Hot Chips-18

Design of a Reusable 1GHz, Superscalar ARM Processor

Stephen Hill

Consulting Engineer

ARM - Austin Design Centre

22 August 2006

Outline

- Overview of Cortex™-A8 (Tiger) processor
- What is reusability or redeployability?
- Why is it important to the Cortex-A8 processor?
- Effects on design flow & microarchitecture
- Interaction of energy efficient & reusable design
- Summary

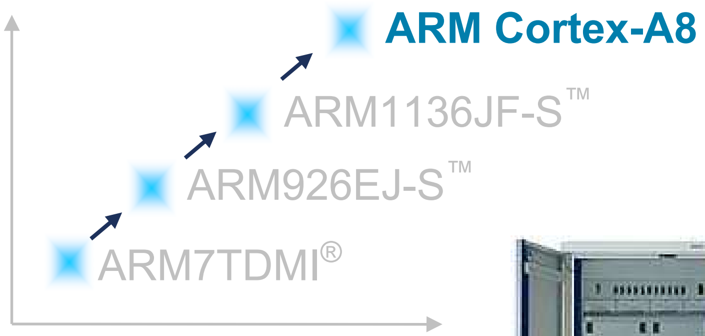
Cortex-A8 Microarchitecture Highlights

Goals:

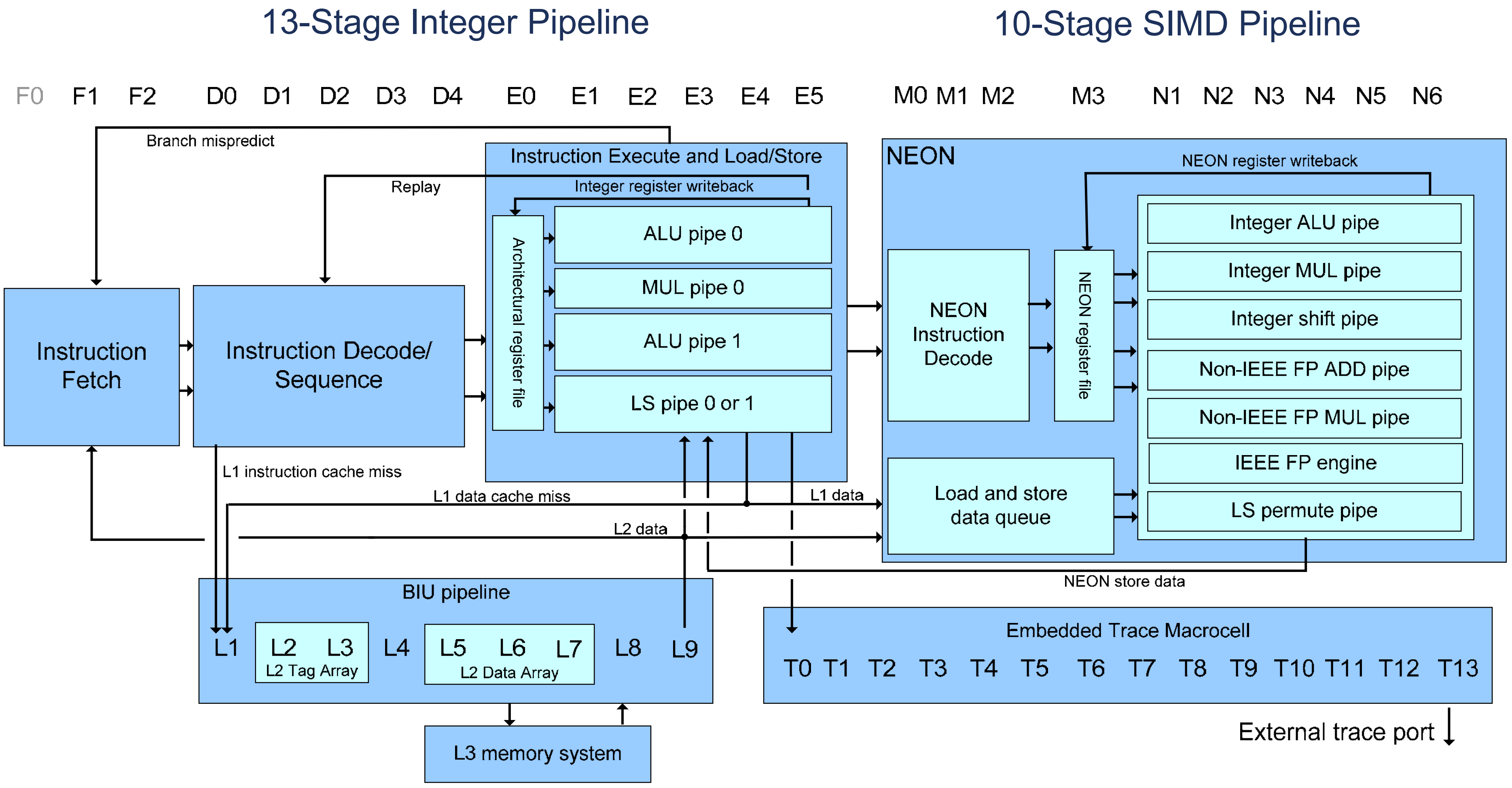
- A new level of performance from ARM
- Uphold core values of energy efficiency & flexibility
- Support both mobile and tethered applications

Microarchitecture:

- Dual issue superscalar
- In-order/statically scheduled
- 13-stage integer pipeline
- 10-stage int+float SIMD unit
- 2 level branch prediction
- 2x32/16/0K level-1 caches
- Integrated 0 to 2M level-2 cache



Cortex-A8 Processor Pipeline



Reusability/Redeployability

What is it?

