



The QFP Packet Processing Chip Set

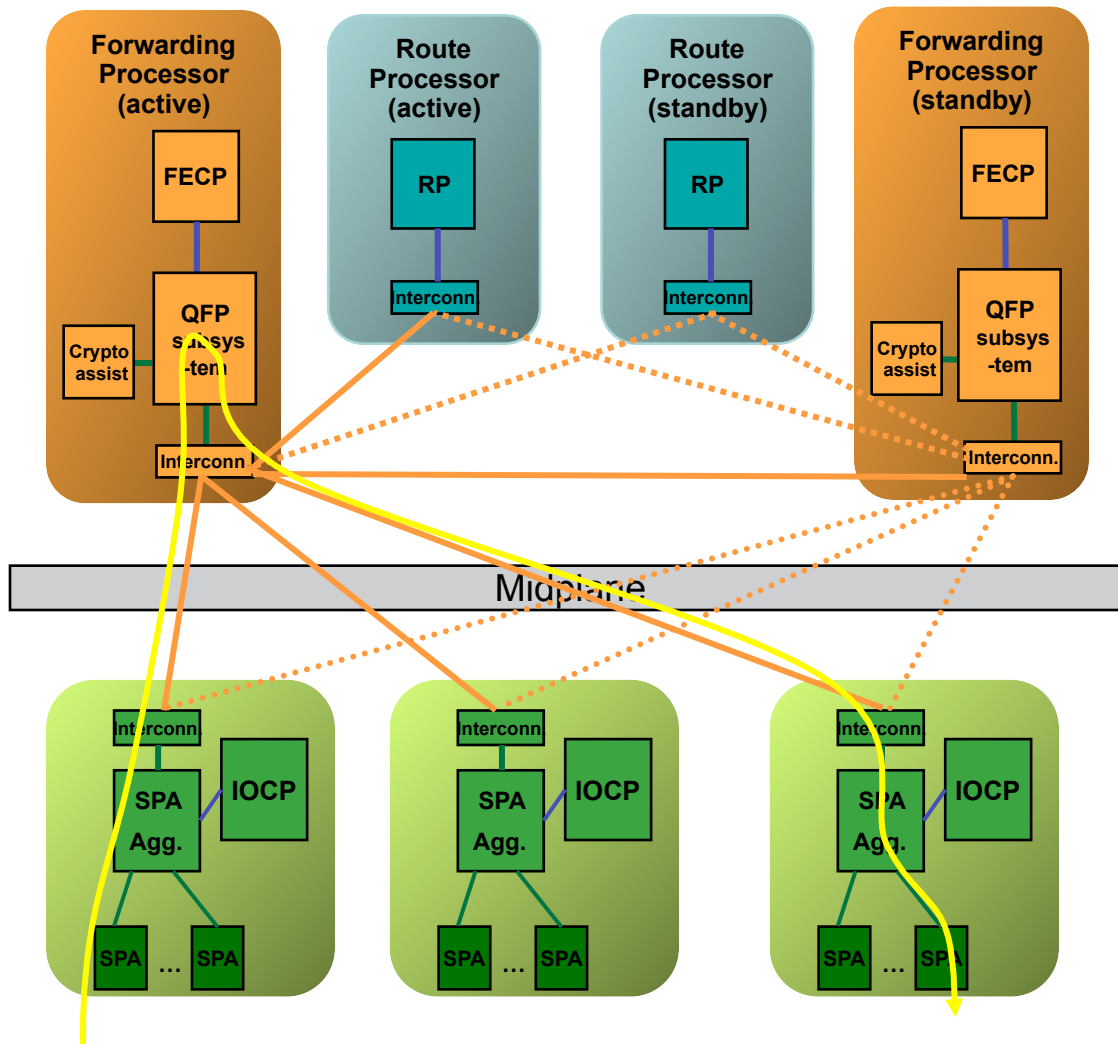


**Will Eatherton (Speaker), Don Steiss (Speaker),
James Markevitch
Cisco Systems,
Cisco Development Organization**

Agenda

- Overview of Cisco's ASR-1000 Quantum Flow Processor
- Architecture and Implementation Tradeoffs
- Software Development and Debug Environments
- Silicon Details and Design Methodology
- Results
- Summary

