Deliverable d1: Processor Architecture block level design

Sistemi di Elaborazione a.a. 2018/2019

Group M
Revision 1

Produced by	Date of Approval	Approved by	Version
Fernando Manna, Giovanni Di Prisco, Ilaria Gigi	15 Dec. 2018	Francesco de Pertis, Fernando Manna	1.1

Deliverable d1:
Processor Architecture block level design by group M

Release Information

The following changes have been made to this document.

Table 1 Change History

Date	Revision	Authors	Change Details
08 December 2018	0	Fernando Manna, Giovanni Di Prisco, Francesco De Pertis	First complete draft
15 December 2018	1	Fernando Manna, Giovanni Di Prisco, Ilaria Gigi	Updated all contents and reordered the entire document. Restyled chapters and schemas with signals semantics.

Contents

Deliverable d1: Processor Architecture block level design

Preface ____ x
About this manual ____ xi
Using this manual ____ xi
Conventions ____ xii
References ____ xii

Chapter 1: Branch Prediction Unit Block ____ 1

1 Introduction ____ 2

2 Prediction bits ____ 4

3 BTB Entry fields ____ 5

4 Signals ____ 6

Chapter 2: Control Unit ____ 7

1 Introduction ____ 8

2 Instruction Decoder ____ 9

3 Supported instruction set ____ 10

4 Forward Decoder ____ 11

Chapter 3: Out-of-Order Block ____ 12

1 Introduction ____ 13

2 Issue Register ____ 14
2.1 Introduction ____ 14
2.2 Entry fields ____ 15
2.3 Signals ____ 15

3 RSR ____ 17
3.1 Introduction ____ 17

3.2 Entry fields 17
3.3 Signals 18
4 Shunt Unit 19
4.1 Introduction 19
4.2 Signals 19
5 Acceptance Unit 21
5.1 Introduction 21
5.2 Signals 22
6 Condition Verifier 23
6.1 Introduction 23
6.2 Signals 23
7 Issuing Unit 24
7.1 Introduction 24
7.2 Signals 25
8 Shifter 26
8.1 Introduction 26
8.2 Signals 26
9 Routing Unit 27
9.1 Introduction 27
9.2 Signals 28
10 Functional Units 29
10.1 Introduction 29
10.2 Signals 30
11 ROB 31
11.1 Introduction 31
11.2 Entry fields 32
11.3 Signals 32
Chapter 4: Forwarding Unit 33
1 Introduction 34
2 Entry fields 35
3 Signals 36
Chapter 5: Hazard Unit 38

1 Introduction 39

2 Behaviour and signals 40

Chapter 6: Resources 42

Resources allocation 43

Resources per leadership 43

Resources timesheet 43

List of Tables

Deliverable d1:

Processor Architecture block level design

Table 1 Change History ii
Table 2: BTB Entry fields 5
Table 3: Branch Recognizer signals 6
Table 4: BPU signals 6
Table 5: Aliasing Recognizer signals 6
Table 6: BTB signals 6
Table 7: Priority Updater signals 6
Table 8: Instruction Decoder signals 9
Table 9: Supported instruction set 10
Table 10: Forward Decoder signals 11
Table 11: Issue Register entry fields 15
Table 12: Issue Register signals 16
Table 13: RSR table entry fields 18
Table 14: RSR signals 18
Table 15: Shunt Unit signals 20
Table 16: Acceptance Unit signals 22
Table 17: Condition Verifier signals 23
Table 18: Issuing Unit signals 25
Table 19: Table signals 26
Table 20: Routing Unit signals 28
Table 21: Functional Units signals 30
Table 22: ROB entry fields 32
Table 23: ROB signals 32
Table 24: Forwarding Unit table entry fields 35
Table 25: Forwarding Unit table state update logic 36
Table 26: Forwarding Unit signals 37
Table 27: Exception Handler signals 41

Table 28: Wrong Prediction Handler signals _____41
Table 29: Full Issue Register and Miss Handler signals_____41

List of Figures

Deliverable d1:

Processor Architecture block level design

Figure 1: Branch Unit Block 2
Figure 2: Prediction bits update logics 4
Figure 3: Control Unit 8
Figure 4: Out-of-Order Block 13
Figure 5: Issue Register 14
Figure 6: RSR 17
Figure 7: Shunt Unit 19
Figure 8: Acceptance Unit 21
Figure 9: Condition Verifier 23
Figure 10: Issuing Unit 24
Figure 11: Shifter 26
Figure 12: Routing Unit 27
Figure 13: Functional Units 29
Figure 14: ROB 31
Figure 15: Forwarding Unit 34
Figure 16: Hazard Unit 39

List of Acronyms

Deliverable d1:

Processor Architecture block level design

ASID	Address Space Identifier
AU	integer Arithmetic Unit
BHT	Branch History Table
BPB	Branch Prediction Buffer
BPU	Branch Prediction Unit
BTB	Branch Target Buffer
BU	Branch instruction Unit
CC	Clock Cycle
EX	Execute
FU	Functional Unit
ID	Instruction Decode
IF	Instruction Fetch
L1	Level 1
L2	Level 2
LSU	Load and Store instructions Unit
MEM	Memory
MM	Main Menu
MMU	Memory Management Unit
MU	Multiplication Unit
OOO	Out-of-Order
PC	Program Counter
ROB	Re-Order Buffer
RSR	Result Shift Register
TLB	Translation Lookaside Buffer
VIPT	Virtually Indexed Physically Tagged
VM	Virtual Memory
WB	Write Back
WT	Write Through

Preface

This preface introduces the Deliverable d1: Processor Architecture block level design (Revision 1). It contains the following sections:

- *About this manual* on page xi
- *Using this manual* on page xi
- *Conventions* on page xii
- *References* on page xii

About this manual

The purpose of this manual is to describe a detailed design implementation of the following blocks at the elementary logic block level:

- Branch Prediction Unit
- Control Unit
- Out-of-Order
- Forwarding Unit
- Hazard Unit

The motivations of design choices are not analysed in this document, but rather their implementation is described in nuce, providing a brief description of the individual blocks, the diagram and the list of signals used with their semantics.

Main purpose of this design is to keep a good trade-off among cost, complexity and performance; metrics adopted follow the idea of saving resources in absence of significant performances improvements.

Intended audience

This manual is written for experienced hardware and software engineers who might or might not have experience of ARM products.

Using this manual

The information in this manual is organized into three chapters, as described below.

Chapter 1: Branch Prediction Unit Block

Chapter 1 describes the Branch Prediction Unit Block implementation designed by group M.

Chapter 2: Control Unit

Chapter 2 describes the Control Unit implementation according to the instruction set chosen by group M.

Chapter 3: Out-of-Order Block

Chapter 3 describes the Out-of-Order Block implementation designed by group M, reserving a paragraph for each elementary logic block.

Chapter 4: Forwarding Unit

Chapter 4 describes the Forwarding Unit implementation designed by group M.

Chapter 5: Hazard Unit

Chapter 5 describes the Hazard Unit implementation designed by group M.

Chapter 6: Resources

Chapter 6 describes how work has been organized between team members and the resulting total efforts in working hours

Conventions

Graphical

The graphical conventions for the signals within the schemes are:

Denotes aggregate bus

Denotes data bus

Denotes address bus

Denotes control bus

References

ARM publications

- ARM Architecture Reference Manual (ARM DDI 0100I)
- ARM9TDMI Technical Reference Manual (ARM DDI 0180A)

Other sources

- Computer Organization and Design (3rd edition) - David A. Patterson, John L. Hennessy
- Lecture notes of the course of Sistemi di elaborazione a.y. 2018/2019 – Prof. Angelo Marcelli

Chapter 1: Branch Prediction Unit Block

This chapter introduces the Branch Prediction Unit Block implementation and contains the following sections:

- *Introduction* on page 2
- *Prediction bits* on page 4
- *BTB Entry fields* on page 5
- *Signals* on page 6

1 Introduction

BPU block has been divided into five elementary blocks:

- Branch Recognizer
- BPU
- Aliasing Recognizer
- BTB
- Priority Updater

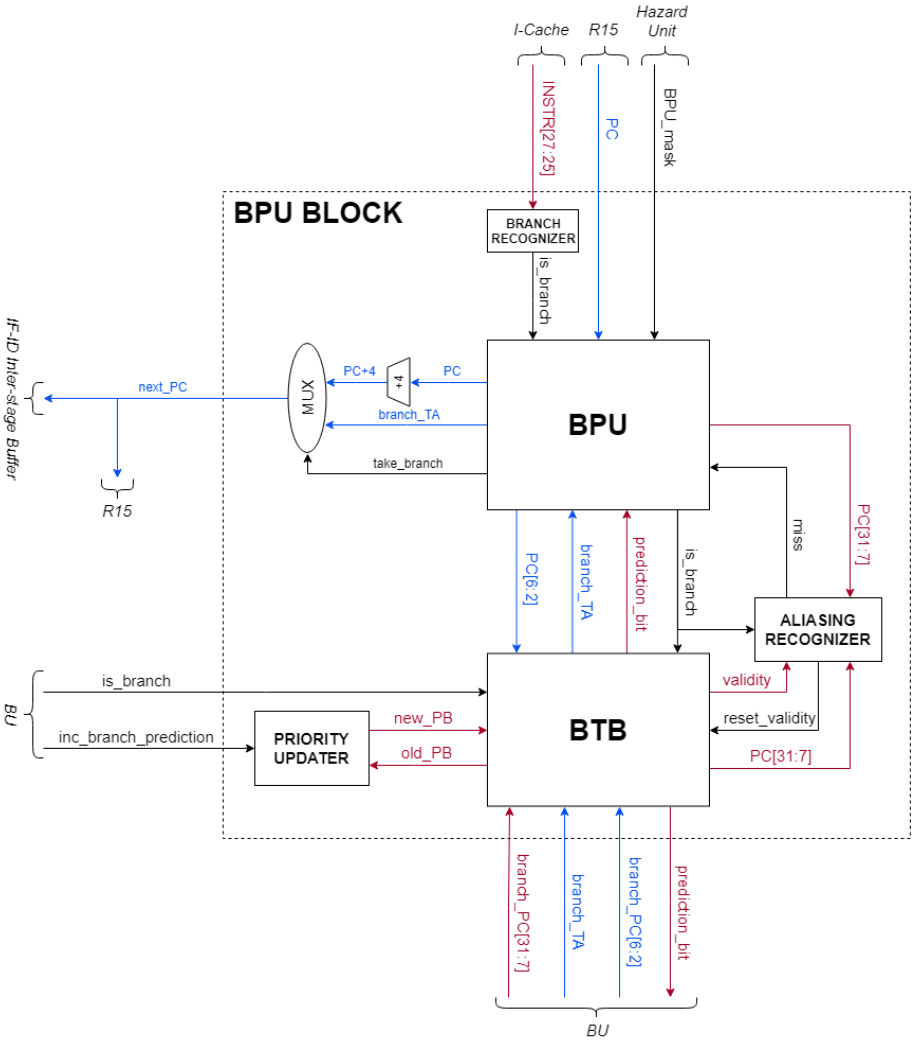

Figure 1: Branch Unit Block

Branch Recognizer performs the decode of bits [27:25] of the instruction, in order to identify branches. Output signal is_branch is asserted when these bits equals 101.

Whether a branch is detected, BPU selects the predicted target of the branch instruction, otherwise PC + 4. When is_branch signal is asserted, BPU interrogates both BTB and Aliasing Recognizer:

- BTB sends back branch target address and most significant prediction bit, required to perform predictions.
- Aliasing Recognizer manages BTB accesses in order to avoid shared entries by different branches with same five less significant bits used for direct access.

Aliasing phenomena are identified by the Program Counter of the instruction and involved entries are invalidated at their occurrence. Whenever a new branch instruction has been fetched, a new BTB entry is associated to it, even in case of aliasing phenomena.

Priority Updater manages the update of prediction bits at the end of each execution of a branch instruction. In this case, BTB access is in write mode, and it is performed by Branch Unit.

2 Prediction bits

Both prediction and update logics follow the behavior defined by the following FSM.

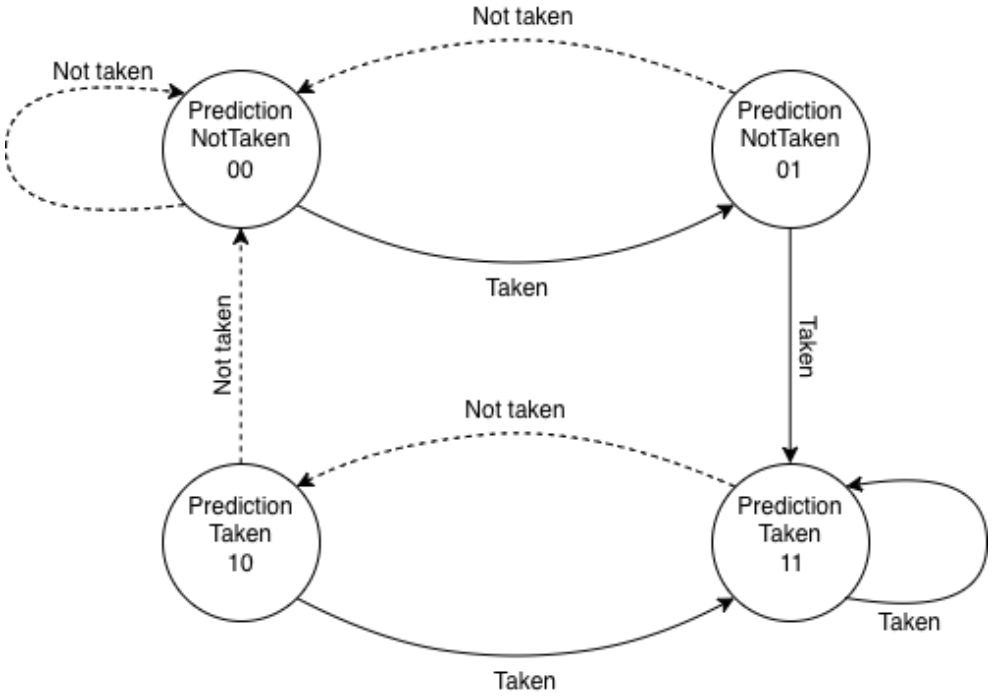

Figure 2: Prediction bits update logics

3 BTB Entry fields

Branch Target Buffer Entry fields are defined in the following table.

