

PULP PLATFORM Open Source Hardware, the way it should be!

A look inside an Application Processor: CVA6

Gianmarco Ottavi < Gianmarco.ottavi2@unibo.it>





https://www.youtube.com/pulp_platform



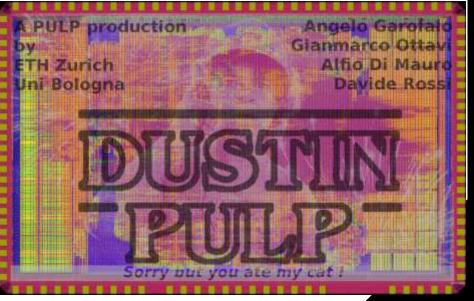


CVA6 is open-source

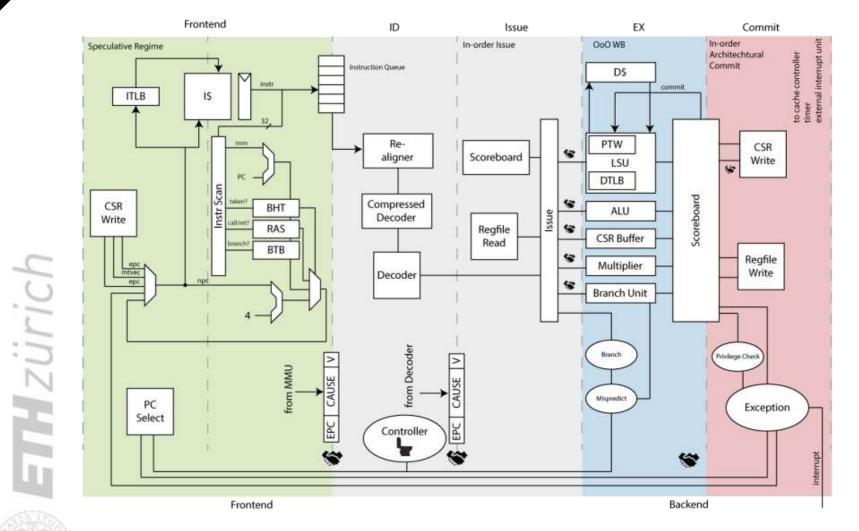
- You can find all the RTL of the Core at:
 - https://github.com/openhwgroup/cva6
- CVA6 is just one of the core born from the PULP Project
 - https://pulp-platform.org/
 - https://github.com/pulp-platform





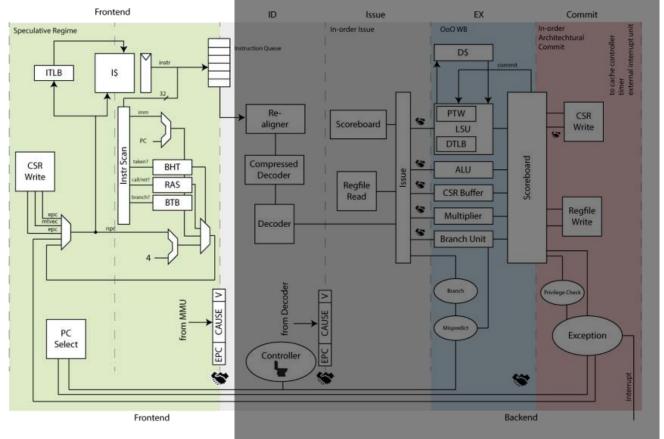


A look Into CVA6



- RISC-V ISA RV64IMAC(F)
- M,S,U Privilege level Spec.
 - Can run linux-like OS (Application class processor).
- In-Order Single Issue with 6 stages of Pipeline

A look Into CVA6: Frontend



PC Generation:

- Sequential Fetch
- Miss Prediction Recovery
- Exception
- Debug
- CSR

Instruction Fetch:

Pre-Decoding for Branch Prediction

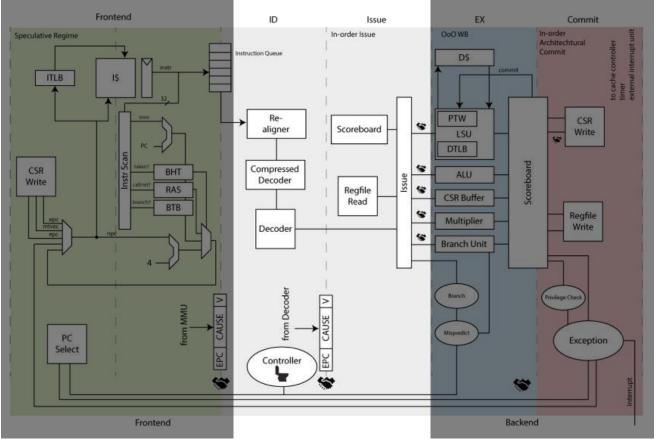
Instruction Cache

- Virtually Indexed and Physically Tagged
- Pipelined to mitigate long propagation delays of memory macros

Frontend Decoupled with backend via Instruction queue

ETH zürich

A look Into CVA6: Backend



Decode Stage:

- Compressed decoder
- Decoder

Issue Stage:

Renaming

Lightweight 1-bit renaming scheme

Operand Read

Instruction Are issued if Operands are ready

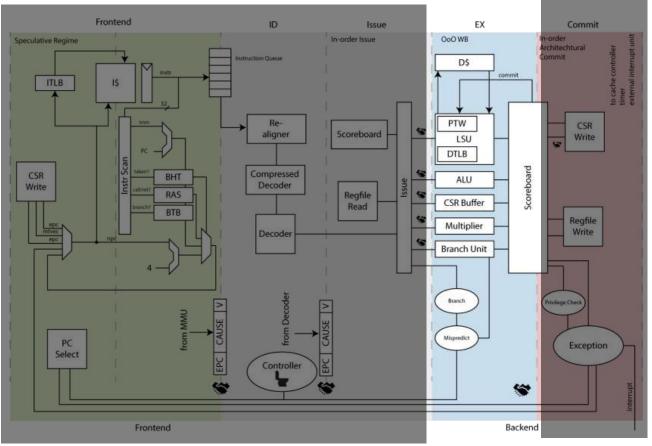
Scoreboarding

- Mixed Scoreboard with Reorder Buffer
- Issued instruction are tracked for data Hazards
- Allow Out-Of-Order completion

ETH zürich

F. Zaruba, L. Benini -- DOI: 10.1109/TVLSI.2019.2926114

A look Into CVA6: Backend



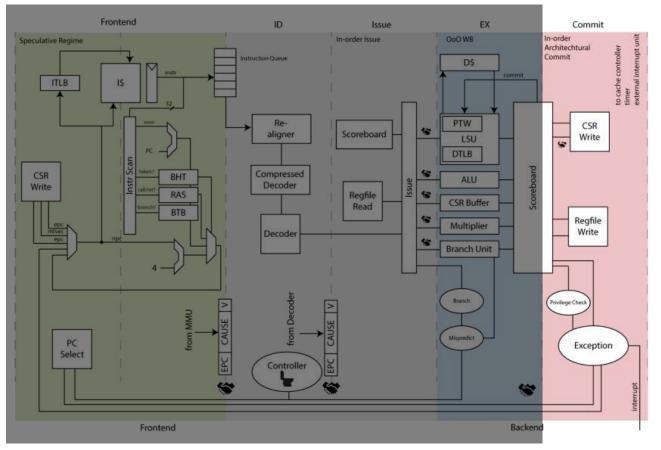
Ex Stage:ALUFPU

- Multiplier/Integer Div
 - 2 Stage/Iterative
- LSU with 3 Master Ports
 - Load
 - Store
 - PTW
- Data Cache 2 Stages Pipeline

Branch Unit

EHZürich

A look Into CVA6: Backend

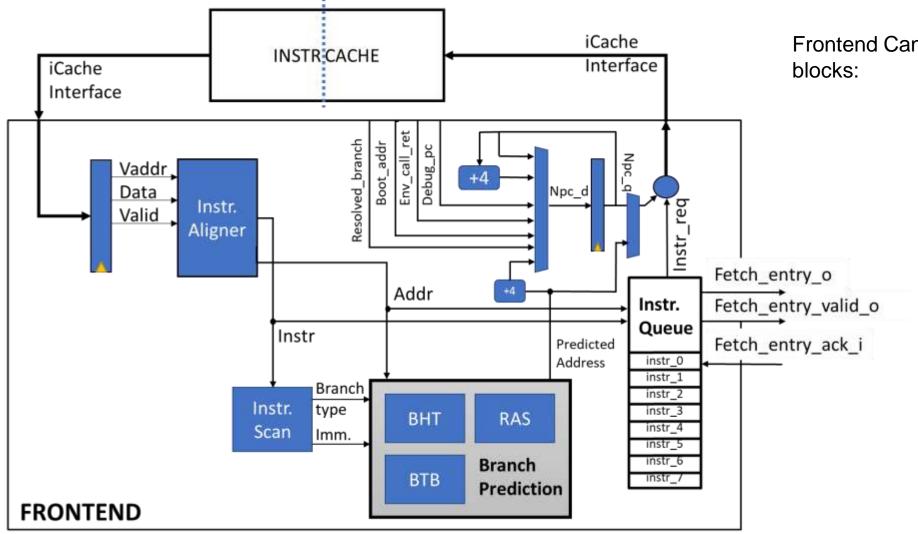


Commit Stage

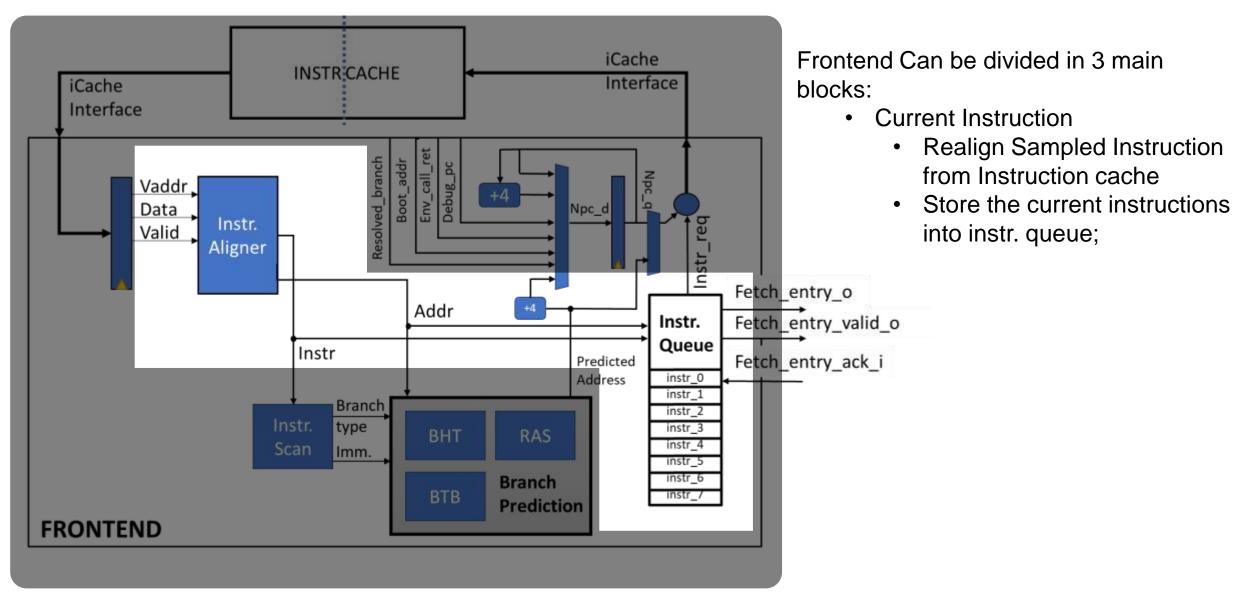
- Commit Completed instruction In-Order
 - Register File
 - Main memory

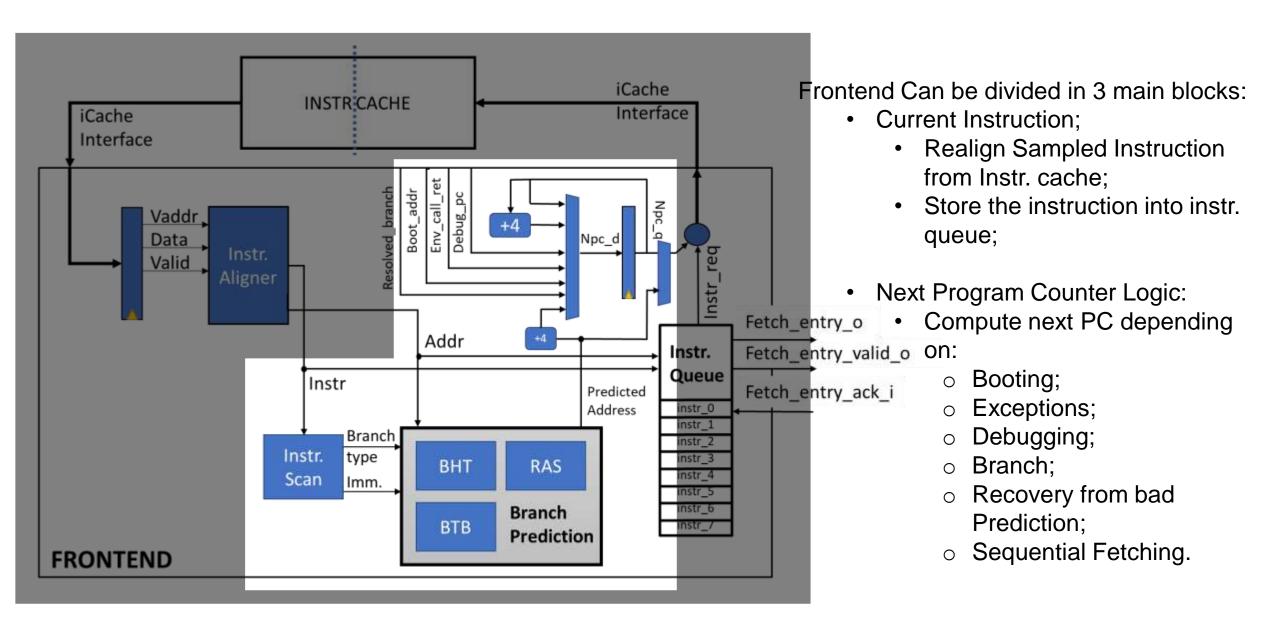
External Interface to main memory AXI4 Compliant

