GPIO Module for RISC-V SoC

EE 705: VLSI DESIGN LAB 2025

Team: Silicon Titans
Saurav Raj (21D070064)
Mrudul Jambhulkar (21D070044)
Prayag Mohanty (24M1177)

February 21, 2025

Abstract

This report details the design and implementation of a General Purpose Input/Output (GPIO) module for a RISC-V SoC. The module allows direction control, input/output operations, and interrupt handling for 32 GPIO pins using a memory-mapped register interface. The implementation follows an AXI-Lite protocol to enable seamless integration into a SoC.

1 Introduction

The GPIO (General Purpose Input/Output) module provides direction control, input/output operations, and interrupt handling for 32 GPIO pins through a memory-mapped register interface. RISC-V processors often rely on GPIOs for basic input/output operations, making them essential for implementing low-level control and communication tasks.

2 Functional Description

The purpose of this project is to design a GPIO controller with an AXI-Lite interface that facilitates read and write operations for controlling GPIO pins. The controller should allow configuration of pin directions (input or output), enable data writing to output pins, and read data from input pins. Additionally, the design should support interrupt generation for specific GPIO events. This GPIO controller will be interfaced with an AXI-Lite master interface for integration with a larger system-on chip (SoC) architecture.

2.1 Requirements

1. *GPIO Pin Direction Configuration:* The controller should be capable of configuring the direction of individual GPIO pins, either as input(0) or output(1), using the AXI-Lite interface.

2. *Data Input/Output:* The controller should allow writing data to GPIO output pins and reading data from GPIO input pins. The module can set or clear specific GPIO pins without affecting others .

3. *Interrupts:* The controller should generate interrupts on certain GPIO events (e.g., edge detection or state changes) and read the status of interrupts. The controller should be able to enable or disable interrupts for specific GPIO pins. It supports software-triggered interrupt generation for GPIOs as well as well as acknowledgement and clearing of active GPIO interrupts . It supports configuration of GPIO interrupt polarity and interrupt mode for GPIO pins (edge or level triggered).

4. *AXI-Lite Interface:* The module should support a standard AXI-Lite interface for control and data access.

2.2 GPIO Features

- Supports 32 GPIO pins.
- Direction configuration for each pin (input/output).
- Output register for controlling pin states.
- Interrupt system for event-driven operations.
- Memory-mapped interface for software control.

3 Memory-Mapped Register Definitions

The GPIO module registers are mapped to specific addresses for software access.

Address	**Register Name**	**Description**
0x00	GPIO_DIRECTION	Controls GPIO direction (input/output)
0x04	GPIO_INPUT	Reads real-time input values
0x08	GPIO_OUTPUT	Controls output state of GPIOs
0x0C	GPIO_OUTPUT_SET	Sets specific GPIO output bits
0x10	GPIO_OUTPUT_CLR	Clears specific GPIO output bits
0x14	GPIO_INT_MASK	Enables/disables GPIO interrupts
0x18	GPIO_INT_SET	Software-triggered interrupts
0x1C	GPIO_INT_CLR	Clears active GPIO interrupts
0x20	GPIO_INT_STATUS	Reads raw interrupt status
0x24	GPIO_INT_LEVEL	Configures GPIO interrupt polarity
0x28	GPIO_INT_MODE	Configures GPIO interrupt mode

Table 1: Memory-Mapped Registers for GPIO Module

4 GPIO Module Design

4.1 AXI Lite Interface

We have 2 Finite State Machine algorithms (FSM) to ensure proper synchronization and handshaking in the AXI-Lite protocol, allowing smooth and correct read/write operations within the GPIO module.

4.1.1 Write FSM:

- Starts in `AXI_WRITE_IDLE`, where it waits for a valid write address and possibly data.
- If only the address is available, it moves to `AXI_WRITE_DATA` to wait for the data.
- Once both are received, it transitions to `AXI_WRITE_RESP` to issue the response, then returns to idle upon acknowledgment.

4.1.2 Read FSM:

- It begins in `AXI_READ_IDLE`, waiting for a read address.
- On receiving a valid address, it moves to `AXI_READ_DATA`, outputs the corresponding data, and then goes back to idle after the master acknowledges the data.

