ECAP5-DPROC

RISC-V processor

Architecture Document

Revision: 1.0.0-draft1

Issue Date: July 17, 2023

Contents

1 Introduction

1.1 Purpose

This documents aims at defining the requirements for ECAP5-DPROC as well as describing its architecture. Both user and product requirements will be covered.

1.2 Intended Audience and Use

This document targets hardware engineers who shall implement ECAP5-DPROC by refering to the described architecture. It is also intended for system engineers working on the integration of ECAP5-DPROC in ECAP5. Finally, this document shall be used as a technical reference by software engineers configuring ECAP5-DPROC through hardware-software interfaces.

1.3 Product Scope

ECAP5-DPROC is an implementation of the RISC-V instruction set architecture targetting *Educational Computer Architecture Platform 5* (ECAP5). It will provide the main means of software execution in ECAP5.

1.4 Conventions and Definitions

hardware-configurable

software-configurable

The bit indexing shall be described somewhere.

Byte size as well.

Inputs missing from timing diagrams are considered low or undefined.

Italic names are timing diagram parameters.

Pipeline bubble.

1.5 References

Date	Version	Title
December 13, 2019	20191213	The RISC-V Instruction Set Manual Volume I: User-Level ISA
March 22, 2019	0.13.2	RISC-V External Debug Support
September 7, 2002	B.3	WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Core

2 Overall Description

2.1 User needs

ECAP5 is the primary user for ECAP5-DPROC. ECAP5-DPROC could however be used as a standalone RISC-V processor. The following requirements define the user needs.

ID	U_INSTRUCTION_SET_01
Description	ECAP5-DPROC shall implement the RV32I instruction set.

In order to improve the usability of ECAP5-DPROC, it shall have a *von Neumann* architecture as it only requires one memory interface.

ID	U_MEMORY_INTERFACE_01
Description	ECAP5-DPROC shall access both instructions and data through a unique memory interface.

ID	U_MEMORY_INTERFACE_02
Description	ECAP5-DPROC's unique memory interface shall be compliant with the Wishbone specification.

ID	U_MEMORY_INTERFACE_03
Description	ECAP5-DPROC's unique memory interface shall be designed such that memory protocols can be interchanged at compile time.

ID	U_RESET_01
Description	ECAP5-DPROC shall provide a signal which shall hold ECAP5-DPROC in a reset state while asserted.

The polarity of the reset signal mentionned in `U_RESET_01` is not specified by the user.

ID	U_BOOT_ADDRESS_01
Description	The address at which ECAP5-DPROC jumps after the reset signal is deasserted shall be hardware-configurable.

The address mentionned in `U_BOOT_ADDRESS_01` can be either configured through hardware signals or can be selected at compile time.

ID	U_HARDWARE_INTERRUPT_01
Description	ECAP5-DPROC shall provide an signal which shall interrupt ECAP5-DPROC's execution flow while asserted.

ID	U_HARDWARE_INTERRUPT_02
Description	ECAP5-DPROC shall jump to a software-configurable address when it is interrupted.

The memory address at which ECAP5-DPROC shall jump to when interrupted is not specified by the user.

ID	U_DEBUG_01
Description	ECAP5-DPROC shall be compliant with the RISC-V External Debug Support specification.

There is no performance goal required by ECAP5 for ECAP5-DPROC as ECAP5 is an educational platform.

2.2 Assumptions and Dependencies

Describe what the assumptions for the product are : Targeting the ecp5 family, based around opensource toolchains.

3 Requirements

3.1 External Interface Requirements

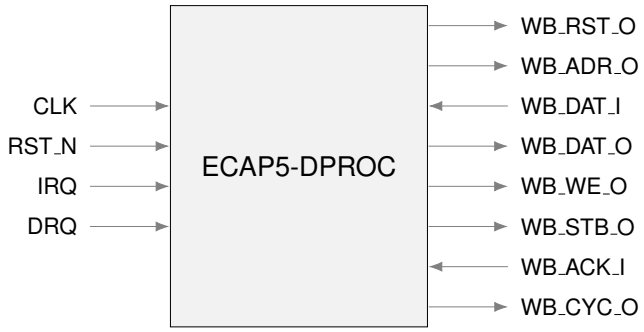


Figure 1: Schematic view of the external interface of ECAP5-DPROC

NAME	TYPE	WIDTH	DESCRIPTION
CLK	I	1	Clock input.
RST_N	I	1	Hardware reset. Active low.
IRQ	I	1	External interrupt request.
DRQ	I	1	Debug request.

Table 1: ECAP5-DPROC control signals

NAME	TYPE	WIDTH	DESCRIPTION
WB_RST_O	O	1	TBD
WB_ADR_O	O	32	TBD
WB_DAT_I	I	32	TBD
WB_DAT_O	O	32	TBD
WB_WE_O	O	1	TBD
WB_STB_O	O		TBD
WB_ACK_I	I	1	TBD
WB_CYC_O	O		TBD

Table 2: ECAP5-DPROC memory interface signals

ID	I_CLK_01
Description	All inputs and outputs of ECAP5-DPROC shall belong to CLK's clock domain.
Refers to	

ID	I_RESET_01
Description	The RST_N signal shall hold ECAP5-DPROC in a reset state while asserted.
Refers to	U_RESET_01

ID	I_RESET_02
Description	RST_N polarity shall be active low.
Refers to	

ID	I_IRQ_01
Description	ECAP5-DPROC shall jump to a software-configurable address when input IRQ is asserted.
Refers to	U_HARDWARE_INTERRUPT_01, U_HARDWARE_INTERRUPT_02

ID	I_DIRQ_01
Description	TBD
Refers to	

ID	I_MEMORY_INTERFACE_01
Description	Signals from table 2 shall be compliant with the Wishbone specification.
Refers to	U_MEMORY_INTERFACE_02

Behavioral specification for symbols in table 2 is outlined in the functional requirements

section, subsection 3.2.5.

3.2 Functional Requirements

3.2.1 Register file

ID	F_REGISTERS_01
Description	ECAP5-DPROC shall implement 31 user-accessible general purpose registers ranging from `x0` to `x31`.
Refers to	U_INSTRUCTION_SET_01

ID	F_REGISTERS_02
Description	Register `x0` shall be hardwired to the constant zero.
Refers to	U_INSTRUCTION_SET_01

ID	F_REGISTERS_03
Description	ECAP5-DPROC shall implement a `pc` user-accessible register storing the address of the current instruction.
Refers to	U_INSTRUCTION_SET_01

3.2.2 Instruction decoding

Figure 2 outlines the different instruction encodings for the RV32I instruction set. The `opcode` parameter is a unique identifier for each instruction. The instruction encoding is infered from the opcode as there can only be one encoding per opcode.

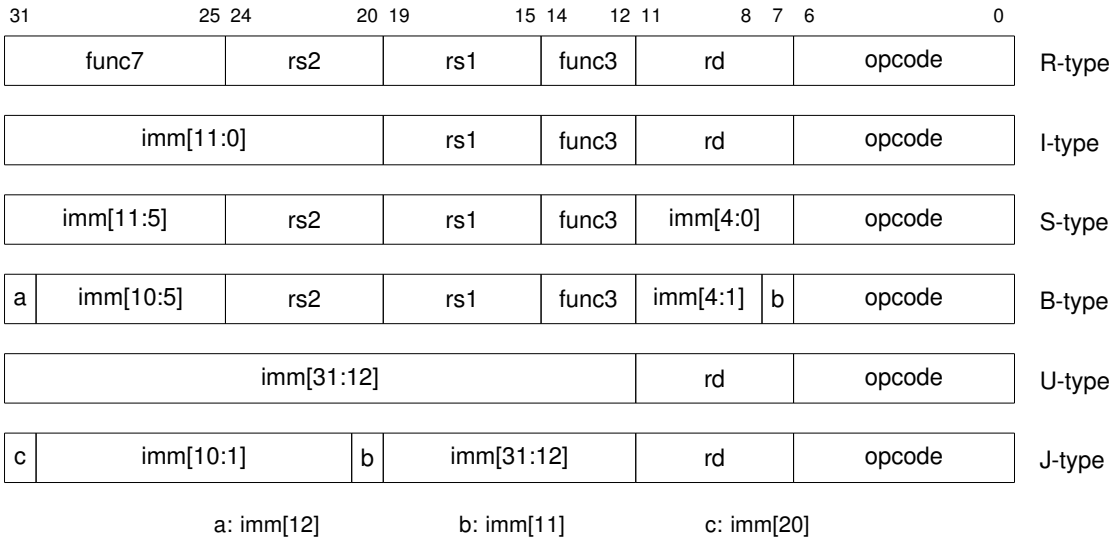


Figure 2: Instruction encodings of the RV32I instruction set

Immediate encoding

Only one immediate value can be encoded in one instruction. The value can be reconstructed from fragments of the following format : imm[x] representing the x^{th} bit or imm[x:y] representing bits from the x^{th} to the y^{th} both included.

ID	F_INSTR_IMMEDIATE_01
Description	Immediate values shall be sign-extended.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_IMMEDIATE_02
Description	The value of an instruction immediate shall be the concatenation of immediate fragments from the instruction encoding.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_IMMEDIATE_03
Description	Missing immediate fragments shall be replaced by zeros.
Refers to	U_INSTRUCTION_SET_01

RV32I is called a Load/Store ISA, meaning that instructions inputs and outputs are passed through registers or through an instruction immediate. There are specific instructions for loading and storing data into memory.

Instruction inputs

ID	F_INSTR_FIRST_INPUT_01
Description	Instructions encoded using the R-type, I-type, S-type and B-type shall take as their first input the value stored in the register designated by the `rs1` parameter.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_FIRST_INPUT_02
Description	Instructions encoded using the U-type and J-type shall take as their first input the immediate value encoded in the instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_SECOND_INPUT_01
Description	Instructions encoded using the R-type, S-type and B-type shall take as their second input the value stored in the register designated by the `rs2` parameter.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_SECOND_INPUT_02
Description	Instructions encoded using the I-type shall take as its second input the immediate value encoded in the instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_THIRD_INPUT_01
Description	Instructions encoded using the S-type and B-type shall take as their third input the immediate value encoded in the instruction.
Refers to	U_INSTRUCTION_SET_01

Instruction outputs

ID	F_INSTR_OUTPUT_01
Description	Instructions encoded using the R-type, I-type, U-type and J-type shall store their result in the register designated by the `rd` parameter.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_OUTPUT_02
Description	Instructions encoded using the S-type and B-type do not produce any result.
Refers to	U_INSTRUCTION_SET_01

Instruction variants

ID	F_INSTR_VARIANT_01
Description	Instructions encoded using the R-type, I-type, S-type and B-type shall use the `func3` parameter as a behavior variant selector.
Refers to	U_INSTRUCTION_SET_01

ID	F_INSTR_VARIANT_02
Description	Instructions encoded using the R-type shall use the `func7` parameter as a secondary behavior variant selector.
Refers to	U_INSTRUCTION_SET_01

Opcodes

Table 3 outlines the different opcodes values of the RV32I instruction set. Cells marked as *noimp* are for opcodes that are not implemented in ECAP5-DPROC.

opcode[1:0]	11							
opcode[4:2] / opcode[6:5]	000	001	010	011	100	101	110	111
00	LOAD	*noimp*	*noimp*	MISC-MEM	OP-IMM	AUIPC	*noimp*	*noimp*
01	STORE	*noimp*	*noimp*	*noimp*	OP	LUI	*noimp*	*noimp*
10	*noimp*	*noimp*	*noimp*	*noimp*	*noimp*	*noimp*	*noimp*	*noimp*
11	BRANCH	JALR	*noimp*	JAL	SYSTEM	*noimp*	*noimp*	*noimp*

Table 3: Opcode values for the RV32I instruction set.

