Comm. Driven HW synthesis using Metropolis

A new processor design methodology

Trevor Meyerowitz

Mentors: **Michael Kishinevsky,**
Timothy Kam, Luciano Lavagno

Advisor: **Alberto Sangiovanni-Vincentelli**

DUSD(S&T)

Outline

- **Goals**
- **Processor Design with Current Methodology**
- **Proposed Methodology Additions**
- **Methodology Example**
- **Conclusions**

Goals

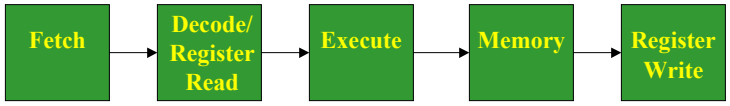
- Integrated processor design methodology using Metropolis framework
  - Refine down to a processor design instead of rewriting model at each stage
  - Find natural representation of communication
- How pure hardware design differs from metro
  - Pure hardware
  - Denser communication
  - Tightly coupled
    - High performance
    - Implicit communication + synchronization
  - (Relatively) fixed set of elements

Metro Methodology:
Step 1: Functional Decomposition

These slides look at how a processor would be designed using the current metro methodology.

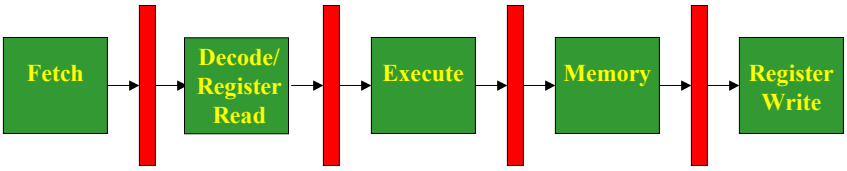
Decomposition is taken from Patterson & Hennessey book.

Metro Methodology:
Step 2: Behavior Adaptation



Connect the functional blocks together, translating the domains if necessary.

Metro Methodology:
Steps 3-5



Pipeline registers inserted.

Where would complicated architectural additions be placed?

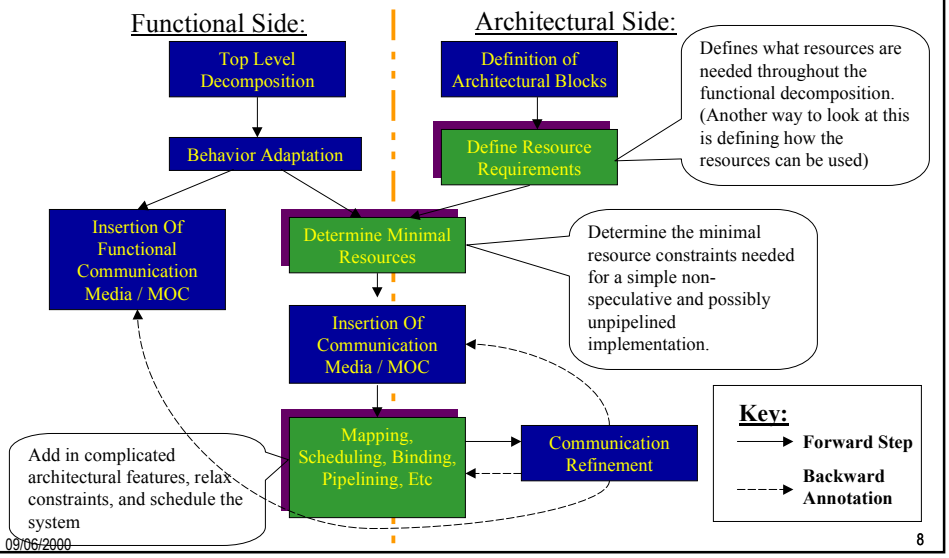
What is the comm. refinement?

What about the mapping?

