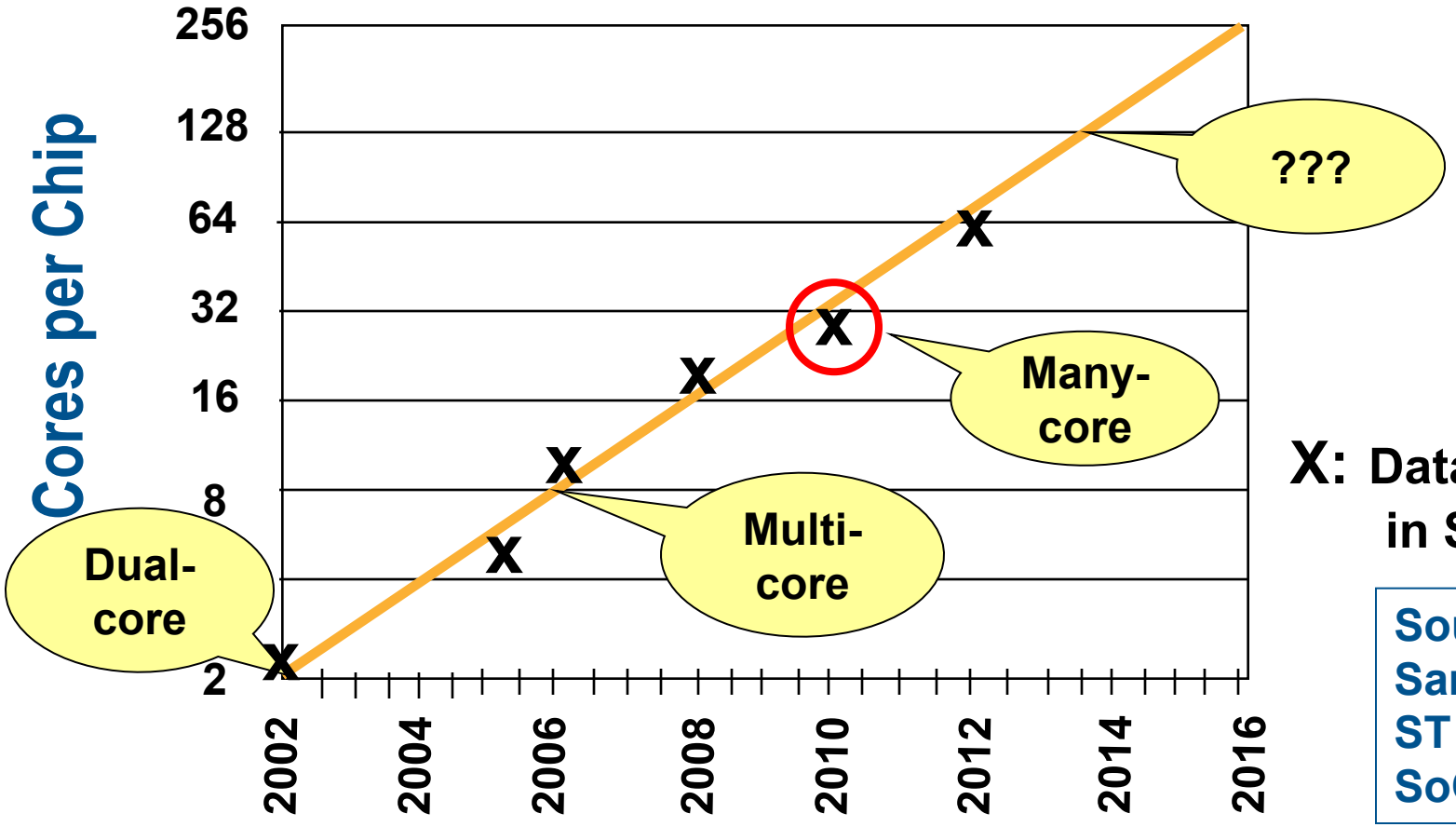
Exploring H/W and S/W solutions to MP-SoC platform mapping: An Industrial Perspective

Pierre Paulin
Director, SoC Platform Automation
STMicroelectronics (Canada) Inc.

Multi-Processor System-on-Chip Symposium
7 July 2011, Beaune, France

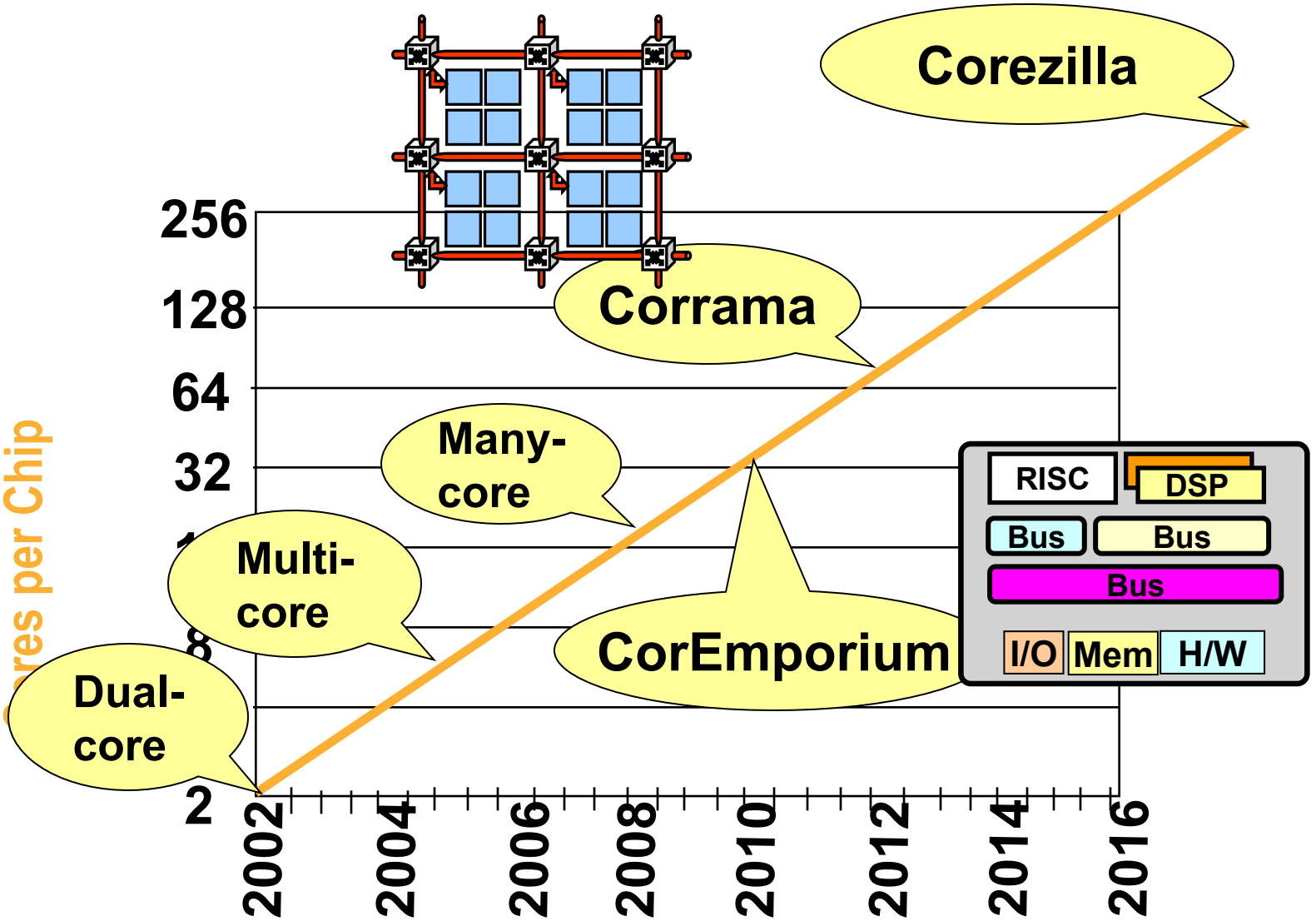
Core's Law (for Embedded SoCs)



Source: Sample ST consumer SoCs

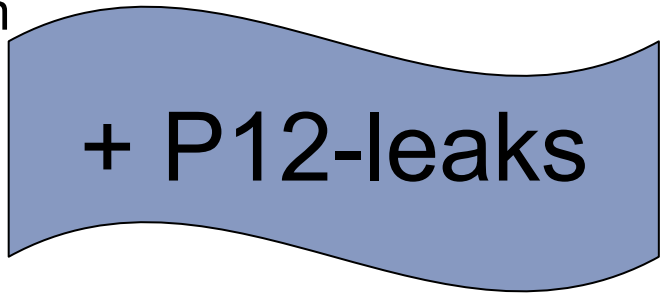
- # Embedded cores in SoCs doubles every ~2 years
- Total SoC area very stable across tech nodes

Core's Law: Zillion Core Chip

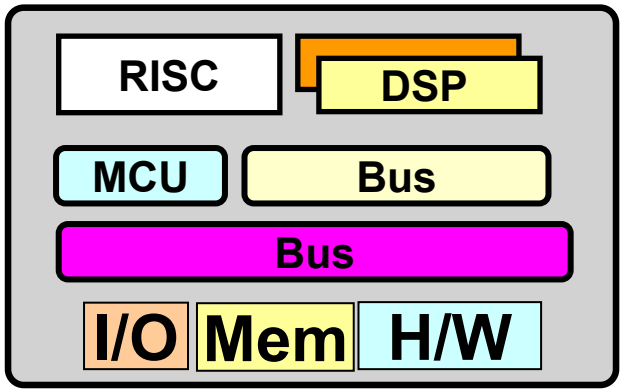


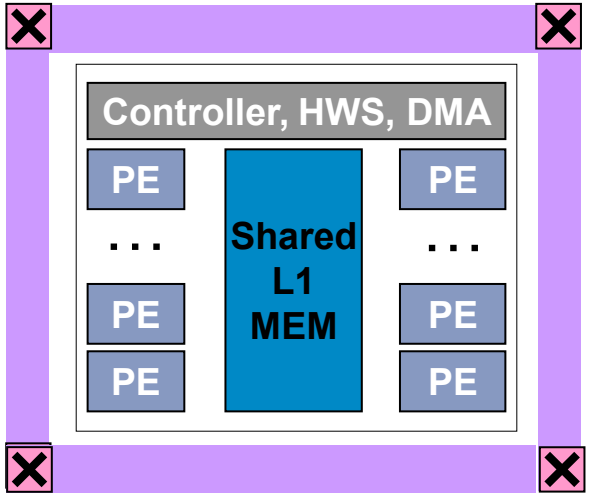
Outline

- Platform 2012 Multicore Fabric
- Platform 2012 Programming Environment
  - Programming models
  - Programming tools
- Case Studies, Lessons Learned
  - High-Quality Rescaling application
    - Mapped to S/W dominated platform
    - Mapped to H/W dominated platform



