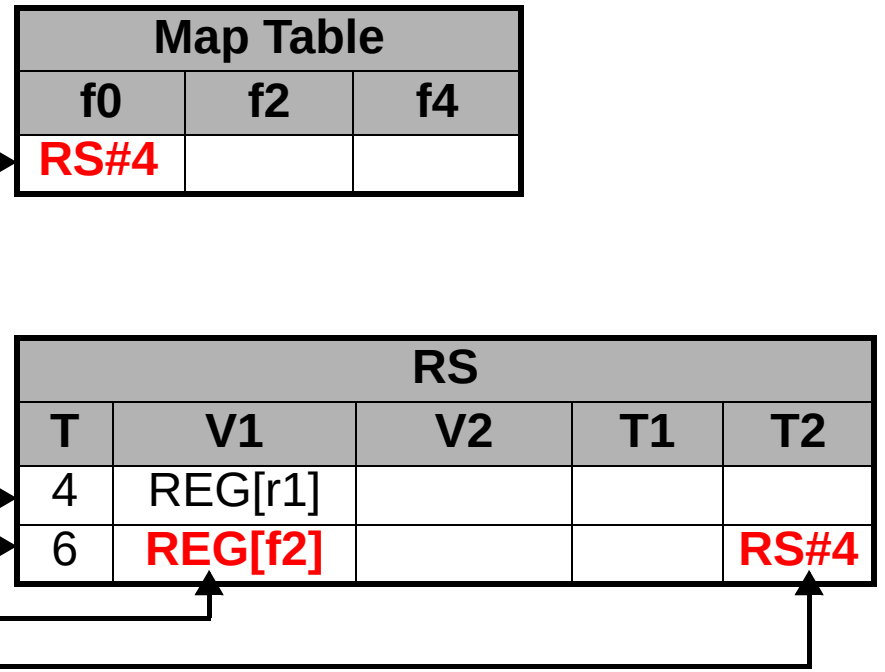
Tomasulo-Style Register Renaming

`ldf f0,X(r1)`

- allocate RS#4
- map **`f0`** to RS#4

`mulf f4,f0,f2`

- allocate RS#6
- **`f2`** ready, copy *value*
- **`f0`** not ready, copy *tag*



`addf f2,f6,f4`

- key: can write **`f2`** before mulf executes (why?)

`ldf f0,X(r1)`

- key: can write **`f0`** before first mulf (why?)

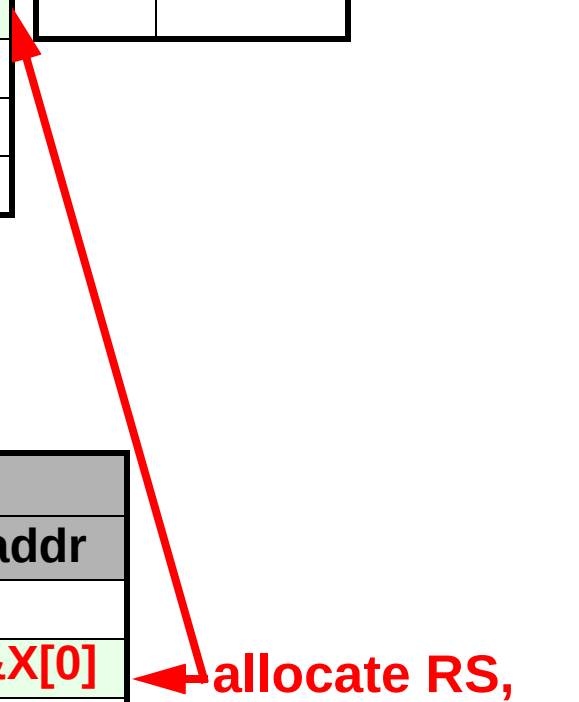
Tomasulo Example: Cycle 1

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1			
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				
`add r1,r1,#4`				
`ldf f0,X(r1)`				
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				

Reg. Status	
reg	T
f0	RS#2
f2	
f4	
r1	

CDB	
V	T

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	Yes	ldf		REG[r1]			&X[0]
3	store	No						
4	FP1	No						
5	FP2	No						



