

DLX with a scoreboard

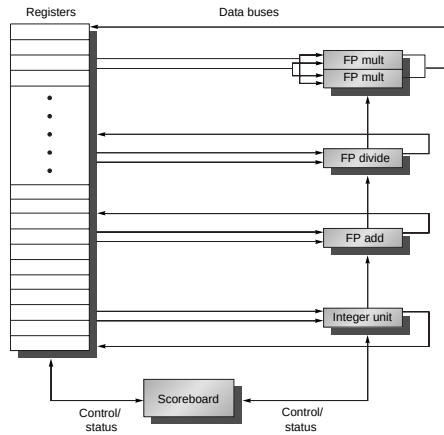


FIGURE 4.3 The basic structure of a DLX processor with a scoreboard.

Scoreboard Summary

- Speedup 1.7 from compiler; 2.5 by hand
BUT slow memory (no cache) limits benefit
- Limitations of 6600 scoreboard:
 - No forwarding hardware
 - Limited to instructions in basic block (small *window*)
 - why?
 - Small number of functional units (structural hazards)
 - insts to same fu cannot be reordered
 - Wait for WAR hazards
 - Prevent WAW hazards

Another Dynamic Algorithm: Tomasulo Algorithm

- For IBM 360/91 about 3 years after CDC 6600
- Goal: High Performance without special compilers
- Differences between IBM 360 & CDC 6600 ISA
 - IBM has only 2 register specifiers/instr vs. 3 in CDC 6600
 - IBM has 4 FP registers vs. 8 in CDC 6600

Differences between Tomasulo Algorithm & Scoreboard

- Control & buffers distributed with Function Units vs. centralized in scoreboard; called “reservation stations”
=> instrs schedule themselves
- Registers in instructions replaced by pointers to reservation station buffer
scoreboard => registers primary operand storage
Tomasulo => reservation stations primary operand storage
- HW renaming of registers to avoid WAR, WAW hazards
Tomasulo => all register reads happen in-order!
- Common Data Bus broadcasts results to all FUs
RS's (FU's), registers, etc. responsible for collecting own data off CDB
- Load and Stores treated as FUs as well

Tomasulo Organization

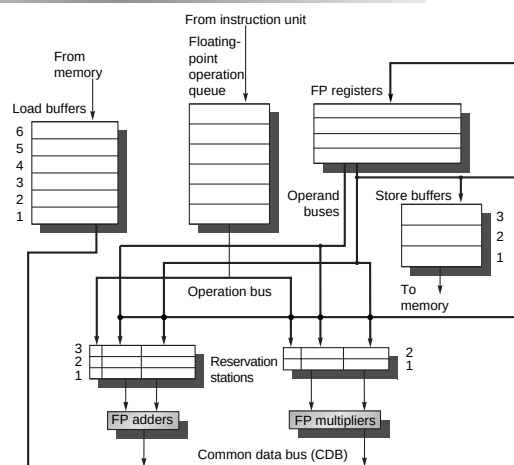


FIGURE 4.8 The basic structure of a DLX FP unit using Tomasulo's algorithm.

Reservation Station Components

Op—Operation to perform in the unit (e.g., + or −)
 Qj, Qk—Reservation stations producing source registers
 Vj, Vk—Value of Source operands
 Rj, Rk—Flags indicating when Vj, Vk are ready
 Busy—Indicates reservation station and FU is busy

Register result status—Indicates which functional unit will write each register, if one exists. Blank when no pending instructions that will write that register.

Three Stages of Tomasulo Algorithm

- Issue**—get instruction from FP Op Queue
 If reservation station free, the scoreboard issues instr & sends operands (renames registers).
- Execution**—operate on operands (EX)
 When both operands ready then execute;
 if not ready, watch CDB for result
- Write result**—finish execution (WB)
 Write on Common Data Bus to all waiting units;
 mark reservation station available.