ENTRY FIELDS	FROM	TYPE	CARDINALITY	DESCRIPTION
V	BTB	Control	1	Validity bit
prediction_bits	Priority Updater	Data	2	Entry's prediction bits
PC[31:7]	Branch Unit	Data	25	Program Counter's 25 most significant bits to detect aliasing
branch_TA	Branch Unit	Address	32	Branch target address

Table 2: BTB Entry fields

4 Signals

Branch Prediction Unit signals are defined in the following tables.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
INSTR[27:25]	I-MMU	Data	3	Instruction bits between 27 and 25
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
is_branch	BPU	Control	1	If asserted, fetched instruction is a branch

Table 3: Branch Recognizer signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
PC	R15	Address	32	Program Counter
is_branch	Branch Recognizer	Control	1	If asserted, fetched instruction is a branch
miss	Aliasing Recognizer	Control	1	If asserted, BTB entry is not valid
prediction_bit	BTB	Data	1	Most significant prediction bit
branch_TA	BTB	Address	32	Branch target address
BPU_mask	Wrong Prediction Handler/Full Issue Register and Miss Handler	Control	1	If asserted, BPU has to be stalled
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
PC[31:7]	Aliasing Recognizer	Data	25	Program Counter's 25 most significant bits to detect aliasing
is_branch	BTB	Control	1	If asserted, fetched instruction is a branch
PC[6:2]	BTB	Data	5	Program Counter's 5 less significant bits to access BTB
branch_TA	IF-ID Inter-stage Buffer	Address	32	Branch target address
PC	IF-ID Inter-stage Buffer	Address	32	Program Counter
take_branch	IF-ID Inter-stage Buffer	Control	1	If asserted, branch is taken and next PC is branch_TA

Table 4: BPU signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
is_branch	BPU	Control	1	If asserted, fetched instruction is a branch
validity	BTB	Control	1	If asserted, searched BTB entry is valid
PC[31:7]	BTB	Data	25	Program Counter's 25 most significant bits to detect aliasing
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
miss	BPU	Control	1	If asserted, BTB entry is not valid
reset_validity	BTB	Control	1	If asserted, current BTB entry is no more valid

Table 5: Aliasing Recognizer signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
PC[6:2]	BPU	Address	5	Program Counter
is_branch	BPU	Control	1	If asserted, fetched instruction is a branch
reset_validity	Aliasing Recognizer	Control	1	If asserted, current BTB entry is no more valid
new_PB	Priority Updater	Data	2	Updated prediction bits
is_branch	Branch Unit	Control	1	If asserted, BTB writing is allowed
branch_PC[6:2]	Branch Unit	Address	5	Program Counter's 5 less significant bits to access BTB
branch_PC[31:7]	Branch Unit	Data	25	Program Counter's 25 most significant bits to detect aliasing
branch_TA	Branch Unit	Address	32	Branch target address to store
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
branch_TA	BPU	Address	32	Branch target address from BTB
prediction_bit	BPU	Control	1	Most significant prediction bit
validity	Aliasing Recognizer	Data	1	If asserted, searched BTB entry is valid
PC[31:7]	Aliasing Recognizer	Data	25	Program Counter's 25 most significant bits to detect aliasing
old_PB	Priority Updater	Data	2	Old prediction bits to update
prediction_bit	Branch Unit	Data	1	Most significant prediction bit

Table 6: BTB signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
old_PB	BTB	Data	2	Old prediction bits to update
inc_branch_priority	Branch Unit	Control	1	If asserted, old prediction bits have to be incremented by one
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
new_PB	BTB	Data	2	Updated prediction bits

Table 7: Priority Updater signals

Chapter 2: Control Unit

This chapter introduces the Control Unit implementation and contains the following sections:

- *Introduction* on page 8
- *Instruction Decoder* on page 9
- *Supported instruction set* on page 10
- *Forward Decoder* on page 11

1 Introduction

Block level design of Control Unit consists in two elements:

- Instruction Decoder
- Forward Decoder

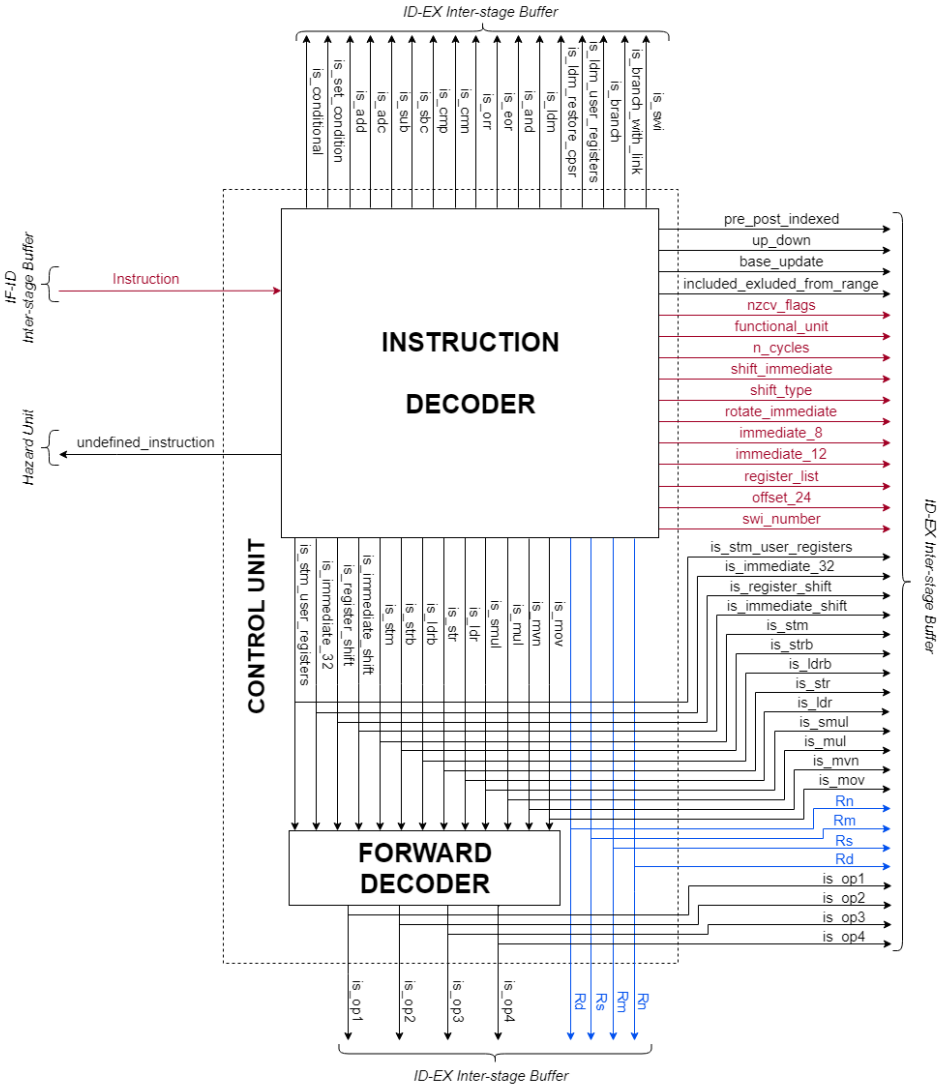

Figure 3: Control Unit

2 Instruction Decoder

The Instruction Decoder generates a 34-bits header of control signals and all data and address fields required for the pipeline operations.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
Instruction	IF-ID Inter-stage Buffer	Data	32	Fetched instruction
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
is_conditional	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction execution is conditional
is_set_condition	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction sets conditional flags
is_add	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is an ADD
is_adc	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is an ADD with carry
is_sub	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a SUB
is_sbc	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a SUB with carry
is_cmp	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a COMPARE
is_cmn	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a COMPARE NOT
is_orr	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is an OR
is_eor	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is an EXCLUSIVE OR
is_and	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is an AND
is_ldm	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a LOAD MULTIPLE
is_ldm_restore_cpsr	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a LOAD MULTIPLE, PC is involved and CPSR is restored from SPSR
is_ldm_user_registers	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a LOAD MULTIPLE, user registers are involved
is_branch	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a branch
is_branch_with_link	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a branch with link
is_swi	ID-EX Inter-stage Buffer	Control	1	If asserted, istruction is a software interrupt
pre_post_indexed	ID-EX Inter-stage Buffer	Control	1	If asserted, pre-indexed addressing is applied, otherwise, post-indexed addressing.
up_down	ID-EX Inter-stage Buffer	Control	1	If asserted, increment is applied to the base register; otherwise, decrement.
base_update	ID-EX Inter-stage Buffer	Control	1	If asserted, calculated memory address is written back to the base register; otherwise it's not updated
included_exluded_from_range	ID-EX Inter-stage Buffer	Control	1	If asserted, word addressed by base register is included in the range of memory locations accessed; otherwise, it's excluded
nzcv_flags	ID-EX Inter-stage Buffer	Data	4	NZCV condition flags of the instruction
functional_unit	ID-EX Inter-stage Buffer	Data	2	Functional Unit assigned: 00 = AU - 01 = MU - 10 = BU - 11 = LSU
n_cycles	ID-EX Inter-stage Buffer	Data	4	Number of clock cycles to execute the instruction
shift_immediate	ID-EX Inter-stage Buffer	Data	5	Data processing immediate shift and Load/Store register offset instructions' immediate shift amount
shift_type	ID-EX Inter-stage Buffer	Data	2	Data processing and load/store instructions' rotate, logical or arithmetic shift type
rotate_immediate	ID-EX Inter-stage Buffer	Data	4	Data processing immediate instructions' immediate rotate amount
immediate_8	ID-EX Inter-stage Buffer	Data	8	Data processing immediate instructions' immediate value
immediate_12	ID-EX Inter-stage Buffer	Data	12	Load/Store immediate offset instructions' immediate offset
register_list	ID-EX Inter-stage Buffer	Data	16	Load/Store multiple instructions' register list
offset_24	ID-EX Inter-stage Buffer	Data	24	Branch instructions' offset
swi_number	ID-EX Inter-stage Buffer	Data	24	Software interrupt instructions' number
is_immediate_shift	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a data processing immediate shift or a load/store immediate offset
is_register_shift	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a data processing register shift or a load/store register offset
is_immediate_32	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a data processing immediate
is_mov	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a MOVE
is_mvn	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a MOVE NOT
is_mul	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a MUL
is_smul	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a SIGNED MUL
is_ldr	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a LOAD WORD
is_str	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a STORE WORD
is_ldrb	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a LOAD BYTE
is_strb	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a STORE BYTE
is_stm	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE
is_stm_user_registers	ID-EX Inter-stage Buffer/Forward Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE, user registers are involved
Rn	ID-EX Inter-stage Buffer/Forwarding Unit	Address	4	Data processing instructions' 1st operand register, or Load/Store instructions' base register
Rm	ID-EX Inter-stage Buffer/Forwarding Unit	Address	4	Data processing instructions' 2nd operand register, or Load/Store register offset instructions' offset register
Rs	ID-EX Inter-stage Buffer/Forwarding Unit	Address	4	Data processing register shift instructions' register shift amount
Rd	ID-EX Inter-stage Buffer/Forwarding Unit	Address	4	Data processing instructions' destination register, or Load/Store instructions' destination/source register
undefined_instruction	Hazard Unit	Control	1	If asserted, an undefined instruction has been decoded

Table 8: Instruction Decoder signals

3 Supported instruction set

Supported instruction set is based on ARMv4T.