From CISP to RISP





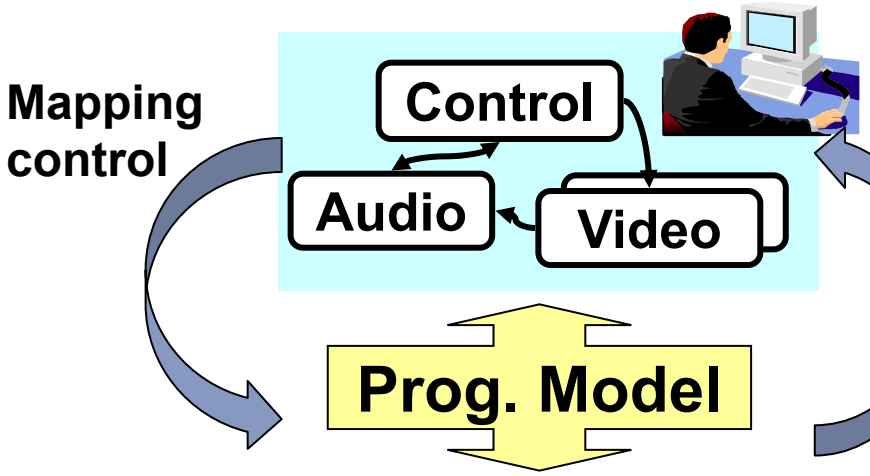
'CISP'

(Complex Integrated System Platform)

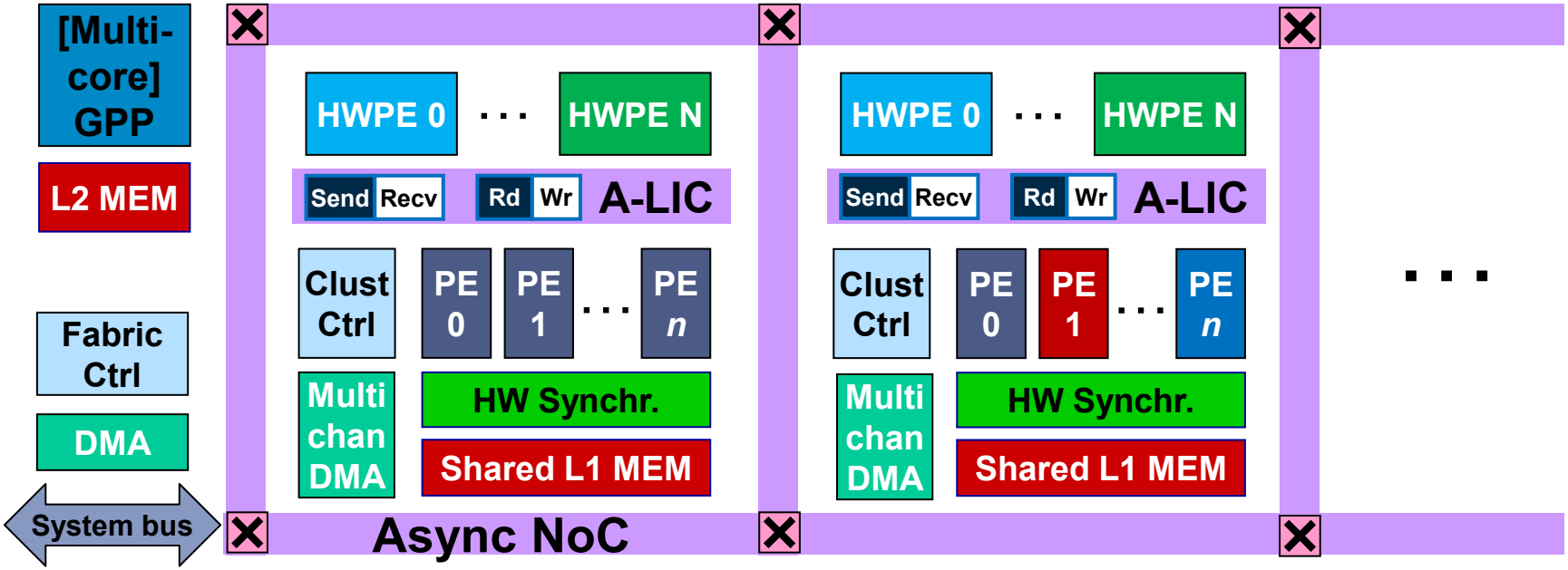
'RISP'

(Regular Integrated Sytem Platform)

Platform 2012 Overview



- 1-32 clusters
- 1-16 cores/cluster
- ~25 GOPs to several TOPS
- Optional H/W PEs



Outline

- Platform 2012 Multicore Fabric
- ***Platform 2012 Programming Environment***
  - Programming models
  - Platform mapping tools
- Case Studies, Lessons Learned
  - Video High-Quality Rescaling
    - Mapped to S/W platform
    - Mapped to H/W-S/W platform

Programming models

- Each group has favorite one!
  - Set-top box, modem: Synchr. dataflow w. simple control
  - Mobile multimedia: Dynamic Task Dispatch (DTD), OpenCL
  - Video algorithm developers: CUDA/OpenCL
  - ST R&D organizations:
    Components/patterns, GCD subset, Streamit, UML…
  - Management: “OpenAnything”
- Sum of all forces → favor PPMs that are
  - Industry ‘standards’ → OpenCL
  - C-based dialects
  - Those of the customer ☺ → Predicated Exec. Data Flow
  - Exploiting platform efficiently → Native Prog. Models

P2012 Software Development Kit

Programming Models

Standard

OpenCL

Streaming

PEDF
(Pred.Exec. Data Flow)