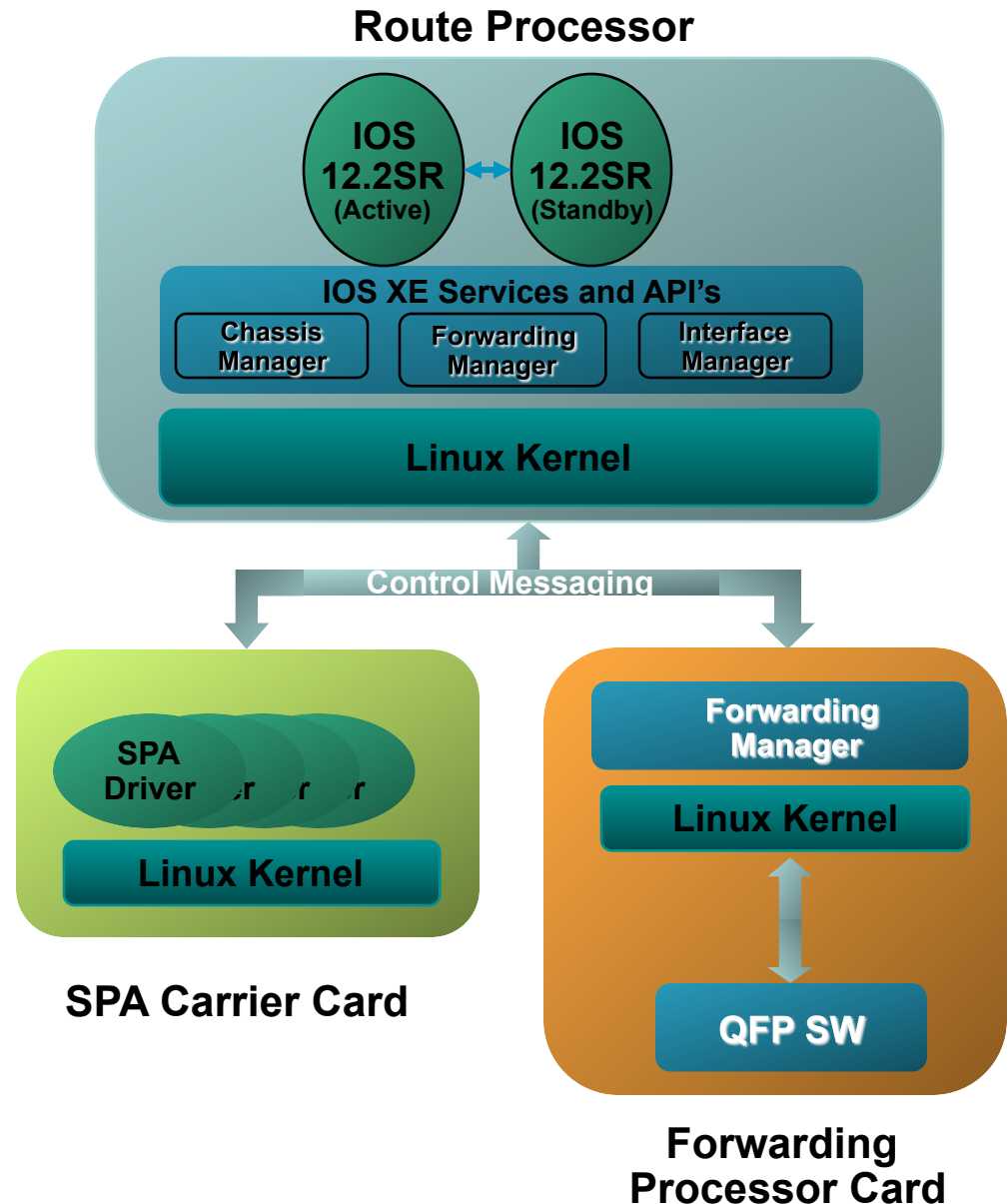
ASR 1000 Building Blocks



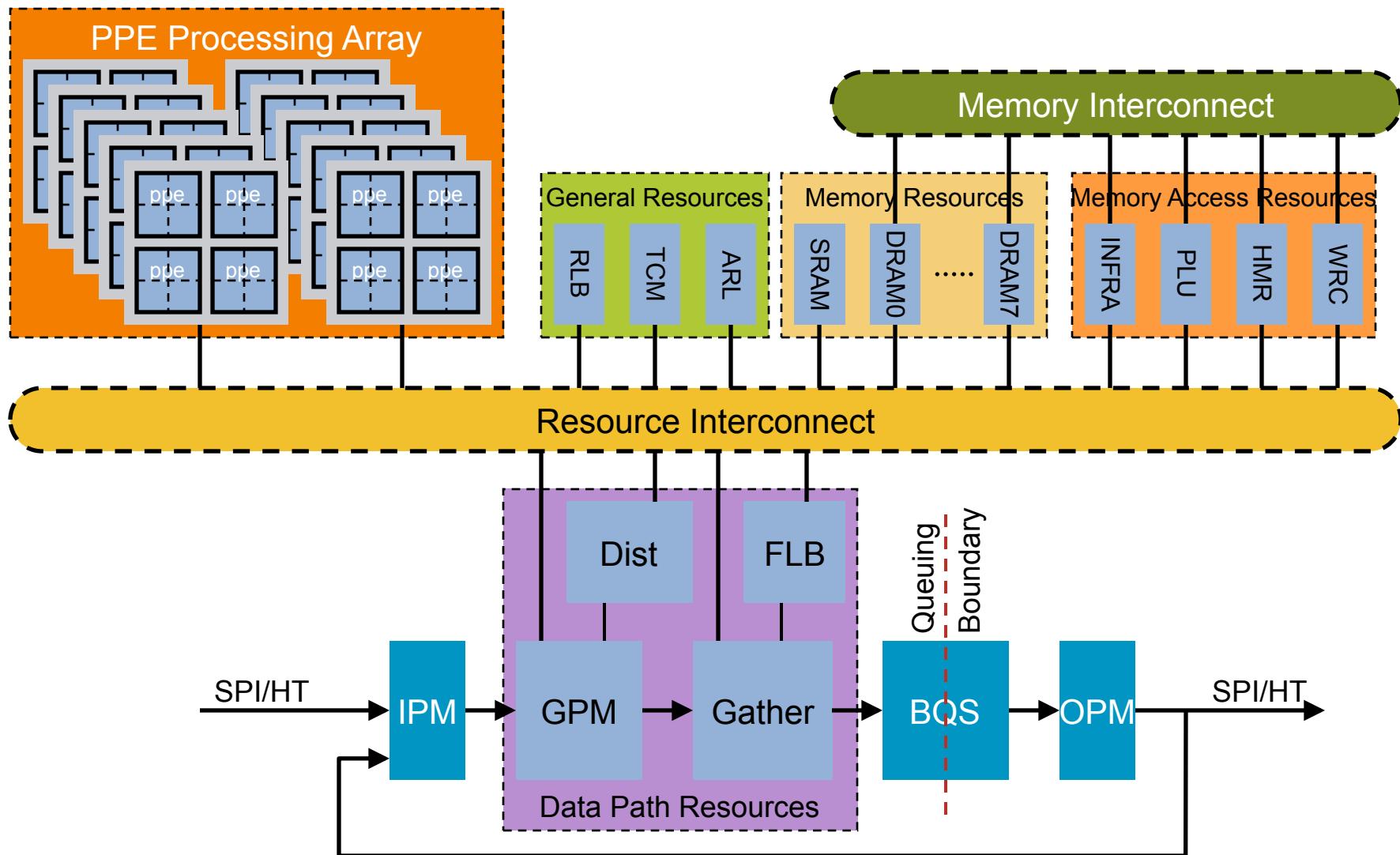
- 3 Chassis types
6 / 4 / 2 RU
- RP (Route Processor)
Handles control plane traffic
Manages system
- ESP/FP (Forwarding Processor)
Handles forwarding plane traffic
- Shared Port Adapter Carrier Card
Houses the SPAs
- SPAs
Provide interface connectivity
- Centralized Forwarding Architecture
All traffic flows through the ESP/FP

ASR 1000 Software Architecture – IOS XE

- IOS XE is:
 - IOSd
 - plus IOS XE Services/API's
 - plus QFP Datapath Software
- IOS runs as its own Linux process for control plane (Routing, SNMP, CLI etc). Linux kernel with multiple processes running in protected memory
- QFP Datapath Software
 - Fully multiprocessor code base covering range of features across Security, Voice, Deep packet inspection
 - Code base is ANSI-C, written to run based on packet loop and with function libraries for OS like services (Memory Mgmt, Timers, ..)



QFP10 Architecture



Architecture and Implementation Tradeoffs

- Programming Environment

Assembly code vs. HLL: ANSI C with typical C runtime system for portability and productivity. Assembly coded Hardware Abstraction Layer (HAL).

