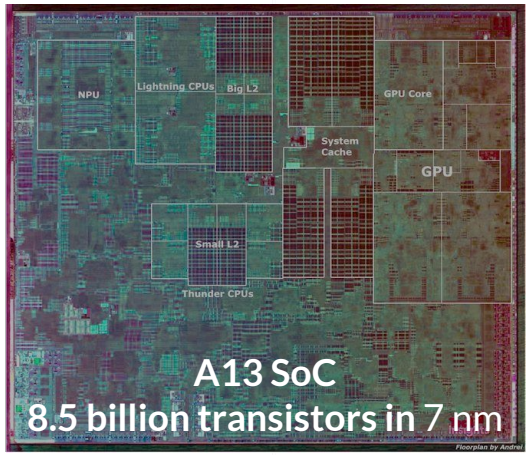
Agile Co-design of Domain-Specific Accelerators and Compilers

Priyanka Raina
Stanford University
praina@stanford.edu

Domain-Specific Accelerators

- With the slowdown of Moore's law and end of Dennard scaling, hardware specialization is necessary to improve performance and energy efficiency of computing systems



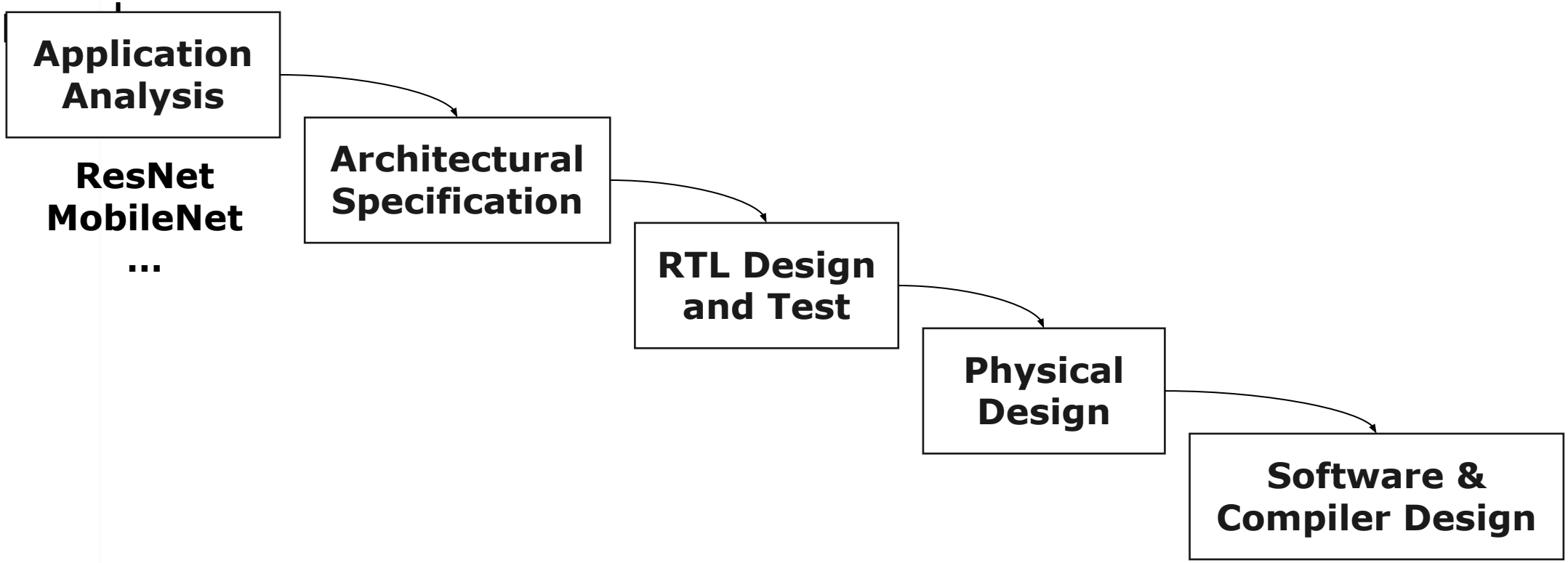
Modern SoCs have dozens of domain-specific accelerators

- Graphics
- Machine Learning
- Image Processing
- Video Coding
- Cryptography
- Wireless …

Image source: https://www.anandtech.com/show/14892/the-apple-iphone-11-pro-and-max-review/2

