Out-of-Order Superscalar CPU

Cliff Frey and Vicky Liu

May 6th, 2005

## The Basis of Our Design

Tomasulo's Algorithm

- Allows out-of-order execution

- Instructions wait in Reservation Stations

- Execute instructions once operands have been computed

- Can reorder WAW and WAR

The Basis of Our Design

Tomasulo's Algorithm

- In Decode stage, each instruction result is assigned a Tag

- Each register maps to a Value or to a Tag

- When a result is computed, result and tag are broadcast

- All instances of the Tag are updated with the computed value

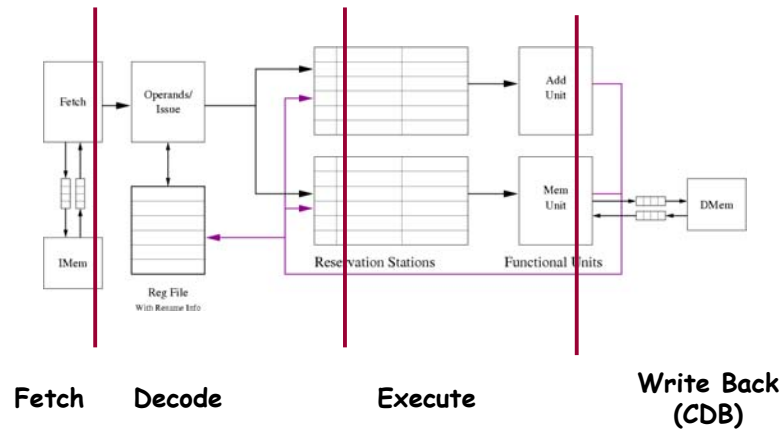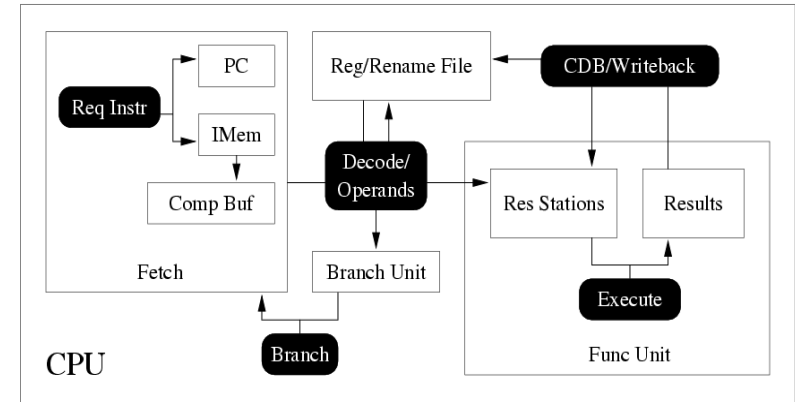- Updates RegFile and Reservation Stations

---

High Level Design
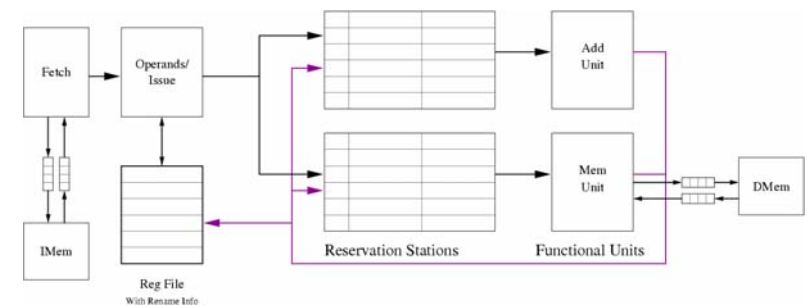
The major components

Fetch Unit	Reservation Stations
Decode	Functional Units
Renaming Register File	Common Data Bus