- Instruction Set Architecture and Implementation

ISA make vs. buy: off-the-shelf ISA to accelerate production software development.

Implementation make vs. buy: custom microarchitecture and circuits to improve power, performance and area.

- Latency Hiding

Implicit vs. explicit mechanisms: Both; threading is implicit, non-blocking messages and data prefetch are explicit.

Architecture Tradeoffs – Continued

- Memory Subsystem

Ordering: intentionally weak. Loads and stores have no guaranteed MP ordering. Barriers, indivisible operations and serialization are provided. Atomic ordering facilities are provided by special resources.

D\$ parameters: enough capacity to cover parts of the working set with high locality, allocation policies to avoid cache pollution.

I\$ parameters: as large as possible and large second level cache bandwidth to reduce thread performance interference.

- Processor and Resource Communication

Address mapping and/or message passing: message passing hardware infrastructure with an address mapped layer visible in the programming model.

Architecture Tradeoffs – Continued

- Accelerators: what goes in hardware ?

Message passing coprocessor: performance (latency hiding), code size, encapsulation.

Scheduling: performance, encapsulation.

Crypto: performance (parallelism), stable algorithms.

Lookup/Classification/Deep Packet Inspection algorithms: performance (parallelism), stable algorithms.

Traffic Manager

- 128K queues and the ability to set Max/Min/Excess BW
- Two Priority Queues can be enabled for each QoS policy applied.
- The number and makeup of layers inside the QoS policy are flexible. The possible hierarchies are not tied to any existing hierarchies used in networks today.
- The traffic manager can schedule multiple levels of hierarchy in one pass through the chip.

Queuing operations can be split across multiple passes through the chipset (for example, ingress shaping or queuing followed by egress queuing).

Software Code Development & Debug Environments

- Simulation, code debug and hardware debug software components integrated in the Eclipse platform.
 - Multiple speed vs. accuracy processor simulation options
 - Compiler, assembler, linker, code debugger
 - Chipset state inspection (registers, interrupt sources)
- Enabled production code development and performance tradeoffs in parallel with hardware development.
- The same environment is available for hardware analysis and debug running on the router's control plane processors.

QFP Code Debug Environment

The screenshot displays the GDB Platform interface for debugging a QFP code. The main window is divided into several panes:

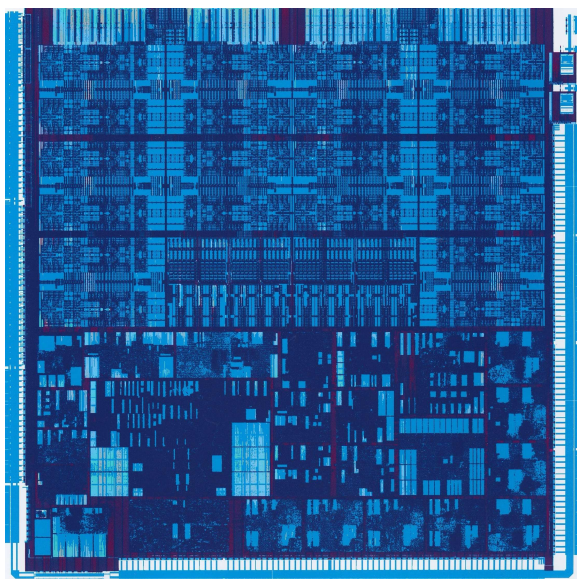
- Debug Console:** Shows the simulation status: "Simulation 9/10/06 1:39 PM (Suspended)". It lists two cores, each with two suspended contexts. The current context is "2 main() at /vob/cppsw/platform/c12000/ingress/c12000_ingress.c:35 0x801c8c27".
- Registers:** A table showing register values for AR0..AR31, Misc, and Coprocessor (Core 0, Context 0). The "random" register is highlighted with a value of 1658605825.
- Expressions:** Shows the evaluation of expressions like "psv_ipv4_pbr_end_sim" and "pkt_state->switching_state".
- Code Editor:** Displays the C source code for "c12000_ingress.c". The line "pkt_state = thread_get_pkt_state();" is highlighted, corresponding to the current context.
- Disassembly:** Shows the assembly code for the current context, with the instruction "s32i.n a10, a1, 0" at address 0x801c8c30 highlighted. An address breakpoint is set at 0x801c8c38.