ID	F_OPCODE_ENCODING_01
Description	Instructions which use the LUI opcode shall be decoded as an U-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_02
Description	Instructions which use the AUIPC opcode shall be decoded as an U-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_03
Description	Instructions which use the JAL opcode shall be decoded as a J-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_04
Description	Instructions which use the JALR opcode shall be decoded as an I-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_05
Description	Instructions which use the BRANCH opcode shall be decoded as a B-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_06
Description	Instructions which use the LOAD opcode shall be decoded as an I-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_07
Description	Instructions which use the STORE opcode shall be decoded as a S-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_08
Description	Instructions which use the OP-IMM opcode shall be decoded as an I-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_09
Description	Instructions which use the OP opcode shall be decoded as a R-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_10
Description	Instructions which use the MISC-MEM opcode shall be decoded as an I-type instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_OPCODE_ENCODING_11
Description	Instructions which use the SYSTEM opcode shall be decoded as an I-type instruction.
Refers to	U_INSTRUCTION_SET_01

3.2.3 Instructions behaviors

LUI

ID	F_LUI_01
Description	The LUI behavior shall be applied when the opcode is LUI.
Refers to	U_INSTRUCTION_SET_01

ID	F_ADDI_02
Description	The output of LUI shall be the value of its first input.
Rationale	The LUI instruction shall load the 20 upper bits of the instruction immediate into the destination register and fill the remaining bits with zeros. This is the default behavior for instruction immediates as stated in F_INSTR_IMMEDIATE_02 and F_INSTR_IMMEDIATE_03.
Refers to	U_INSTRUCTION_SET_01

AUIPC

ID	F_AUIPC_01
Description	The AUIPC behavior shall be applied when the opcode is AUIPC.
Refers to	U_INSTRUCTION_SET_01

ID	F_AUIPC_02
Description	The output of AUIPC shall be the sum of its first input and the address of the AUIPC instruction.
Refers to	U_INSTRUCTION_SET_01

JAL

ID	F_JAL_01
Description	The JAL behavior shall be applied when the opcode is JAL.
Refers to	U_INSTRUCTION_SET_01

ID	F_JAL_02
Description	The `pc` register shall be updated with the sum of the address of the JAL instruction with the first instruction input.
Refers to	U_INSTRUCTION_SET_01

ID	F_JAL_03
Description	The output of JAL shall be the address of the JAL instruction incremented by 4.
Rationale	The JAL instruction shall output the address to the following instruction for it to be used as a *return address* in the case of a function call.
Refers to	U_INSTRUCTION_SET_01

JALR

ID	F_JALR_01
Description	The JALR behavior shall be applied when the opcode is JALR and func3 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_JALR_02
Description	The `pc` register shall be updated with the sum of the first and second inputs of the JALR instruction.
Refers to	U_INSTRUCTION_SET_01

ID	F_JALR_03
Description	The output of JALR shall be the address of the JALR instruction incremented by 4.
Rationale	The JALR instruction shall output the address to the following instruction for it to be used as a *return address* in the case of a function call.
Refers to	U_INSTRUCTION_SET_01

BEQ

ID	F_BEQ_01
Description	The BEQ behavior shall be applied when the opcode is BRANCH and func3 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_BEQ_02
Description	When the first and second instruction inputs are equal, the `pc` register shall be updated with the signed sum of the address of the BEQ instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

BNE

ID	F_BNE_01
Description	The BNE behavior shall be applied when the opcode is BRANCH and func3 is 0x1.
Refers to	U_INSTRUCTION_SET_01

ID	F_BNE_02
Description	When the first and second inputs are not equal, the pc register shall be updated with the signed sum of the address of the BNE instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

BLT

ID	F_BLT_01
Description	The BLT behavior shall be applied when the opcode is BRANCH and func3 is 0x4.
Refers to	U_INSTRUCTION_SET_01

ID	F_BLT_02
Description	When the first input is lower than the second input using a signed comparison, the pc register shall be updated with the signed sum of the address of the BLT instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

BGE

ID	F_BGE_01
Description	The BGE behavior shall be applied when the opcode is BRANCH and func3 is 0x5.
Refers to	U_INSTRUCTION_SET_01

ID	F_BGE_02
Description	When the first input is greater or equal to the second input using a signed comparison, the pc register shall be updated with the signed sum of the address of the BGE instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

BLTU

ID	F_BLTU_01
Description	The BLTU behavior shall be applied when the opcode is BRANCH and func3 is 0x6.
Refers to	U_INSTRUCTION_SET_01

ID	F_BLTU_02
Description	When the first input is lower than the second input using an unsigned comparison, the `pc` register shall be updated with the signed sum of the address of the BLTU instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

BGEU

ID	F_BGEU_01
Description	The BGEU behavior shall be applied when the opcode is BRANCH and func3 is 0x7.
Refers to	U_INSTRUCTION_SET_01

ID	F_BGEU_02
Description	When the first input is greater or equal to the second input using an unsigned comparison, the `pc` register shall be updated with the signed sum of the address of the BGEU instruction with the third input.
Refers to	U_INSTRUCTION_SET_01

LB

ID	F_LB_01
Description	The LB behavior shall be applied when the opcode is LOAD and func3 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_LB_02
Description	The output of LB shall be the 8-bit value stored in memory at the address determined by the signed sum of its first and second inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_LB_03
Description	The remaining bits of the loaded value shall be filled with the value of its 7^{th} bit.
Refers to	U_INSTRUCTION_SET_01

LH

ID	F_LH_01
Description	The LH behavior shall be applied when the opcode is LOAD and func3 is 0x1.
Refers to	U_INSTRUCTION_SET_01

ID	F_LH_02
Description	The output of LH shall be the 16-bit value stored in memory at the address determined by the signed sum of its first and second inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_LH_03
Description	The remaining bits of the loaded value shall be filled with the value of its 15^{th} bit.
Refers to	U_INSTRUCTION_SET_01

LW

ID	F_LW_01
Description	The LW behavior shall be applied when the opcode is LOAD and func3 is 0x2.
Refers to	U_INSTRUCTION_SET_01

ID	F_LW_02
Description	The output of LW shall be the 32-bit value stored in memory at the address determined by the signed sum of its first and second inputs.
Refers to	U_INSTRUCTION_SET_01

LBU

ID	F_LBU_01
Description	The LBU behavior shall be applied when the opcode is LOAD and func3 is 0x4.
Refers to	U_INSTRUCTION_SET_01

ID	F_LBU_02
Description	The output of LBU shall be the 8-bit value stored in memory at the address determined by the signed sum of its first and second inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_LBU_03
Description	The remaining bits of the loaded value shall be filled with zeros.
Refers to	U_INSTRUCTION_SET_01

LHU

ID	F_LHU_01
Description	The LHU behavior shall be applied when the opcode is LOAD and func3 is 0x5.
Refers to	U_INSTRUCTION_SET_01

ID	F_LHU_02
Description	The output of LHU shall be the 16-bit value stored in memory at the address determined by the signed sum of its first and second inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_LHU_04
Description	The remaining bits of the loaded value shall be filled with zeros.
Refers to	U_INSTRUCTION_SET_01

SB

ID	F_SB_01
Description	The SB behavior shall be applied when the opcode is STORE and func3 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_SB_02
Description	The lowest byte of the second input of SB shall be stored in memory at the address determined by the signed sum of its first and third inputs.
Refers to	U_INSTRUCTION_SET_01

SH

ID	F_SH_01
Description	The SH behavior shall be applied when the opcode is STORE and func3 is 0x1.
Refers to	U_INSTRUCTION_SET_01

ID	F_SH_02
Description	The two lowest bytes of the second input of SB shall be stored in memory at the address determined by the signed sum of its first and third inputs.
Refers to	U_INSTRUCTION_SET_01

SW

ID	F_SW_01
Description	The SW behavior shall be applied when the opcode is STORE and func3 is 0x2.
Refers to	U_INSTRUCTION_SET_01

ID	F_SH_02
Description	The value of the second input of SB shall be stored in memory at the address determined by the signed sum of its first and third inputs.
Refers to	U_INSTRUCTION_SET_01

ADDI

ID	F_ADDI_01
Description	The ADDI behavior shall be applied when the opcode is OP-IMM and when func3 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_ADDI_02
Description	The output of ADDI shall be the signed integer sum of its two inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_ADDI_03
Description	The output of ADDI shall be truncated to 32-bits.
Refers to	U_INSTRUCTION_SET_01

SLTI

ID	F_SLTI_01
Description	The SLTI behavior shall be applied when the opcode is OP-IMM and when func3 is 0x2.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLTI_02
Description	The output of SLTI shall be 1 when the signed value of its first input is lower that the signed value of its second input. It shall be 0 otherwise.
Refers to	U_INSTRUCTION_SET_01

SLTIU

ID	F_SLTIU_01
Description	The SLTIU behavior shall be applied when the opcode is OP-IMM and when func3 is 0x3.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLTIU_02
Description	The output of SLTI shall be 1 when the unsigned value of its first input is lower that the unsigned value of its second input. It shall be 0 otherwise.
Refers to	U_INSTRUCTION_SET_01

XORI

ID	F_XORI_01
Description	The XORI behavior shall be applied when the opcode is OP-IMM and when func3 is 0x4.
Refers to	U_INSTRUCTION_SET_01

ID	F_XORI_02
Description	The output of XORI shall be the result of a bitwise xor between its two inputs.
Refers to	U_INSTRUCTION_SET_01

ORI

ID	F_ORI_01
Description	The ORI behavior shall be applied when the opcode is OP-IMM and when func3 is 0x6.
Refers to	U_INSTRUCTION_SET_01

ID	F_ORI_02
Description	The output of ORI shall be the result of a bitwise or between its two inputs.
Refers to	U_INSTRUCTION_SET_01

ANDI

ID	F_ANDI_01
Description	The ANDI behavior shall be applied when the opcode is OP-IMM and when func3 is 0x7.
Refers to	U_INSTRUCTION_SET_01

ID	F_ANDI_02
Description	The output of ANDI shall be the result of a bitwise and between its two inputs.
Refers to	U_INSTRUCTION_SET_01

SLLI

ID	F_SLLI_01
Description	The SLLI behavior shall be applied when the opcode is OP-IMM and func3 is 0x1.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLLI_02
Description	The output of SLLI shall be its first input shifted left by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLLI_03
Description	Zeros shall be inserted in the lower bits when shifting.
Refers to	U_INSTRUCTION_SET_01