Tomasulo Example

Instruction status			Execution			Write					
Instruction	j	k	Issue	complete	Result			Busy	Address		
LD F6	34+	R2						Load1	No		
LD F2	45+	R3						Load2	No		
MULTD F0	F2	F4						Load3	No		
SUBD F8	F6	F2									
DIVD F10	F0	F6									
ADD F6	F8	F2									
Reservation Stations			S1	S2	RS for j		RS for k				
	Time	Name	Busy	Op	Vj	Vk	Qj	Qk			
	0	Add1	No								
	0	Add2	No								
		Add3	No								
	0	Mult1	No								
	0	Mult2	No								
Register result status											
Clock			F0	F2	F4	F6	F8	F10	F12	...	F30
		FU									

Tomasulo Example

Instruction status				Execution/Write							
Instruction	j	k		Issue	complete	Result		Busy	Address		
LD F6	34+	R2						Load1	No		
LD F2	45+	R3						Load2	No		
MULTD F0	F2	F4						Load3	No		
SUBD F8	F6	F2									
DIVD F10	F0	F6									
ADDD F6	F8	F2									
Reservation Stations				S1	S2	RS for j	RS for k				
Time	Name	Busy	Op	Vj	Vk	Qj	Qk				
0	Add1	No									
	0 Add2	No									
	Add3	No									
	0 Mult1	No									
	0 Mult2	No									
Register result status											
Clock				F0	F2	F4	F6	F8	F10	F12	... F30
		FU									

CSE 240

Dean Tullsen

Tomasulo Loop Example

Loop: LD	F0	0	R1
MULTD	F4	F0	F2
SD	F4	0	R1
SUBI	R1	R1	#8
BNEZ	R1	Loop	

Multiply takes 4 clocks
Loads have cache misses

CSE 240

Dean Tullsen

Loop Example

Instruction status				Execution/Write							
Instruction	j	k	iteration	Issue	complete	Result		Busy	Address		
LD F0	0	R1	1					Load1	No		
MULTD F4	F0	F2	1					Load2	No		
SD F4	0	R1	1					Load3	No	Qi	
LD F0	0	R1	2					Store1	No		
MULTD F4	F0	F2	2					Store2	No		
SD F4	0	R1	2					Store3	No		
Reservation Stations				S1	S2	RS for j	RS for k				
Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
0	Add1	No						LD	F0	0	R1
0	Add2	No						MULT	F4	F0	F2
0	Add3	No						SD	F4	0	R1
0	Mult1	No						SUBI	R1	R1	#8
0	Mult2	No						BNEZ	R1	Loop	
Register result status											
Clock	R1			F0	F2	F4	F6	F8	F10	F12	... F30
	8 0	Qi									

CSE 240

Dean Tullsen

Loop Example

Instruction status				Execution/Write							
Instruction	j	k	iteration	Issue	complete	Result		Busy	Address		
LD F0	0	R1	1					Load1	No		
MULTD F4	F0	F2	1					Load2	No		
SD F4	0	R1	1					Load3	No	Qi	
LD F0	0	R1	2					Store1	No		
MULTD F4	F0	F2	2					Store2	No		
SD F4	0	R1	2					Store3	No		
Reservation Stations				S1	S2	RS for j	RS for k				
Time	Name	Busy	Op	Vj	Vk	Qj	Qk	Code:			
0	Add1	No						LD	F0	0	R1
0	Add2	No						MULT	F4	F0	F2
0	Add3	No						SD	F4	0	R1
0	Mult1	No						SUBI	R1	R1	#8
0	Mult2	No						BNEZ	R1	Loop	
Register result status											
Clock	R1			F0	F2	F4	F6	F8	F10	F12	... F30
	8 0	Qi									

CSE 240

Dean Tullsen

Tomasulo Summary

- Prevents Register as bottleneck
- Avoids WAR, WAW hazards of Scoreboard
- Allows loop unrolling in HW
- Not limited to basic blocks (provided branch prediction)
- Lasting Contributions
 - Dynamic scheduling
 - Register renaming
 - Load/store disambiguation

Dynamic Scheduling Key Points

- Dynamic scheduling is code motion in HW.
- Dynamic scheduling can do things SW scheduling (static scheduling) cannot.
- Scoreboard, Tomasulo have various tradeoffs
- Register renaming eliminates WAW, WAR dependencies.
- To get cross-iteration parallelism, we need to eliminate WAW, WAR dependencies.

Scoreboard vs. Tomasulo, the score

	<u>Scoreboard</u>	<u>Tomasulo</u>
issue	when FU free	when RS free
read operands	from reg file	from reg file, CDB
write operands	to reg file	to CDB
structural hazards	functional units	reservation stations
WAW, WAR hazards	problem	no problem
register renaming	no	yes
instructions completing	no limit	1 per cycle (per CDB)
instructions beginning exec	1 (per set of read ports)	no limit