



Advisor



Student



Software

Hardware



uIR

Hardware Construction by Software

Arrvindh Shriraman,
Simon Fraser University

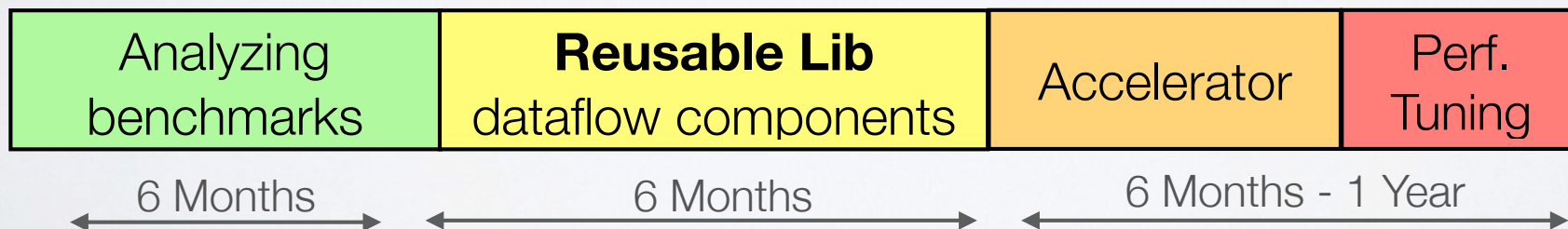
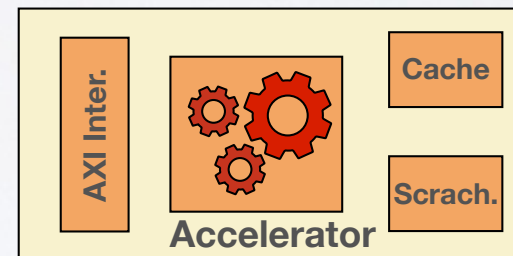
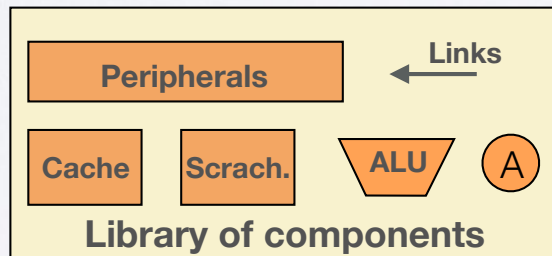
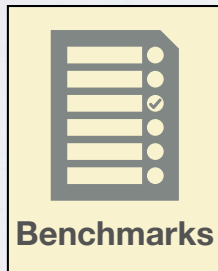
<https://github.com/sfu-arch/muir>

ARM Summit

Current Phd/MASc Pipeline

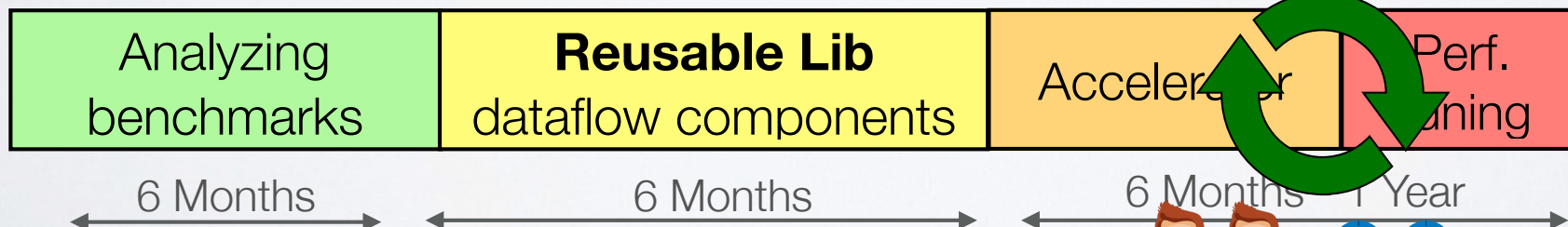
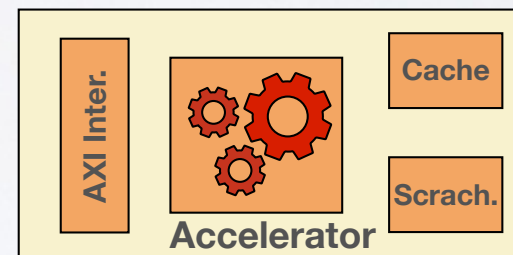
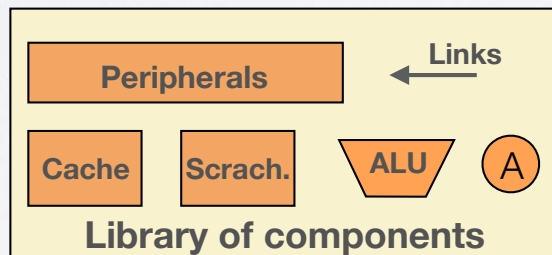
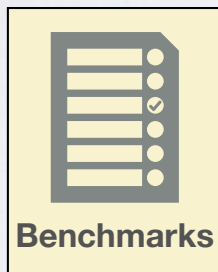


