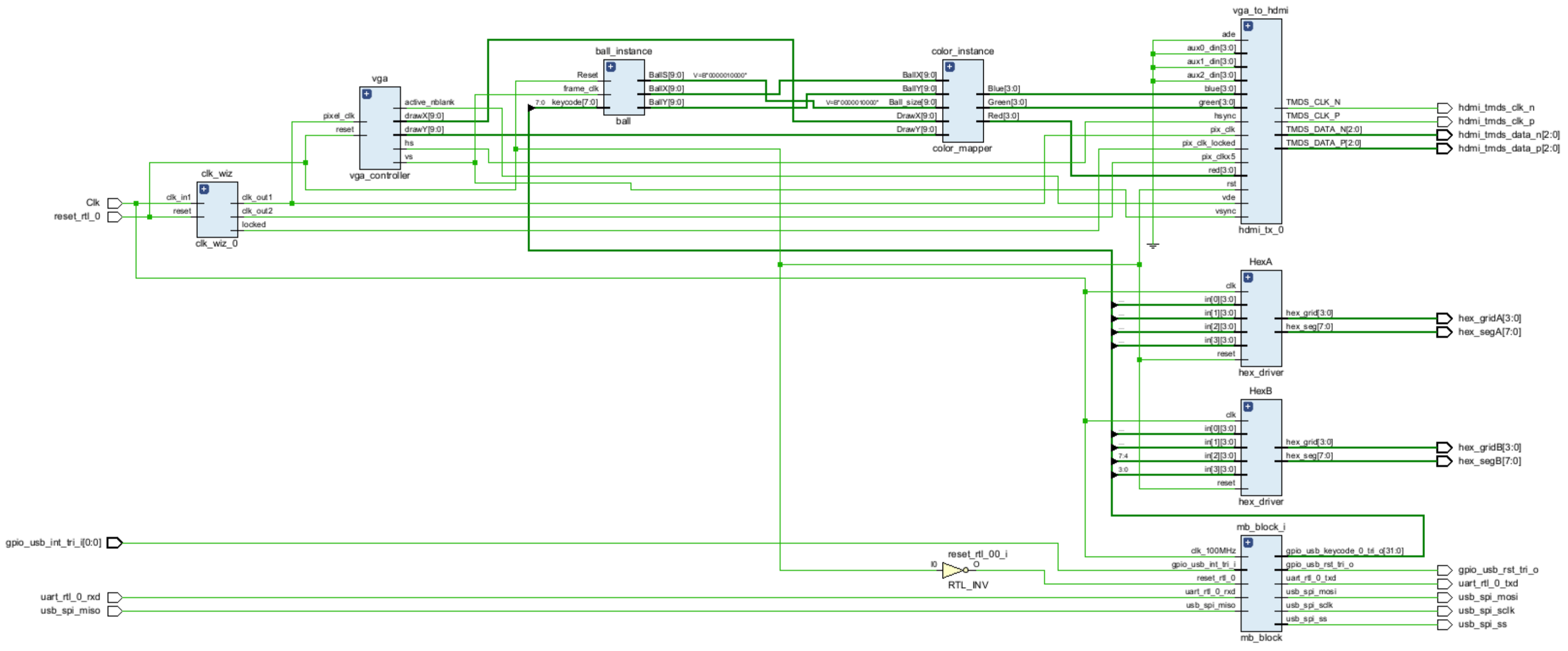

Figure 1: Block diagram of the MicroBlaze and the other modules

4. <u>Software Components</u>

Accumulator and Blinker Description

The accumulator code is designed to manipulate the state of LEDs based on inputs from buttons and switches. It continuously checks the state of the buttons and switches, updating the LED output accordingly. The main loop checks for conditions such as overflow (when all LEDs are on) and resets the LED state when a specific button is pressed. If the user presses the button while the LEDs are not all lit, the code adds the values from the switches to the current LED output, which allows for cumulative control of the LED states. The program also handles scenarios where the button is pressed to reset the LEDs, providing feedback through console messages when an overflow is detected.

In contrast, the blinker code is a simpler program that focuses on turning an LED on and off at regular intervals, effectively creating a blinking effect. This code operates in an infinite loop, first turning the LED on by writing a specific value to the GPIO data address, waiting for a second, and then clearing that value to turn the LED off. The blinker code also includes print statements to provide real-time feedback in the console regarding the LED's state.

- **What are some primary differences between the various presets that are available?**

Preset	Focus
Microcontroller	Used for simple-oriented projects with basic performance using minimal resources. Perfect for small-scale tasks.
Real-Time Processor	Used for time-sensitive tasks with enhanced timers and interrupts
Application Processor	Used for high-performance tasks, offering large memory, more power, and advanced peripherals.

5. Conclusion:

In conclusion, this lab successfully demonstrated the integration of multiple hardware and software components into a cohesive System-on-Chip (SoC) design using the MicroBlaze processor. Through this project, we explored MicroBlaze's capabilities in managing peripheral devices and real-time tasks, including USB keyboard input, VGA/HDMI output, and interactive ball movement. Each component, from the VGA controller to the color mapping module, played a vital role in creating a responsive and dynamic display system, reinforcing key concepts in embedded system design.

The project underscored the importance of AXI bus communication, synchronization signals, and real-time processing in complex digital systems. By achieving stable communication with the USB host controller (MAX3421E) and generating accurate VGA signals for HDMI output, we demonstrated how programmable logic and processor cores can be used to create interactive, embedded applications.