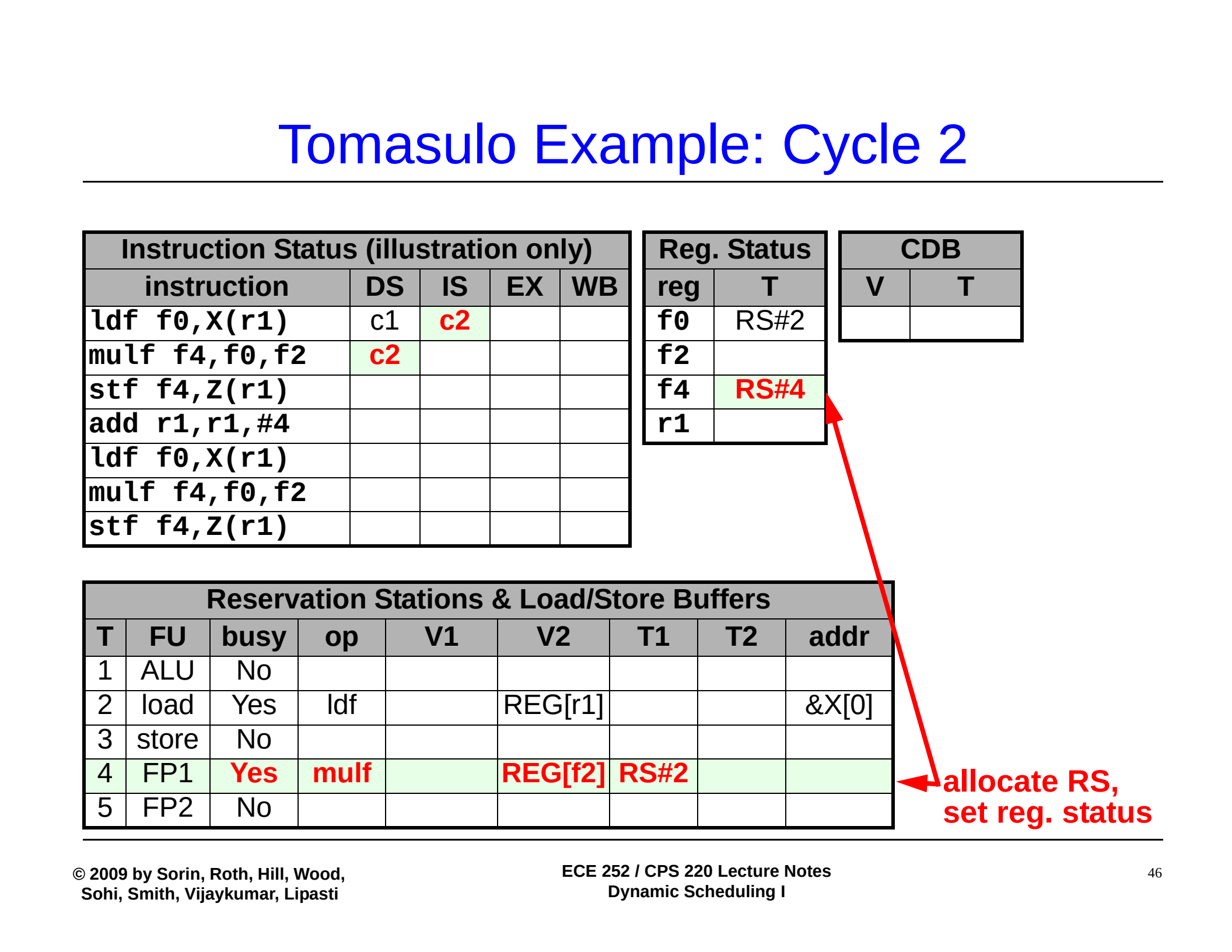
Tomasulo Example: Cycle 2

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2		
`mulf f4,f0,f2`	c2			
`stf f4,Z(r1)`				
`add r1,r1,#4`				
`ldf f0,X(r1)`				
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				

Reg. Status	
reg	T
f0	RS#2
f2	
f4	RS#4
r1	

CDB	
V	T

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	Yes	ldf		REG[r1]			&X[0]
3	store	No						
4	FP1	Yes	mulf		REG[f2]	RS#2		
5	FP2	No						



Tomasulo Example: Cycle 3

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	**c3**	
`mulf f4,f0,f2`	c2			
`stf f4,Z(r1)`	**c3**			
`add r1,r1,#4`				
`ldf f0,X(r1)`				
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				

Reg. Status	
reg	T
`f0`	RS#2
`f2`	
`f4`	RS#4
`r1`	

CDB	
V	T

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	Yes	ldf		REG[r1]			&X[0]
3	store	**Yes**	**stf**		**REG[r1]**	**RS#4**		**&Z[0]**
4	FP1	Yes	mulf		REG[f2]	RS#2		
5	FP2	No						

allocate RS, no reg. status (store)

