GPIODevice Specification

Introduction

The `GPIODevice` module is designed to interface with a peripheral bus and manage General Purpose Input/Output (GPIO) operations. It provides a configurable number of GPIO pins, allowing for both input and output operations. The module supports interrupt generation based on GPIO input changes and is designed to be integrated into larger systems requiring GPIO control. It adheres to a simple peripheral bus protocol for register access and control.

Architecture

The `GPIODevice` module is structured to include several key components:

- **Top-Level Structure**: The module is parameterized by `ID` and `IO_COUNT`, allowing for flexible identification and configuration of the number of GPIO pins.
- **Peripheral Bus Interface**: Handles read and write operations to internal registers via a peripheral bus.
- **Registers**: Includes output enable (OE), output data, input data, and interrupt enable registers.
- **GPIO Handling**: Manages GPIO input buffering and output control, with interrupt generation based on input changes.
- **Device Selection**: Utilizes a `DeviceSelect` submodule to determine if the device is enabled based on the peripheral bus address.

Key Operations

- **Data Flow**: Data is read from or written to the GPIO pins through the peripheral bus interface, with registers controlling the direction and state of each pin.
- **Interrupt Generation**: Interrupts are generated when enabled GPIO pins detect a change in input state.

Implementation Details

- **Clock Domain**: The module operates synchronously with the `clk` signal.
- **Reset Behavior**: On reset (`rst`), all internal states and registers are initialized to default values.

Interface

Signal	Width	In/Out	Description
`clk`	1	In	Clock signal for synchronous operations.
`rst`	1	In	Asynchronous reset signal, active high.

Signal	Width	In/Out	Description
`peripheralEnable`	1	In	Enables the peripheral bus interface.
`peripheralBus_we`	1	In	Write enable signal for the peripheral bus.
`peripheralBus_oe`	1	In	Output enable signal for the peripheral bus.
`peripheralBus_busy`	1	Out	Indicates if the peripheral bus is busy.
`peripheralBus_address`	16	In	Address bus for accessing internal registers.
`peripheralBus_byteSelect`	4	In	Byte select signals for the peripheral bus.
`peripheralBus_dataRead`	32	Out	Data read from the peripheral bus.
`peripheralBus_dataWrite`	32	In	Data to be written to the peripheral bus.
`requestOutput`	1	Out	Indicates a request for output data.
`gpio_input`	IO_COUNT	In	Input signals from GPIO pins.
`gpio_output`	IO_COUNT	Out	Output signals to GPIO pins.
`gpio_oe`	IO_COUNT	Out	Output enable signals for GPIO pins.
`gpio_irq`	1	Out	Interrupt request signal generated based on GPIO input changes.

Timing

- **Latency**: The module operates with a single clock cycle latency for register read/write operations.
- **Signal Behavior**: All operations are synchronous to the rising edge of the `clk` signal. The `gpio_irq` is updated on the rising edge of `clk` based on the input changes.

Usage

To use the `GPIODevice` module: 1. **Initialization**: Ensure the module is reset by asserting the `rst` signal. 2. **Configuration**: Set the desired GPIO direction and initial output states by writing to the OE and output data registers via the peripheral bus. 3. **Operation**: Enable the module by asserting `peripheralEnable` and perform read/write operations using the peripheral bus signals. 4. **Interrupt Handling**: Configure the interrupt enable register to specify which GPIO inputs should trigger an interrupt. Monitor the `gpio_irq` signal for interrupt requests.

This module is suitable for applications requiring flexible GPIO control with interrupt capabilities, integrated into systems with a compatible peripheral bus interface.

Functional Description (Generated by funcgen)

Verilog Design Modules Functional Description

1. Module: GPIODevice

- **Source File**: `GPIODevice.v`

Purpose

The `GPIODevice` module is designed to interface a set of General-Purpose Input/Output (GPIO) pins with a peripheral bus. It manages input buffering, output control, and interrupt handling for the GPIO pins, allowing external control and communication through the peripheral bus.

Parameters

- **ID** (default value: 4'h0): A unique identifier for the device, used during device selection from a shared bus.
- **IO_COUNT** (default value: 16): Specifies the number of GPIO pins managed by the device.

Ports

- **clk** (input, 1-bit): Clock signal for synchronous operations.
- **rst** (input, 1-bit): Reset signal to initialize the module state.
- **peripheralEnable** (input, 1-bit): Enables the device to respond to bus transactions.
- **peripheralBus_we** (input, 1-bit): Write enable for bus transactions.
- **peripheralBus_oe** (input, 1-bit): Output enable for bus transactions.
- **peripheralBus_busy** (output, 1-bit): Indicates the bus is busy (currently unused, always 0).
- **peripheralBus_address** (input, 15:0-bit): Address input for accessing registers within the device.
- **peripheralBus_byteSelect** (input, 3:0-bit): Selects active bytes for bus transactions.
- **peripheralBus_dataRead** (output, 31:0-bit): Data read from the GPIO device to the bus.
- **peripheralBus_dataWrite** (input, 31:0-bit): Data written from the bus to the device.
- **requestOutput** (output, 1-bit): Indicates an output request from the registers.