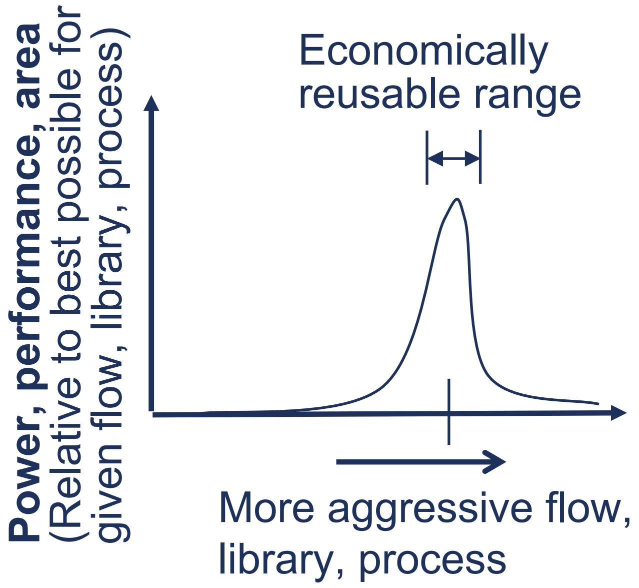
- **Basics:** Well commented RTL model, good documentation, system development models, software development models, test vectors…
- **Microarchitecture:** How well the microarchitecture can be ***usefully*** implemented in new EDA flows, cell libraries, processes and process generations… ***for a reasonable effort/cost***

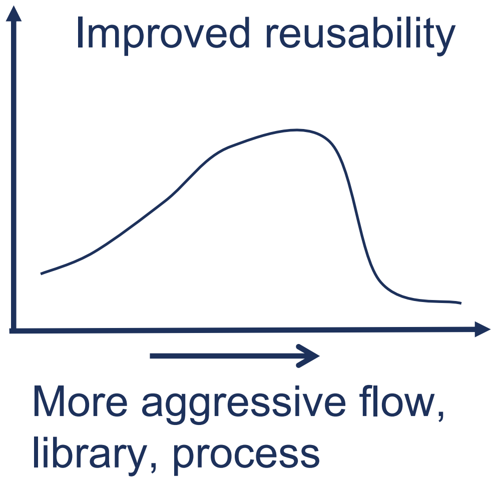
Why is it important?

- Economics of intellectual property
- Flexibility compensates for imperfect foresight
- Fabrication advances keep coming

Some Factors Effecting Reusability

Reduced reusability	Improved reusability
Microarchitecture reflects strengths and weaknesses of one process or circuit style	Microarchitecture avoids strong process ties
Non-standard logic style, dynamic logic, lots of CAMs and wide input gates	Standard complementary logic gates, RAMs and register files.
Mixed-edge or level sensitive clocking	Pos-edge triggered clocking
Home grown tools used extensively	Standard SOC EDA tools used by default. Home-grown tools are carefully managed
Non-synthesizable sections of hand-implemented code	100% synthesizable code. Non-synthesis implementation only where necessary
Circuits & custom layout pushed to close timing and power	Timing and power fixes fed into microarchitecture and RTL
Setup/hold timing verified for one process	Timing verified in a wide range of processes