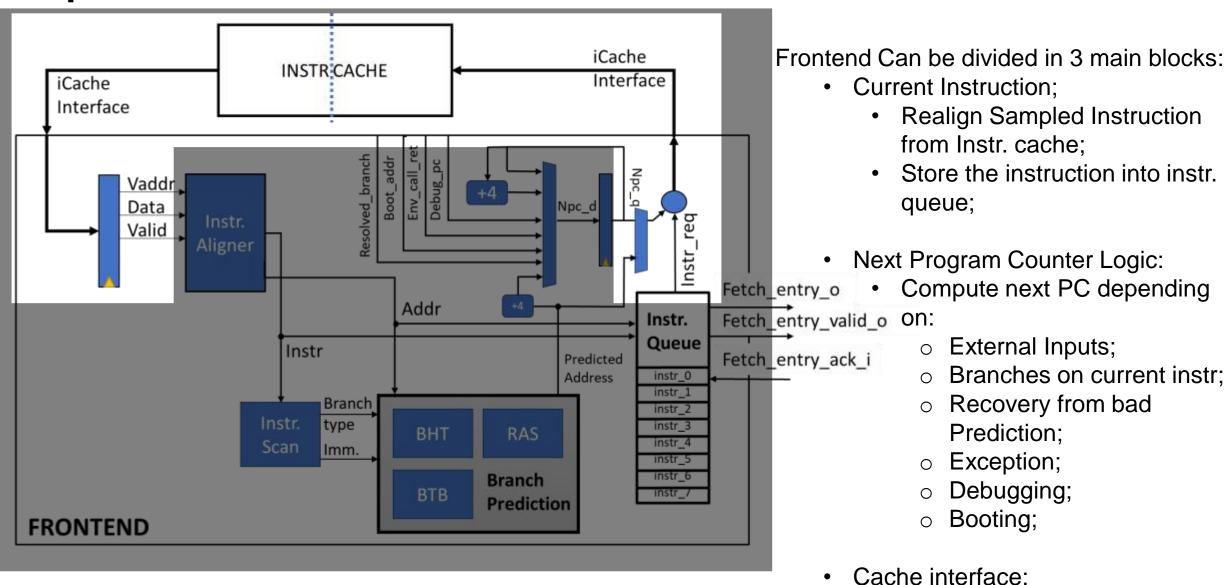
Entzürich



Frontend Can be divided in 3 main blocks:







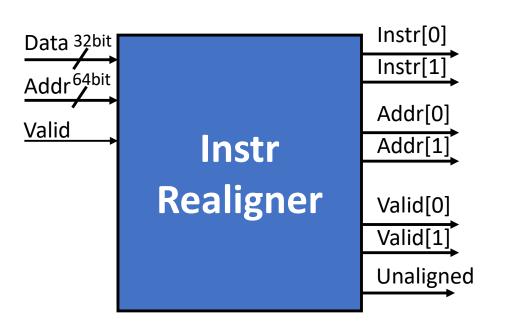
- Request Side;
- Response Side;

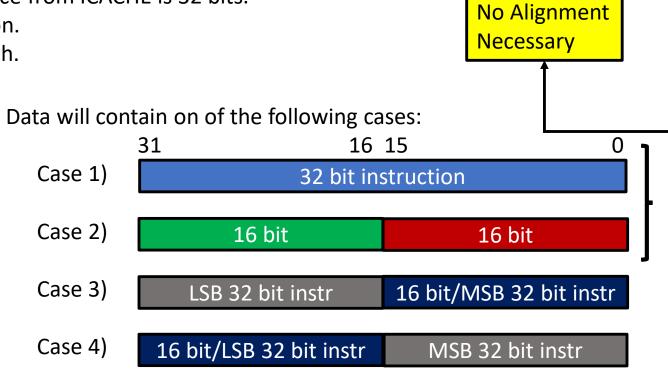
Why do we need an istruction realigner?

CVA6 supports Compressed Instructions and data interface from ICACHE is 32 bits.

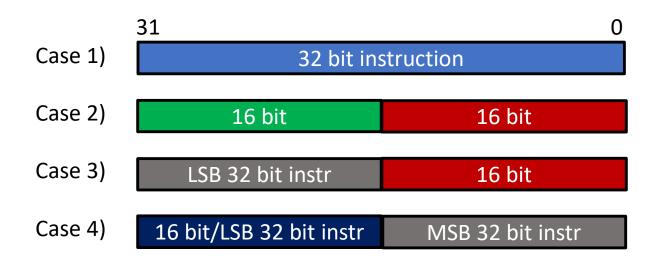
٠

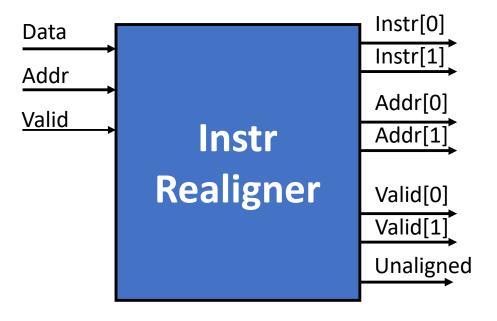
- Compressed Instructions ~70% of total instruction.
- On 32bit fetch width equals to ~1.5 instr per fetch.



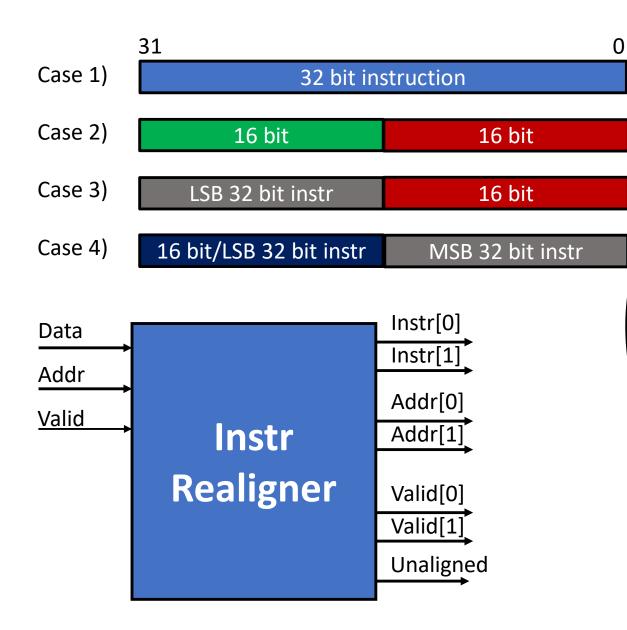


- Addr corresponds to the fetch address of the presented data.
- Non Compressed instruction have the first 2 bits = 1 Check bits [1:0] and [17:16]
- Valid tells if the data that is presented at the interface is usable.



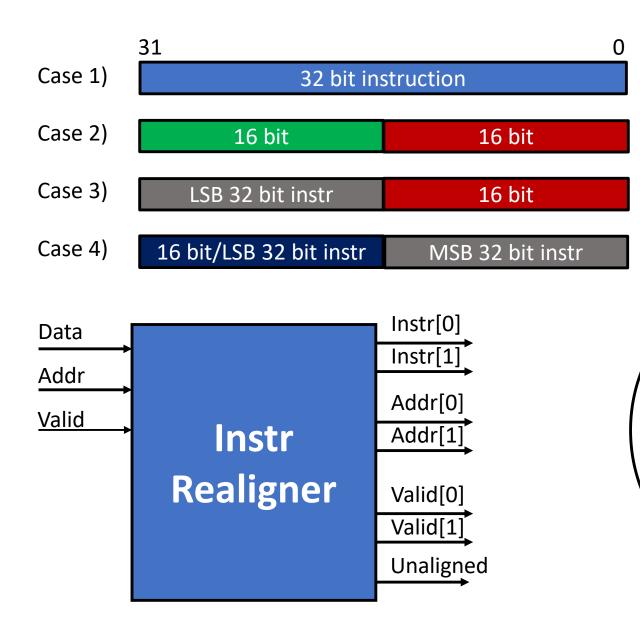


clk_i	1'h1		
Input Interface			
🖬 🔷 icache_data_q	32'h0012d293	830500c7	0012d293 8563
🖬 🔷 icache_vaddr_q	64'h00000000800001f0	00000008	0000000080000 [0000
🔷 icache_valid_q	1'h1		
Output Intf	-		
🗖 🔶 instr	32'h0000000 32'h0012	00008305	00000000 0012 00000
😐 🔷 [1]	32'h0000000	00008305	0000000
e 🔶 [0]	32'h0012D293	00C7F433	0012D293 8563
🗖 🔶 addr	64'h00000000800001f2	00000008	0000000080000 0000
😐 🔶 [1]	64'h00000000800001F2	800000008	0000000080000 0000
😐 🔷 [0]	64'h00000000800001F0	00000008	0000000080000 ¦ 0000
Instruction_valid	2'h1	3	1
	0		
L	1		
🔷 serving_unaligned	1'h0		

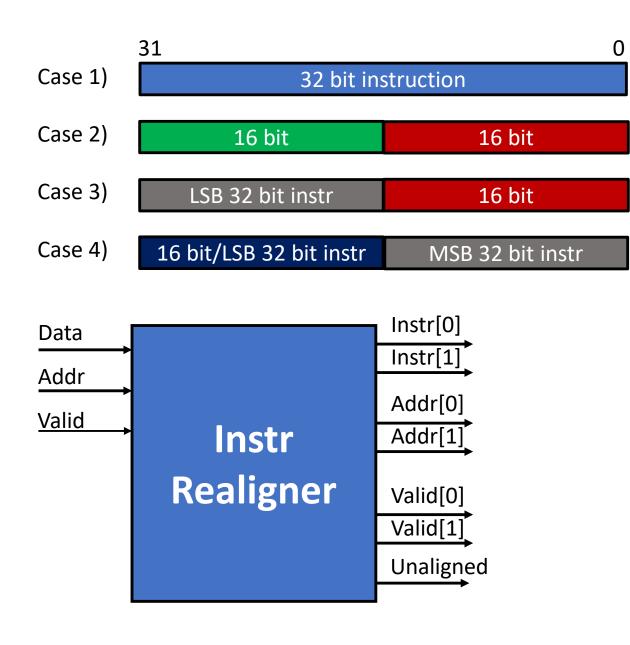


	1'h1			
- Input Interface	-			
icache_data_q	32'h0012d293	830500c7	0012d293	8563
<pre>exactly icache_vaddr_q</pre>	64'h00000000800001f0	00000008	0000000080000	00000
🔶 icache_valid_q	1'h1			
Output Intf				
📥 🔶 instr	32'h00000000 32'h0012	00008305	00000000 0012	00000
= 🔶 [1]	32'h0000000	00008305	0000000	
🖬 🔶 [0]	32'h0012D293	00C7F433	0012D293	8563
🗖 🔶 addr	64'h0000000800001f2	00000008	000000080000	0000
= 🔶 [1]	64'h00000000800001F2	00000008	0000000080000	00000
🖬 🔶 [0]	64'h0000000800001F0	00000008	0000000080000	00000
instruction_valid	2'h1	3	1	
	0		2	
L	1			
🔶 serving_unaligned	1'h0			
> 0x0012 d29	<mark>3</mark> → Data[:	1:0] = 11		

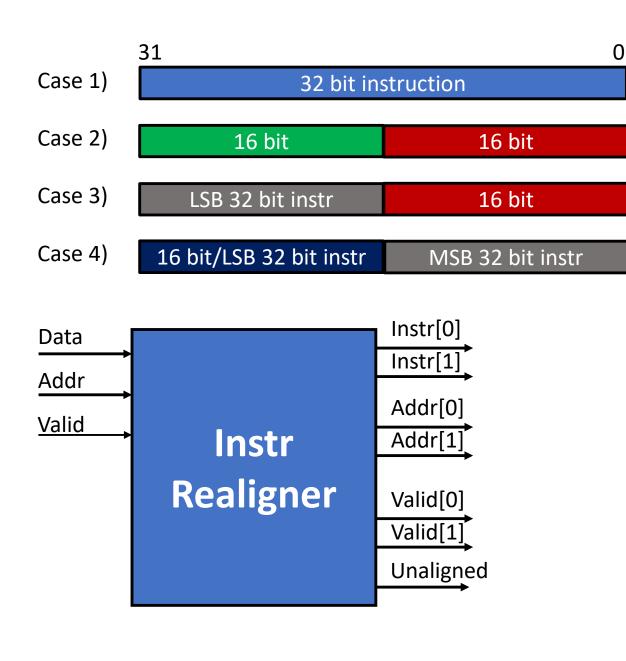
Non Compressed Instructions



≪ clk_i	1'h1			
Input Interface	32'h0012d293	830500c7	0012d293	8563
<pre>pdf icache_vaddr_q</pre>	64'h00000000800001f0	00000008	0000000080000	0000
<pre>icache_valid_q</pre>	1'h1			
Output Intf	32'h00000000 32'h0012	00008305	00000000 0012	10000
• 🔶 [1]	32'h00000000		0000000	
u 💎 [0]	32'h0012D293	00C7F433	0012D293	8563
addr	64'h00000000800001f2		0000000080000	
□	64'h00000000800001F2 64'h0000000800001F0		000000080000	0000
[0]	2'h1	3	1	10000
	0			
L-🔶 [0]	1			
ligned	1'h0			
0x0012 d29	<mark>3</mark> → Data[:	1:0] = Da	ta[17:16] =	= 11
No	n Compressed I	nstructic	ons	
All data is p	out into Instr[0]			



⊶ ∕ clk_i	1'h1		
- Input Interface			
🖬 🔷 icache_data_q	32'h0012d293	830500c7	0012d293 8563
🖬 🔷 icache_vaddr_q	64'h00000000800001f0	00000008	000000080000 0000
🔷 icache_valid_q	1'h1		
- Output Intf	-		
📫 🔶 instr	32'h0000000 32'h0012	00008305	00000000 0012 00000
	32'h0000000	00008305	0000000
😐 🔶 [0]	32'h0012D293	00C7F433	0012D293 8563
🗖 🔶 addr	64'h0000000800001f2	800000008	000000080000 00000
P [1]	64'h0000000800001F2	00000008	000000080000 00000
E 🔶 [0]	64'h00000000800001F0		
a 🔶 instruction_valid	2'h1	3	1
[1]	0		
[0]	1		
serving_unaligned	1'h0		
Serving_unangrieu	TIN		
0x0012 d20	<mark>3</mark> → Data[1·0] = Da	ta[17.16] = 11
		1.0] - 00	
No	n Compressed	Instructio	ons
All data is I	out into Instr[0]		
We have h	oth Addr[0] and	l [1] with	valid addresse



clk_i	1'h1			
- Input Interface				
🖬 🔷 icache_data_q	32'h0012d293	830500c7	0012d293	8563
🖬 🔶 icache_vaddr_q	64'h0000000800001f0	00000008	0000000080000	0000
🔶 icache_valid_q	1'h1			
Output Intf				
🗖 🔷 instr	32'h00000000 32'h0012	00008305	00000000 0012	0000
• 🔶 [1]	32'h0000000	00008305	00000000	
😐 🔶 [0]	32'h0012D293	00C7F433	0012D293	8563
🗖 🔷 addr	64'h0000000800001f2	00000008	0000000080000	0000
😐 🔶 [1]	64'h0000000800001F2	00000008	0000000080000	0000
😐 🔶 [0]	64'h0000000800001F0	00000008	0000000080000	0000
instruction_valid	2'h1	3	1	
	0		l	
(0]	1			
serving_unaligned	1'h0			

