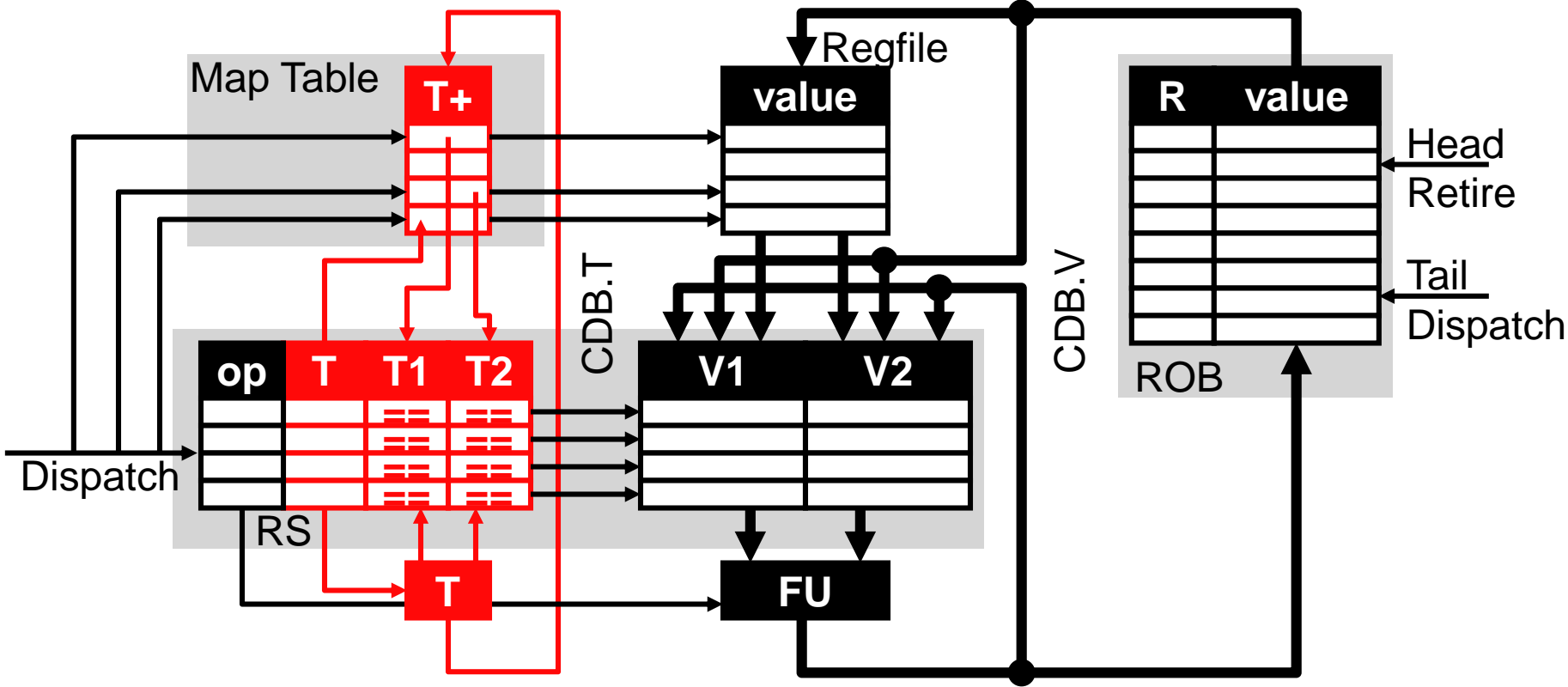


MIPS R10000 (R10K) Out-of-Order Pipeline

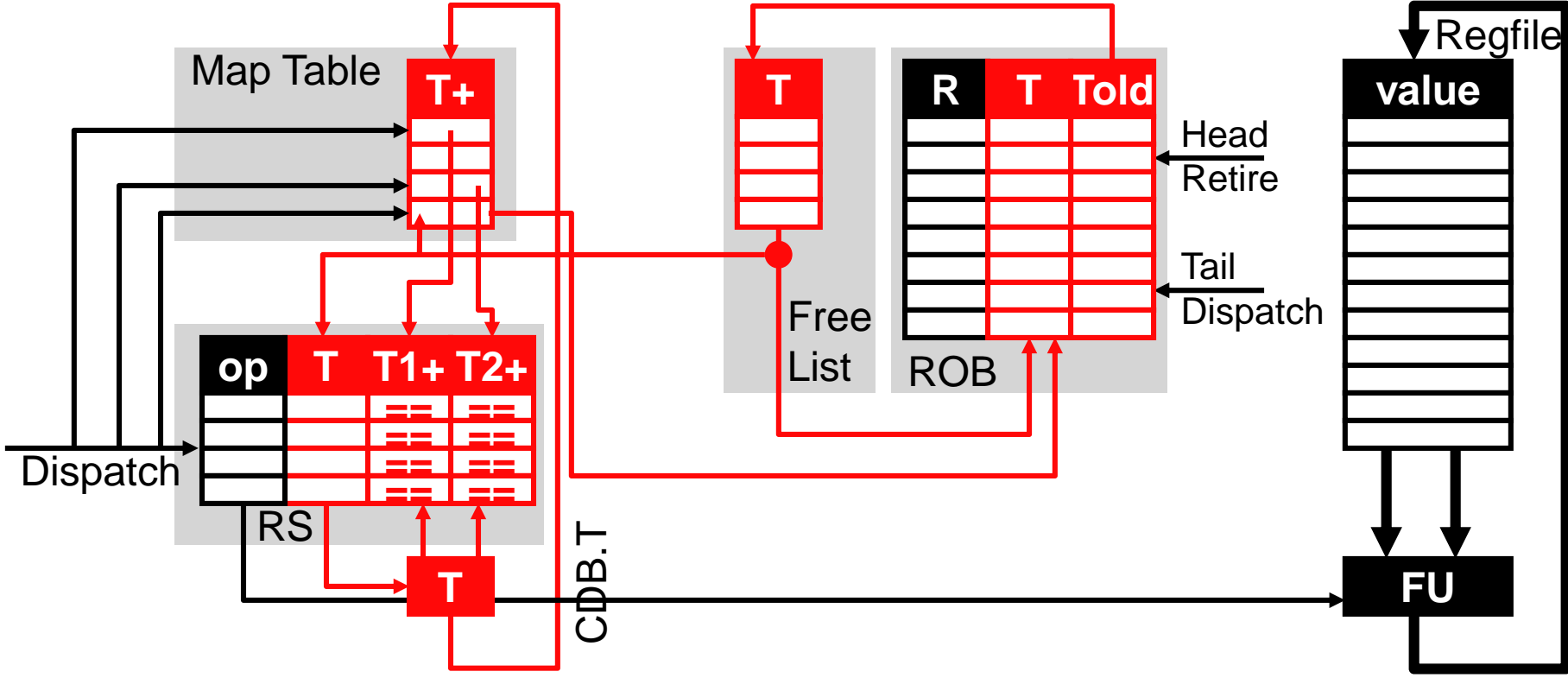
Nima Honarmand

The Problem with P6



- Problem for high performance implementations
 - Too much **value movement** (Regfile/ROB→RS→ROB→Regfile)
 - Multi-input muxes, long buses, slow clock

MIPS R10K: Alternative Implementation



- One big **physical register file** holds all data - no copies
 - + Register file close to FUs → small and fast data path
 - ROB and RS “on the side” used only for control and tags

R10K-Style Register Renaming

- Architectural register file? Gone
- Physical register file holds all values
 - #physical registers = #architectural registers + #ROB entries
 - Map (rename) architectural registers to physical registers
 - No WAW or WAR hazards (physical regs. replace RS values)
- Fundamental change to map table
 - Mappings cannot be 0 (no architectural register file)
- **Explicit free list** tracks unallocated physical regs.
 - **Retire stage** returns physical regs. to free list

Physical Register Reclamation

- P6
 - No need to free speculative (“in-flight”) values explicitly
 - Temporary storage comes with ROB entry
- R10K
 - Can’t free physical regs. when insn. retires
 - Younger insns. likely depend on it
 - But...
 - In Retire stage, can free physical reg. previously mapped to logical *destination* reg.
 - Why?

