technology
from seed

What is so special in the Tomasulo Hardware Algorithm?

Leonel Sousa

INSTITUTO SUPERIOR TÉCNICO

inesc id lisboa

Outline

technology
from seed

1. Motivation and main objective

2. Tomasulo hardware algorithm

3. Impact in current processors and systems

- Robert Tomasulo life and OoO history

Motivation and main objective

technology
from seed

1. Motivation and main objective
2. Tomasulo hardware algorithm
3. Impact in current processors and systems
4. Robert Tomasulo life and OoO history

Motivation and main objective

- Computers in the late 60's start having several execution units
  - First step toward exploiting concurrency was to divide execution into two independent parts: <u>fixed-point</u> *vs* <u>floating-point</u>
  - However the program must contain a balanced mixture of both type of instructions, for this scheme to be effective

- Therefore, the main objective of the Tomasulo's paper is
  - ***to achieve concurrent execution of floating-point instructions in the IBM System/360 Model 91***
  - In the case of the IBM System/360 Model 91, an *adder* and a *multiplier/divider* *!!*

Motivation and main objective

technology
from seed

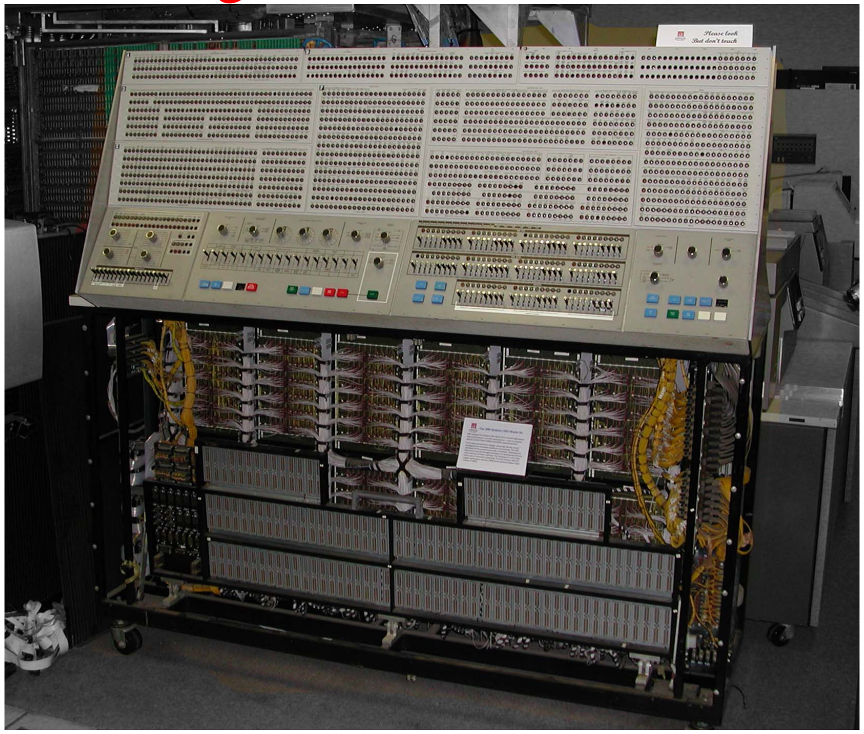
- It is no longer a matter of classifying individual instruction
  - Independent of the previous instructions

- Rather, it is a question of determining each instruction’s relationship with all previous, incomplete instructions

- The final objective is in HARDWARE
  - To preserve the precedences while allowing the execution overlap independent operations
  - for the first time to dynamically exploit concurrency in programs with none effort by the programmer or by the compiler

Motivation and main objective
IBM System/360 Model 91

- Designed to handle data processing for scientific applications
  - such as space exploration, theoretical astronomy, subatomic physics and global weather forecasting



- Instruction time: $60x10^{-9}$ s (ns)
- Time to fetch/store 8 Bytes: 780 *ns*
- Main memory: 2,097,152 Byte

