The Need for Architecture Simulation: Lessons from Computer Architecture

Bob Wray

Inspired and informed by collaborations with:
- Jacob Crossman
- Randolph Jones
- Christian Lebiere

Where are we…?

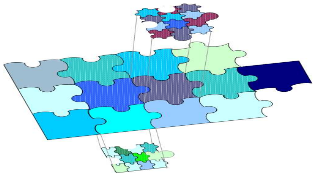
- Cognitive Architecture has not made the revolutionary impact one might expect from the anticipatory rhetoric of the late 1980's
- Continuing concern that there is little/poorly-founded scientific basis for cognitive architecture research as practiced

Larger Agenda: Return to the computer science notion of computer-systems architecture as a basis for cognitive architecture (e.g., Bell & Newell)

Today's talk:
- What can our community learn from the ways CS architectures are researched, developed, and deployed?

Computer Systems Architecture

- Traditional computer systems: successive levels of abstraction
- Each level describes an *architecture*, together with one or more languages running programs at that level
  - Architecture defines the primitive representations and processes
  - The language defines the instruction set for instantiating the representations and processes



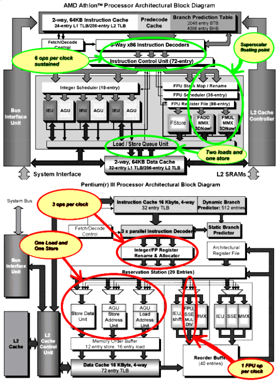
Outside ("user") level
- Users design, compose, and implement solutions (via arch-defined language)
- Abstracts lower-level details

Inside ("implementation") level:
- "User" level is implemented via composition of lower-level components
- *Implementation* of user-level constructs can change without changing the *definition* of those constructs

Every technology follows this pattern...

Computer Architecture

- Common functional components
- Similar organization and data flows
- Differences in implementation (design and fabrication)
- Look (mostly) the same at the user level


AMD Athlon™ Processor Architectural Block Diagram
2-way, 64KB Instruction Cache
24-entry L1 TLB/256-entry L2 TLB
Predecode Cache
Branch Prediction Table
2048 entry BTB
4096 entry BHB
Fetch/Decode Control
3-Way x86 Instruction Decoders
6 ops per clock sustained
Superscalar floating point
Instruction Control Unit (72-entry)
Integer Scheduler (18-entry)
IEU
AGU
FPU Stack Map / Rename
FPU Scheduler (36-entry)
FPU Register File (88-entry)
FStore
FADD MMX 3DNow!
FMUL MMX 3DNow!
Bus Interface Unit
L2 Cache Controller
Load / Store Queue Unit
Two loads and one store
2-way, 64KB Data Cache
32-entry L1 TLB/256-entry L2 TLB
System Interface
L2 SRAMs
Pentium(r) III Processor Architectural Block Diagram
System Bus
3 ops per clock
Instruction Cache 16 Kbyte, 4-way
32 entry TLB
Dynamic Branch Predictor: 512 entries
Fetch/Decode Control
3 x parallel Instruction Decoder
Static Branch Predictor
One Load and One Store
Bus Interface Unit
Integer/FP Register Rename & Allocator
Architectural Register File
Reservation Station (20 Entries)
Store Data Unit
AGU Store Address Unit
AGU Load Address Unit
IEU, shift
FPU, SSE MUL DIV
MMX
IEU
SSE
MMX
L2 Cache
L2 Cache Control Unit
Memory Order Buffer
12 entry store, 16 entry load
Data Cache 16 KByte, 4-way
72 entry TLB
Reorder Buffer (40 entries)
1 FPU op per clock

Images: 7th Generation CPU Comparisons Paul Hsieh

How do computer-system architectures get created today?