Freeing Registers

MapTable			FreeList	Original insns.	Renamed insns.
r1	r2	r3	p4, p5, p6, p7	add r2, r3, r1	add p2, p3, p4
p1	p2	p3	p5, p6, p7	sub r2, r1, r3	sub p2, p4, p5
p4	p2	p3	p6, p7	mul r2, r3, r3	mul p2, p5, p6
p4	p2	p5	p7	div r1, 4, r1	div p4, 4, p7
p4	p2	p6	p1	add r1, r3, r2	add p7, p6, p1
p7	p2	p6			

- When ***add*** retires, free p1
- When ***sub*** retires, free p3
- When ***mul*** retires, free p5
- When ***div*** retires, free p4

Always OK to free ***old*** mapping

Hardware Data Structures

- New tags (again)
 - P6: ROB# → R10K: PR# (physical register #)
- ROB
 - **T**: PR# corresponding to insn's logical output
 - **Told**: PR# previously mapped to insn's logical output
- RS
 - **T**, **T1**, **T2**: output, input physical registers
- Map Table
 - **T+**: PR# (never empty) + “ready” bit
- Free List
 - **T**: PR#

No values in ROB, RS, or on CDB

Hardware Data Structures

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)					
	2	f2 = mulf f0, f1					
	3	stf f2, (r1)					
	4	r1 = addi r1, 4					
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#2+
f2	PR#3+
r1	PR#4+

Free List
PR#5, PR#6, PR#7, PR#8

CDB
T

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	no				
4	FP1	no				
5	FP2	no				