Tomasulo hardware algorithm

Tomasulo hardware algorithm: IBM360/91 datapath

R1 op R2 → R1

R1	op	R2	R1
Register		Register	Register
		or	
		buffer	
source		source	sink

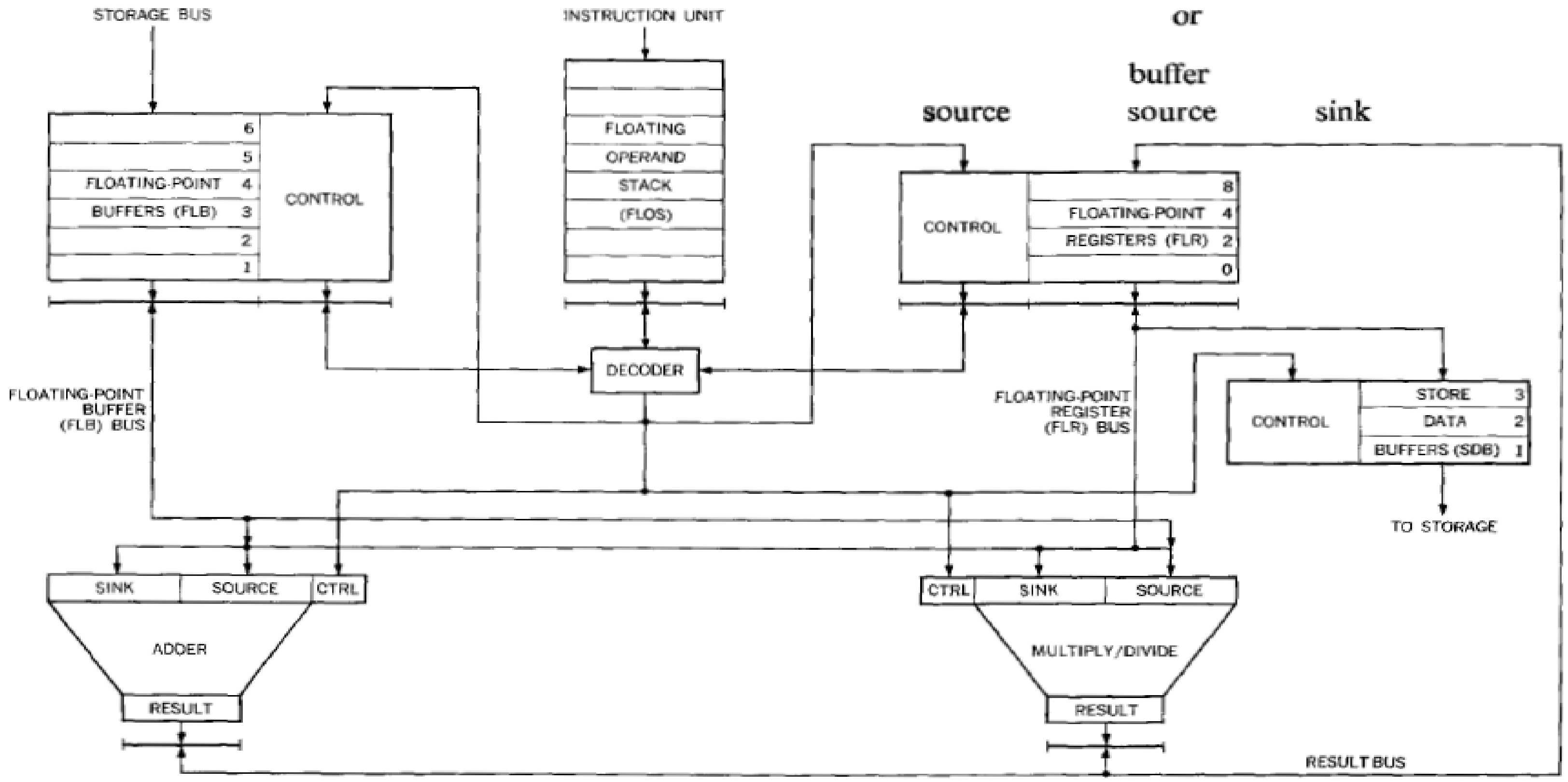
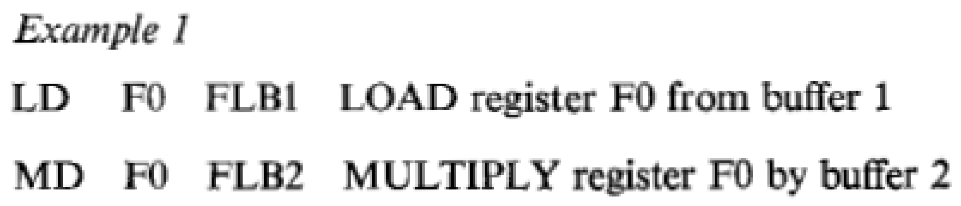


