Simulation-driven verification

Design Automation Summer School, 2009

Bob Bentley
Director, Pre-silicon Validation
Enterprise Microprocessor Group
Intel Corporation
Hillsboro, Oregon, U.S.A.

Agenda

- Microprocessor design scope
- What is Nehalem?
- RTL High Level Timeline
- RTL coding strategy
- RTL model health indicators
- RTL model build methodology
- Microprocessor Validation & Verification
- Verification environment
- Verification progress indicators
- Verification results

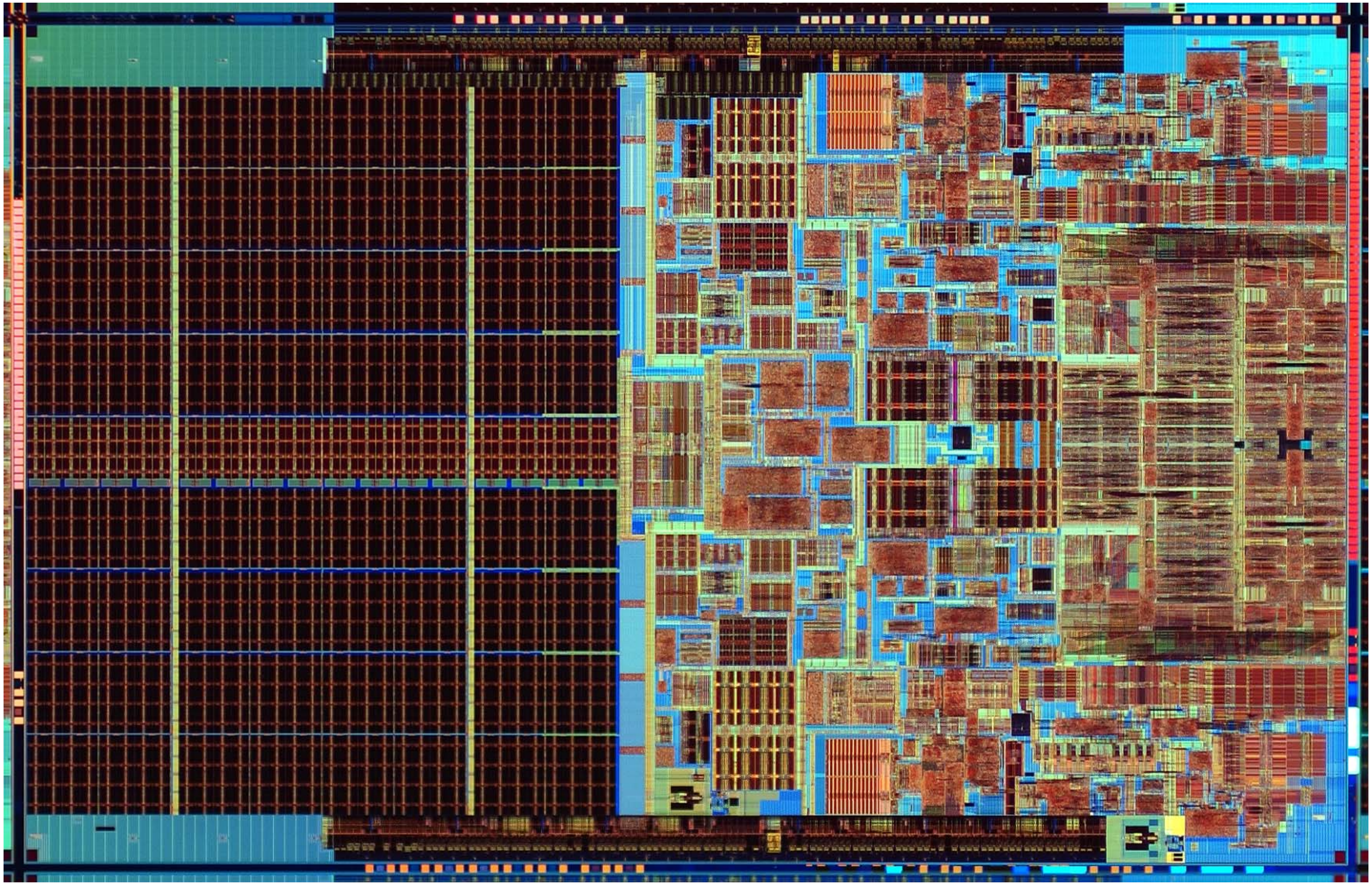
Microprocessor Design Scope

- Typical lead x86 CPU design attributes:
- 500+ person design team:
  - logic and circuit design
  - physical design
  - validation and verification
  - design automation & other support
- 24-30 months from start of RTL to A0 tapeout
- 12-18 months from A0 tapeout to first production

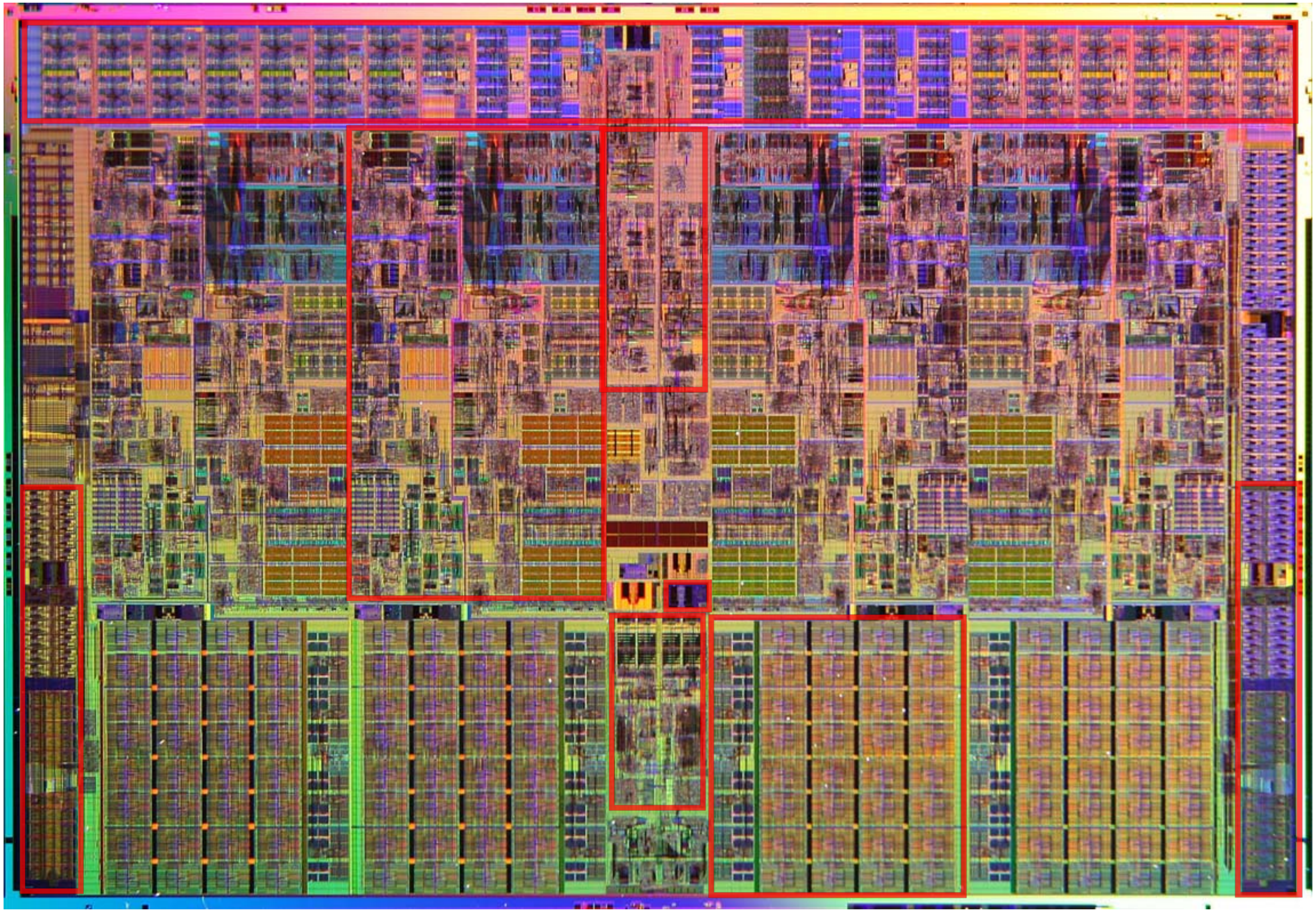
What is Nehalem?

