Hybrid Architecture Validation

Suneil Mohan
Agenda

- Introduction to Intel Hybrid architecture
- Pre-Silicon challenges and solutions
- Post Silicon functional validation methodologies
- OS Based Verification
- Functional Validation Sign Off

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Intel Performance Hybrid Architecture\textsuperscript{[1][2]}

Designed to deliver efficient high-compute performance in a large dynamic power and performance range

**Performance-cores**
- Larger, high-performance cores designed for speed while maintaining efficiency.
- Tuned for high IPC (instructions per cycle) and high turbo frequencies.
- Supports hyper-threading

**Efficient-cores**
- Smaller, with multiple E-cores fitting into the physical space of one P-core.
- Designed to maximize CPU efficiency, measured as performance-per-watt.
- Ideal for scalable, multi-threaded performance. Does not support hyper-threading
Pre-Silicon Verification

- 3-prong approach
- Each CPU team performs dedicated IP validation (Simulation, Emulation & FPGA environments)
- SoC validation team performs
  - Integration
  - Hybrid Validation
- Periodic, Consistent check-ins between each Core and the SoC
IP Validation teams (E-core and P-core)

- Experts in their uArch
- Validate their IP in isolation
- Support SoC / Hybrid / Integration Debug
SoC Validation team

- Validate Hybrid Integration
- Run Hybrid Workloads
- Verify Key Performance indicators
- Coordinate debug with the appropriate IP team
Extensive Pre-Silicon Microcode Validation

- Built a combined Microcode simulator model.
- Allows validation of interaction before Simulation/Emulation models are built
- Faster turnaround of short experiments for validation
Post Silicon Validation

- Similar approach to Pre Si val
- Each CPU team performs dedicated IP validation
- SoC validation team performs
  - Integration
  - Hybrid Validation
- Periodic, Consistent check-ins between each core family and the SoC
OS Based Verification

What if Synthetic Workloads are not stressful enough?

- External Software may have corner case behaviors that are not always modelled.
- External Software may do things that are not completely to ‘spec’
- Need to see how random task switching might behave

1. Analyzed Open-Source OS scheduler source code for task switching
2. Ran Task Switching OS Scheduler code on the combined microcode simulator model to understand the behavior
3. Built a randomized OS task switcher for Post Silicon validation
Functional Validation Sign off

**Pre-Silicon**

- Pass rates for synthetic test content.
- All failures accounted for, understood and dispositioned.
- Coverage data analysis complete.
- Power and Perf data collected, analyzed and within expected ranges.

**Post Silicon**

- Acceptable pass rates for synthetic test content.
- All failures accounted for, understood and dispositioned.
- Power and Performance data meeting projections.
- 100% pass rate for the OS task switching tests.
- Thread Director working as expected.
# References and Additional Resources


<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
</table>