High Level Design

Fetch   **Decode**   **Execute**   **Write Back (CDB)**

BlueSpec Rule & Module Design

High Level Design Issues

- Unresolved branches stall decode stage

- Memory operations need to be in order

- Back to back dependent adds take 2 cycles

Design Exploration: Supporting Precise Exceptions

Short-comings of Tomasulo's algorithm

- Register File contents can be lost

- external changes need to ordered

Design Exploration: Supporting Precise Exceptions

A Processor Supports Precise Exceptions If…

… instructions before the excepting instruction, execute normally

… instructions after and including the excepting instruction do not change any programmer visible state of the processor

---

Design Exploration: Supporting Precise Exceptions

A Processor Supports Precise Exceptions If…

… instructions before the excepting instruction, execute normally

… instructions after and including the excepting instruction do not change any programmer visible state of the processor
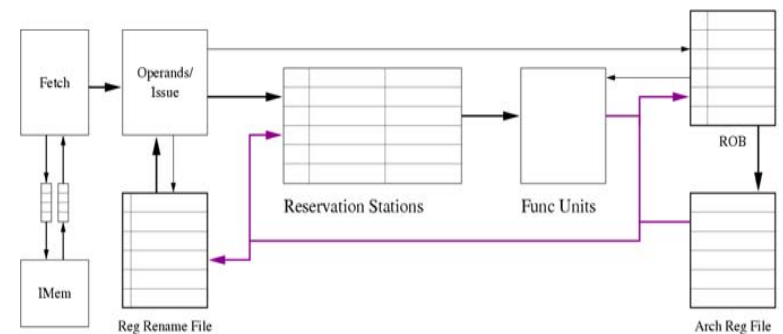
Short-comings of Tomasulo's algorithm

- Register File contents can be lost
- external changes need to ordered
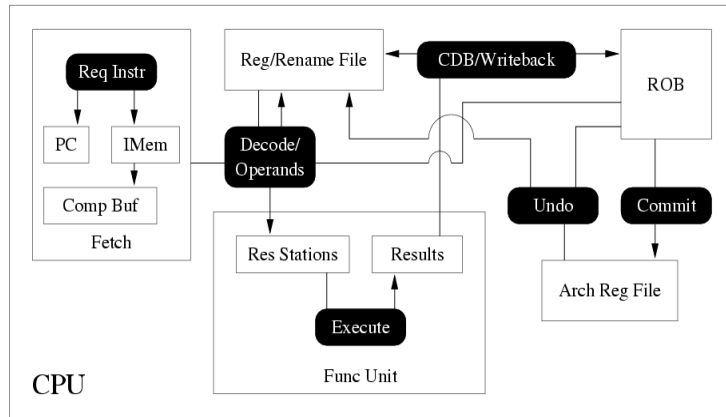

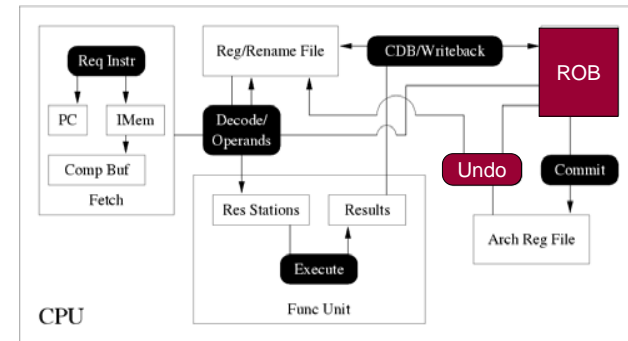## Original High Level Design

---
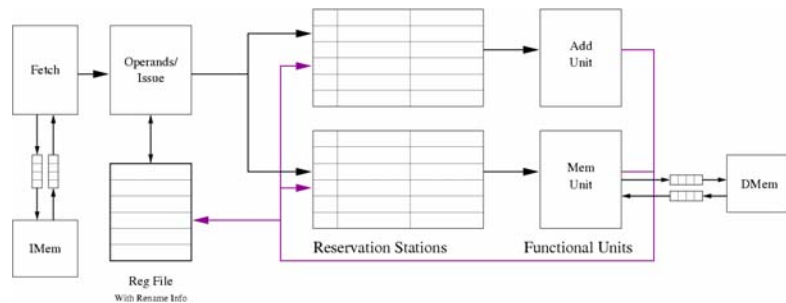
Updated High Level Design

Our Solution
- Minimal changes to original design
- Reorder Buffer (ROB) and Commit stage
- Architectural Register File
- External changes made at commit time