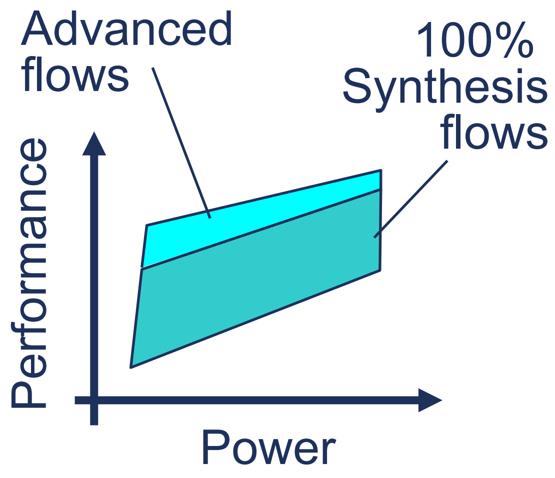


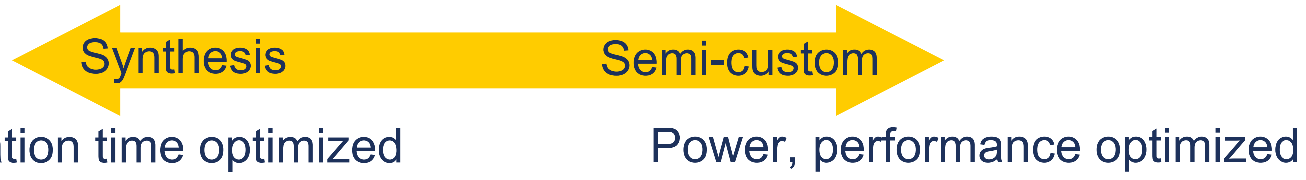
Effects on Cortex-A8 Design Flow

- Would like to use 100% synthesis and place route
  - Ideal for reusability
  - Allows fast spins of design during development
- Couldn't cover all corners of the Cortex-A8 performance-efficiency envelope
- Plan: deliver performance, power and area, targets while minimizing the additional effort required from the silicon partner to get their design to market
- Conclusion: Cortex-A8 processor uses:

  As much synthesis as possible in all applications.

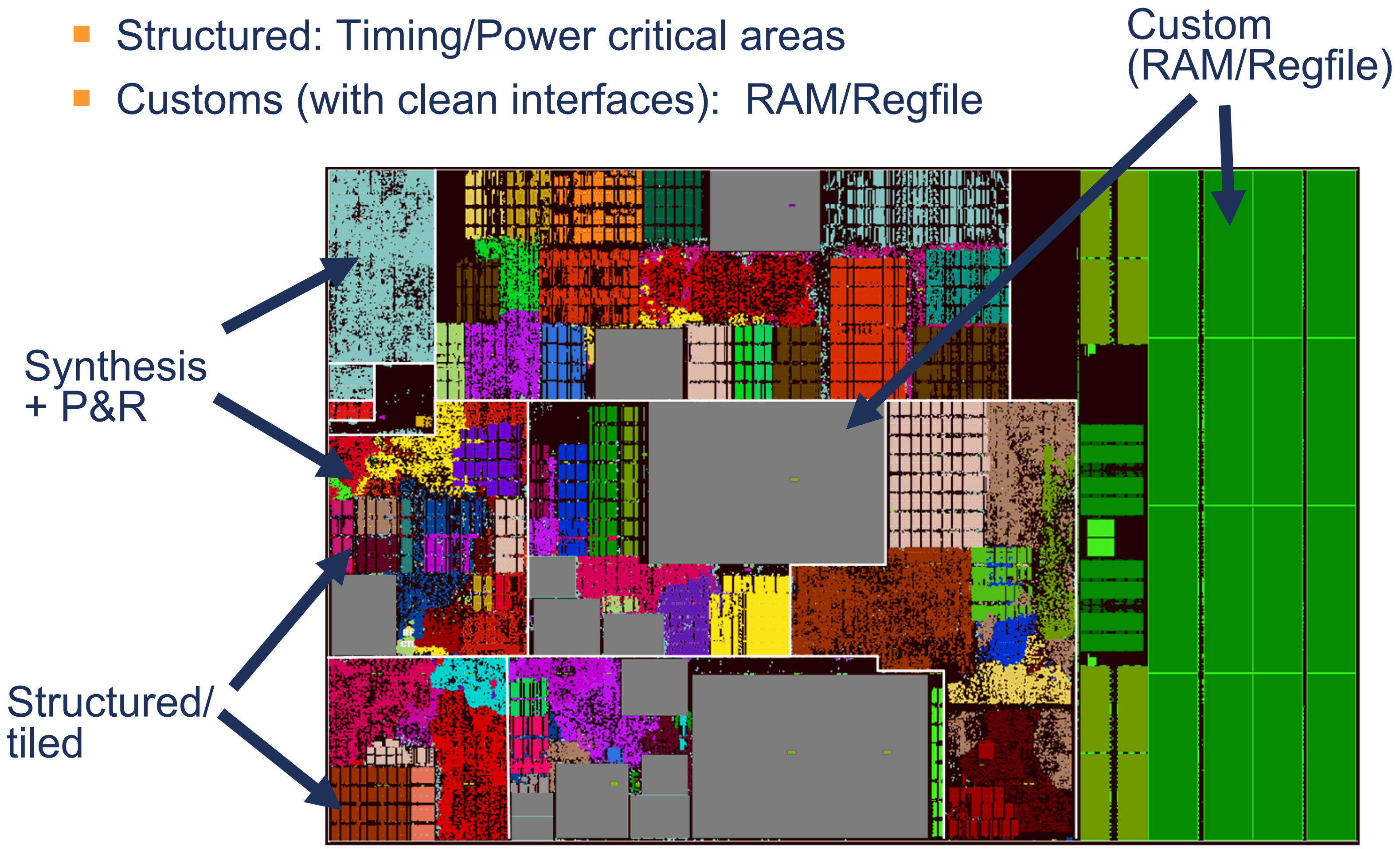
  100% synthesis in appropriate applications.