SRLI

ID	F_SRLI_01
Description	The SRLI behavior shall be applied when the opcode is OP-IMM, func3 is 0x5 and the 30^{th} bit of its second input is 0.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRLI_02
Description	The output of SRLI shall be its first input shifted right by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRLI_03
Description	Zeros shall be inserted in the upper bits when shifting.
Refers to	U_INSTRUCTION_SET_01

SRAI

ID	F_SRAI_01
Description	The SRAI behavior shall be applied when the opcode is OP-IMM, func3 is 0x5 and the 30^{th} bit of its second input is 1.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRAI_02
Description	The output of SRAI shall be its first input shifted right by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRAI_03
Description	The most significant bit of the first input shall be inserted in the upper bits when shifting.
Refers to	U_INSTRUCTION_SET_01

ADD

ID	F_ADD_01
Description	The ADD behavior shall be applied when the opcode is OP, func3 is 0x0 and func7 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_ADD_02
Description	The output of ADD shall be the signed integer sum of its two inputs.
Refers to	U_INSTRUCTION_SET_01

ID	F_ADD_03
Description	The output of ADD shall be truncated to 32-bits.
Refers to	U_INSTRUCTION_SET_01

SUB

ID	F_SUB_01
Description	The SUB behavior shall be applied when the opcode is OP, func3 is 0x0 and func7 is 0x20.
Refers to	U_INSTRUCTION_SET_01

ID	F_SUB_02
Description	The output of SUB shall be the signed integer difference of its first input minus its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SUB_03
Description	The output of SUB shall be truncated to 32-bits.
Refers to	U_INSTRUCTION_SET_01

SLL

ID	F_SLL_01
Description	The SLL behavior shall be applied when the opcode is OP and func3 is 0x1.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLL_02
Description	The output of SLL shall be its first input shifted left by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLL_03
Description	Zeros shall be inserted in the lower bits when shifting.
Refers to	U_INSTRUCTION_SET_01

SLT

ID	F_SLT_01
Description	The SLT behavior shall be applied when the opcode is OP and func3 is 0x2.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLT_02
Description	The output of SLT shall be 1 when the signed value of its first input is lower that the signed value of its second input. It shall be 0 otherwise.
Refers to	U_INSTRUCTION_SET_01

SLTU

ID	F_SLTU_01
Description	The SLTU behavior shall be applied when the opcode is OP and func3 is 0x3.
Refers to	U_INSTRUCTION_SET_01

ID	F_SLTU_02
Description	The output of SLTU shall be 1 when the unsigned value of its first input is lower that the unsigned value of its second input. It shall be 0 otherwise.
Refers to	U_INSTRUCTION_SET_01

XOR

ID	F_XOR_01
Description	The XOR behavior shall be applied when the opcode is OP and func3 is 0x4.
Refers to	U_INSTRUCTION_SET_01

ID	F_XOR_02
Description	The output of XOR shall be the result of a bitwise xor between its two inputs.
Refers to	U_INSTRUCTION_SET_01

SRL

ID	F_SRL_01
Description	The SRL behavior shall be applied when the opcode is OP, func3 is 0x5 and func7 is 0x0.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRL_02
Description	The output of SRL shall be its first input shifted right by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRL_03
Description	Zeros shall be inserted in the upper bits when shifting.
Refers to	U_INSTRUCTION_SET_01

SRA

ID	F_SRA_01
Description	The SRA behavior shall be applied when the opcode is OP, func3 is 0x5 and func7 is 0x20.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRA_02
Description	The output of SRA shall be its first input shifted right by the amount specified by the first 5 bits of its second input.
Refers to	U_INSTRUCTION_SET_01

ID	F_SRA_03
Description	The most significant bit of the first input shall be inserted in the upper bits when shifting.
Refers to	U_INSTRUCTION_SET_01

OR

ID	F_OR_01
Description	The OR behavior shall be applied when the opcode is OP and func3 is 0x6.
Refers to	U_INSTRUCTION_SET_01

ID	F_OR_02
Description	The output of OR shall be the result of a bitwise or between its two inputs.
Refers to	U_INSTRUCTION_SET_01

AND

ID	F_AND_01
Description	The AND behavior shall be applied when the opcode is OP and func3 is 0x7.
Refers to	U_INSTRUCTION_SET_01

ID	F_AND_02
Description	The output of AND shall be the result of a bitwise and between its two inputs.
Refers to	U_INSTRUCTION_SET_01

FENCE TBD

ECALL TBD

EBREAK TBD

3.2.4 Exceptions

ID	F_INSTR_ADDR_MISALIGNED_01
Description	An Instruction Address Misaligned exception shall be raised when the target address of a taken branch or an unconditional jump if not four-byte aligned.
Refers to	U_INSTRUCTION_SET_01

ID	F_MISALIGNED_MEMORY_ACCESS_01
Description	A Misaligned Memory Access exception shall be raised when the target address of a load/store instruction is not aligned on the referenced type size.
Refers to	U_INSTRUCTION_SET_01

3.2.5 Memory interface

ID	F_ENDIANNESS_01
Description	Memory accesses shall use the little-endian format.
Refers to	

Outline requirements to be compliant with the Wishbone specification.

3.2.6 Debugging

3.3 Nonfunctional Requirements

ID	N_FORMAL_PROOF_01
Description	Each part of ECAP5-DPROC shall be formally proven when possible, otherwise thouroughly tested
Refers to	

These can be : performance, safety, security, usability, scalability.

4 Configuration

4.1 Constants

Table 4: Constants used for implementing the RISC-V ISA

NAME	TYPE	WIDTH	DESCRIPTION	VALUE

4.2 Instanciation parameters

ECAP5-DPROC can be parameterized at build-time through instanciation parameters. Default constant values are defined for such parameters.

Table 5: Instanciation parameters of ECAP5-DPROC

NAME	TYPE	WIDTH	DESCRIPTION	DEFAULT VALUE

5 Architecture overview

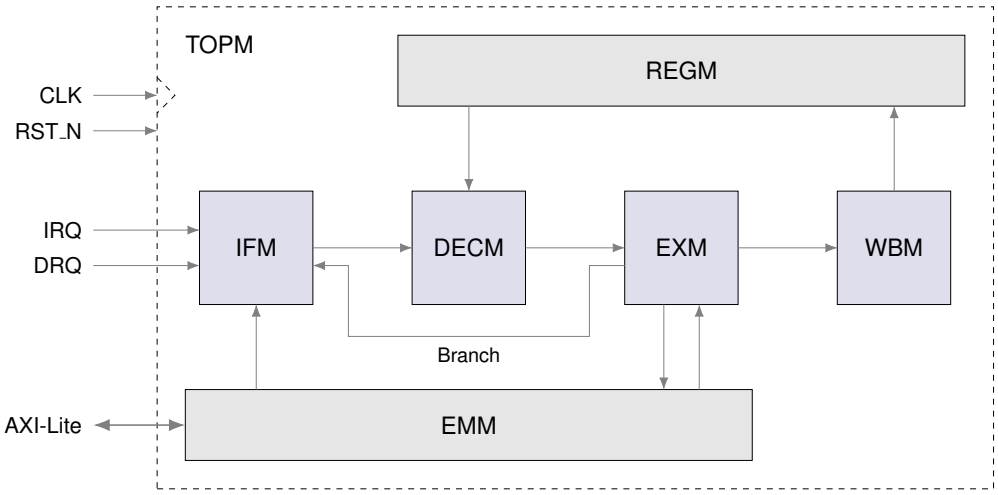


Figure 3: Schematic view of the architecture of ECAP5-DPROC

5.1 Pipeline stages

ECAP5-DPROC is built around a pipelined architecture with the following stages :

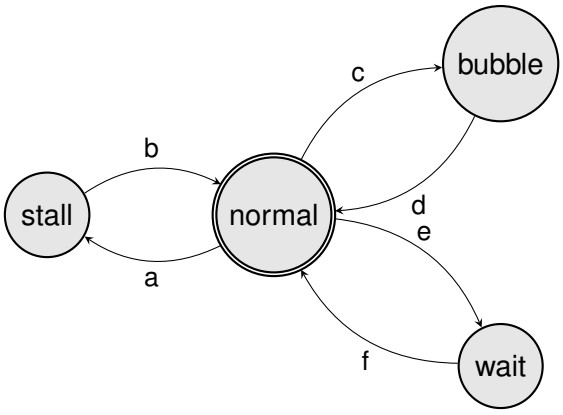
- The instruction fetch stage loads the next instruction from memory.
- The decode stage handles the instruction decoding to provide the next stage with the different instruction input values including reading from internal registers.
- The execute stage implements all arithmetic and logic operations. This includes load and store operations to the memory.
- The write-back stage which handles storing instructions outputs to internal registers.

Considering the load-store architecture of the RISC-V instruction set, the choice has been made, for revision 1.0.0, to include the memory stage of the typical 5-stage pipeline within the execute stage. This will decrease the latency while keeping a similar throughput, as any memory access will produce a pipeline stall as of revision 1.0.0.

TBD Handshaking between stages.

5.2 Pipeline stall management

TBD



a : stage stalls, b : stage unstalls, c : input valid = 0, d : input valid = 1, e : output ready = 0, f : output valid = 1,

Figure 4: State diagram of the operating modes of pipeline stages

TBC (Description of the behavior of the modes. Stall =¿ output valid = 0 and input ready = 0 Bubble =¿ normal operation with nop instruction Wait =¿ input ready = 0)

Pipeline stages located at the start and end of the pipeline do not implement the bubble and wait modes respectively.

5.2.1 Wait state stalls

Pipeline stages are expected to perform their operation within one clock cycle. In cases where they are unable to achieve this goal, a wait state stall of the pipeline shall happen where preceding stages shall stop while following stages shall finish processing their instruction.

The handshake mechanism between pipeline stages is used to produce wait state stalls. The stalling stage deasserts its input ready signal leading to preceding stages waiting for completion. The stalling stage deasserts its output valid signal leading to following stages taking a bubble as their input.

The figure 5 is a diagram of the stall behavior on a 5-stage pipeline. By stalling the 3rd stage, this example provides a representative visualisation of all the stalling cases of a 4-stage pipeline.

