GPIO Interface Module Specification

Introduction

The `gpio_s` module is a General Purpose Input/Output (GPIO) interface designed to provide configurable input and output capabilities for digital signals. This module supports dynamic configuration of input/output directions, interrupt handling, and optional pull-up/pull-down resistors. It is suitable for applications requiring flexible digital I/O control, such as microcontroller interfacing and peripheral device management.

Architecture

The `gpio_s` module is structured as a single top-level module with the following key components: - **Parameterization**: The module is highly configurable through parameters that define input/output behavior, interrupt settings, and address space. - **Registers**: Internal registers manage I/O direction (`IO_DIR`), output values (`IO_OUT`), interrupt masks (`INTMASK`), and interrupt flags (`INTFLAGS`). - **Interrupt Handling**: The module supports edge-triggered interrupts, configurable for rising, falling, or both edges. Interrupts can be dynamically configured or statically set via parameters. - **I/O Control**: The module provides bidirectional I/O control with optional bus keepers and pull-up/pull-down resistors.

Key Operations

- **Read/Write Operations**: The module supports read and write operations to configure I/O directions and output values.
- **Interrupt Detection**: Edge detection logic is implemented to set interrupt flags based on configured edge sensitivity.
- **Dynamic Configuration**: Parameters allow for dynamic configuration of I/O and interrupt settings, enabling flexible adaptation to different application requirements.

Interface

Signal	Width	In/Out	Description
`clk`	1	Input	Clock signal for synchronous operations.
`rst`	1	Input	Asynchronous reset signal, active high.
`addr`	16	Input	Address bus for register access.
`wr`	1	Input	Write enable signal for register access.

Signal	Width	In/Out	Description
`rd`	1	Input	Read enable signal for register access.
`bus_in`	8	Input	Data bus for input data during write operations.
`bus_out`	8	Output	Data bus for output data during read operations.
`int`	1	Output	Interrupt output signal, active high when any interrupt flag is set.
`int_rst`	1	Input	Interrupt reset signal, active high.
`io`	8	Inout	Bidirectional I/O port for external signal interfacing.

Parameters

Parameter	Type	Default Value	Description
`DINAMIC_IN_OUT_CONFIG`	String	"FALSE"	Enables dynamic I/O direction configuration.
`IN_OUT_MASK_CONFIG`	8-bit	8'h00	Static I/O direction mask.
`USE_INTERRUPTS`	String	"FALSE"	Enables interrupt functionality.
`DINAMIC_INTERRUPT_CONFIG`	String	"FALSE"	Enables dynamic interrupt configuration.
`INTERRUPT_MASK_CONFIG`	8-bit	8'h00	Static interrupt mask configuration.
`INTERRUPT_UP_DN_EDGE_DETECT`	8-bit	8'h00	Configures edge detection for interrupts.
`INTERRUPT_BOTH_EDGES_DETECT_MASK`	8-bit	8'h00	Configures both edge detection for interrupts.
`BUS_KEPPER_EN_MASK`	8-bit	0	Enables bus keeper functionality.
`BUS_PULL_UP_EN_MASK`	8-bit	0	Enables pull-up resistors.
`BUS_PULL_DN_EN_MASK`	8-bit	0	Enables pull-down resistors.

Parameter	Type	Default Value	Description
`ADDRESS`	Integer	0	Base address for register access.
`BUS_ADDR_DATA_LEN`	Integer	16	Width of the address bus.

Timing

The `gpio_s` module operates synchronously with the `clk` signal. All register updates and I/O operations are triggered on the rising edge of the clock. The module supports asynchronous reset (`rst`), which initializes all internal registers and flags to their default states.

Usage

To use the `gpio_s` module, follow these steps: 1. **Configuration**: Set the desired parameters for I/O direction, interrupt settings, and address space before synthesis. 2. **Initialization**: Apply a high signal to `rst` to initialize the module. Release `rst` to begin normal operation. 3. **I/O Control**: Use the `addr`, `wr`, and `rd` signals to configure I/O directions and output values via the `bus_in` and `bus_out` buses. 4. **Interrupt Handling**: If interrupts are enabled, monitor the `int` signal for active interrupts. Use `int_rst` to clear interrupt flags as needed. 5. **Dynamic Configuration**: If dynamic configuration is enabled, update I/O and interrupt settings during operation as required.

This specification provides a comprehensive overview of the `gpio_s` module, enabling hardware engineers to implement and utilize the module effectively in their designs.

Functional Description (Generated by funcgen)

Verilog Design Modules Functional Description

Module: gpio_s (File: gpio_s.v)

Purpose

The `gpio_s` module is a General-Purpose Input/Output (GPIO) interface designed to control input and output pins with optional interrupt capabilities. It allows dynamic configuration of pin directions, reading or writing data, and managing interrupt signals for edge detection.

