"Intel Inside!,"

Intel vied,
with ample pride
world wide
in guide
"Proven and tried!".

"Intel Inside!"

Budgets sighed;
millions buyed;
RISC sales dried
like ancient bride.

"Intel Inside?"

Can't divide!
Scientists cried,
fit to be tied,
and numbers fried.

"Intel Inside?"

Can't divide!!
Pi's pied
then FDIV died
and accuracy denied.

"Intel Inside?"

Can't divide?
Executives hide
from "outside"
during Intel bide
on warranty decide.

"Intel Inside?"

Can't divide?!!
See "Thalidomide".
(Taken for a ride.)

Intel lied.

CSE548 Project
Proof of Optimality of Tomasulo's Algorithm

Amol Prakash

Sumit Sanghai

Presentation Layout

- Introduction
- Problem Description
- Related Work
  - Correctness
- Project Plan
- Early Results
  - 5 assumptions, performance loss
- Conclusion

Introduction

- Tomasulo’s Algorithm : 1967
  - Classic Scheduler algorithm
  - Out-of-order execution
  - Virtual Register Renaming, Common Data Bus
- Current situation
  - Tomasulo’s algorithm still there
  - Are the architectures the same ?

What changes have happened?

- Cycle Speed
  - Way much faster processors !!
  - So, what ?
- Memory
  - Faster access, caches
- Hardware : cheaper, faster
- Communication
  - Better, but seems to be turning into a bottleneck
  - Global Bus ??

Is Tomasulo the best ?

- We need to answer the following :
  - What can we assume so as to make it the best ?
  - “Now” if the assumptions fail, does it still remain the best ?
  - If no, then :
    - How much worse are we doing ?
    - What better can we do ?
- We TRY and answer few of the above.

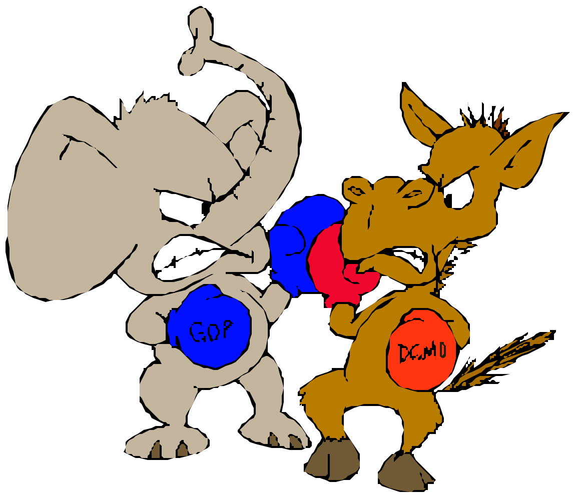
What is best ?

- Optimal.
- For a set of instructions, the algorithm requires minimum number of cycles to execute
- Assume best possible “out-of-order issue”

Problem Description

- Under a necessary and sufficient set of assumptions about the hardware that we are working with, prove Tomasulo optimal.
- Give worst case bounds on performance loss if assumptions do not hold.
- Analyse performance over future architectures.

Related Work



Related Work

- Optimality vs Correctness
  - Recent work focussed on correctness
- Pipelined Processor Verification
  - Increasingly complex designs
  - Need for formal verification
- Formal Verification task
    - Does this circuit implement the specified instruction set ?

Formal Verification

- To verify large, complex designs
  - Automation
  - Decomposition
- Problem definition
  - Need a verification methodology that
    - Is amenable to decomposition
    - Uses decision procedures
- Desirable Properties
  - Independent of configuration and operations
  - Should handle out-of-order executions, interrrupts etc

Approach

- Refinement
  - Mapping between a **abstract system** (specification) and **concrete system** (algorithm)
  - Prove this mapping
- Manual work involved in finding mapping
- Subsequent approaches minimize manual work
  - Compositional Model Checking
  - Incremental Flushing
  - Completion Functions Approach
- Long term Verification challenge
  - Widening gap between abstract specification and algorithm

Project Plan

- Tomasulo Algorithm non-optimal without certain assumptions
- Find these assumptions
  - Infinite reservation stations, functional units etc
- Performance Analysis when assumption fails
  - Find worst case
  - Quantify the degradation

Project Plan [contd…]

- Prove optimality under the given assumption set
  - Iterative process ?
- Analysis of Tomasulo Algorithm over future architectures
  - Wire delay

We hope you are having fun

Early Results

Assumption 1 : Infinite Functional Units

- Tomasulo's approach :
  - Instructions not dispatched till dependency resolved (both operands ready).
  - Functional Unit available.
- Intuition :
  - In the above two requirements, second could be the cause of instruction stalling in reservation station.
  - More instructions waiting to get dispatched than the number of functional units available.