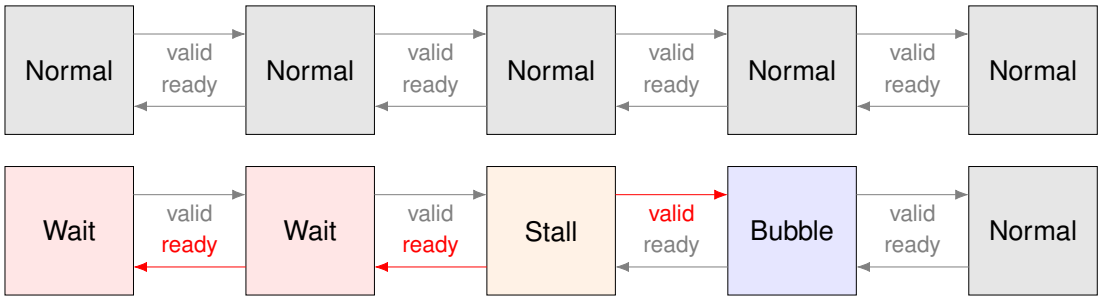


Figure 5: Diagram of a wait state stall behavior on a 6-stage pipeline

TBD(Description of the timing diagram)

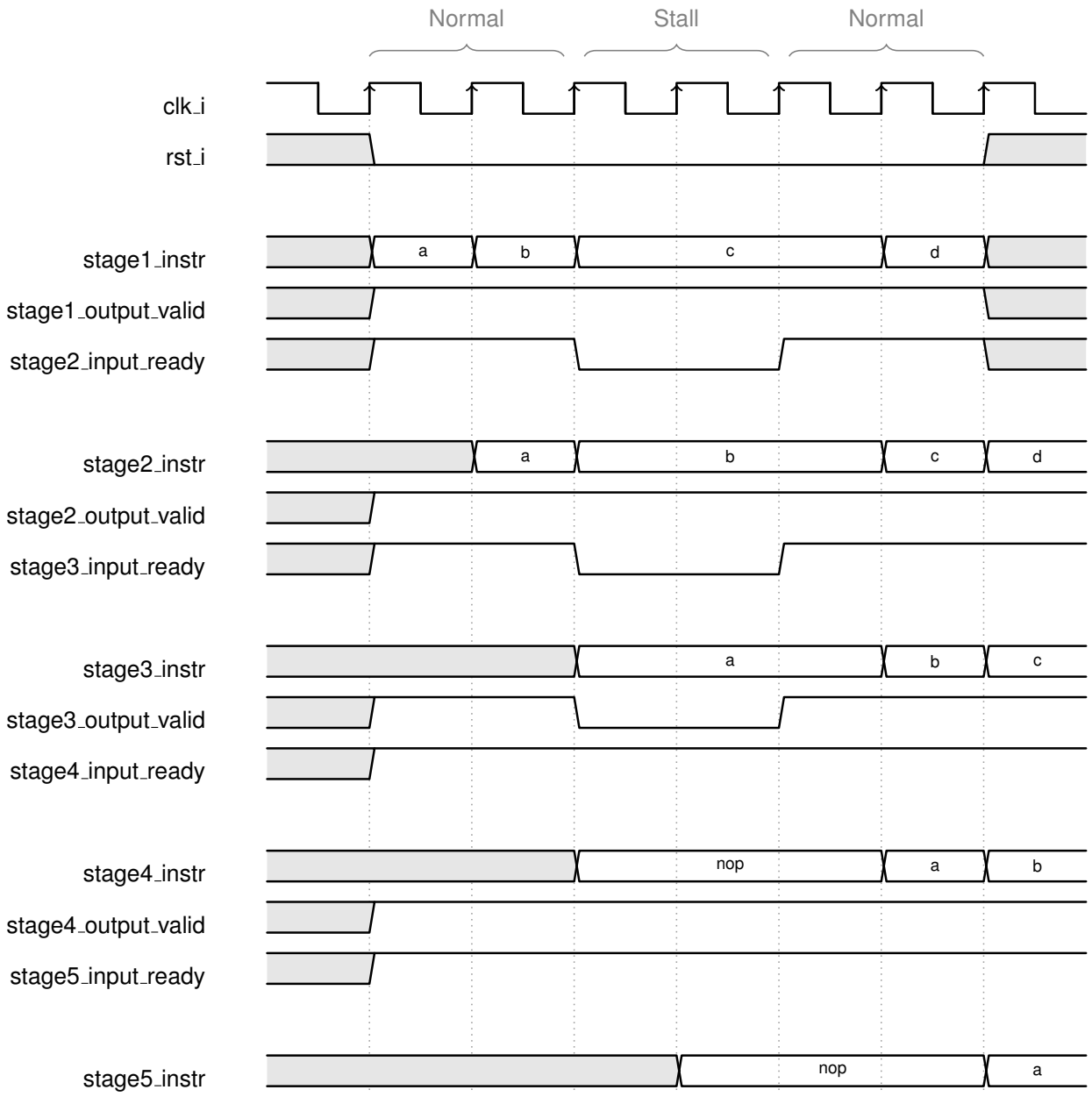


Figure 6: Timing diagram of a wait state stall resolution behavior on a 6-stage pipeline

5.2.2 Instruction dependency stalls

TBD

5.3 Functional partitioning

The design is split into the following functional modules :

- The **Top Module** (TOPM) which integrates all other modules.
- The **External Memory Module** (EMM), in charge of accessing memory and peripherals.
- The **Instruction Fetch Module** (IFM), in charge of implementing the instruction fetch stage.
- The **Decode Module** (DECM), in charge of implementing the decode stage.
- The **Register Module** (REGM), implementing the internal registers.
- The **Execute Module** (EXM), in charge of implementing the execute stage.
- The **Write-Back Module** (WBM), in charge of implementing the write-back stage.

6 Register Module

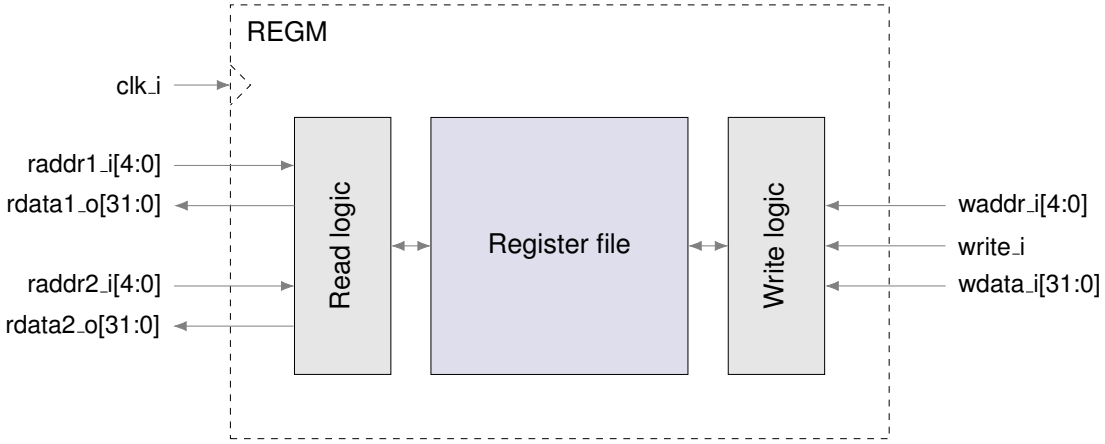


Figure 7: Schematic view of the Register Module

6.1 Interface

The register module implements the 32 internal registers of ECAP5-DPROC. It has two reading port and one writing port. The signals are described in table 6.

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
FIRST READING PORT			
raddr1_i	I	5	Register selector.
rdata1_o	O	32	Selected register value.
SECOND READING PORT			
raddr2_i	I	5	Register selector.
rdata2_o	O	32	Selected register value.
WRITING PORT			
waddr_i	I	5	Register selector.
write_i	I	1	Asserted to indicate a write.
wdata_i	I	32	Data to be written.

Table 6: Register Module interface signals

6.2 Behavior

6.2.1 Read behavior

When reading, `rdata1_i` and `rdata2_i` output, on the rising edge of `clk_i`, the value of the register respectively selected by `raddr1_i` and `raddr2_i`.

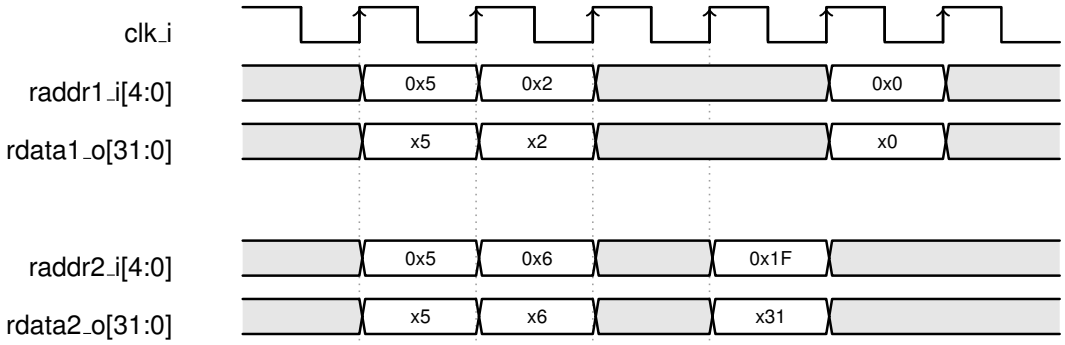


Figure 8: Timing diagram of the read behavior of the register module

6.2.2 Write behavior

A register write happens on the rising edge of `clk_i` when `write_i` is asserted, writing the value `wdata_i` in the register selected by `waddr_i`.

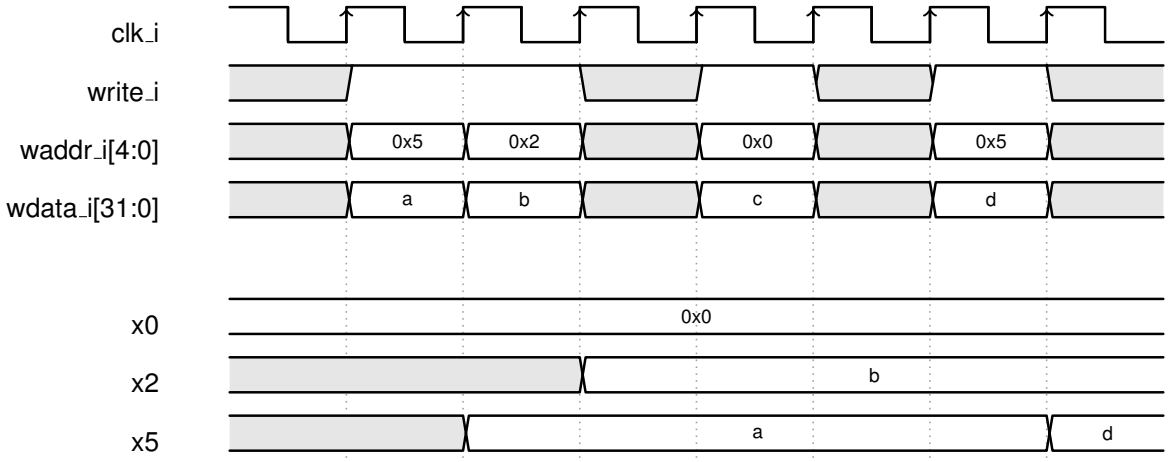


Figure 9: Timing diagram of the write behavior of the register module

6.2.3 Read-before-write behavior

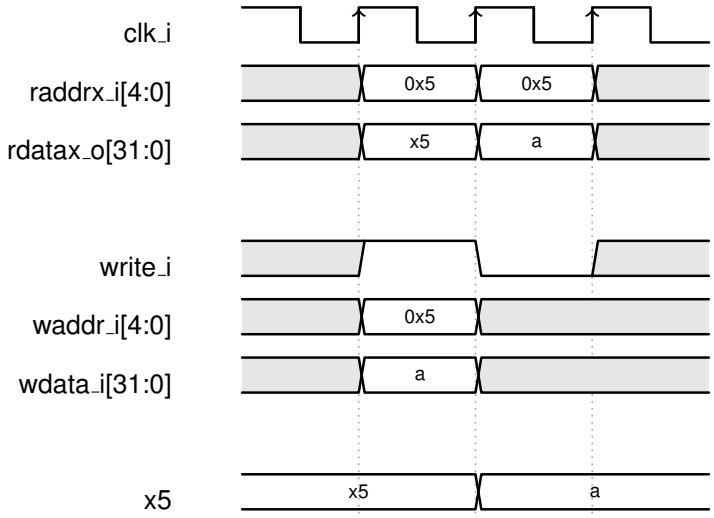


Figure 10: Timing diagram of the read-before-write behavior of the register module

