
Bridging the High-level and Implementation Divide: Mission Impossible?

**Victor Konrad
April 2002**

intel

4/25/2002

Agenda

- ✦ **Background**
- ✦ **Philosophy**
- ✦ **Experiments in speedup of HLM**
- ✦ **Conclusions**
- ✦ **Disclaimer: view of HLM from the (narrow) vantage point of large general-purpose microprocessors**

intel

4/25/2002

A somewhat pessimistic report from two projects

❖ Yosemite (R.I.P.) HLM

- Project cancelled

❖ IA 64 project X (HLM R.I.P.)

- Project still alive, but no HLM

❖ Information is 1.5 – 2 yrs old

- But still valid

intel

4/25/2002

The Yosemite project HLM

❖ Yosemite was to be next-generation Itanium

- Was developed alongside Itanium for several years
- Extended honeymoon period

❖ Yosemite HLM

- Structural iHDL
- Written by architects with a microarchitectural bend
- Many thousands of lines of code
- Clock accurate
- Intention: match RTL on major signals

❖ Objectives (and wishful thinking)

- Architects' sandbox
- Mixed-level simulation with RTL (plug-and-play)
- Checkers for RTL validation developed early
- FV...

intel

4/25/2002

A word about iHDL

⊗ Developed ~15 years ago, in continuous use since

- Slowly being phased out in favor of Verilog

⊗ Explicitly synchronous, logic-design oriented

- "glorified netlist"
- Some high-level built-in constructs (*, +)
- very rich bit-vector manipulation features
 - » $a[5:2] \& b[16:11] \& '1::(b-a) := (d[b:a] \& '0::9) + SCVN(31);$
- Missing basic high-level capabilities (e.g. user-defined behavioral procedures)
 - » De-prioritized due to lack of interest from users
- Underlying timing paradigm: FSM (no explicit concurrency, threads etc.)

⊗ Very slow simulation speed

- Itanium ran at ~1Hz
- Yosemite HLM very slow for an "architect's sandbox"
 - » Harder to use word-level parallelism, NetBatch & other techniques from validation



4/25/2002

Approaches to speedup

⊗ Use C/C++ - based modeling (SystemC, CynApps)

- Was making its first strides when the project was cancelled
- Showed good speedup, though probably insufficient

⊗ Library of high-level models

- Encompass ubiquitous structures
- Use simple simulation paradigm ("Execute this code in 3 cycles")
- Tuned for simulation performance

⊗ Classification of library elements

- Simple logic design structures
- Elaborate logic design structures
- Micro-architectural primitives



4/25/2002

Simple logic design primitives (compiler enhancements? Macros?)

❖ **Paradigm: c:= a+b, c:=a*b etc.**

- Language built-ins
- Software executable model bears no resemblance to the final hardware

❖ **A ubiquitous primitive: "find first"**

In: x = (0 1 0 0 1 1 0 1 0 0)

Out: y = (0 0 0 0 0 0 0 1 0 0)

Quasi-C solution:

```
y=0
if (x[0]=='1') y[0]='1;
else if (x[1]=='1') y[1]='1;
else if (x[2]=='1') y[1]='1;
...
....
```

Better: y = (-x) & x

Proof:

$\sim x = (1011001011)$

$(\sim x) + 1 = (1011001100)$

$x \text{ AND } ((\sim x) + 1) = (0100110100) \text{ AND } (1011001100) = (0000000100)$



4/25/2002

Elaborate logic structures: PLA

❖ **PLA:**

$$y_1 = f_1(x_1, x_2, \dots, x_n)$$

$$y_2 = f_2(x_1, x_2, \dots, x_n)$$

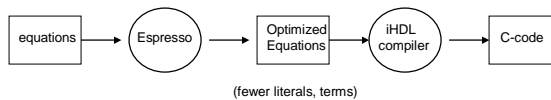
...

$$y_m = f_m(x_1, x_2, \dots, x_n), \text{ where } f_k \text{ are sums of products}$$

Simple-minded approach:



A better approach:

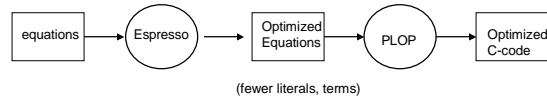


Problem: Espresso is a hardware optimizer, not tuned to improve performance of the software model!



