ID_EXE Stage Module

The `ID_EXEstage` module is a **pipeline register** (or pipeline latch) placed between the **Instruction Decode (ID)** stage and the **Execute (EXE)** stage of the CPU pipeline.

In a pipelined processor, each stage processes one part of an instruction at a time during a clock cycle, and pipeline registers store the outputs of one stage to pass them **cleanly and synchronously** to the next stage on the next clock.

What Does This Module Do?

- **Buffers all important signals produced in the ID stage**, like:
    - The current instruction's program counter (`pc_id`)
    - Decoded immediate value (`imm_id`)
    - Register read data (`rdata1_id`, `rdata2_id`)
    - Destination register number (`rd_id`)
    - Control signals (`alusrc_id`, `regwrite_id`, etc.)
    - ALU operation code (`aluop_id`)
- On every **positive clock edge** (rising clock edge), it updates the outputs (`*_exe`) with the inputs (`*_id`), forwarding instruction data and control signals to the EXE stage.
- On **reset**, all outputs are cleared to zero (safe defaults), preventing invalid data from propagating in the pipeline during reset.

Signal	Direction	Width	Description
Inputs from ID:			
clk	input	1-bit	Clock, synchronizes data transfer
rst	input	1-bit	Reset signal to clear pipeline registers
pc_id	input	32-bit	PC value for the current instruction from ID
imm_id	input	32-bit	Immediate value decoded in ID
rdata1_id	input	32-bit	Register file read data 1
rdata2_id	input	32-bit	Register file read data 2

rd_id	input	5-bit	Destination register address
alusrc_id	input	1-bit	ALU source select signal (immediate or reg)
regwrite_id	input	1-bit	Register write enable
memwrite_id	input	1-bit	Memory write enable
memread_id	input	1-bit	Memory read enable
branch_id	input	1-bit	Branch control signal
aluop_id	input	3-bit	ALU operation code
Outputs to EXE:			
pc_exe	output	32-bit	Forwarded PC to EXE
imm_exe	output	32-bit	Forwarded immediate value to EXE
rdata1_exe	output	32-bit	Forwarded register data 1 to EXE
rdata2_exe	output	32-bit	Forwarded register data 2 to EXE
rd_exe	output	5-bit	Forwarded destination register address
alusrc_exe	output	1-bit	Forwarded ALU source select
regwrite_exe	output	1-bit	Forwarded register write control
memwrite_exe	output	1-bit	Forwarded memory write control
memread_exe	output	1-bit	Forwarded memory read control
branch_exe	output	1-bit	Forwarded branch control
aluop_exe	output	3-bit	Forwarded ALU opcode

Why Is This Important?

- **Synchronizes data/control signals between stages:**
  The processor operates in stages, and these registers act like "checkpoints" between them. They make sure data from one stage is available and stable for processing by the next stage.
- **Maintains pipeline timing:**
  Since all pipeline registers latch data on the clock edge, the whole pipeline moves instructions smoothly step-by-step each clock cycle.

- **Ensures state reset:**
  Clearing the registers on reset prevents the pipeline from working on garbage data when starting up or during exception recovery.

Overall Working Flow

1. **During the Decode (ID) stage:**
   - Instruction fetched from memory is decoded.
   - Register operands are read.
   - Immediate values and control signals are generated.
   - All this information is presented at the outputs of the ID stage (`*_id` signals).

2. **At the next clock edge:**
   - This module captures all those ID signals and stores them.
   - The stored values are output as `*_exe` signals, making them inputs for the Execute (EXE) stage.

3. **In the Execute (EXE) stage:**
   - The ALU, branch comparator, and other functional units use these signals to perform the actual computation or decide on a branch.

Summary

- **`ID_EXEstage` is a pipeline latch between ID and EXE stages.**
- Stores data and control signals from the decode stage.
- Updates stored values every clock cycle.
- Resets registers to default values on reset.
- Enables smooth, clocked flow of instructions and control signals through the pipeline stages.