6.3 Design

7 External Memory Module

7.1 Interface

7.2 Behavior

7.3 Design

8 Instruction Fetch Module

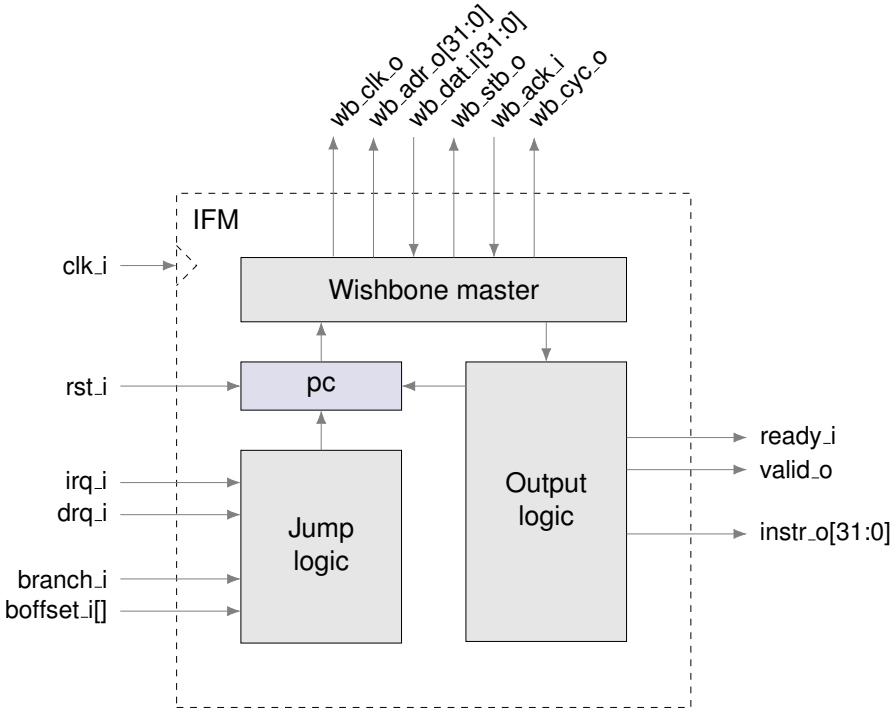


Figure 11: Schematic view of the Instruction Fetch Module

8.1 Interface

The instruction fetch module handles fetching from memory the instructions to be executing. The signals are described in table 7.

Table 7: Instruction Fetch Module interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
rst_i	I	1	Reset input.
JUMP LOGIC			
irq_i	I	1	External interrupt request.
drq_i	I	1	External debug request.
branch_i	I	1	Branch request.
boffset_i	I	20	Branch offset from the pc of the instruction in the execute stage (pc - 8).
WISHBONE MASTER			

wb_clk_o	O	1	Wishbone clock output. This is hardwired to clk_i.
wb_adr_o	O	32	Wishbone read address.
wb_dat_i	I	32	Wishbone read data.
wb_stb_o	O	1	
wb_ack_i	I	1	Acknowledge. Indicates a normal termination of a bus cycle.
OUTPUT LOGIC			
ready_i	I	1	
valid_o	O	1	Asserted when the output is ready to be sent.
instr_o	O	32	Instruction to be executed.

8.2 Behavior

This module doesn't contain any prefetch mechanism as there is no performance requirement for version 1.0.0. This will lead to a performance bottleneck due to the number of cycles needed for fetching instructions from memory.

8.2.1 Normal behavior

TBD

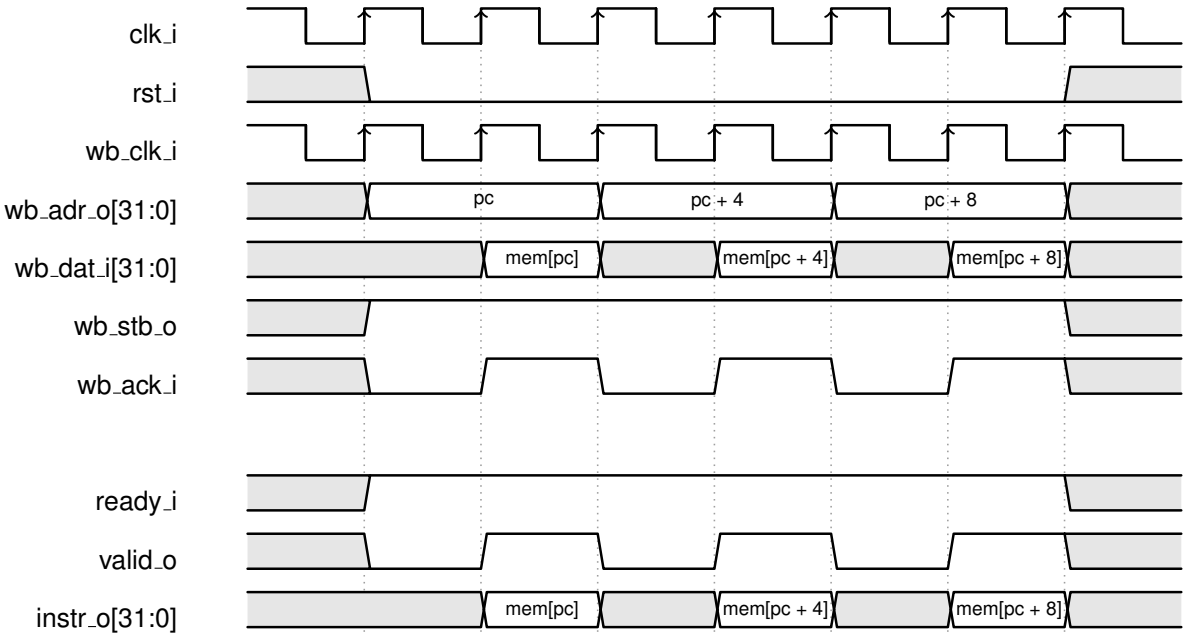


Figure 12: Timing diagram of the normal behavior of the instruction fetch module

8.2.2 Reset behavior

TBD

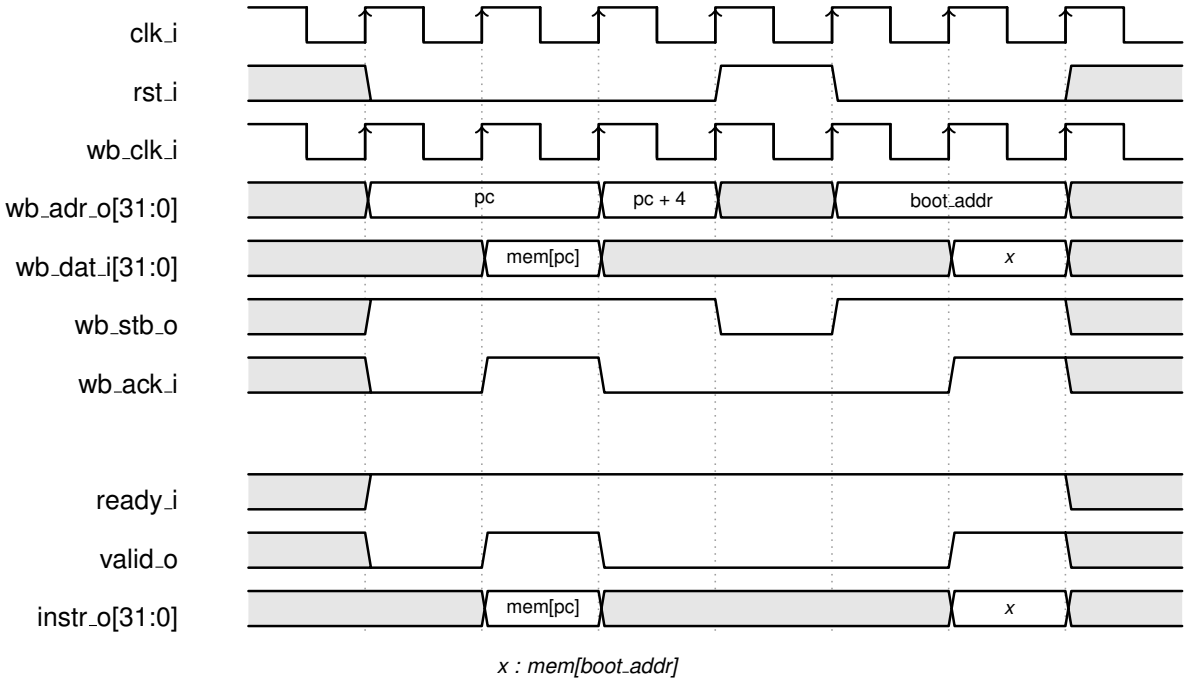


Figure 13: Timing diagram of the reset behavior of the instruction fetch module

8.2.3 Resource busy behavior

TBD

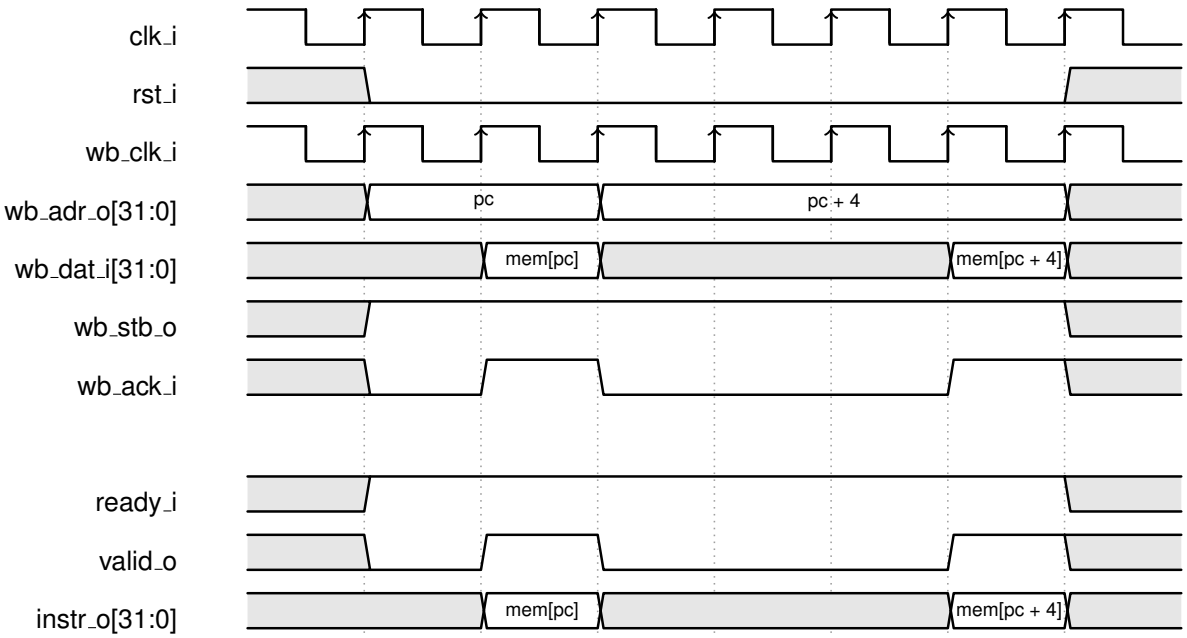


Figure 14: Timing diagram of the memory resource busy behavior of the instruction fetch module

8.2.4 Jump behavior

TBD

Figure 15: Timing diagram of the jump behavior of the instruction fetch module for interrupt events

Figure 16: Timing diagram of the jump behavior of the instruction fetch module for branch events

8.2.5 Pipeline stall behavior

Pipeline stall behaviors are described in section 5.2.