- Nehalem (Intel® Core™ i7) is Intel's newest production x86 microprocessor on 45nm technology
  - Nehalem is the basis of a family of 45nm and 32nm products for mobile, desktop and server systems
- Nehalem *core* was derived from Merom (Intel® Core™2 Duo), with significant algorithmic and pipeline changes for:
  - Higher performance
  - Better power efficiency
  - SMT (multithreading) support
  - Higher frequency
  - Greater parallelism via increases in buffer sizes, etc.
- Nehalem *Uncore* (System Agent) is completely new
  - High speed point-to-point interconnect (QPI)
  - Cross-bar interconnect (GQ) for LLC, cores, memory and I/O
  - Integrated DDR3 memory controller
  - Power control unit (PCU)

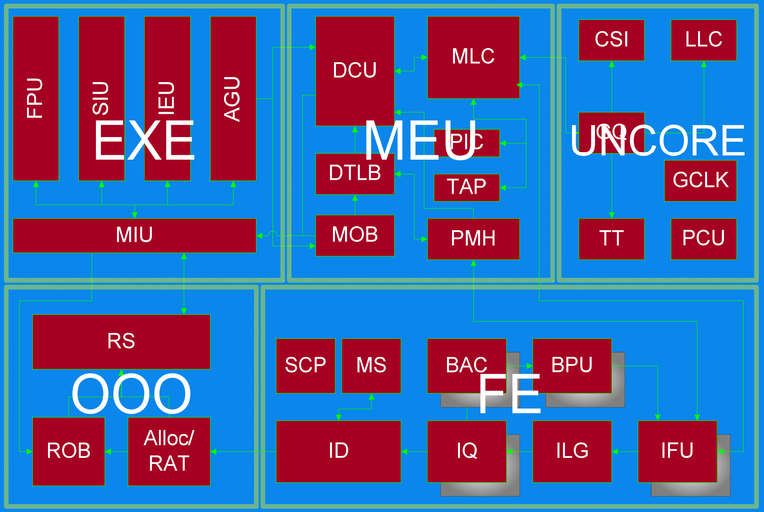
Intel® Core™2 Duo, a.k.a. “Merom”

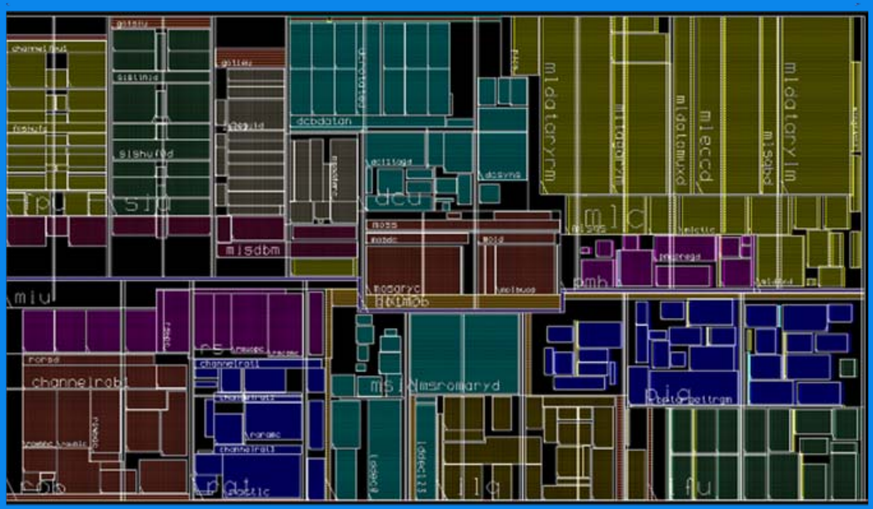


Intel® Core™ i7, a.k.a. “Nehalem”