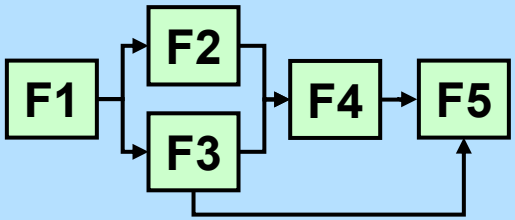
Native

Components
Dyn. Task Dispatch

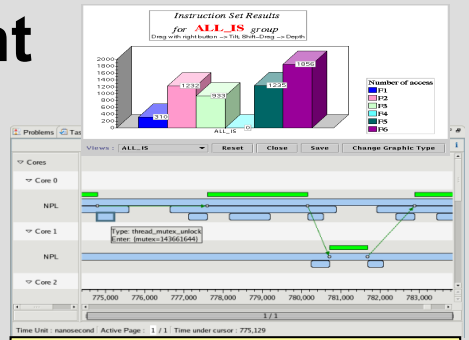
Programming Environment

Language-based

Component-based



C API-based



Debug, Analysis, Viz

System Infrastructure & Runtime

Component-Based Dynamic Deployment

QoS

Power Management

Execution Engines

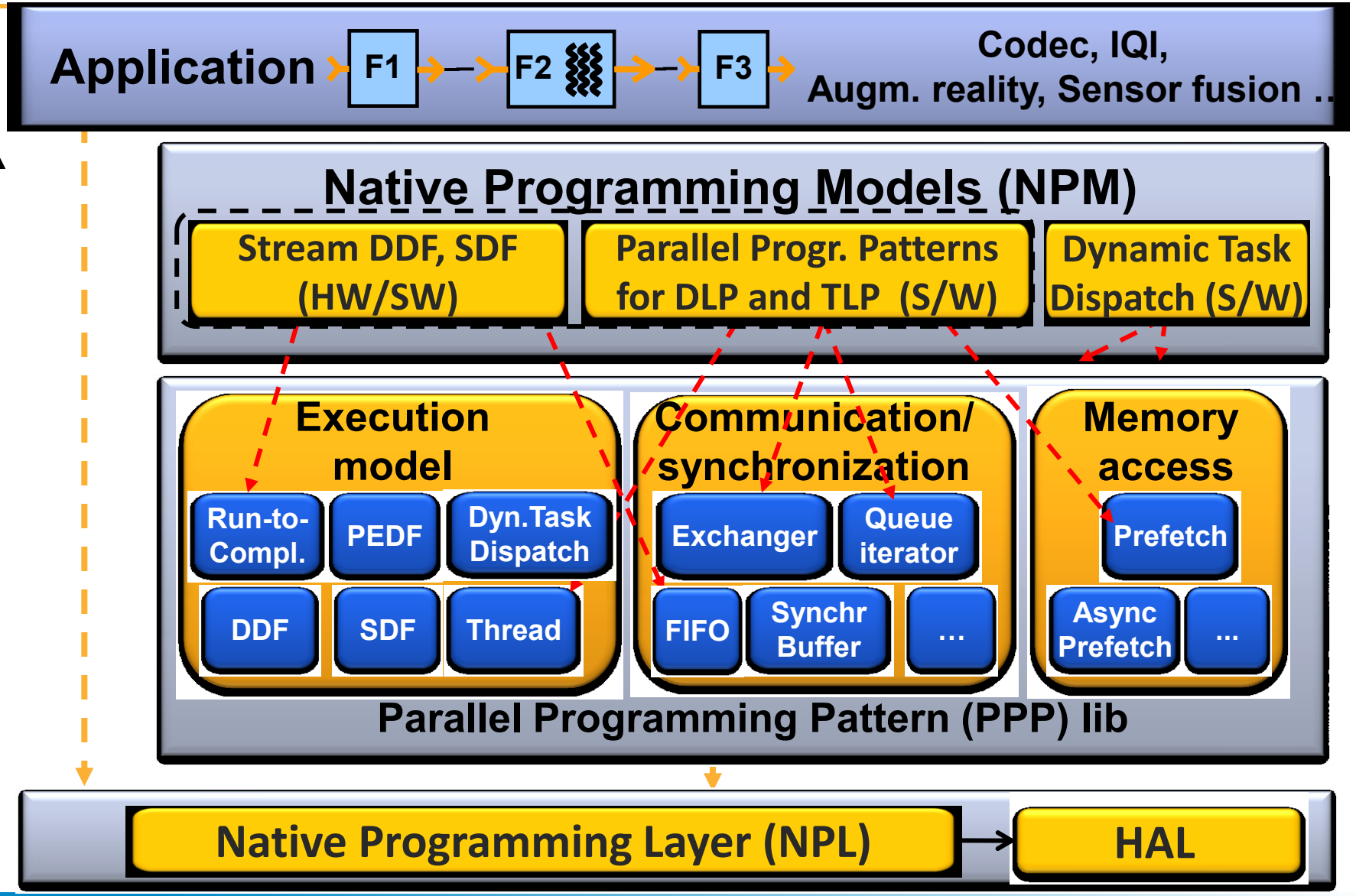
Platforms

Functional

SYSTEMC™TLM

XXL Emulator

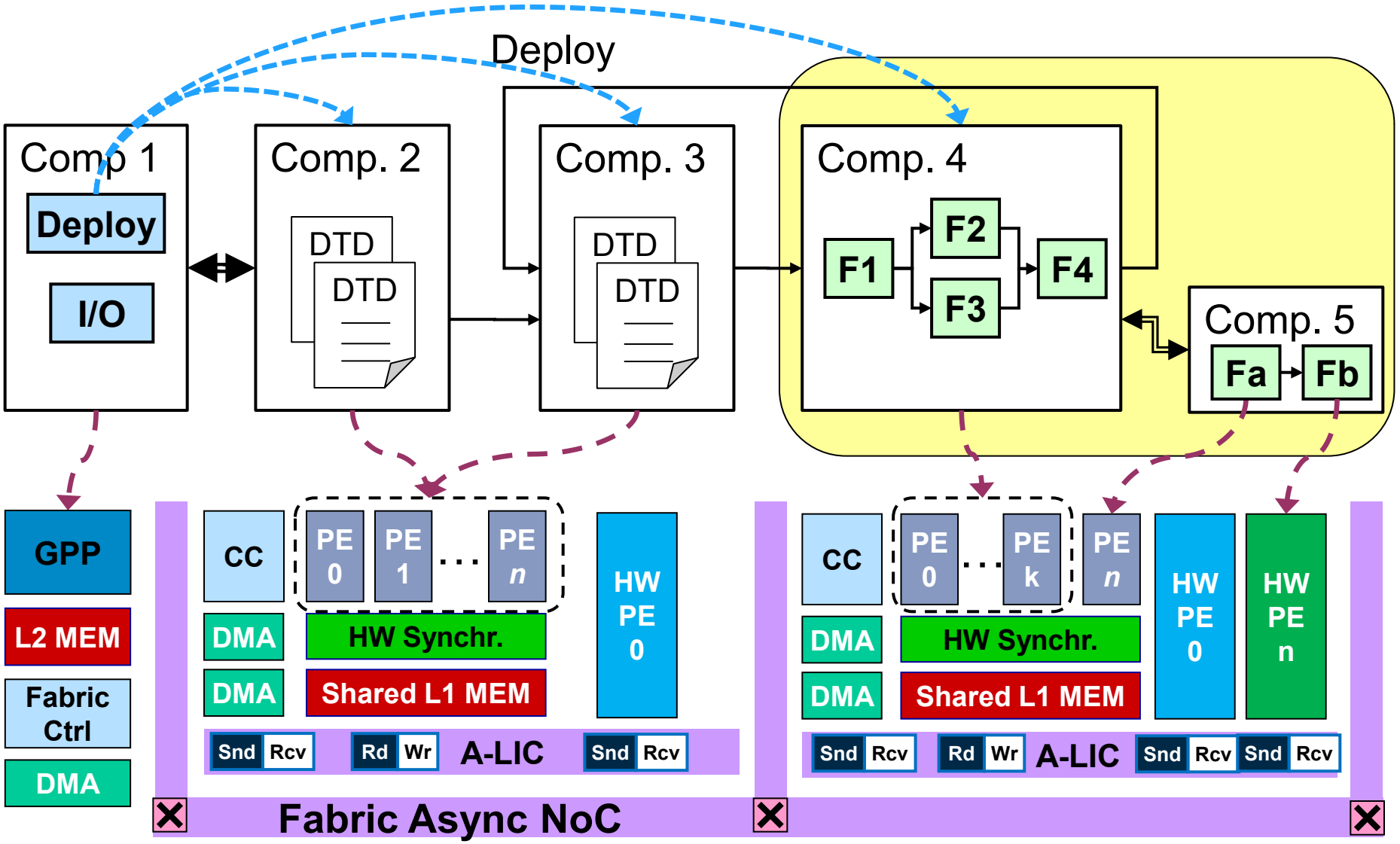
Parallel Programming Models



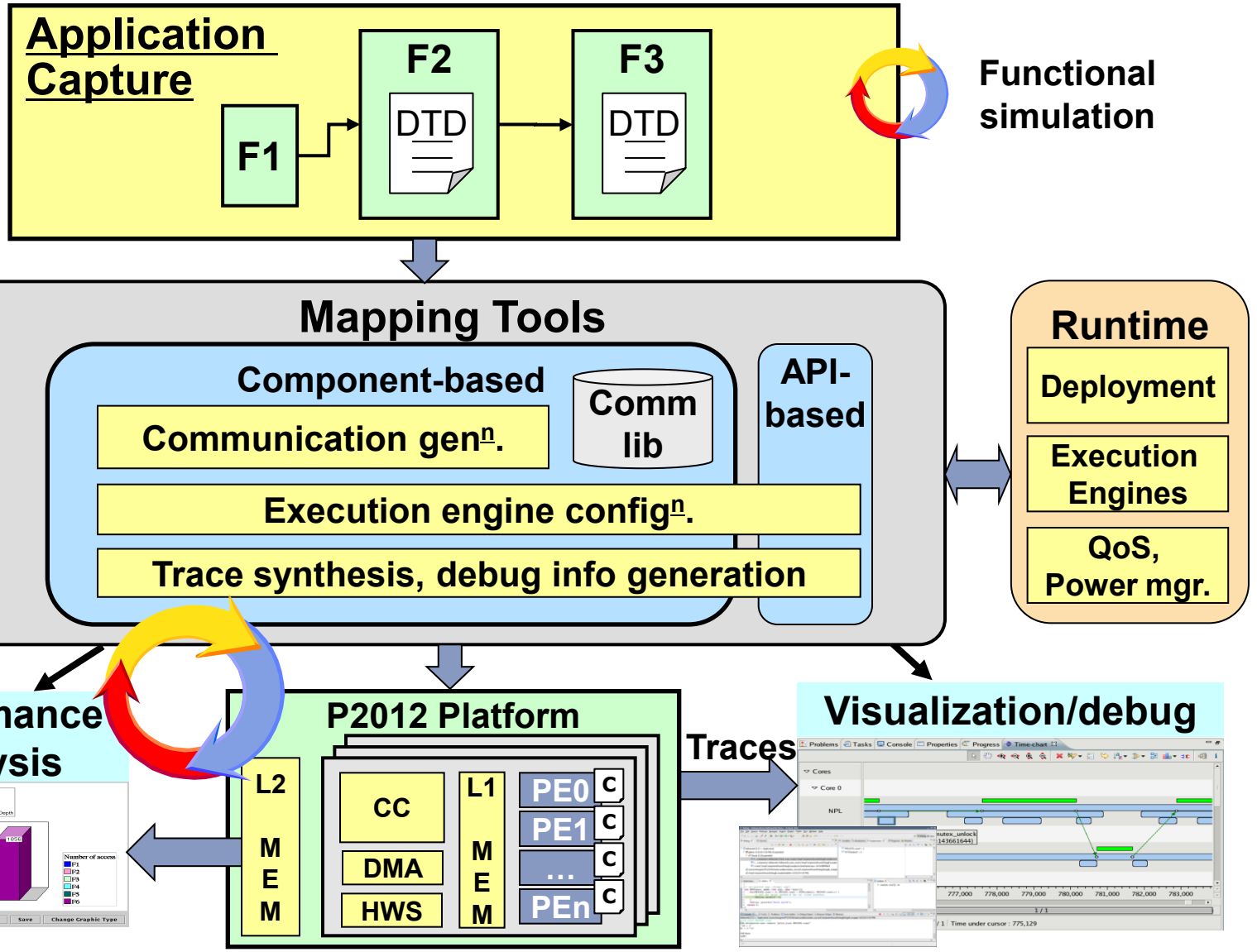
Outline

- Platform 2012 Multicore Fabric
- Platform 2012 Programming Environment
  - Programming models
  - ***Programming tools***
- Case Studies, Lessons Learned
  - VC1 video decoder example