8.3 Design

`pc` stores the value of the next instruction to be loaded from memory. It is connected to the wishbone master which performs the memory read. The read data is transfered to the output logic, in charge of handling the pipeline's handshaking protocol. The value of `pc` can be either incremented by the output logic, reset by `rst_i` or loaded with a specific value through the jump logic.

8.3.1 Jump logic

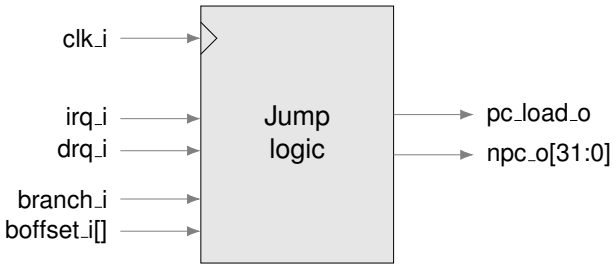


Figure 17: Schematic view of the interface of the Jump Logic

The jump logic loads a value into the `pc` register based on inputs. Its interface is described in table 8.

Table 8: Jump logic interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
irq_i	I	1	External interrupt request.
drq_i	I	1	External debug request.
branch_i	I	1	Branch request.
boffset_i	I	TBC	Branch offset. TBC
pc_load_o	O	1	Asserted when the pc register shall be updated.
npc_o	O	32	Value used to update the pc register.

Figures 18 and 19 describe the behavior of the jump logic. The values to be loaded in memory are hardcoded and set at compile time.

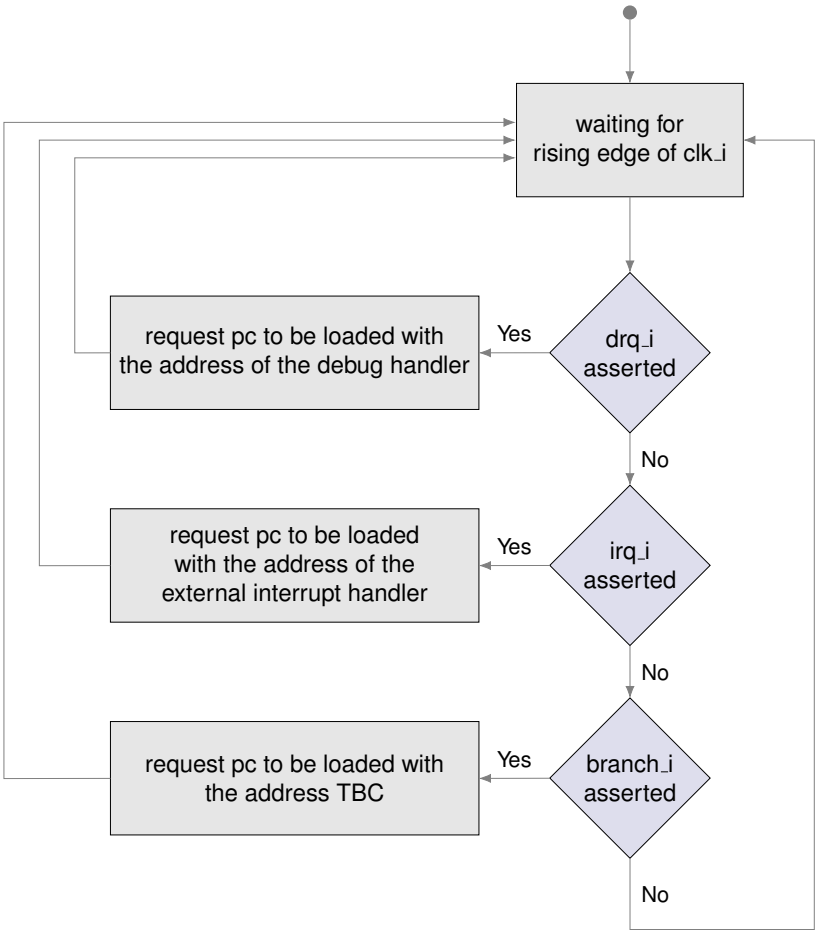


Figure 18: Activity diagram of the jump logic behavior

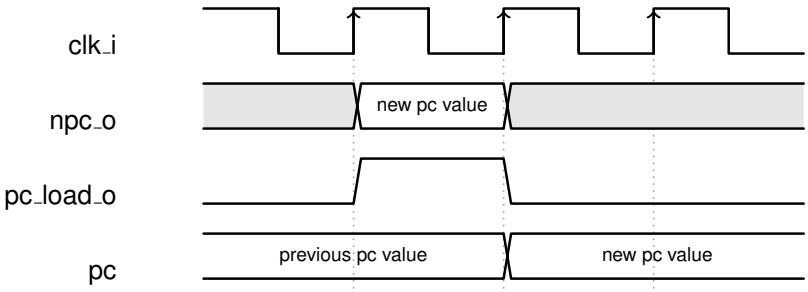


Figure 19: Timing diagram for the output port of the jump logic.

8.3.2 PC register

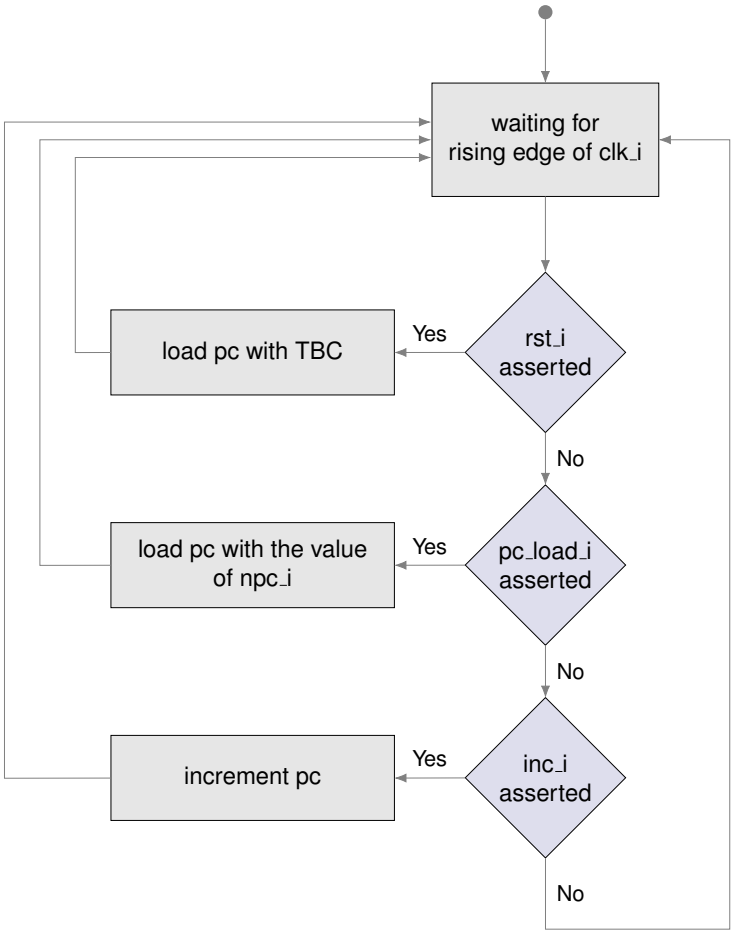


Figure 20: Activity diagram of the pc register

8.3.3 Wishbone master

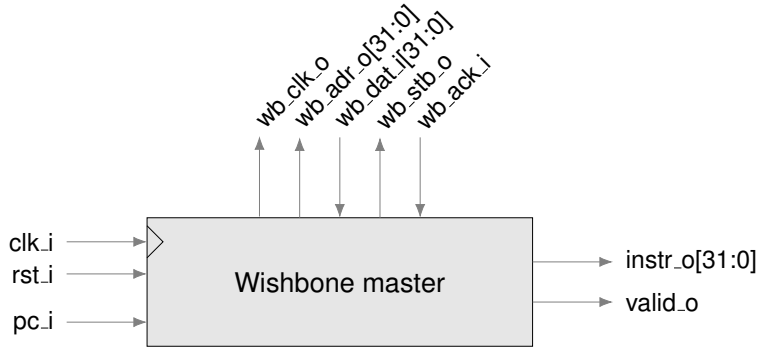


Figure 21: Schematic view of the interface of the Wishbone Master

The wishbone master fetches from memory the instruction to be executed. Its interface is described in table 9. The memory port of the wishbone master is described in figure 22.

Table 9: Instruction Fetch Module interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
rst_i	I	1	Reset input.
WISHBONE MASTER			
wb_clk_o	O	1	Wishbone clock output. This is hardwired to clk_i.
wb_adr_o	O	32	Wishbone read address.
wb_dat_i	I	32	Wishbone read data.
wb_stb_o	O	1	Strobe output indicates a valid data transfer cycle.
wb_ack_i	I	1	Acknowledge. Indicates a normal termination of a bus cycle.
OUTPUT			
instr_o	O	32	Instruction to be executed.
valid_o	O	1	Asserted when the instruction output is valid.