0x0012 d29<mark>3</mark> —

Data[1:0] = Data[17:16] = 11

Non Compressed Instructions

All data is put into Instr[0]

We have both Addr[0] and [1] with valid addresses

→ But only valid[0] = 1 → Subsequent block in the Frontend will consider only instr[0]

Case 3)

- 🦑 clk_i	1'h1								
Input Interface	· · · · · · · · · · · · · · · · · · ·								
🖽 🔷 icache_data_q	32'h90e333fd	92c20	90e333fd		e032fe03				
🖬 🔷 icache_vaddr_q	64'h00000008000200	00000	0000000080000200		0000000	80000204			
🛷 icache_valid_q	1'h1								
Output Intf									
🗖 🔶 instr	32'h0000000 32'h90e3.	00009	00000000 90e333fd		0000e032	fe0390e3			000
😐 🔶 [1]	32'h00000000	00009	0000000		0000E03	2			000
🖬 🔶 [0]	32'h90E333FD 🥢	00E4	90E333FD		FE0390E	3			E0
🗖 🔷 addr	64'h000000087,J00202	00000	0000000080000202 00000000	80000	0000000	80000206	000000008	0000)	000
a 🔶 [1]	64'h000000°_/80000202	00000	000000080000202		0000000	80000206			
🗉 🔷 [0]	64'h0000',00080000200	00000	000000080000200		0000000	80000202			000
instruction_valid	2'h1 🦯	3	1		3				0
	0								
L	1								
🛷 serving_unaligned 🦯	1'h0								

 $0x90E^3 33F^2 \rightarrow 0xD = 1101$ (Compressed) 0x3 = 0011 (Not Compressed)



Store the Upper 16 bits into an inside register and serve the Non-Compressed instruction the next Cycle

Case 3)

	1'h1				
— Input Interface ————————————————————————————————————					
🗉 🔷 icache_data_q	32'h90e333fd	92c20	90e333fd	e032fe03	
🗉 🔷 icache_vaddr_q	64'h00000008000200	00000	000000080000200	000000080000204	
🔶 icache_valid_q	1'h1				
Output Intf					
🗖 🔶 instr	32'h0000000 32'h90e3	00009	00000000 90e2c3fd	0000e032 fe0390e3	000
💶 🔶 [1]	32'h0000000	00009	0000000	(0000E032	000
•• 🔶 [0]	32'h90E333FD	00E4	905-33FD	FE0390E3	ΪE0
🗖 🔶 addr	64'h00000008000202	00000	000000080000202 000000080000	0000000080000206 0000000080000	000
• 🔶 [1]	64'h00000008000202	02000	000000080000202	000000080000206	
😐 🔶 [0]	64'h00000008000200	00000	0000000080000200	0000000080000202	000
at instruction_valid	2'h1	3	1	3	0
	0				
·	1				
serving_unaligned	1"0				

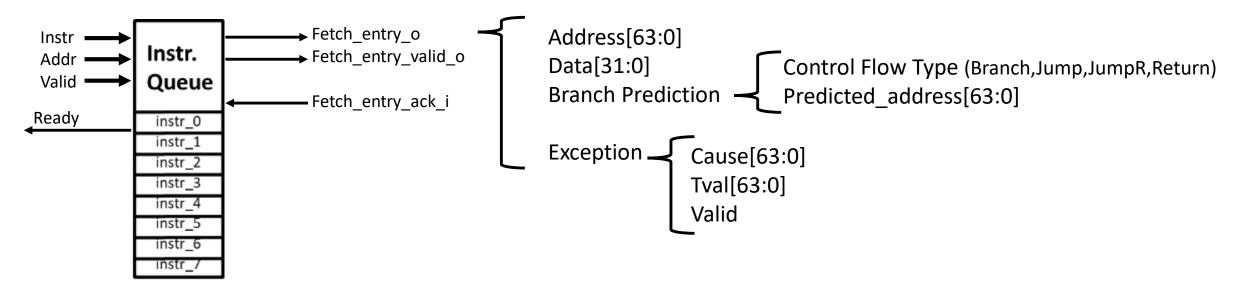
 $0xE03^{2}$ FE03 \rightarrow Upper 16bits Compressed!



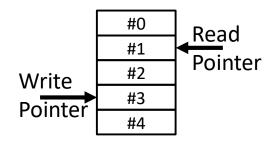
We can serve a compressed and the 32bit unaligned instruction in the same Cycle

Pushing into Instruction Queue

• Valid Instructions can be Pushed into the instruction Queue



• Form of circular Buffer: pointer for write and read.

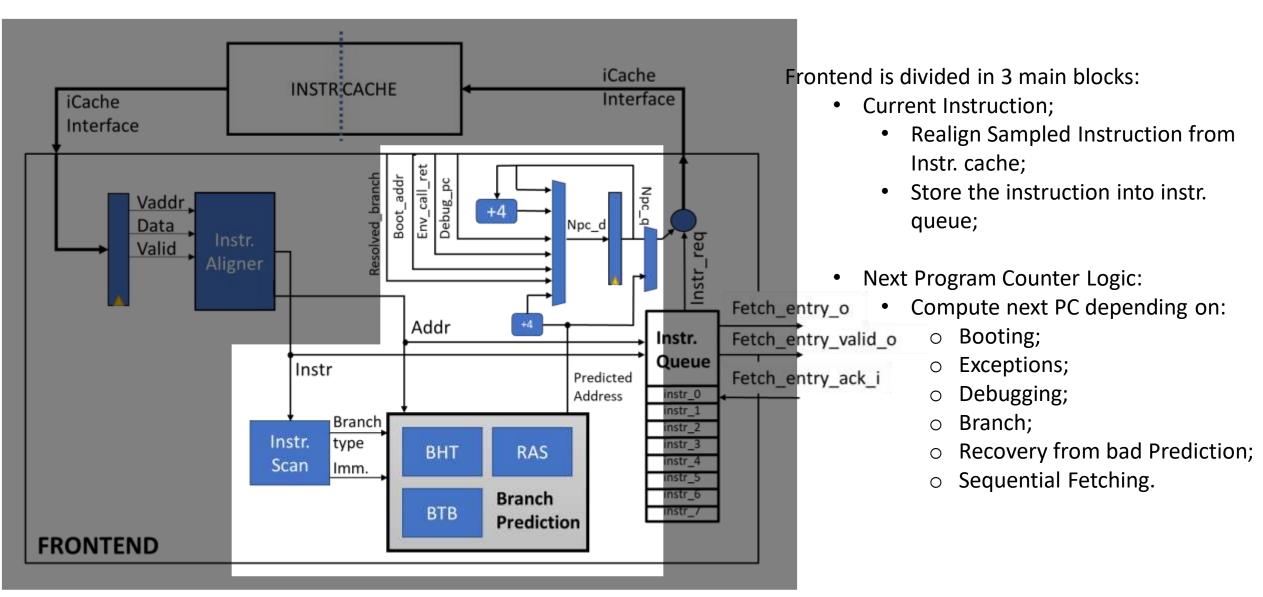


Valid Instructions Advance Write Pointer

Entry Acknowledge Advance Read Pointer

• If the Queue becomes full \rightarrow ready = 0 stops request to cache

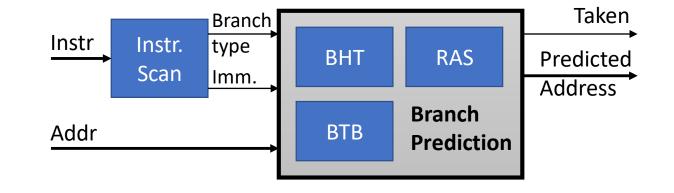
Next Program Counter Logic



Branch Prediction

CVA6 implements Dynamic Branch Prediction:

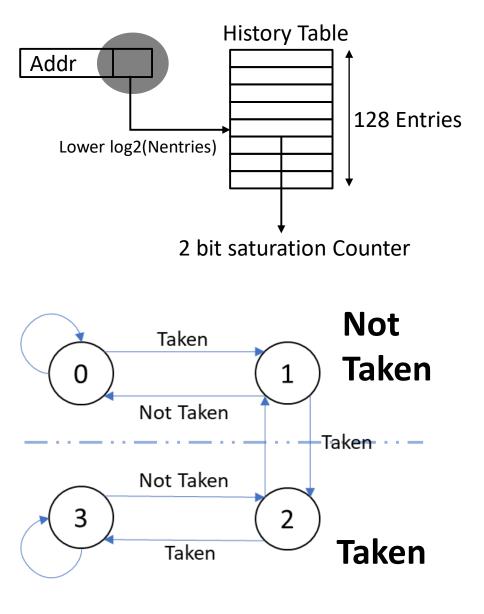
- Branch History Table (BHT): Used for predicting Branches (BEQ, BNE, etc).
- Branch Target Buffer (BTB): Used for predicting JALR.
- Return Address Stack (RAS): Used for predicting returns from function calls.



Instruction Scan Pre-decodes the instructions to control branch prediction:

- Extract Type of branch \rightarrow Select Predictor.
- Extract Immediate \rightarrow Calculation of predicted Address.

BHT: Bimodal



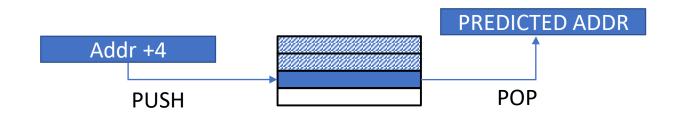
Addr Lower log2(Nentries)

Each Buffer contains the last Address Known of the predicted JALR (that isn't a return).

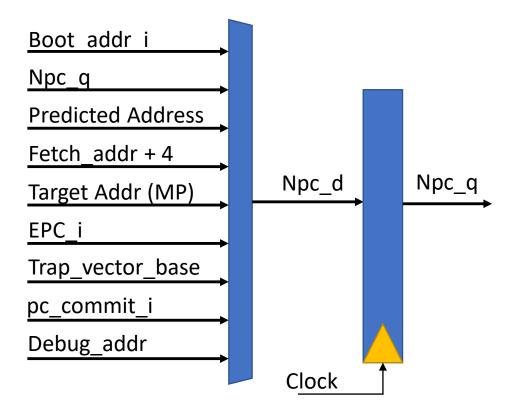
RAS

On Function Calls: PUSH \rightarrow Save Return Address on the stack

On Function Return: POP \rightarrow Use the popped Address as Predicted Address



Next Program Counter

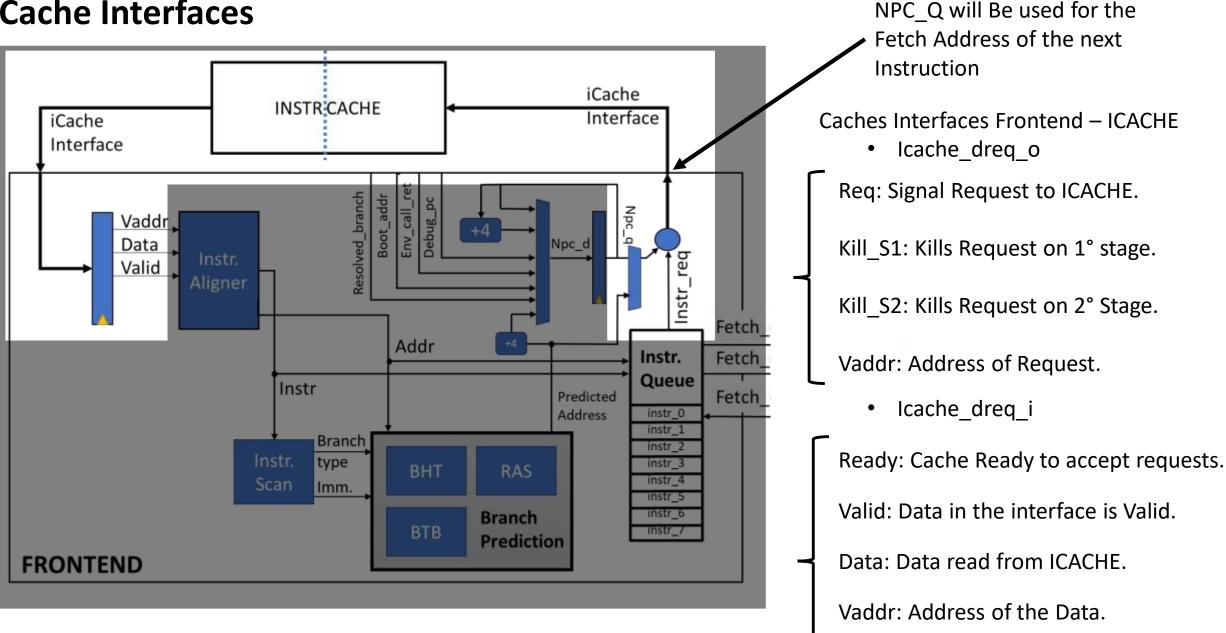


- NPC_Q → Next Program Counter at t0
- NPC_D → Next Program Counter at t1
- NPC_Q is used to decide the address of the next instruction to fetch.

