

Liquid Architecture - Results of Optimization of the LEON,
Invited talk for RAMP January 2007 Workshop,
University of California at Berkeley, January 12, 2007.

Liquid Architecture – Results of Optimization of the LEON



John W Lockwood
Visiting Associate Professor
Stanford University
jwlockwd@stanford.edu
<http://stanford.edu/~jwlockwd>

Slides from Raw 2004, 2006, by Shobana Padmanbhan, et. al.
In Collaboration with: Ron Cytron, Roger Chamberlain, Jason Fritts
David Schuehler, Phillip Jones, Scott Friedman, Ben Brodie, Huakai Zhang



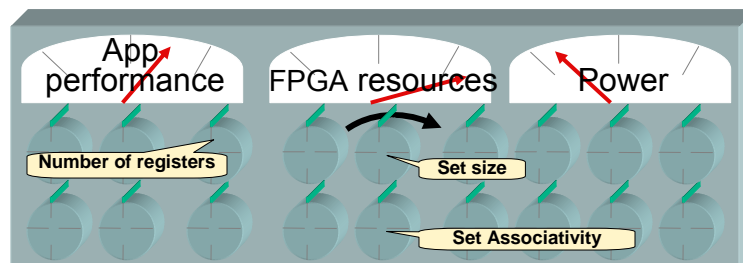
Funded by : NSF 03-13203

Washington University in St. Louis
www.arl.wustl.edu/arl/projects/fpx/projects/liquid_arch/



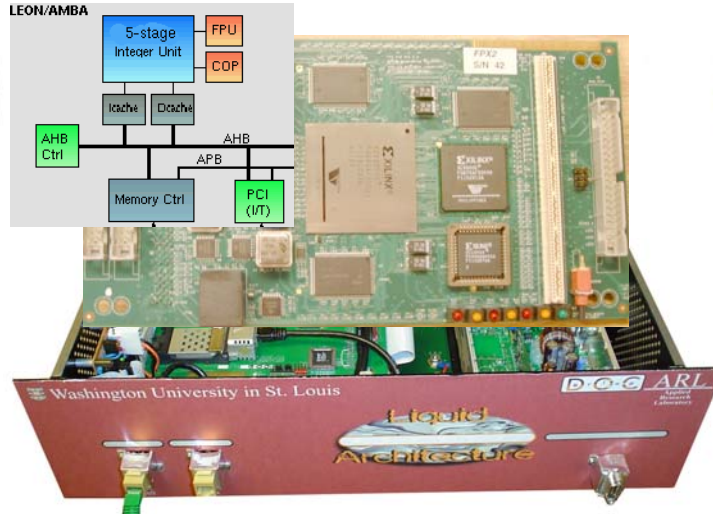
Soft processors

- Parameterized general purpose processors



- Customization is performance-cost tradeoff
- More “knobs” ⇒ more options for customization

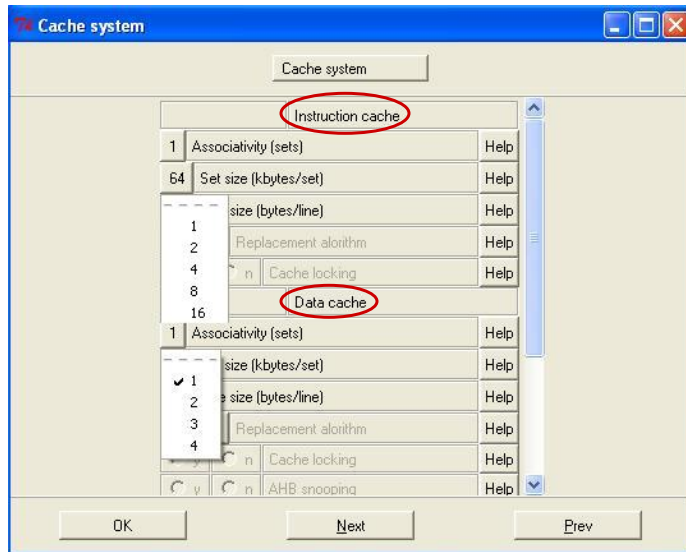
Actual System Configuration



Soft processor customization

- **LEON: 10 reconfigurable subsystems**
 - **Instruction cache**
 - Parameters: sets, set size, line size, replacement policy
 - $4 * 7 * 2 * 3 = 168$ configurations (4 parameters; 16 values)
 - **Data cache**
 - sets, set size, line size, replacement, fast read, fast write, local RAM, local RAM size
 - $168 * 2 * 2 * 2 * 7 = 9,408$ configs (8 params; 29 values)
 - **Integer unit**
 - multiplier, registers, fast jump, fast decode, ICC, load delay, FPU enable, co-processor enable, hardware watchpoints
 - 119,040 configurations (10 parameters; 56 values)
 - **& Floating-point unit, memory controller, peripherals,...**
- **In total**
 - 190 parameter values; $5^*(10^{24})$ configurations!!

