Frontend SoC design: The neglected frontier

Arvind
Computer Science & Artificial Intelligence Lab.
Massachusetts Institute of Technology

NSF Workshop: Electronic Design Automation — Past, Present, and Future
Arlington, VA
July 8-9, 2009
The future would be dominated by the concerns of

- Cheap & powerful handheld devices
- Powerful infrastructure needed to support services on these devices.

July 8, 2009

http://csg.csail.mit.edu/arvind
Current smart phone Architecture

Two chips, each with an ARM general-purpose processor (GPP) and a DSP (TI OMAP 2420) +
~80 complex specialized blocks

Complex, High Performance but must not dissipate more than 3 watts
Real power saving implies specialized hardware

- H.264 video decoder implementations in software vs. hardware
  - the power/energy savings could be 100 to 1000 fold

**but our mind set is that hardware design is:**
- Difficult, risky
  - Increases time-to-market
- Inflexible, brittle, error prone
  - Difficult to deal with changing standards, ...

New design flows and tools can change this mind set
What we need: # 1

Design methodologies and tools to facilitate extreme IP reuse
What we need: # 2

Design methodologies and tools to facilitate architectural exploration
What we need: # 3

Design methodologies and tools with *abstraction and composition rules with predictable outcome*
Verification?

The degree of correctness required depends upon the application

- Different applications require vastly different formal and informal techniques

Formal tools must be tied directly to high-level design languages

Formal techniques should be presented as debugging aids during the design process

- A designer is unlikely to do anything for the sake of helping the post design verification
- Specifications of complex systems evolve continuously
Desired level of verification depends upon the application

- IP Lookup in a router
  - Functional correctness is easy, proving that packets come out in order is difficult
- 802.11a Transceiver
  - Few lost packets do not matter but showing that all the correctable packets are being received is tough
- H.264 Video Codec
  - Lossy encoding! Theoretical criteria for good encoding are of no use in verification
- OOO Processors
  - One would want total correctness but usually correct results on old programs gets one most of the way
- Cache Coherence Protocols
  - Total correctness essential – even the designer does not trust testing
A designer wants

To trust commonly used components

- Arithmetic; common datastructures like queues, lists, hash tables, ...; common routines like sorting, maps, folds, ...;

To trust commonly used tools and tool flow

- Compilers, simulators, ...
- “no silent failures”
Cost Matters

The goal is to design systems that meet cost, performance, power, correctness, compatibility, robustness, etc.
- Design time \( \approx \$\$

Designers will use any technique that increases their confidence in the system provided it:
- gives useful feedback quickly
- is better than manual debugging
- doesn’t require learning a “foreign language”
- is not elitist (No PhD requirement)
Some “Do”s and “Don’t”s

- Most successful formal techniques (e.g., types) help the designer, *not* just the verifier.

- Separation of design and verification languages is a non-starter
  - what are you verifying?
  - manual abstraction, changing specs, ...

- Writing specs is a good idea, but it rarely happens
  - error prone
  - time consuming
  - incomplete
  - incomprehensible
  - changing requirements
What about technology related issues

- Increasing uncertainty
- Increasing variability
- Increasing soft-errors

*all these issues have to be dealt with by essentially masking them at the lowest possible level of design*
Front-end design needs a big boost

- High-level notation
  - capable of expressing parallelism and nondeterminism
  - amenable to synthesis of actual implementation
  - Must include proven language concepts: e.g., types, abstractions, higher-order functions

- Powerful tools for
  - synthesis
  - proving properties of such designs
  - estimating area, speed, power, ...

- Rich and ever increasing set of IP blocks

Thanks!