SystemVerilog Code Snippet

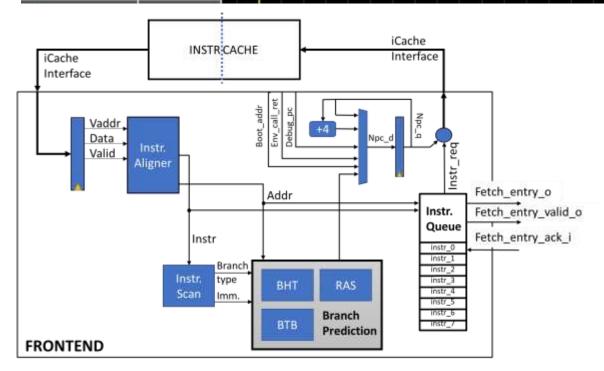
automatic logic [riscv::VLEN-1:0] fetch address; // check whether we come out of reset this is a workaround, some tools have issues having boot addr i in the asynchronous // reset assignment to npc q, even though // boot addr i will be assigned a constant / on the top-level if (npc rst load q) begin npc d = boot addr i; fetch address = boot addr i: end else begin fetch address = npc q; // keep stable by default npc d = npc q; // 8. Branch Prediction if (bp valid) begin fetch address = predict address; npc d = predict address; // 1. Default assignment if (if ready) npc d = {fetch address[riscv::VLEN-1:2], 2'b0} + 'h4; Replay instruction fetch if (replay) npc d = replay addr; // 3. Control flow change request if (is mispredict) npc d = resolved branch i.target address; // 4. Return from environment call if (eret_i) npc_d = epc_i; 5. Exception/Interrupt if (ex valid i) npc d = trap vector base i; Pipeline Flush because of CSR side effects On a pipeline flush start fetching from the next address of the instruction in the commit stage we came here from a flush request of a CSR instruction or AMO, as CSR or AMD instructions do not exist in a compressed form // we can unconditionally do PC + 4 here // TODO(zarubaf) This adder can at least be merged with the one in the csr regfile stage if (set pc commit i) npc d = pc commit i + {{riscv::VLEN-3{1'b0}}, 3'b100}; // enter debug on a hard-coded base address if (set debug pc_i) npc_d = ArianeCfg.DmBaseAddress[riscv::VLEN-1:0] + dm::HaltAddress[riscv::VLEN-1:0]; icache dreg o.vaddr = fetch address;

Cache Interfaces



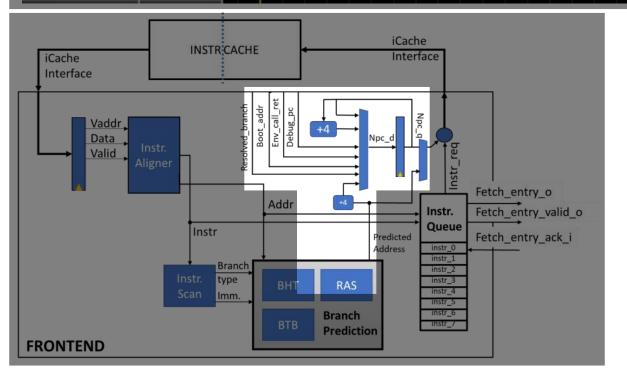
Ex: Exceptions from the request.

- di j	1761							
🔷 bp_valid	110	_						
🗖 🔷 npc_d	641600000000000000000000000000000000000	000000	0000000080000208	000000008000020c	00000000800003e4	00000003800001e8	00000000800001ec	Įα
🖬 🔦 IVC_Q	641h000000080000204	000000	0000000080000204	0000000080000208	000000008000020c	000000000001e4	0000000800001e8	10
- IS Out Interface								
<pre>included exactly dwg_o</pre>	111 110 110 111 641 1 0	1001	100100000000000000000000000000000000000	1 0 0 1 000300080000206	1 0 1 1 00000000600001e2	1 0 0 0 00000000000000001e4	1 0 0 0 000000000000001ei	341
NII_52	0							
spec	1 641000000080000204	000000	0000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	100000000001E 1	000000000000000000000000000000000000000	Ter
- IS ing interface	04 100000000000000000	000000	0000000060000204	000000000000000000000000000000000000000	00000000000001E2	00000000800001E4	00000000800001E8	00
kache_drog_i	116 116 32160000000 1161	0.0.000	0 0 0000000 00000000	0 0 0000000 000000000000000000000000000	10 0 0000000 000000000	0 0 0000000 000000000	0 0 0000000 00000000	0
📣 valid	1741							
🗢 📣 cinta	32160e333td	92c200e4	90e333td	e032fe03	6702e416	c7334605	719300c7	14
🗖 🥔 vialda	641100000000000000000000000000000000000	000000	0000000080000200	00000000000000000	000000060000208	0000000600001e2	00000000800001e4	00
D 📣 EX	641h000000000000000000000000000000000000	00000000	0000000 0000000000000000000000000000000	0.0	والمتحدة المتحد المتحد	والمتفاقية وتنتق	والمتحد المتراد والمراد	
- Sampled Cache Data	S CHARGE STATE		a second and a second					
🛛 🍨 icache_data_q	32h92c200o4	06330001	92c200e4	90e3331d	Le032fe03		0000:733	171
🔷 icache_valid_q	TM			and a second second second			STORE STORE STORE	
🗖 🔹 icache vaddr_q	64*h00000008000011c	000000	000000008000016	0000000060000200	0000000060000204		0000000600001e2	Įα



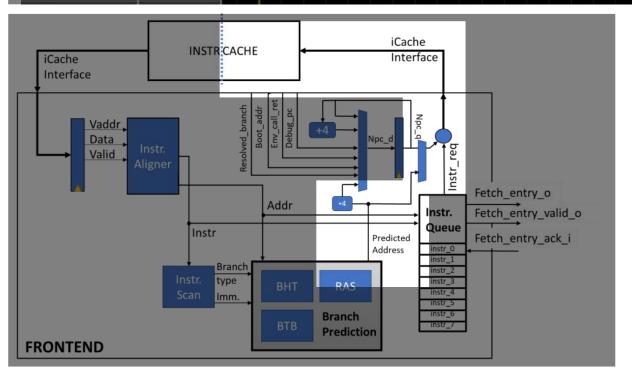
- NPC_Q & NPC_D;
- ICACHE_DREQ_O: Request Interface
- ICACHE_DREQ_I: Response Interface
- Sampled Data from ICACHE:
 - Icache_data_q
 - Icache_valid_q
 - Icache_vaddr_q

-4° ck_l -4° bp_valid	1761 1760							
🛛 🗣 npc_d	641100000000000000000000000000000000000	000000	0000000080000208	[000000008000020c	[00000000800003e4	00000003800001e8	00000000800001ec	10
P * rpc_q - IS Out Interface	64%000000080000204	000000	0000000080000204	0000000080000208	000000008000020c	000000000000001e4	00000000600601e8	[0
 icaclu_ducq_n inq inl[51 inl[52 	17h1 17h0 17h1 647h 1 0 0	1001	100100000000000000000000000000000000000	1 0 0 1 00000000000000208	\$ 1 0 1 1 0000000800001e2	10000000000000000000000000000000000000	11 0 0 0 00000000000000000000000000000	8 11
spec								
IS inp interface	641h000000060000204	000000	0000000060000204	000000000000000208	_00000000800001E2	00000000800001E4	00000000000001E8	
 icache_drog_i icache_drog_i medy valid 	110 110 32960000000 111 111	0.0:000	0.0.0000000.000000000	10 0 0000000 000000000	10 0 0000000 000000000.	3 0 0000000 000000000	0 0 00000000 00000000	
o 4 cinta	32'h90e333Hd	92c200e4	90e333fd	e032fe03	6702e416	c7334605	719300:7	14
n 🖉 vatidr n 🖉 ex	641100000000000000000000000000000000000	and the other designs in the local division of the local divisiono	0000000080000200 0000000 000000000000000	0.000000000000000000000000000000000000	0000000080000208	0000000000001e2	00000000800001e4	
Sampled Cacho Data								
icache_data_q icache_valid_q	32h92c200o4 11t1	06330001	92:200e4	(90e333ld	1e032fe03		0000:733	171
🗖 🕈 icacte_vaddr_q	647h0000000800001tc	000000	00000000800001/c		0000000080000204		00000000000001e2	



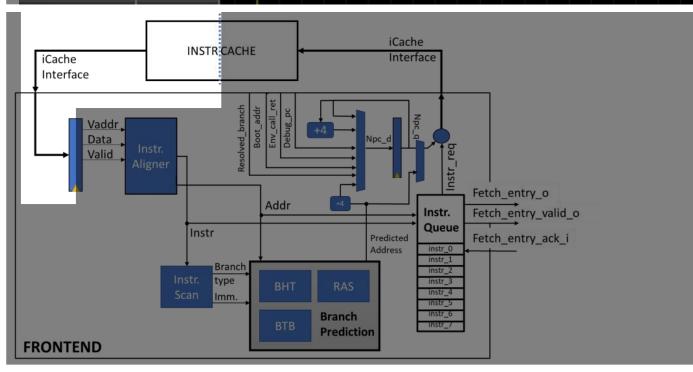
- NPC_Q & NPC_D;
- ICACHE_DREQ_O: Request Interface
- ICACHE_DREQ_I: Response Interface
- Sampled Data from ICACHE:
 - Icache_data_q
 - Icache_valid_q
 - Icache_vaddr_q

- ≪ clk_i - ≪ bp_valid © ≪ apc_d © ≪ apc_q	1%1 1%0 64%000000000000000000000 64%0000000000	000000	0000000080000208	000000008000020c 000000008000020c	[00000000800061e4 [00000008000820e	00000000800001e6	000000080001=c 0000000000001=6	[00
IS Out interfece IS Out inter	1h1 1h0 1h0 1h1 64h 1 0 0 1	1001	1 0 0 1 0000000000000000000000000000000	(10010000000000000000000000000000000000	1 0 1 1 000000000000000000000000000000	11 0 0 0 00000000000000000000000000000	[] 0 0 0 000000000000000000000000000000	eë [1]
🗖 🕗 vaddr	641x000000080000204	000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000800001E2	00000000800001E4	000000000000001E8	00
Is inplimented	0.0.000	0.0.0000000.00000000	1e e aapaacoo occoocacoo .	10 0 0000000 000000000	0 0 0000000 000000000	0 0 00000000 000000000	01	
🗢 🗢 chin	32160e333td	92c200e4	90e333fd	e032fe03	6702e416	c7334605	719300c7	[64
🖬 🥔 viaddr	641100000000000000000000000000000000000	000000	0000000080000200	000000080000204	0000000080000208	00000000800001e2	00000000800001e4	100
N# \$40	641100000000000000000000000000000000000	00000000	000000 00000000000000000000000000000000					
Sempled Cache Data cache_data_q cache_valid_q	32/62:20064 1761	06330007	92c200e4	(90e338td	0321003		0000c733	171
🛛 🔶 icache_vaddr_q	64"h0000000800001/c	000000	00000000800001fc		0000000080000204		00000000800001e2	100



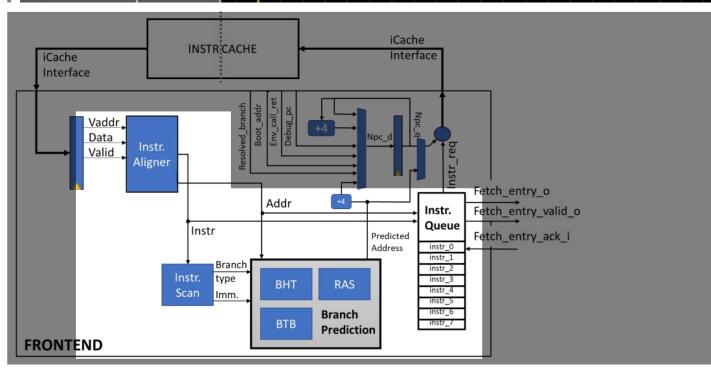
- NPC_Q & NPC_D;
- ICACHE_DREQ_O: Request Interface
- ICACHE_DREQ_I: Response Interface
- Sampled Data from ICACHE:
 - Icache_data_q
 - Icache_valid_q
 - Icache_vaddr_q

- et al i	1711						
🔷 bp_valid	110						
🖬 🔹 npc_d	641100000000000000000000000000000000000	000000	0000000080000208	00000008000020c	00000000800003e4	00000000800001e8	00000000800001ec
<pre>Pair Pair Pair Pair Pair Pair Pair Pair</pre>	6414000000060000204	000000	000000080000204	0000000080000208	000000008000020e	000000000000001e4	(0000000800001e8
 kickelugen ing kiljs1 kiljs2 	111 110 110 111 641 1 0 0	1001	100100000000000000000000000000000000000	100100000000000000000000000000000000000	1 0 1 1 000000000000000102	100000000000000000000000000000000000000	[1 0 0 0 000000000000000000000000000000
spec sec	1 641h000000000000000000000000000000000000	000000	00000000080000204	000000000000000000000000000000000000000	00000000800001E2	00000000800001E4	00000000000001E8
1\$ Inp Interface	110 110 32100000000	0.000	0 0 00000000 00000000	0 0 00000000 00000000000000000000000000	0 0 00000000 000000000	0 0 00000000 000000000	0 0 0000000 000000000
Icache_droq_i ready valid	110 110 37 1000000	00000				, 3 3 3000000 00000000	
🛛 🧢 cata	32160e333td	92c200e4	90e333td	e032fe03	6702e416	c7334605	719300:7
🖬 🧢 vaddr	6411000000080000200	and the second s	0000000080000200	000000080000204	0000000080000208	00000000000001e2	00000000800001e4
EX EX	64'h000000000000000000	000000000	2000000 2000000000000000000000000000000	0.0		والمتغا الناع الناع	المعر الأنتق التقار التقر
- Surrelid Cache Deta	and the second second		and the second second				
icache_data_q icache_valid_q	32/162c200e4 1711	06330001	92c200e4	(90e338td	e032fe03		0000c733
o 🔹 icache vaddr g	6410000000800001%	000000	00000000800001fc		0000000080000204		00000000800001e2



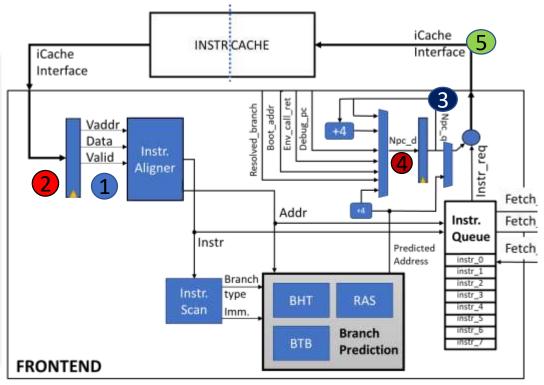
- NPC_Q & NPC_D;
- ICACHE_DREQ_O: Request Interface
- ICACHE_DREQ_I: Response Interface
- Sampled Data from ICACHE:
 - Icache_data_q
 - Icache_valid_q
 - Icache_vaddr_q

 icache_valid_q icache_valid_q 	17h1 647h00000000800001fc	000000	000000008000016	10000000080000200	0000000080000204		00000000600001=2	0
Sampled Cache Data	32162:2004	06330007	92c200e4	90e3331d	e032fe03		0000:733	71
EX EX	641100000000000000000000000	00000000	0000000 0000000000000000000000000000000					
🗖 🥔 vaddr	641100000000000000000000000000000000000	000000	0000000080000200	000000080000204	0000000080000208	00000000800001e2	00000000800001e4	
o data	17h1 327h90e333Hd	92c200e4	90e3331ti	e032te03	6702e416	c7334605	719300c7	
1\$ Inp Inteface	110 110 32100000000 111	0.0.000	0.00000000.000000000	0 0 0000000 000000000000000000000000000	0 0 00000000 000000000	0 0 00000000 000000000	0 0 0000000 0000000	
kil_s1 kil_s2 spec D valda	1 0 1 6416000000000000000000000000000000	000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000000001E2	000000000000001E4	(00000000000001E8	
IS Out Interface	111 110 110 111 641	1001	100100000000000000000000000000000000000	100100000000000000000000000000000000000	1 0 1 1 00000000800001e2	1 0 0 0 000000000000001e4	110000000000000000000000000000000000000	66 [1
© ● npc_d © ● npc_d	641h000000080000208 641h000000080000204	000000	0000000080000208	[000000008000020c [0000000080000208	[00000000600001e4 [000000008000020c	00000000800001e6 00000000800001e4	00000000800001ec 00000000800001e8	0] [0
dk_i → bp_velid	1711 1710							