QFP Hardware Debug Environment Example

Remote Proxy Debug [demo-proxy - Full Control] 3/2/07 11:02 AM

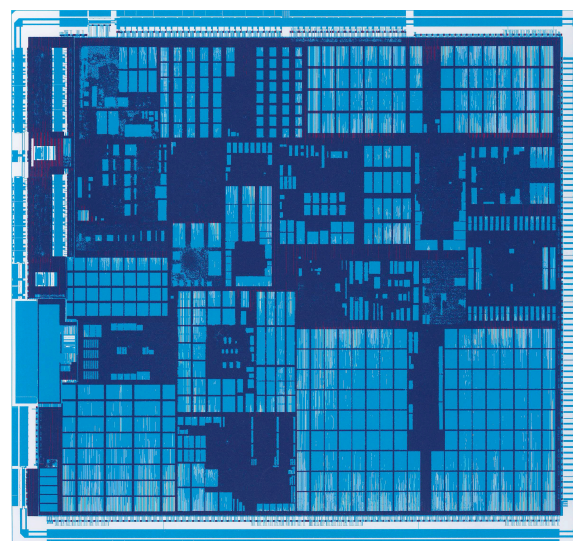
Name	Address	Datum	Full name	Description
Physical				
hedp			HED CSR Address Map	Address Map of HED CPU register
hed_halted_in_63_0_leaf_int			Hed Halted_in_63_0 Leaf Interrupt Register File	This is the set of Hed Halted_in_63_0 Leaf Interrupt Register File
hed_halted_in_127_64_leaf_int			Hed Halted_in_127_64 Leaf Interrupt Register File	This is the set of Hed Halted_in_127_64 Leaf Interrupt Register File
hed_halt_out_63_0_leaf_int			Hed Halt_out_63_0 Leaf Interrupt Register File	This is the set of Hed Halt_out_63_0 Leaf Interrupt Register File
hed_halt_out_127_64_leaf_int			Hed Halt_out_127_64 Leaf Interrupt Register File	This is the set of Hed Halt_out_127_64 Leaf Interrupt Register File
hed_elam_leaf_int			Hed Elam Leaf Interrupt Register File	This is the set of Hed Elam leaf interrupt Register File
hed_leaf_leaf_int			Hed Leaf Leaf Interrupt Register File	This is the set of Hed Leaf leaf interrupt Register File
hed_top_hier_int			Hed Top Hierarchical Interrupt Register File	This is the set of Hed Top hierarchical Interrupt Register File
int_stat	0x24044080	0x0000000000000025	Hed Top Interrupt Status Register	This register contains the Hed Top Interrupt Status Register
int_leaf	bit 0	1	An Interrupt Occurred in the HED's CSR Leaf	Read the hed_leaf interrupt register
int_halted_in_63_0	bit 1	0	Halted 63:0 input Interrupt	Halted 63:0 input from top level
int_halted_in_127_64	bit 2	1	Halted 127:64 input Interrupt	Halted 127:64 input from top level
int_halt_out_63_0	bit 3	0	Halt 63:0 output Interrupt	Halt 63:0 output to top level
int_halt_out_127_64	bit 4	0	Halt 127:64 output Interrupt	Halt 127:64 output to top level
int_elam	bit 5	1	ELAM Interrupt	ELAM Block Interrupt
int_en_rw1c	0x24044088	0x0000000000000000	Hed Top Interrupt Enable Register RW1C	This register contains the interrupt enable register RW1C
int_en_rw1s	0x24044090	0x0000000000000000	Hed Top Interrupt Enable Register RW1S	This register contains the interrupt enable register RW1S
gal				