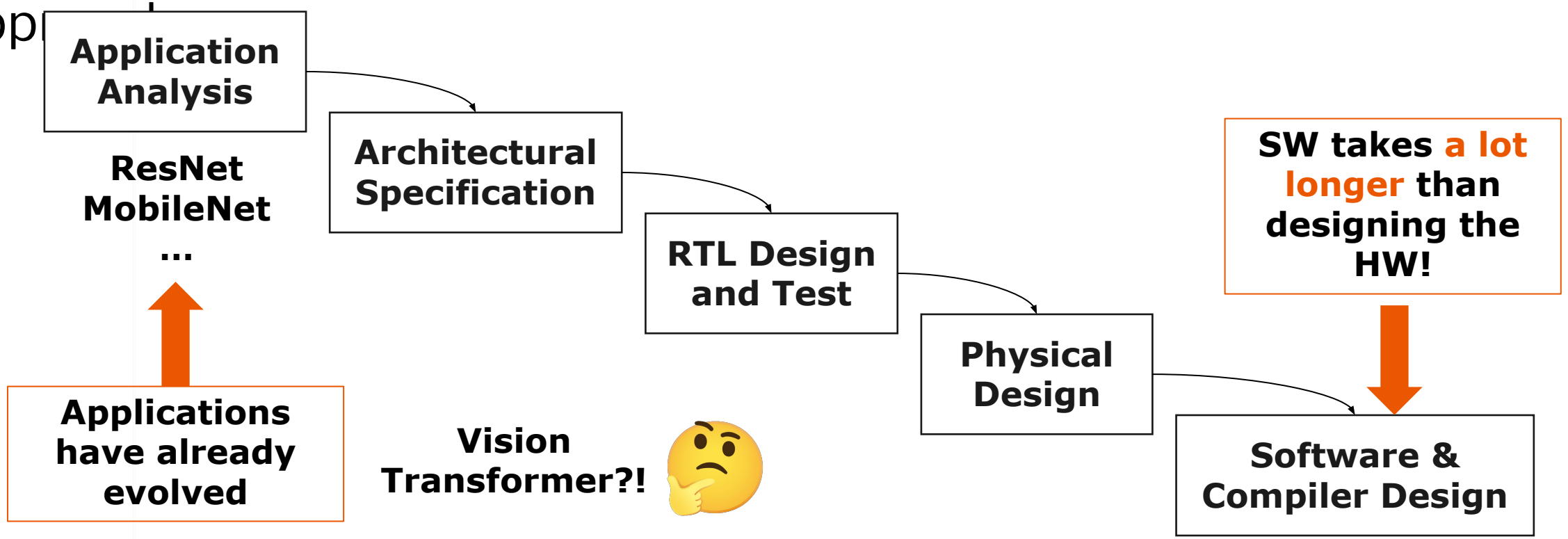
Existing Approach to Accelerator Design

- The most common approach to create accelerators is a waterfall app



Existing Approach to Accelerator Design

- The most common approach to create accelerators is a waterfall app



Cost: Software > Verification > Design > Prototype

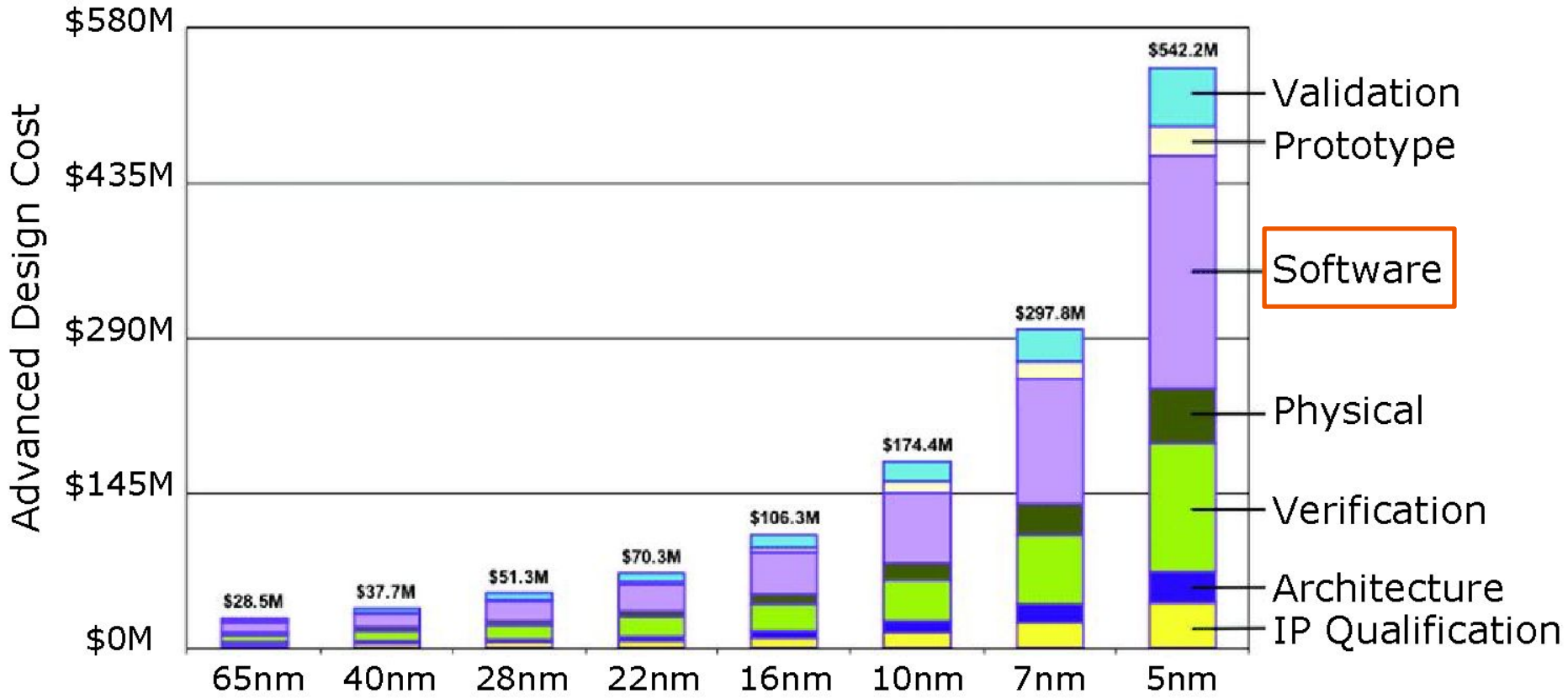


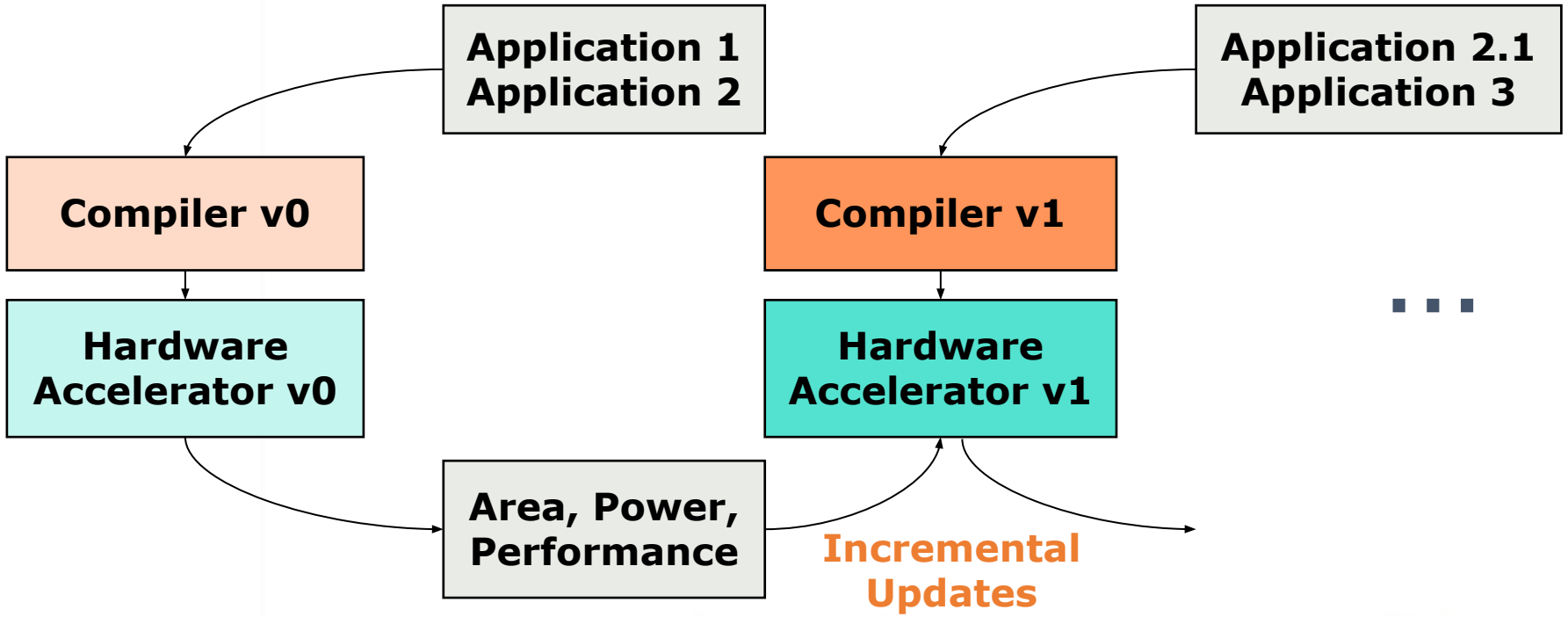
Image source: https://www.researchgate.net/figure/Chip-Design-and-Manufacturing-Cost-under-Different-Process-Nodes-Data-Source-from-IBS_fig1_340843129

Challenges to Adopting Hardware Specialization at Scale

- Prohibitive **cost of design and verification** of accelerator-based systems
- Lack of a **structured approach for evolving the software stack** as the underlying hardware becomes more specialized
- No general **methodology for specializing hardware to *domains***, rather than a few benchmarks

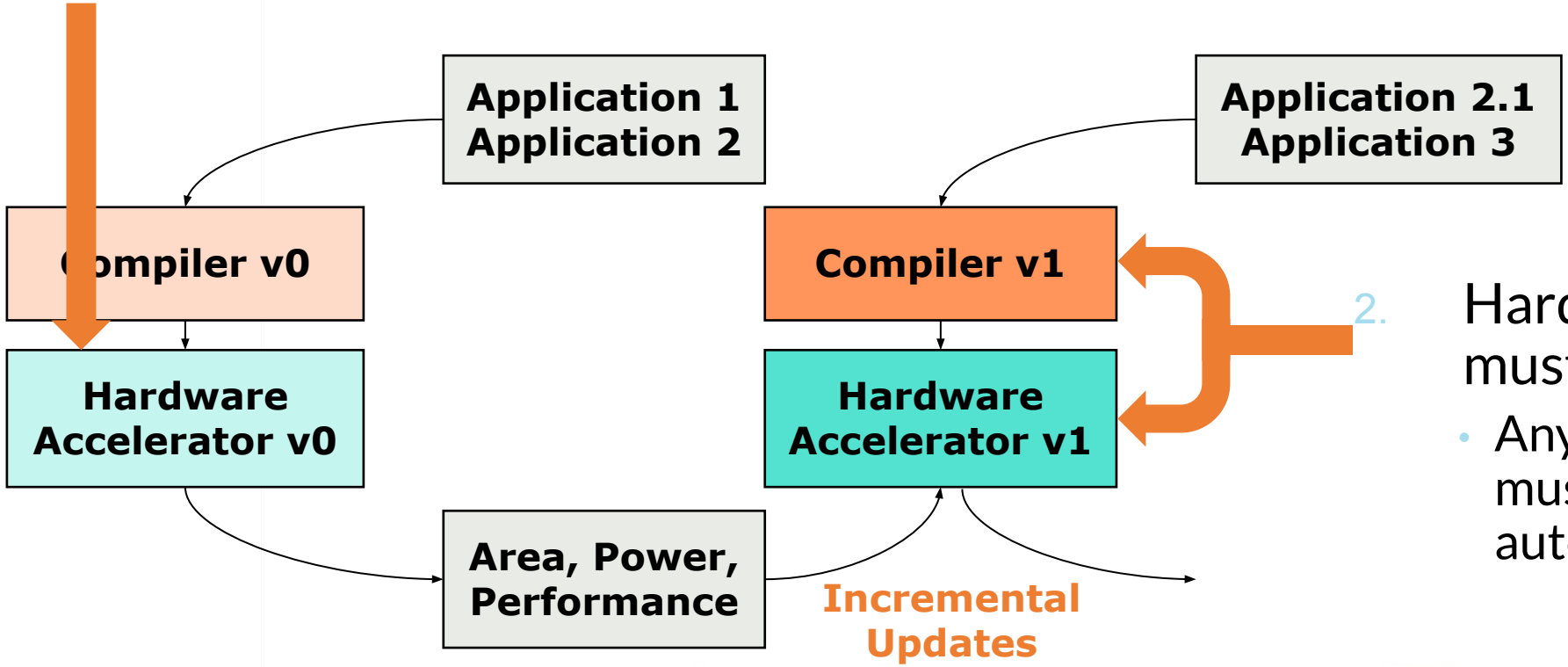
Agile Hardware-Software Design (AHA)

- Agile approaches are very common in SW development --- we create tools to adapt them to HW/compiler co-design
- Incrementally update the hardware accelerator and software to map to it



Requirements for the Agile Approach

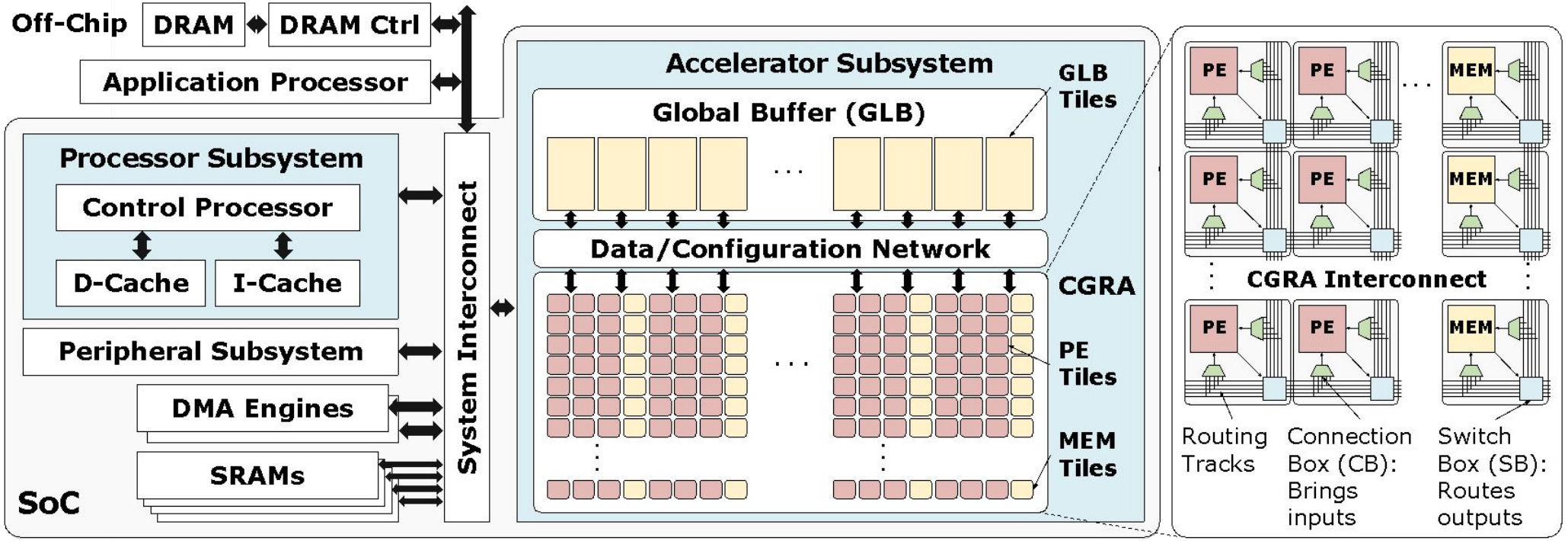
1. Accelerator must be **configurable**
   - So we can map new or modified applications to it (although with lower efficiency)