Application-to-Platform Mapping



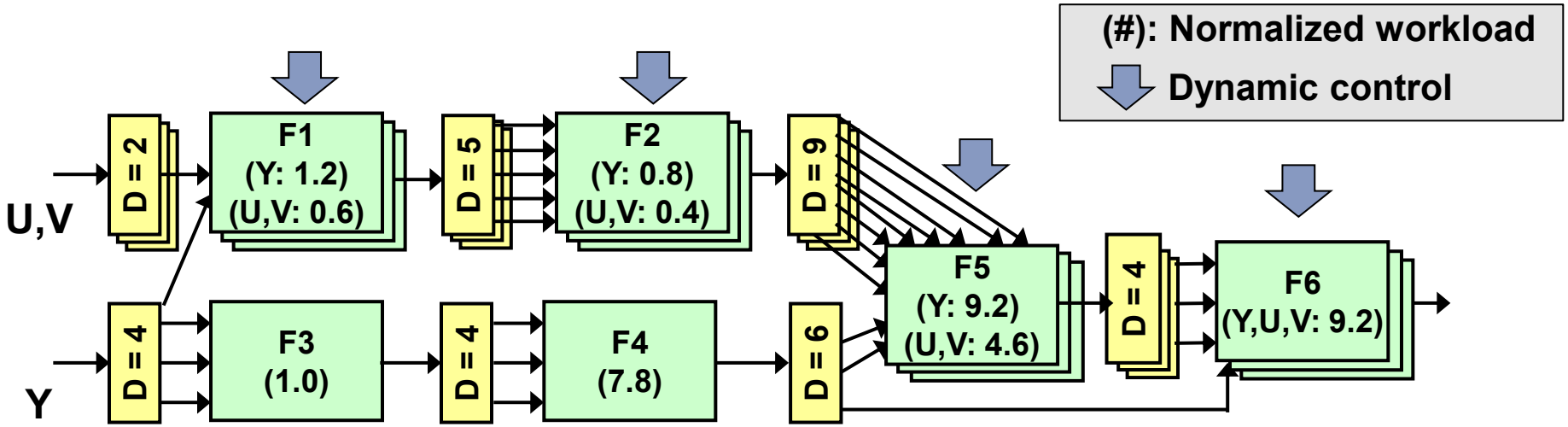
Native Programming Tools Flow



Outline

- Platform 2012 Multicore Fabric
- Platform 2012 Programming Environment
  - Programming models
  - Platform mapping tools
  - Programming model-aware debug and visualization
- ***Case Studies, Lessons Learned***
  - Video High-Quality Rescaling
    - Mapped to S/W-dominated platform
    - Mapped to H/W-dominated platform

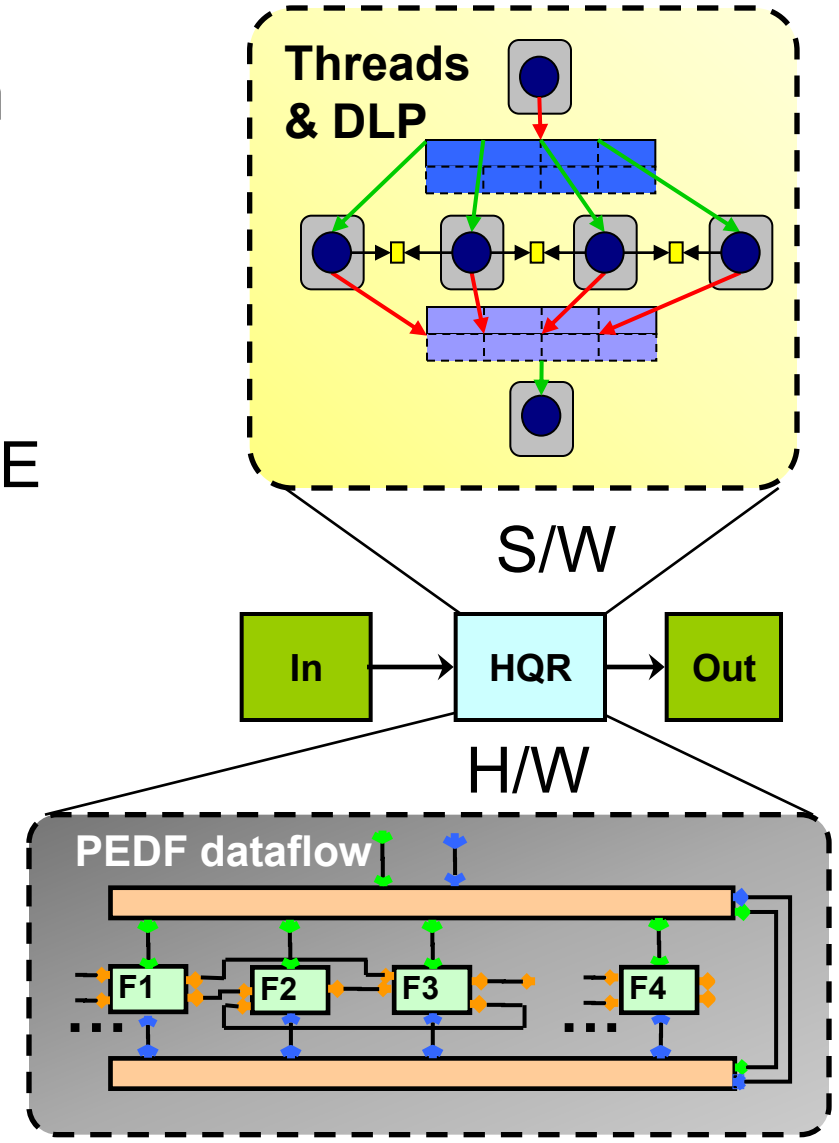
HQR (High-Quality Rescaling)



- HD 1080p, 60 fps
- SDF model variant
  - One “token” on in/out per link per filter firing
    - Or simple static multi-rate
  - Tokens typically a line of pixel data
  - Multiple modes (on frame-by-frame basis)
  - Some dynamic control flow, exceptions
    - E.g. dynamic bypass of a filter, frame edges

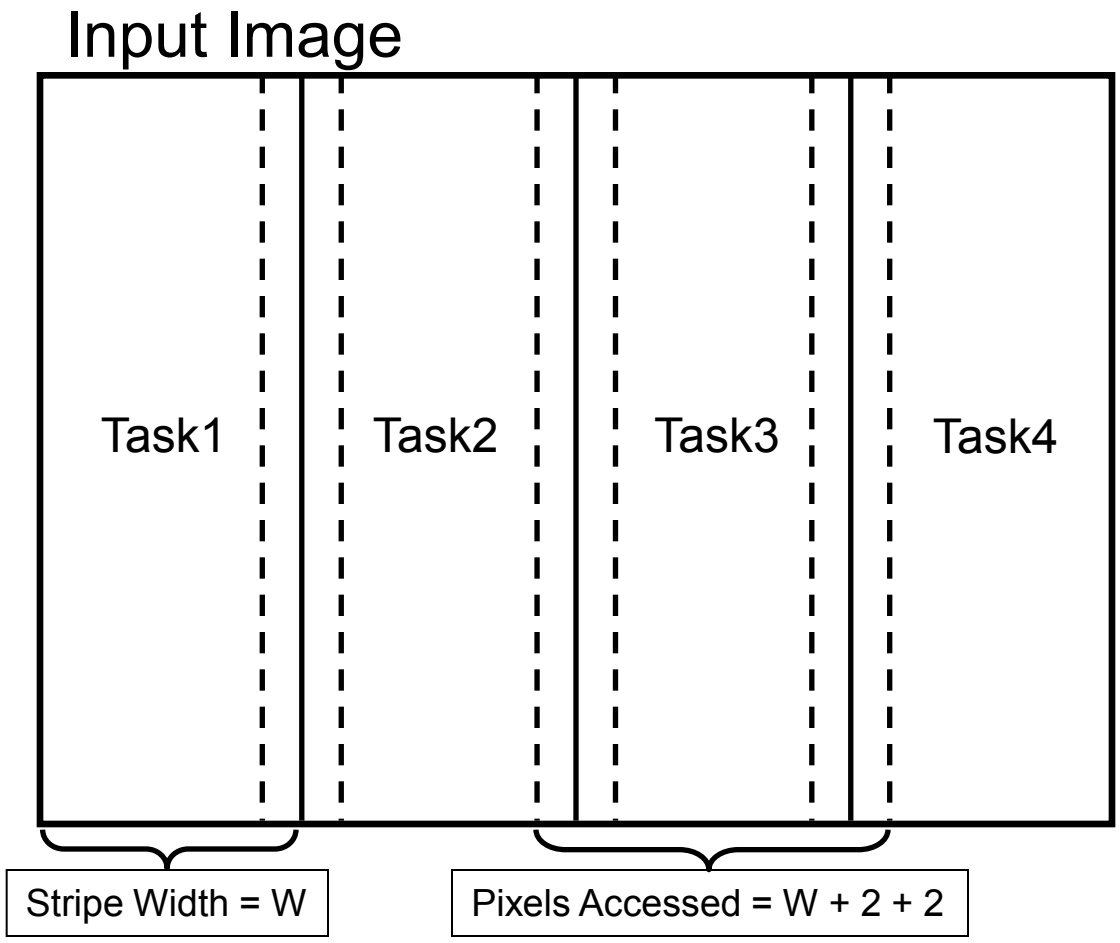
Two Mapping Approaches

- Map to S/W-based platform
  - Data-level parallelism
    - Structured programming patterns
    - Multi-processor & SIMD
  - All tasks for a given data element assigned to single PE
- Map to H/W-dominated platform
  - Task-level parallelism
    - Dataflow programming model
    - Software-based control
  - Tasks assigned to a single H/W Processing Unit
    - DLP inside each H/W PU