Arithmetic	Logical	Multiply	Load/Store	Branch	Software Interrupt
ADD	MOV	MUL	LDR	B	SWI
ADC	MVN	SMUL	STR	BL	
SUB	CMP		LDRB		
SBC	CMN		STRB		
	ORR		LDM		
	EOR		STM		
	AND				

Table 9: Supported instruction set

4 Forward Decoder

Forward Decoder generates four different control signals to interrogate the forwarding table; each one of them is needed to determine whether one of the four admitted operands (that is to say Rn, Rm, Rs and Rd) has been taken from the Register Bank. By performing this, a first check of the coherence between values of registers stored in the Register Bank and those stored in the forwarding table is made in the instruction decode. An additional check is performed in the execution stage, due to possible changes in the status of the forwarding table.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
is_immediate_shift	Instruction Decoder	Control	1	If asserted, istruction is a data processing immediate shift or a load/store immediate offset
is_register_shift	Instruction Decoder	Control	1	If asserted, istruction is a data processing register shift or a load/store register offset
is_immediate_32	Instruction Decoder	Control	1	If asserted, istruction is a data processing immediate
is_mov	Instruction Decoder	Control	1	If asserted, istruction is a MOVE
is_mvn	Instruction Decoder	Control	1	If asserted, istruction is a MOVE NOT
is_mul	Instruction Decoder	Control	1	If asserted, istruction is a MUL
is_smul	Instruction Decoder	Control	1	If asserted, istruction is a SIGNED MUL
is_ldr	Instruction Decoder	Control	1	If asserted, istruction is a LOAD WORD
is_str	Instruction Decoder	Control	1	If asserted, istruction is a STORE WORD
is_ldrb	Instruction Decoder	Control	1	If asserted, istruction is a LOAD BYTE
is_strb	Instruction Decoder	Control	1	If asserted, istruction is a STORE BYTE
is_stm	Instruction Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE
is_stm_user_registers	Instruction Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE, user registers are involved
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
is_op1	ID-EX Inter-stage Buffer/Forwarding Unit	Control	1	If asserted, Rn is from register
is_op2	ID-EX Inter-stage Buffer/Forwarding Unit	Control	1	If asserted, Rm is from register
is_op3	ID-EX Inter-stage Buffer/Forwarding Unit	Control	1	If asserted, Rs is from register
is_op4	ID-EX Inter-stage Buffer/Forwarding Unit	Control	1	If asserted, Rd is from register

Table 10: Forward Decoder signals

Chapter 3: Out-of-Order Block

This chapter introduces the Out-of-Order block implementation and contains the following sections:

- *Introduction* on page 13
- *Issue Register* on page 14
- *RSR* on page 17
- *Shunt Unit* on page 19
- *Acceptance Unit* on page 21
- *Condition Verifier* on page 23
- *Issuing Unit* on page 24
- *Shifter* on page 26
- *Routing Unit* on page 27
- *Functional Units* on page 29
- *ROB* on page 31

1 Introduction

Main purpose of this block is to avoid as much as possible pipeline stalls. In order to achieve this purpose, an ad-hoc solution for the Forwarding Unit has been adopted. To pursue the line of cost-complexity-performance trade-off, Forwarding Unit provides both operand forward and dependency detection by using a table, thus a memory element. Whereas, the other blocks defined within the OOO Block are intended to guarantee low complexity and good performances.

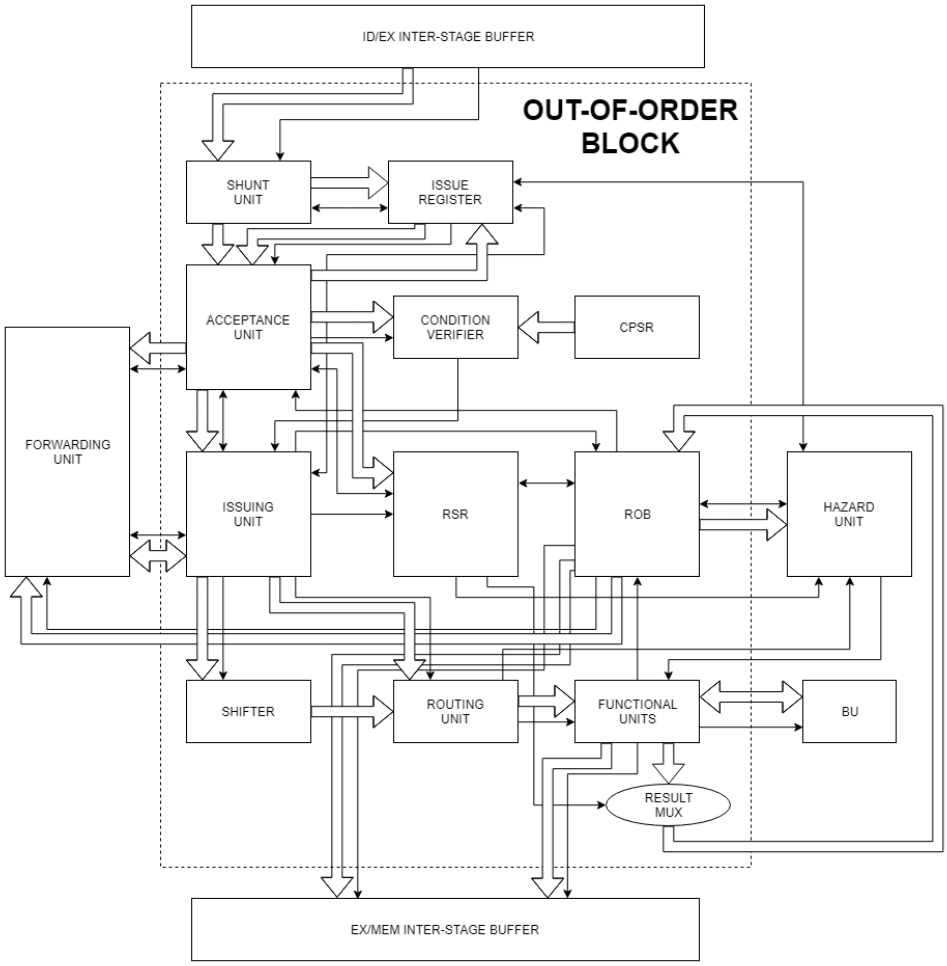

Figure 4: Out-of-Order Block

2 Issue Register

2.1 Introduction

Issue Register facilities have been divided into three different blocks:

- Shunt Unit
- Acceptance Unit
- Issuing Unit

These blocks are defined inside the Out-Of-Order Execution Block and their purposes are discussed further on in this chapter.

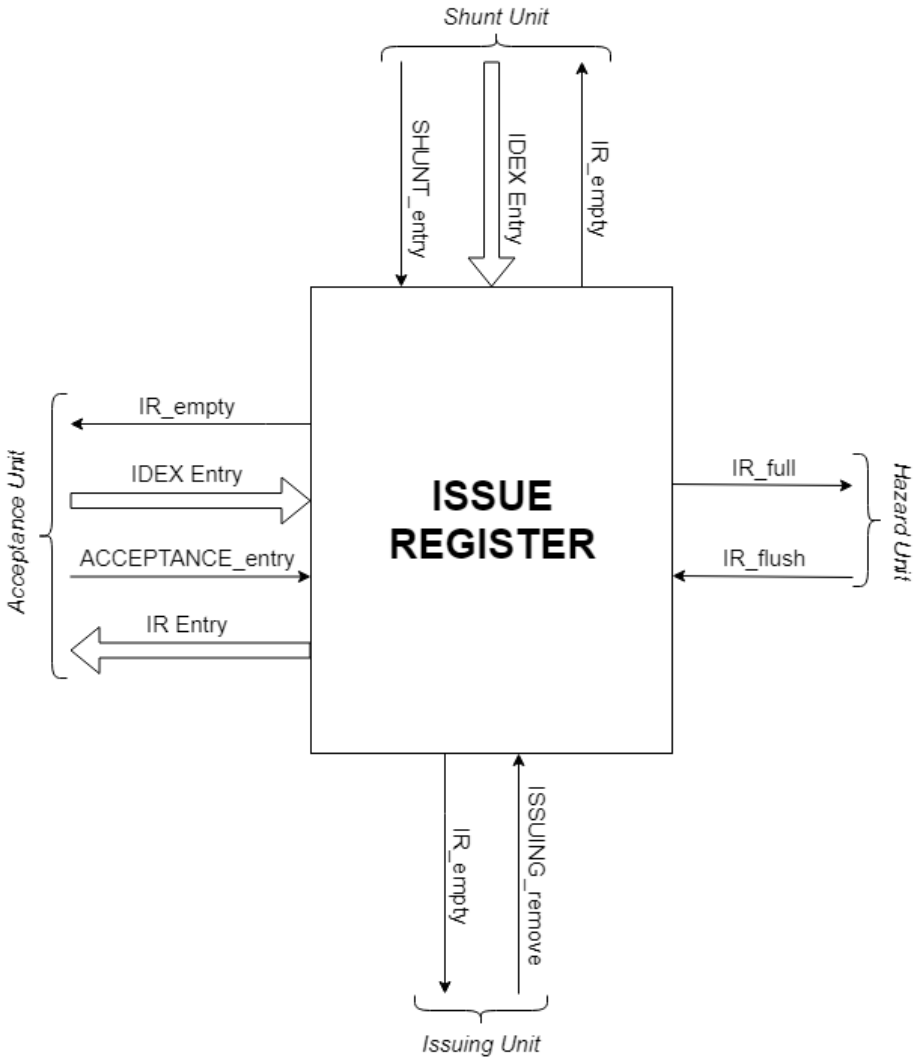

Figure 5: Issue Register

2.2 Entry fields

The Issue Register itself acts as a 16-entries memory component and its fields are defined in the following table.

ENTRY	FROM	TYPE	CARDINALITY	DESCRIPTION
PC	IF-ID Inter-stage Buffer	Data	32	Program Counter
is_immediate_shift	Instruction Decoder	Control	1	If asserted, istruction is a data processing immediate shift or a load/store immediate offset
is_register_shift	Instruction Decoder	Control	1	If asserted, istruction is a data processing register shift or a load/store register offset
is_immediate_32	Instruction Decoder	Control	1	If asserted, istruction is a data processing immediate
is_conditional	Instruction Decoder	Control	1	If asserted, istruction execution is conditional
is_set_condition	Instruction Decoder	Control	1	If asserted, istruction sets conditional flags
is_add	Instruction Decoder	Control	1	If asserted, istruction is an ADD
is_adc	Instruction Decoder	Control	1	If asserted, istruction is an ADD with carry
is_sub	Instruction Decoder	Control	1	If asserted, istruction is a SUB
is_sbc	Instruction Decoder	Control	1	If asserted, istruction is a SUB with carry
is_mov	Instruction Decoder	Control	1	If asserted, istruction is a MOVE
is_mvn	Instruction Decoder	Control	1	If asserted, istruction is a MOVE NOT
is_cmp	Instruction Decoder	Control	1	If asserted, istruction is a COMPARE
is_cmn	Instruction Decoder	Control	1	If asserted, istruction is a COMPARE NOT
is_orr	Instruction Decoder	Control	1	If asserted, istruction is an OR
is_eor	Instruction Decoder	Control	1	If asserted, istruction is an EXCLUSIVE OR
is_and	Instruction Decoder	Control	1	If asserted, istruction is an AND
is_mul	Instruction Decoder	Control	1	If asserted, istruction is a MUL
is_smul	Instruction Decoder	Control	1	If asserted, istruction is a SIGNED MUL
is_ldr	Instruction Decoder	Control	1	If asserted, istruction is a LOAD WORD
is_str	Instruction Decoder	Control	1	If asserted, istruction is a STORE WORD
is_ldrb	Instruction Decoder	Control	1	If asserted, istruction is a LOAD BYTE
is_strb	Instruction Decoder	Control	1	If asserted, istruction is a STORE BYTE
is_ldm	Instruction Decoder	Control	1	If asserted, istruction is a LOAD MULTIPLE
is_stm	Instruction Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE
is_ldm_restore_cpsr	Instruction Decoder	Control	1	If asserted, istruction is a LOAD MULTIPLE, PC is involved and CPSR is restored from SPSR (return from exception)
is_ldm_user_registers	Instruction Decoder	Control	1	If asserted, istruction is a LOAD MULTIPLE, user registers are involved (only privileged mode)
is_stm_user_registers	Instruction Decoder	Control	1	If asserted, istruction is a STORE MULTIPLE, user registers are involved (only privileged mode)
is_branch	Instruction Decoder	Control	1	If asserted, istruction is a branch
is_branch_with_link	Instruction Decoder	Control	1	If asserted, istruction is a branch with link
is_swi	Instruction Decoder	Control	1	If asserted, istruction is a software interrupt
pre_post_indexed	Instruction Decoder	Control	1	If asserted, pre-indexed addressing is applied, otherwise, post-indexed addressing.
up_down	Instruction Decoder	Control	1	If asserted, increment is applied to the base register; otherwise, decrement.
base_update	Instruction Decoder	Control	1	If asserted, calculated memory address is written back to the base register; otherwise it's not updated
included_exluded_from_range	Instruction Decoder	Control	1	If asserted, word addressed by base register is included in the range of memory locations accessed; otherwise, it's excluded
nzcv_flags	Instruction Decoder	Data	4	NZCV condition flags of the instruction
functional_unit	Instruction Decoder	Data	2	Functional Unit assigned: 00 = AU - 01 = MU - 10 = BU - 11 = LSU
n_cycles	Instruction Decoder	Data	4	Number of clock cycles to execute the instruction
Rn	Instruction Decoder	Address	4	Data processing instructions' 1st operand register, or Load/Store instructions' base register
Rm	Instruction Decoder	Address	4	Data processing instructions' 2nd operand register, or Load/Store register offset instructions' offset register
Rs	Instruction Decoder	Address	4	Data processing register shift instructions' register shift amount
Rd	Instruction Decoder	Address	4	Data processing instructions' destination register, or Load/Store instructions' destination/source register
shift_immediate	Instruction Decoder	Data	5	Data processing immediate shift and Load/Store register offset instructions' immediate shift amount
shift_type	Instruction Decoder	Data	2	Data processing and load/store instructions' rotate, logical or arithmetic shift type
rotate_immediate	Instruction Decoder	Data	4	Data processing immediate instructions' immediate rotate amount
immediate_8	Instruction Decoder	Data	8	Data processing immediate instructions' immediate value
immediate_12	Instruction Decoder	Data	12	Load/Store immediate offset instructions' immediate offset
register_list	Instruction Decoder	Data	16	Load/Store multiple instructions' register list
offset_24	Instruction Decoder	Data	24	Branch instructions' offset
swi_number	Instruction Decoder	Data	24	Software interrupt instructions' number
is_op1	Forward Decoder	Control	1	If asserted, Rn is from register
is_op2	Forward Decoder	Control	1	If asserted, Rm is from register
is_op3	Forward Decoder	Control	1	If asserted, Rs is from register
is_op4	Forward Decoder	Control	1	If asserted, Rd is from register
word_1	Register Bank	Data	32	Rn register value
word_2	Register Bank	Data	32	Rm register value
word_3	Register Bank	Data	32	Rs register value, or Store instructions' Rd register value
coherence_1	Forwarding Unit	Data	1	Rn register coherence bit from forwarding table
coherence_2	Forwarding Unit	Data	1	Rm register coherence bit from forwarding table
coherence_3	Forwarding Unit	Data	1	Rs register coherence bit from forwarding table
coherence_4	Forwarding Unit	Data	1	Rd register coherence bit from forwarding table

Table 11: Issue Register entry fields

2.3 Signals

Fetch and Decode are always performed when IR_full output signal is not asserted, otherwise mask signals are sent to IF/ID and ID/EX Inter-stage Buffers.

