

Architetture Out-of-order

Andrea Bartolini <a.bartolini@unibo.it>

(Architettura dei) Calcolatori Elettronici, 2021/2022

Concept of dyanamic scheduling

1. `fdiv.d f0, f2, f4`

2. `fadd.d f10, f0, f8`

3. `fsub.d f12, f8, f14`

Concept of dynamic scheduling

[illegible]

Concept of dynamic scheduling

	T1	T2	T3	T4	T5	T6	T7	T9	T10	T11	
fdiv.d f0 , f2, f4	IF	ID	EX	EX							
fadd.d f10, f0 , f8		IF	ID	ID							
fsub.d f12, f8, f14			IF	IF							

- True dependency on f0
- fsub.d stalls yet isn't dependent on fadd.d nor fdiv.d
- Suppose we let fsub.d move around the stall and execute out of order?

Pipeline bloccanti / in-order

✓ Simple pipelines fetch an instruction and issue it

✗ Unless there is a data dependence between an instruction already in the pipeline and the fetched instruction that cannot be hidden with bypassing or forwarding.

- Forwarding logic reduces the effective pipeline latency so that certain dependences do not result in hazards.
- If unavoidable hazard => the hazard detection hardware stalls the pipeline
- No new instructions are fetched or issued until the dependence is cleared.
- Structural hazards and data hazards are checked in the ID stage.
- Static Scheduling:
 - The compiler reorders instructions to avoid/reduce stalls.
- Dynamic Scheduling:
 - The hardware rearranges the instruction execution to avoid/reduce stalls.
 - Handles dependencies not known at compile time
 - The same binary runs efficiently on multiple architectures

Pipeline non bloccanti / out-of-order

We want an instruction to begin execution as soon as its operands are available “even if a predecessor is stalled”.

=> separate the issue process into two parts:

- checking the structural hazards
- waiting for the absence of a data hazard.

1. Decode and issue instructions in order, but
2. Execute instructions as soon as their data operands are available
=> out-of-order execution => out-of-order completion.

Pipeline non bloccanti / out-of-order

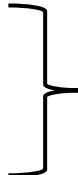
The pipeline will do out-of-order execution, which implies out-of-order completion.

⇒ ID stage split into 2 stages:

1. Issue: Decode instruction, check for structural hazards.
2. Read Operands: Wait for data hazard to be resolved, read the operands

⇒ IF unchanged.

⇒ EX considered multicycle, Two phases:

- Instruction begin execution
 - Instruction complete execution
- 
- Between two times an instruction is in execution!

Scoreboarding

Dynamically scheduled pipeline (in-order issue)

- ⇒ all instructions pass through the issue stage in order but,
- ⇒ they can be stalled or bypass each other in the second stage (read operands) and thus enter execution out of order.

Scoreboarding

- ⇒ allows instructions to execute out of order when sufficient resources & no data dependences
- ⇒ records data dependences
 - i. Corresponds to instruction issue
 - ii. Replaces part of the ID step in the RISC V pipeline
- ⇒ determines when the instruction can read its operands and begin execution.
- ⇒ decides if the instruction cannot execute immediately
 - ⇒ it monitors every change in the hardware and decides when the instruction can execute.
- ⇒ controls when an instruction can write its result into the destination register.
- ⇒ All hazard detection and resolution are centralized in the scoreboard.