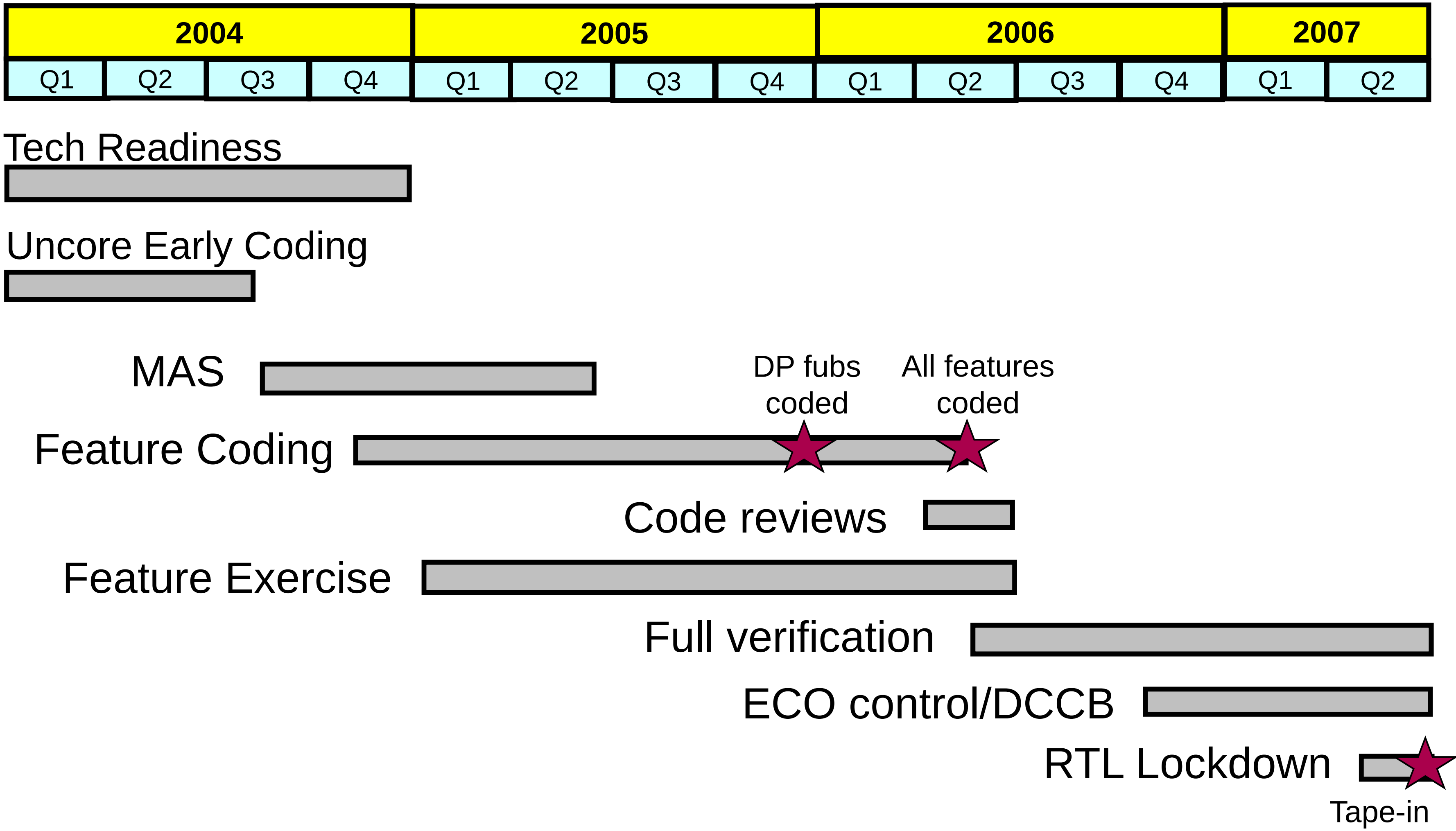
NHM Core Clusters





- Core is organized as 4 clusters
  - FE: Fetch bytes, decode instructions, provide uops
  - OOO: Resource allocation, uop scheduling, retirement
  - EXEC: FP/INT execution
  - MEU: Load/store handling

Nehalem RTL - High Level Timeline



2004 2005 2006 2007

Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2

Tech Readiness

Uncore Early Coding

MAS

DP fubs coded

All features coded

Feature Coding

Code reviews

Feature Exercise

Full verification

ECO control/DCCB

RTL Lockdown

Tape-in

SystemVerilog Motivation

- In 2004 a cross-Intel working group recommended SystemVerilog as the HDL convergence target:

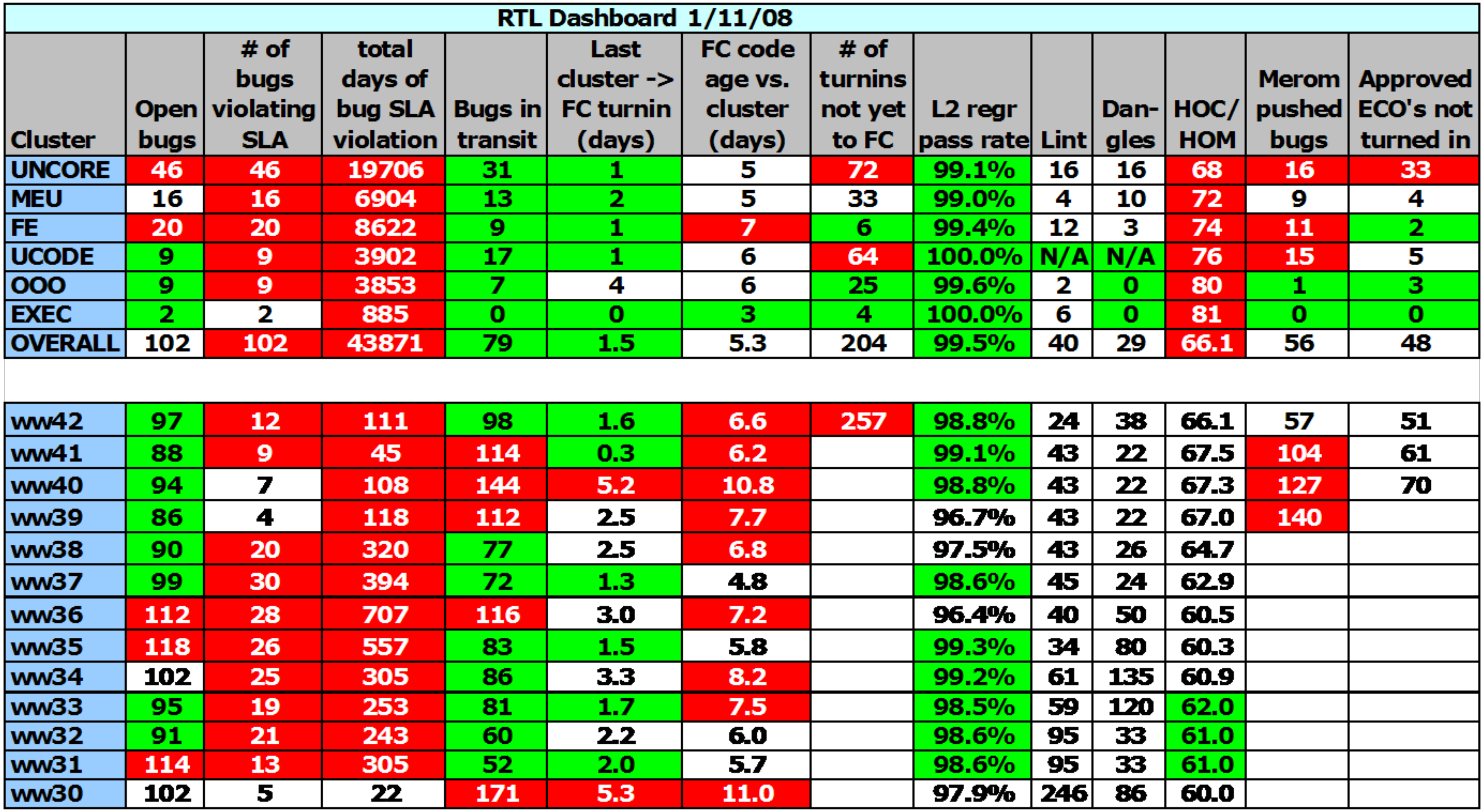
  > **"The only HDL capable of meeting the collection of varied requirements for HDL standardization is SystemVerilog (SV), given its continuing growth and acceptance as an industry standard."**

