omsp_gpio Specification

Introduction

The `omsp_gpio` module is a digital I/O interface designed to manage multiple GPIO ports. It provides configurable input/output operations for up to six ports, each with interrupt capabilities. The module is intended for use in embedded systems where efficient and flexible GPIO management is required. It adheres to standard digital design practices, ensuring compatibility with a wide range of digital systems.

Architecture

The `omsp_gpio` module is structured to handle up to six GPIO ports, each with its own set of registers for input, output, direction, interrupt flags, and function selection. The architecture includes:

- **Top-Level Structure**: The module consists of a main system clock (`mclk`) and a reset signal (`puc_rst`). It interfaces with peripheral data and address buses for configuration and data exchange.
- **Key Operations**: Each port can be individually enabled or disabled through parameters. The module supports reading and writing operations to the port registers, interrupt generation based on edge detection, and data synchronization.
- **Implementation Details**: The design uses parameterized masks to enable or disable ports, and a register decoder to manage register access. The module employs synchronous cells for input data synchronization and edge detection logic for interrupt generation.

Interface

Parameters

Parameter	Type	Default	Description
P1_EN	parameter	1'b1	Enable Port 1
P2_EN	parameter	1'b1	Enable Port 2
P3_EN	parameter	1'b0	Enable Port 3
P4_EN	parameter	1'b0	Enable Port 4
P5_EN	parameter	1'b0	Enable Port 5
P6_EN	parameter	1'b0	Enable Port 6

Clock and Reset

Signal	Width	In/Out	Description
mclk	1	Input	Main system clock
puc_rst	1	Input	Main system reset

Input Ports

Signal	Width	In/Out	Description
p1_din	8	Input	Port 1 data input
p2_din	8	Input	Port 2 data input
p3_din	8	Input	Port 3 data input
p4_din	8	Input	Port 4 data input
p5_din	8	Input	Port 5 data input
p6_din	8	Input	Port 6 data input
per_addr	14	Input	Peripheral address
per_din	16	Input	Peripheral data input
per_en	1	Input	Peripheral enable (active high)
per_we	2	Input	Peripheral write enable (active high)

Output Ports

Signal	Width	In/Out	Description
irq_port1	1	Output	Port 1 interrupt
irq_port2	1	Output	Port 2 interrupt
p1_dout	8	Output	Port 1 data output
p1_dout_en	8	Output	Port 1 data output enable
p1_sel	8	Output	Port 1 function select
p2_dout	8	Output	Port 2 data output
p2_dout_en	8	Output	Port 2 data output enable
p2_sel	8	Output	Port 2 function select
p3_dout	8	Output	Port 3 data output
p3_dout_en	8	Output	Port 3 data output enable
p3_sel	8	Output	Port 3 function select
p4_dout	8	Output	Port 4 data output
p4_dout_en	8	Output	Port 4 data output enable
p4_sel	8	Output	Port 4 function select
p5_dout	8	Output	Port 5 data output
p5_dout_en	8	Output	Port 5 data output enable
p5_sel	8	Output	Port 5 function select
p6_dout	8	Output	Port 6 data output
p6_dout_en	8	Output	Port 6 data output enable
p6_sel	8	Output	Port 6 function select
per_dout	16	Output	Peripheral data output

Internal Signals

Signal	Type	Description
reg_sel	wire	Local register selection
reg_addr	wire	Register local address
reg_dec	wire	Register address decode
reg_lo_write	wire	Low byte write enable
reg_hi_write	wire	High byte write enable
reg_read	wire	Register read enable
reg_hi_wr	wire	High byte write vector
reg_lo_wr	wire	Low byte write vector
reg_rd	wire	Register read vector

Timing

The `omsp_gpio` module operates synchronously with the `mclk` signal. All input signals are sampled on the rising edge of `mclk`, and outputs are updated accordingly. The module's latency is determined by the clock cycle required for register access and interrupt generation. Interrupts are generated based on edge detection, which is also synchronized to the clock.