2. Hardware and compiler must **evolve together**
   - Any change in hardware must propagate to compiler automatically

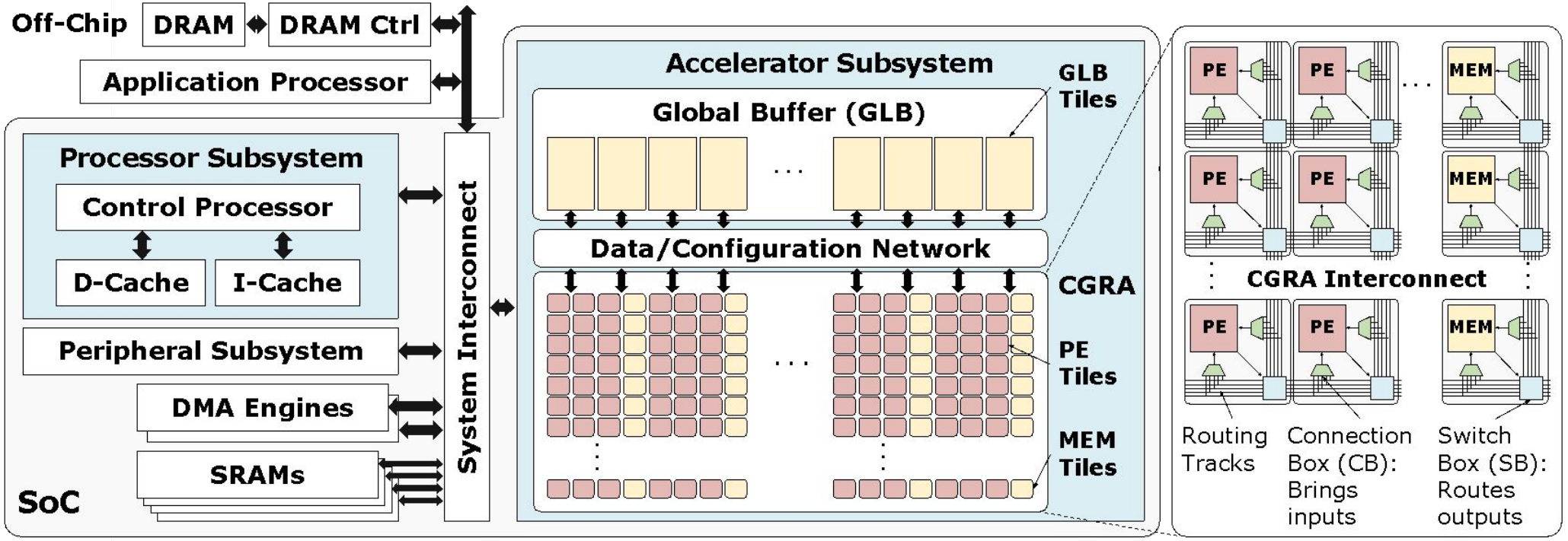
CGRAs as Accelerator Templates

- Think about **accelerators as specialized coarse-grained reconfigurable arrays** (CGRAs)
  - Like an FPGA but with larger compute and memory units, and word-level interconnect



CGRAs as Accelerator Templates

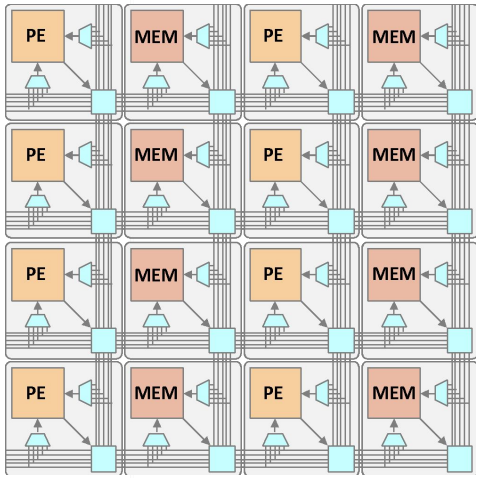
- Is programmable enough to accommodate application evolution
- Allows specialization and exploiting parallelism and locality --- characteristics that make an accelerator efficient



CGRAs as Accelerator Templates

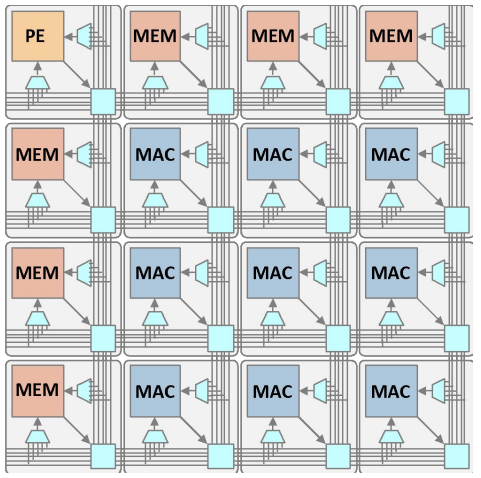
- By tuning the amount of configurability in CGRA PEs, MEMs and the interconnect, we can create more specialized (closer to ASICs) or more general-purpose accelerators (closer to FPGAs)

Base CGRA



Incremental Updates

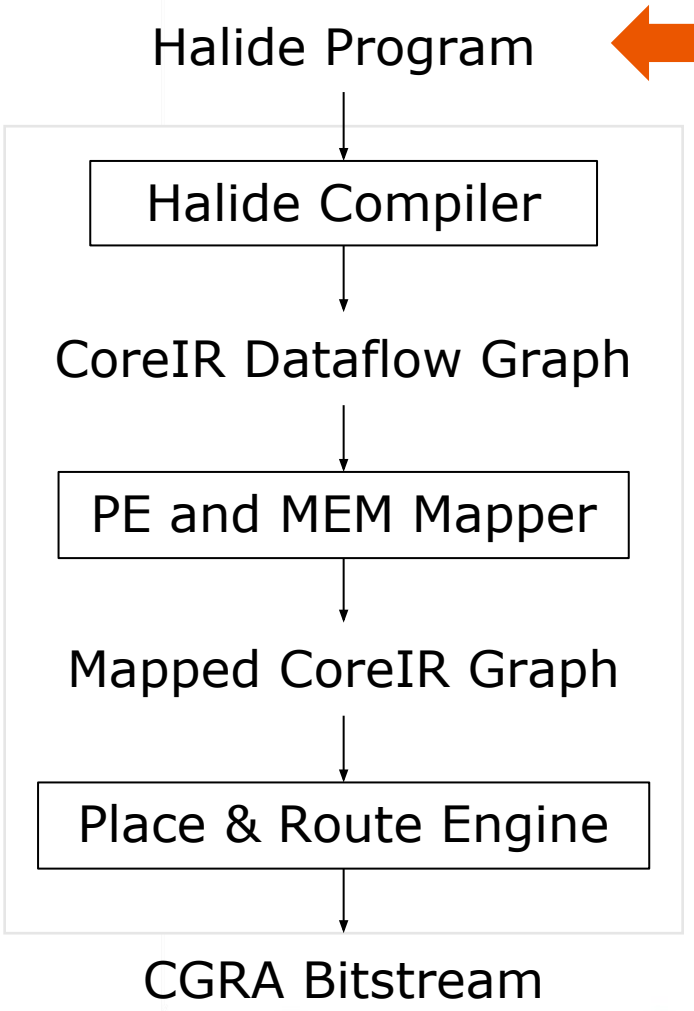
- Make PE a MAC
- Simplify interconnect to systolic connections
- …



NN-CGRA ≡ TPU-like Accelerator

- More importantly, thinking of accelerators as specialized CGRAs provides a standard accelerator template for a compiler to target

Compiler from Halide Applications to CGRAs



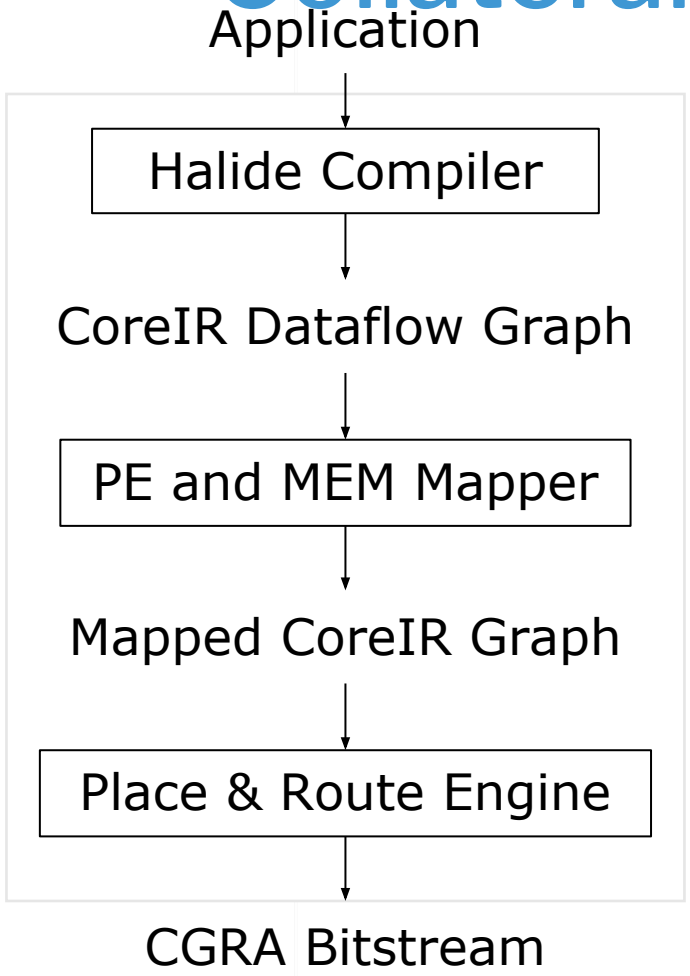
Dense Linear Algebra
(Machine Learning, Image Processing, etc.)