- Rationale for choosing SystemVerilog:
  - **Strategic**: Language convergence across all platform groups (CPU, chipset, graphics, external IP)
  - **Strategic**: Aligns internal resources with cutting edge tool development, and leverages EDA industry for baseline tools
  - **Practical**: Native SV language provides a significant leap in coding capabilities over iHDL, enabling RTL abstraction
- Challenges
  - Significant re-training of teams on SV and tools/flows
  - Pioneering use of new versions of internal and external tools
  - Changing RTL coder behavior to take advantage of new options

Nehalem RTL coding strategy

- Nehalem was the first design project at Intel to use SystemVerilog as the RTL development language
- Switching from an internal HDL to SystemVerilog posed a fundamental decision
  - Starting from Merom, should we morph it into Nehalem (while trying to keep it "always alive"), or recode it from scratch?
- We came up with a combo strategy
  - UNCORE and MEU were coded from scratch
  - FE, OOO and EXE were auto-translated from Merom
- In the end, we ended up with a lot of recoding in the auto-translated clusters as well, due to the scope of the uarch changes (e.g. SMT)

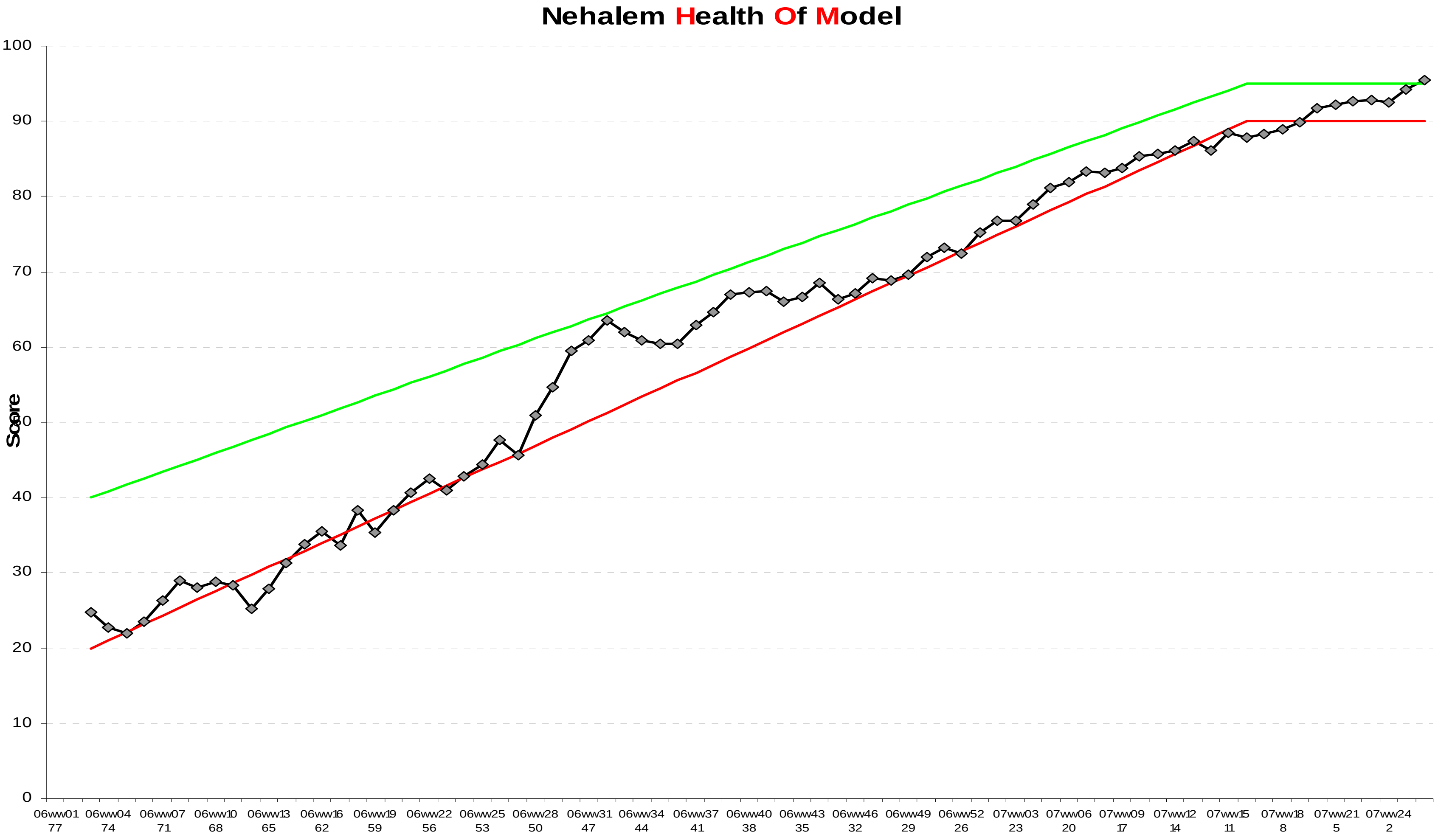
Sample RTL Dashboard



RTL Dashboard 1/11/08

Cluster	Open bugs	# of bugs violating SLA	total days of bug SLA violation	Bugs in transit	Last cluster -> FC turnin (days)	FC code age vs. cluster (days)	# of turnins not yet to FC	L2 regr pass rate	Lint	Dan-gles	HOC/ HOM	Merom pushed bugs	Approved ECO's not turned in
UNCORE	46	46	19706	31	1	5	72	99.1%	16	16	68	16	33
MEU	16	16	6904	13	2	5	33	99.0%	4	10	72	9	4
FE	20	20	8622	9	1	7	6	99.4%	12	3	74	11	2
UCODE	9	9	3902	17	1	6	64	100.0%	N/A	N/A	76	15	5
OOO	9	9	3853	7	4	6	25	99.6%	2	0	80	1	3
EXEC	2	2	885	0	0	3	4	100.0%	6	0	81	0	0
OVERALL	102	102	43871	79	1.5	5.3	204	99.5%	40	29	66.1	56	48
ww42	97	12	111	98	1.6	6.6	257	98.8%	24	38	66.1	57	51
ww41	88	9	45	114	0.3	6.2		99.1%	43	22	67.5	104	61
ww40	94	7	108	144	5.2	10.8		98.8%	43	22	67.3	127	70
ww39	86	4	118	112	2.5	7.7		96.7%	43	22	67.0	140	
ww38	90	20	320	77	2.5	6.8		97.5%	43	26	64.7		
ww37	99	30	394	72	1.3	4.8		98.6%	45	24	62.9		
ww36	112	28	707	116	3.0	7.2		96.4%	40	50	60.5		
ww35	118	26	557	83	1.5	5.8		99.3%	34	80	60.3		
ww34	102	25	305	86	3.3	8.2		99.2%	61	135	60.9		
ww33	95	19	253	81	1.7	7.5		98.5%	59	120	62.0		
ww32	91	21	243	60	2.2	6.0		98.6%	95	33	61.0		
ww31	114	13	305	52	2.0	5.7		98.6%	95	33	61.0		
ww30	102	5	22	171	5.3	11.0		97.9%	246	86	60.0		