Scoreboarding

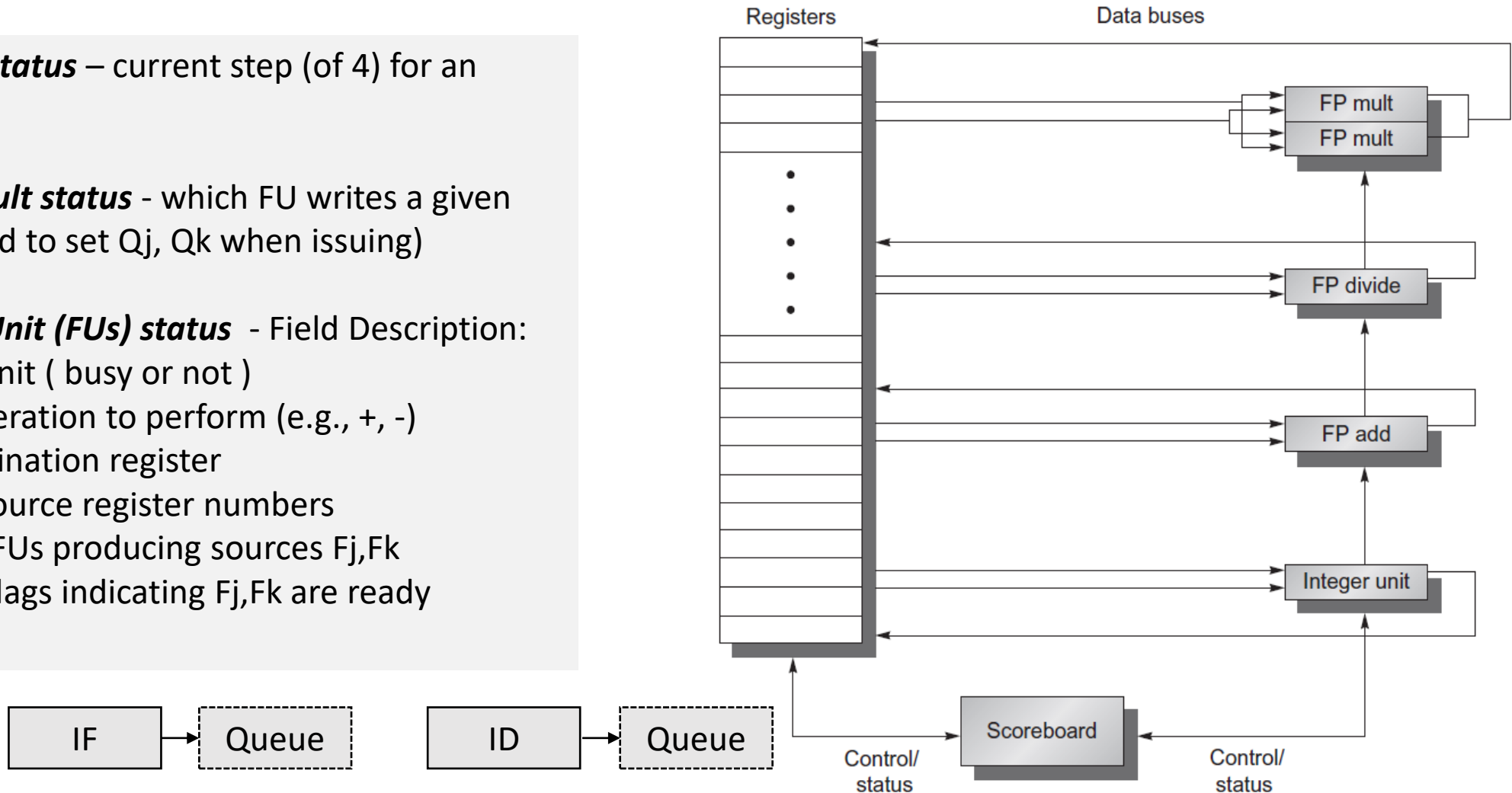
- Keep track of instructions, functional units, and registers to handle hazards

Goal: CPI=1

- “Pull out” independent instructions later in the “issue window”.
- Wait queue after Issue to hold stalled instructions waiting for operands
 - It may not really exist => implemented by the scoreboard!
- Multiple or pipelined functional units
 - recall: during issue, we stall on a structural hazard

Scoreboard:

1. **Instruction status** – current step (of 4) for an instruction
2. **Register result status** - which FU writes a given register (used to set Qj, Qk when issuing)
3. **Functional Unit (FUs) status** - Field Description:
 - i. *Busy Unit* (busy or not)
 - ii. *Op* Operation to perform (e.g., +, -)
 - iii. *Fi* Destination register
 - iv. *Fj,Fk* Source register numbers
 - v. *Qj,Qk* FUs producing sources *Fj,Fk*
 - vi. *Rj,Rk* Flags indicating *Fj,Fk* are ready