Figure 1 Data registers and transfer paths without CDB.

Tomasulo hardware algorithm: Data (True) dependencies

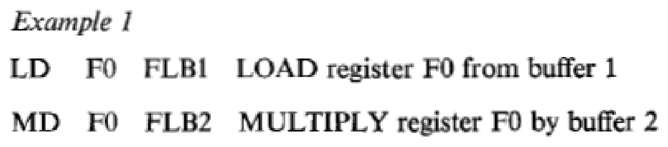


Example 1

LD	F0	FLB1	LOAD register F0 from buffer 1
MD	F0	FLB2	MULTIPLY register F0 by buffer 2

- FO and FLB2 cannot be sent to the multiplier
  - FLB1 has not yet been set into FO.
- This sequence illustrates the cardinal precedence principle:
  - No floating-point register may participate in an operation if it is the sink of another incomplete instruction
    - A register cannot be used until its contents reflects the result of the most recent operation that uses that register as a sink

Tomasulo hardware algorithm


Example 1

LD	F0	FLB1	LOAD register F0 from buffer 1
MD	F0	FLB2	MULTIPLY register F0 by buffer 2

- Before Tomasulo algorithm
  - Designs didn’t incorporate mechanisms to deal with this dependence
- Such hardware mechanism require the following functions
  - To identify dependencies and cause the correct sequencing of the dependent instructions
  - To distinguish between the given sequence from others

LD FO, FLB1

MD F2, FLB2

where it must allow the independent *MD* to proceed regardless of the disposition of the *LD*

Tomasulo hardware algorithm: simplest approach doesn’t work

- One possible hardware approach to identify true depend.
  - *busy* bit associated with each register
    - set when instruction designate sink register and reset when result is written
  - No instruction can be issued if the busy bit of its sink is on
  - If busy bit of source register is on, control bits are set
  - When result is written into the register, these control bits cause the register to be sent via the FLR bus to the unit waiting it as a source
- However in practice this simplest hardware scheme does not allow to exploit concurrency by two main reasons
  - Execution units are occupied waiting for the operands
  - It does not solve not-true name dependencies, such as in the code

    LD FO, Ei; MD FO, Di; AD FO, Ci; LD F0,Ei-1

Tomasulo hardware algorithm: RS + CDB/Tag

- Reservation Stations (RS)
  - Several set of registers (control, sink, source) associated to each execution unit - > Each set is called a **RS**
  - Instruction issuing depends on the availability of reservation station
  - In the Model 91 there are three add and two multiply/divide RS

- Common Data Bus (CDB)
  - The CDB is fed by all units that can modify a register and feeds all units which can have a register as an operand (RS and FLR)
  - A TAG is used to identify each contributor to the CDB,
    - Since there are eleven contributors in the 360/91 a 4-bit tag is used