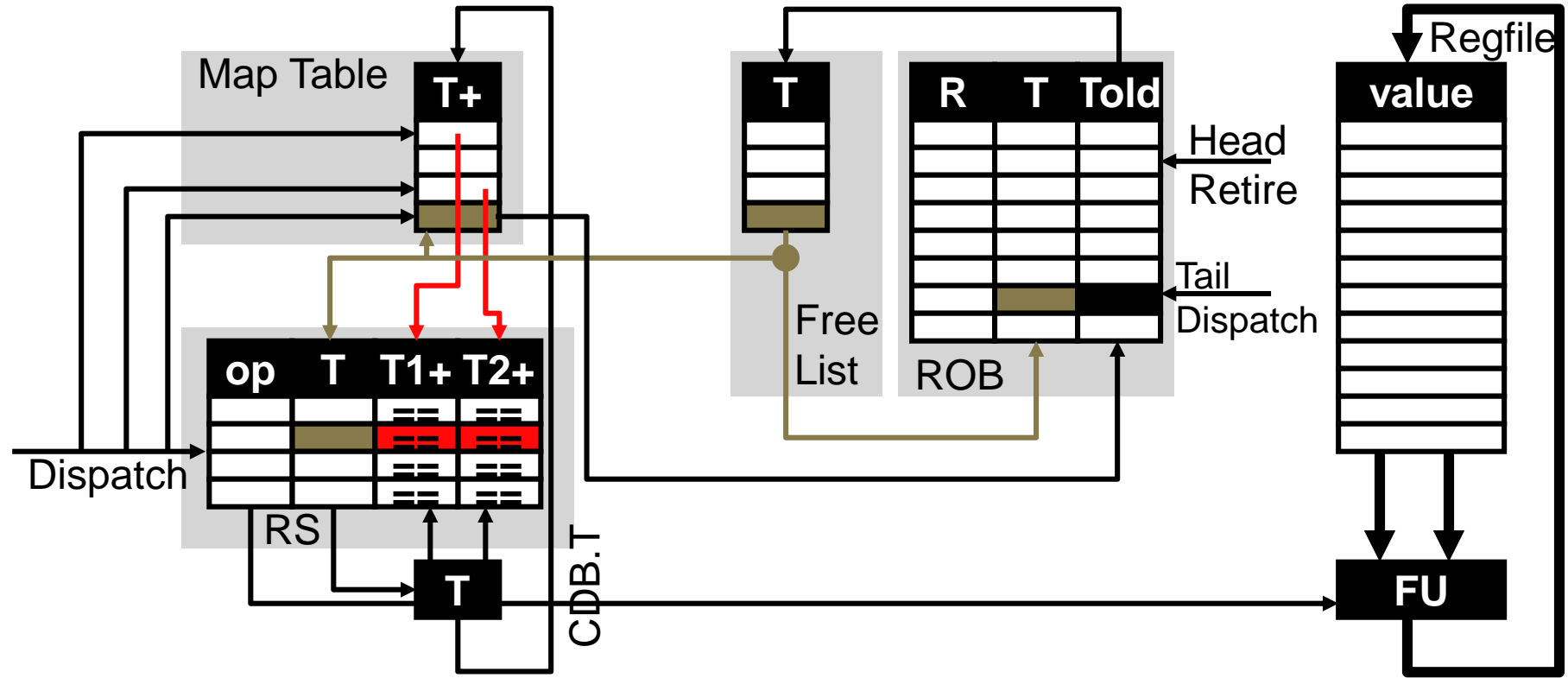
Notice I: no values anywhere

Notice II: MapTable is never empty

R10K Pipeline

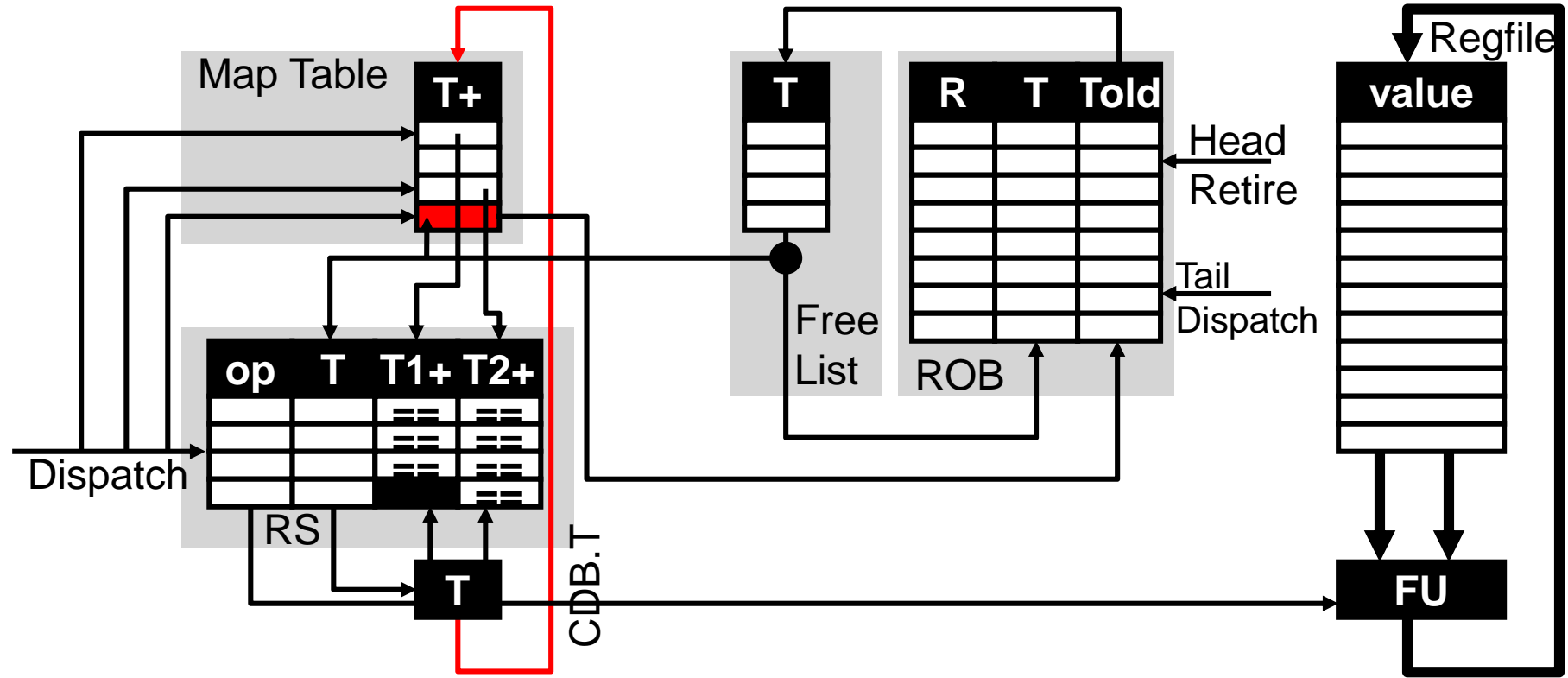
- R10K pipeline structure: F, **D**, S, X, **C**, **R**
 - **D (dispatch)**
 - Structural hazard (RS, ROB, **physical registers**) ? stall
 - Allocate RS, ROB, and new physical register (T)
 - **Record previously mapped physical register (Told)**
 - **C (complete)**
 - Write destination physical register
 - **R (retire)**
 - ROB head not complete ? stall
 - Handle any exceptions
 - Free ROB entry
 - **Free previous physical register (Told)**