Tomasulo Example: Cycle 4

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4		
`stf f4,Z(r1)`	c3			
`add r1,r1,#4`	c4			
`ldf f0,X(r1)`				
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				

Reg. Status	
reg	T
f0	~~RS#2~~
f2	
f4	RS#4
r1	RS#1

CDB	
V	T
[f0]	RS#2

ldf finished
1. write/clear status
2. CDB broadcast

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	Yes	add	REG[r1]				
2	load	No						
3	store	Yes	stf		REG[r1]	RS#4		&Z[0]
4	FP1	Yes	mulf	CDB.V	REG[f2]	~~RS#2~~		
5	FP2	No						

allocate RS, set reg. status

f0 is ready grab from CDB

Tomasulo Example: Cycle 5

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	**c5**	
`stf f4,Z(r1)`	c3			
`add r1,r1,#4`	c4	**c5**		
`ldf f0,X(r1)`	**c5**			
`mulf f4,f0,f2`				
`stf f4,Z(r1)`				

Reg. Status	
reg	T
`f0`	**RS#2**
`f2`	
`f4`	RS#4
`r1`	RS#1

CDB	
V	T

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	Yes	add	REG[r1]				
2	load	**Yes**	**ldf**				**RS#1**	
3	store	Yes	stf		REG[r1]	RS#4		&Z[0]
4	FP1	Yes	mulf		REG[f2]			
5	FP2	No						

allocate

Tomasulo Example: Cycle 6

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	
`stf f4,Z(r1)`	c3			
`add r1,r1,#4`	c4	c5	**c6**	
`ldf f0,X(r1)`	c5			
`mulf f4,f0,f2`	**c6**			
`stf f4,Z(r1)`				

Reg. Status	
reg	T
`f0`	RS#2
`f2`	
`f4`	**RS#5**
`r1`	RS#1

CDB	
V	T

no stall on WAW (scoreboard would)

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	Yes	add	REG[r1]				
2	load	Yes	ldf				RS#1	
3	store	Yes	stf		REG[r1]	RS#4		&Z[0]
4	FP1	Yes	mulf		REG[f2]			
5	FP2	**Yes**	**mulf**		**REG[f2]**	**RS#2**		

allocate

Tomasulo Example: Cycle 7

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	
`stf f4,Z(r1)`	c3			
`add r1,r1,#4`	c4	c5	c6	c7
`ldf f0,X(r1)`	c5	c7		
`mulf f4,f0,f2`	c6			
`stf f4,Z(r1)`				

Reg. Status	
reg	T
f0	RS#2
f2	
f4	RS#5
r1	~~RS#1~~

CDB	
V	T
[r1]	RS#1

add finished
1. write/clear status
2. CDB broadcast

no stall on WAR (scoreboard would)

stall DS due to 1 store RS

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	Yes	lf		CDB.V		~~RS#1~~	&X[1]
3	store	Yes	sf		REG[r1]	RS#4		&Z[0]
4	FP1	Yes	mulf		REG[f2]			
5	FP2	Yes	mulf		REG[f2]	RS#2		

r1 (RS#1) is ready
grab CDB value

Tomasulo Example: Cycle 8

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	**c8**
`stf f4,Z(r1)`	c3	**c8**		
`add r1,r1,#4`	c4	c5	c6	c7
`ldf f0,X(r1)`	c5	c7	**c8**	
`mulf f4,f0,f2`	c6			
`stf f4,Z(r1)`				

Reg. Status	
reg	T
`f0`	RS#2
`f2`	
`f4`	RS#5
`r1`	

CDB	
V	T
[f4]	RS#4

mulf finished
1. don't write/clear! this tag is for second mulf (important!)
2. do CDB broadcast

stall DS due to 1 store RS

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	Yes	ldf					&X[1]
3	store	Yes	stf	**CDB.V**	REG[r1]	~~**RS#4**~~		&Z[0]
4	FP1	**No**						
5	FP2	Yes	mulf		REG[f2]	RS#2		

f4 (RS#4) is ready grab value from CDB

free

Tomasulo Example: Cycle 9

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	c8
`stf f4,Z(r1)`	c3	c8	**c9**	
`add r1,r1,#4`	c4	c5	c6	c7
`ldf f0,X(r1)`	c5	c7	c8	**c9**
`mulf f4,f0,f2`	c6	**c9**		
`stf f4,Z(r1)`				