Tomasulo hardware algorithm: Microarchitecture

technology from seed

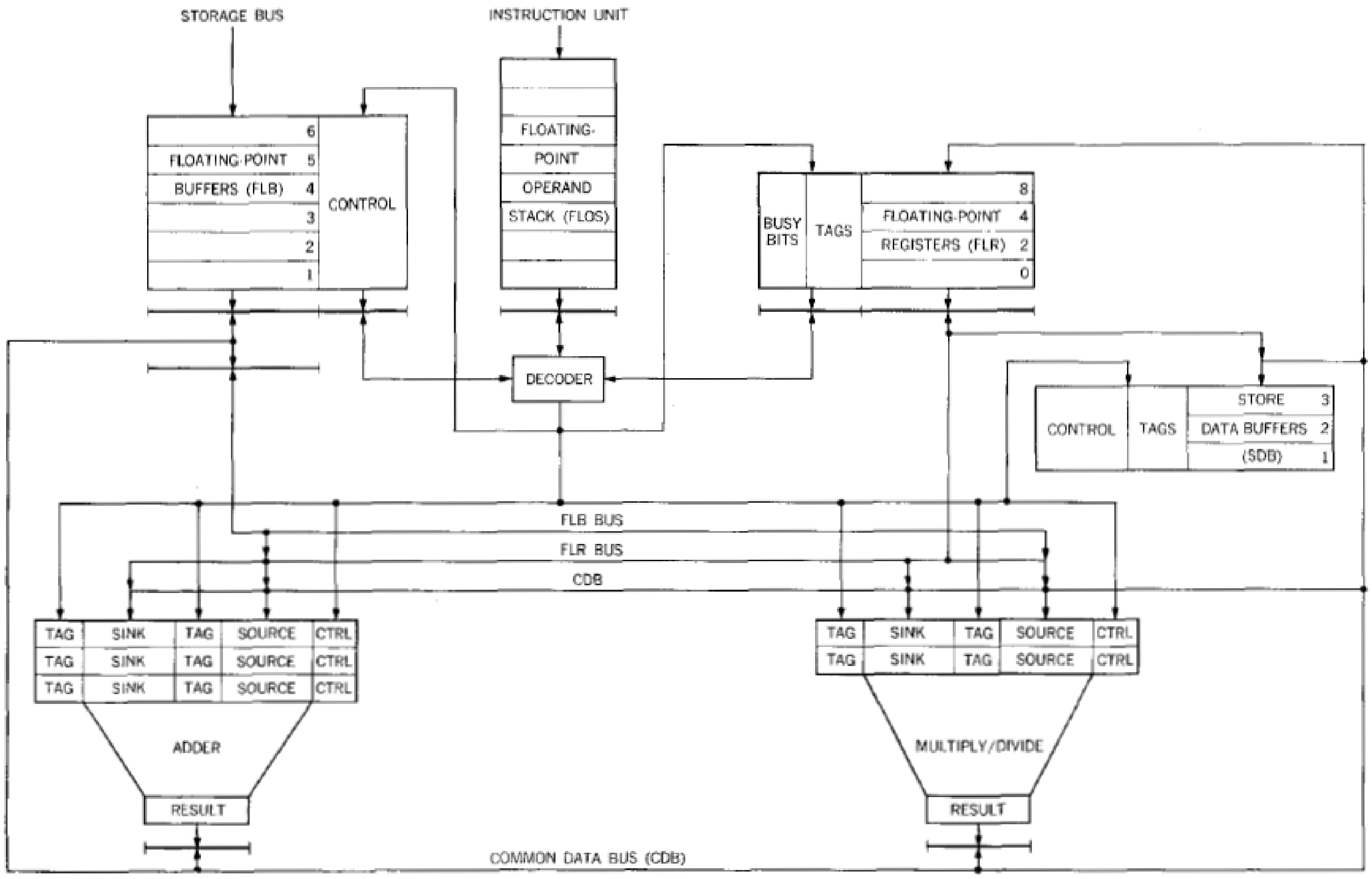


Figure 4 Data registers and transfer paths, including CDB and reservation stations.