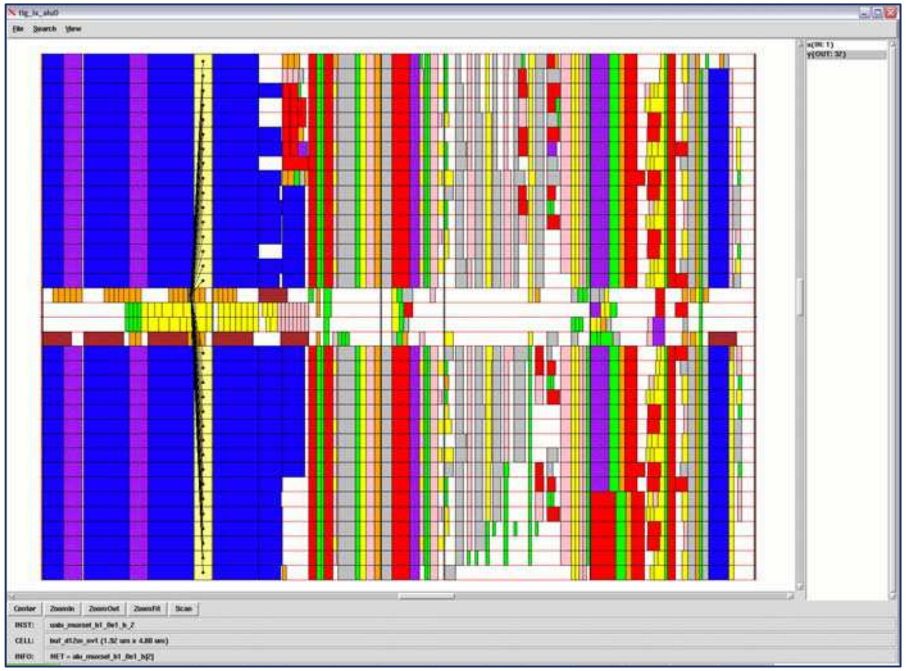
Implementation Regions

- The Cortex-A8 processor is partitioned at the microarchitecture level into:
  - Synthesis: Non-critical areas
  - Structured: Timing/Power critical areas
  - Customs (with clean interfaces): RAM/Regfile



Structured Regions

- "Structured" means:
  - Synthesizable RTL
  - Hand-mapped
  - Hand placed
  - Auto routed
- Best for regular datapath structures
- Critical routes:
  - Minimum length
  - Regular, predictable & repeatable
- Allowed safe use of restricted cells
- Relative placement was captured, not absolute coordinates
  - As physical factors become more important, EDA tools may need to evolve better ways to preserve the valuable IP content of designed placement



INST: uabs_muxsel_b1_0e1_b_2

CELL: buf_d12m_ovt (1.92 um x 4.80 um)

INFO: NET = alu_muxsel_b1_0e1_b[2]

Results for Structured Implementation

Results of detailed comparison of structured vs synthesis for one unit:

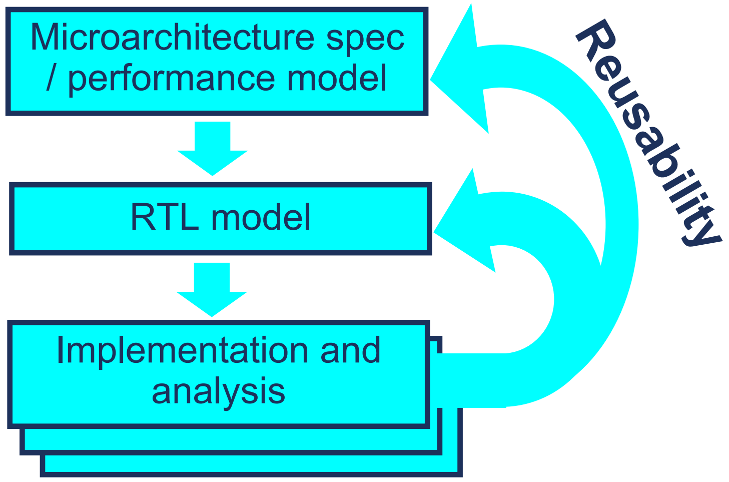
Structured vs. Synthesis	**Structured Improvement**
Cycle time (nvt only)	18%
Cycle time (mixed vt)	8%*
Area	-5%
Dynamic Power	6%
Static Power	53%
Cell Count	2%

Gains most apparent only after RTL optimized from synthesis feedback

*Synthesized had 27% LVT cells. Structured 0.7%

Developing a Reusable Microarchitecture

- IP partnership business model helps
  - Cortex-A8 microarchitecture got feedback from multiple implementation teams
- Reuse problems (mostly) dropped off quickly with additional implementations