RTL Model Health Indicator



-- NHM VTMB

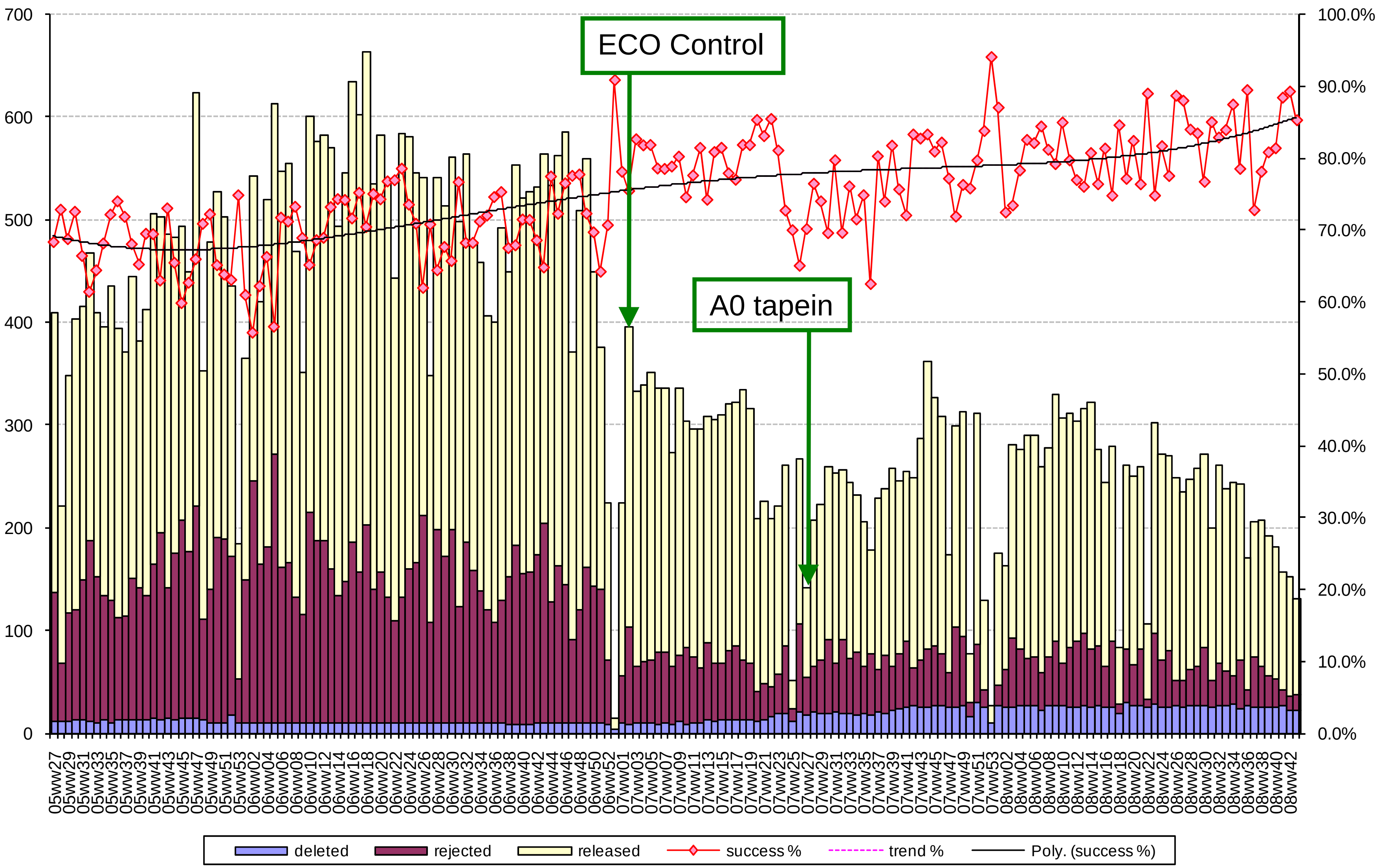
Nehalem RTL model build methodology

- Normal process was to code RTL and turn it in to a cluster model first
  - Microcode was treated as its own cluster for this purpose
  - Special handling/synchronization was needed for cross-cluster features
- As much validation as possible was done at the cluster level, using a Cluster Test Environment (CTE)
  - Tried to flush out as many bugs as possible at this level
  - When the cluster appeared to be healthy then the code was promoted to full-chip
- RTL model turnin/release methodology based on “Gatekeeper” (GK)
  - Automated system for processing code turnins in parallel
  - Nightly automatic release of fully approved code for each cluster and full-chip model
- Issues
  - High turnin reject rate caused computing thrash - addressed by “Mock Turnin”
  - Queues defaulted to FIFO processing - no automatic priorititization of critical turnins
  - Single GK guardian for queue exception handling and issue resolution
  - Increased turnin latency at critical points of the project

Turnin process

- GK processed a huge volume of turnins
  - There were ~33K cluster turnins that were accepted
  - Many more full-chip turnins, plus turnins that GK rejected
- At times the GK pipeline needed to be “actively managed” (read: unclogged)
  - GK by default provides equal opportunity for any turnin
  - At times, human intervention is needed
- Cluster->FC turnins were sometimes very painful
  - Not a big surprise given the volume of turnins
  - Required active management and prioritization for periods of time
  - Needed one person who made the calls for FC pipeline
- Uncore consistently had difficulty getting to FC (and/or pulling back down to cluster)
  - Should have split the Uncore into smaller pieces
  - Subsequent projects have taken this approach

GateKeeper - Weekly Turnins Processed



ECO Control

A0 tapein

Microprocessor Validation & Verification

- "**Verification** is a quality process used to evaluate whether or not a product, service, or system complies with a regulation, *specification*, or conditions imposed at the start of a development phase"*
- "**Validation** is the process of establishing documented evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements"*
- Successful microprocessor verification needs three things:

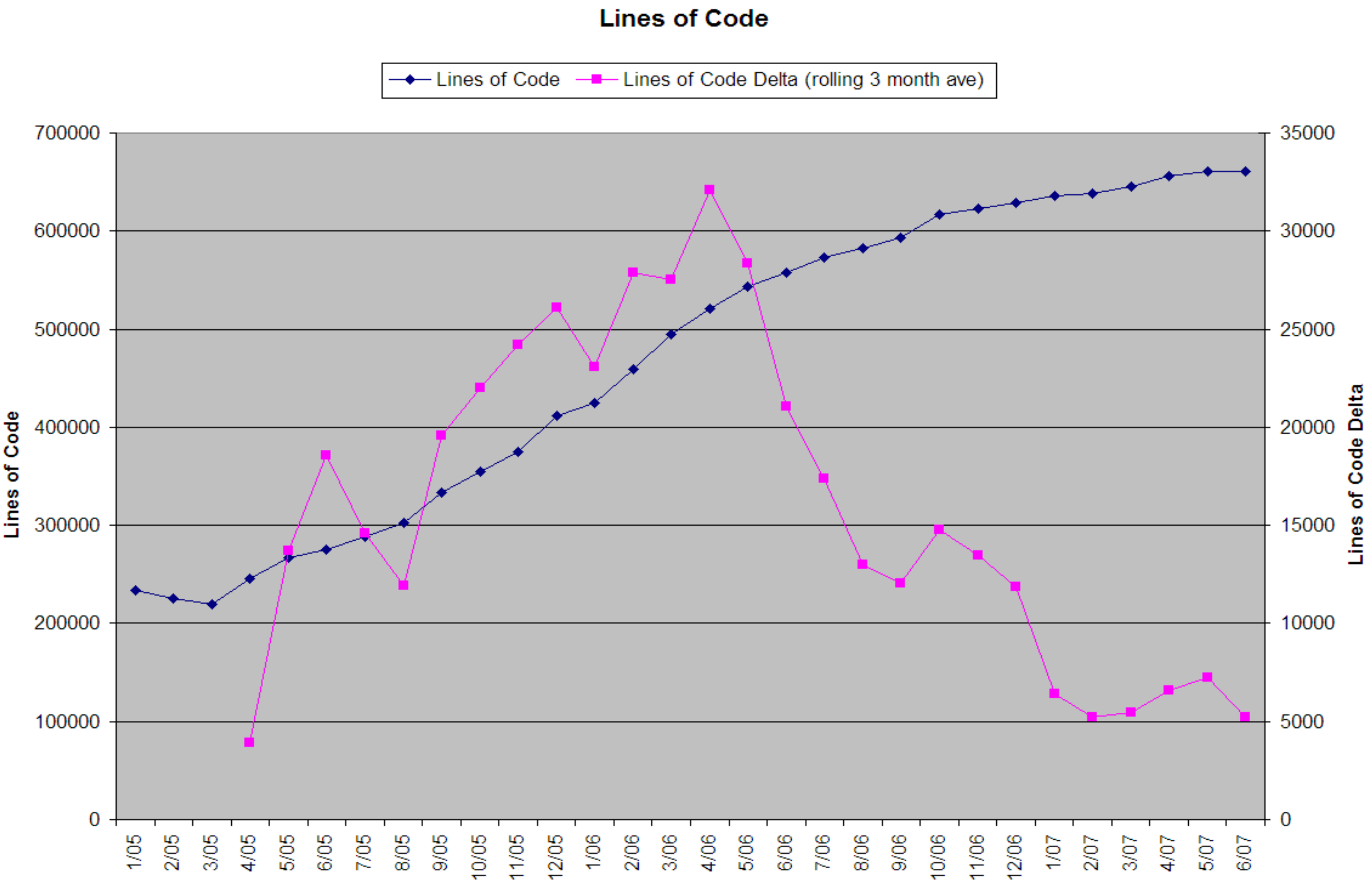
1. A way to stimulate the device under test (*testing*)
2. A way to determine if the device produced the intended results (*checking*)
3. A way to determine how much of the design has been tested (*coverage*)