R10K Dispatch (D)



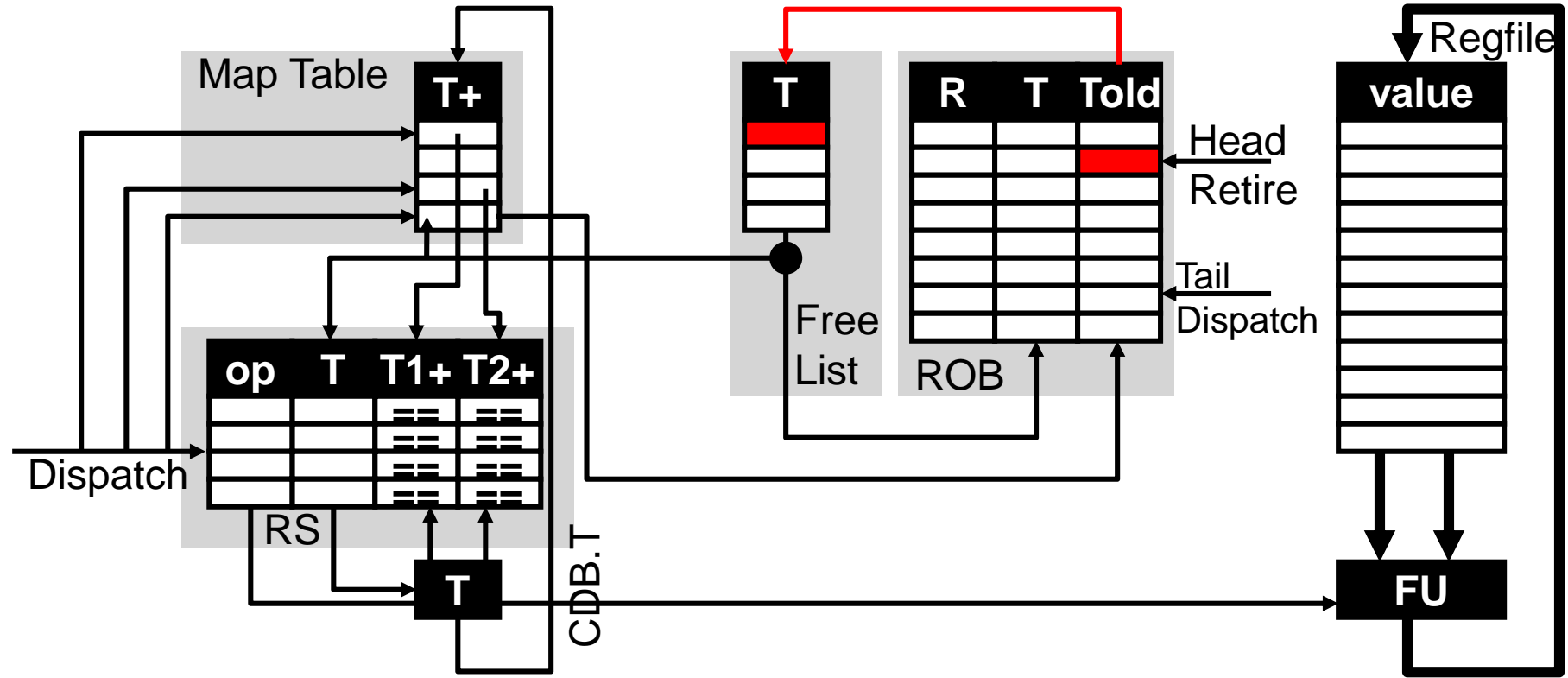
- Read preg (physical register) tags for input registers, store in RS
- Read preg tag for output register, store in ROB (Told)
- Allocate new preg (free list) for output reg, store in RS, ROB, Map Table

R10K Complete (C)



- Set insn's output register ready bit in map table
- Set ready bits for matching input tags in RS

R10K Retire (R)