General pattern:
- Discrete levels of abstraction
  - Transistor/gate level
  - Microprogramming level
  - Assembly language level
  - "High-level" languages
- Tools and R&D teams focused (almost) solely on one level
- Defined fabrication pipeline (path for putting levels together)
- Application developers provide requirements; use the end product & provide feedback

- Simulation is primary methodology for research and development
  - System-level simulation: VHDL, Verilog, SystemC
  - Power, heat, layout simulations
  - Logic-circuit simulations (SPICE)
- Benchmarks and data analysis of patterns of prior use inform simulations for future iterations
  - Pipeline & cache modeling

How do cognitive architectures get created today?

General pattern:
- 2-3 (indistinct) levels
  - User level ("Soar", "ACT-R")
  - *Algorithm Level (JTMS, RL)*
  - Implementation level (C/Lisp/Java)
- Tools and R&D teams effort spread across all levels
- "Release," not fabrication, model
- Application developers mostly get what they get; many architecture developers are also application developers

- Simulation is almost non-existent for the purposes of architecture-level simulation
- Benchmarks inform testing and verification of releases

Simulation for Cognitive Architectures

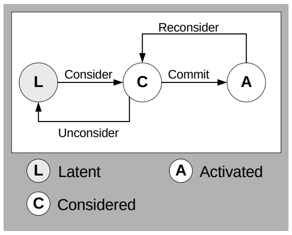
- Recommendations:
  - Recognize consensus and common design patterns
  - Formalize & encapsulate recurring functional elements
- Foundation for cognitive-architecture simulation
  - Enable rapid, empirical design space explorations
  - Facilitate composition of novel architectures
  - Make applying lessons and design principles of computer architecture more feasible
    - Don’t optimize early / Make the common case fast
  - Fabricate, not release

Consensus and Commonality

- There is significant commonality across a range of cognitive (and agent) architectures
  - Mechanisms for associative memory & retrieval
  - Unification over relational representations
  - Integrating parallel associations and serial decisions
  - Reason maintenance, etc.
  - Examples:
    - Soar, ACT-R, Epic, APEX, GLEAN, CAPS, SESAME, …
    - JACK, JAM, RETSINA, SPARK, …
- Convergent evolution?
  - Emergence of similar solutions in different design spaces

Formalizing Common Themes

- Generalized model of memory for cognitive architectures (CCRU)



(Crossman, Wray, Jones, Lebiere, 2004)

- Unique property:
  - Three-state vs. two-state memories
  - Quite common in cog archs
- For each data structure / representational element supported at the user level of an architecture:
  - What process allows that element to considered (part of a decision set)
  - What process allows that element to be committed (selected, activated)
  - What process leads to the reconsideration of commitment?
  - What process leads to complete removal/deactivation?

Representational Level for Simulation

- CCRU Model could potentially be extended to a full framework for architecture simulation
  - Components for all representational primitives
  - Define stubs for each CCRU process
  - Simple, configurable control loop (e.g., Wooldridge, 2000)
- Result: “Primitives” for composing and instantiating existing & novel architectures

Examples

- Soar “beliefs” (i-support)
  - Slots: Id, attr, value, timetag
  - Con/Com: Matching
  - Reconsider: JTMS
  - Uncommit: JTMS
- What are the consequences of different approaches to belief reconsideration?
  - ACT-R: Activation threshold
  - Soar: JTMS
  - 4D/RCS: FIFO
  - SPARK: Belief revision

Summary & Conclusions

Emerging architectural consensus suggests that this field is learning something powerful & important

Nuggets

- Cognitive architectures are CS architectures
- CS methods & tools offer insights for:
  - Speeding cognitive architecture R&D
  - Exploiting architectural advances for applications
  - Organizing community around common concepts and standards
- Common themes & solutions recur across cognitive arch. research

Coal

- Are cognitive architectures *in practice* more like s/w architectures than computer architectures?
- Investment in simulation tools will require a lot of commitment
- What’s the compelling technical (or sociological?) demonstration that could trigger sustained investment?

A good topic re future workshop formats? ☺