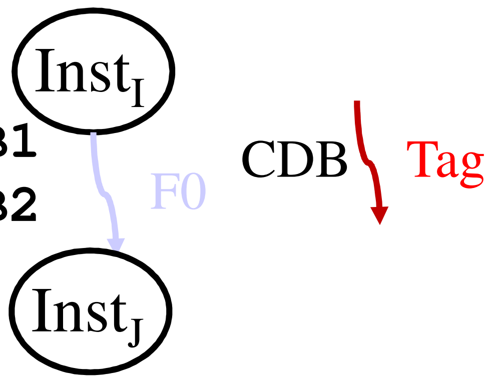
Tomasulo hardware algorithm

1. Instruction issue to a free RS for the current operation
   - Copy the source operands whose value is available to this RS
   - If not available, copy the tag of the source register to the RS
   - Set the tag of the sink register with the identifier of the used RS
2. Instruction execution
   - When all source operands are available in a RS execute operation
   - If not, each RS snoop the CDB to catch missing source operands
3. Write Results
   - When an executed instruction produces result, it is placed in the CDB together with the RS tag
     - This is the opportunity for other RSs and FLRs to update themselves

Tomasulo hardware algorithm: Why is it so important?

- RSs solve structural hazards
  - providing virtual units with a single physical device
- The CDB based Tag scheme solve true dependencies

```
Inst_I: LD F0   FLB1
Inst_J: MD F0   FLB2
```



Tomasulo hardware algorithm: Why is it so important?

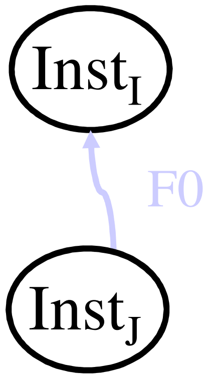
- Better than that it solves all non-true dependencies
  - Name dependencies: anti-depencies or output depedencies
- WHY?
  - Because with RS Tomasulo automaticaly performs renaming

Anti-dependency

$\Downarrow$

$\mathtt{Inst}_I$: DADD R2,R1,R0

$\mathtt{Inst}_J$: DSUB R1,R3,R4



Output-dependency

$\Downarrow$

$\mathtt{Inst}_I$: DADD R2,R1,R0

$\mathtt{Inst}_J$: DSUB R2,R3,R4

Impact in current processors

technology
from seed

inesc id lisboa

1. Motivation and main objective
2. The Tomasulo hardware algorithm
3. Impact in current processors and systems
4. Robert Tomasulo life and OoO history