- Return Told of ROB head to free list

R10K: Cycle 1

ROB							
ht	#	Insn	T	Told	S	X	C
ht	1	f1 = ldf (r1)	PR#5	PR#2			
	2	f2 = mulf f0, f1					
	3	stf f2, (r1)					
	4	r1 = addi r1, 4					
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#3+
r1	PR#4+

CDB
T

Free List
PR#5 , PR#6, PR#7, PR#8

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	yes	ldf	PR#5		PR#4+
3	ST	no				
4	FP1	no				
5	FP2	no				

Allocate new preg (PR#5) to f1

Remember old preg mapped to f1 (PR#2) in ROB

R10K: Cycle 2

ROB							
ht	#	Insn	T	Told	S	X	C
h	1	f1 = ldf (r1)	PR#5	PR#2	c2		
t	2	f2 = mulf f0, f1	PR#6	PR#3			
	3	stf f2, (r1)					
	4	r1 = addi r1, 4					
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#6
r1	PR#4+

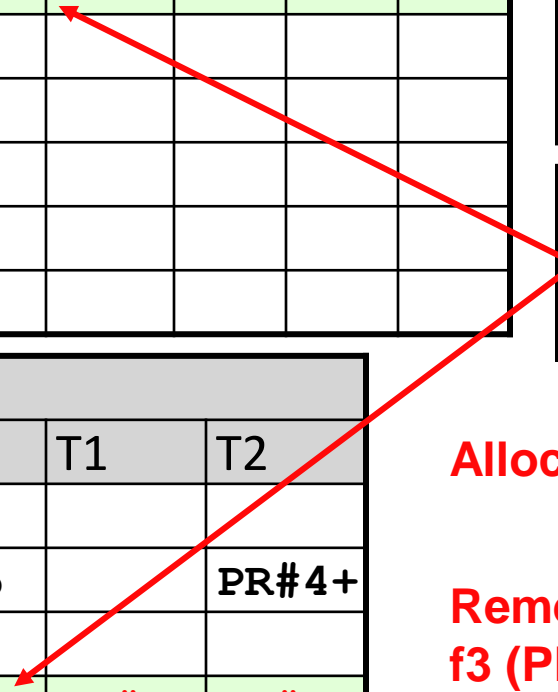
CDB
T

Free List
PR#6, PR#7, PR#8

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	yes	ldf	PR#5		PR#4+
3	ST	no				
4	FP1	yes	mulf	PR#6	PR#1+	PR#5
5	FP2	no				

Allocate new preg (PR#6) to f2

Remember old preg mapped to f3 (PR#3) in ROB



R10K: Cycle 3

ROB							
ht	#	Insn	T	Told	S	X	C
h	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	
	2	f2 = mulf f0, f1	PR#6	PR#3			
t	3	stf f2, (r1)					
	4	r1 = addi r1, 4					
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5
f2	PR#6
r1	PR#4+

CDB
T

Free List
PR#7, PR#8

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	yes	mulf	PR#6	PR#1+	PR#5
5	FP2	no				

Stores are not allocated pregs
free

R10K: Cycle 4

ROB							
ht	#	Insn	T	Told	S	X	C
h	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
	2	f2 = mulf f0, f1	PR#6	PR#3	c4		
	3	stf f2, (r1)					
t	4	r1 = addi r1, 4	PR#7	PR#4			
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#7

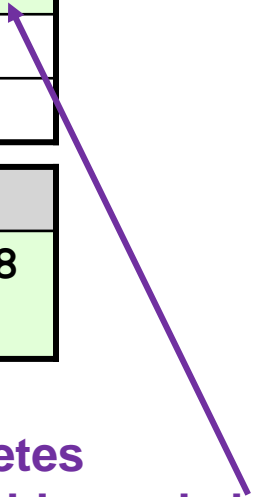
CDB
T
PR#5

Free List
<u>PR#7</u> , PR#8

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	yes	mulf	PR#6	PR#1+	PR#5+
5	FP2	no				

ldf completes set MapTable ready bit

match PR#5 tag from CDB & issue



R10K: Cycle 5

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
h	2	f2 = mulf f0, f1	PR#6	PR#3	c4	c5	
	3	stf f2, (r1)					
	4	r1 = addi r1, 4	PR#7	PR#4	c5		
t	5	f1 = ldf (r1)	PR#8	PR#5			
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#8
f2	PR#6
r1	PR#7