Usage

To use the `omsp_gpio` module, follow these steps:

1. **Configuration**: Set the desired parameters (`P1_EN`, `P2_EN`, etc.) to enable the required ports.
2. **Input Provision**: Provide input data to the respective port data inputs (`p1_din`, `p2_din`, etc.).
3. **Register Access**: Use the peripheral address and data buses (`per_addr`, `per_din`) to read from or write to the port registers.
4. **Interrupt Handling**: Monitor the interrupt outputs (`irq_port1`, `irq_port2`) for any interrupt events and handle them in the system's interrupt service routine.
5. **Output Retrieval**: Read the data outputs (`p1_dout`, `p2_dout`, etc.) to obtain the current state of the ports.

The module supports both input and output operations, with configurable direction and function selection for each port. Ensure that the system clock and reset signals are properly connected to maintain synchronous operation.

Functional Description (Generated by funcgen)

Verilog Design Module Functional Description

Module: omsp_gpio

File: `omsp_gpio.v`

Purpose

The `omsp_gpio` module serves as a digital input/output interface, managing multiple ports (Port 1 to Port 6) for a system. It provides configuration and control over these ports, handling inputs and outputs, data direction, selection, and interrupt generation.

Parameters

- **P1_EN**: Default = `1'b1`. Enables Port 1.
- **P2_EN**: Default = `1'b1`. Enables Port 2.
- **P3_EN**: Default = `1'b0`. Enables Port 3.
- **P4_EN**: Default = `1'b0`. Enables Port 4.
- **P5_EN**: Default = `1'b0`. Enables Port 5.
- **P6_EN**: Default = `1'b0`. Enables Port 6.
- **P1_EN_MSK**: Derived from `P1_EN`; enables masking of Port 1 data.
- **P2_EN_MSK**: Derived from `P2_EN`; enables masking of Port 2 data.
- **P3_EN_MSK**: Derived from `P3_EN`; enables masking of Port 3 data.
- **P4_EN_MSK**: Derived from `P4_EN`; enables masking of Port 4 data.
- **P5_EN_MSK**: Derived from `P5_EN`; enables masking of Port 5 data.
- **P6_EN_MSK**: Derived from `P6_EN`; enables masking of Port 6 data.
- **DEC_WD**: Value = `6`. Sets decoder bit width for address decoding.
- **DEC_SZ**: Calculated as `(1 << DEC_WD)`; total combinations of decoder words.

Ports

- **irq_port1** (output, 1-bit): Interrupt request for Port 1.
- **irq_port2** (output, 1-bit): Interrupt request for Port 2.
- **p1_dout** (output, 8-bit): Port 1 data output.
- **p1_dout_en** (output, 8-bit): Port 1 data output enable.
- **p1_sel** (output, 8-bit): Port 1 function select.
- **p2_dout** (output, 8-bit): Port 2 data output.
- **p2_dout_en** (output, 8-bit): Port 2 data output enable.
- **p2_sel** (output, 8-bit): Port 2 function select.
- **p3_dout** (output, 8-bit): Port 3 data output.
- **p3_dout_en** (output, 8-bit): Port 3 data output enable.
- **p3_sel** (output, 8-bit): Port 3 function select.
- **p4_dout** (output, 8-bit): Port 4 data output.
- **p4_dout_en** (output, 8-bit): Port 4 data output enable.
- **p4_sel** (output, 8-bit): Port 4 function select.
- **p5_dout** (output, 8-bit): Port 5 data output.
- **p5_dout_en** (output, 8-bit): Port 5 data output enable.
- **p5_sel** (output, 8-bit): Port 5 function select.
- **p6_dout** (output, 8-bit): Port 6 data output.
- **p6_dout_en** (output, 8-bit): Port 6 data output enable.
- **p6_sel** (output, 8-bit): Port 6 function select.
- **per_dout** (output, 16-bit): Peripheral data output.
- **mclk** (input, 1-bit): Main system clock.
- **p1_din** (input, 8-bit): Port 1 data input.
- **p2_din** (input, 8-bit): Port 2 data input.
- **p3_din** (input, 8-bit): Port 3 data input.
- **p4_din** (input, 8-bit): Port 4 data input.
- **p5_din** (input, 8-bit): Port 5 data input.
- **p6_din** (input, 8-bit): Port 6 data input.
- **per_addr** (input, 14-bit): Peripheral address.
- **per_din** (input, 16-bit): Peripheral data input.
- **per_en** (input, 1-bit): Peripheral enable.
- **per_we** (input, 2-bit): Peripheral write enable.
- **puc_rst** (input, 1-bit): Main system reset.