- **gpio_input** (input, IO_COUNT-bit): Inputs from the GPIO pins to the device.
- **gpio_output** (output, IO_COUNT-bit): Controlled outputs from the device to the GPIO pins.
- **gpio_oe** (output, IO_COUNT-bit): Output enable signals for the GPIO pins.
- **gpio_irq** (output, 1-bit): Interrupt request signal.

Internal Signals

- **inputBuffered** (reg, IO_COUNT-bit): Buffered storage of input GPIO values for processing.
- **localAddress** (wire, 11:0-bit): Derived address within the device, post device selection.
- **deviceEnable** (wire, 1-bit): Indicates if the device is enabled and addressed from the peripheral bus.
- **oeRegisterOutputData** (wire, 31:0-bit): Output data from the output enable register.
- **oeRegisterOutputRequest** (wire, 1-bit): Request signal from the output enable register.
- **outputRegisterOutputData** (wire, 31:0-bit): Output data from the output register.
- **outputRegisterOutputRequest** (wire, 1-bit): Request signal from the output register.
- **inputRegisterOutputData** (wire, 31:0-bit): Output data from the input register.
- **inputRegisterOutputRequest** (wire, 1-bit): Request signal from the input register.
- **irqEnableRegisterOutputData** (wire, 31:0-bit): Output data from the interrupt enable register.
- **irqEnableRegisterOutputRequest** (wire, 1-bit): Request signal from the interrupt enable register.
- **irqEnable** (wire, IO_COUNT-bit): Interrupt enable mask for GPIO pins.
- **pinIRQ** (wire, IO_COUNT-bit): IRQ source computed from enabled GPIO pins.

Functionality

Sequential Logic

- **Buffered Input**: On each clock cycle, if reset (`rst`) is not active, `inputBuffered` is updated with the current `gpio_input` values.
- **Interrupt Request Logic**: On each clock cycle, if reset is not active, `gpio_irq` is set based on the logical OR of `pinIRQ`, reflecting any interrupt request from enabled GPIO pins.

Combinational Logic

- **Request Output**: The `requestOutput` signal is determined by combining request signals from all internal registers.
- **Peripheral Data Read Multiplexing**: `peripheralBus_dataRead` outputs the corresponding data from one of the internal registers based on which register has an active request.

Instantiations

- **DeviceSelect Instance**:
  - **Type**: `DeviceSelect`
  - **Purpose**: Selects the device based on the parameter `ID` to generate `localAddress` and `deviceEnable`.
  - **Connections**:
    - `.peripheralEnable(peripheralEnable)`
    - `.peripheralBus_address(peripheralBus_address)`
    - `.localAddress(localAddress)`
    - `.deviceEnable(deviceEnable)`
- **OutputRegister Instance (OE Register)**:
  - **Type**: `OutputRegister`
  - **Purpose**: Manages the output enable (`gpio_oe`) control register.
  - **Connections**:
    - Inputs/Outputs tied to peripheral bus signaling and register logic.
- **OutputRegister Instance (Output Register)**:
  - **Type**: `OutputRegister`
  - **Purpose**: Manages the GPIO outputs (`gpio_output`).
  - **Connections**:
    - Similar bus and logic connectivity as the OE register.
- **DataRegister Instance (Input Register)**:
  - **Type**: `DataRegister`
  - **Purpose**: Provides inputs buffering and register functionality.
  - **Connections**:
    - Includes connections for bus interactions and readout of `inputBuffered`.
- **OutputRegister Instance (IRQ Enable Register)**:

  - **Type**: `OutputRegister`
  - **Purpose**: Controls interrupt enable masking for each GPIO pin.
  - **Connections**:
    * Peripheral connections for reading, writing, and enabling IRQs.

2. Inter-Module Connections

The `GPIODevice` is essentially a composite unit integrating several key peripheral features:

- **Hierarchical Control**: A `DeviceSelect` module instance streamlines device-level addressing and selection in shared bus systems, driving the `deviceEnable` signal which conditions subsequent registers.
- **Peripheral Registers**: `OutputRegister` and `DataRegister` instantiations act as core functional blocks handling register-to-bus interactions, parameterized to reflect different hardware aspects (e.g., GPIO data, output enables, interrupts).
- **Flow of Data and Control**: Input GPIO signals are buffered, processed through IRQ logic, while peripheral bus interactions directly influence outputs and register states.

This modular architecture facilitates embedded systems design where peripheral interfacing is critical, balancing flexibility with structured signal handling through well-defined interfaces and parameterization.