* Wikipedia

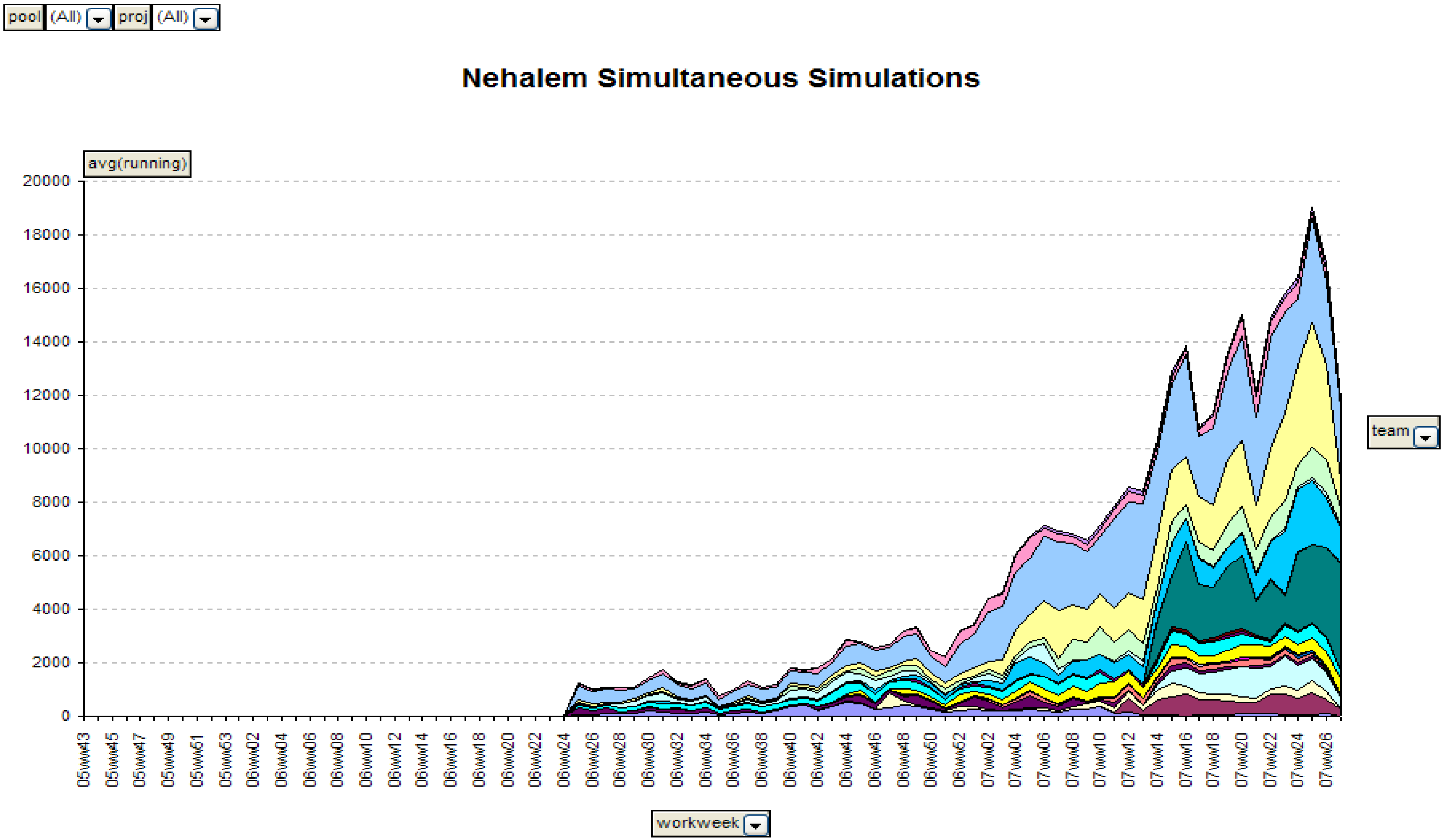
Nehalem verification environment

- RTL model was MUCH slower than real silicon
  - A full-chip simulation with checkers runs at O(Hz) on a ‘best of breed’ Intel Xeon® compute server
  - We used a distributed network of thousands of compute servers to get tens of billions of simulation cycles per week
  - As much dynamic verification as possible was done at cluster rather than full-chip
- Code churn was a major challenge
  - Constant stream of changes due to feature changes/additions, bug fixes, physical design feedback
  - Every model needs to be regressed to prevent changes from cratering model health
  - Automated process developed for continuous model build and regression
- Automation was applied wherever possible
  - Pre- and post-processing for test generation and debug traces
  - ‘Triage’ tools to bucket failures based on similar signatures
  - Visualization tools to drive coverage analysis and identify holes