Updated High Level Design

Handling Exceptions

Set PC to interrupt vector (0x1100)

Exception PC stored in coprocessor register EPC

Correct speculative results in Rename Register File

Clear cached information in Functional Units
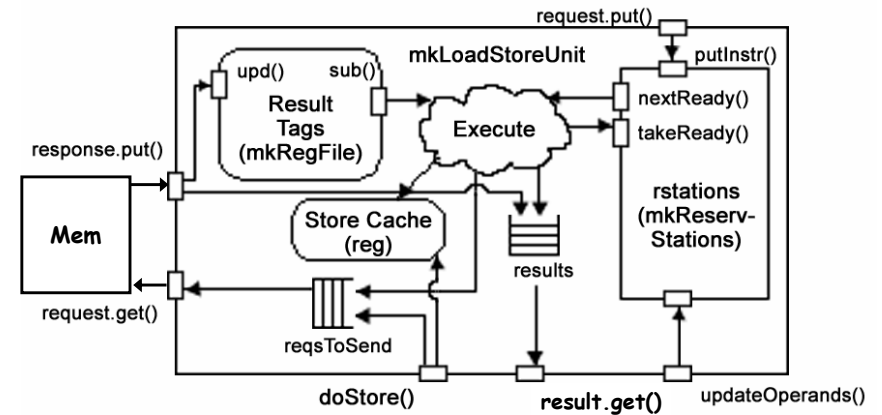
Other Features to Get High Performance

Implemented Features

- Speculative fetch

- external changes need to ordered
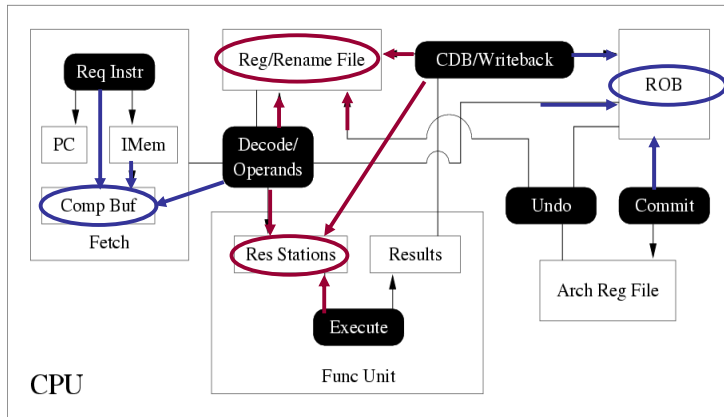
- memory unit can handle many requests at a time

Unimplemented Features

- Branch prediction and target buffering

- Speculative execution

A Closer Look at the Load/Store Unit

BlueSpec Stories: Conflicting Rules

BlueSpec Stories: The Fix

Possible Solutions

- One rule for every possible data path
- Use config regs everywhere
- Be slow and blame BlueSpec =P
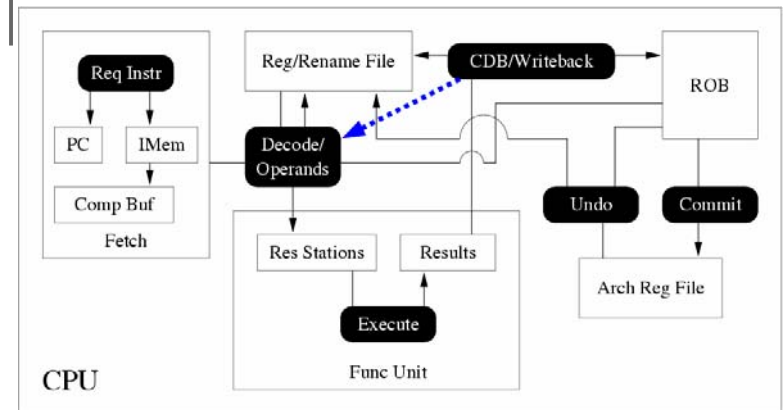
BlueSpec Stories: The Fix