Issues Raised

- Differences introduced by processor design
  - No real mapping, more like refinement + scheduling
  - Architectural exploration means something different than in typical embedded systems
    - Moving from an architecture to a micro-architecture
  - Communication refinement might actually occur after the “mapping” (decision on architecture)
- Needs added
  - A More Specific Methodology
    - Address complicated architectural features
    - Address the when + where of hierarchy
  - Natural way to express resource usage + properties
  - Representation for scheduling + binding

Proposed Methodology: New Steps



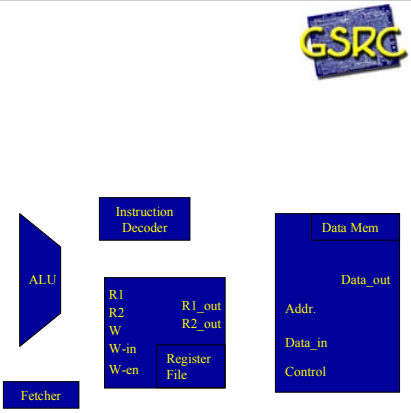
Methodology Example: Simplescalar Processor

- Symbolically scheduled RISC processor taken from Haynal Thesis
  - Attempt it in a more automated + integrated manner.
  - Explore more complicated architectural features
- Illustrated by taking it through the proposed methodology
  - At a high level, most of the details haven't been worked out yet
- Points out problem areas

Metro Methodology: Functional Decomopostion

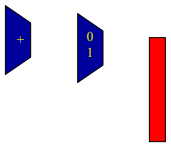
Same functional decomposition as before.

Methodology Example
Architectural Block Definition



- Custom Units:
  - ***All non-memory units execute in < 1 clock cycle***
  - ALU,
  - Instruction Mem (fetcher),
  - Data Memory
  - Register file,
  - Decoder

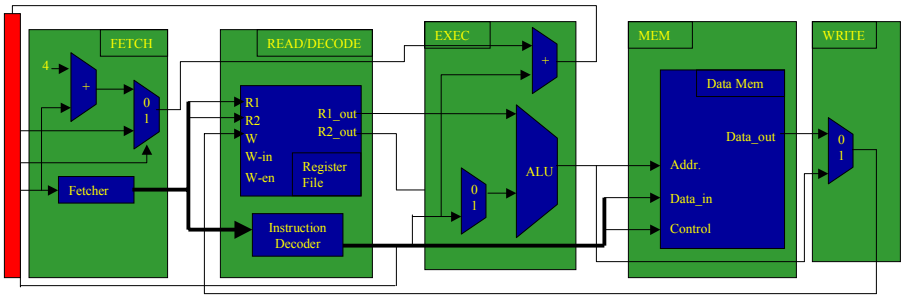
- Standard Blocks:
  - Mux (3)
  - Adder (2)
  - Registers (?)



Methodology Example
Minimal Resources + Simple Mapping

- **Minimal Resources**
  - **One of each unique resource is required**
  - **2 Adders, and 3 Muxes**
  - **# of registers depends on whether or not design is pipelined**
- **Simple Mapping**
  - **Either single cycle or a simple pipeline**
  - **No speculation or branch prediction**
  - **Fetches at most 1 instruction per cycle**
  - **In order execution and commit**
- **Communication Media**
  - **Inputs – simple registers**
  - **Outputs - possibly Registers**

Methodology Example
Simple Mapping



Simple single cycle implementation, an instruction is only fetched after the prior one is completed. Probably represented via constraints.

Methodology Example
MOC wrapper + firing rules

- **MOC wrapper**
  - **Defined by the firing rules, currently fires on clocks.**
- **Firing rules**
  - **Defined by constraints + number of registers**
    - **Single Cycle**
      - Constraints - only 1 instruction in the system at a time
      - Registers – just to hold the PC
    - **Simple Pipelined Design**
      - Constraints - one instruction per stage
      - Registers – set of registers between each stage

Methodology Example
Complicated Mapping and Scheduling

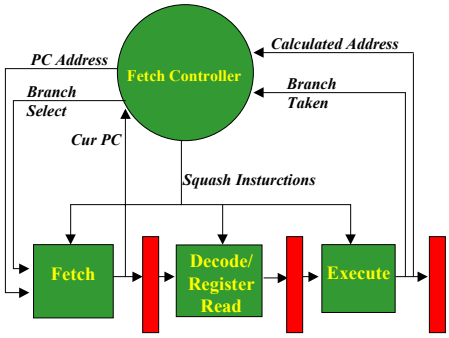
- Based upon relaxation of constraints and duplication of units
  - Combination of manual refinement and symbolic scheduling
  - Select # of resources needed, # instructions in flight at a time, etc.
- Features reviewed
  - Speculation
  - Out of order execution
    - Tomasulo's
    - *Scoreboarding*
  - Superscalar execution
  - *Memory Subsystem organization*

Methodology Example
Mapping Technique

- Blocks should be able to (via control signals + network topology)
  - Stall other blocks and be stalled by others
  - Invalidate their current operations
- Constraints take the form of external schedulers
- Some changes require recoding of behavior (I.e. superscalar implies fetching in blocks)
- Some structural changes
  - Multiple instantiations
  - Addition of arbiters + more control logic
  - Etc.