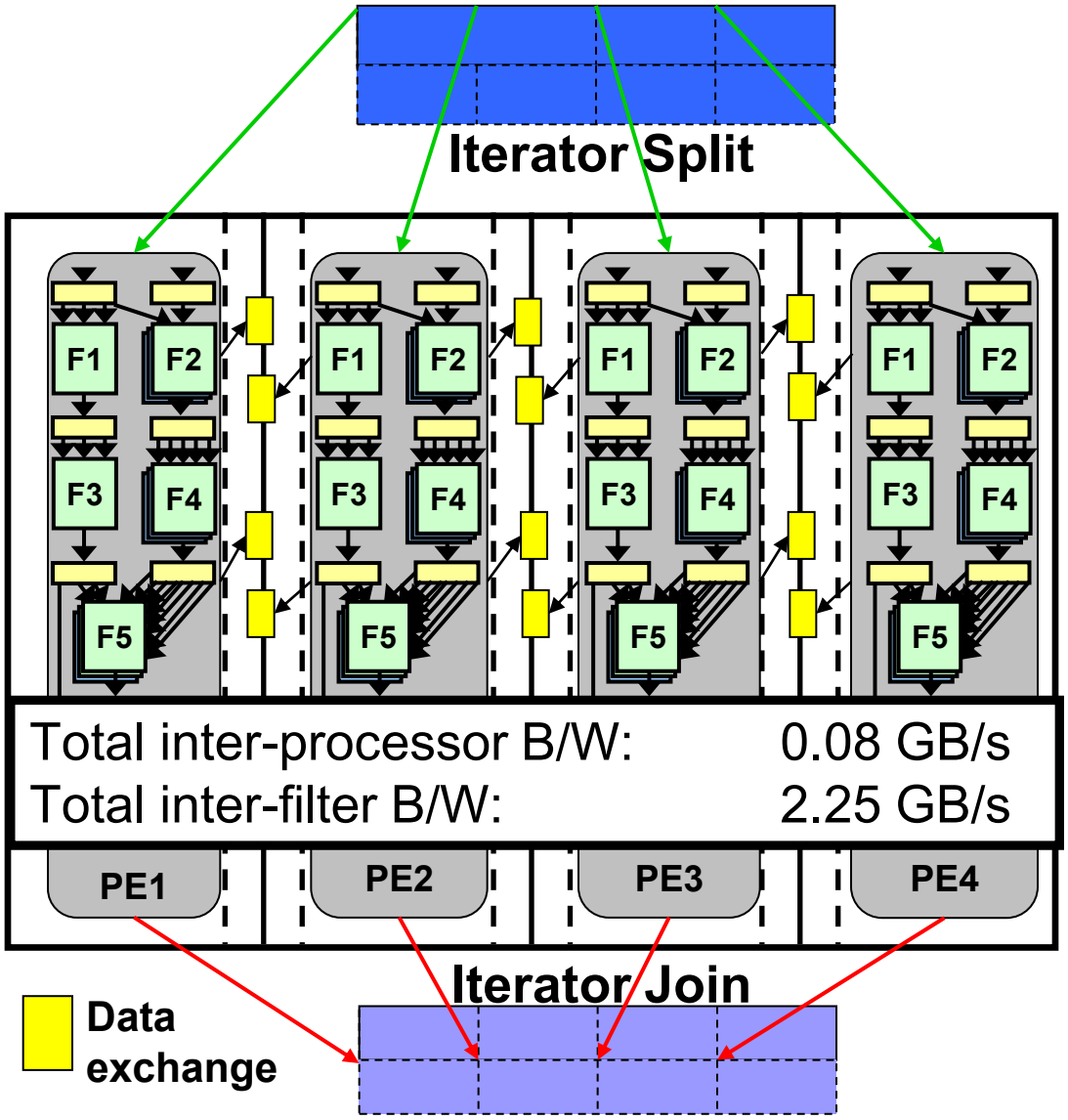
S/W Mapping: HQR example

- Data-level parallelism
  - Each image line split into stripes
  - Each PE runs all filters for a stripe
  - SIMD optimization of each filter
- Parallel Progr. Patterns
  - Data iterator split and join patterns
  - Synchronization between PEs using “exchanger” pattern (for border pixels)

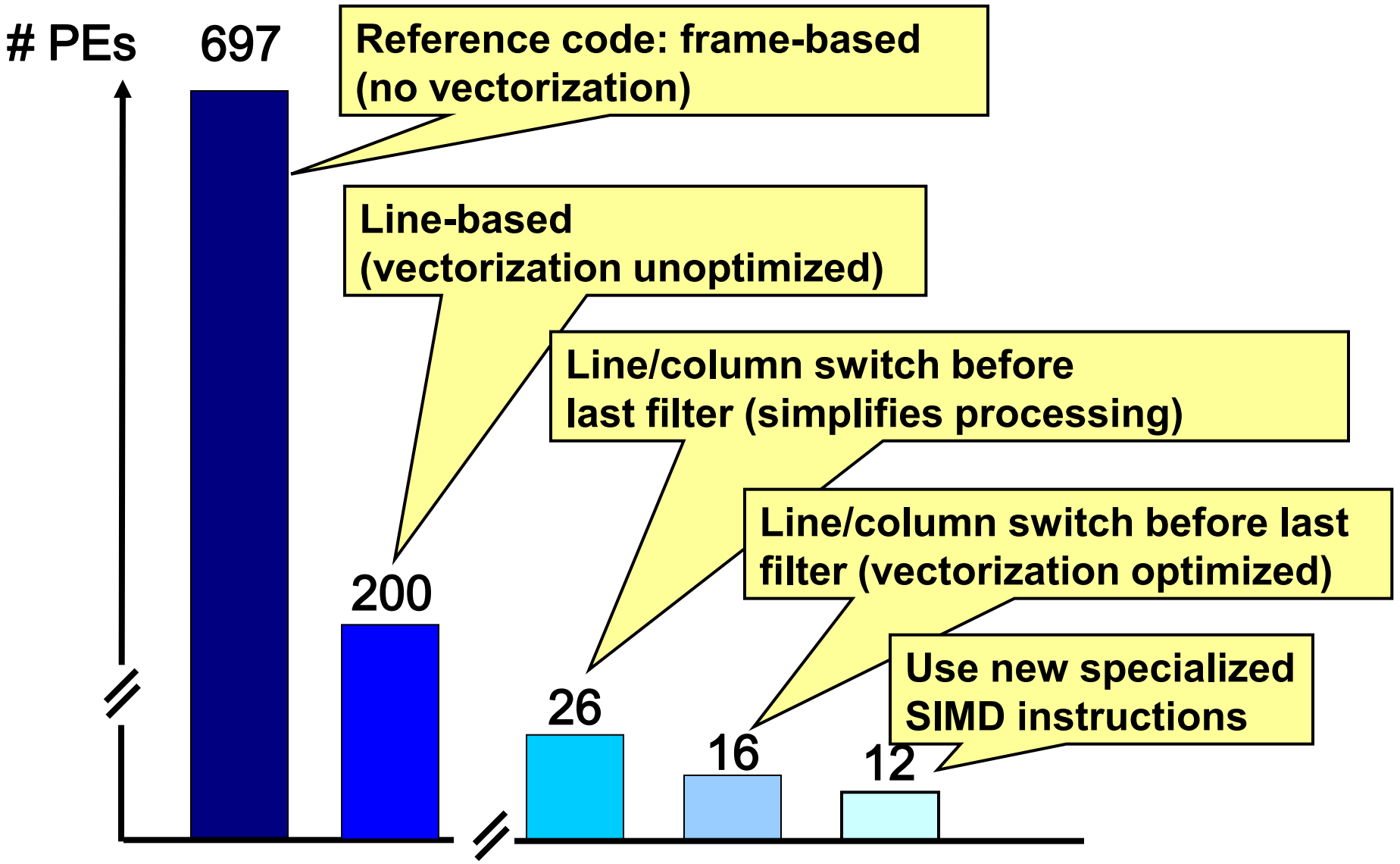


S/W Mapping: HQR example

- Data-level parallelism
  - Each image line split into stripes
  - Each PE runs all filters for a stripe
  - SIMD optimization of each filter
- Parallel Progr. Patterns
  - Data iterator split and join patterns
  - Synchronization between PEs using “exchanger” pattern (for border pixels)