QFP10 Chipset



Multi-Core Packet Processor

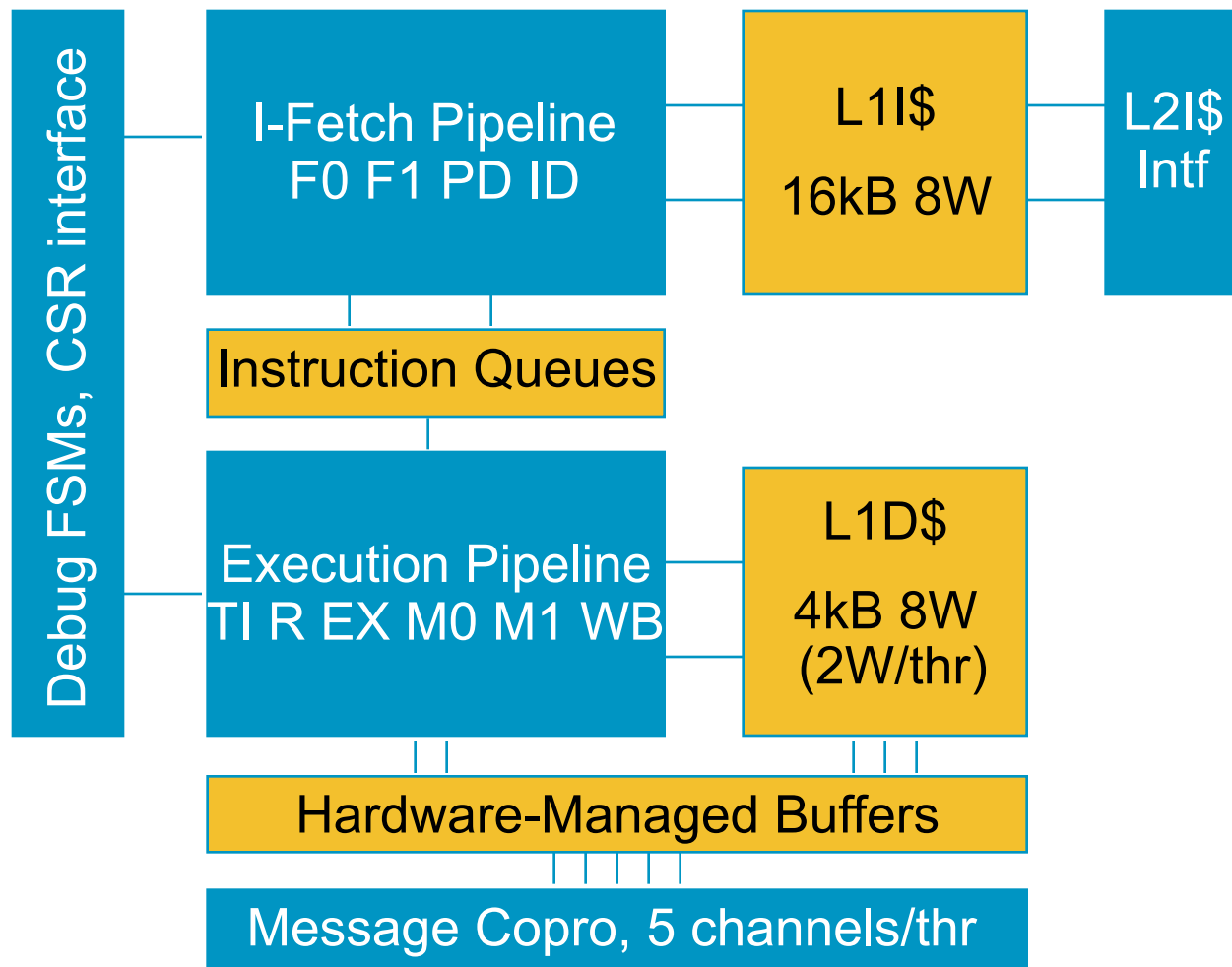
- 1.2 GHz/400 MHz
- 40 custom multi-threaded CPUs
- TI 90nm, 8-layers metal
- 19.54 x 19.54 (382 mm²)
- 307 million transistors
- 20 Mb SRAM
- 1019 I/O, including 800 MHz DDR



Traffic Manager and Interface Chip

- 400 MHz
- Buffering, 200K queues, hardware HQF scheduling
- TI 90nm, 8-layers metal
- 19.0 x 17.48 (332 mm²)
- 522 million transistors
- 70 Mb SRAM
- 1318 I/O, including 800 MHz DDR

Packet Processing Engine (PPE) Structure



Hardware Design Methodology

- Customer Owned Tooling (COT) with GDSII handoff to the foundry partner.

 - Cell library designed and characterized by Cisco

 - SRAM compiler and I/O IP from Texas Instruments.

 - Extensive crosschecking at Cisco and Texas Instruments.

- Synthesis/P&R outside of processor array

 - Multiple commercial synthesis and physical design tools.

 - In-house tools for RTL code generation and documentation.