4.2 Writing data to memory mapped registers

The GPIO module (slave) receives the data from the master via AXI protocol and writes to the desired memory mapped register based on the corresponding address .

4.3 GPIO input

Controller takes the external GPIO input value and stores in read-only-register and value can be read by sending the address corresponding to input register .

4.4 GPIO output

The controller checks for conditions like set/clear and and places the final output on GPIO output port .

4.5 Interrupts

The controller has registers for interrupt configuration . It can enable/disable interrupts via GPIO Interrupt Mask register . It checks for interrupt events - rising / falling edges , level triggers (active high /active low) and raises a flag based on the interrupt mode configuration . The controller can also set/clear interrupts and read the status of active interrupts and generate the final interrupt flag .

4.6 Block Diagram

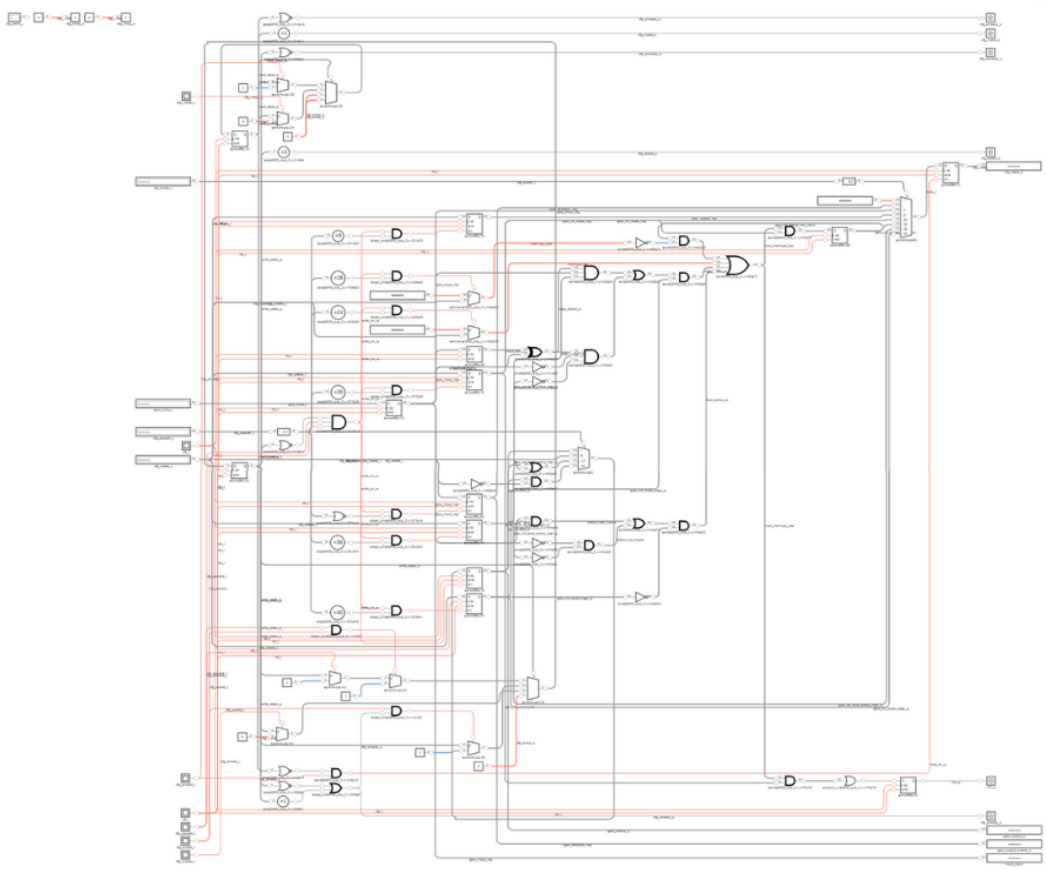

Figure 1: Block Diagram of GPIO Module

5 Testbench and Verification

A Verilog testbench is written to verify the functionality of the GPIO module. The testbench performs:

- GPIO direction configuration test.
- Input and output data verification.
- Interrupt handling validation.

6 Results and Observations

The GPIO module was synthesized .The observed behavior matched the expected results.