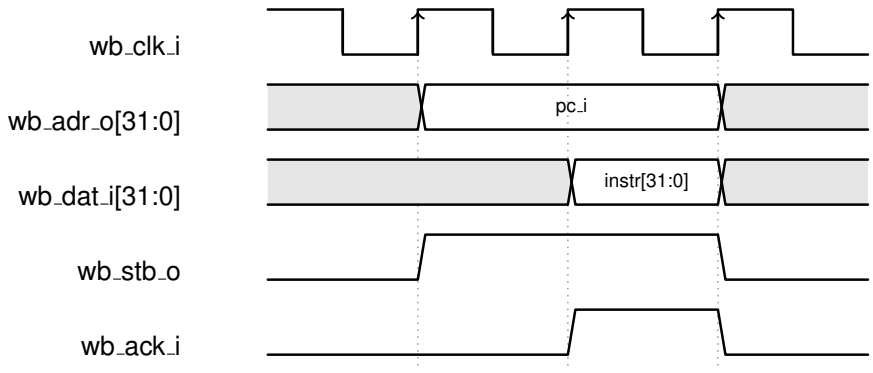


Figure 22: Timing diagram for the memory port of IFM's wishbone master

The output port of the wishbone master has the following properties:

$$valid\_o = (wb\_stb\_o \ \& \ wb\_ack\_i)$$

$$instr\_o[31:0] = wb\_dat\_i$$

8.3.4 Output logic

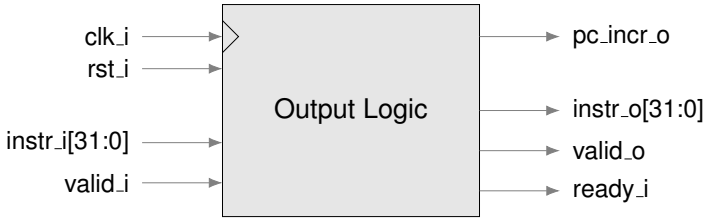


Figure 23: Schematic view of the interface of the Output Logic

9 Decode Module

Figure 24: Schematic view of the Decode Module

9.1 Interface

The decode module implements TBD. The signals are described in table 10.

Table 10: Decode Module interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
rst_i	I	1	Reset input.
INPUT LOGIC			
instr_i	I	32	
ready_o	O	1	
valid_i	I	1	
REGISTER ACCESS			
raddr1_o	O	5	
rdata1_i	I	32	
raddr2_o	O	5	
rdata2_i	I	32	
OUTPUT LOGIC			
opcode_o	O	7	
instr_input1_o	O	32	
instr_input2_o	O	32	
instr_input3_o	O	32	
instr_output1_o	O	32	
instr_output2_o	O	32	
instr_func3_o	O	TBC	
instr_func7_o	O	TBC	

9.2 Behavior

TBD

9.2.1 Pipeline stall behavior

Pipeline stall behaviors are described in section 5.2.

9.3 Design

TBD

10 Execute Module

Figure 25: Schematic view of the Execute Module

10.1 Interface

The execute module implements TBD. The signals are described in table 11.

Table 11: Execute Module interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
rst_i	I	1	Reset input.
INPUT LOGIC			
instr_i	I	TBD	
input1_i	I	32	
input2_i	I	32	
input3_i	I	32	
WISHBONE MASTER			
wb_clk_o	O	1	Wishbone clock output. This is hardwired to clk_i.
wb_adr_o	O	32	Wishbone read address.
wb_dat_i	I	32	Wishbone read data.
wb_dat_o	O	32	Wishbone write data.
wb_we_o	O	1	Wishbone write-enable.
wb_sel_o	O	4	
wb_stb_o	O	1	
wb_ack_i	I	1	Acknowledge. Indicates a normal termination of a bus cycle.
OUTPUT LOGIC			
reg_write_o	O	1	
reg_write_addr_o	O	5	
reg_write_data_o	O	32	
branch_o	O	1	
boffset_o	O	20	

10.2 Behavior

TBD

10.2.1 LUI behavior

TBD

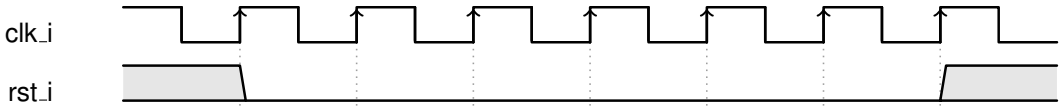


Figure 26: Timing diagram of the LUI behavior of the execute module

10.2.2 AUIPC behavior

TBD

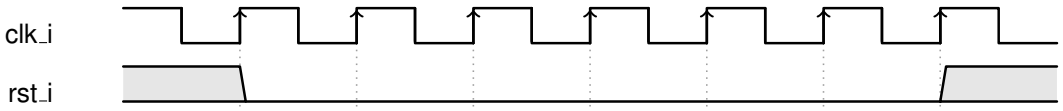


Figure 27: Timing diagram of the AUIPC behavior of the execute module

10.2.3 Jump behavior

JAL

TBD

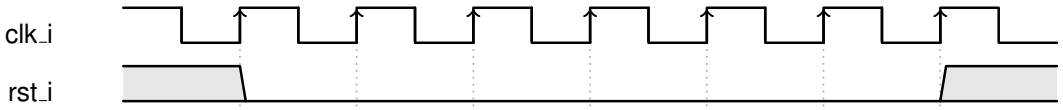


Figure 28: Timing diagram of the jump behavior of the execute module for the JAL instruction

JALR

TBD

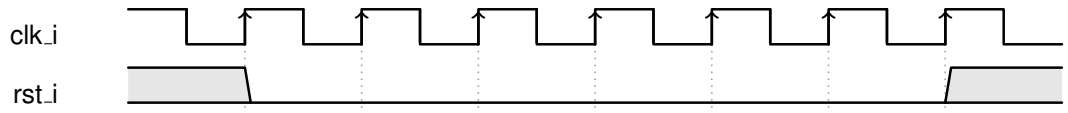


Figure 29: Timing diagram of the jump behavior of the execute module for the JALR instruction

10.2.4 Branch behavior

The branch behavior compares the first two inputs depending on the instruction variant. The BEQ and BNE variants check whether the two inputs are equal or not equal respectively. The BLT and BLTU variants check whether the first input is lower than the second input using a signed and unsigned comparison respectively. The BGE and BGEU variants check whether the first input is greater or equal to the second input using a signed and unsigned comparison respectively.

A branch request is emitted in case of successful comparison, providing the address offset given in the third input.

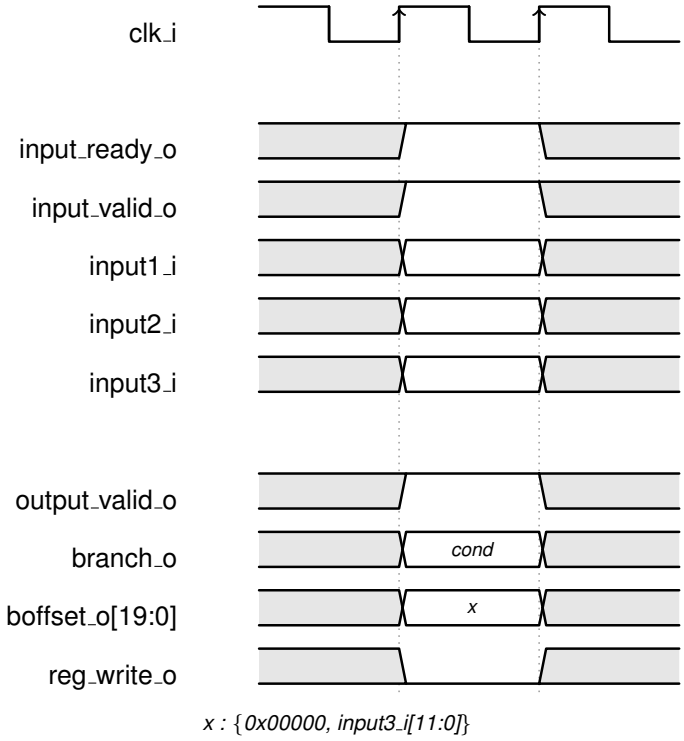


Figure 30: Timing diagram of the branch behavior of the execute module

Table 12: Description of the parameters of figure 30.

VARIANTS	PARAMETERS	DESCRIPTION
BEQ	cond = if(input1_i = input2_i) 1 else 0	1 when the first input is equal to the second input. 0 otherwise.
BNE	cond = if(input1_i = input2_i) 0 else 1	1 when the first input is not equal to the second input. 0 otherwise.
BLT	cond = if(input1_i $<$ input2_i) 1 else 0	1 when the first input is lower than the second input using a signed comparison. 0 otherwise.
BLTU	cond = if(unsigned(input1_i) $<$ unsigned(input2_i)) 1 else 0	1 when the first input is lower than the second input using an unsigned comparison. 0 otherwise.
BGE	cond = if(input1_i $\geq$ input2_i) 1 else 0	1 when the first input is greater or equal to the second input using a signed comparison. 0 otherwise.
BGEU	cond = if(unsigned(input1_i) $\geq$ unsigned(input2_i)) 1 else 0	1 when the first input is greater or equal to the second input using an unsigned comparison. 0 otherwise.

10.2.5 Load behavior

The load behavior fetches an 8/16/32-bit word from memory at the address given in the first instruction input for the LB/LBU, LH/LHU and LW instructions respectively. The value is zero-extended to 32-bits for the U variants while it is signed extended to 32-bits otherwise.

A register write request is then emitted to store the value.

This behavior induces a pipeline stall due to the memory access time.

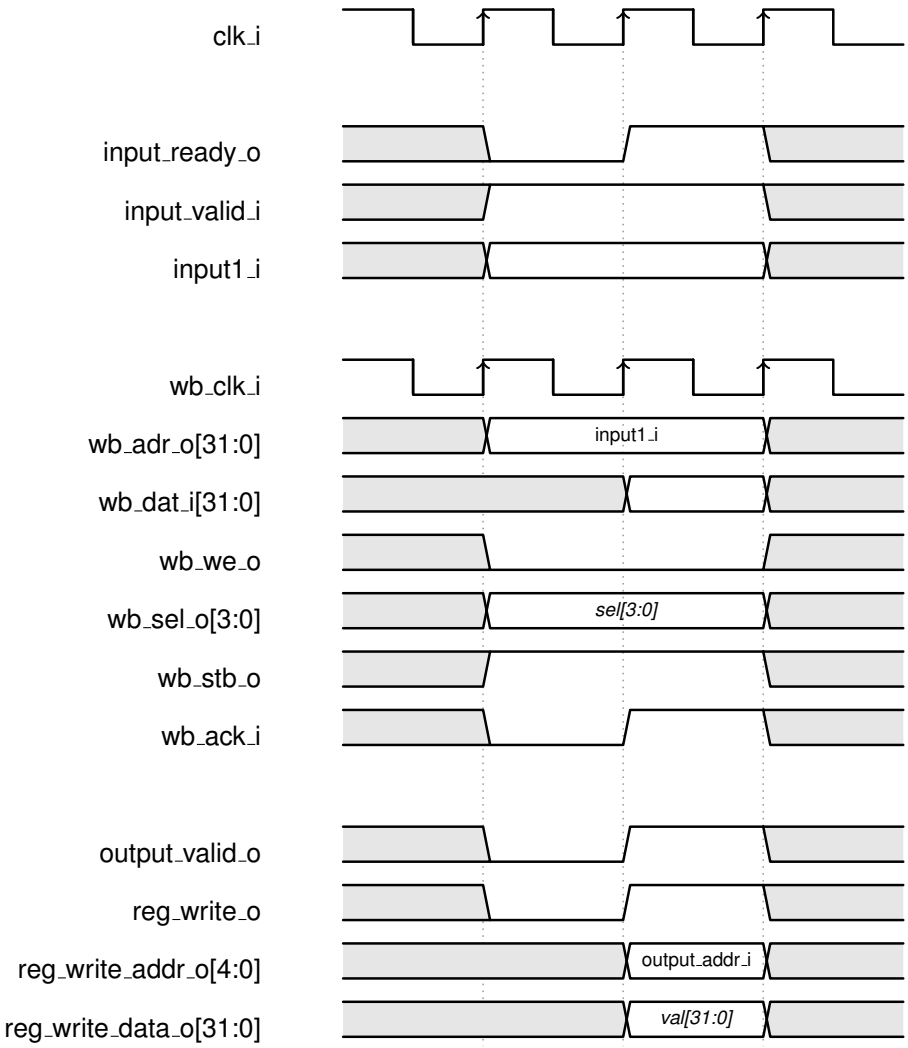


Figure 31: Timing diagram of the load behavior of the execute module

Table 13: Description of the parameters of figure 31.