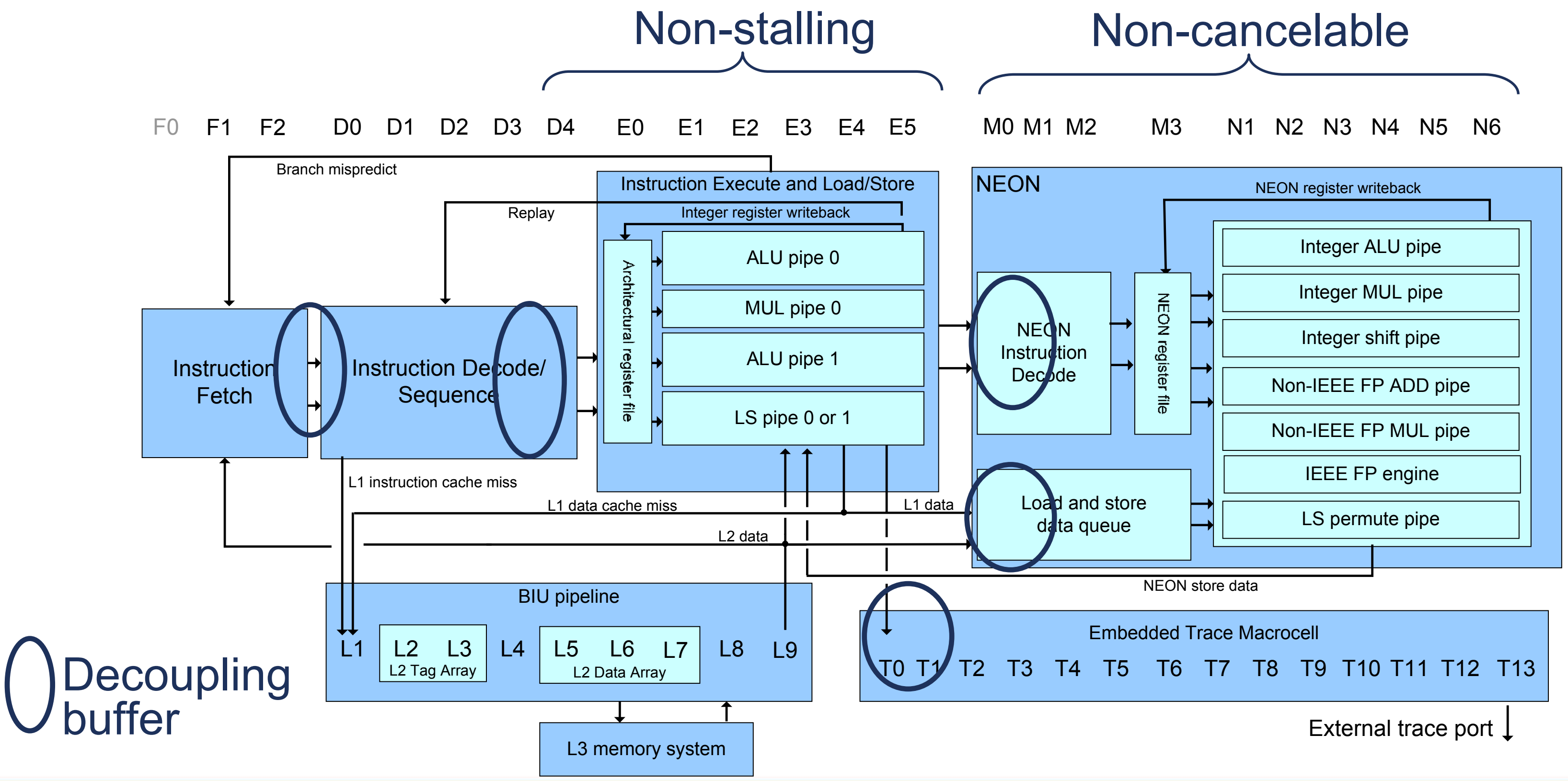
Microarchitecture spec / performance model
RTL model
Implementation and analysis
Reusability

- Timing and power problems fixed in microarchitecture/RTL
  - Performance modeling and microarchitecture/RTL teams stayed 100% assigned to project though tapeout
  - Re-pipelining or restructuring to fix a path for one implementation often helped the others (at least for power or area)

Effects of Reusability on Microarchitecture

Global, high-fanout stall signals were eliminated. Examples:

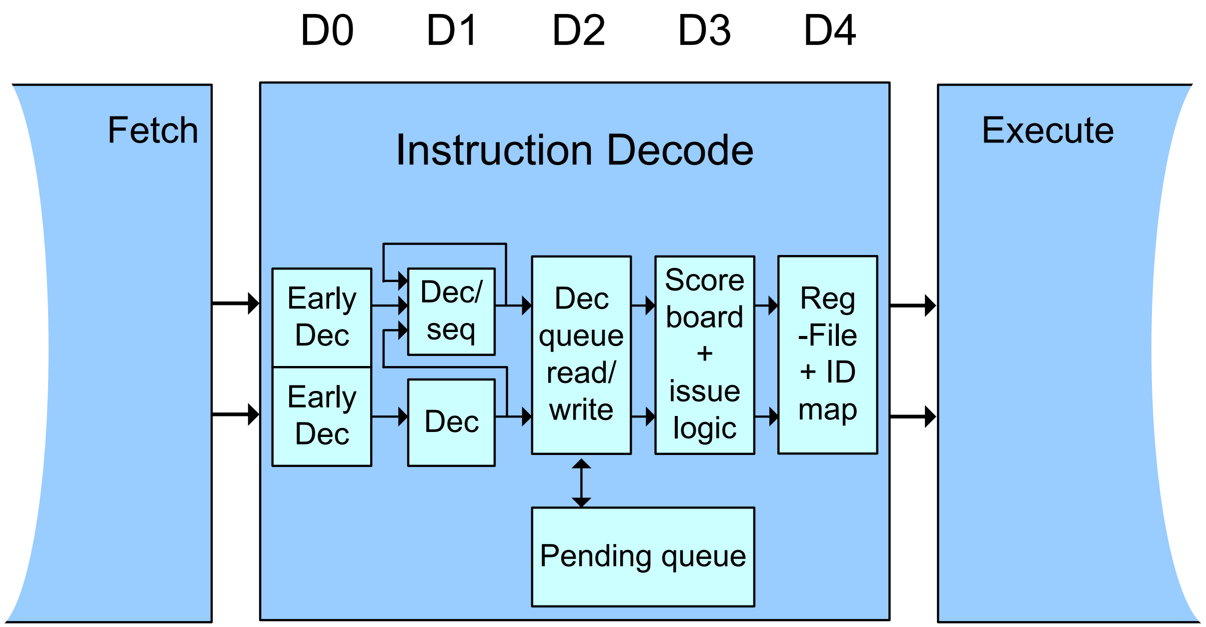
- Data and instruction decoupling queues exist at critical points in pipeline
- Non-stalling execute stages after D4
- Neon instructions cannot be cancelled or flushed after E5



Effects of Reusability on Microarchitecture

Non-critical areas pipelined for maximum synthesis+P&R even for high frequency implementations. Examples:

- Relatively relaxed pipelining of decode/sequencer stages:
  - Cost: 0.39% performance (made up in other areas)
  - Gain: Large chunk of random logic is very reusable
- ETM (Embedded Trace Macrocell™)

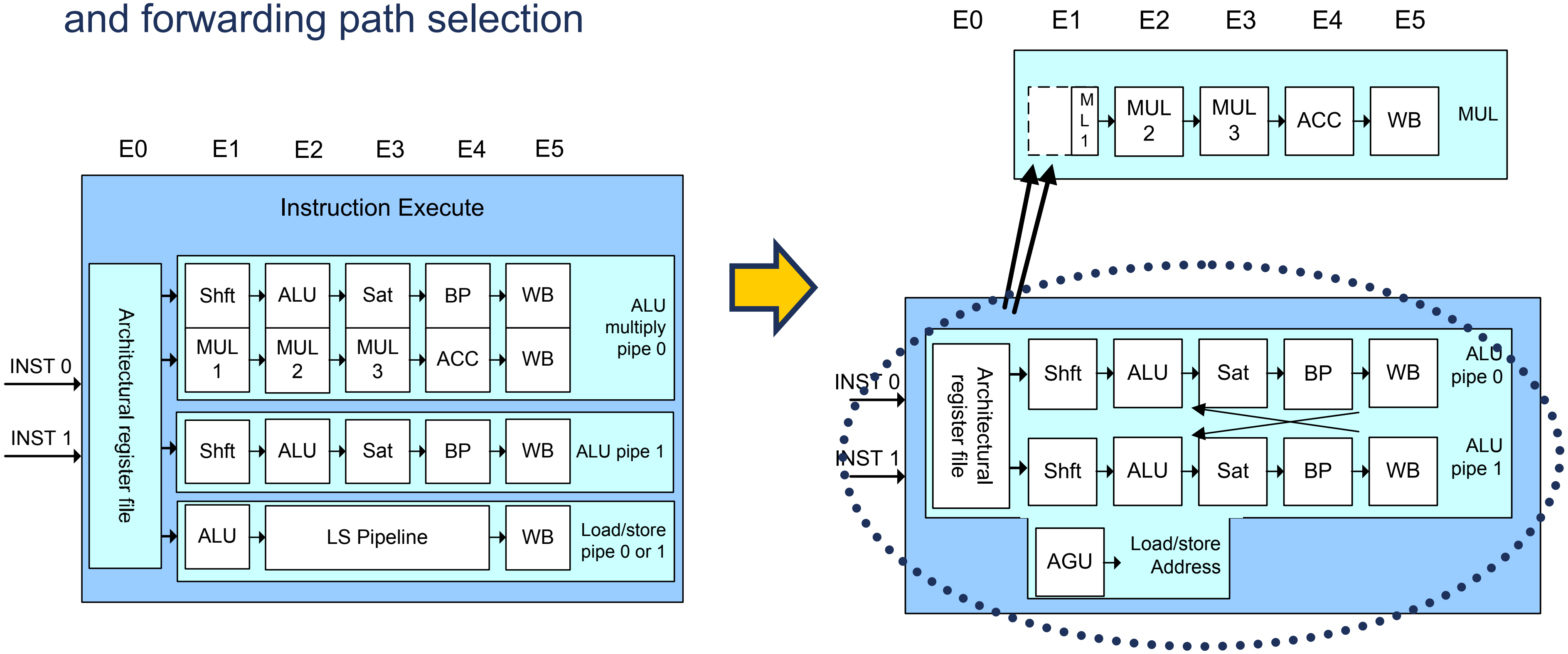


Effects of Reusability on Microarchitecture

Critical areas *logically* clustered to allow effective *physical* clustering:

- Execute pipelines and forwarding clustered

  (Sparse first stage of multiplier allows removal from critical area)
- Scoreboard pulls together all pipeline hazard tracking and detection and forwarding path selection



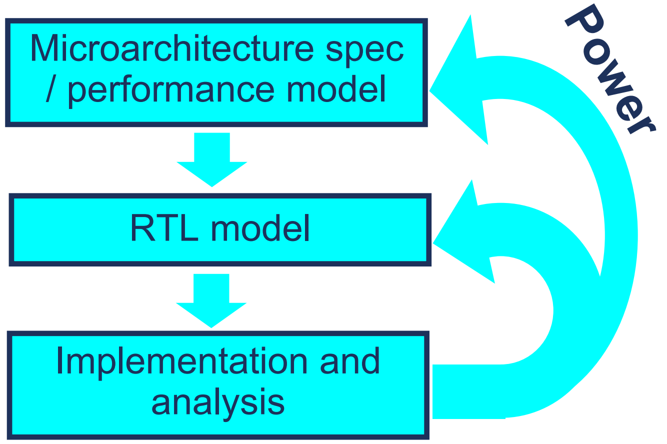
Energy Efficient + Reusable Design

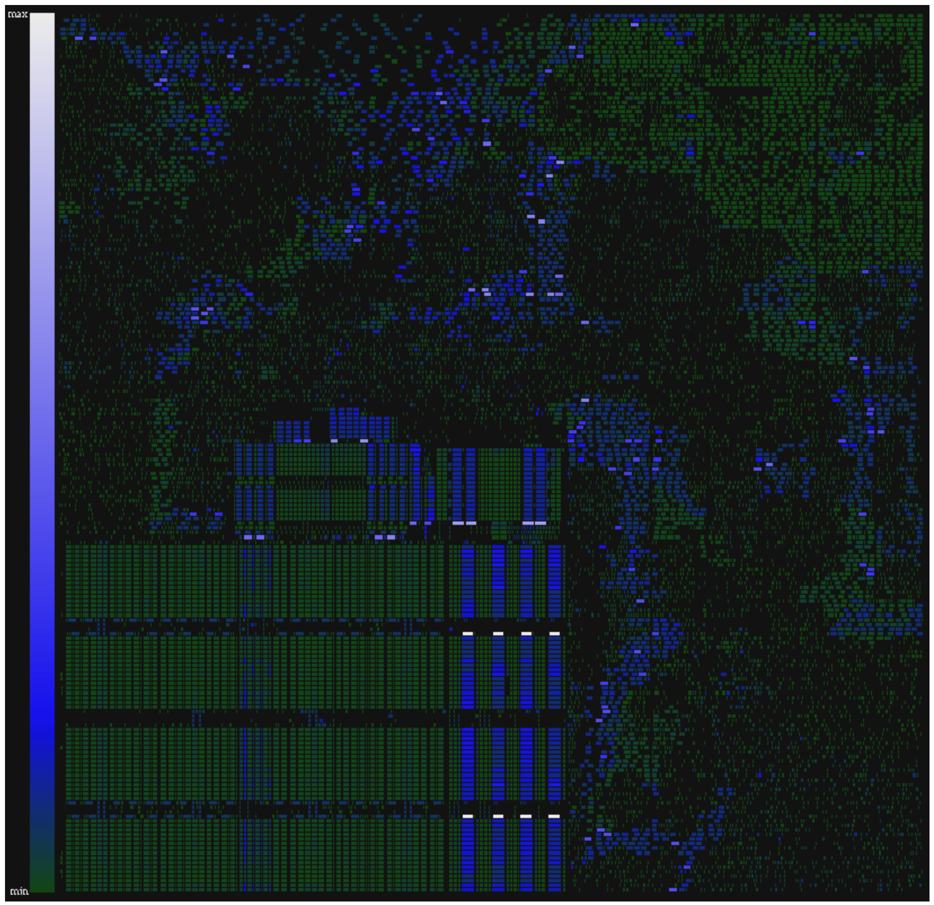
Most reusable power optimizations are in the microarchitecture or RTL:

- **Microarchitecture minimized complexity: in-order, statically scheduled**
  - Simplest microarchitecture that delivered the required performance
  - Simplicity helped reuse and power
  - Minimized reliance on: CAMs & dynamic logic
    i.e. minimized associative searches and wide gates
  - Minimized speculation
    - Reduced energy wasted producing unused results
    - Many power optimizations arose from predictability of pipeline
- **Clock gating**
  - 3 levels: Architectural, Regional (optional) and Local
  - >94% flip-flops locally gated
- **Minimized frequency of accesses to wide or deep structures**
- **Optimizing common cases to use least power**