Hardware **independent**

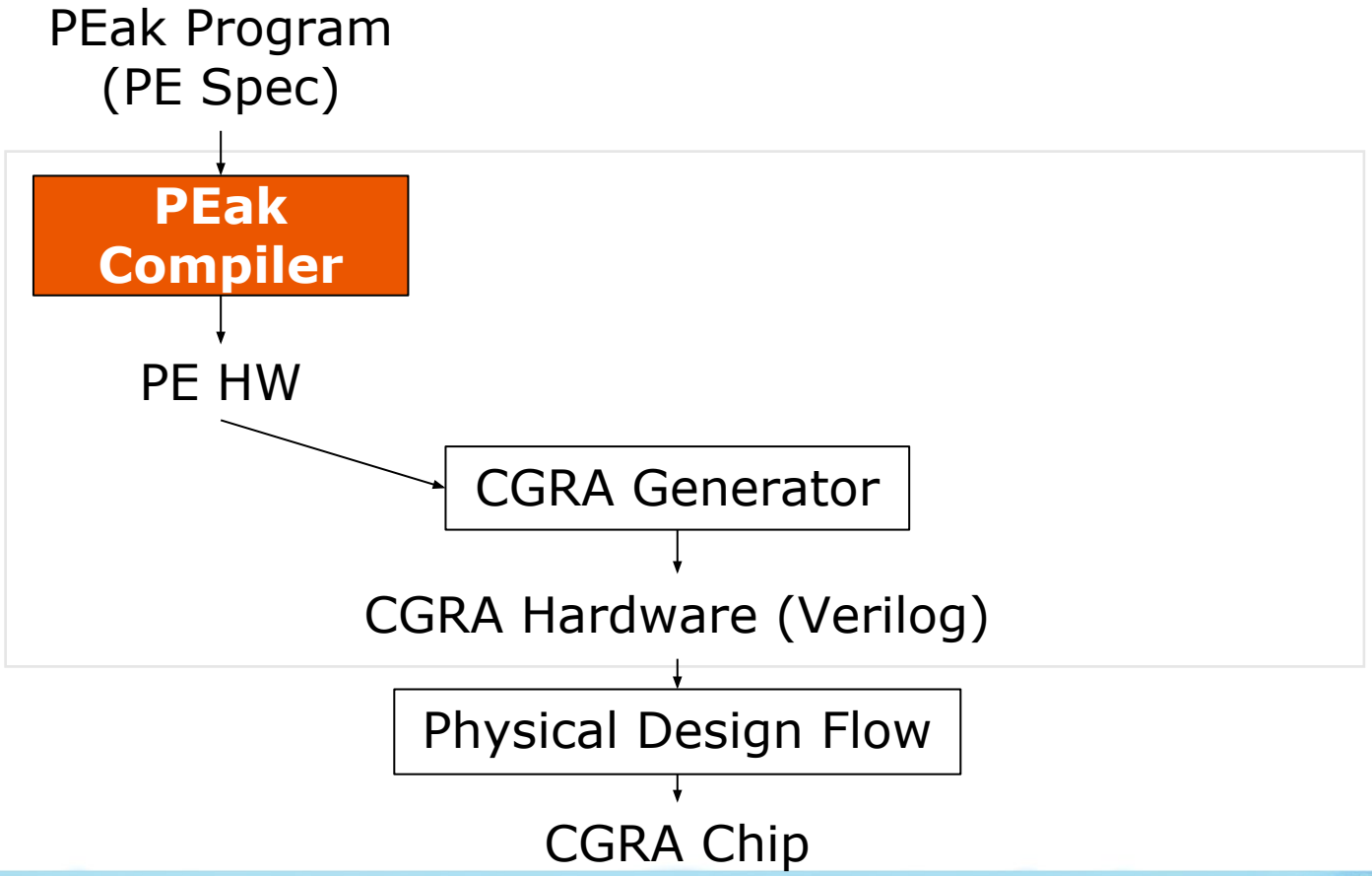
Hardware **dependent**

Q. Liu et al, "Unified Buffer: Compiling image processing and machine learning applications to push-memory accelerators," TACO 2022

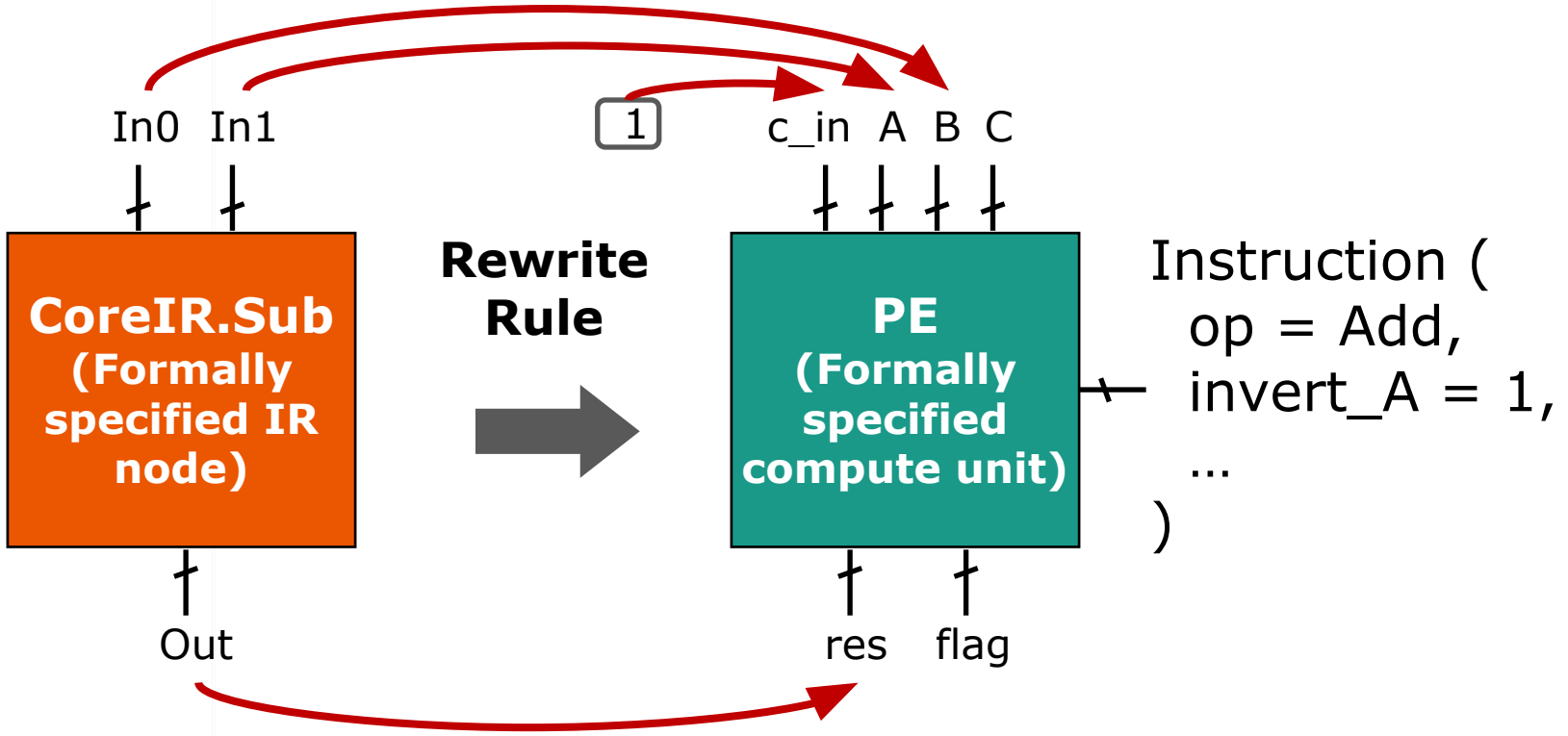
Automatically Generate HW and Compiler Collateral from a Single Source of Truth