Manual Adjustment of LEON Parameters



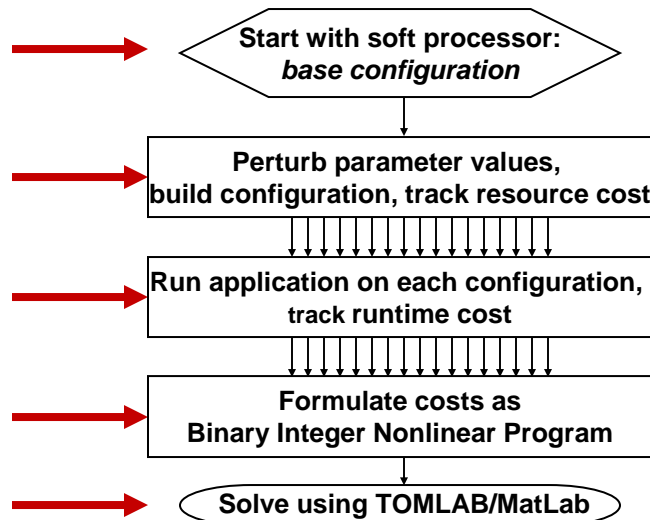
Highlights of optimization technique

- **Optimize and Bound Parameterize**
 - Search space still includes all $5 \cdot (10^{24})$ configurations
 - Build only 100's instead of $5 \cdot (10^{24})$ of configurations
- **Formulate as binary integer nonlinear optimization program**
- **Measure actual cost and performance of resulting application performance and processor area**

Cost measurement

- **Application runtime**
 - Measured in cycles from direct execution
 - Hardware-based profiler provides non-intrusive, cycle-accurate, near-real-time measurement
- **FPGA resources**
 - Measured in terms as number of LUTs and BRAM, from actual *build*
- **Power / Energy : Work in Progress**
 - FPL, FPT, VLSI conference ..

Optimization technique



Processor ICache reconfiguration

Parameter
#sets
Setsize (KB)
Linesz (words)
Replacement

Processor ICache reconfiguration

Parameter	Value	Variable	Runtime	Chip
#sets	1	base	0	0
	2	x_1	r_1	l_1+b_1
	3	x_2	r_2	l_2+b_2
	4	x_3	r_3	l_3+b_3
Setsize (KB)				
Linesz (words)				
Replacement				

$x_i = 0 \text{ or } 1$
 (off or on)

Processor ICache reconfiguration

Parameter	Value	Variable	Runtime	Chip
#sets	1	base	0	0
	2	x_1	r_1	l_1+b_1
	3	x_2	r_2	l_2+b_2
	4	x_3	r_3	l_3+b_3
$x_i = 0 \text{ or } 1$ (off or on)				
$\sum_{i=1}^3 x_i \leq 1$				
Setsize (KB)				
Linesz (words)				
Replacement				

Processor ICache reconfiguration

Parameter	Value	Variable	Runtime	Chip
#sets	1	base	0	0
	2	x_1	r_1	l_1+b_1
	3	x_2	r_2	l_2+b_2
	4	x_3	r_3	l_3+b_3
$x_i = 0 \text{ or } 1$ (off or on)				
$\sum_{i=1}^3 x_i \leq 1$				
Setsize (KB)				
	1	x_4	r_4	l_4+b_4
	2	x_5	r_5	l_5+b_5
	4	base	0	0
	8	x_6	r_6	l_6+b_6
	16	x_7	r_7	l_7+b_7
	32	x_8	r_8	l_8+b_8
$\sum_{i=4}^8 x_i \leq 1$				
Linesz (words)				
	4	x_9	r_9	l_9+b_9
	8	base	0	0
No constraint needed				
Replacement				
	Random	base	0	0
	LRR	x_{10}	r_{10}	$l_{10}+b_{10}$
	LRU	x_{11}	r_{11}	$l_{11}+b_{11}$
$\sum_{i=10}^{11} x_i \leq 1$				