Stage of Scoreboard Control

1. Issue (I)

- decode instruction, check for structural and WAW hazards, stall when necessary

2. Read Operands (RO)

- wait until no RAW hazards, then read operands, send operation to FU

3. Execution (EX)

- FU starts execution upon receiving the operands & notifies scoreboard when it's completed

4. Write Result (WR)

- Scoreboard checks for WAR hazards and stalls write to register file to avoid them

Hazard Detection:

Structural hazards (I):

- ensure that a functional unit is available (makes a “reservation”)

WAW hazards (I):

- ensure that no previous active instruction has the same destination

RAW hazards (RO):

- check that no previous active instruction writes a source register

WAR hazards (WR):

- before writing a result, check if any previous instructions that haven't gone past RO need that register as a source

Scoreboard pipeline control

Status

Issue

Wait until

FU not busy
&& not result

Bookkeeping

Busy(FU) \leftarrow Y; Op(FU) \leftarrow op; Fi(FU) \leftarrow D;
Fj(FU) \leftarrow S1; Fk(FU) \leftarrow S2; Qj \leftarrow Res(S1);
Qk \leftarrow Res(S2); Rj \leftarrow !Qj; Rk \leftarrow !Qk;
Res(D) \leftarrow FU

Read ops

Rj && Rk

Rj \leftarrow N; Rk \leftarrow N

Executed

FU done

Write dest

$\forall f((Fj(f) \neq Fi(FU)) \mid \mid$
Rj(f) = N) &&
(Fk(f) \neq Fi(FU) $\mid \mid$
Rk(f) = N))

$\forall f(\text{if } Qj(f) = FU, Rj(f) \leftarrow Y);$
 $\forall f(\text{if } Qk(f) = FU, Rj(f) \leftarrow Y);$
Result(Fi(FU)) \leftarrow 0; Busy(FU) \leftarrow N

Write Destination

Wait Until

$\forall f((Fj(f) \neq Fi(FU)) \mid \mid$
 $Rj(f) = N) \ \&\&$
 $(Fk(f) \neq Fi(FU)) \mid \mid$
 $Rk(f) = N))$

Bookkeeping

$\forall f(\text{if } Qj(f) = FU, Rj(f) \leftarrow Y);$
 $\forall f(\text{if } Qk(f) = FU, Rj(f) \leftarrow Y);$
 $\text{Result}(Fi(FU)) \leftarrow 0; \text{Busy}(FU) \leftarrow N$

Wait until condition says:

$Fj(f) \neq Fi(FU)$
 $Rj(f) = N$

does this FU write a result used by another FU?
is the other FU waiting for this result?

Bookkeeping says:

$\text{if } Qj(f) = FU, Rj(f) \leftarrow Y$
 $\text{Result}(Fi(FU)) \leftarrow 0;$

set register ready for all consumer FUs
clear entry in the register result table

Scoreboard example

Code Sequence:

```
ld.d    f6, 34(R2)
ld.d    f2, 45(R3)
fmul.d  f0, f2, f4
fsub.d  f8, f6, f2
fdiv.d  f10, f0, f6
fadd.d  f6, f8, f2
```

Functional Units:

- 1 Integer, 2 FP multipliers, 1 FP adder, 1 FP divider

Pipeline Latencies:

- ld: 1 cycle <= uses Integer unit
- fmul: 10 cycles <=used FP multiplier
- fsub, fadd: 2 cycles <= uses FP adder
- fdiv: 40 cycles <= uses FP divider

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2				
ld.d	F2	45+	R3				
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30

CLOCK:

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1			
ld.d	F2	45+	R3				
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	D	S1	S2	F1	F2	R1	R2
				Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Int								

CLOCK: 1) Fill in FU status, mark FU producing results.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2		
ld.d	F2	45+	R3				
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
				Int					

CLOCK: 2) Issue 2° ld ?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	
ld.d	F2	45+	R3				
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
				Int					

CLOCK: 3) Single cycle load completes. What if we have a multi-cycle load?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3				
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F6		R2				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
				Int					

CLOCK: 4) load completes, write result (WAR?) .