Formal Specifications for Accelerator Components



Automatic Rewrite Rule Generation with SMT

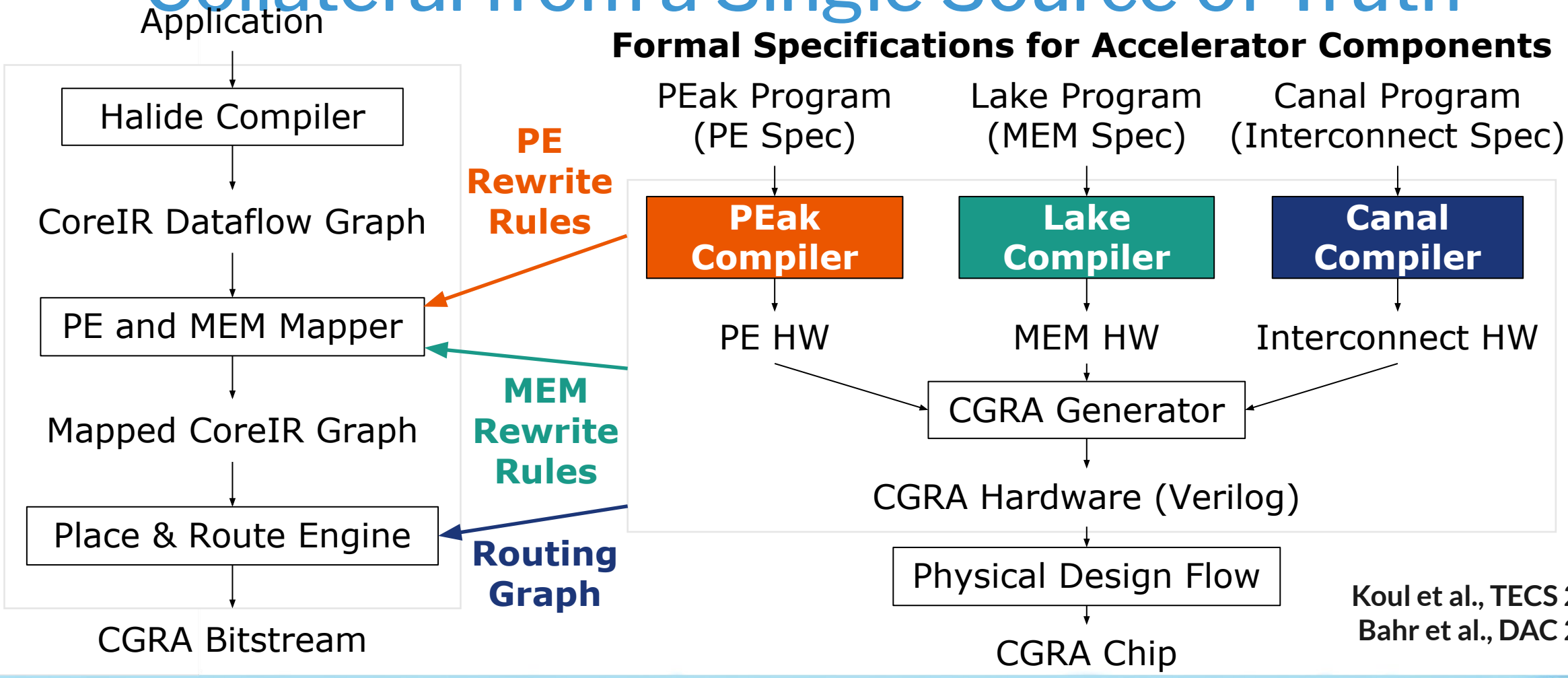


$\exists inst\ \forall inputs :$
$CoreIR.Op(inputs) == PE(inst,inputs)$

Solve using an SMT solver

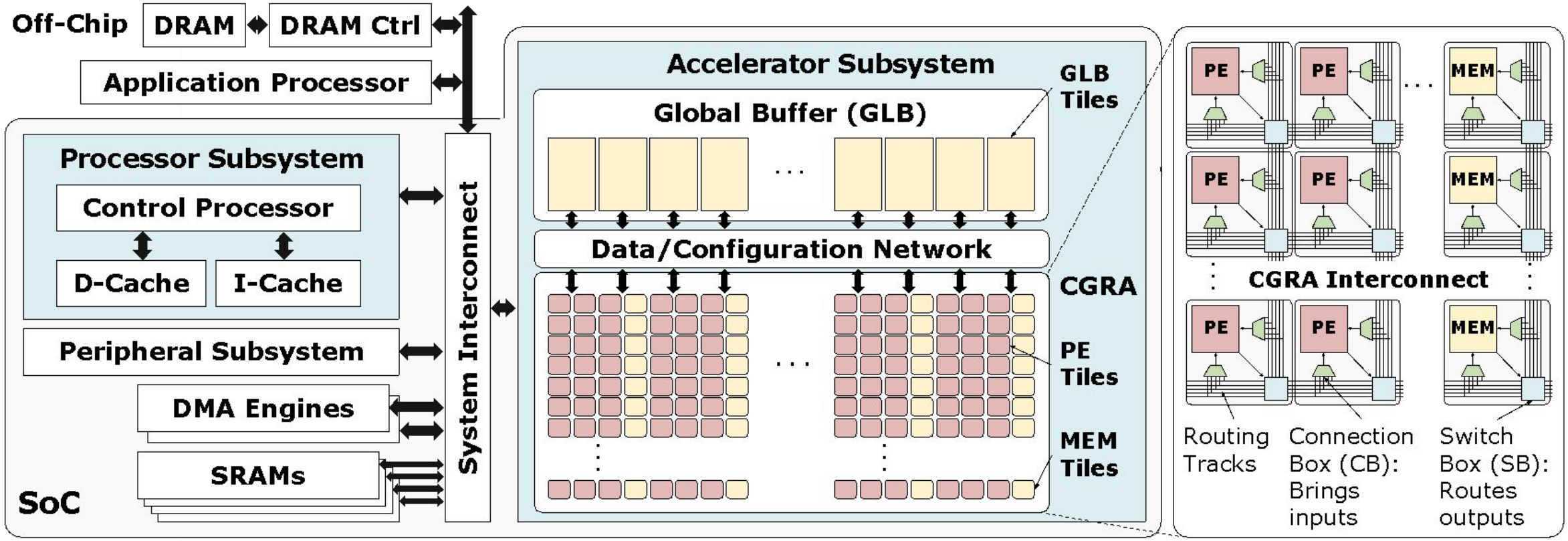
Daly et al., Synthesizing Instruction Selection Rewrite Rules from RTL using SMT, FMCAD 2022

Automatically Generate HW and Compiler Collateral from a Single Source of Truth



Koul et al., TECS 2022
Bahr et al., DAC 2020

Amber: CGRA SoC Designed with Agile Approach



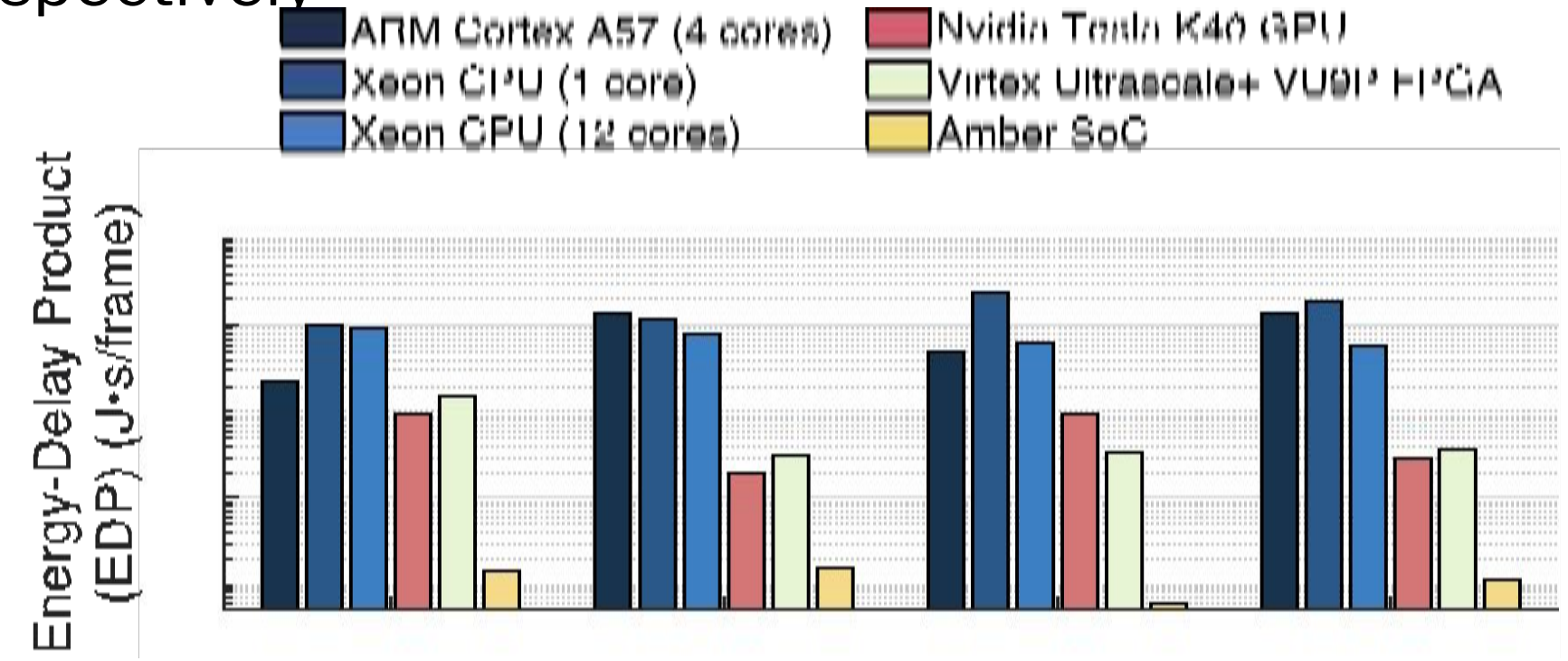
Carsello et al., VLSI 2022

ISSCC 2022 Student Research Preview Poster Award
VLSI 2022 Best Demo Paper Award

Amber: CGRA SoC Designed with Agile Approach

- Amber achieves 160-1200x, 678-3902x, 498-988x, 12-152x, and 20-107x better EDP vs. Cortex A57, 1-core and 12-core Xeon CPUs, GPU, and FPGA respectively

Amber in TSMC 16 nm



Application-Driven Design Space Exploration

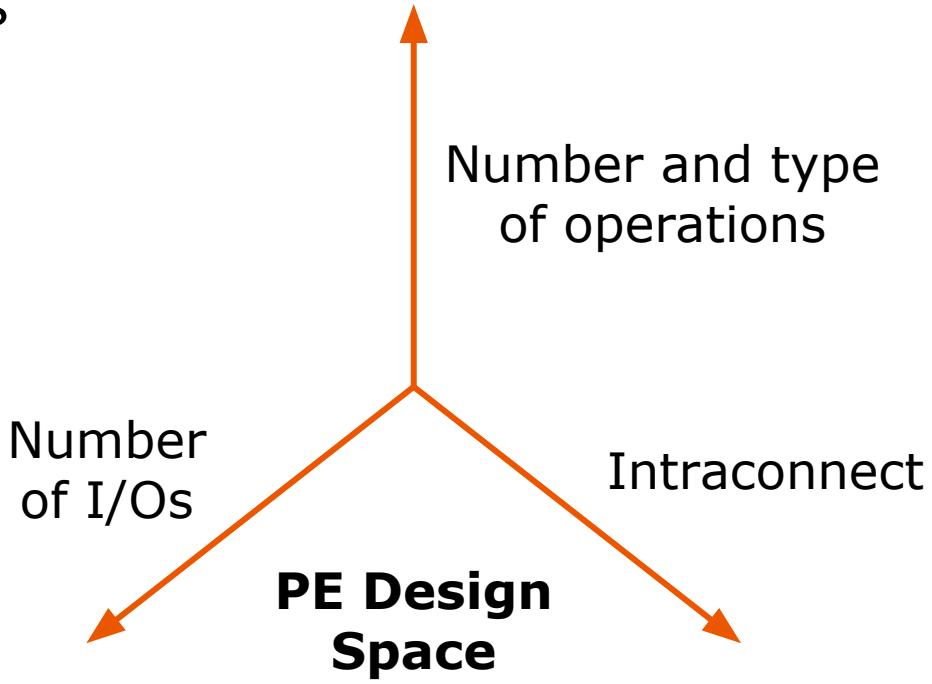
- Architects often explore many alternatives when designing an accelerator
- A major impediment to design-space exploration (DSE) is implementing the software changes needed to compile the application to the new accelerator
- Hardware-compiler codesign can automate large-scale DSE of accelerators