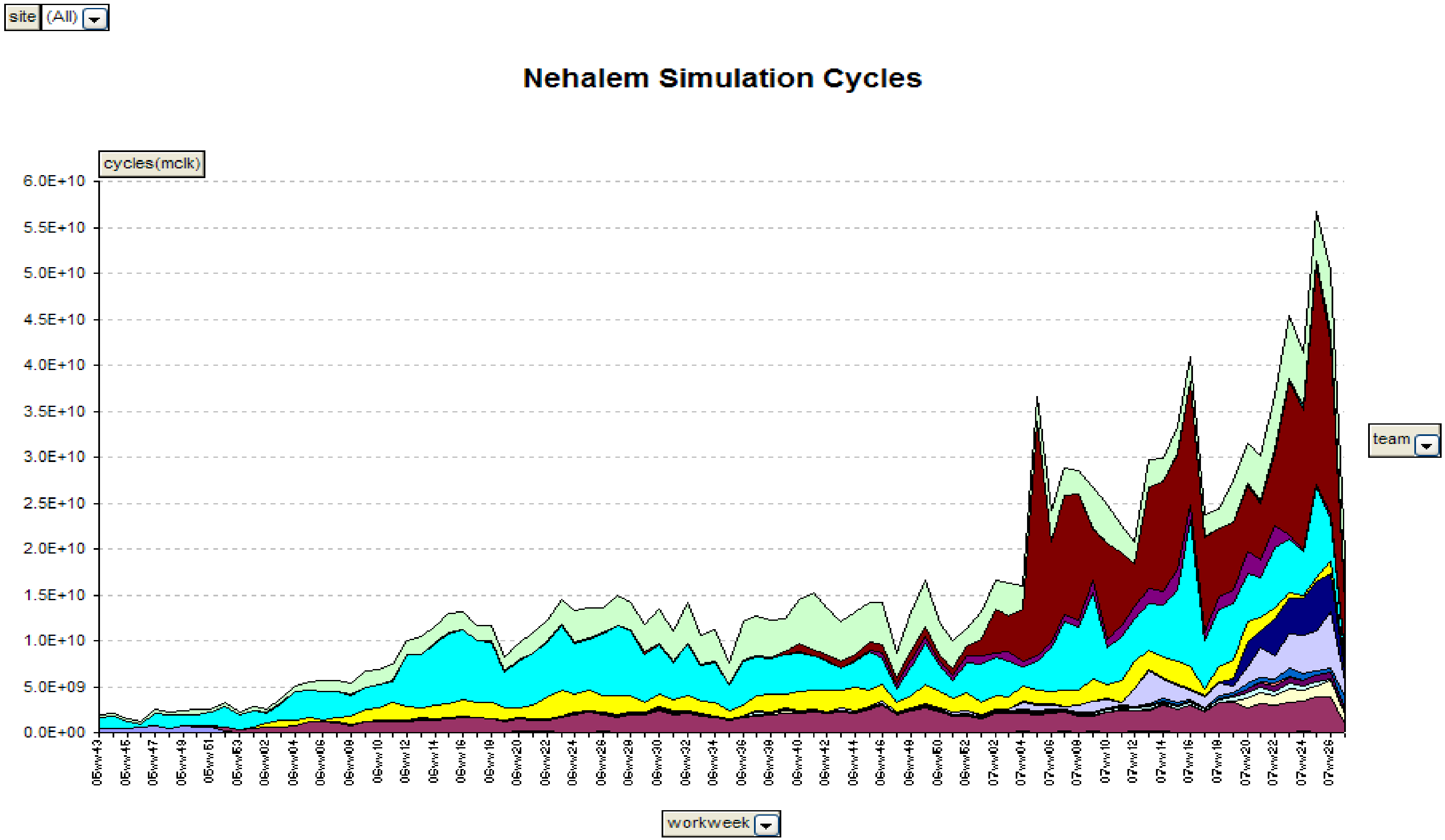
Nehalem RTL code change



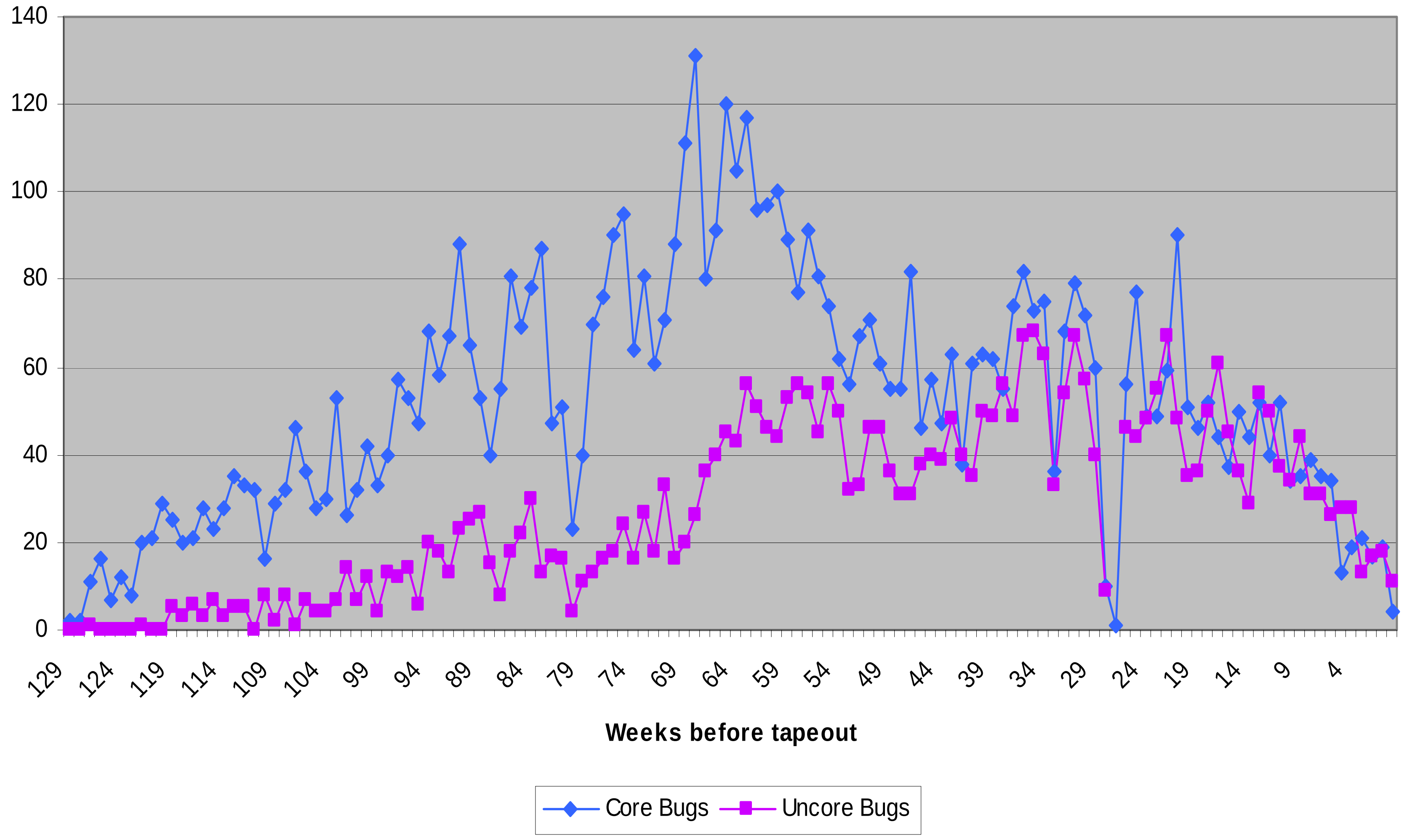
Nehalem weekly simulation jobs



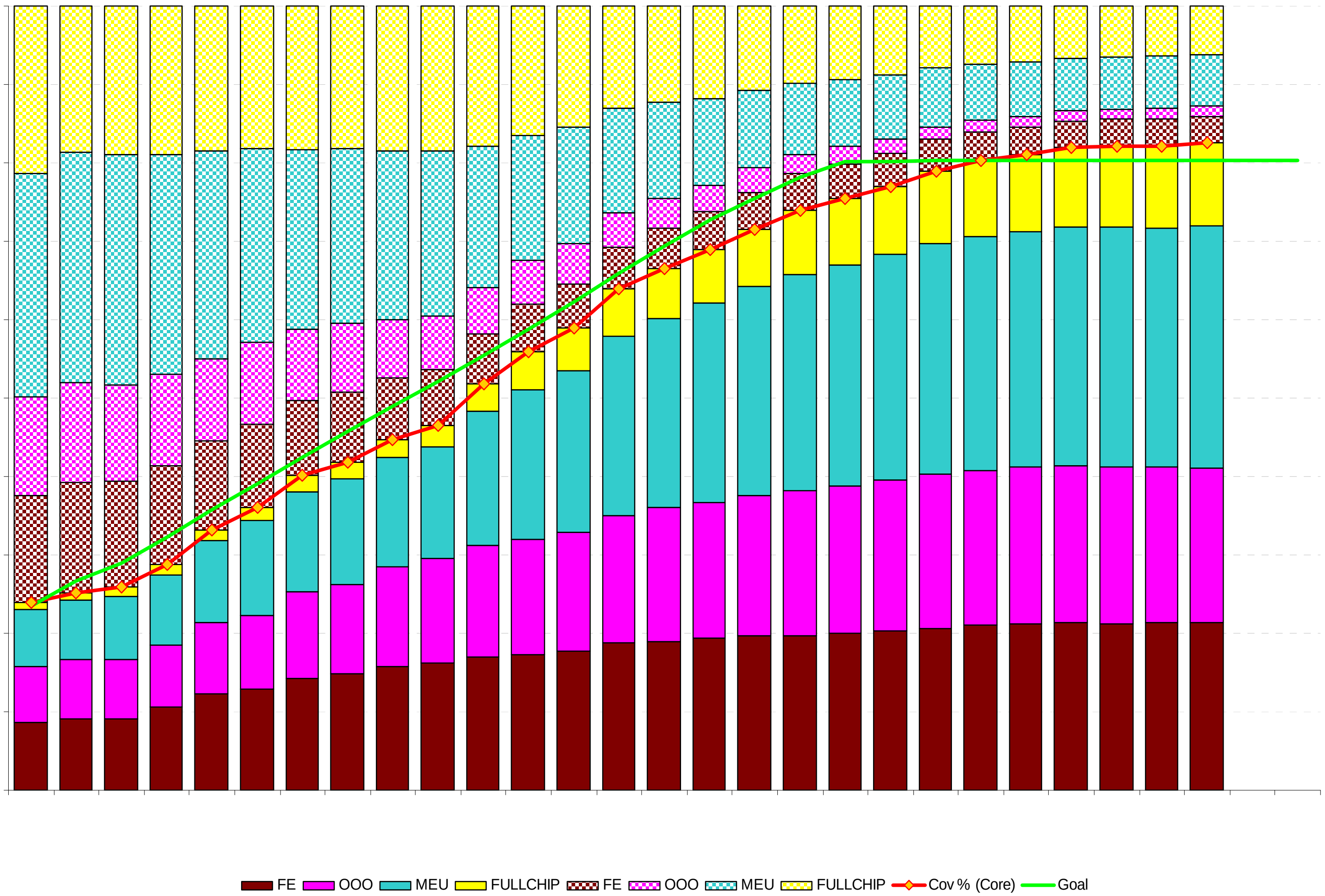
Nehalem weekly simulation cycles



Nehalem Weekly Bugs



Cluster-Level Coverage



-- NHM VTM B

Verification results

From Jim Brayton, Nehalem Design Manager, at 6:53 a.m. on 9/1/2007:

"I am thrilled to announce that the Nehalem/Tylersburg Platform has successfully booted DOS, Windows, and Linux on A-0 silicon!

First silicon arrived Monday WW35.1 @ 2:57 p.m., with this significant milestone completed Saturday WW35.6 @ 2:10 a.m.

This marks a historic moment in the history of Intel: a brand new native quad-core CPU, chipset, QPI Interconnect and an integrated DDR3 memory controller, all working together in concert, 4 days and 13 hours from receiving first silicon.

Congratulations to all the teams who have helped us reach this monumental milestone! Take a moment to cherish this proud moment.

Onward to PRQ!"