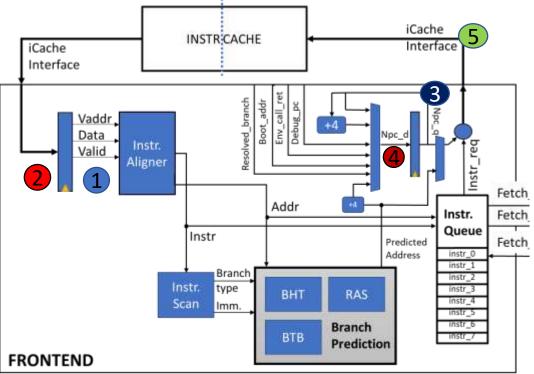
- NPC_Q & NPC_D;
- ICACHE_DREQ_O: Request Interface
- ICACHE_DREQ_I: Response Interface
- Sampled Data from ICACHE:
 - Icache_data_q
 - Icache_valid_q
 - Icache_vaddr_q

 the valid the valid the diagonal <lithe diagonal<="" li=""> the diagonal</lithe>	1741 1740 647x000000080000208 647x000000080000204	Conception of the local division in which the local division in th	000000008000208			
4 Out Interface	1761 1760 1760 1761 6476 1 0 0	1001	1 0 0 1 0000000000000000000000000000000	100100000000000000000000000000000000000		
spec 5	1 641x000000080000204	000000	0000000080000204	0000000000000208		
International Strength Stre	1760 1760 327600000000 1761 1761	0.0.000	0 0 00000000 000000000	0.0.0000000.0000000000		
D / cinta	32160e333td	92c200e4	90e332fri	e0321e03		
	641100000000000000000000000000000000000	000000	00000000000000200 0000000 0000000000000	000000080000204		
Sampled Cache Data statu data_q statu data_q scache_valid_q	32162c200e4	06330001	92c200e4	90x3331d		
D icache_vaddr_q	64'h00000008000011c	000000	00000000800001fc	000000000000000000000000000000000000000		

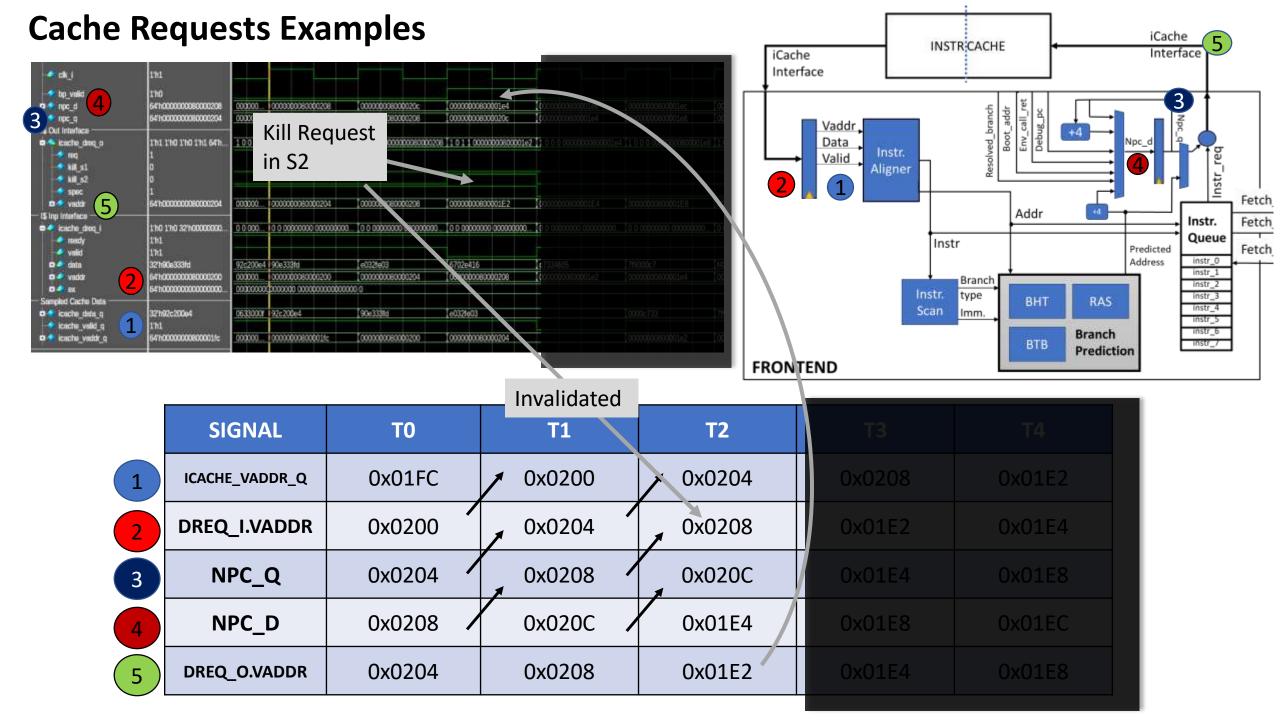


	SIGNAL	то	T1			
1	ICACHE_VADDR_Q	0x01FC	0x0200	0x0204	0x0208	0x01E2
2	DREQ_I.VADDR	0x0200	0x0204	0x0208	0x01E2	0x01E4
3	NPC_Q	0x0204	0x0208	0x020C	0x01E4	0x01E8
4	NPC_D	0x0208	0x020C	0x01E4	0x01E8	0x01EC
5	DREQ_O.VADDR	0x0204	0x0208	0x01E2	0x01E4	0x01E8

de cikji	110						
	641400000000000000000000000000000000000	000000	0000000080000208	000000008000020c	10000		
o nuc_q	6411000000080000204	000000	0000000080000204	0000000080000208	10000		
Out Interface	111 110 110 111 641 1	1001	1 0 0 1 000000080000204	1 0 0 1 0000000080000208	101		
4 kill_s1 ≤ kill_s2	0 0						
vector 5	1 641x000000000000000000000000000000000000	000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	0000		
kache_drog_i	116 116 32100000000 171	0.0:000	0.0.00000000000000000000000000000000000	0 0 00000000 00000000000000000000000000	000		
olaria dista	1%1 32%90e333fd	92-200e4	90e333tri	e032fe03	16702		
vaddr 2	647h000000080000200 647h00000000000000000000000	000000		0000000080000204	0000		
mpled Cache Data	0-1000000000000000000000000000000000000	1000000					
• icache_data_q • icache_valid_q 1	32h92c200e4	06330001	92c200e4	90e333td	1e 032		
icacte_veddr_q	647h000000008000011e	000000	00000000800001fc	0000000080000200	10000		



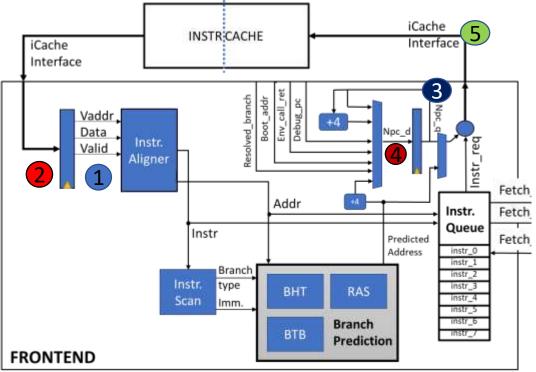
	SIGNAL	то	T1	T2		Т4
1	ICACHE_VADDR_Q	0x01FC	0x0200	0x0204	0x0208	0x01E2
2	DREQ_I.VADDR	0x0200	0x0204	0x0208	0x01E2	0x01E4
3	NPC_Q	0x0204	0x0208	0x020C	0x01E4	0x01E8
4	NPC_D	0x0208	0x020C	0x01E4	0x01E8	0x01EC
5	DREQ_O.VADDR	0x0204	0x0208	0x01E2	0x01E4	0x01E8



Cache F	Reque	sts Exan	nples				iCache	INSTRCACHE	4	Cache 5	
to vaid	171 170 647600000008000208	000000_00000000000000000000000000000000	0000000380000206	[00000000600001e4	0000000000000146		Interface			8	
3 Dut Interface	641h000000080000204	000000 00000000000000000000000000000000	000000000000000000000000000000000000000	000000008000020c	0000000800001e4	00000000000146	Vaddr	addin rand		Np	
ininia inini	171 176 176 171 647 1 0 0	1001 1001000000000000000000000000000000	94 [3 0 0 1 0000000000000000	96 1 0 1 1 00000000000000000000000000000	2 1 0 0 0 00000000000000000000000000000	11 0 0 0 00000000000000000000000000000	Data Valid	instr.		Instr_req	
visidar - 15 ing interface	1 647h0000000080000204	000000 00000000000000000000000000000	000000000000000000000000000000000000000	000000000000001E2	00000000800001E4	(0 50000000000001£8 (00		A	ddr 📕		Fetch
<pre>cache_drog_i</pre>	1%0 1%0 32%00000000 1%1 1%1	0 0 000 0 0 0000000 00000000		0.00000000.0000000000	0 0 00000000 000000000	0 0000000000000000000000000000000000000		Instr	Predict	Queue	Fetch Fetch
o 4 cata	32190e3331d	92c200e4 90e333trl	e032fe03	6702e416	c7334605	1 9300:7			Addres	is instr_0	
Sempled Cache Data	641H200000080002200 641H200000000000000000000000		0000000080000204	0000000060000208	0000000800001e2	(0 :0000006000;01=1 (00		Branch Instr. type	BHT RAS	instr_2 instr_3	
o ● icache_deta_q	32162c200e4	06330007 92c200e4	90e338ld	e032fe03		00006733		Scan Imm.		instr_4 instr_5	
of icacte vaddr q	641h0000000800001/c	000000	0000000080000200	0000000080000204		(g *00200000000001+2(0)			BTB Branch Prediction	instr_6 instr_/	
							FRONTEND				

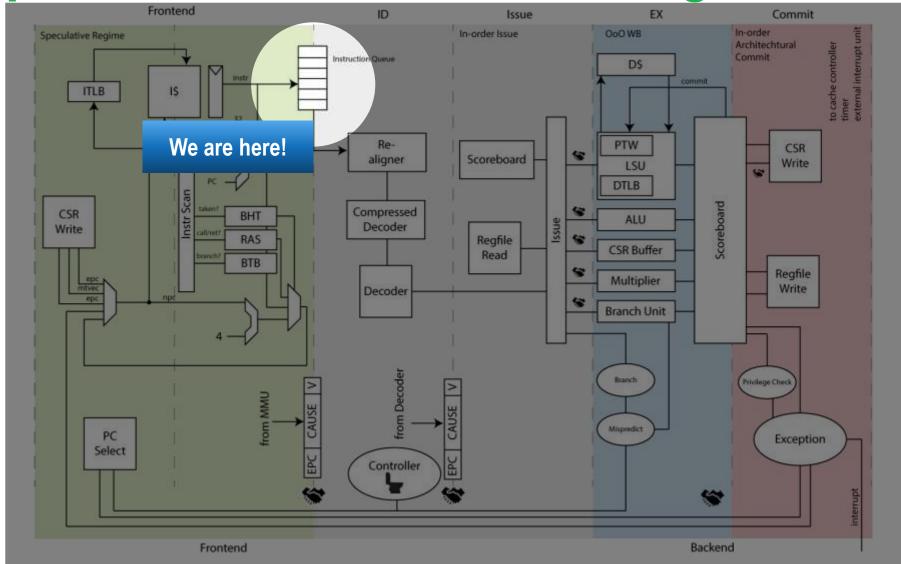
	SIGNAL	то	T1	T2	T3 alid	Т4
1	ICACHE_VADDR_Q	0x01FC	0x0200	0x0204	0x0208	0x01E2
2	DREQ_I.VADDR	0x0200	0x0204	0x0208	0x01E2	0x01E4
3	NPC_Q	0x0204	0x0208	0x020C	0x01E4	0x01E8
4	NPC_D	0x0208	0x020C	0x01E4	0x01E8	0x01EC
5	DREQ_O.VADDR	0x0204	0x0208	_{0x01E2} /	0x01E4	0x01E8

to valid	1711 1740						
senped 4	641100000000000000000000000000000000000	000000	0000000080000208	000000008000020c	00000000600001e4	0000000800001e8	00000000800001ec
• npc_q 4 Out Interface	6411000000080000294	000000	0000000080000204	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000800001e4	[00000000800001e8
 kache_dwq_p mq kall_s1 kall_s2 	17h1 17h0 17h0 17h1 647h 1 0 0	1001	100100000000000000000000000000000000000	[1 0 0 1 000000000000000000000000000000	11011000000000000000000000000000000000	10000000000000000000000000000000000000	[1 0 0 0 000000000000000000000000000000
spec static strug interface	1 641h0000000080000204	000000	000000000000000000000000000000000000000	0000000000000208	00000000800001E2	00000000800001E4	(00000000000001E8
icache_dicq_i nandy valid	1%0 1%0 32%00000000 1%1 1%1	0.0000	0.0.00000000.000000000	to o nadoacco occosocoo.	0 0 0000000 000000000.	0 0 0000000 000000000	0 0 0000000 000000000
🖸 🧢 cinta	32190e3331d	92c200e4	90e333fd	e032fe03	6702e416	c7334605	719300c7
o vaddr	64100000008000200		0000000080000200	0000000080000204	0000000080000208	00000000600001e2	00000000800001e4
Sempled Cache Data	641h000000000000000000000000000000000000	000000000		0			يسير النظر أتقار التقار
🛛 🔹 icacha_data_q	32162c200e4	06330001	92c200e4	(90e3381d	00321003		0000:733
<pre>icache_valid_q icache_valid_q icache_validr_q</pre>	17h1 647h0000000800001/c	000000	000000006006016	000000000000000000000000000000000000000	000000000000000000000000000000000000000		00000000800001e2