FPGA resource constraints

- **LUTs**
$$\sum_{i=1}^n l_i x_i \leq L$$

- **BRAM**
$$\sum_{i=1}^n b_i x_i \leq B$$

$x_i = 0$ or 1
(off or on)

r_i, l_i, b_i : delta costs from
base configuration

n is number of
configurations

Optimization

- **Optimize application runtime**

$$\text{Minimize } \sum_{i=1}^n r_i x_i$$

- **Optimize resource utilization also**

$$\text{Minimize } \sum_{i=1}^n r_i x_i + (l_i + b_i) x_i$$

$$\text{Minimize } \sum_{i=1}^n w_1 (r_i x_i) + w_2 ((l_i + b_i) x_i)$$

$x_i = 0$ or 1
(off or on)

r_i, l_i, b_i : delta costs from
base configuration

n is number of
configurations

Problem formulation

- **Minimize** $\sum_{i=1}^n w_1(r_i x_i) + w_2((l_i + b_i)x_i)$
 - **Subject to**
 - $\sum_{i=1}^3 x_i \leq 1$ } Parameter validity constraints
 - \dots
 - $\sum_{i=1}^n l_i x_i \leq L$ } FPGA resource constraints
 - \dots
 - $x_i(x_i - 1) = 0$ } Binary variables constraint
- where $i = 1, 2, \dots, 52$

$x_i = 0$ or 1
(off or on)

r_i, l_i, b_i : delta costs from
base configuration

n is number of
configurations

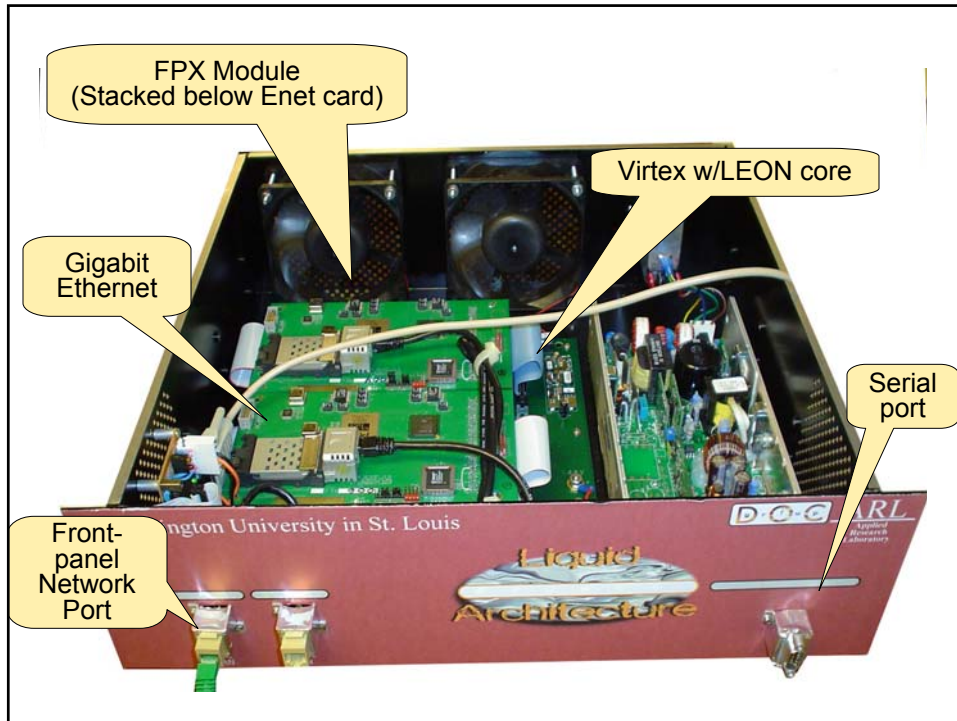
Evaluation

Exhaustive (CommBench DRR): dcache #sets, set size				
#sets	setszKB	runtime secs	LUTs%	BRAM%
1	1	356.504	38	47
1	2	317.071	38	48
1	4	297.979	39	51
1	8	283.656	39	56
1	16	271.597	38	68
1	32	271.609	38	90
2	1	305.391	39	49
2	2	274.519	39	51
2	4	268.325	40	56
2	8	261.915	39	68
2	16	261.609	39	90
3	1	279.431	39	51
3	2	268.894	39	55
3	4	263.349	41	62
3	8	261.609	39	79
4	1	271.765	39	53
4	2	267.285	39	58
4	4	261.656	41	68
4	8	261.609	39	90