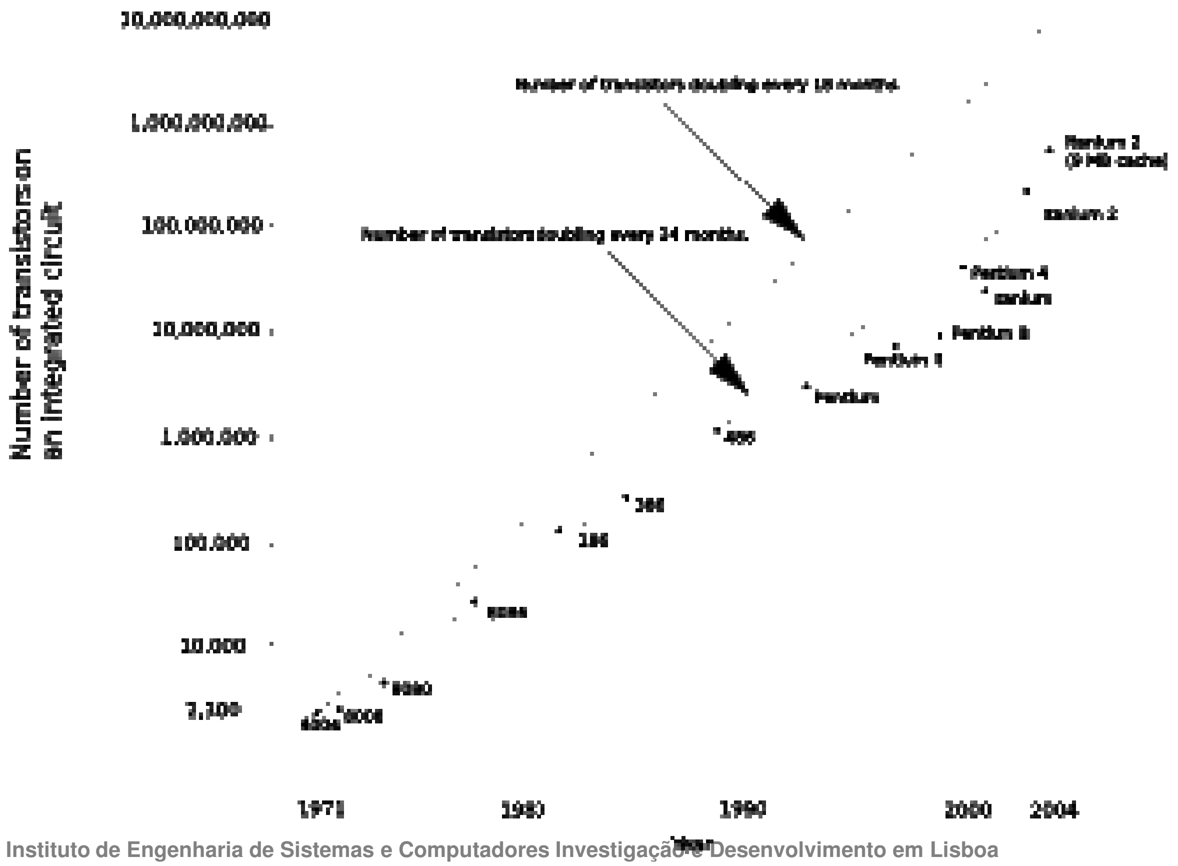
Impact in current processors

- It naturally supports Out-of-Order (OoO) execution
  - With only slight changes it supports Speculative Execution
    - Additional commit step and ReOrder Buffer (ROB)
- OoO speculative execution allows processors to be really parallel, executing several instructions per single cycle:
    - Instruction Level Parallelism (ILP)

- All actual superscalar processors are based on the Tomasulo Hardware algorithm
  - General Purpose Processors: PowerPC (IBM), Intel, AMD
  - Embedded Processors: ARM,MIPS

Impact in current processors

- It has been fundamental to follow the Moore's law
  - Number of transistors on an integrated circuit double every two years



Robert Tomasulo life and OoO history

technology
from seed

1. Motivation and main objective
2. The Tomasulo hardware algorithm
3. Impact in current processors and systems
4. Robert Tomasulo life and OoO history

Robert Tomasulo life and OoO history

- **Eckert-Mauchly Award 1997, *citation***
  - For the ingenious Tomasulo's algorithm, which enabled out-of-order execution processors to be implemented

- Why did it take so much time to make use of Tamsulo's hardware algorithm?
  - Mainly because at that time it did not significantly improve the performance
    - Memory is slow, the number of execution units is not too much, and no ILP was exploited…

Robert Tomasulo life and OoO history

technology
from seed

- Last Seminar January 2008 at University of Michigan:
  - Out-of-Order processing-History of the IBM System/360 Model 91

    http://inst-tech.engin.umich.edu/media/index.php?sk=cse-dls-07&id=1683
- Age 73, Robert Tomasulo passed away on April 3. His friends call him "Uncle Bob"
- WHAT is GOING ON?
  - Limit in the exploitation of ILP, starting multi-core era
    - Several processors for processing in parallel
    - More on the software side -> programming models and compilers
- PPL Stanford: http://ppl.stanford.edu/wiki/index.php/Pervasive_Parallelism_Laboratory

See John Hennessy on YouTube:
http://www.youtube.com/watch?v=FOU3zzh5QO4

http://sips.inesc-id.pt

inesc id lisboa