Whenever a wrong prediction occurs, IR_flush input signal is asserted to request the reset of the register.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
SHUNT_entry	Shunt Unit	Control	1	If asserted, new entry is being queued into Issue Register
IDEX Entry	Shunt Unit/Acceptance Unit	Address/Data/Control	Aggregate	If issue_empty is not asserted, ID-EX Entry stored into Issue Register
ACCEPTANCE_entry	Acceptance Unit	Control	1	If asserted, new entry is being queued into Issue Register
ISSUING_remove	Issuing Unit	Control	1	If asserted, instruction pointed by header pointer of issue register has to be removed
IR_flush	Hazard Unit	Control	1	If asserted, flush of Issue Register executed
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
IR_empty	Shunt Unit/Acceptance Unit/Issuing Unit	Control	1	If asserted, Issue Register is empty
IR Entry	Acceptance Unit	Address/Data/Control	Aggregate	Issue Register entry to accept
IR_full	Hazard Unit	Control	1	If asserted, Issue Register is full, fetch and decode stage stall is required

Table 12: Issue Register signals

3 RSR

3.1 Introduction

Result Shift Register has been designed to manage the reservation of Re-Order Buffer and also the access to the four different functional units.

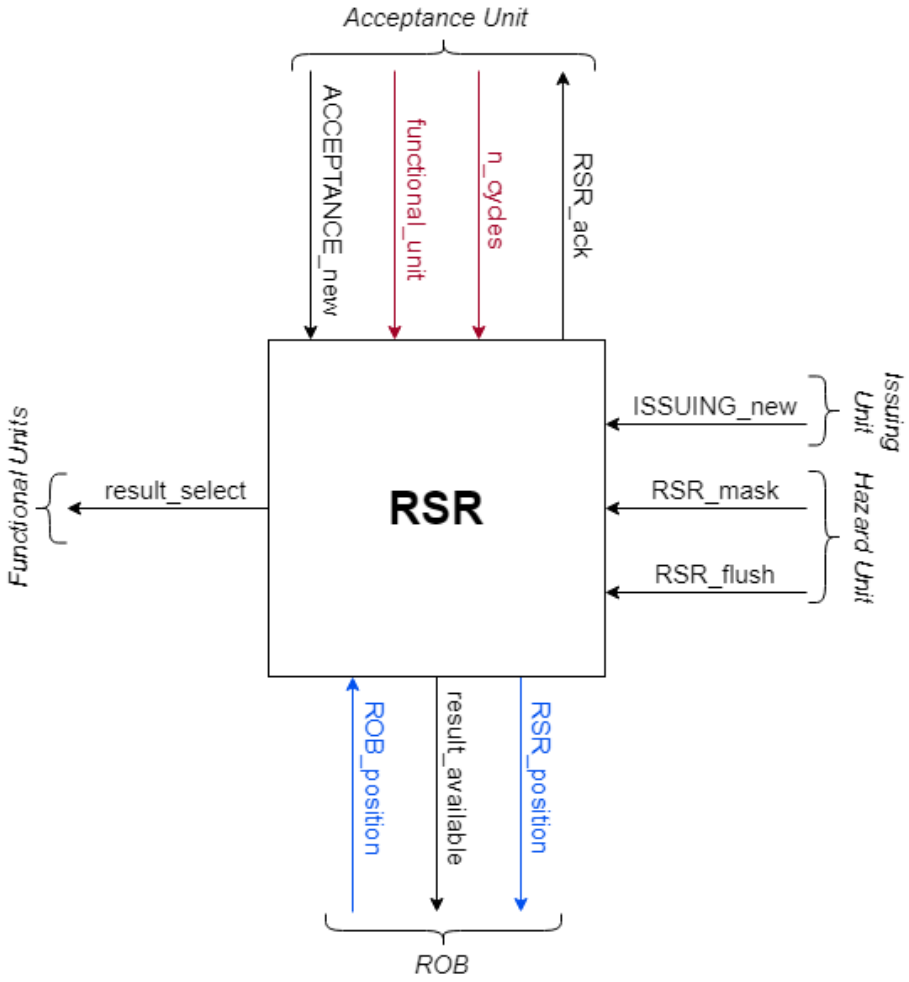

Figure 6: RSR

3.2 Entry fields

Register's size is fixed to the longest clock cycle time among the supported instructions, that is 16 clock cycles for Load/Store Multiple instructions and its entry is defined as follows.

ENTRY FIELDS	FROM	TYPE	CARDINALITY	DESCRIPTION
V	RSR	Control	1	Validity Bit
functional_unit	Acceptance Unit	Data	2	Functional Unit assigned: 00 = AU - 01 = MU - 10 = BU - 11 = LSU
ROB_position	ROB	Address	4	ROB position to store the result in

Table 13: RSR table entry fields

3.3 Signals

Control signals are sent through the OOO-Block to support issuing process and ROB management.

RSR can be stalled, through RSR_mask input signal, on D-Cache misses during Load/Store Multiple operations, or be flushed, with RSR_flush, in case of wrong prediction.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
ACCEPTANCE_new	Acceptance Unit	Control	1	If asserted, functional_unit and n_cycles input from Acceptance Unit
functional_unit	Acceptance Unit	Data	2	Functional Unit assigned: 00 = AU - 01 = MU - 10 = BU - 11 = LSU
n_cycles	Acceptance Unit	Data	4	Number of clock cycles to execute the instruction
ISSUING_new	Issuing Unit	Control	1	If asserted, current instruction has been issued
RSR_mask	Hazard Unit	Control	1	If asserted, RSR has to be stalled
RSR_flush	Hazard Unit	Control	1	If asserted, RSR has to be flushed
ROB_position	ROB	Address	4	ROB position pointed by trailer pointer
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
RSR_ack	Acceptance Unit	Control	1	If asserted, time slot and functional unit required are available
result_select	Functional Units	Control	2	Control signal to manage result selector multiplexer
result_available	ROB	Control	1	If asserted, a new result is available from the functional units
RSR_position	ROB	Address	4	ROB position to access for storing result

Table 14: RSR signals

4 Shunt Unit

4.1 Introduction

The Shunt Unit performs bypass of Issue Register by forwarding the data coming from ID/EX Inter-stage to the Acceptance Unit, when IR_empty input signal is asserted, otherwise the data is stored inside the register.

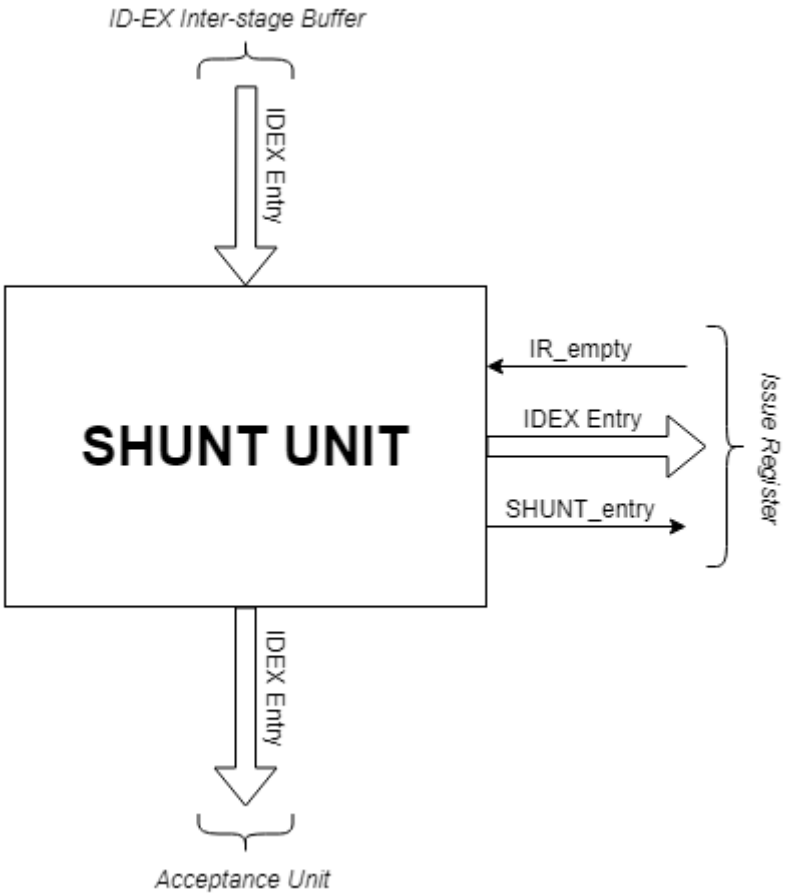

Figure 7: Shunt Unit

4.2 Signals

Shunt Unit signals are defined in the following table.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
IDEX Entry	ID-EX Inter-stage Buffer	Address/Data/Control	Aggregate	If issue_empty is not asserted, ID-EX Entry stored into Issue Register
IR_empty	Issue Register	Control	1	If asserted, Issue Register is empty and bypass is legal
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
SHUNT_entry	Issue Register	Control	1	If asserted, new entry is being queued into Issue Register
IDEX Entry	Issue Register/Acceptance Unit	Address/Data/Control	Aggregate	If issue_empty is not asserted, ID-EX Entry stored into Issue Register

Table 15: Shunt Unit signals

5 Acceptance Unit

5.1 Introduction

The Acceptance Unit provides:

- Operand forward request
- Data dependency detection
- Conditional execution dependency detection
- Functional Units and ROB reservation

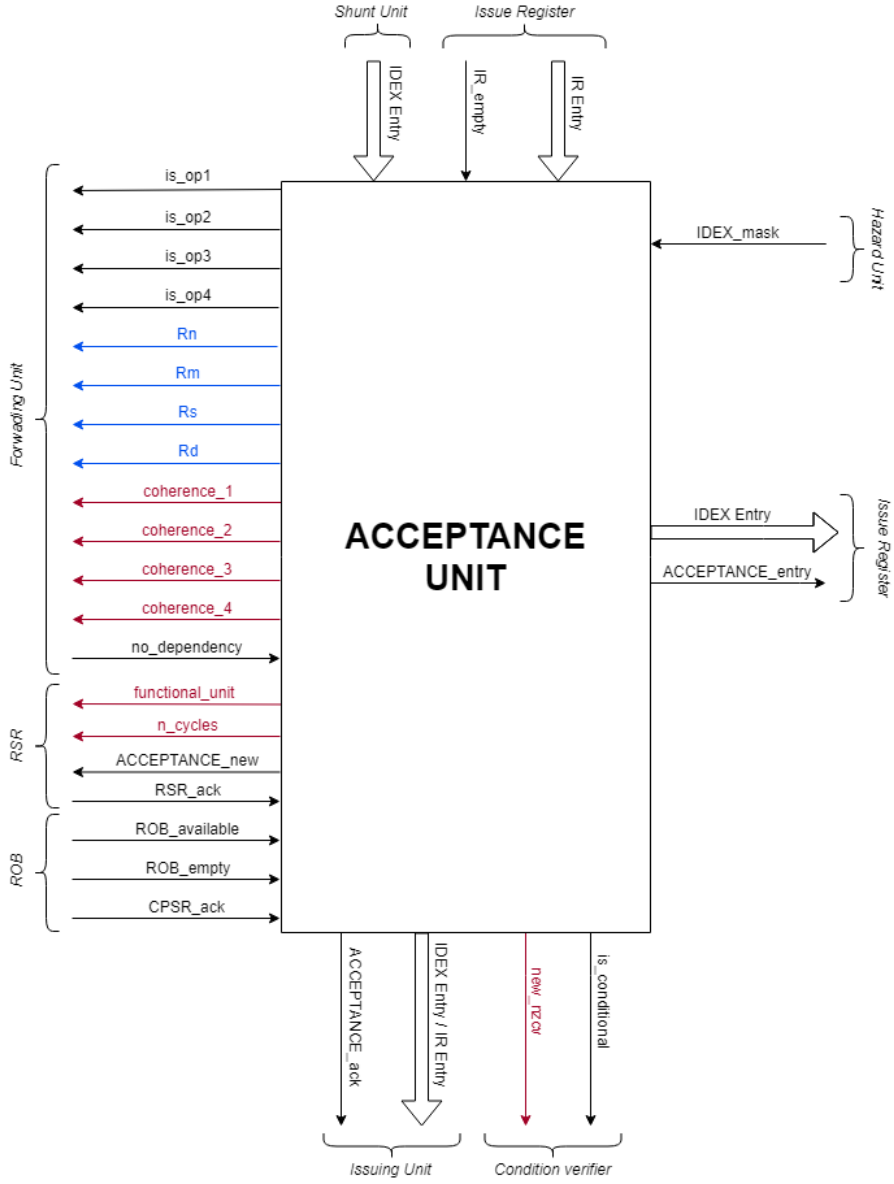

Figure 8: Acceptance Unit

5.2 Signals

Operand forward and data dependency detection are both demanded to the Forwarding Unit, but while operand forward cannot be performed within this block, only a no_dependency control signal is sent back to Acceptance Unit to notify if the instruction can be issued or not, due to a possible data dependency. Thus, if no_dependency is asserted, the instruction is not issued.