Methodology Example
Speculation

- Add a control logic scheduler
  - Handles
    - Branch Prediction
    - Recovery from mispredicts
  - Is this a process or a scheduler?

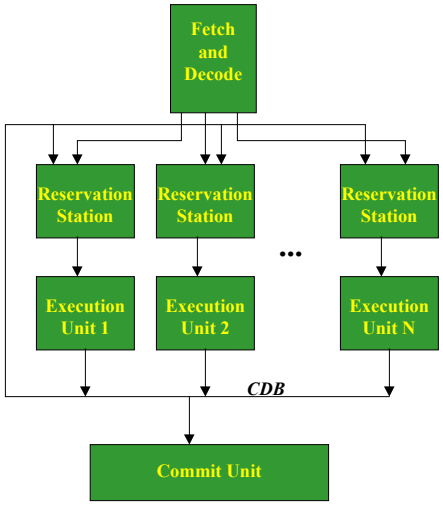


Out of Order Execution

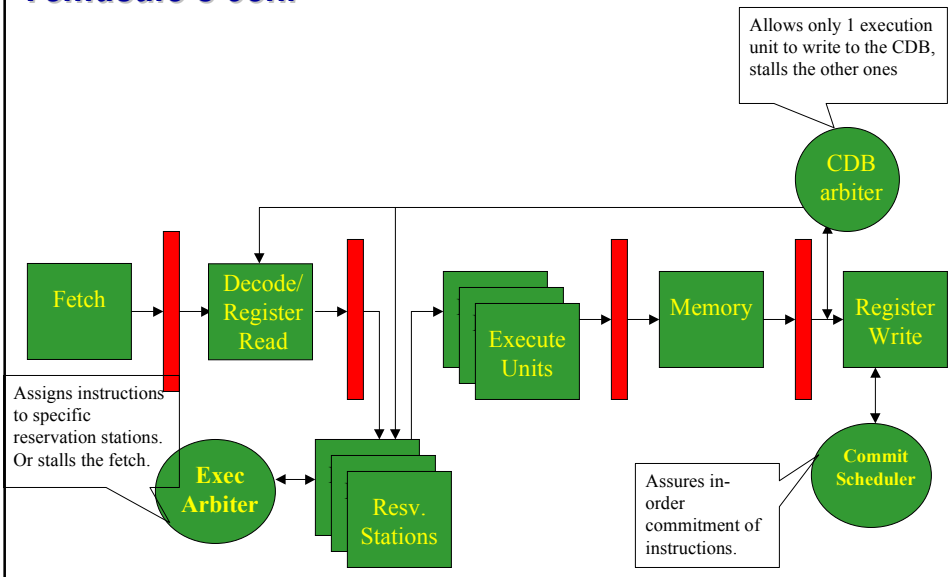
- Requirements
  - Execution times > 1 (*, /, floating point, mem)
  - Multiple execution units
  - Ability to stall and squash instructions
- 2 Techniques
  - *Scoreboarding (Not explained)*
  - Tomasulo's
  - Both introduce new arch. elements.

Out of Order Execution
Tomasulo's

- Distributed Control
- Register Renaming
- Added Arch. Features
  - Reservation Stations
  - Arbiter
  - Common Data Bus (CDB)
  - Commit Unit (optional (for in-order commit))



Tomasulo's con.



Superscalar Execution

- Fetches more than one instruction per clock cycle
  - Typically 2 or 4
- Can execute more than one instruction per clock cycle
  - Similar effect as speculation, and you must be able to squash the instructions that weren't intended to be executed
- Changes required
  - Behavior of fetcher
  - Control similar to that of speculative architectures

Methodology Example
Communication Refinement

- Communication Refinement
  - Delay Calculations
  - Queue Sizing (LID, QSS(?))
  - Pipelining, Binding, etc.
  - Feed updated data back to previous step and back to functional model

Conclusions / What is Missing?

- Conclusions
  - Investigated processor design methodology
    - Using current metro methodology
    - Provided more natural modification of metro methodology
  - Partial Example of new methodology
- What is vague/missing
  - How to specify constraints, resources, and properties
  - How to transform the specification into something that is symbolically schedulable
  - Terminology
  - How to do the complicated mapping
- Comments?