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5			
fmul.d	F0	F2	F4				
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU		Int							

CLOCK: 5) structural hazard for Int unit resolved. 2nd load can issue, FMUL is fetched.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
Ld.d	F6	34+	R2	1	2	3	4
Ld.d	F2	45+	R3	5	6		
fmul.d	F0	F2	F4	6			
fsub.d	F8	F6	F2				
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1	Int							

CLOCK: 6) - LD operands available, issue FMUL. Can the FSUB issue on the next cycle?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	
fmul.d	F0	F2	F4	6			
fsub.d	F8	F6	F2	7			
fdiv.d	F10	F0	F6				
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Int	Yes	No
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1	Int			Add				

CLOCK: 7) - LD finishes, FMUL stalls (why?)

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
Ld.d	F6	34+	R2	1	2	3	4
Ld.d	F2	45+	R3	5	6	7	
fmul.d	F0	F2	F4	6			
fsub.d	F8	F6	F2	7			
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	Yes	Ld	F2		R3				Yes
	Mult1	Yes	Mult	F0	F2	F4	Int		No	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2		Int	Yes	No
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

	F0	F2	F4	F6	F8	F10	F12	...	F30
FU	Mult1	Int			Add	Div			

CLOCK: 8(a) - immediately before LD completes.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6			
fsub.d	F8	F6	F2	7			
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1				Add	Div			

CLOCK: 8(b) (the LD completes. FMUL and FADD ready.)

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9		
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
10	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
2	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1				Add	Div			

CLOCK: 9) - FMUL and FSUB's operands ready, go! Can the FADD issue next cycle? What happens?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
8	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
0	Add	Yes	Sub	F8	F6	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1				Add	Div			

CLOCK: 11) - FSUB finishes before FMUL. Can FSUB write its result?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2				

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
7	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	No								
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1								Div

CLOCK: 12) - FSUB completes before FMUL. Can we read operands for FDIV? What about FADD?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13			

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
6	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 13) - FADD structural hazard cleared.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14		

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
5	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
2	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 14) - FADD reads operands.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14		

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
4	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
1	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 15)

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
2	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 16) - FADD finishes.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9		
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
1	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 18) - Yikes! Add FU still occupied?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
0	Integer	No								
	Mult1	Yes	Mult	F0	F2	F4			Yes	Yes
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6	M1		No	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Mult1			Add		Div			

CLOCK: 19) - FMUL finishes after FSUB.

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	20
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8			
fadd.d	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
				Add		Div			

CLOCK: 20) - FMUL completes, FDIV ready to go

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	20
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8	21		
fadd.d	F6	F8	F2	13	14	16	

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	Yes	Add	F6	F8	F2			Yes	Yes
	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
				Add		Div			

CLOCK: 21 - FDIV has operands. Can FADD now complete on next cycle?

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	20
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8	21		
fadd.d	F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
40	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Div								

CLOCK: 22) FDIV executes, FADD writes result .

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	20
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8	21	61	
fadd.d	F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
0	Divide	Yes	Div	F10	F0	F6			Yes	Yes

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30
	Div								

CLOCK: 61) FDIV finishes

Instruction status

Op	d	j	k	Issue	Read	Finish	Write
ld.d	F6	34+	R2	1	2	3	4
ld.d	F2	45+	R3	5	6	7	8
fmul.d	F0	F2	F4	6	9	19	20
fsub.d	F8	F6	F2	7	9	11	12
fdiv.d	F10	F0	F6	8	21	61	62
fadd.d	F6	F8	F2	13	14	16	22

Functional unit status

Time	Name	Busy	Op	Fi	Fj	Fk	Qj	Qk	Rj	Rk
	Integer	No								
	Mult1	No								
	Mult2	No								
	Add	No								
	Divide	No								

Register result status

FU	F0	F2	F4	F6	F8	F10	F12	...	F30

CLOCK: 62 (FDIV writes its result back)

Scoreboarding: Pro and Contra

Structural hazards possible on buses

- Buses to/from register file may be limited
- Ensure there are not more instructions “in flight” than can successfully fetch registers

No forwarding

- Operands are read only when both are available in the register file
- Fortunately, instructions write into the register file right away

Summary:

- Stalls on I if WAW or structural hazards.
- Stalls on RO if RAW
- Stalls on WR if WAR