Reg. Status	
reg	T
f0	~~RS#2~~
f2	
f4	RS#5
r1	

CDB	
V	T
[f0]	RS#2

ldf finished
1. write/clear status
2. CDB broadcast

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	**No**						
3	store	Yes	stf		REG[r1]			&Z[0]
4	FP1	No						
5	FP2	Yes	mulf	**CDB.V**	REG[f2]	~~RS#2~~		

free

f0 is ready (CDB)

Tomasulo Example: Cycle 10

Instruction Status (illustration only)				
instruction	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	c8
`stf f4,Z(r1)`	c3	c8	c9	c10
`add r1,r1,#4`	c4	c5	c6	c7
`ldf f0,X(r1)`	c5	c7	c8	c9
`mulf f4,f0,f2`	c6	c9	c10	
`stf f4,Z(r1)`	c10			

Reg. Status	
reg	T
f0	
f2	
f4	RS#5
r1	

CDB	
V	T

stf finished
1. don't write/clear status
2. no CDB broadcast
(this is a store)

DS in same cycle as WB

Reservation Stations & Load/Store Buffers								
T	FU	busy	op	V1	V2	T1	T2	addr
1	ALU	No						
2	load	No						
3	store	Yes	stf		REG[r1]	RS#5		&Z[1]
4	FP1	No						
5	FP2	Yes	mulf		REG[f2]			

free, then allocate

Tomasulo Redux

- what about out-of-order loads and stores?
  - compare load address against the addresses in store buffers
  - wait on a match
  - much more on this tough problem later ...
- RS
  + distributed hazard detection
  + register renaming eliminates WAW hazards
  + copying values to RS eliminates WAR hazards
  – CDB tag matches require many associative compares

Scoreboard vs. Tomasulo

	Scoreboard				Tomasulo			
instruction	DS	IS	EX	WB	DS	IS	EX	WB
`ldf f0,X(r1)`	c1	c2	c3	c4	c1	c2	c3	c4
`mulf f4,f0,f2`	c2	c4	c5+	c8	c2	c4	c5+	c8
`stf f4,Z(r1)`	c3	c8	c9	**c10**	c3	c8	c9	**c10**
`add r1,r1,#4`	c4	c5	c6	c9	c4	c5	c6	c7
`ldf f0,X(r1)`	c5	c9	**c10**		c5	c7	c8	c9
`mulf f4,f0,f2`	c8				c6	c9	**c10**	
`stf f4,Z(r1)`	**c10**				**c10**			

hazard	Scoreboard	Tomasulo
insn buffer	stall in DS	stall in DS
FU	wait in IS	wait in IS
RAW	wait in IS	wait in IS
WAR	wait in WB	none
WAW	stall in DS	none

RS Implementation and Design

- tag/CDB part is called “instruction window” or “scheduler”
  - tag/CDB are universal, values not necessarily (later)
  - two components (can be pipelined)
- (1) wakeup: CDB tag broadcast and match
  - long bus, many comparators
- (2) select: choose instructions to issue this cycle
  - M→N priority encoder
- design: split (separate for each FU) vs. unified (shared)
  - split: +faster, +simpler logic C(N,1), unified: +utilization
- design: FIFO vs. RAM
  - FIFO: +simple (only with split), RAM: +high performance

Superscalar (Wide) Tomasulo

- dynamic scheduling and multiple issue are orthogonal
  - modern processors have both
  - two dimensions
  - **N**: superscalar width (number of parallel operations)
  - **W**: window size (number of reservation stations)
  - sometimes called “max parallelism” and “max reordering”
- what do we need for N-by-W Tomasulo?
  - DS: N RS write ports, 2N RegStatus read ports, N RSt write ports, 2N RS value write ports, 2N RF read ports
  - IS: N RS read ports
  - WB: N CDBs, 2NW comparators, 2N RS value write ports
  - it’s complicated!

Dynamic Scheduling Summary

- dynamic scheduling: out-of-order execution
  - higher utilization, improved performance
  - easier in hardware (we'll see why later in course)
- single F/D latch: structural impediment to OOO execution
  - instruction buffer: multiple F/D latches
  - split ID into in-order DS and out-of-order IS
- DS implementations
  - Scoreboard: out-of-order without renaming
  - Tomasulo: out-of-order with renaming

next up: DS + precise state + speculation