Power Optimization in Design Flow

- Cortex-A8 design flow placed power in inner-loop of design flow
- Allowed tradeoffs with frequency/ IPC/area/reusability
- Both dynamic and static power analyzed
- Multiple vectors were required to cover interesting cases
- Unexpected activity & hot spots fed back
- Quiet vectors (idle/interlocked cases) especially useful in spotting power bugs

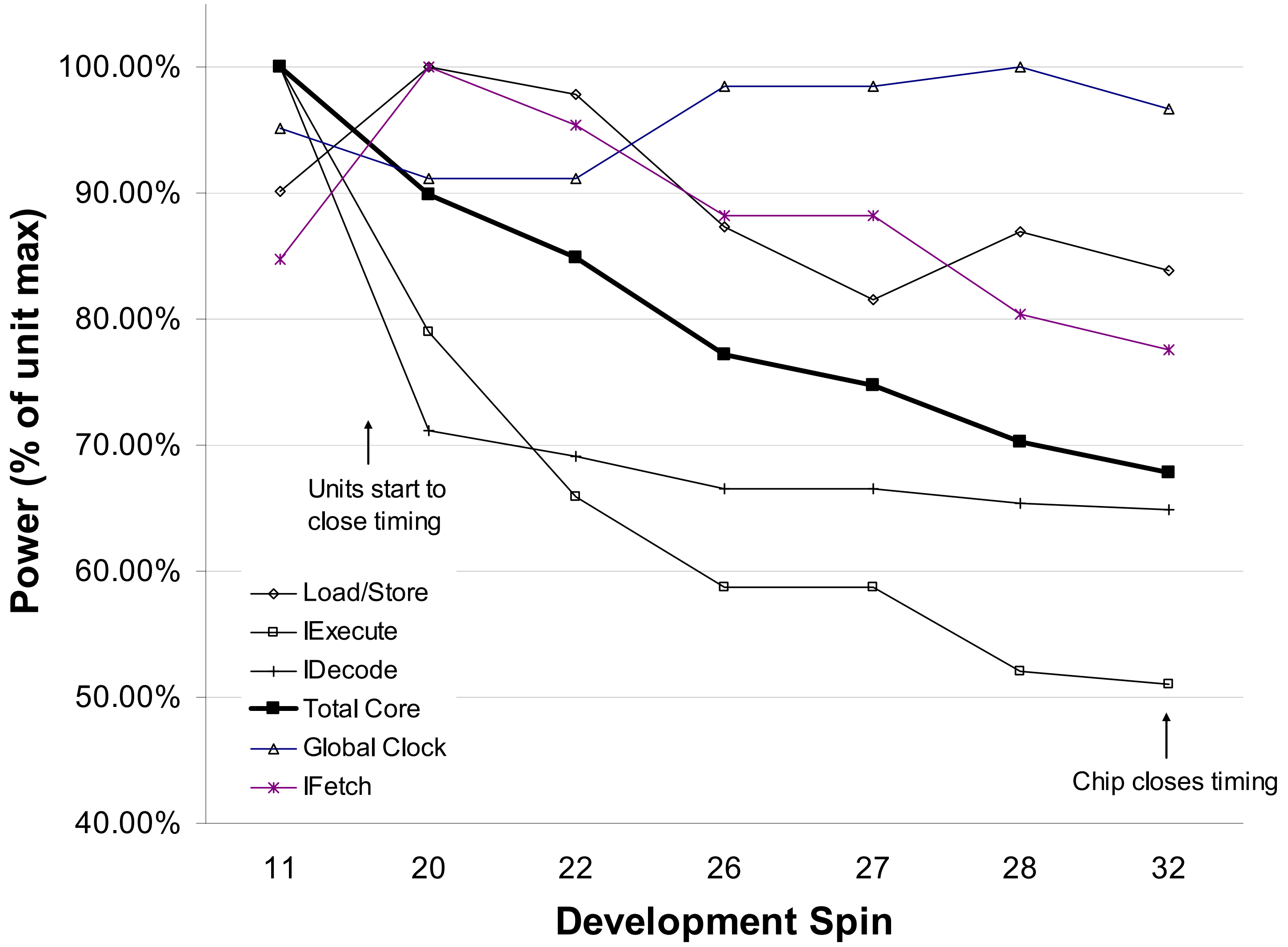

Microarchitecture spec / performance model
RTL model
Implementation and analysis
Power



Dynamic power feedback plot
Key: white (max) down to black (min)

Power Closure

Significant power savings came from the sum of many small improvements:



Cortex-A8 Processor Summary

Goal of 2x performance of previous generation

- Achieved at 800MHz
- Exceeded at 1GHz
- Across 150+ ARM and industry benchmarks including EEMBC, SpecInt95, Mediabench, and partner provided applications

For mobile applications

- >600Mhz or 1200 DMIPS at <300mW
- Low-power 65nm technologies

For tethered applications

- >1GHz or 2000DMIPS
- 90nm and 65nm technologies

6 licensees and counting

- $^1/_3$ of the Top 15 WW Semiconductor Vendors