Optimize runtime (DRR): dcache #sets, set size $w_2=0$				
#sets	setszKB	runtime secs	LUTs%	BRAM%
2	4	268.325	40	56
3	4	263.349	41	62
4	4	261.656	41	68
1	1	356.504	38	47
1	2	317.071	38	48
1	4	297.979	39	51
1	8	283.656	39	56
1	16	271.597	38	68
1	32	271.609	38	90
2	16	261.609	39	90

Our technique selects the
same configuration

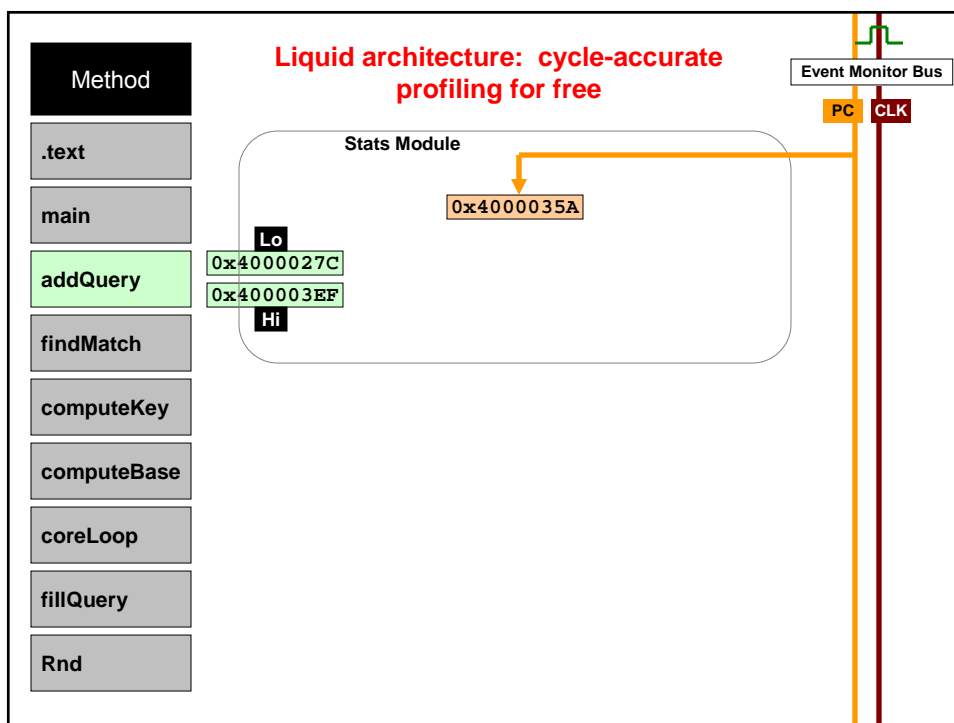
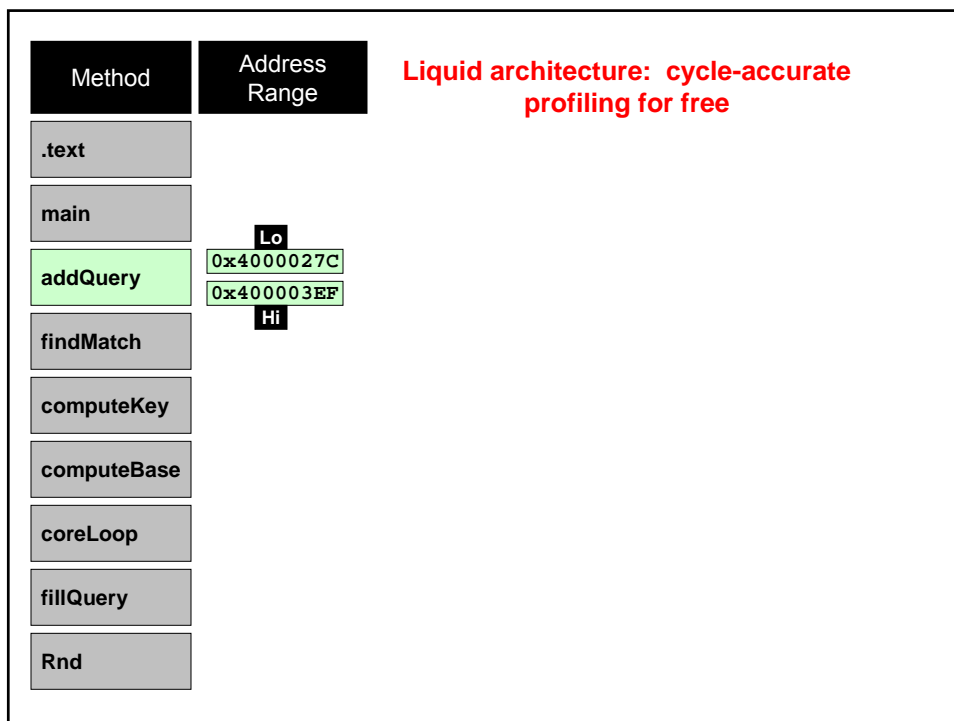
Despite parameter
independence assumption,
near-optimal configuration