RSR sends its own RSR_ack signal to notify about Functional Units and time slots availability, and the same is performed by ROB. Moreover, Load/Store and Software Interrupt instructions need the ROB to be empty before being issued, while any other instruction just needs availability.

Conditional or set condition instructions cannot be issued while CPSR_ack is not asserted, due to CPSR status not updated. CPSR_ack will be asserted only when every instruction stored into ROB marked as is_set_condition will have completed its execution.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
IDEX Entry/IR Entry	Shunt Unit/Issue Register	Address/Data/Control	Aggregate	Entry from Issue Register, or ID-EX Inter-stage Buffer if bypass has been performed
IR_empty	Issue Register	Control	1	If asserted, Issue Register bypass has been performed
RSR_ack	RSR	Control	1	if asserted, functional unit and time slot dedicated for the instruction are available
no_dependency	Forwarding Unit	Control	1	If asserted, no dependency occurs for the instruction
IDEX_mask	Hazard Unit	Control	1	If asserted, ID-EX Inter-stage Buffer has been stalled
ROB_available	ROB	Control	1	If asserted, ROB is not full and the instruction can be accepted
ROB_empty	ROB	Control	1	If asserted, ROB is empty and the instruction can be accepted
CPSR_ack	ROB	Control	1	If asserted, CPSR has been updated and conditional execution can be verified
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
IDEX Entry	Issue Register	Address/Data/Control	Aggregate	If instruction is not accepted, ID-EX Entry stored into Issue Register
ACCEPTANCE_entry	Issue Register	Control	1	If asserted, new entry is being queued into Issue Register
ACCEPTANCE_new	RSR	Control	1	If asserted, a new instruction requests to be issued to RSR
functional_unit	RSR	Data	2	Functional Unit assigned: 00 = AU - 01 = MU - 10 = BU - 11 = LSU
n_cycles	RSR	Data	4	Number of clock cycles to execute the instruction
is_op1	Forwarding Unit	Control	1	If asserted, Rn is from register
is_op2	Forwarding Unit	Control	1	If asserted, Rm is from register
is_op3	Forwarding Unit	Control	1	If asserted, Rs is from register
is_op4	Forwarding Unit	Control	1	If asserted, Rd is from register
Rn	Forwarding Unit	Address	4	Data processing instructions' 1st operand register, or Load/Store instructions' base register
Rm	Forwarding Unit	Address	4	Data processing instructions' 2nd operand register, or Load/Store register offset instructions' offset register
Rs	Forwarding Unit	Address	4	Data processing register shift instructions' register shift amount
Rd	Forwarding Unit	Address	4	Data processing instructions' destination register, or Load/Store instructions' destination/source register
coherence_1	Forwarding Unit	Data	1	Rn register Coherence bit from forwarding table
coherence_2	Forwarding Unit	Data	1	Rm register Coherence bit from forwarding table
coherence_3	Forwarding Unit	Data	1	Rs register Coherence bit from forwarding table
coherence_4	Forwarding Unit	Data	1	Rd register Coherence bit from forwarding table
is_conditional	Condition Verifier	Control	1	If asserted, istruction execution condition has to be verified
new_nzcv	Condition Verifier	Data	4	NZCV condition flags of current instruction
ACCEPTANCE_ack	Issuing Unit	Control	1	If asserted, current instruction has been accepted and needs to be issued
IDEX Entry/IR Entry	Issuing Unit	Address/Data/Control	Aggregate	ID-EX or Issue Register Entry being accepted and to be issued

Table 16: Acceptance Unit signals

6 Condition Verifier

6.1 Introduction

Condition Verifier block performs a comparison between instruction's NZCV flags and CPSR's ones and the result is sent to the Issuing Unit as a control signal.

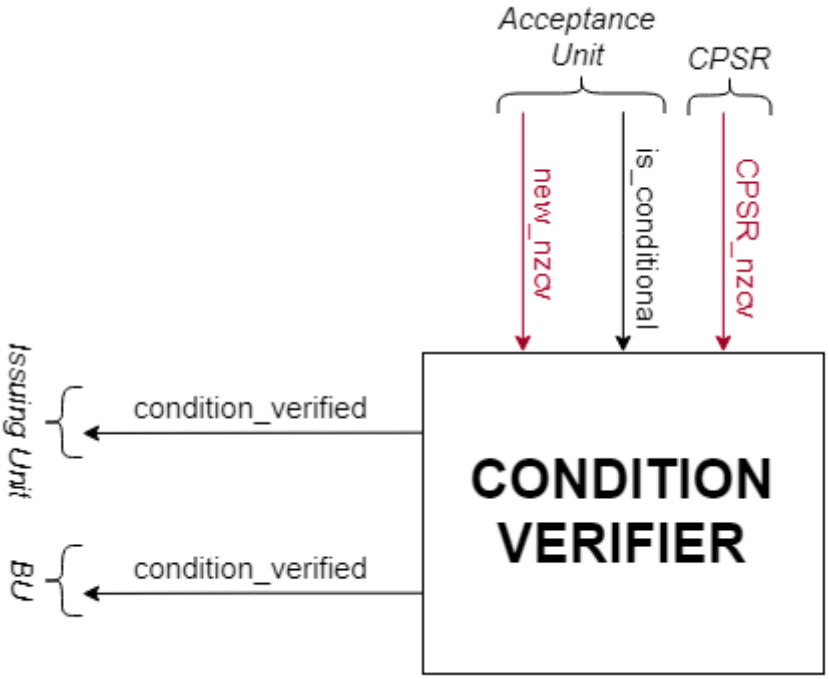

Figure 9: Condition Verifier

6.2 Signals

Condition Verifier signals are defined in the following table.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
is_conditional	Acceptance Unit	Control	1	If asserted, istruction execution condition has to be verified
new_nzcv	Acceptance Unit	Data	4	NZCV condition flags of current instruction
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
condition_verified	Issuing Unit	Control	1	If asserted, conditional execution is allowed

Table 17: Condition Verifier signals

7 Issuing Unit

7.1 Introduction

Once the instruction has been accepted, the Issuing Unit receives updated operands from the Forwarding Unit and the result from the Condition Verifier.

For unconditional execution or condition match, proper fields are sent to ROB, including all data required to obtain the shifter operand sent to the shifter and the instruction is removed from Issue Register. On condition mismatch, the instruction is removed from Issue Register, but it is not executed. The removal from Issue Register happens only if bypass has not been executed, otherwise the instruction coming from ID/EX Inter-stage Buffer is ignored.

In case of Load Multiple instructions, the register list is sent to the Forwarding Unit with a proper control signal.

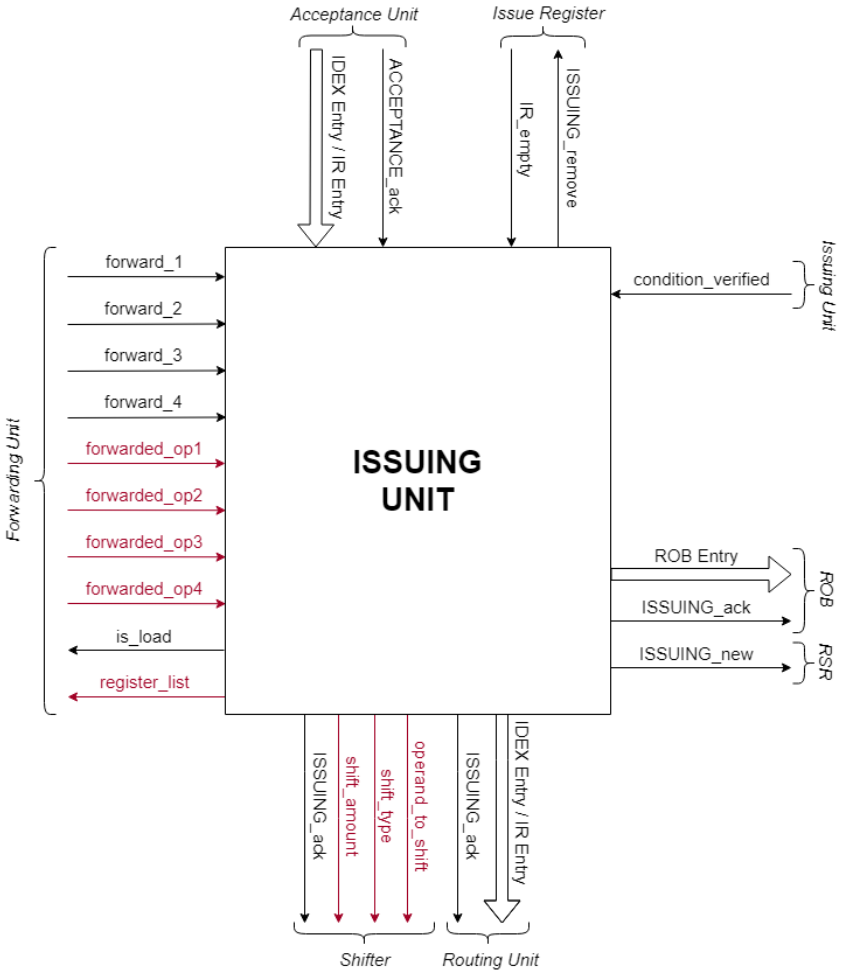

Figure 10: Issuing Unit

7.2 Signals

Issuing Unit signals are defined in the following table.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
IR_empty	Issue Register	Control	1	If asserted, no remove from Issue Register is required
IDEX Entry/IR Entry	Acceptance Unit	Address/Data/Control	Aggregate	Entry from Issue Register, or ID-EX Inter-stage Buffer if bypass has been performed
ACCEPTANCE_ack	Acceptance Unit	Control	1	If asserted, a new entry has been accepted
condition_verified	Condition Verifier	Control	1	If asserted, conditional execution is allowed
forward_1	Forwarding Unit	Control	1	If asserted, operand 1 has been forwarded
forward_2	Forwarding Unit	Control	1	If asserted, operand 2 has been forwarded
forward_3	Forwarding Unit	Control	1	If asserted, operand 3 has been forwarded
forward_4	Forwarding Unit	Control	1	If asserted, operand 4 has been forwarded
forwarded_op1	Forwarding Unit	Data	32	Operand 1 from forwarding table
forwarded_op2	Forwarding Unit	Data	32	Operand 2 from forwarding table
forwarded_op3	Forwarding Unit	Data	32	Operand 3 from forwarding table
forwarded_op4	Forwarding Unit	Data	32	Operand 4 from forwarding table
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
ISSUING_remove	Issue Register	Control	1	If asserted, instruction pointed by header pointer of issue register has to be removed
ISSUING_new	RSR	Control	1	If asserted, current instruction is being issued
is_load	Forwarding Unit	Control	1	If asserted, a load multiple is being issued and multiple dependency bits need to be asserted
register_list	Forwarding Unit	Data	16	Register list to decode to assert dependency bits of registers involved in load multiple instructions
shift_type	Shifter	Data	2	Rotate, logical or arithmetic shift type
shift_amount	Shifter	Data	32	Shift amount
operand_to_shift	Shifter	Data	32	Operand to shift
ISSUING_ack	Shifter/Routing Unit/ROB	Control	1	If asserted, a new entry has been issued
IDEX Entry/IR Entry	Routing Unit	Address/Data/Control	Aggregate	ID-EX or Issue Register Entry being issued, needs to be routed to the correct Functional Unit or to Hazard Unit
ROB Entry	ROB	Address/Data/Control	Aggregate	Entry to be stored into ROB

Table 18: Issuing Unit signals

8 Shifter

8.1 Introduction

Shifter block evaluates the shifter operand according to the shift type and amount.