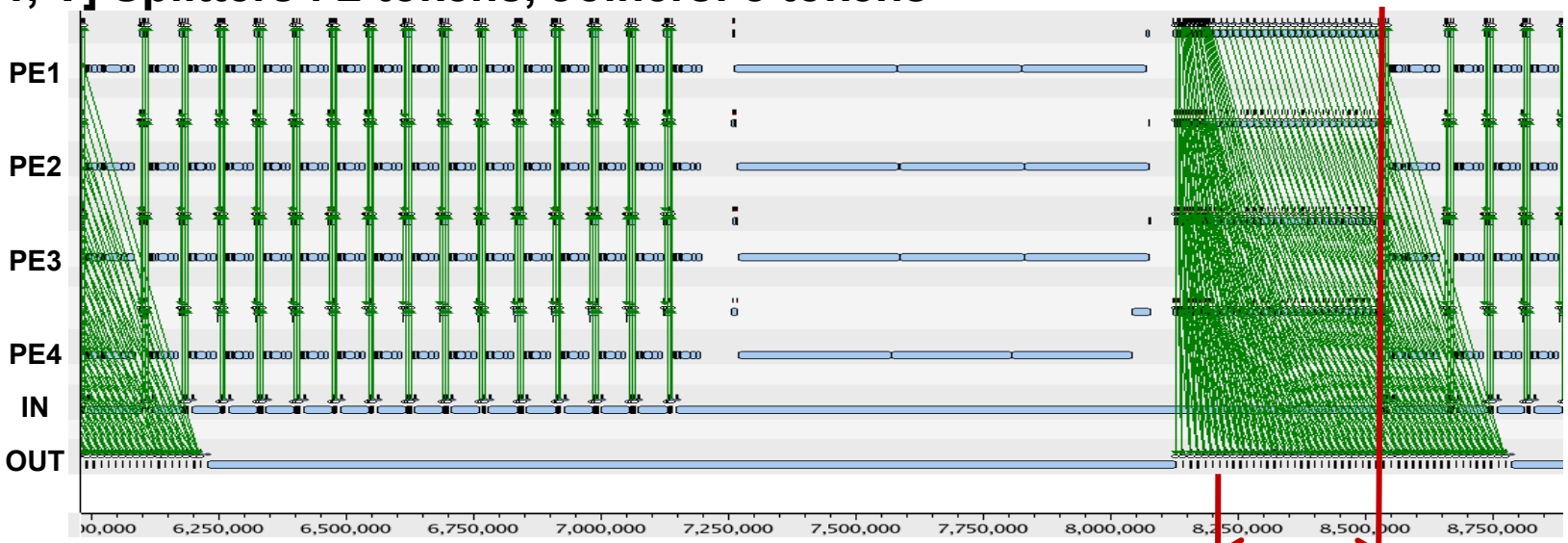


HQR Optimization Process

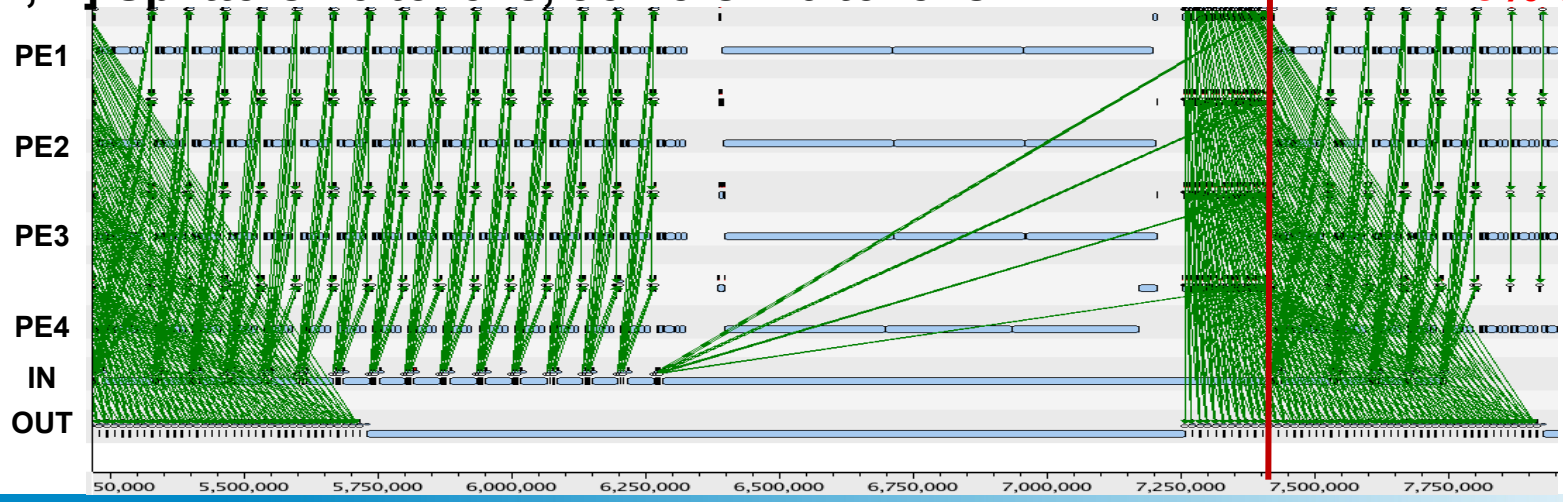


HQR Macro trace analysis
Exploring Queue Iterator Depths

[U, Y, V] Splitters : 2 tokens, Joiners: 8 tokens



[U, Y, V] Splitters : 3 tokens, Joiners: 16 tokens

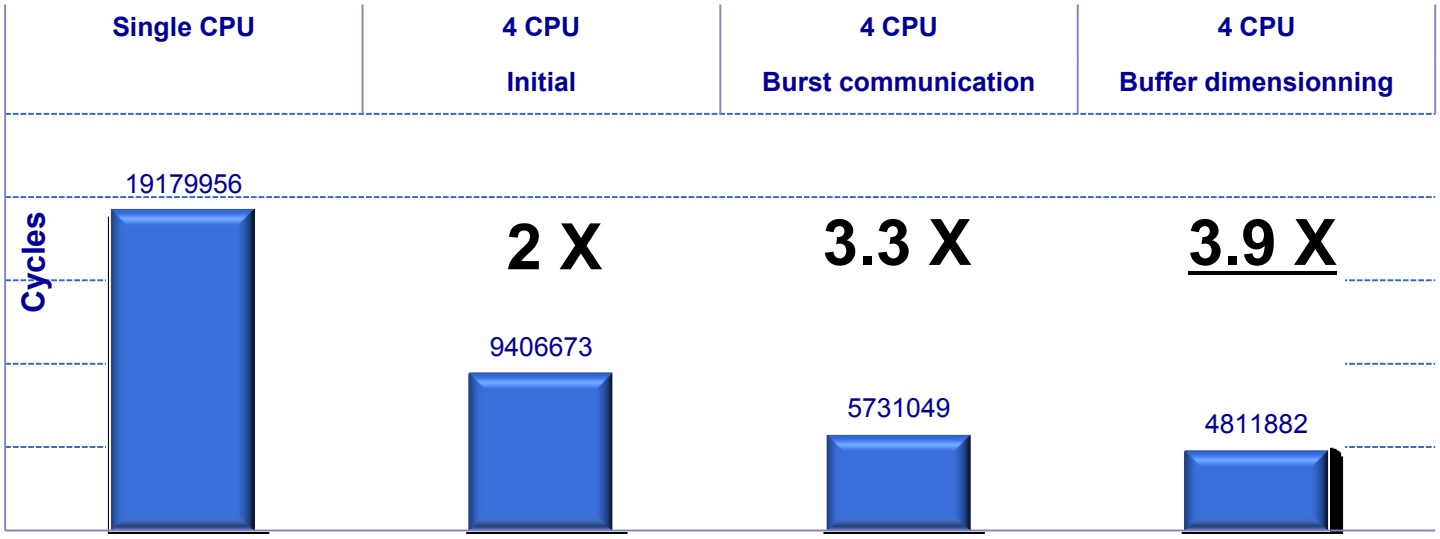


13% speedup

Prog. Tools: HQR Mapping Results

- **Vectorization results (16 way VECx EFU):**
  - **Results for standalone CA-ISS**
  - **Average vector unit utilization 79%**

- **Parallel processing results (1 vs. 4 PEs)**

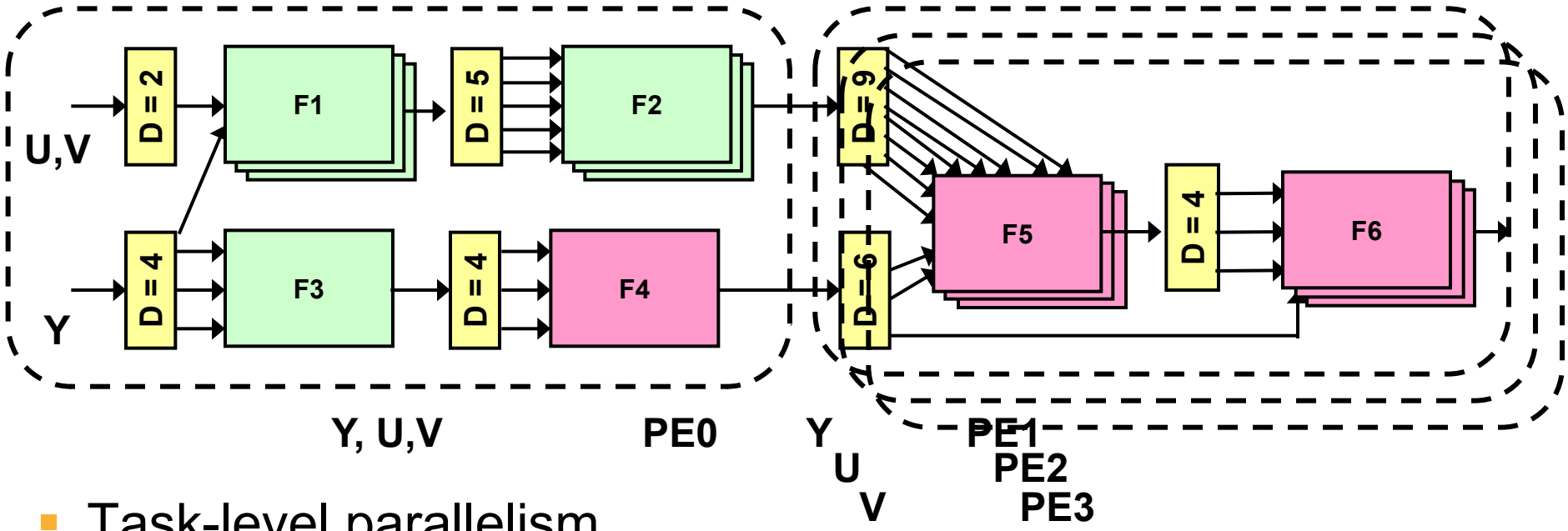


Results on cycle-approx TLM platform

S/W-based solutions

- P2012 Group
  - We reduced cost of S/W by over 50X
  - This is great!
- Customer
  - You increased cost over H/W by over 5X
  - This is a disaster!
- Hard lesson
  - Customer is always right - especially when it is true …
- Conclusion
  - Mixed HW/SW platforms for low-cost consumer
  - Pure SW platforms for mass market