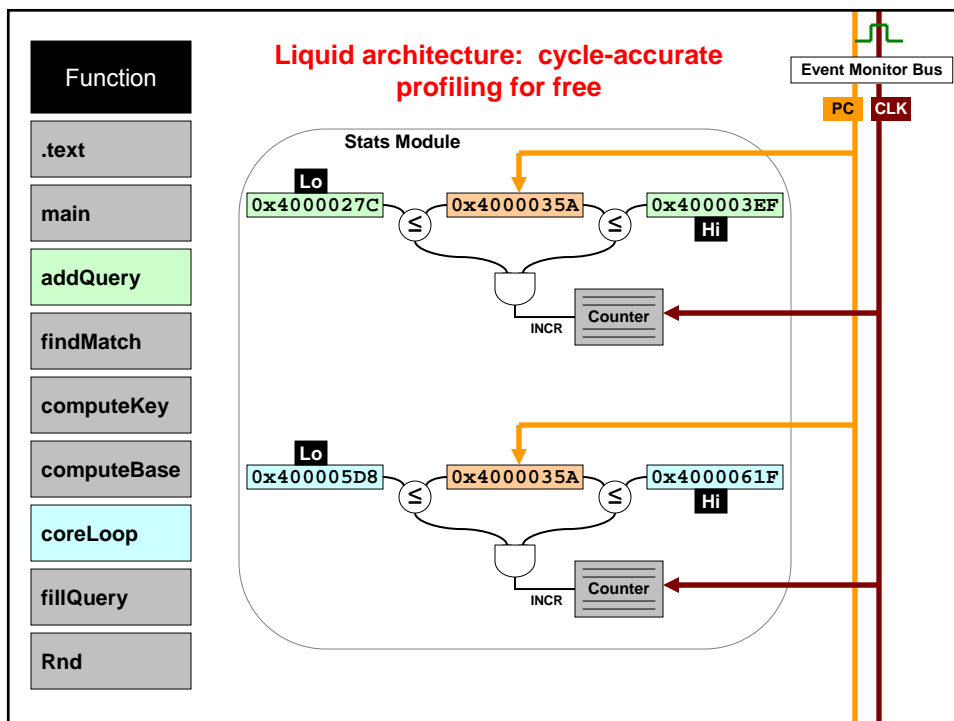
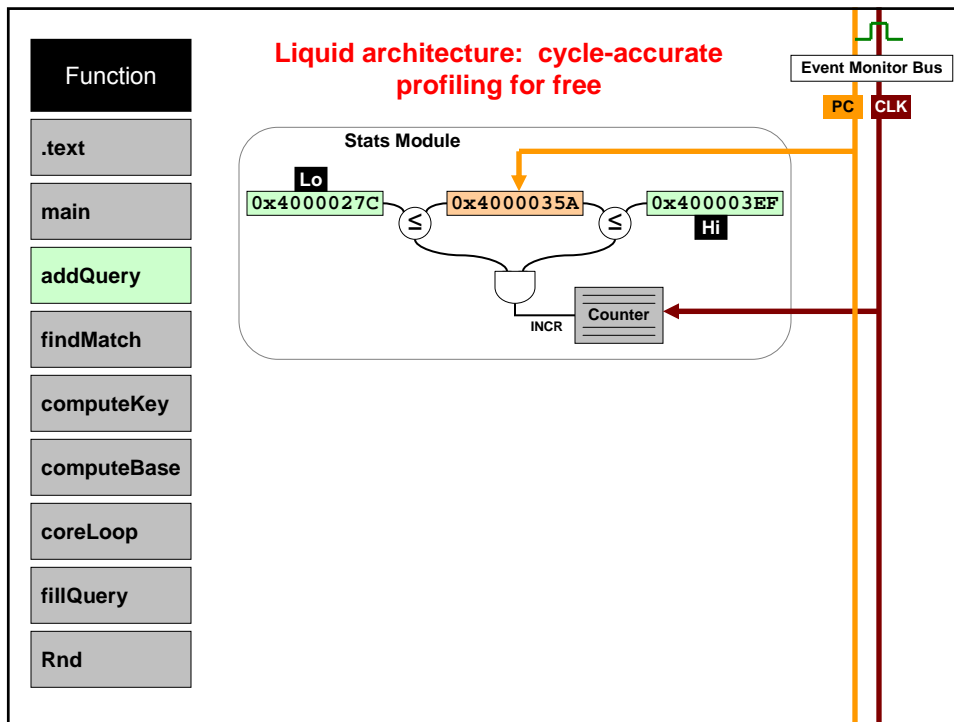