CDB
T

Free List
<u>PR#8</u> , PR#2

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	yes	ldf	PR#8		PR#7
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

free

ldf retires
Return PR#2 to free list

Precise State in R10K

- Precise state is more difficult in R10K
 - Physical registers are written out-of-order (at C)
 - To recover precise state, roll back the Map Table and Free List
 - “free” written registers and “restore” old ones
- Two ways of restoring Map Table and Free List
 - Option I: serial rollback using T , T_{old} ROB fields
 - ± Slow, but simple
 - Option II: single-cycle restoration from some checkpoint
 - ± Fast, but checkpoints are expensive
 - Modern processor compromise: **make common case fast**
 - Checkpoint only for branch prediction (frequent rollbacks)
 - Serial recovery for exceptions and interrupts (rare rollbacks)

R10K: Cycle 5 (with precise state)

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
h	2	f2 = mulf f0, f1	PR#6	PR#3	c4	c5	
	3	stf f2, (r1)					
	4	r1 = addi r1, 4	PR#7	PR#4	c5		
t	5	f1 = ldf (r1)	PR#8	PR#5			
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#8
f2	PR#6
r1	PR#7

CDB
T

Free List
<u>PR#8</u> , PR#2

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	yes	ldf	PR#8		PR#7
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

undo insns 3-5
(doesn't matter why)
use serial rollback

R10K: Cycle 6 (with precise state)

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
h	2	f2 = mulf f0, f1	PR#6	PR#3	c4	c5	
	3	stf f2, (r1)					
t	4	r1 = addi r1, 4	PR#7	PR#4	c5		
	5	f1 = ldf (r1)	PR#8	PR#5			
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#7

CDB
T

Free List
PR#2, PR#8

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	yes	addi	PR#7	PR#4+	
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

- undo ldf (ROB#5)**
1. free RS
 2. free T (PR#8), return to Free List
 3. restore MT[f1] to Told (PR#5)
 4. free ROB#5

R10K: Cycle 7 (with precise state)

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
h	2	f2 = mulf f0, f1	PR#6	PR#3	c4	c5	
t	3	stf f2, (r1)					
	4	r1 = addi r1, 4	PR#7	PR#4	c5		
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#4+

CDB
T

Free List	
PR#2, PR#8,	
PR#7	

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	yes	stf		PR#6	PR#4+
4	FP1	no				
5	FP2	no				

- undo addi (ROB#4)**
1. free RS
 2. free T (PR#7), return to Free List
 3. restore MT[r1] to Told (PR#4)
 4. free ROB#4

R10K: Cycle 8 (with precise state)

ROB							
ht	#	Insn	T	Told	S	X	C
	1	f1 = ldf (r1)	PR#5	PR#2	c2	c3	c4
ht	2	f2 = mulf f0, f1	PR#6	PR#3	c4	c5	
	3	stf f2, (r1)					
	4	r1 = addi r1, 4					
	5	f1 = ldf (r1)					
	6	f2 = mulf f0, f1					
	7	stf f2, (r1)					

Map Table	
Reg	T+
f0	PR#1+
f1	PR#5+
f2	PR#6
r1	PR#4+

CDB
T

Free List
PR#2, PR#8, PR#7

Reservation Stations						
#	FU	busy	op	T	T1	T2
1	ALU	no				
2	LD	no				
3	ST	no				
4	FP1	no				
5	FP2	no				

- undo stf (ROB#3)**
1. free RS
 2. free ROB#3
 3. no registers to restore/free
 4. how is L1-D write undone?

Renaming & OoO in P6 vs. R10K

Feature	P6	R10K
Value storage	ARF,ROB,RS	PRF
Register read	@D: ARF/ROB → RS	@S: PRF → FU
Register write	@R: ROB → ARF	@C: FU → PRF
Speculative value free	@R: automatic (ROB)	@R: overwriting insn
Data paths	ARF/ROB → RS RS → FU FU → ROB, RS ROB → ARF	PRF → FU FU → PRF
Precise state	Simple: clear everything	Complex: serial/checkpoint

- R10K-style became popular in late 90's, early 00's
 - E.g., MIPS R10K (duh), DEC Alpha 21264, Intel Pentium 4
- P6-style is making a comeback
 - Why? Frequency (power) is on the retreat, simplicity is important