4/25/2002

New algorithm: PLOP



- ✦ **At the core of PLOP is a procedure similar to the recursive splitting which occurs in *TAUTOLOGY* of Espresso**
 - Different heuristic of choosing the splitting variable
- ✦ **Allows flexible memory/speedup tradeoffs**
- ✦ **Heuristic, but a very good one:**
 - ✦ Tested on ~40 PLAs from Itanium and Banias: speedups of 3x-20x achieved
- ✦ **A modification of this algorithm reduces significantly dynamic PLA power: patent pending**

intel

4/25/2002

Microarchitectural primitives

- ✦ **CAMs, FIFOs, LIFOs, pipelines**
 - Modeled in (more or less) straightforward ways
 - ✦ **TLB**
 - Structure found in all microprocessor designs
 - Used for mapping of virtual address to physical address
 - Given a virtual address, determine if the data is contained within a page which is currently in memory
 - Hardware scans a list of pages and determines if the given address is contained in any of them.
 - » If so, translate; if not found, page fault
 - The hardware scan is in parallel on all entries of the array, but this algorithm, if done in software, is linear in the number of entries in page table
 - » Very time-consuming in simulation: TLB activated for every memory access
- Appears to be another application of hashing, but it is not:
Determining if an address is within a given page requires a masking operation, and the mask is specific to each page.**

intel

4/25/2002

Better software algorithms for TLB simulation

- ⊛ Found and implemented an algorithm which implements the procedure in logarithmic time
- ⊛ **Discovery: Problem is isomorphic to that of fast subnet-based routing**
 - A very important problem in networking
- ⊛ **See: “Fast Longest Prefix Matching: Algorithm, Analysis, and Applications”, Marcel Waldvogel, Ph.D. Thesis, ETH Zurich, 2000**

intel

4/25/2002

When the project was cancelled...

- ⊛ **...we pretty much convinced ourselves that, speedwise, HLM was doable**
 - Combination of C/C++ programming and faster models
- ⊛ **But the other, more important business/methodology issues remain unresolved:**
 - Can we make the HLM a clock/signal accurate golden model for RTL?
 - Can we make its modules plug-and-play interchangeable with RTL?

And, most importantly,

Is there sufficient return on the investment of building and maintaining this model (and keeping it in synch with the RTL)?

intel

4/25/2002

HLM lessons from project X (McKinley follow-on)

- ⊗ **Project X RTL based on existing McKinley, changes in certain units.**
- ⊗ **No high-level model for McKinley**
- ⊗ **Question: Is there a sufficient ROI in retrofitting an HLM to existing RTL?**
- ⊗ **Answer: no**
 - Thus HLM was canned, but not before I had lots of fun reverse-engineering the McKinley RTL and writing a high-level model for it...

intel

4/25/2002

Summary

- ⊗ **Each successive generation of Intel microprocessors has toyed with high-level modeling, with limited or no success.**
 - Hope springs eternal: even as we speak, new experiments are underway
- ⊗ **Missing is the bridge between HLM and RTL**
 - No progress unless we have an HLM model which
 - » Is orders of magnitude faster than RTL
 - » Is cycle- and major signal- compatible with RTL
 - » Its modules can be plugged seamlessly into an RTL model
- ⊗ **No ROI for proliferations**
- ⊗ **High-level synthesis?**
 - Likely doable for narrower, special-purpose domains of applications (DSP etc.)
 - Not yet there for microprocessors

intel

4/25/2002