Method	Time / Cycles
.text	<input type="checkbox"/>
main	<input type="checkbox"/>
addQuery	<input checked="" type="checkbox"/>
findMatch	<input type="checkbox"/>
computeKey	<input type="checkbox"/>
computeBase	<input type="checkbox"/>
coreLoop	<input checked="" type="checkbox"/>
fillQuery	<input type="checkbox"/>
Rnd	<input type="checkbox"/>

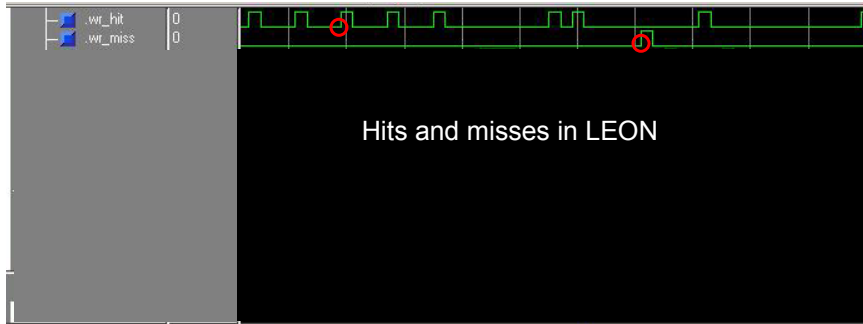
Liquid architecture: cycle-accurate profiling for free