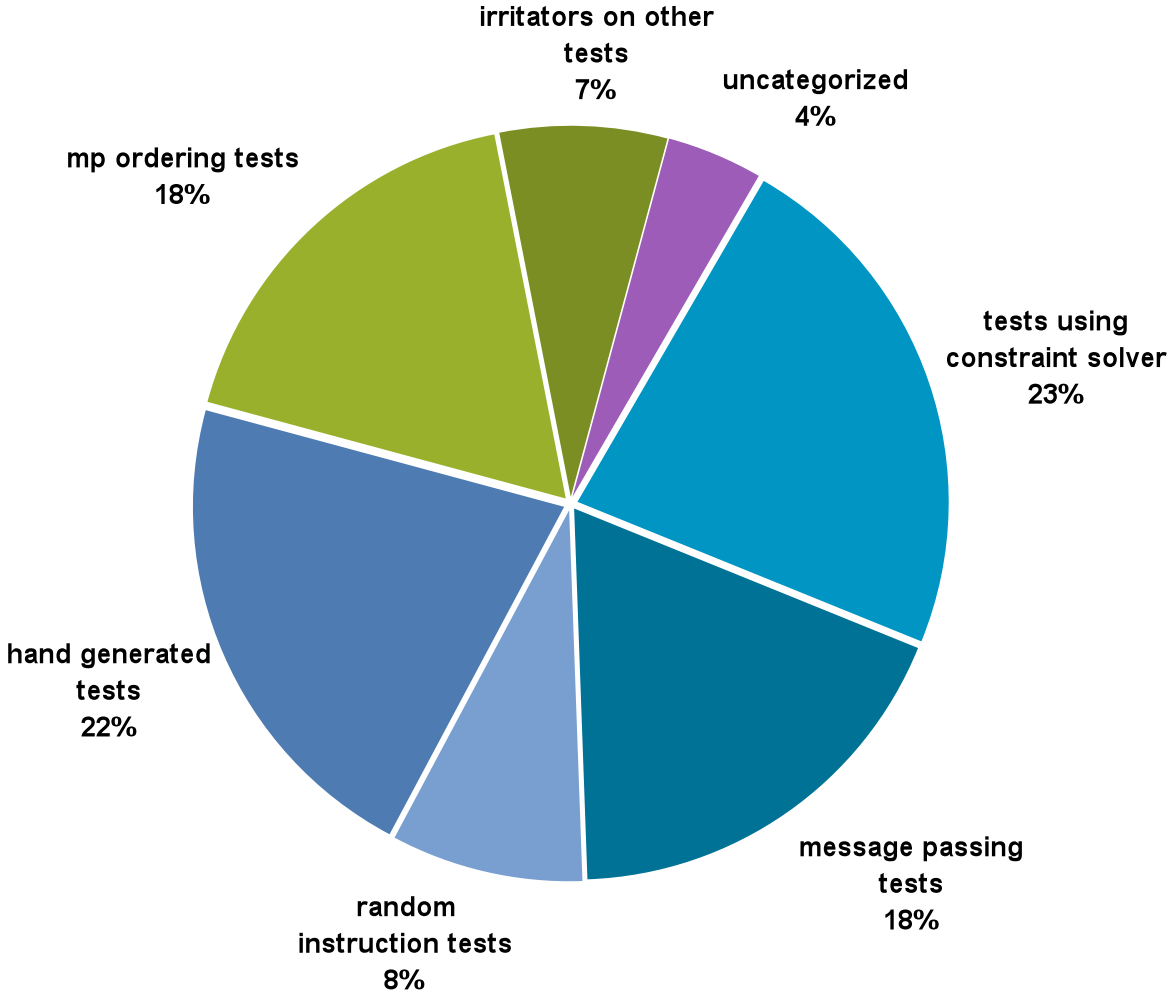
Hardware Design Methodology - Processor

- Cell-based to leverage ASIC tools.
- Static CMOS with selective use of domino circuits
- Schematic design entry with physical specifications in instance names for fast, deterministic placement.
- Polygon-level artwork in critical modules.
- Autorouted signals with many pre-routes above the cell level
- Multiple functional reference models and equivalence checking