	SIGNAL	то	T1	T2	T3 alid	Τ4
1	ICACHE_VADDR_Q	0x01FC	0x0200	0x0204	0x0208	0x01E2
2	DREQ_I.VADDR	0x0200	0x0204	0x0208	0x01E2	0x01E4
3	NPC_Q	0x0204	0x0208	0x020C	0x01E4	0x01E8
4	NPC_D	0x0208	0x020C	0x01E4	0x01E8	0x01EC
5	DREQ_O.VADDR	0x0204	0x0208	_{0x01E2} /	0x01E4	0x01E8

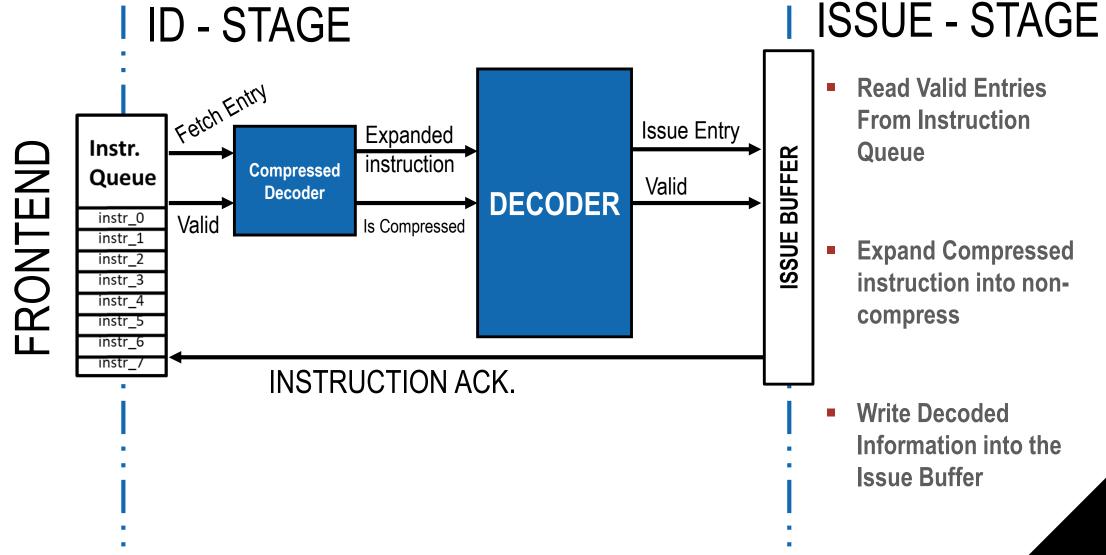
Deep Dive: Decode and Issue-Stage



ETH zürich

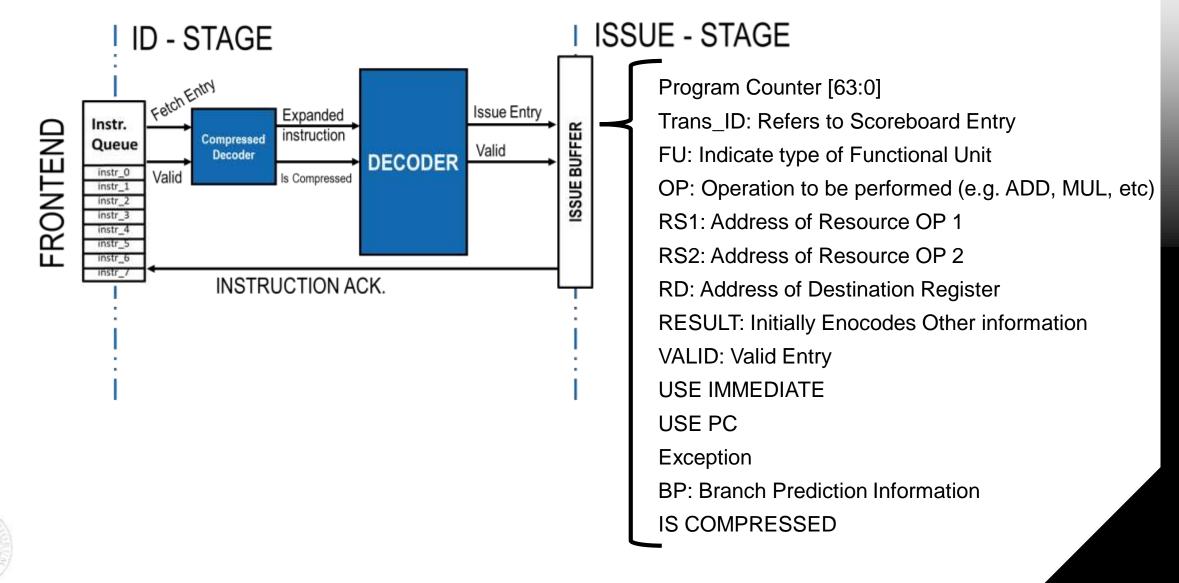


Decode Stage ID - STAGE

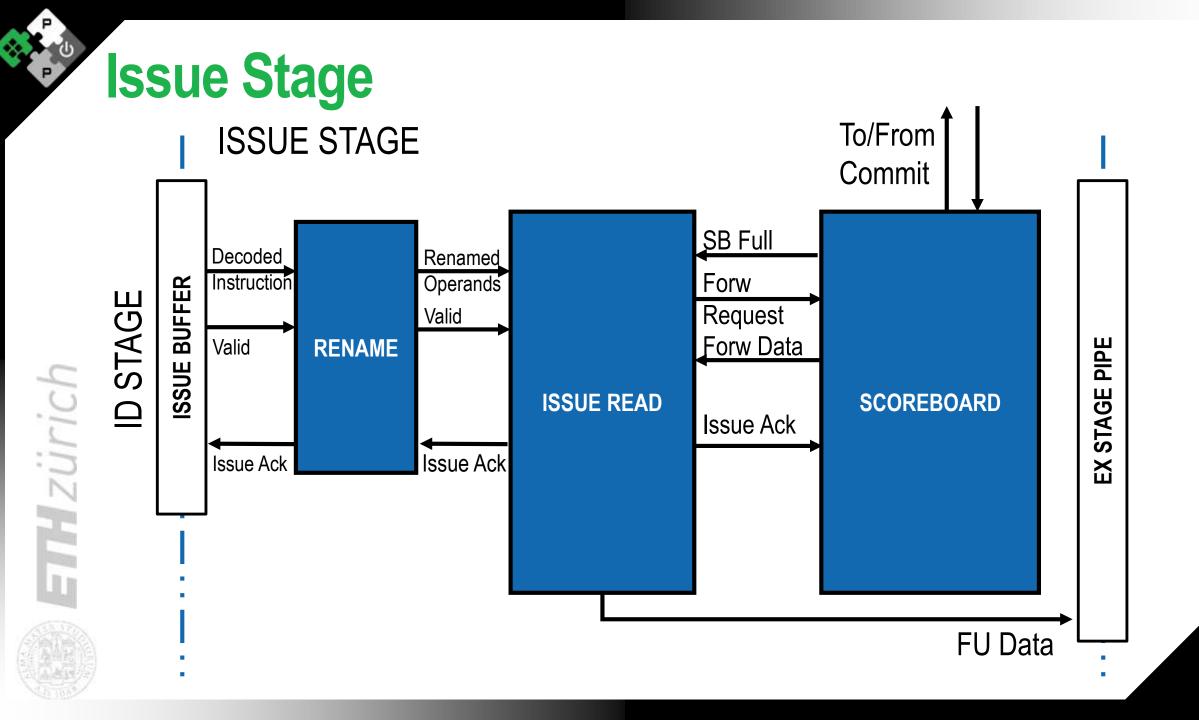


ETH zürich

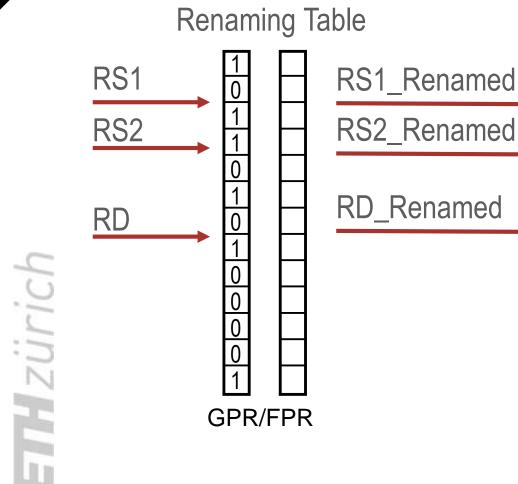
Decode Stage: Issue Interface



EnHzürich



RENAMING



Avoids WAW Hazard by renaming Register

Rare on In-order Single-Issue configuration but some Application benefits significantly

Example:

Check renaming table

Feed renamed operands to scoreboard

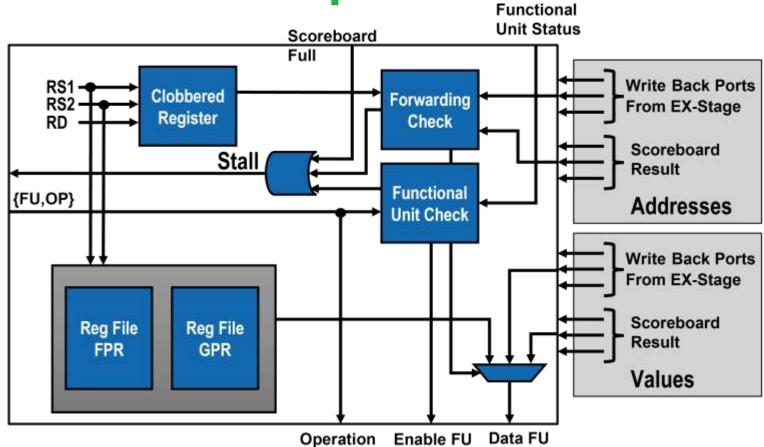
RS1	RS2	RNT[17]	RNT[10]	RS1 REN.	RS2 REN.
17	10	0	1	17	42

Update Renaming table:

- The renaming Table is updated only by the RD
- Tick/Untick Corresponding RD bit

RD	RNT[15]	RD REN.	RNT_N[15]
15	0	15	1

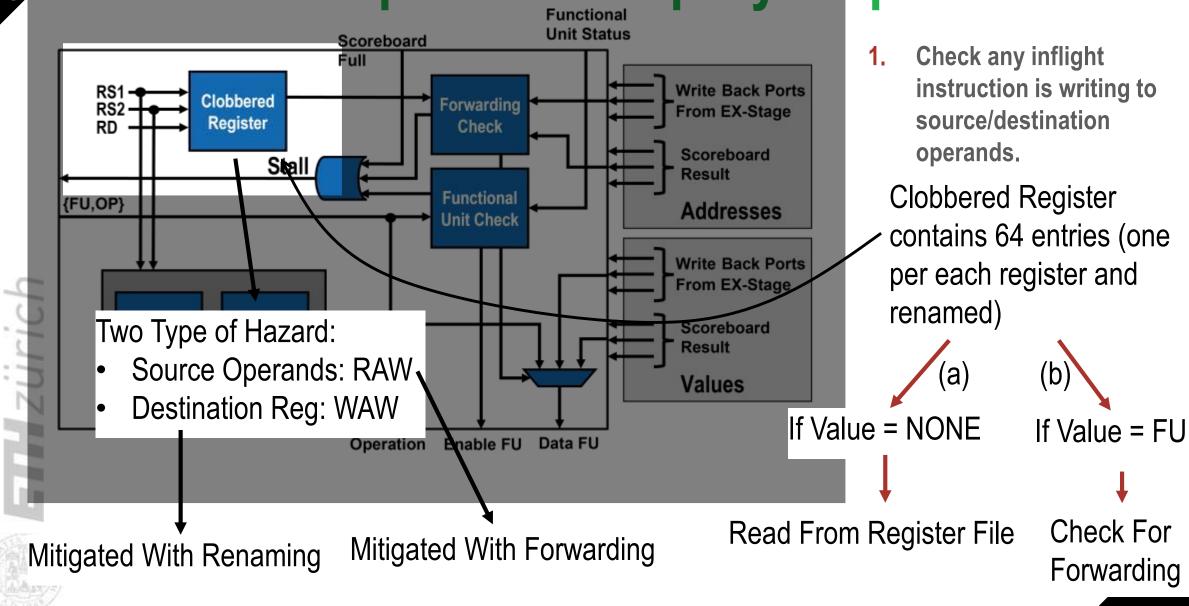
Issue Read Operands

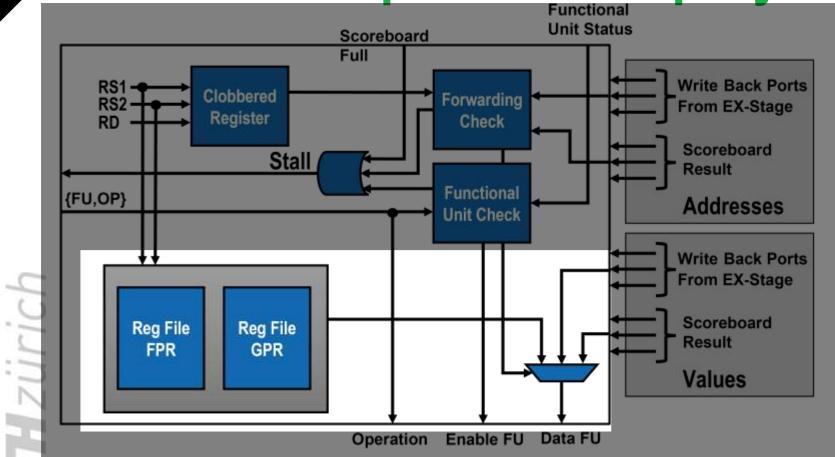


Main Tasks:

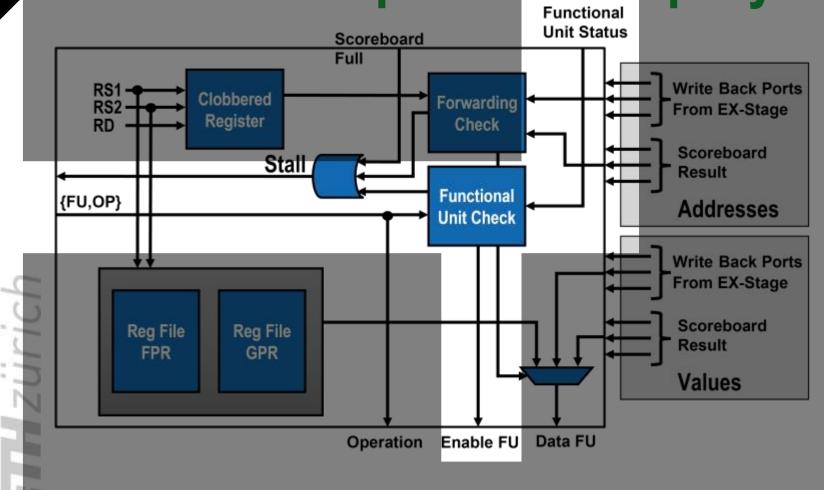
- Check for Data Hazard
- Check if Functional Units are Ready
- Forwarding Operands
- Interfacing to the Ex-Stage

ETHZürich



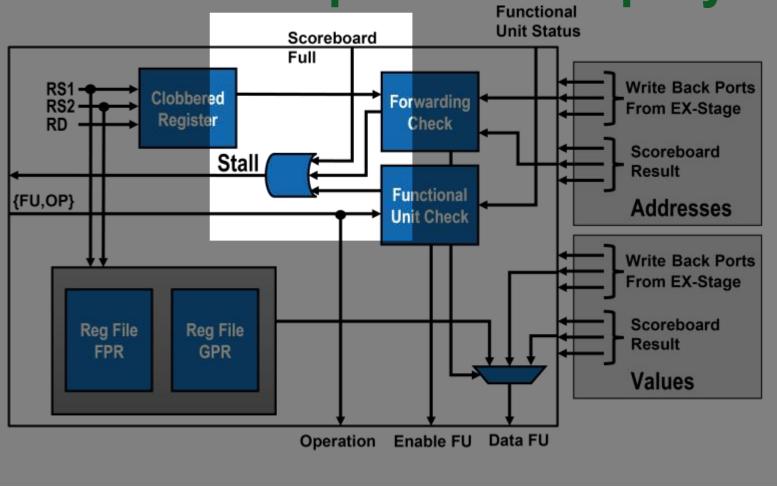


- 1. Check any inflight instruction is writing to source/destination operands.
- 2. (a) Read Values From Register File



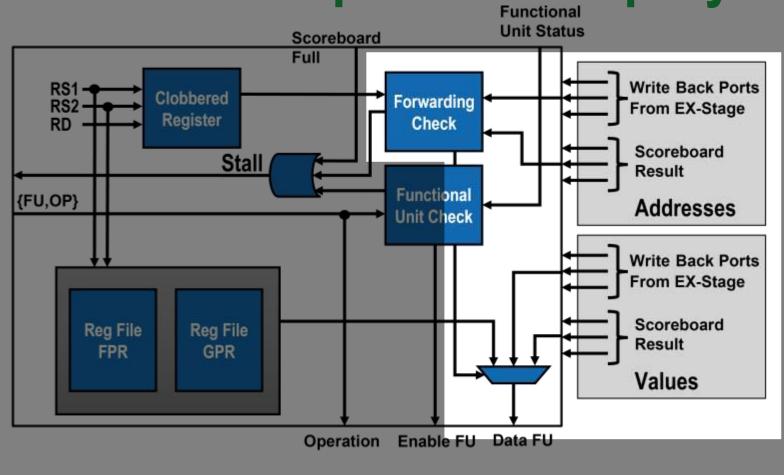
- 1. Check any inflight instruction is writing to source/destination operands.
- 2. (a) Read Values From Register File
- 3. Check If FU is Ready
 - Stall If not





Hzürich

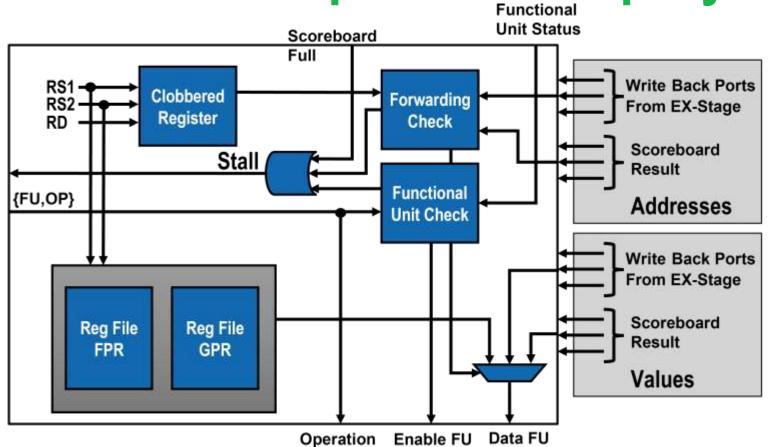
- 1. Check any inflight instruction is writing to source/destination operands.
- 2. (a) Read Values From Register File
- 3. Check If FU is Ready
 - Stall If not
- 4. Check if Scoreboard is full
 - Stall if it is
- 5. ISSUE



- 1. Check any inflight instruction is writing to source/destination operands.
- 2. (b) Check if Forwardable
 - Write Back Ports of Functional Units
 - Scoreboard Entries

Enhzürich

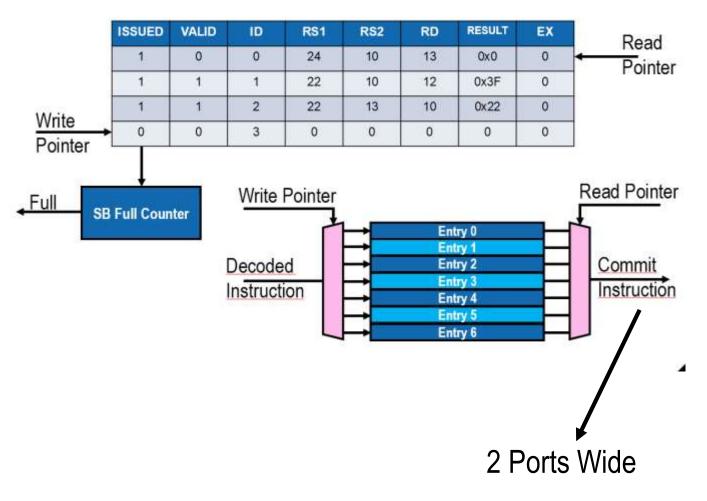




Hzürich

- 1. Check any inflight instruction is writing to source/destination operands.
- 2. (b) Check if Forwardable
 - Write Back Ports of Functional Units
 - Scoreboard Entries
- 3. Check If FU is Ready
 - Stall If not
- 4. Check if Scoreboard is full
 - Stall if it is
- 5. ISSUE

ScoreBoard



Track Each Issued and not Committed Instruction

Used for hiding load latency

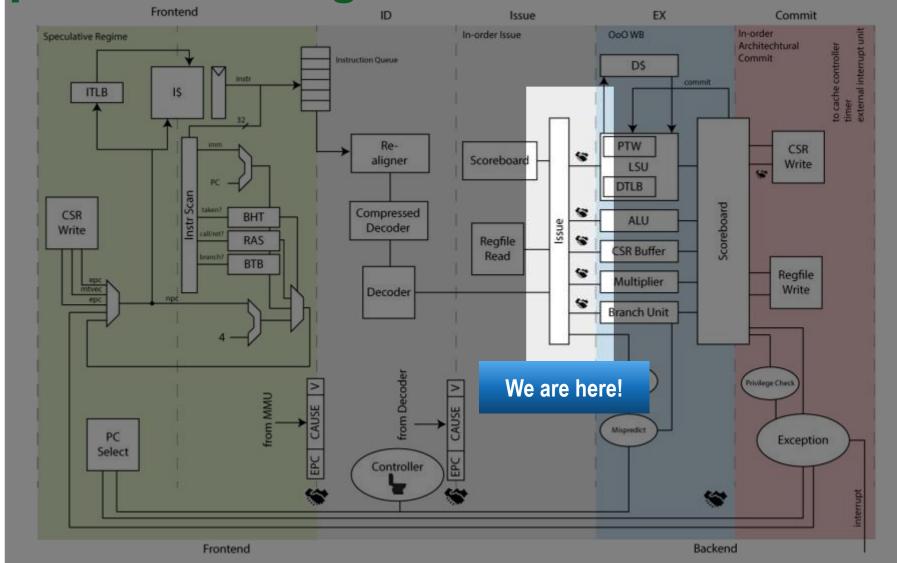
Write pointer Advance on issued Instructions

Read pointer Advance on Valid Instructions

Issued Instructions are counted to check if SB is Full

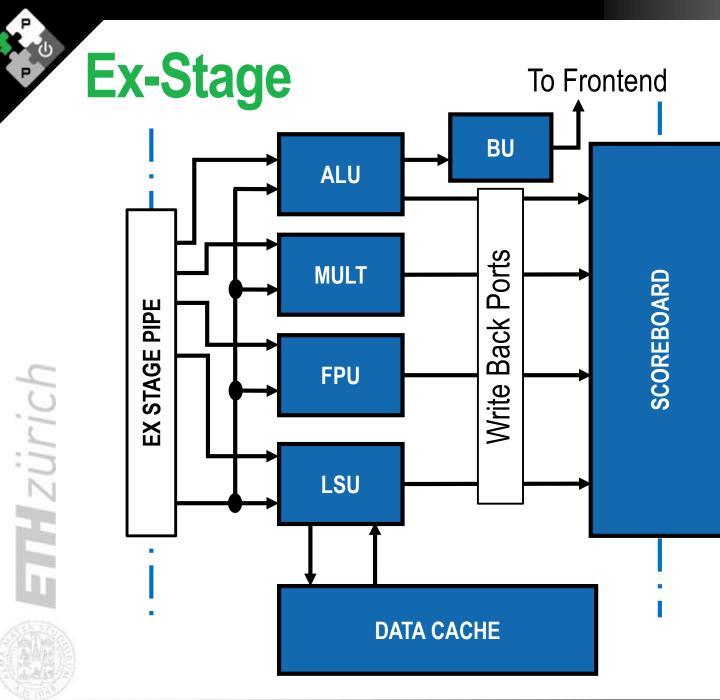


Deep Dive: Ex Stage



ETH zürich





Functional Unit Are selected Via enable from issue stage

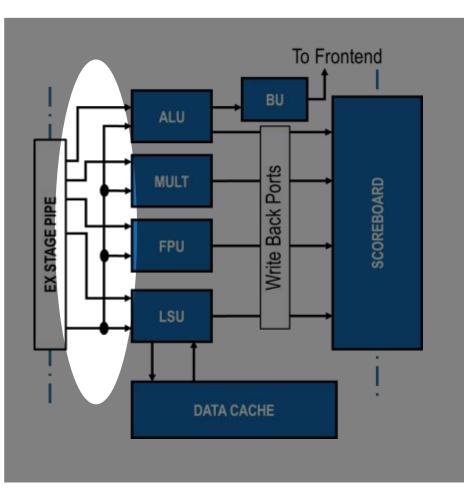
Write Back Ports are Connected to the Scoreboard

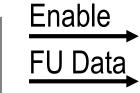
Branch Unit resolution is Connected to Frontend

Multiplier and FPU are internally Pipelined

Data Cache has 2 pipeline Stages, support hit under miss







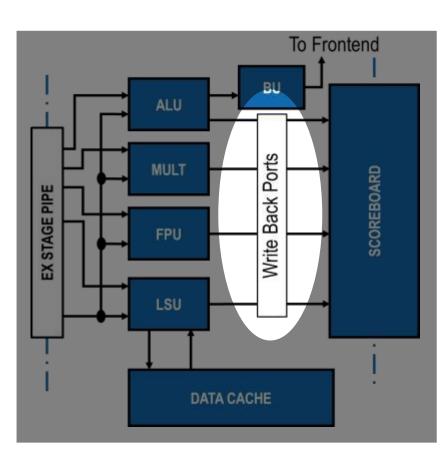
Operator: Type Of Operation Operand_a[63:0]: Value of OP_A Operand_b[63:0]: Value of OP_B Immediate[63:0]: Value of Immediate/OP_C Trans_ID: ID in the Scoreboard

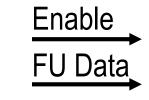
- The FU Data is shared Among Functional Units
- Enable Decide which FU to Activate



nh zürich







Operator: Type Of Operation Operand_a[63:0]: Value of OP_A Operand_b[63:0]: Value of OP_B Immediate[63:0]: Value of Immediate/OP_C Trans_ID: ID in the Scoreboard

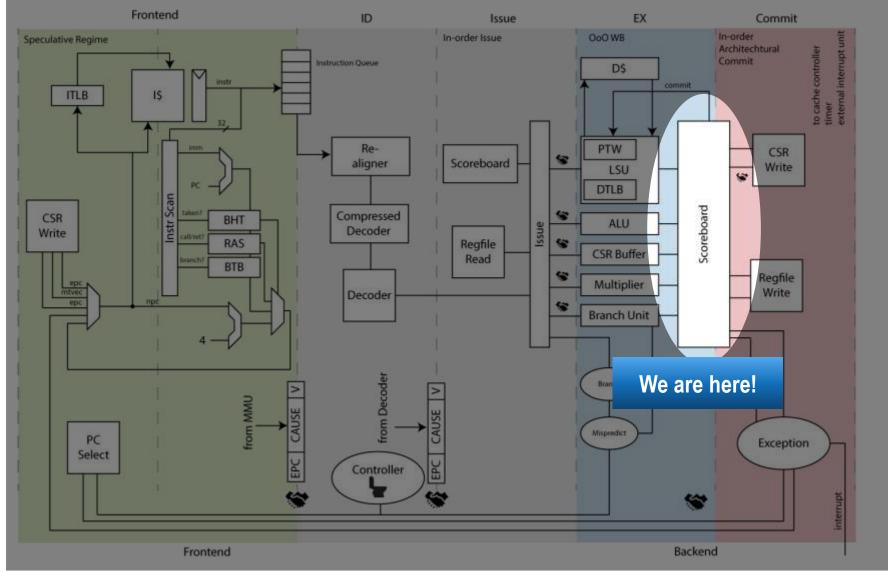
→ Valid → Trans_ID → Result[63:0]