- Choose methods to profile from the user interface

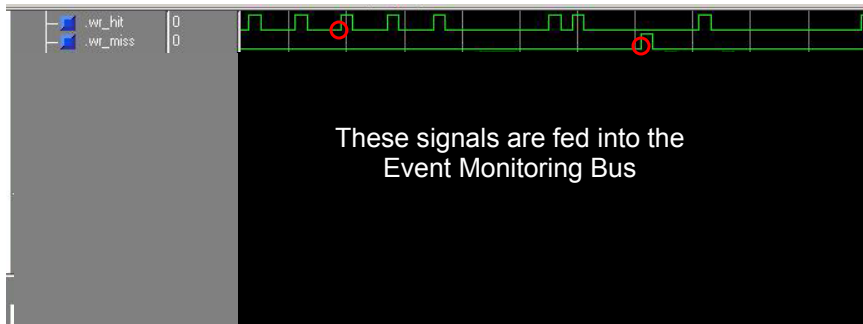




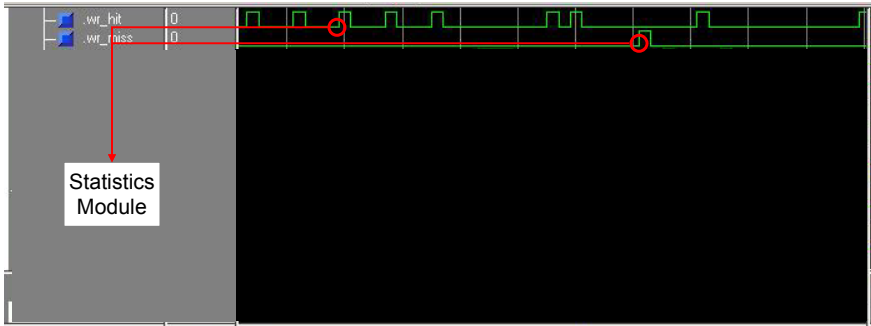
Cache behavior



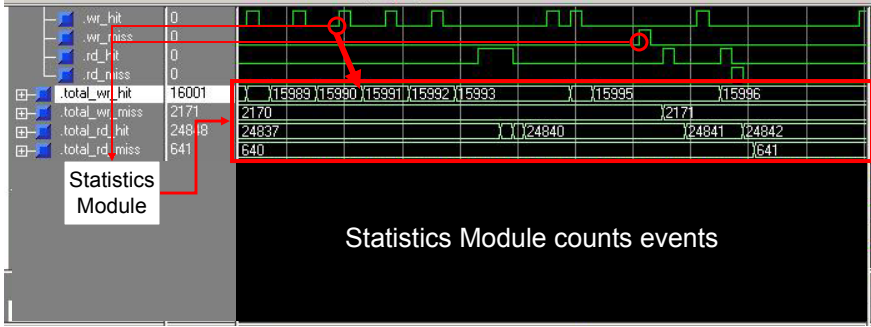
Cache behavior



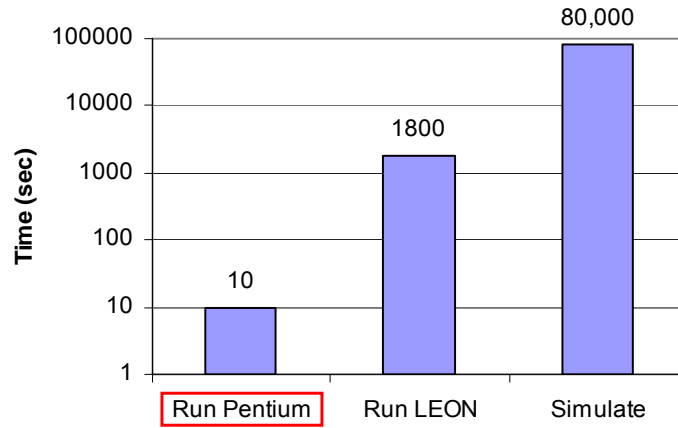
Cache behavior



Cache behavior

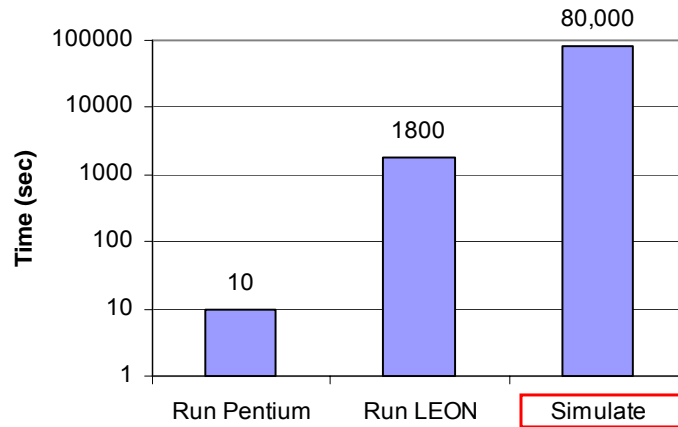


Liquid architecture enables fast, accurate results



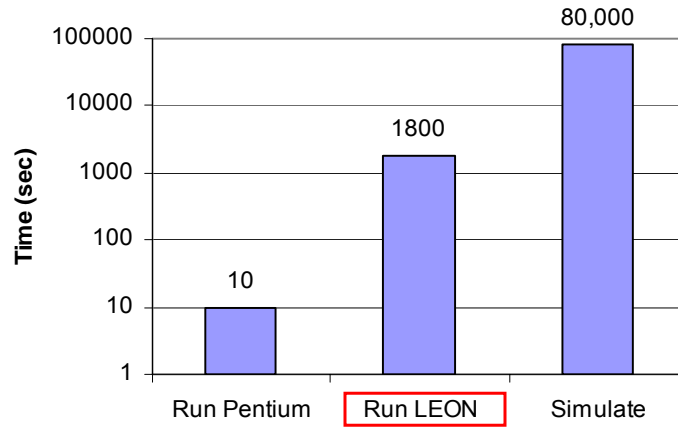
Seconds: fast, but no cache performance data available

Liquid architecture enables fast, accurate results



Days: so slow you wouldn't do this on the whole program

Liquid architecture enables fast, accurate results



½ hour: Practical, reasonably fast, totally accurate

Illustration of all configurations being searched