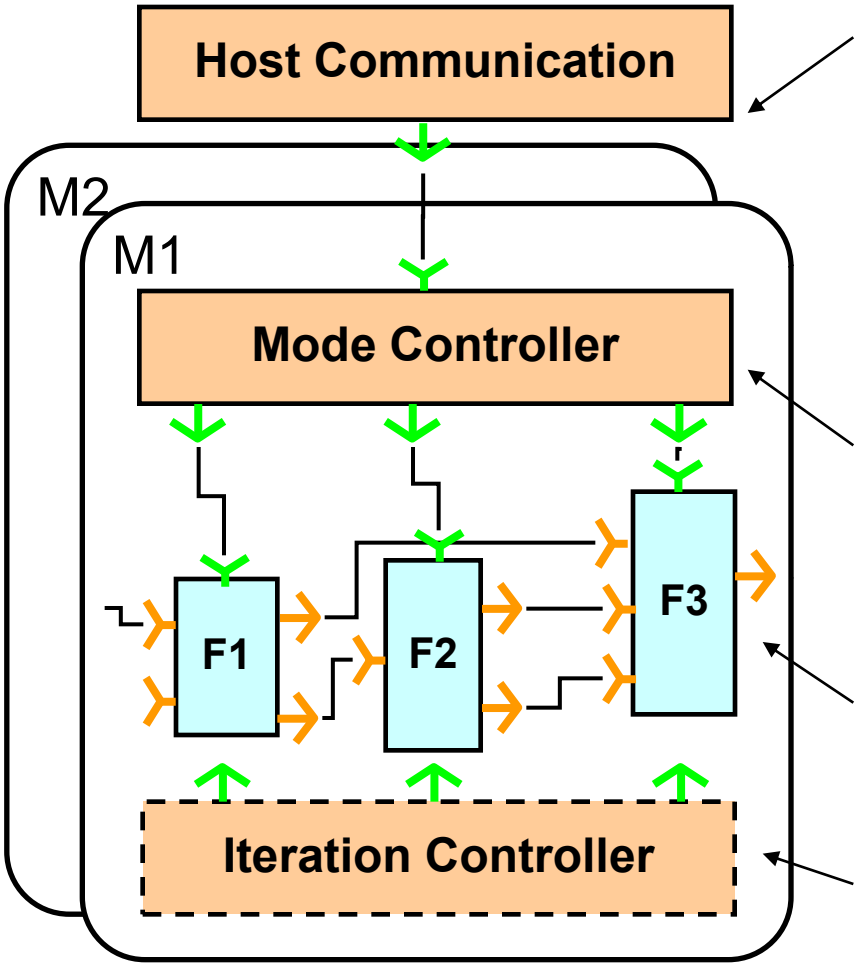
H/W Mapping: HQR Example



- Task-level parallelism
  - Assignment of each filter to a H/W PU
  - Grouping of highly communicating PUs to a single PE
- In contrast with S/W mapping, where
  - Data-level parallelism exploited (Multi-PE and SIMD)
  - Each PE performs all tasks

PEDF Dataflow Programming Model

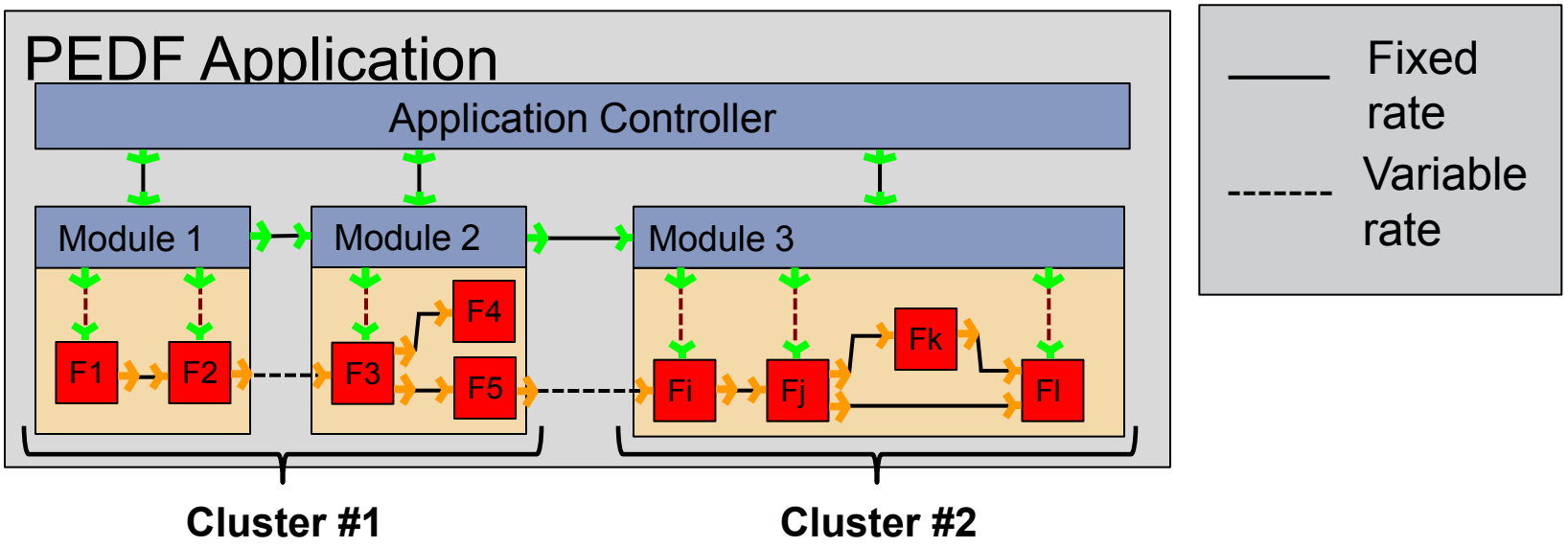
Predicated Execution Data Flow



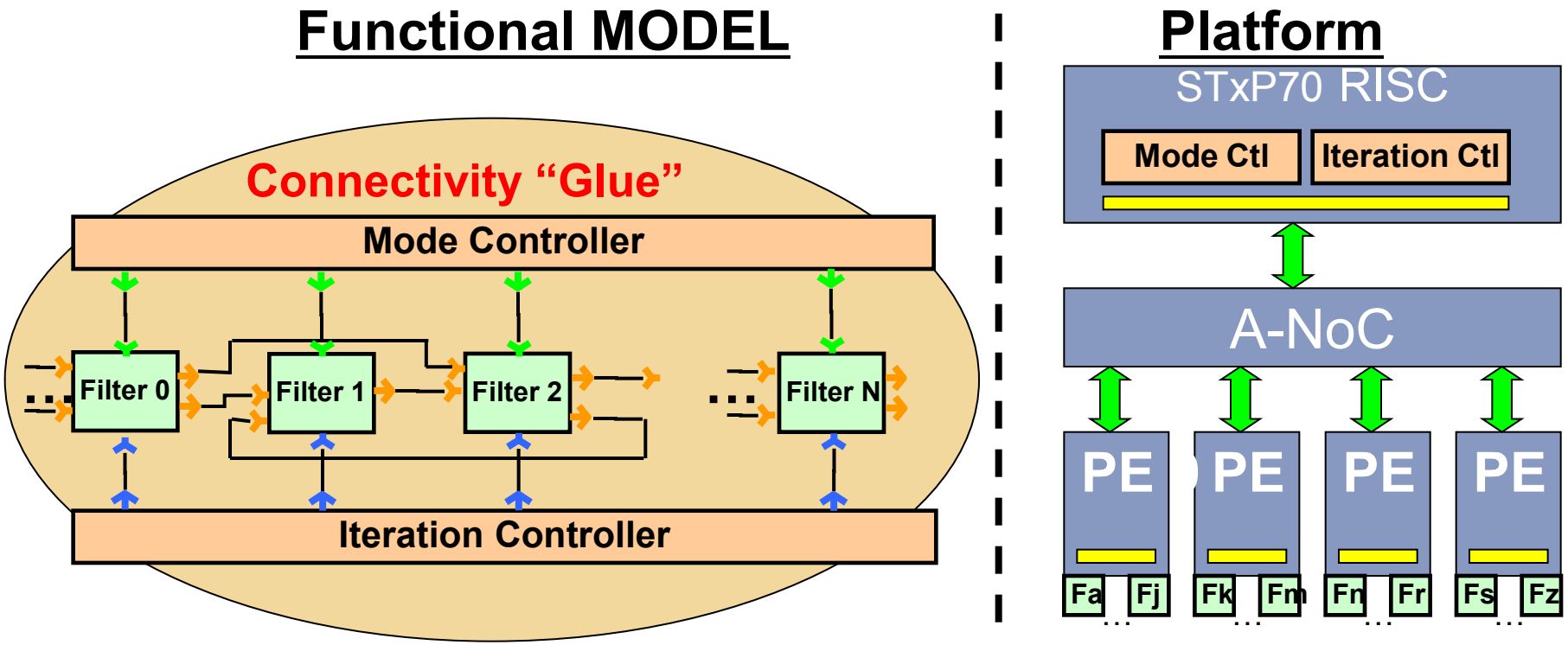
- Host Communication Component
  - Gets host request params.
    - Frame data
    - Required processing type
    - Processing parameters
- Mode Controller
  - Configures control parameters, steps pipeline
- Filters
  - Actual data computation
  - Fixed or variable data rate
- Auto-gen. Iteration Controller

PEDF Progr. Model (contd.)