Application-Driven Processing Element Design

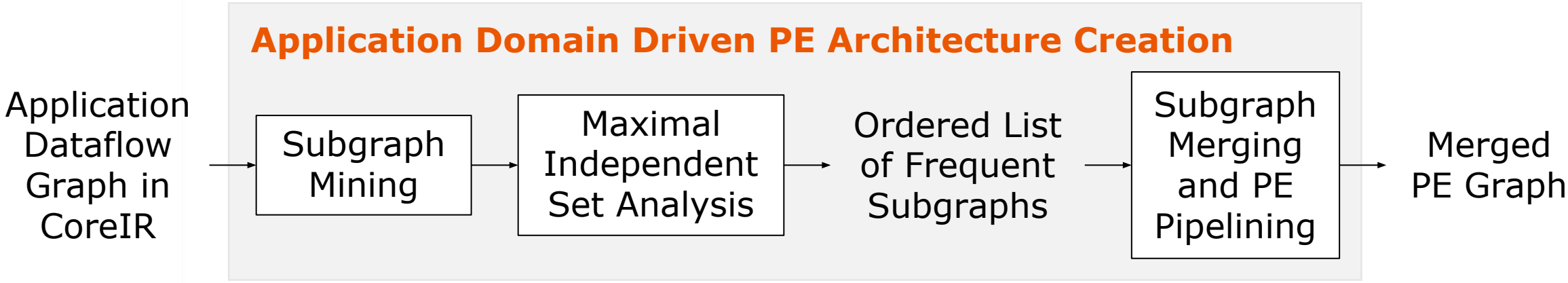
- PEs have a complex design space: (Simple instructions in PE, more PEs, more interconnect overhead) vs. (Complex instructions in PE, potentially fewer PEs)
- A naïve enumeration leads to many candidate Pes
- Key idea: **Application-driven exploration**

Generate efficient PE architectures from the application graphs themselves using frequent subgraph mining and merging

Automatically update the compiler with AHA flow and measure impact on applications

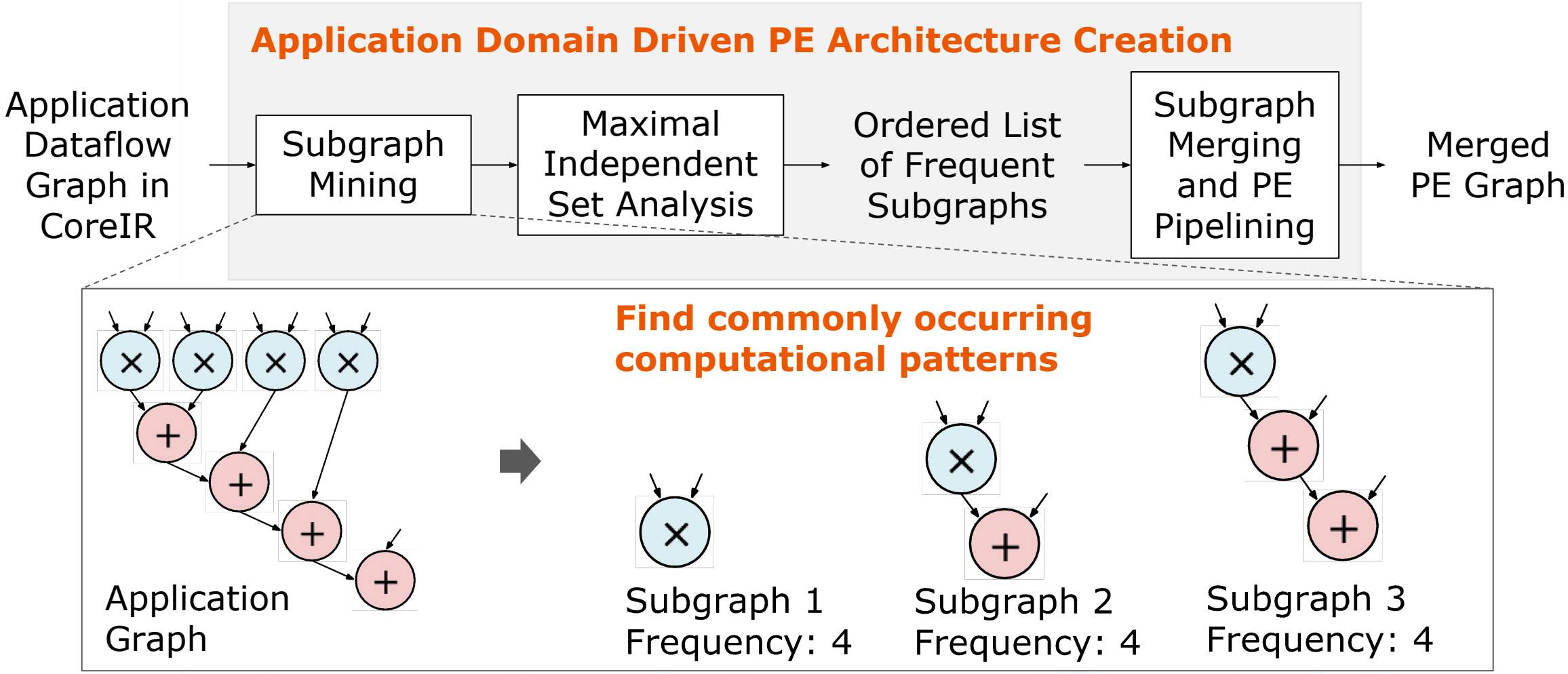


APEX: Application-Driven PE Exploration

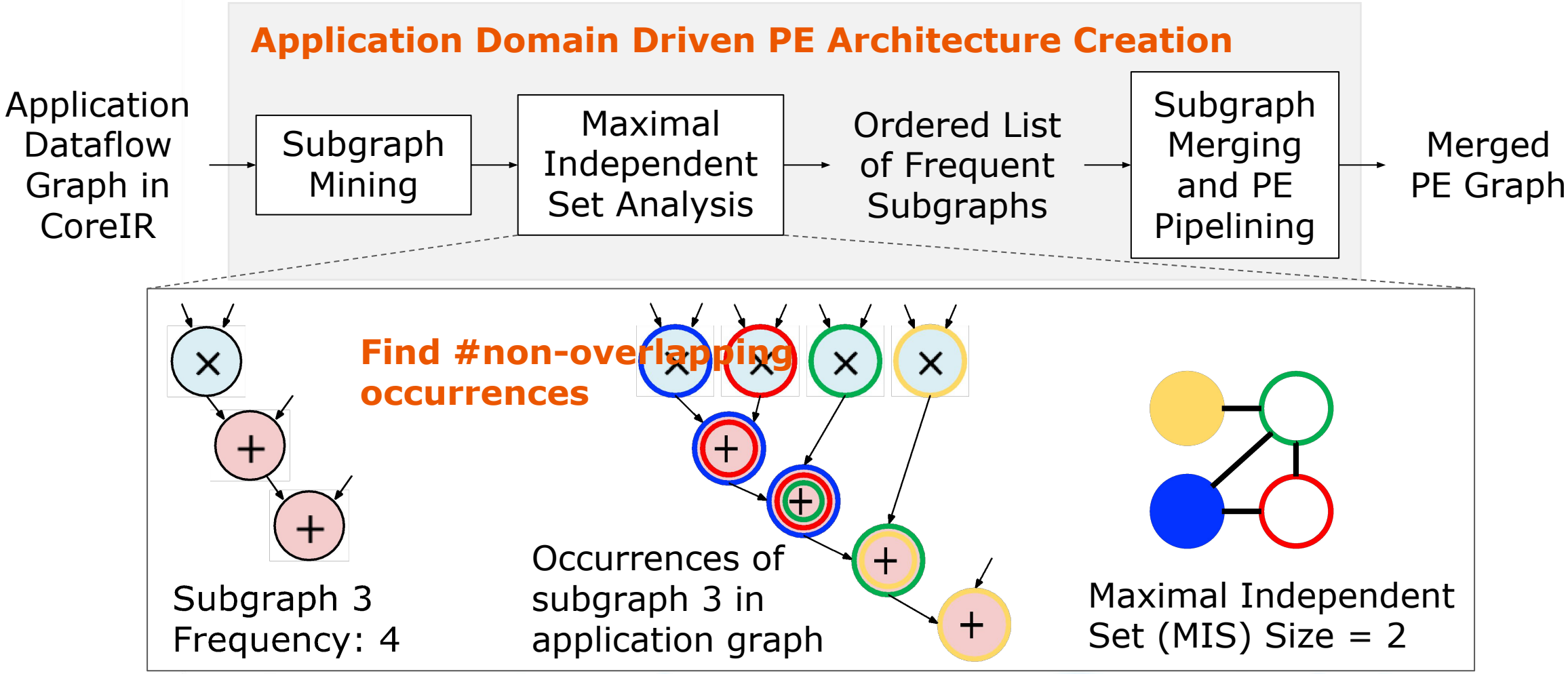


Melchert et al., APEX: A Framework for Automated Processing Element Design Space Exploration using Frequent Subgraph Analysis, ASPLOS 2023