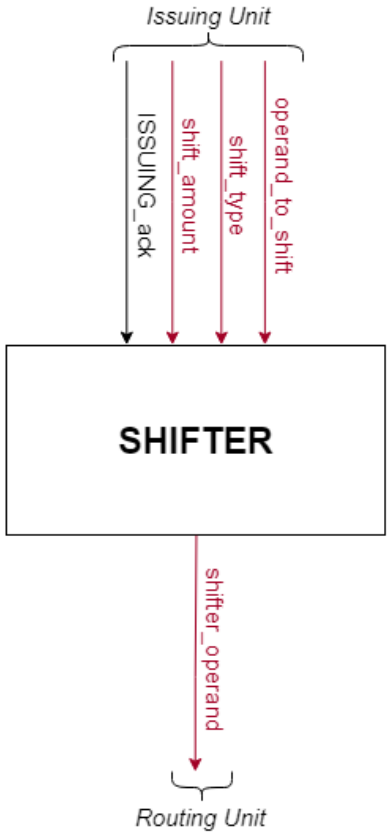

Figure 11: Shifter

8.2 Signals

Shifter signals are defined in the following table.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
ISSUING_ack	Issuing Unit	Control	1	If asserted, a new entry has been issued
shift_type	Issuing Unit	Data	2	Rotate, logical or arithmetic shift type
shift_amount	Issuing Unit	Data	32	Shift amount
operand_to_shift	Issuing Unit	Data	32	Operand to shift
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
shifter_operand	Routing Unit	Data	32	Shifter Operand value

Table 19: Table signals

9 Routing Unit

9.1 Introduction

Routing Unit provides the selection of the correct functional unit to serve.

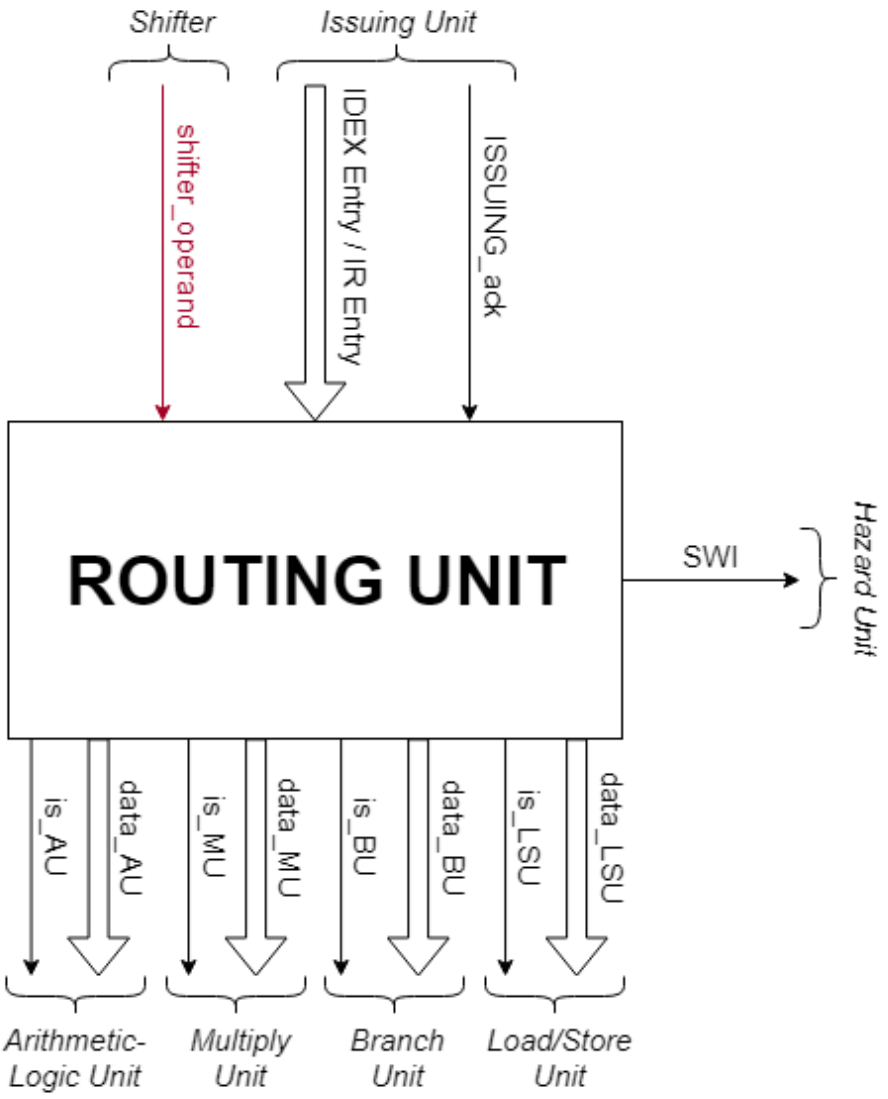

Figure 12: Routing Unit

9.2 Signals

In case of Software Interrupt instruction, SWI control signal is sent to Hazard Unit. In any other case, proper data flows to the selected functional units and the instruction can be executed.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
IDEX Entry/IR Entry	Issuing Unit	Address/Data/Control	Aggregate	Entry from Issue Register, or ID-EX Inter-stage Buffer if bypass has been performed
ISSUING_ack	Issuing Unit	Control	1	If asserted, a new entry has been issued
shifter_operand	Shifter	Data	32	Shifter Operand value
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
data_AU	Arithmetic-Logic Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
is_AU	Arithmetic-Logic Unit	Control	1	A new instructions has to be executed by AU
data_MU	Multiply Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
is_MU	Multiply Unit	Control	1	A new instructions has to be executed by MU
data_BU	Branch Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
is_BU	Branch Unit	Control	1	A new instructions has to be executed by BU
data_LSU	Load/Store Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
is_LSU	Load/Store Unit	Control	1	A new instructions has to be executed by LSU
SWI	Hazard Unit	Control	1	If asserted a software interrupt instruction has been issued

Table 20: Routing Unit signals

10 Functional Units

10.1 Introduction

Each functional unit receives ad-hoc data and control signals to execute different instructions and support different addressing modes.

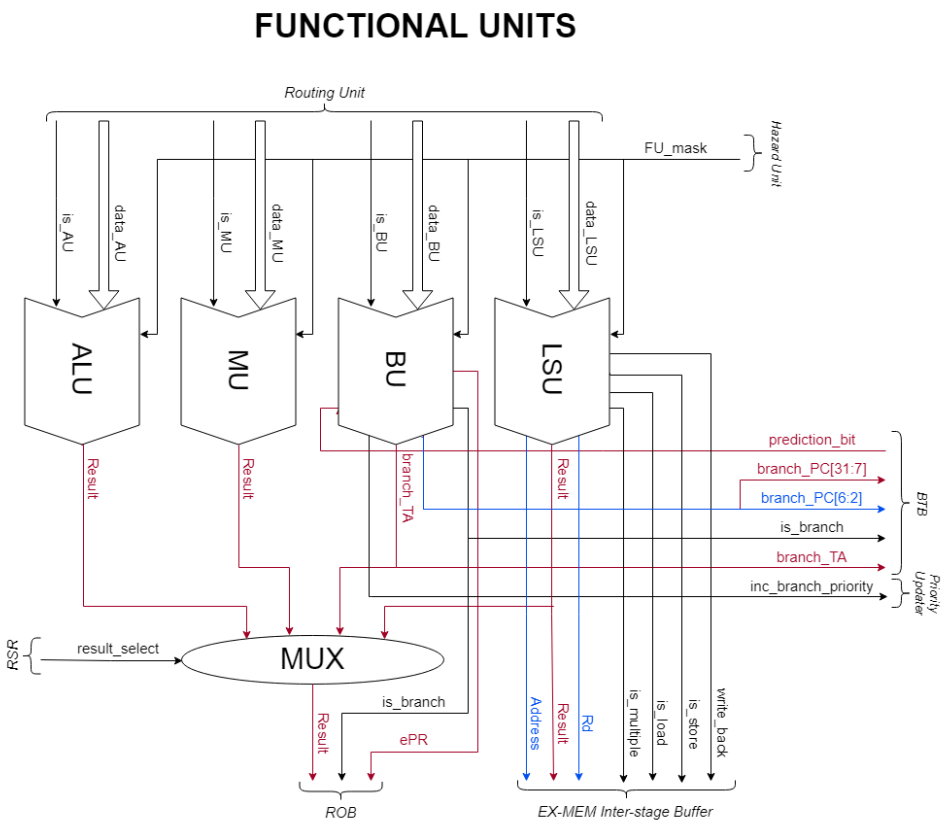

Figure 13: Functional Units

10.2 Signals

Result commit is managed by RSR with the signal result_select sent to the multiplexer adopted for result selection among the four functional units.

In addition, Branch Unit provides the update of both BTB's prediction bits and target address, and to notify wrong predictions to ROB. ePR is generated from prediction_bit depending of the branch instruction's result.

With Multiple instructions, Load/Store Unit performs direct access to EX/MEM Inter-stage Buffer. In this case, a symbolic entry is stored into ROB to let other instructions to be executed while LSU operates with memory. At the occurrence of a D-Cache miss, during the execution of a Load/Store Multiple instruction, each functional unit is stalled since RSR must be stalled too and to avoid issues on result commit.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
prediction_bit	BTB	Data	1	Most significant prediction bit from BTB entry of current Branch instruction
result_select	RSR	Control	2	Control signal to manage result selector multiplexer
data_AU	Routing Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
data_MU	Routing Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
data_BU	Routing Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
data_LSU	Routing Unit	Data/Control	Aggregate	Data and Control signals to execute instructions
is_AU	Routing Unit	Control	1	A new instructions has to be executed by AU
is_MU	Routing Unit	Control	1	A new instructions has to be executed by MU
is_BU	Routing Unit	Control	1	A new instructions has to be executed by BU
is_LSU	Routing Unit	Control	1	A new instructions has to be executed by LSU
FU_mask	Full Issue Register and Miss Handler	Control	1	If asserted, a miss occurred on a multiple instruction and LSU has to be stalled
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
branch_PC[31:7]	BTB	Data	5	If asserted, BTB writing is allowed
branch_PC[6:2]	BTB	Address	25	Program Counter's 5 less significant bits to access BTB
branch_TA	BTB	Address	32	Program Counter's 25 most significant bits to detect aliasing
inc_branch_priority	Priority Updater	Control	1	If asserted, old prediction bits have to be incremented by one
Result	ROB	Data	32	Result coming from one of the Functional Units to be stored into ROB
ePR	ROB	Data	1	Wrong prediction signal from Branch Unit to ROB
is_branch	ROB	Control	1	If asserted, a Branch instruction has endend its execution and ePR has to be stored
is_store	EX-MEM Inter-stage Buffer	Control	1	If asserted, the instruction is a store and the address field is valid
is_load	EX-MEM Inter-stage Buffer	Control	1	If asserted, the instruction is a load and the address field is valid
is_multiple	EX-MEM Inter-stage Buffer	Control	1	If asserted, the instruction is a load/store multiple
write_back	EX-MEM Inter-stage Buffer	Control	1	If asserted, the instruction performs write back
Rd	EX-MEM Inter-stage Buffer	Address	4	Load instruction's destination address
Result	EX-MEM Inter-stage Buffer	Data	32	Value to load/store
Address	EX-MEM Inter-stage Buffer	Address	32	Load/Store instructions' memory address

Table 21: Functional Units signals

11 ROB

11.1 Introduction

ROB size is fixed at 16 entries, due to the maximum latency introduced by Load/Store Multiple instructions of 16 clock cycles.

