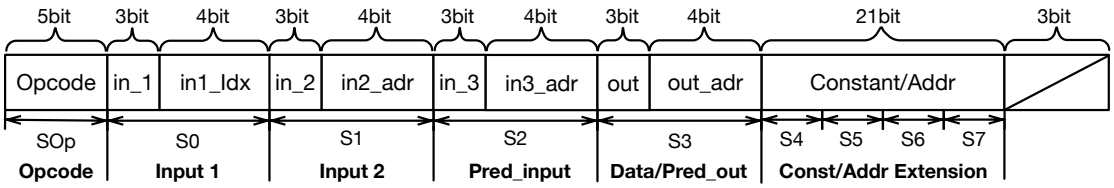


The context architecture

1. Opcode

Opcode	Symbol	Function	Operation	Data_out	Pred_out
0		Nop	T='z	Data_out ='z	Pred_out ='z
1	+	signed addition	T=input1+input2+Pred_input	Data_out =T[31:0]	Pred_out =T[31]
2	-	signed subtraction	T=input1-input2- Pred_input	Data_out =T[31:0]	Pred_out =T[31]
3	*	signed multiplication	T=input1 x input2	Data_out =T[31:0]	Pred_out =T[0]
4	&	and	T=input1 & input2 & {31{1'b1}, Pred_input }	Data_out =T[31:0]	Pred_out =&T[31:0]
5	|	or	T=input1 | input2 | {31{1'b0}, Pred_input }	Data_out =T[31:0]	Pred_out =|T[31:0]
6	^	xor	T=input1 ^ input2 ^ {31{1'b0}, Pred_input }	Data_out =T[31:0]	Pred_out =^T[31:0]
7	~	abs subtraction	T=|input1-input2- Pred_input |	Data_out =T[31:0]	Pred_out =(T!=0)
8	?	selection	T= Pred_input? input1 : input2	Data_out =T[31:0]	Pred_out =in3
9	u<	left logical shift	T={input1, Pred_input }<<input2	Data_out =T[31:0]	Pred_out =T[32]
10	u>	right logical shift	T={input1, Pred_input }>>input2	Data_out =T[31:0]	Pred_out =T[-1]
11	>	right	T={input2{input1[31]},input1>>in	Data_out	Pred_ou

		arithmetic shift	put2}	=T[31:0]	t =T[31]
12	u+	unsigned addition	T=input1+input2+ Pred_input	Data_out =T[31:0]	Pred_out =T[32]
13	u-	unsigned subtraction	T=input1-input2- Pred_input	Data_out =T[31:0]	Pred_out =T[32]
14	Comp>	Compare whether A>B	If A>B, Pred_out = 1	Data_out ='z	Pred_out =T[32]
15	Comp>=	Compare whether A>=B	If A>=B, Pred_out = 1	Data_out ='z	Pred_out =T[32]
16	Comp ==	Compare whether A=B	If A=B, Pred_out = 1	Data_out ='z	Pred_out =T[32]
17	load				
18	store				

2. Input

Load context

Operand	Index	From	Source	Value
input1	0	local register	LR[in1_adr]	SM[input1]
	1	the other PE	neighbor PE[in1_adr]'s Data_out	SM[input1]
	2	global register	GR[in1_adr]	SM[input1]
	3	immediate	{constant, in1_adr}	SM[input1]
	4	shared memory	SM[constant]	SM[input1]
input2/Pred_input	-	-	-	-

Arithmetic and logic operation context

Operand	Index	Source	Value
input1/ input2	0	local register	LR[in1_adr/in2_adr]
	1	the other PE	neighbor PE[in1_adr/in2_adr]'s Data_out

	2	global register	GR[in1_adr/in2_adr]
	3	immediate	constant
Pred_input	0	local fine register	LFR[Pred_input_adr]
	1	the other PE	neighbor PE[Pred_input_adr]'s Pred_out
	2	global fine register	GFR[Pred_input_adr]

Store context

Operand	Index	Source	Value
input1	0	local register	LR[in1_adr], 0~16
	1	the other PE	neighbor PE[in1_adr]'s Data_out, up to 16PEs
	2	global register	GR[in1_adr] 0~16
	3	immediate	constant,
input2/ Pred_input	-	-	-

3. Output

Store context

Operand	out	Buffer type	Value	Destination
Data_out	0	local register	LR[out_adr]	SM[Data_out]
	1	the other PE	neighbor PE[out_adr]'s Data_out	SM[Data_out]
	2	global register	GR[out_adr]	SM[Data_out]
	3	immediate	{constant, out_adr }	SM[Data_out]
	4	shared memory	SM[constant]	SM[Data_out]
Pred_out	-	-	-	

Arithmetic and logic operation context

Operand	out	Buffer type	Destination
Data_out	0	output register	output register
	1	local register	LR[out_adr]
	2	global register	GR[out_adr]

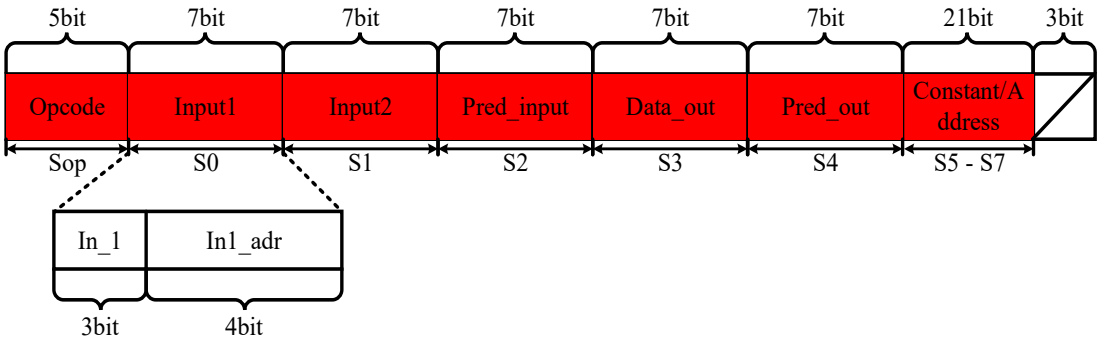
	3	self input	self input2
Pred_out	0	output register	output fine register
	1	local fine register	LFR[out_adr]
	2	global fine register	GFR[out_adr]
	3	Self input	self input 3

Load context

Operand	Index	To	Destination
Data_out	0	output register	output register
	1	local register	LR[out_adr]]
	2	global register	GR[out_adr]
	3	self input	self input2
Pred_out	-	-	-

4. Valid Section

Arithmetic and logic operation context



Load/Store context