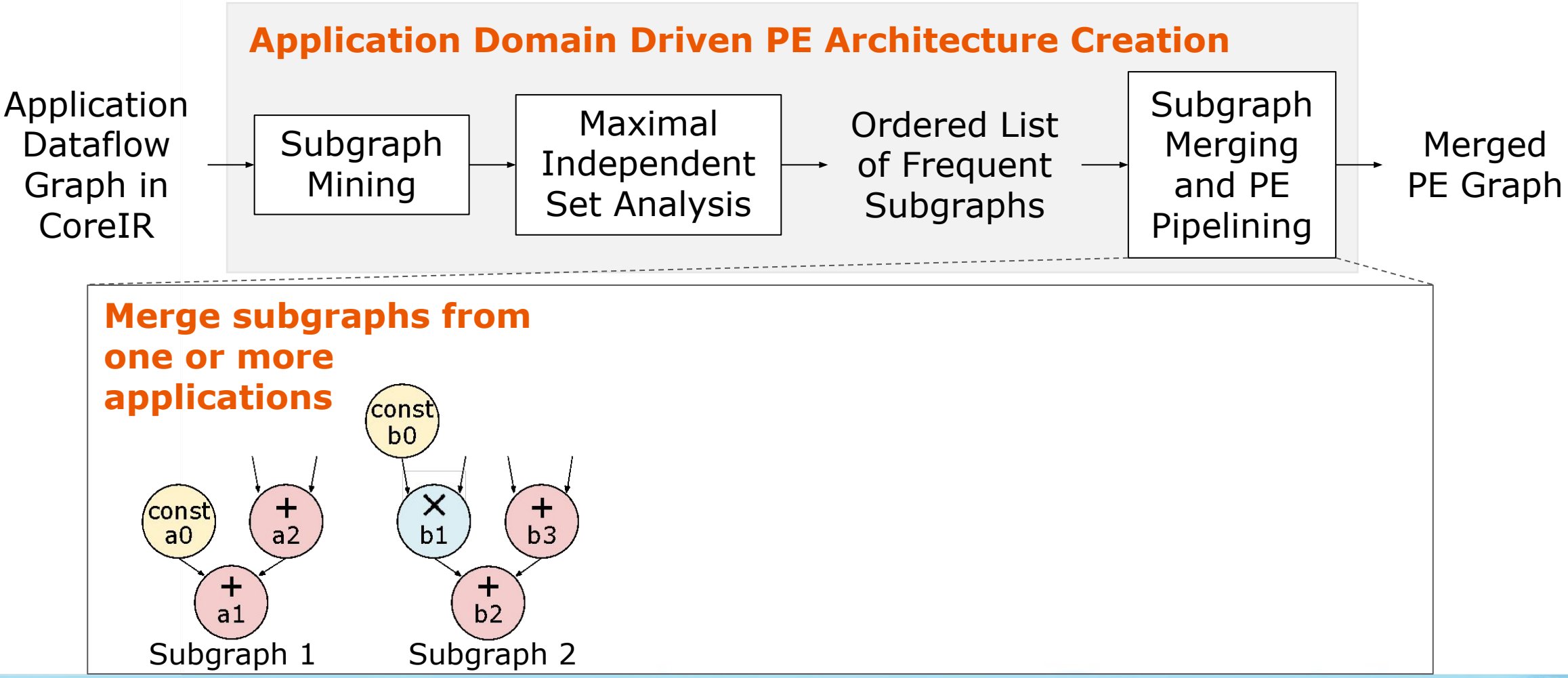
APEX: Application-Driven PE Exploration



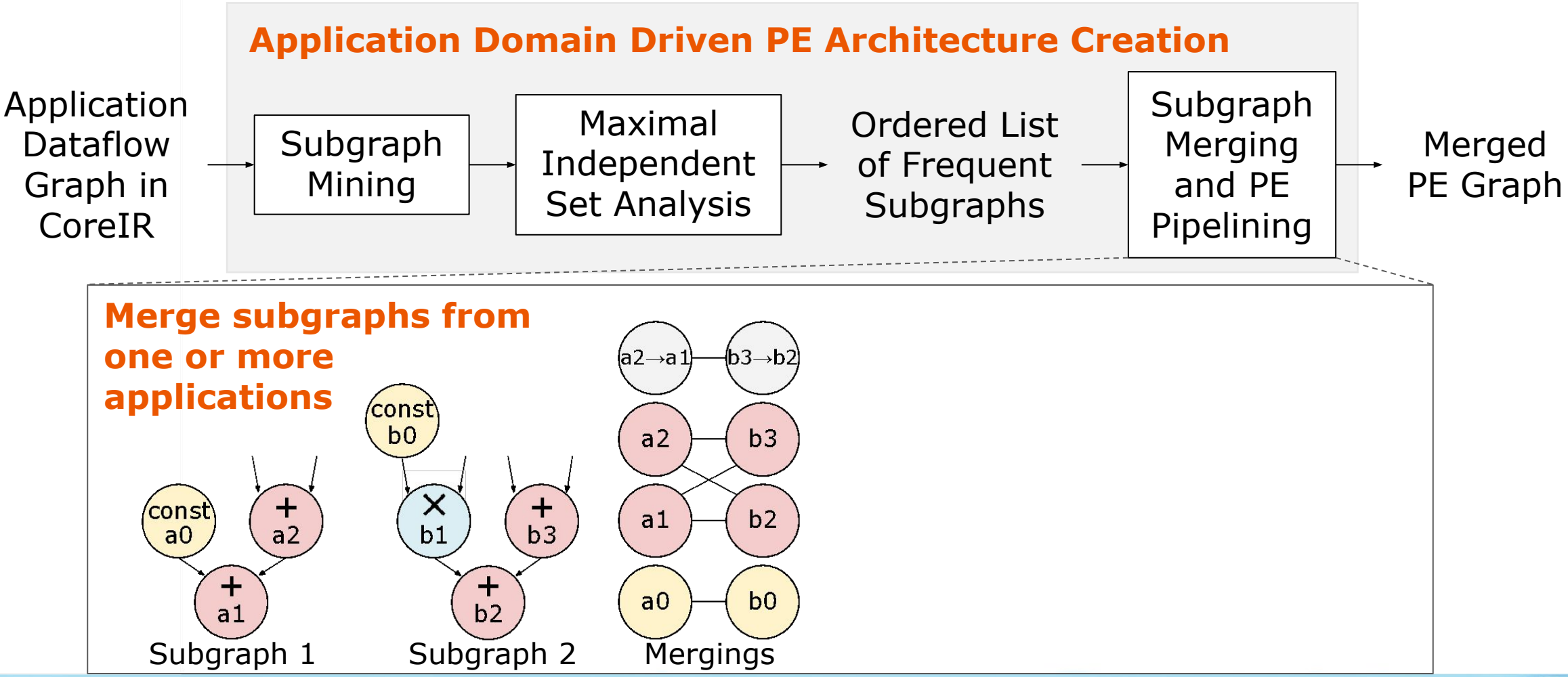
APEX: Application-Driven PE Exploration



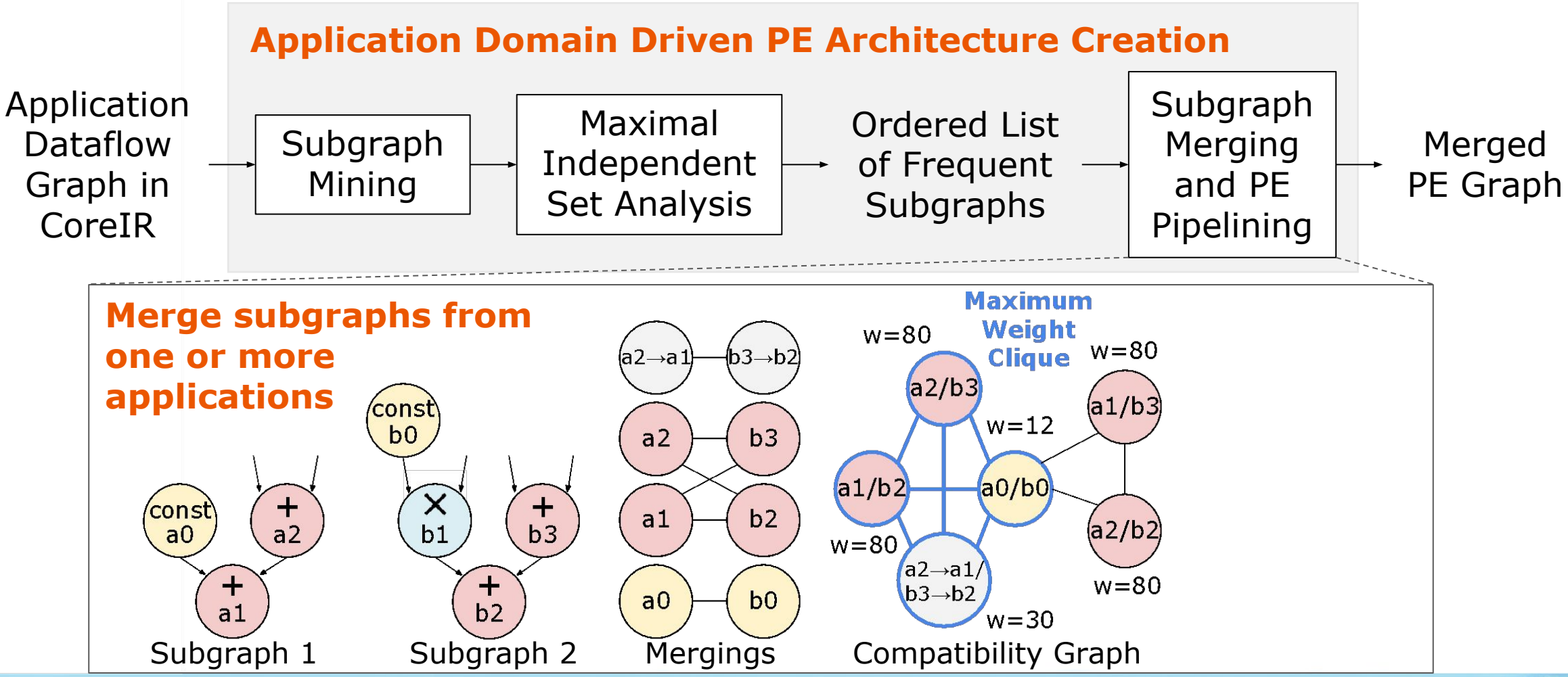
APEX: Application-Driven PE Exploration



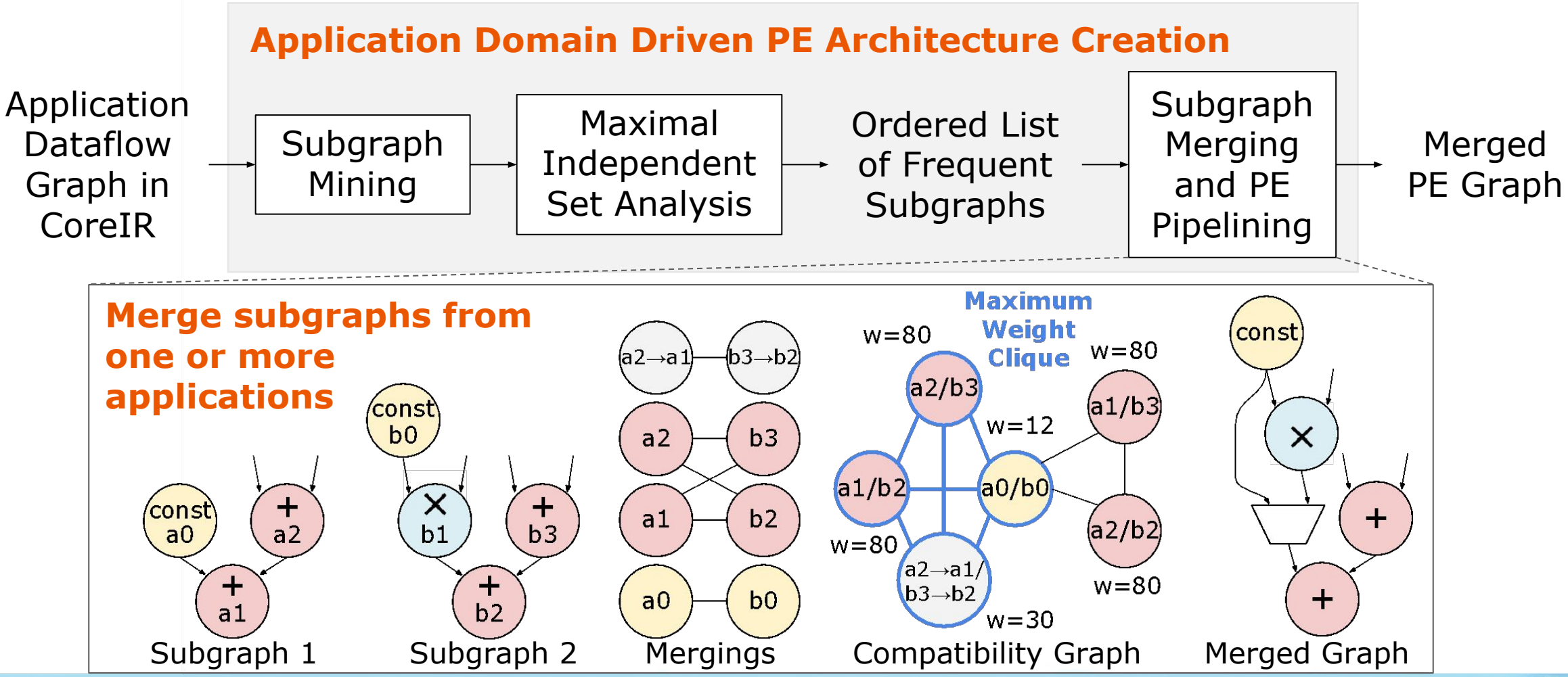
APEX: Application-Driven PE Exploration



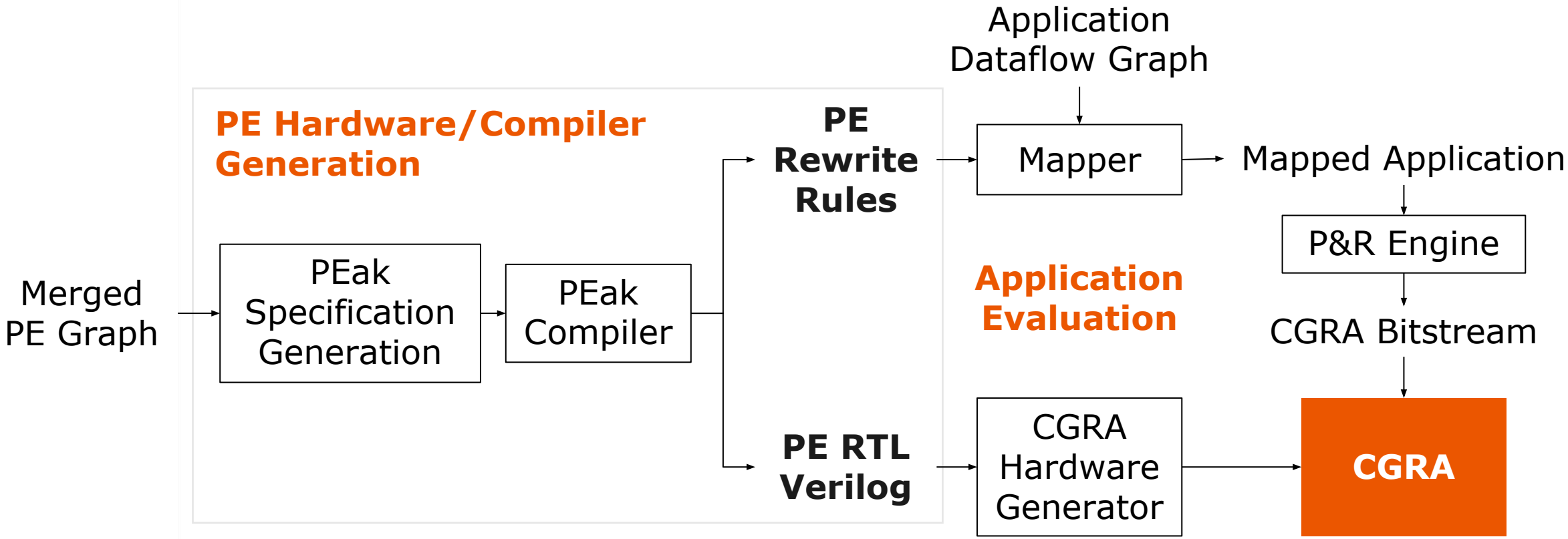
APEX: Application-Driven PE Exploration



APEX: Application-Driven PE Exploration

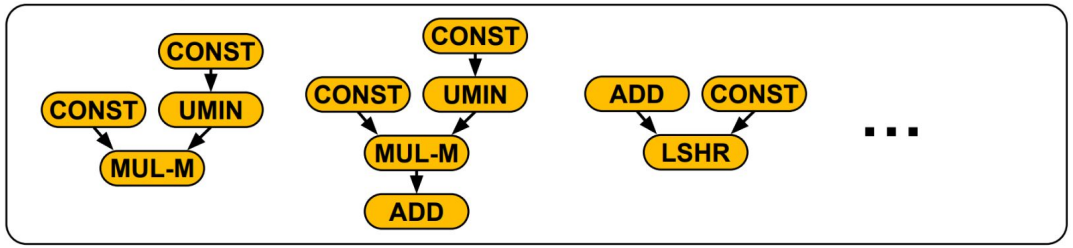


APEX: Application-Driven PE Exploration

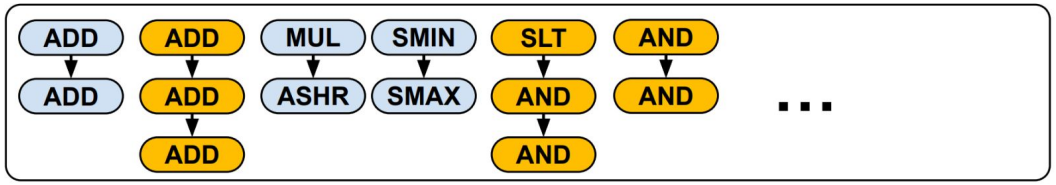


Using APEX for Onyx PE Optimization

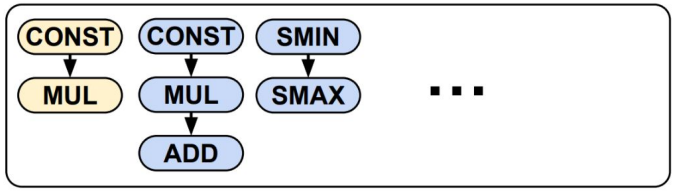
Camera Pipeline



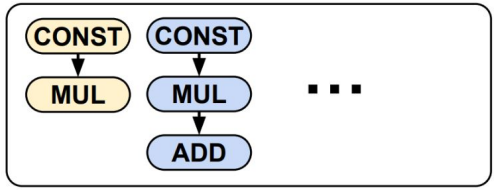
Harris Corner Detection



Unsharp

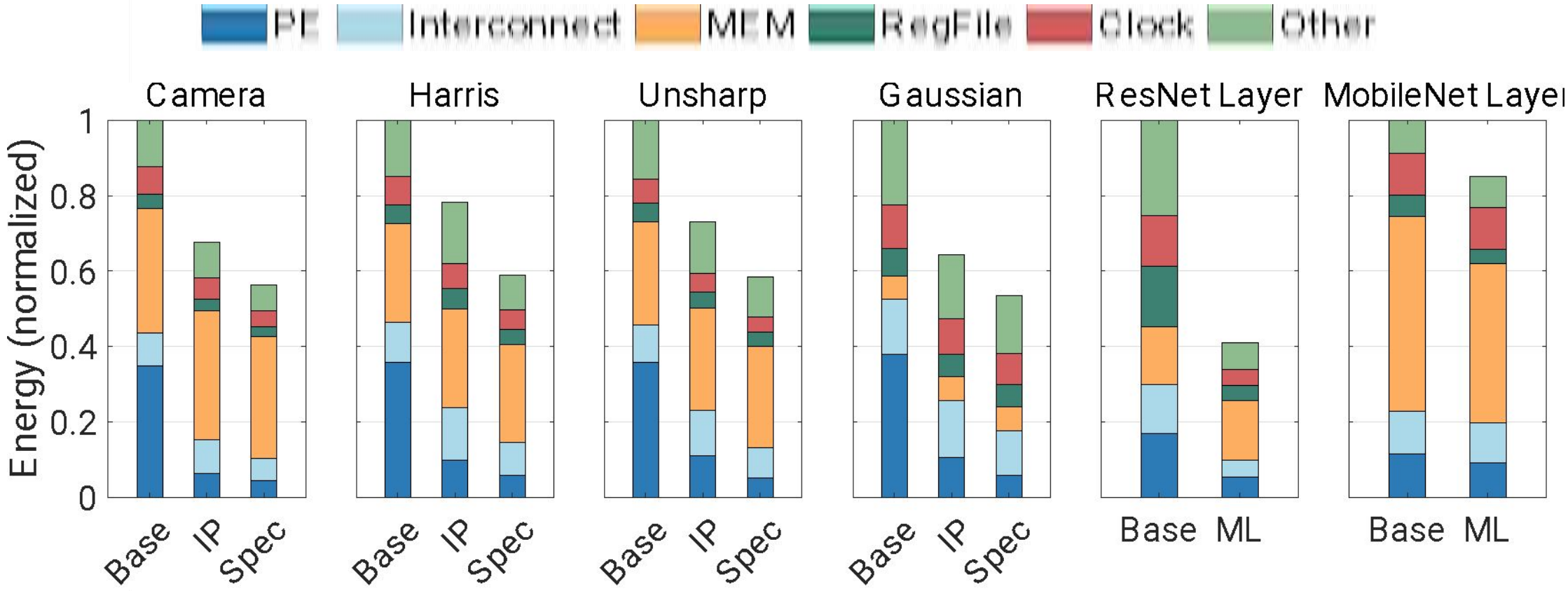


Gaussian Filter



Decreasing MIS

PE Optimization Results



Summary: Agile Hardware-Software System Co-design Methodology

- Thinking about **accelerators as specialized CGRAs** provides a standard accelerator template for a compiler to target --- allowing us to create such a compiler
- Using a combination of new programming languages and formal methods, we can **automatically generate the accelerator hardware and its compiler** from a single source of truth
- Enables creating higher-level **design-space exploration (DSE) frameworks** for application-driven optimization of accelerators

Potential for **agile evolution of specialized systems** allowing for reuse of large parts of the hardware as well as the compiler toolchain, improving design productivity