Results Are Redirected to the scoreboard



- Hzürich

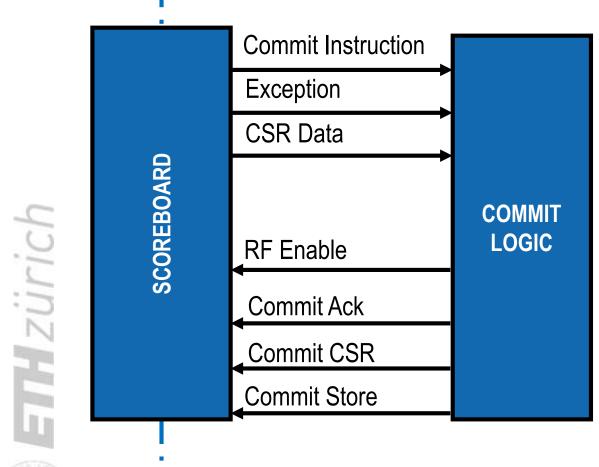
Deep Dive: Commit Stage



ETH zürich

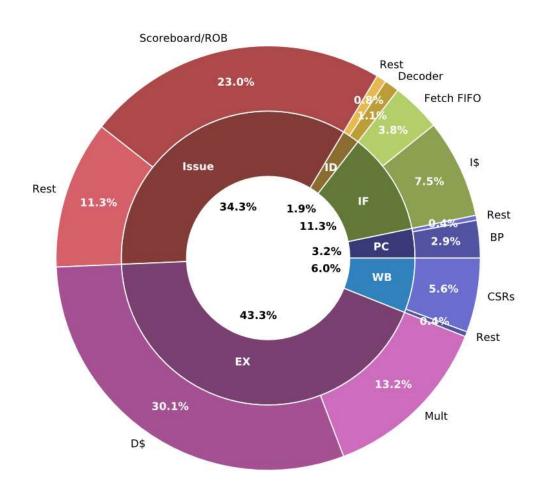
Commit Stage

Commit Stage



- Commit Stage Read the Scoreboard for Valid Instructions
- Committing an instruction has the following effect:
 - Write result to register File
 - Write to CSR
 - Allow Store to write main Memory
- Commit Stage retires Instruction in Order

IMPLEMENTATION RESULTS: Area



Area Results at 22nm FDSOI from GF

 Biggest Contribution are Cache Macros

 In general Sequential logic dominates Area





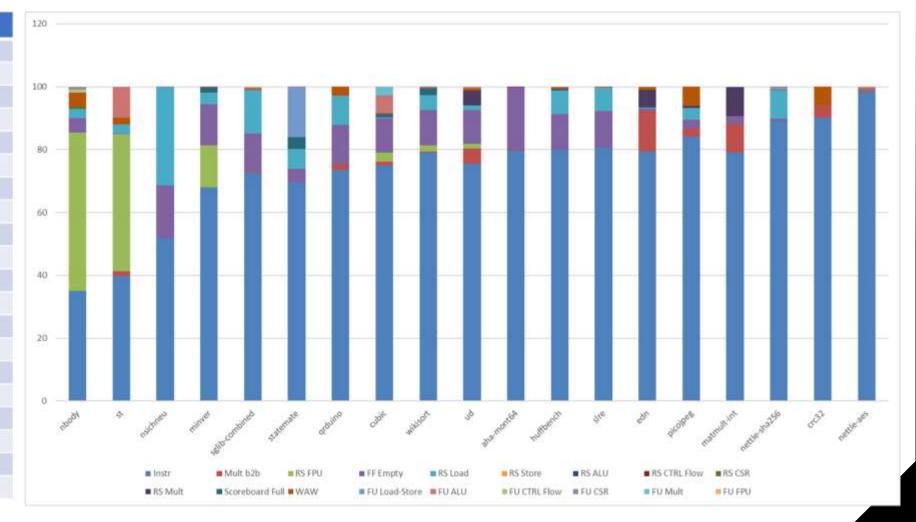
IMPLEMENTATION RESULTS: Energy

TABLE IV ENERGY PER OPERATION CLASS [pJ], LEAKAGE [mW]

Instr. Class	PC	IF S	tage	ID S	Stage	Issue			EX	Stage			WB	CSR	CTS	Rest	Tot
		I\$	Rest	Dec	Rest		L/S	VM	Mult	ALU	D\$	Rest					
Mul	0.30	4.72	0.51	0.01	0.09	1.42	0.22	3.46	0.97	0.02	5.53	0.07	0.05	0.22	4.25	0.76	22.60
%	1.33	20.88	2.26	0.04	0.40	6.28	0.97	15.31	4.29	0.09	24.47	0.31	0.22	0.97	18.81	3.36	100.00
Div	0.25	3.19	0.35	0.00	0.02	1.11	0.22	3.43	0.68	0.00	5.54	0.05	0.02	0.20	4.07	0.81	19.94
%	1.25	16.00	1.76	0.00	0.10	5.57	1.10	17.20	3.41	0.00	27.78	0.25	0.10	1.00	20.41	4.06	100.00
LS w/ VM	0.32	4.63	0.54	0.01	0.09	1.38	0.30	3.50	0.09	0.03	9.18	0.18	0.06	0.22	4.06	0.62	25.21
%	1.27	18.37	2.14	0.04	0.36	5.47	1.19	13.88	0.36	0.12	36.41	0.71	0.24	0.87	16.10	2.46	100.00
LS w/o VM	0.30	4.39	0.51	0.00	0.07	1.36	0.30	3.48	0.07	0.02	9.12	0.17	0.06	0.22	4.04	0.64	24.75
%	1.21	17.74	2.06	0.00	0.28	5.49	1.21	14.06	0.28	0.08	36.85	0.69	0.24	0.89	16.32	2.59	100.00
ALU	0.30	4.36	0.50	0.05	0.13	1.69	0.24	3.47	0.11	0.03	5.53	0.08	0.08	0.24	4.05	0.72	21.58
%	1.39	20.20	2.32	0.23	0.60	7.83	1.11	16.08	0.51	0.14	25.63	0.37	0.37	1.11	18.77	3.34	100.00
IGEMM	0.61	10.17	1.59	0.19	0.65	5.88	0.61	3.84	4.41	0.71	13.75	1.00	0.31	1.12	4.68	2.28	51.80
%	1.18	19.63	3.07	0.37	1.25	11.35	1.18	7.41	8.51	1.37	26.54	1.93	0.60	2.16	9.03	4.40	100.00
Leakage	0.02	0.11	0.02	0.00	0.00	0.12	0.02	0.07	0.08	0.01	0.33	0.04	0.01	0.05	0.00	0.20	1.08



Test Name	IPC
aha-mont64	0,751773
crc32	0,901578
cubic	0,730032
edn	0,837711
huffbench	0,736758
matmult-int	0,889643
minver	0,512024
nbody	0,327144
nettle-aes	0,986432
nettle-sha256	0,862099
nsichneu	0,429473
picojpeg	0,831389
primecount	0,57092
qrduino	0,694356
sglib-combined	0,656466
slre	0,75285
st	0,396254
statemate	0,701807
ud	0,647145
wikisort	0,813702

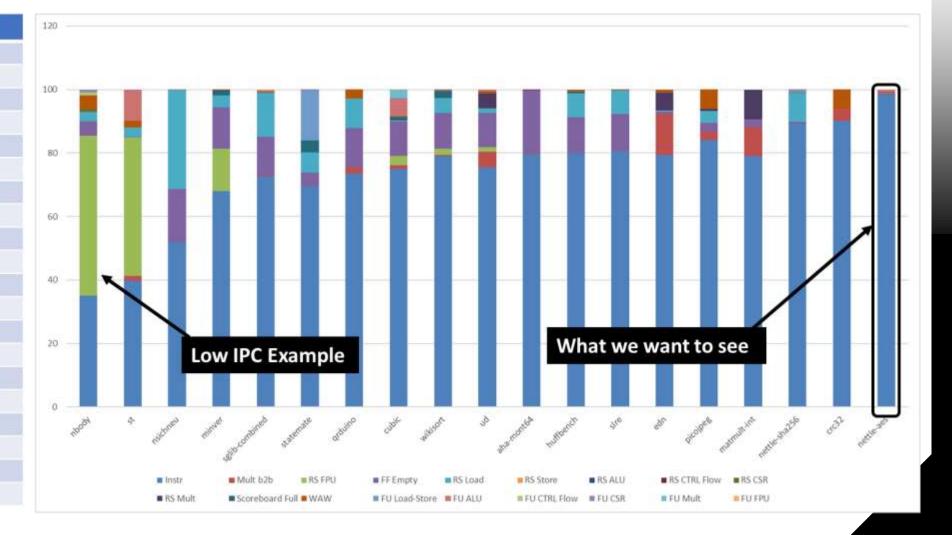


Enhzürich

Test Name	IPC
aha-mont64	0,751773
crc32	0,901578
cubic	0,730032
edn	0,837711
huffbench	0,736758
matmult-int	0,889643
minver	0,512024
nbody	0,327144
nettle-aes	0,986432
nettle-sha256	0,862099
nsichneu	0,429473
picojpeg	0,831389
primecount	0,57092
qrduino	0,694356
sglib-combined	0,656466
slre	0,75285
st	0,396254
statemate	0,701807
ud	0,647145
wikisort	0,813702

Tool Ma

Hzürich

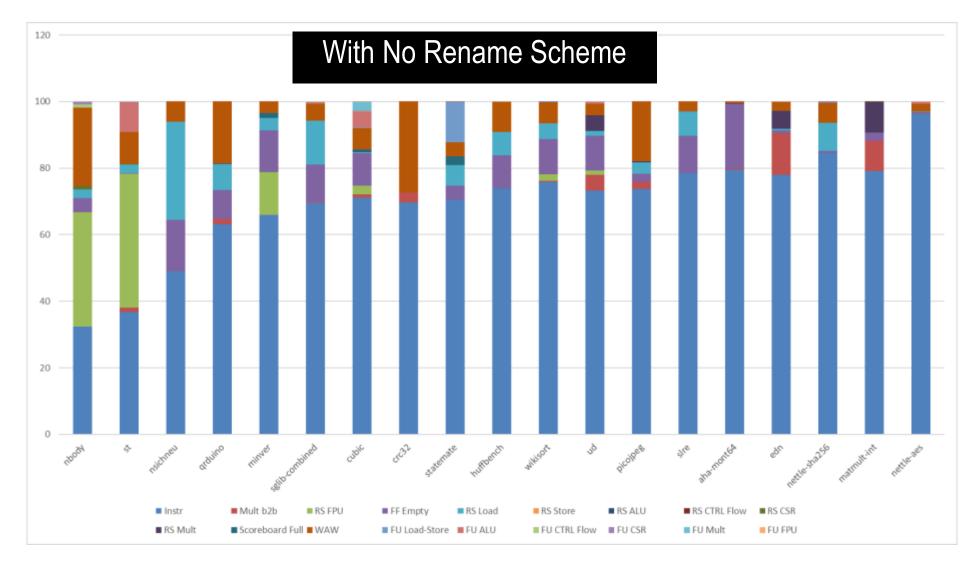


		HEAVY ON DIV/SQRT
Test Name	IPC	120 //
aha-mont64	0,751773	
crc32	0,901578	
cubic	0,730032	
edn	0,837711	
huffbench	0,736758	
matmult-int	0,889643	
minver	0,512024	
nbody	0,327144	4 60
nettle-aes	0,986432	
nettle-sha256	0,862099	
nsichneu	0,429473	
picojpeg	0,831389	
primecount	0,57092	
qrduino	0,694356	
sglib-combined	0,656466	
slre	0,75285	
st	0,396254	and a serie and and and and at and a part of an a serie and
statemate	0,701807	and an
ud	0,647145	Instr Mult b2b RS FPU FF Empty RS Load RS Store RS ALU RS CTRL Flow RS CSR
wikisort	0,813702	RS Mult Scoreboard Full WAW FU Load-Store FU ALU FU CTRL Flow FU CSR FU Mult FU FPU

ETH zürich

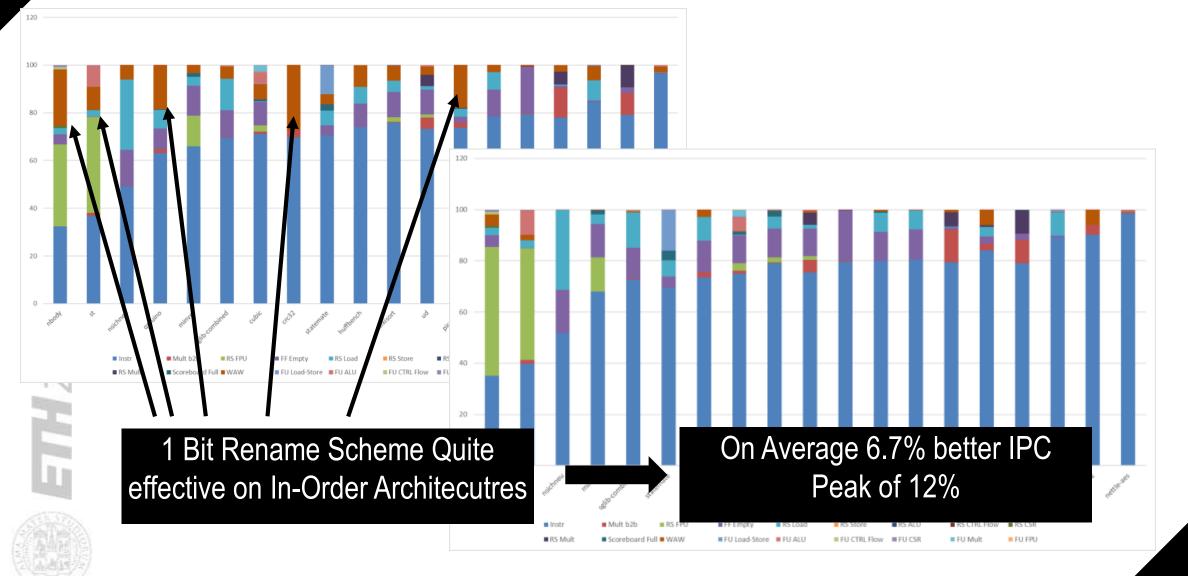
Test Name	IPC
aha-mont64	0,751773
crc32	0,901578
cubic	0,730032
edn	0,837711
huffbench	0,736758
matmult-int	0,889643
minver	0,512024
nbody	0,327144
nettle-aes	0,986432
nettle-sha256	0,862099
nsichneu	0,429473
picojpeg	0,831389
primecount	0,57092
qrduino	0,694356
sglib-combined	0,656466
slre	0,75285
st	0,396254
statemate	0,701807
ud	0,647145
wikisort	0,813702





Entzürich





CVA6 is open-source

- You can find all the RTL of the Core at:
 - https://github.com/openhwgroup/cva6
- CVA6 is just one of the core born from the PULP Project
 - https://pulp-platform.org/
 - https://github.com/pulp-platform