Parameters

- **DINAMIC_IN_OUT_CONFIG**: (Default = "FALSE") Configures if the input/output mask can be dynamically updated.
- **IN_OUT_MASK_CONFIG**: (Default = 8'h00) Predefines static configuration for IO direction if dynamic configuration is disabled.
- **USE_INTERRUPTS**: (Default = "FALSE") Enables or disables the use of interrupt functionalities.
- **DINAMIC_INTERRUPT_CONFIG**: (Default = "FALSE") Determines if interrupt configuration can be dynamically set.
- **INTERRUPT_MASK_CONFIG**: (Default = 8'h00) Static configuration for which pins can trigger interrupts.
- **INTERRUPT_UP_DN_EDGE_DETECT**: (Default = 8'h00) Specifies the edge detection configuration for interrupts when dynamic configuration is disabled.
- **INTERRUPT_BOTH_EDGES_DETECT_MASK**: (Default = 8'h00) Configures which pins should detect interrupts on both rising and falling edges.
- **BUS_KEPPER_EN_MASK**: (Default = 0) Enables bus-keeper functionality to hold bus states.
- **BUS_PULL_UP_EN_MASK**: (Default = 0) Enables pull-up resistors on specified pins.
- **BUS_PULL_DN_EN_MASK**: (Default = 0) Enables pull-down resistors on specified pins.
- **ADDRESS**: (Default = 0) Base address for accessing the module's registers.
- **BUS_ADDR_DATA_LEN**: (Default = 16) The width of the addressing bus for internal control, typically set to accommodate larger addressing space requirements.

Ports

- **rst**: (Input, 1-bit) Asynchronous reset that initializes or resets the module.
- **clk**: (Input, 1-bit) Clock input for synchronous operations.
- **addr**: (Input, BUS_ADDR_DATA_LEN-bit) Address bus for selecting registers within the module.
- **wr**: (Input, 1-bit) Write enable signal to indicate when bus data should be written to registers.
- **rd**: (Input, 1-bit) Read enable signal to direct reading from module registers.
- **bus_in**: (Input, 8-bit) Data input bus for transferring data to module registers.
- **bus_out**: (Output, 8-bit) Data output bus for exporting data from the module.
- **int**: (Output, 1-bit) Interrupt output signal indicating if an interrupt condition has been triggered.

- **int_rst**: (Input, 1-bit) Resets the interrupt logic, typically used after an interrupt has been serviced.
- **io**: (Inout, 8-bit) Bi-directional I/O bus for direct interaction with the GPIO pins configured as inputs or outputs.

Internal Signals

- **IO_DIR**: (Reg, 8-bit) Determines the direction (input or output) of each GPIO pin.
- **IO_OUT**: (Reg, 8-bit) Stores the output values for GPIO pins configured as outputs.
- **INTMASK**: (Reg, 8-bit) Mask register for enabling specific interrupt conditions.
- **INTFLAGS**: (Reg, 8-bit) Flags register to store interrupt status for each pin, indicating an active interrupt condition.
- **PINCTRL[0:7]**: (Reg Array, 8-bit each) Control registers for each pin to configure interrupt conditions and behavior.
- **INT_DETECT_POSEDGE**, **INT_DETECT_NEGEDGE**: (Reg, 8-bit each) Registers for detecting rising (positive) and falling (negative) edge interrupts, respectively.
- **INT_DETECT_POSEDGE_n**, **INT_DETECT_NEGEDGE_n**: (Reg, 8-bit each) Inverted edge detection registers used for comparison logic to determine edge transitions.

Functionality

Sequential Logic

- On a rising edge of `clk` or a reset (`rst`) signal, the module initializes its registers or performs state updates:
  - Initializes direction, output, interrupt mask, and flag registers.
  - If `wr_int` is active, updates specific registers based on the address (`addr`) selection.

Combinational Logic

- Continuously updates the `bus_out` based on read operations (`rd_int`) from the command registers.
- Sets the I/O pins (`io`) based on configured directions and output states.
- Determines dynamic interrupt conditions and sets flags for them if enabled.

State Machines or Control Logic

- Uses case statements within write logic to selectively update control and data registers.

- Employs generate-for loops to manage multiple instances of the same logic pattern, particularly for detection of edge-triggered interrupts.

Instantiations

- **KEEPER**, **PULLUP**, **PULLDOWN**: Instantiated within a generate loop for each pin to optionally apply bus-keeper, pull-up, or pull-down functionality based on enable masks (`BUS_KEPPER_EN_MASK`, etc.).

Inter-Module Connections

Currently, `gpio_s` operates as a standalone module with no direct sub-module instantiations except for primitive logic elements like KEEPER, PULLUP, and PULLDOWN. It creates its own internal control and data pathways through its ports and parameters. While the design is not inherently hierarchical, it maintains versatile configurability and interaction through its bus and port structure, allowing integration as a building block in more complex systems.