An example

DIV R1, R2, R3

$$\left.\begin{array}{l} \textit{DIV R4, R1, R5} \\ \textit{DIV R6, R1, R7} \\ \textit{DIV R8, R1, R9} \\ . \\ . \\ . \\ \textit{DIV Ri, R1, Rj} \end{array}\right\} m+1$$

Assume m DIV functional units

DIV takes >m cycles

“Worst” Case Analysis

- Let there be n independent instructions all of same type each taking k cycles.
- If infinite functional units, #cycles = n+k
- If ‘m’ functional units, #cycles = [n/m]*k +m
  - Assuming k > m
  - Each m-instruction block gets executed in k cycles
  - #blocks : [n/m]
  - Scaleup of [k/m]

Assumption 2 : Infinite Reservation Stations

- Tomasulo’s approach :
  - Instructions wait in reservation station if operands are not available (dependency, memory)
  - Issued if there is space in reservation station.
- Intuition :
  - Dependent Instructions hog the Reservation Station.
  - More instructions waiting to get issued than the size of reservation stations.

An example

DIV R1, R2, R3

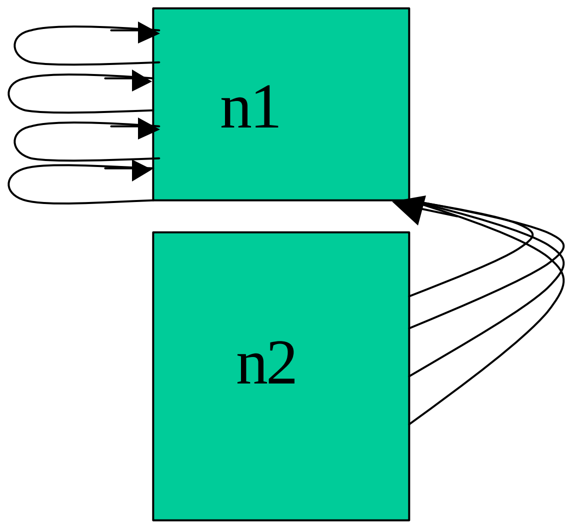
$$\left.\begin{array}{l} \textit{DIV R4, R1, R5} \\ \textit{DIV R6, R1, R7} \\ \textit{DIV R8, R1, R9} \\ . \\ . \\ . \\ \textit{DIV Ri, R1, Rj} \end{array}\right\} \text{m+1}$$

Assume *m* Reservation Stations.

DIV takes > *m* cycles

Assume >*m* functional units.

“Worst” Case Analysis



Each instruction takes *k* cycles

n2 = n1*(k-1)

If infinite RS, #cycles = (n1* k) + k

If *m* RS, #cycles = (n1*k) + k + (n2-m)

Substituting for n2, scaleup of 2

Assumption 3 : Global Communication

- Tomasulo’s approach :
  - Common Data Bus
  - In case of contention, priority resolution
    - Units with more delay have higher priority.
- Intuition :
  - Multiple instructions can finish execution at the same time.
  - But only one result can be put on the CDB.

An example

DIV R1, R2, R3

$$\left.\begin{array}{l}\textit{DIV R4, R1, R5}\\ \textit{DIV R6, R1, R7}\\ \textit{DIV R8, R1, R9}\\ .\\ .\\ .\\ \textit{DIV Ri, R1, Rj}\end{array}\right\} k$$

ADD Rm, Ri, Rk

DIV takes k cycles.

# reservation stations > k

#functional units > k

All k instructions finish at the same time

ADD gets dispatched k cycles after last DIV instruction finished execution.

(assume worst case)

Scaleup : 1.5

Conclusion

- Given some necessary assumptions
  - Infinite functional units, reservation stations, Instant Global communication
- Gut Feeling
  - Assumption set is also sufficient
  - nearly ready with a proof of optimality
- Wish us luck !!