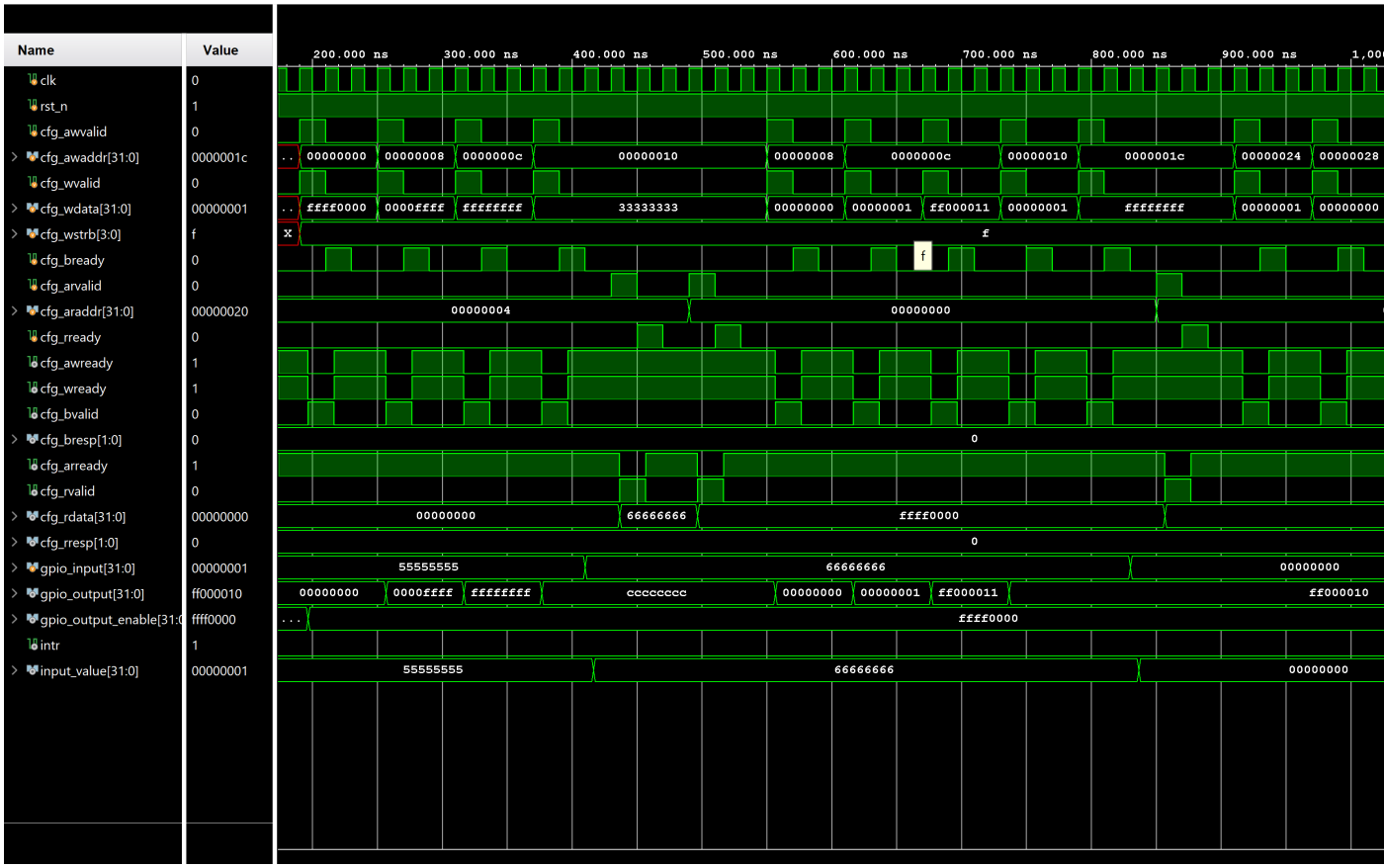

Figure 2: Simulation results

7 Conclusion

The GPIO module was successfully designed, implemented, and verified. It provides a functional interface for configuring GPIO pins, handling inputs/outputs, and managing interrupts within a RISC-V SoC.