- Multiple module support
  - **As a mapping unit**: graphs can be mapped on *multiple clusters,* or *multiple Control PEs* of the same cluster.
  - **As an execution unit**: provide a natural split of the applications' execution controller, which can be distributed onto *multiple clusters*.
- Variable-rate support
  - Filters produces and consumes data on as-needed basis



Mapping to HW/SW Platform



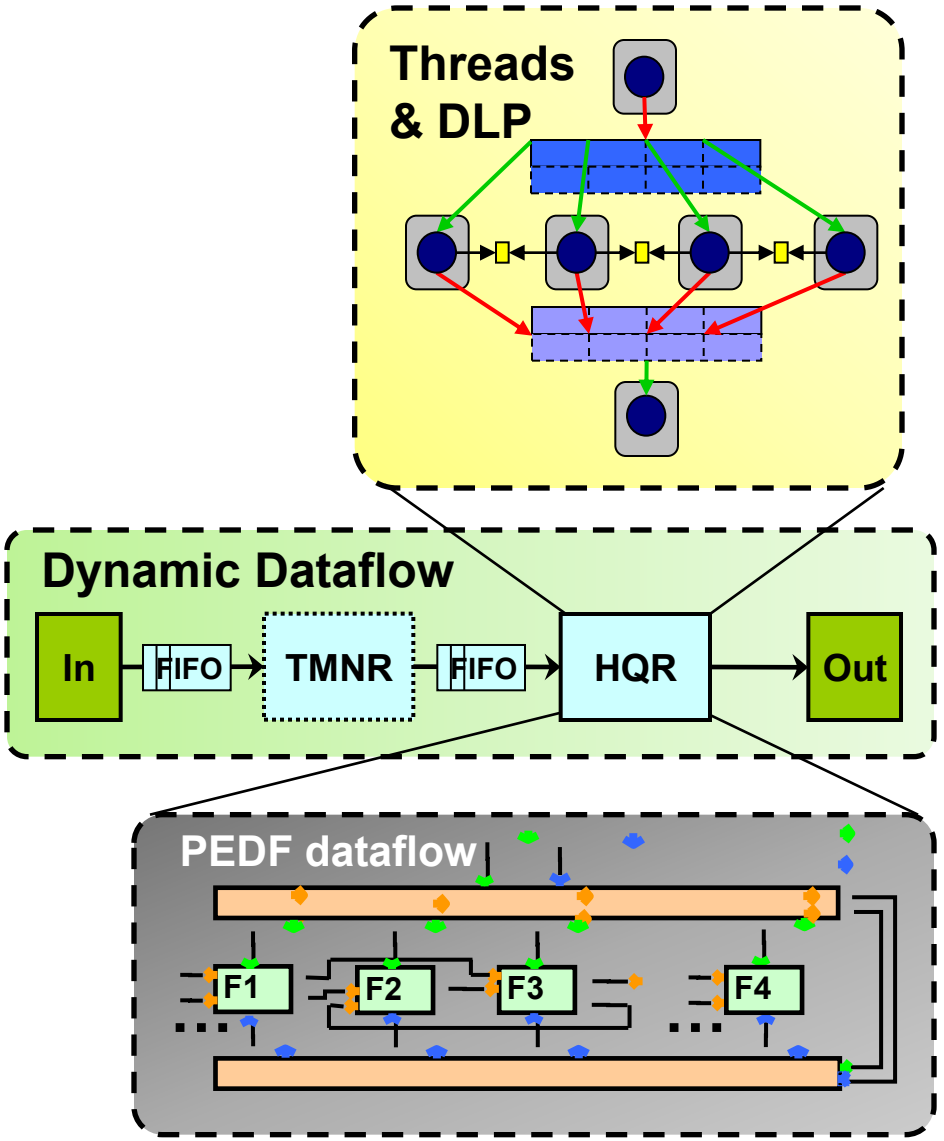
- Application capture using PEDF prog. model
- Functional validation using Apex (on host)
- Automatic control code generation on platform
- Performance analysis

Single progr. model

- Management vision
  - Single programming model for mapping to either H/W or S/W Processing Units
- Reality
  - Single programming model
    - For reference algorithm mostly
    - For H/W-dominated platform with simple S/W filters
  - Decomposition into H/W and S/W-dominated parts
  - Refinement of H/W
    - Mostly task-level parallelism
  - Refinement of S/W
    - Mostly data-level parallelism

Multiple Programming Models

- Top-level
  - Dynamic Dataflow pipeline
- Interchangeable implementations of HQR
  - Thread DLP
  - PEDF TLP
- Other stages of pipeline
  - Can use any of the most appropriate prog. models: PEDF, PPP, DTD ...
- Components act as semantic-neutral structuring mechanism

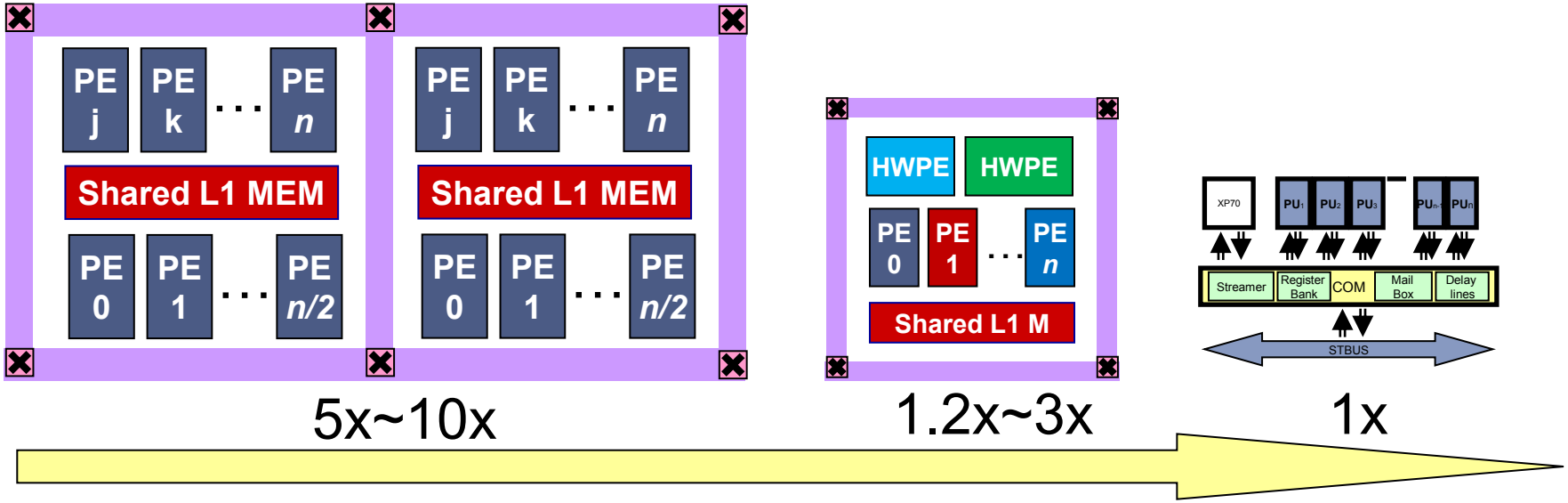


Final lessons

- Need to support multiple programming models
  - PEDF streaming model for H/W dominated platforms
    - Predictable data communication (fixed or variable rate)
    - Simple, well-structured control
    - Highest performance, lowest cost
    - Flexibility in scheduling/control of H/W PEs
  - Native Progr. Models for S/W dominated platforms
    - Components for high-level dataflow
      - Abstract, close to reference algorithm
      - Mapping control, predictable performance
      - Support for platform scaling
    - Effective exploitation of data level parallelism

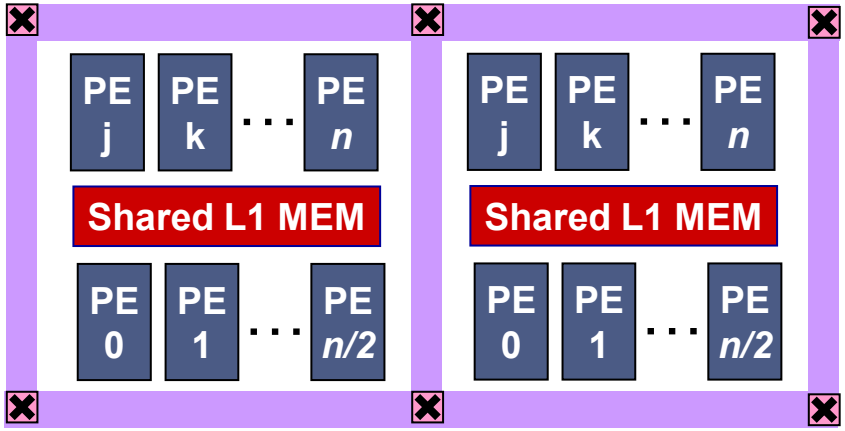
Final lessons (contd.)

- Optimization process is multi-dimensional, multi-level
  - Use of DLP and TLP
  - High-level bandwidth analysis
  - Algorithmic transformations, Vectorization
  - Contention & buffer analysis & optimization
- Need to support range of HW/SW platform variants

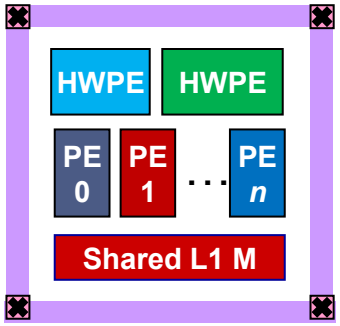


Platform 2012 Use Cases

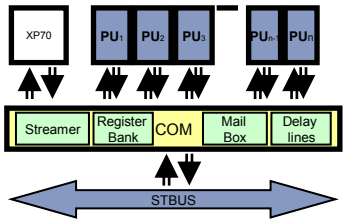
- Mass-market ASSP
  - Full S/W platform
  - Homogeneous processors
  - Time-to-market
- Consumer SoC
  - Mix of H/W and S/W
  - Customized processors
  - Low-cost
  - Tuned flexibility



5x~10x



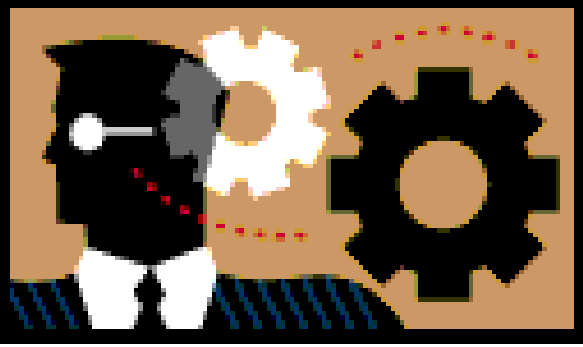
1.2x~3x



1x

Bottom Line

Multi-Processor SoC for Smart People



Programming Models:
- **Higher productivity**
- **Increased platform independence**
- **Multiple objectives**
- **No single silver bullet**