Possible Solutions

- One rule for every possible data path
- Use config regs everywhere
- Be slow and blame BlueSpec =P

Our Solutions

- Homemade completion buffer
- Make methods write to RWires
- Write "magic" rule to handle all combination of cases

Bypassing from writeback to decode

Fetch	Decode	Execute	Writeback	Commit
F	|	[ ]	- -	|
F	|00001000=0 ADD	[ ]	- -	|
F	|00001004=1 ADD	[0]	- -	|
F	|00001008=2 ADD	[1]	A-0 -00000001	|
	|0000100c=3 ADD	[2]	A-1 -00000001	| 0
	|	[3]	A-2 -00000002	| 1
	|	[ ]	A-3 -00000002	| 2
	|	[ ]	- -	| 3

Back to back, nondependent adds

An Excerpt from our Trace Output

Decode			add	mem	BR	WB	commit
001398	LW	r1, r10	[	|M	|	]	|
00139c	ADDI	r2, r2, -4	[	|M LW	|	]	|
0013a0	SLT	r1, r11, r1	[ADDI	|M	|	]	|
0013a4	BEQZ	r1, 0x13d8	[	|M	|	]ADDI	|
0013a8	SUBI	r3, r12, -1	[	|M LW|		]	|
			[SUBI	|M	|	]LW	|
			[SLT	|M	|	]SUBI	|LW
			[	|M	|	]SLT	|ADDI
			[	|M	|BEQZ]		|SLT
			[	|M	|	]BEQZ	|
			[	|M	|	]	|BEQZ *taken!
			[	|M	|	]	|SUBI

Instruction stream with reordering

Synthesis Results
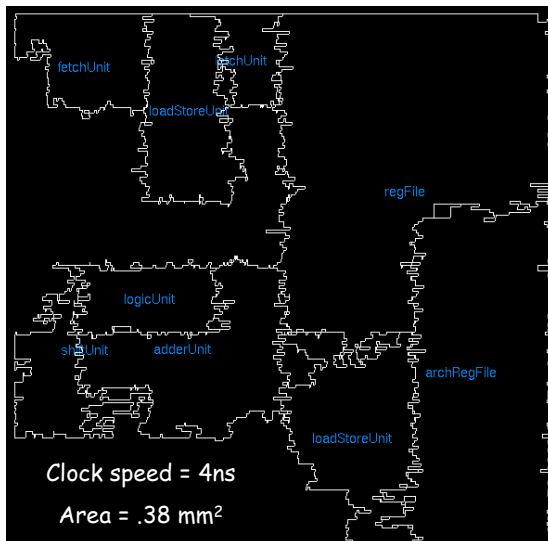
fetchUnit, archUnit, loadStoreUnit, regFile, logicUnit, shiftUnit, adderUnit, archRegFile, loadStoreUnit

Clock speed = 4ns

Area = .38 mm^2

Design Choices and Performance

Configurable Parameters

Resizing reservation stations

Number of slots in ROB and the Fetch Unit buffer

Different functional unit setup

Easily support multicycle functional units

Design Choices and Performance

Configurable Parameters

Resizing reservation stations

Number of slots in ROB and the Fetch Unit buffer

Different functional unit setup

Easily support multicycle functional units

Performance

Branches and stores really hurt performance

Achieved IPC ≈ .5 on vector-add and quicksort