ROB provides different functionalities:

- Re-Order instructions
- Update the forwarding table's status
- Synchronize conditional executions and instructions' issuing process
- Support the Hazard Unit

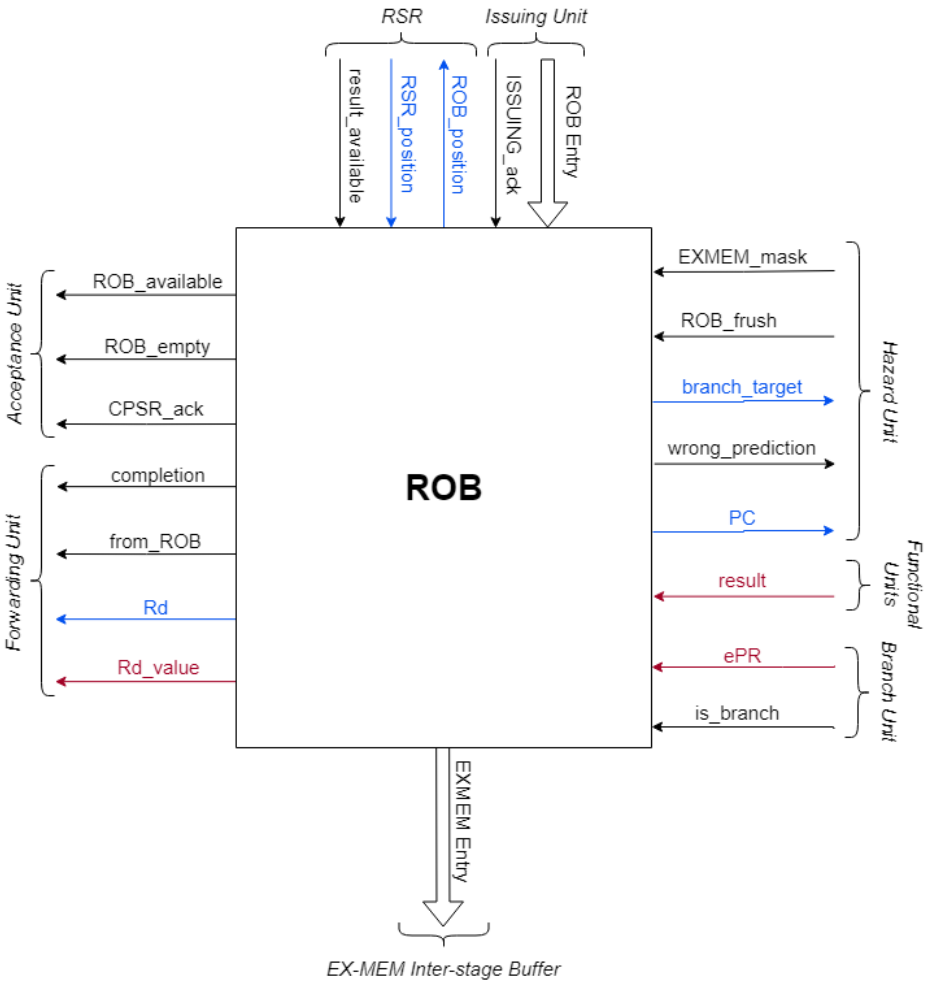

Figure 14: ROB

11.2 Entry fields

ROB stores data and control information in its entries, described below:

ENTRY	FROM	TYPE	CARDINALITY	DESCRIPTION
ePR	Branch Unit	Control	1	If asserted, a wrong prediction occurred
Address	Load/Store Unit	Address	32	Load/Store instructions' memory address
Result	Functional Units	Data	32	Result value, nextPC for branch instructions
C	Functional Units/Issuing Unit	Control	1	If asserted, execution has completed and the result has been stored into ROB
is_set_conditional	Issuing Unit	Control	1	If asserted, the istruction sets the conditional flags
is_load	Issuing Unit	Control	1	If asserted, the instruction is a load and the address field is valid
is_store	Issuing Unit	Control	1	If asserted, the instruction is a store and the address field is valid
is_branch_with_link	Issuing Unit	Control	1	If asserted, PC + 4 has to be written back to r14
write_back	Issuing Unit	Control	1	If asserted, the instruction preforms write back
Rd	Issuing Unit	Address	4	Destination register, r14 for branch with link
PC	Issuing Unit	Data	32	Program Counter

Table 22: ROB entry fields

11.3 Signals

The status of Forwarding table is updated on C bit commutation or whenever new instructions or results are stored into ROB.

Hazard Unit is notified about wrong predictions and the correct Program Counter is provided to restore correct status.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
result_available	RSR	Control	1	If asserted, a new result is available from the functional units
RSR_position	RSR	Address	4	ROB position to access for storing result
ISSUING_ack	Issuing Unit	Control	1	If asserted, a new entry has been issued
ROB Entry	Issuing Unit	Address/Data/Control	Aggregate	Entry to be stored into ROB
Result	Functional Units	Data	32	Result coming from one of the Functional Units to be stored into ROB
ePR	Branch Unit	Data	1	Wrong prediction data from Branch Unit to ROB
is_branch	Branch Unit	Control	1	If asserted, a Branch instruction has endend its execution and ePR has to be stored
EXMEM_mask	Hazard Unit	Control	1	If asserted, EX-MEM Inter-stage Buffer is stalled
ROB_flush	Hazard Unit	Control	1	If asserted, ROB flush is required
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
ROB_position	RSR	Address	4	ROB position pointed by trailer pointer
ROB_available	Acceptance Unit	Control	1	If asserted, ROB is not full and the instruction can be accepted
ROB_empty	Acceptance Unit	Control	1	If asserted, ROB is empty and the instruction can be accepted
CPSR_ack	Acceptance Unit	Control	1	If asserted, CPSR has been updated and conditional execution can be verified
completion	Forwarding Unit	Control	1	Update signal from ROB
from_ROB	Forwarding Unit	Control	1	If asserted, an update from ROB is available
Rd	Forwarding Unit	Address	4	Data processing instructions' destination register, or Load/Store instructions' destination/source register
Rd_value	Forwarding Unit	Data	32	Value from ROB to update forwarding table's registers
branch_target	Hazard Unit	Address	32	Correct target to restore of the Branch Instruction causing the wrong prediction
wrong_prediction	Hazard Unit	Control	1	If asserted, a wrong prediction occurred and the pipeline needs to be flushed
PC	Hazard Unit	Address	32	Program Counter to restore after context switch
EXMEM Entry	EX-MEM Inter-stage Buffer	Address/Data/Control	Aggregate	Entry to store into EX-MEM Inter-stage Buffer

Table 23: ROB signals

Chapter 4: Forwarding Unit

This chapter introduces the Forwarding Unit implementation and contains the following sections:

- *Introduction* on page 34
- *Entry fields* on page 35
- *Signals* on page 36

1 Introduction

Forwarding Unit provides:

- Operand Forwarding
- Dependency Detection

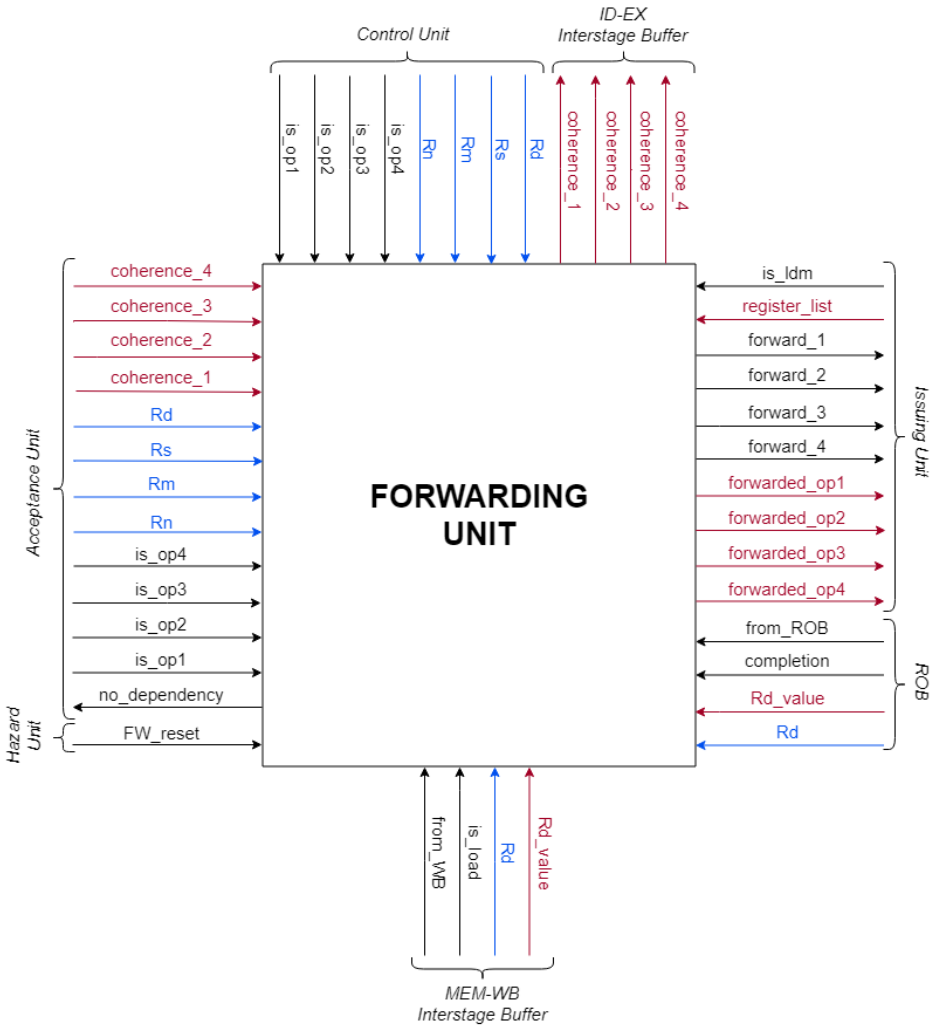

Figure 15: Forwarding Unit

2 Entry fields

The 16-entries forwarding table identifies the current status of each register. Each entry has 2 status bit and the current value of the register.

ENTRY FIELDS	FROM	TYPE	CARDINALITY	DESCRIPTION
Dependency	Issuing Unit/ROB	Data	1	If asserted, a dependency occurs, and the register is being updated
Coherence	ROB/MEM-WB Inter-stage Buffer	Data	1	If asserted, current status of Register Bank is coherent with the status of the forwarding table
Value	ROB/MEM-WB Inter-stage Buffer	Data	32	Updated register's value, forwarded from ROB or MEM/WB Inter-stage Buffer

Table 24: Forwarding Unit table entry fields

3 Signals

The update of the status is managed by Issuing Unit, ROB and MEM/WB Inter-stage Buffer, and respectively:

- ROB and MEM/WB access the table directly by register address Rd
- Issuing Unit accesses only when a Load Multiple is being issued, by decoding the register list of the instruction and updating Dependency bits of each register involved in the load multiple operation

Differently, Acceptance Unit accesses the forwarding table through four control signals and up to three addresses of the registers.

Instruction		Table				
Coherence	Value	Dependency	Coherence	Value	Result	Description
1	Updated	0	1	Updated	No Forward	Instruction got an updated value, no update occurred for the table's one.
1	Old	1	0	Old	No Issue	Instruction got an updated value, but table's one is being updated.
1	Old	0	0	Updated	Forward	Instruction got an updated value, but then table's one update occurred.
0	Old	0	1	Updated	Forward	Instruction got an old value, but table's one is updated.
0	Old	1	0	Old	No Issue	Instruction got an old value, but table's one is being updated.
0	Old	0	0	Updated	Forward	Instruction got an old value, but table's one is updated.

Table 25: Forwarding Unit table state update logic

The status of the table may change after Decode stage. In order to detect any forward needed, Forwarding Unit performs logical NAND of coherence bits of the instructions from Acceptance Unit and coherence bits of the table. Moreover, when Dependency is asserted, forward is not possible and the instruction cannot be issued.

Once forward need is detected, Forwarding Unit provides updated operand to Issuing Unit.

ENTRY	FROM	TYPE	CARDINALITY	DESCRIPTION
Dependency	Issuing Unit/ROB	Data	1	If asserted, a dependency occurs, and the register is being updated
Coherence	ROB/MEM-WB Inter-stage Buffer	Data	1	If asserted, current status of Register Bank is coherent with the status of the forwarding table
Value	ROB/MEM-WB Inter-stage Buffer	Data	32	Updated register's value, forwarded from ROB or MEM/WB Inter-stage Buffer
INPUTS	**FROM**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
is_op1	Control Unit/Acceptance Unit	Control	1	If asserted, Rn is from register
is_op2	Control Unit/Acceptance Unit	Control	1	If asserted, Rm is from register
is_op3	Control Unit/Acceptance Unit	Control	1	If asserted, Rs is from register
is_op4	Control Unit/Acceptance Unit	Control	1	If asserted, Rd is from register
Rn	Control Unit/Acceptance Unit	Address	4	Data processing instructions' 1st operand register, or Load/Store instructions' base register
Rm	Control Unit/Acceptance Unit	Address	4	Data processing instructions' 2nd operand register, or Load/Store register offset instructions' offset register
Rs	Control Unit/Acceptance Unit	Address	4	Data processing register shift instructions' register shift amount
Rd	Control Unit/Acceptance Unit/ROB/MEM-WB Inter-stage Buffer	Address	4	Data processing instructions' destination register, or Load/Store instructions' destination/source register
coherence_1	Acceptance Unit	Data	1	Rn register coherence bit of the instruction
coherence_2	Acceptance Unit	Data	1	Rm register coherence bit of the instruction
coherence_3	Acceptance Unit	Data	1	Rs register coherence bit of the instruction
coherence_4	Acceptance Unit	Data	1	Rd register coherence bit of the instruction
is_ldm	Issuing Unit	Control	1	If asserted, the instruction is a load multiple and register list field is valid
register_list	Issuing Unit	Data	16	Register list of load multiple instruction
completion	ROB	Control	1	Update signal from ROB
from_ROB	ROB	Control	1	If asserted, an update from ROB is available
Rd_value	ROB/MEM-WB Inter-stage Buffer	Data	32	Value from ROB or WB to update forwarding table's registers
FW_reset	Hazard Unit	Control	1	If asserted, a reset of the forwarding table is required
is_load	MEM-WB Inter-stage Buffer	Control	1	If asserted, Rd_value from MEM/WB comes from a load instruction
from_WB	MEM-WB Inter-stage Buffer	Control	1	If asserted, an update from MEM/WB is available
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
coherence_1	ID-EX Inter-stage Buffer	Data	1	Coherence Bit associated to operand 1
coherence_2	ID-EX Inter-stage Buffer	Data	1	Coherence Bit associated to operand 2
coherence_3	ID-EX Inter-stage Buffer	Data	1	Coherence Bit associated to operand 3
coherence_4	ID-EX Inter-stage Buffer	Data	1	Coherence Bit associated to operand 4
no_dependency	Acceptance Unit	Control	1	If asserted, no dependency occurred for the current instruction in Acceptance Unit
forward_1	Issuing Unit	Control	1	If asserted, operand 1 has been forwarded from forwarding table
forward_2	Issuing Unit	Control	1	If asserted, operand 2 has been forwarded from forwarding table
forward_3	Issuing Unit	Control	1	If asserted, operand 3 has been forwarded from forwarding table
forward_4	Issuing Unit	Control	1	If asserted, operand 4 has been forwarded from forwarding table
forwarded_op1	Issuing Unit	Data	32	Operand 1 from forwarding table
forwarded_op2	Issuing Unit	Data	32	Operand 2 from forwarding table
forwarded_op3	Issuing Unit	Data	32	Operand 3 from forwarding table
forwarded_op4	Issuing Unit	Data	32	Operand 4 from forwarding table