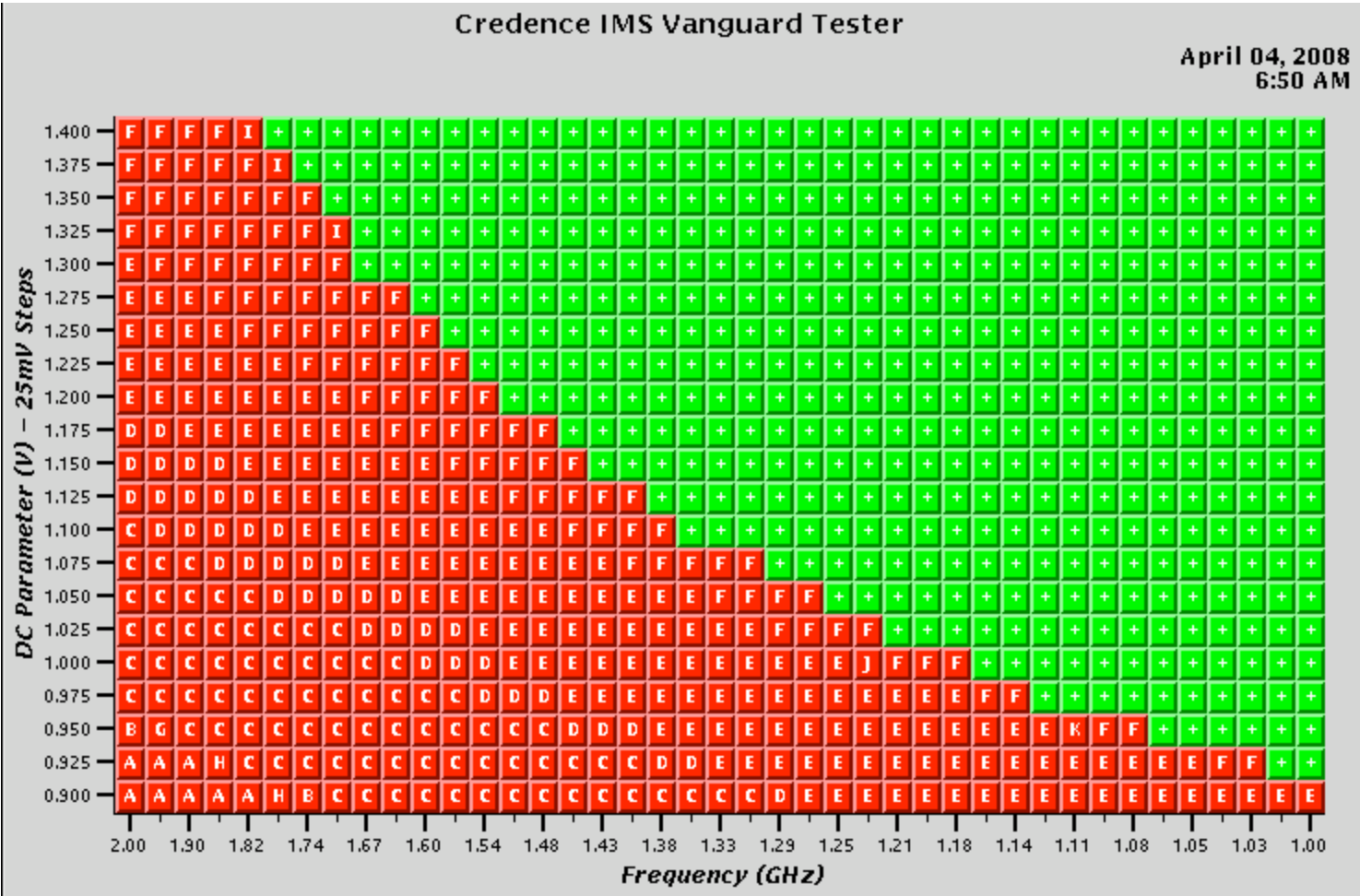
Processor Design Verification

- Strong block level verification methodology
- Heavy use of code generators and constraint solving
- Processor verification statistics:
 - 1,903,282 RTL simulation runs
 - 95,185,665,935 total clock cycles
 - 17,421 total test failures

Processor Design Verification – Continued Failures by Test Source



Results – Shmoo at Room Temp



Results - Functional

- On rev 1.0 silicon:
 - Silicon delivered for system integration, January 2007
 - First packets through production software, February 2007
 - Customer testing, August 2007
- Product Launch March 4, 2008
- All “mission mode” registers and instructions in the ISA are used in production software.

Summary

- Router Architecture today involves partitioning of software across multiple CPU complexes, Multiple Cores per CPU, Multiple threads per Core
- Processor architecture in networking is still evolving
- Many architectural and implementation tradeoffs are a result of software engineering complexity that rivals hardware complexity and often 10x more staffing.