Internal Signals

- **reg_sel** (wire, 1-bit): Indicates local register selection.
- **reg_addr** (wire, 6-bit): Local address derived from `per_addr`.
- **reg_dec** (wire, 64-bit): Register address decoder.
- **reg_lo_write** (wire, 1-bit): Low write enable for register.
- **reg_hi_write** (wire, 1-bit): High write enable for register.
- **reg_read** (wire, 1-bit): Register read enable.
- **reg_hi_wr** (wire, 64-bit): High write mask for registers.
- **reg_lo_wr** (wire, 64-bit): Low write mask for registers.
- **reg_rd** (wire, 64-bit): Read mask for registers.
- **p1in** (wire, 8-bit): Synchronization of Port 1 input data.
- **p1out** (reg, 8-bit): Port 1 output data register.
- **p1dir** (reg, 8-bit): Port 1 direction register.
- **p1ifg** (reg, 8-bit): Port 1 interrupt flag register.
- **p1ifg_set** (wire, 8-bit): Port 1 interrupt flag set.
- **p1ies** (reg, 8-bit): Port 1 interrupt edge select.
- **p1ie** (reg, 8-bit): Port 1 interrupt enable.
- **p1sel** (reg, 8-bit): Port 1 select register.
- Similar internal signals exist for Ports 2 through 6.
- **p1in_dly** (reg, 8-bit): Previous state of Port 1 input for edge detection.
- **p2in_dly** (reg, 8-bit): Previous state of Port 2 input for edge detection.

Functionality

Sequential Logic

- Registers like `p1out`, `p1dir`, `p1ifg`, `p1ies`, `p1ie`, and `p1sel` are updated based on the system clock (`mclk`) and reset (`puc_rst`). Each port (1 through 6) shares a similar structure.
- **Interrupt Generation**: Interrupt requests (`irq_port1` and `irq_port2`) are generated based on edge detection of the inputs, using delayed input signals (`p1in_dly` and `p2in_dly`) to detect edges (rising or falling), configured by `p1ies` and `p2ies`.

Combinational Logic

- Address decoding (`reg_dec`) is a combinational process using `per_addr` to generate register select signals for read and write operations.
- Combinational output logic combines various register reads depending on the `reg_rd` signal, producing the `per_dout` output.

State Machines or Control Logic

- There is no explicit state machine or control logic board here. The behavior largely revolves around the combinational and sequential update logic for each port register.

Instantiations

- **Sub-modules**: Instances of `omsp_sync_cell` are used to synchronize inputs (`p1_din` to `p6_din`) to the system clock domain. Each `omsp_sync_cell` connects its `clk` port to `mclk` and `rst` port to `puc_rst`.

Inter-Module Connections

The `omsp_gpio` module operates standalone and does not instantiate other complex modules except for instances of `omsp_sync_cell` used for synchronization. It encapsulates GPIO control functionality, interacting with a higher-level system through its multiple output and input ports. The top-level connectivity is determined by the peripheral addressing and enable signals (`per_addr`, `per_en`). Data input/output operations are managed through the ports (`p1_din` to `p6_din`, `p1_dout` to `p6_dout`) based on the control logic specified in the registers (`p1sel`, `p2sel`, etc.). The interrupt outputs provide signals to potentially trigger external control flows or processor interrupts.