Table 26: Forwarding Unit signals

Chapter 5: Hazard Unit

This chapter introduces the Forwarding Unit implementation and contains the following sections:

- *Introduction* on page 39
- *Behaviour and signals* on page 40

1 Introduction

The Hazard unit is expected to handle extraordinary events occurring during the execution of instructions in the pipeline, therefore providing stall or flush operation as far as it is needed, that is to say:

- Exceptions and interrupts
- Wrong predictions on branch instructions
- Cache misses and full buffers in memory occurrences

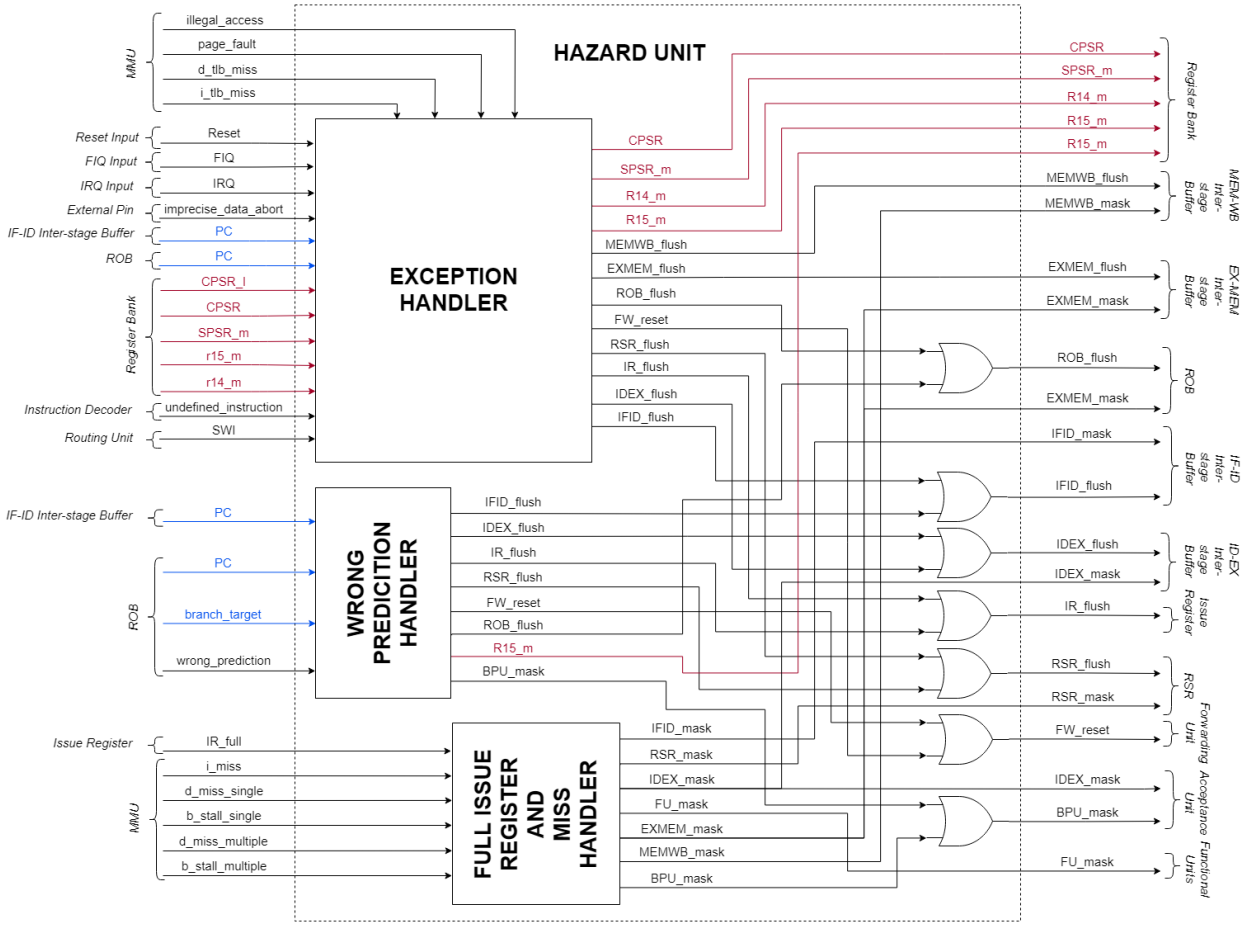

Figure 16: Hazard Unit

2 Behaviour and signals

In case of wrong prediction, the Wrong prediction handler must supply the update of program counter taking the correct address out of the ROB, the stall of BPU, the flush of Forwarding Unit, Issue Register, ROB, RSR and the related buffers.

Whether the issue register is detected to be full, the Full issue register and miss handler must stall BPU and related buffers; whereas in case of single instruction issues from memory (which means cache miss or buffer filling) the related buffers are stalled; eventually, for multiple instructions the Functional unit and RSR are stalled. Indeed, since cache miss does not trigger an exception, the unit must handle the consequent operations to preserve the coherence of the pipeline.

The Exception Handler must manage the occurrence of exceptions and interrupts which may happen in each stage of the pipeline, considering the nature of simultaneity. In accordance with the ARM reference manual, once the exception has been detected, the routine generally expects to:

- Save the current program counter in the link register
- Disable interrupts which must not overstep the current exception in case of simultaneity, properly setting the mask (that is CPSR)
- Change the mode of operation of the processor accordingly
- Access the vector table of the ARM to pick up the correct address of ISR
- Fetch the first instruction of ISR to start the software handling of the exception
- After the software has ended, go back to the interrupted instruction, properly recovering the state of the process and the program counter

In order to quicken the exception handling operations, the vectorized management of interruptions is decided to be implemented, already provided by the ARM reference manual.

To this purpose, the Exception Handler includes:

- Interrupt Request Register to significantly measure the occurrence of the exceptions
- A Programmable Interrupt Controller to organize the priority of exceptions to guarantee a deterministic service time of the interruption and obtain immediately the address to be written in the program counter
- An Interrupt Handler to manage the context switching
- A logic of update of the pipeline coherent with the nature of the exception served.

The set of exceptions considered is that of the ARM Reference Manual, keeping the provided Vector Table and list of priorities. Data Abort exception includes Illegal

accesses, Tlb misses on data cache and page faults whereas Prefetch Abort involves Tlb miss on instruction cache.

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
Reset	Reset Input	Control	1	If asserted, a reset interrupt has occurred
FIQ	FIQ Input	Control	1	If asserted, a request of fast interrupt has occurred
IRQ	IRQ Input	Control	1	If asserted, a request of interrupt has occurred
imprecise_data_abort	External Pin	Control	1	If asserted, an imprecise data abort has occurred
PC	IF-ID Inter-stage Buffer/ROB	Data	32	Program Counter to restore when returning from an exception
CPSR_I	Register Bank	Data	1	IRQ mask read on the handling of exceptions' priorities
CPSR	Register Bank	Data	32	CPSR to preserve before the ISR call
SPSR_m	Register Bank	Data	32	CPSR stored into SPSR of the processor mode m the exception is taken to
r14_m	Register Bank	Data	32	PC stored into Link Register of the processor mode m the exception is taken to
r15_m	Register Bank	Data	32	PC to store into Link Register of the processor mode m the exception is taken to
undefined_instruction	Instruction Decoder	Control	1	If asserted, an undefined instruction has been decoded
SWI	Routing Unit	Control	1	If asserted, a Software Interrupt instruction has been issued
i_tlb_miss	MMU	Control	1	If asserted, call to OS is required (Prefetch Abort)
d_tlb_miss	MMU	Control	1	If asserted, call to OS is required (Data Abort)
page_fault	MMU	Control	1	If asserted, call to OS is required (Data Abort)
illegal_access	MMU	Control	1	If asserted, call to OS is required (Data Abort)
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
IFID_flush	IF-ID Inter-stage Buffer	Control	1	If asserted, reset of IF/ID Inter-stage Buffer is required
CPSR	Register Bank	Data	32	CPSR updated for the processor mode m when entering an exception or restored from SPSR_m when returning from an exception
SPSR_m	Register Bank	Data	32	SPSR of the processor mode m updated from CPSR
r14_m	Register Bank	Data	32	PC stored into Link Register of the processor mode m the exception is taken to
R15_m	Register Bank	Data	32	Vector Table address stored into R15 of the processor mode m or PC restored when returning from an exception
IDEX_flush	ID-EX Inter-stage Buffer	Control	1	If asserted, reset of ID/EX Inter-stage Buffer is required
IR_flush	Issue Register	Control	1	If asserted, reset of Issue Register is required
RSR_flush	RSR	Control	1	If asserted, reset of RSR is required
FW_reset	Forwarding Unit	Control	1	If asserted, reset of the forwarding table is required
ROB_flush	ROB	Control	1	If asserted, reset of ROB is required
EXMEM_flush	EX-MEM Inter-stage Buffer	Control	1	If asserted, reset of EX/MEM Inter-stage Buffer is required
MEMWB_flush	MEM-WB Inter-stage Buffer	Control	1	If asserted, reset of MEM/WB Inter-stage Buffer is required

Table 27: Exception Handler signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
PC	IF-ID Inter-stage Buffer/ROB	Data	32	Program Counter to restore when returning from an exception
branch_target	ROB	Address	32	Correct target to restore of the Branch Instruction causing the wrong prediction
wrong_prediction	ROB	Control	1	If asserted, a wrong prediction occurred and the pipeline needs to be flushed
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
BPU_mask	BPU	Control	1	If asserted, BPU has to be stalled
IFID_flush	IF-ID Inter-stage Buffer	Control	1	If asserted, reset of IF/ID Inter-stage Buffer is required
R15_m	Register Bank	Data	32	PC restored from ROB
IDEX_flush	ID-EX Inter-stage Buffer	Control	1	If asserted, reset of ID/EX Inter-stage Buffer is required
IR_flush	Issue Register	Control	1	If asserted, reset of Issue Register is required
RSR_flush	RSR	Control	1	If asserted, reset of RSR is required
FW_reset	Forwarding Unit	Control	1	If asserted, reset of the forwarding table is required
ROB_flush	ROB	Control	1	If asserted, reset of ROB is required

Table 28: Wrong Prediction Handler signals

INPUTS	FROM	TYPE	CARDINALITY	DESCRIPTION
IR_full	Issue Register	Control	1	If asserted, Issue Register is full, fetch and decode stage stall is required
i_miss	MMU	Control	1	If asserted, stall required for IF-ID Inter-stage Buffer
d_miss_single	MMU	Control	1	If asserted, stall required for EX-MEM Inter-stage Buffer
d_miss_multiple	MMU	Control	1	If asserted, stall required for Functional Units, RSR and EX-MEM
b_stall_single	MMU	Control	1	If asserted, stall required for EX-MEM Inter-stage Buffer
b_stall_multiple	MMU	Control	1	If asserted, stall required for Functional Units, RSR and EX-MEM
OUTPUTS	**TO**	**TYPE**	**CARDINALITY**	**DESCRIPTION**
BPU_mask	BPU	Control	1	If asserted, BPU has to be stalled
IFID_mask	IF-ID Inter-stage Buffer	Control	1	If asserted, IF/ID Inter-stage Buffer has to be stalled
IDEX_mask	ID-EX Inter-stage Buffer/Acceptance Unit	Control	1	If asserted, ID/EX Inter-stage Buffer has to be stalled
RSR_mask	RSR	Control	1	If asserted, RSR has to be stalled
FU_mask	Functional Units	Control	1	If asserted, Functional Units have to be stalled
EXMEM_mask	ROB/EX-MEM Inter-stage Buffer	Control	1	If asserted, EX/MEM Inter-stage Buffer has to be stalled
MEMWB_mask	MEM-WB Inter-stage Buffer	Control	1	If asserted, MEM/WB Inter-stage Buffer has to be stalled

Table 29: Full Issue Register and Miss Handler signals

Chapter 6: Resources

This chapter introduces the resources involved in the design and contains the following sections:

- *Resources allocation* on page 43
- *Resources timesheet* on page 43

Resources allocation

Resources per leadership

Leadership	Member
Team Leader	Fernando Manna
CPU Leader	Giovanni Di Prisco
MMU Leader	Edoardo Cossentino
Documentation Leader	Francesco de Pertis

Resources timesheet

Sub-team	Team members	Tot. work hours d1	Tot. work hours d1 review
CPU	Fernando Manna	40	-
	Giovanni Di Pisco	40	-
	Marco Schettini	30	-
	Ilaria Gigi	35	-
	Mario Mupo	30	-
	Romeo Rinaldi	30	-
	Michele Rescigno	30	-
	Ascanio Guglielmelli	30	-
Documentation	Francesco de Pertis	25	5
	Giuseppe Cirillo	25	5