VARIANTS	PARAMETERS	DESCRIPTION
LB	val[31:0] = {24{wb_data_i[7]}, wb_data_i[7:0]}	The sign-extended 8-bit data retrieved from memory.
	sel[3:0] = 0b0001	Only the lowest significant byte is selected.
LH	val[31:0] = {16{wb_data_i[15]}, wb_data_i[15:0]}	The sign-extended 16-bit data retrieved from memory.
	sel[3:0] = 0b0011	Only the two lowest significant bytes are selected.
LW	val[31:0] = wb_data_i[31:0]	The 32-bit data retrieved from memory.

	sel[3:0] = 0b1111	All bytes are selected.
LBU	val[31:0] = {0x000000, wb_data_i[7:0]}	The zero-extended 8-bit data retrieved from memory.
	sel[3:0] = 0b0001	Only the lowest significant byte is selected.
LHU	val[31:0] = {0x0000, wb_data_i[15:0]}	The zero-extended 16-bit data retrieved from memory.
	sel[3:0] = 0b0011	Only the two lowest significant bytes are selected.

10.2.6 Store behavior

The store behavior writes a 8/16/32-bit word to memory at the address given in the first instruction input for the SB, SH and SW instructions respectively.

Although there is no need for the pipeline to wait for the data to be written, this behavior induces a pipeline stall in revision 1.0.0 as a write request takes at least two cycles to complete while the memory interface shall be ready to receive new requests in the next pipeline cycle.

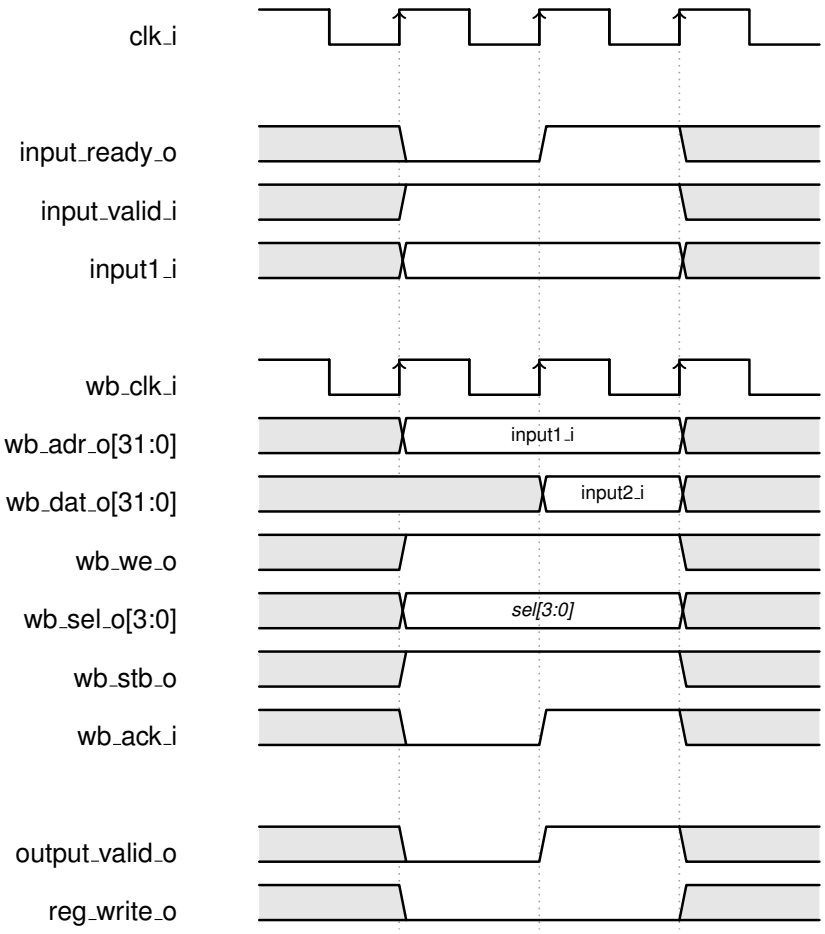


Figure 32: Timing diagram of the store behavior of the execute module

Table 14: Description of the parameters of figure 32.

VARIANTS	PARAMETERS	DESCRIPTION
SB	sel[3:0] = 0b0001	Only the lowest significant byte is selected.
SH	sel[3:0] = 0b0011	Only the two lowest significant bytes are selected.
SW	sel[3:0] = 0b1111	All the bytes are selected.

10.2.7 Arithmetic and logic behavior

The arithmetic and logic behavior applies to OP and OP-IMM opcodes. As DECM handles the processing of instruction inputs, both OP and OP-IMM instructions have the same associated behavior in EXM.

The arithmetic and logic behavior computes the following integer operations : signed sum, signed substraction, bitwise exclusive-or, bitwise or, bitwise and, signed comparison, unsigned comparison, left logical shift, right logical shift and right arithmetic shift.

A register write request is then emitted to store the value.

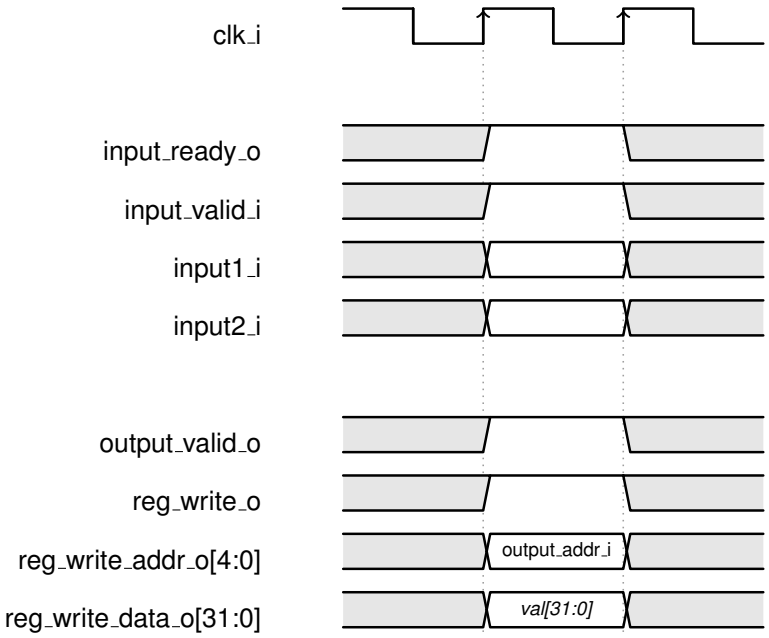


Figure 33: Timing diagram of the arithmetic and logic behavior of the execute module

Table 15: Description of the parameters of figure 33.

VARIANTS	PARAMETERS	DESCRIPTION
ADD/ADDI	val[31:0] = input1_i + input2_i	The signed sum of the first and second input.
SUB	val[31:0] = input1_i - input2_i	The signed difference of the first input minus the second input.

XOR/XORI	val[31:0] = input1_i $\oplus$ input2_i	The bitwise exclusive-or of the first and second inputs.
OR/ORI	val[31:0] = input1_i $\vee$ input2_i	The bitwise or of the first and second inputs.
AND/ANDI	val[31:0] = input1_i $\wedge$ input2_i	The bitwise and of the first and second inputs.
SLT/SLTI	val[31:0] = if(input1_i $<$ input2_i) 1 else 0	1 when the first input is lower than the second input using a signed comparison. 0 otherwise.
SLTU/SLTIU	val[31:0] = if(unsigned(input1_i) $<$ unsigned(input2_i)) 1 else 0	1 when the first input is lower than the second input using an unsigned comparison. 0 otherwise.
SLL/SLLI	val[31:0] = $\{$input1_i[31 - input2_i[4:0] : 0], input2_i[4:0]$\{0\}\}$	The first input is shifted left by the amount specified by the first 5 bits of the second input. The right bits are filled with zeros.
SRL/SRLI	val[31:0] = $\{$input2_i[4:0]$\{0\}$, input1_i[31 : input2_i[4:0]]$\}$	The first input is shifted right by the amount specified by the first 5 bits of the second input. The left bits are filled with zeros.
SRA/SRAI	val[31:0] = $\{$input2_i[4:0]$\{$input1_i[31]$\}$, input1_i[31 : input2_i[4:0]]$\}$	The first input is shifted right by the amount specified by the first 5 bits of the second input. The left bits are filled with the sign bit of the first input.

10.2.8 Pipeline stall behavior

Pipeline stall behaviors are described in section 5.2.

10.3 Design

TBD

11 Write-Back Module

Figure 34: Schematic view of the Write-Back Module

11.1 Interface

The write-back module implements TBD. The signals are described in table 16.

Table 16: Write-back Module interface signals

NAME	TYPE	WIDTH	DESCRIPTION
clk_i	I	1	Clock input.
rst_i	I	1	Reset input.
INPUT LOGIC			
write_i	I	1	Asserted when the instruction shall output a value to a register.
instr_output_reg_i	I	5	Address of the register to be written to.
instr_output_i	I	32	Value to be written to the register.
REGISTER ACCESS			
waddr_o	O	5	
wdata_o	O	32	
write_o	0	1	

11.2 Behavior

11.2.1 Instruction with output behavior

TBD

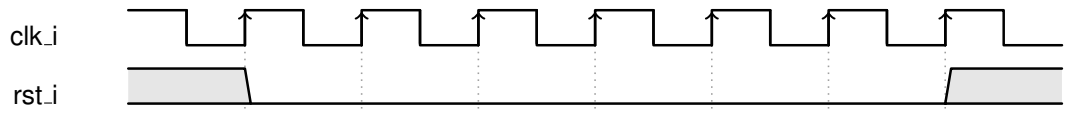


Figure 35: Timing diagram of the instruction with output behavior of the write-back module

11.2.2 Instruction without output behavior

TBD

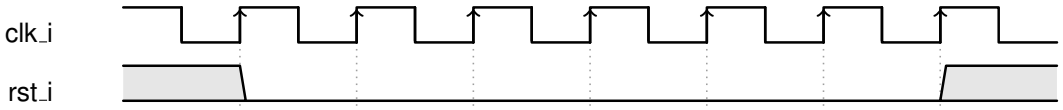


Figure 36: Timing diagram of the instruction without output behavior of the write-back module

11.2.3 Pipeline stall behavior

Pipeline stall behaviors are described in section 5.2.

11.3 Design

TBD

12 Debug