Lets build an accelerator



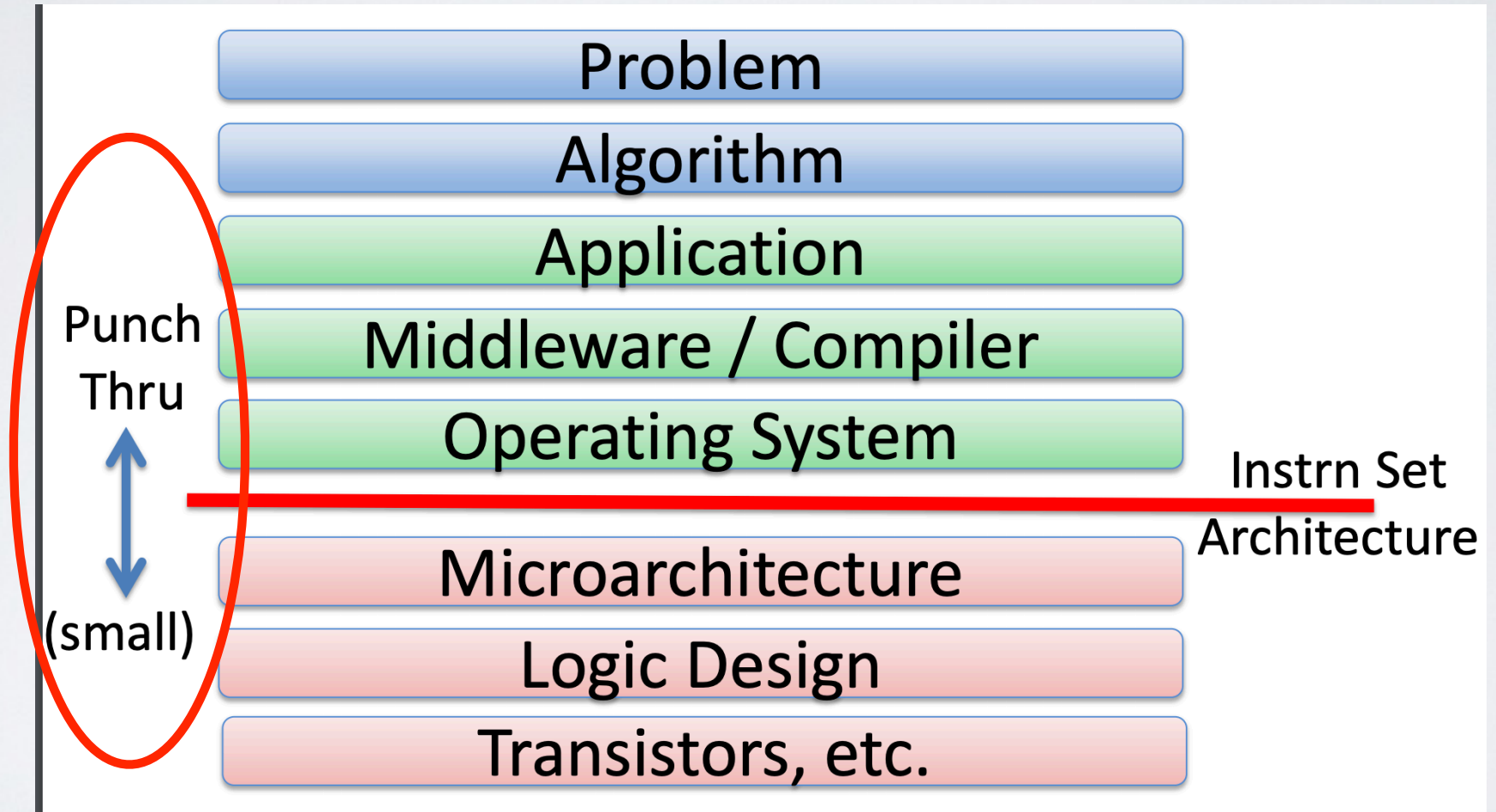
Current Phd/MASc Pipeline

Lets build another accelerator

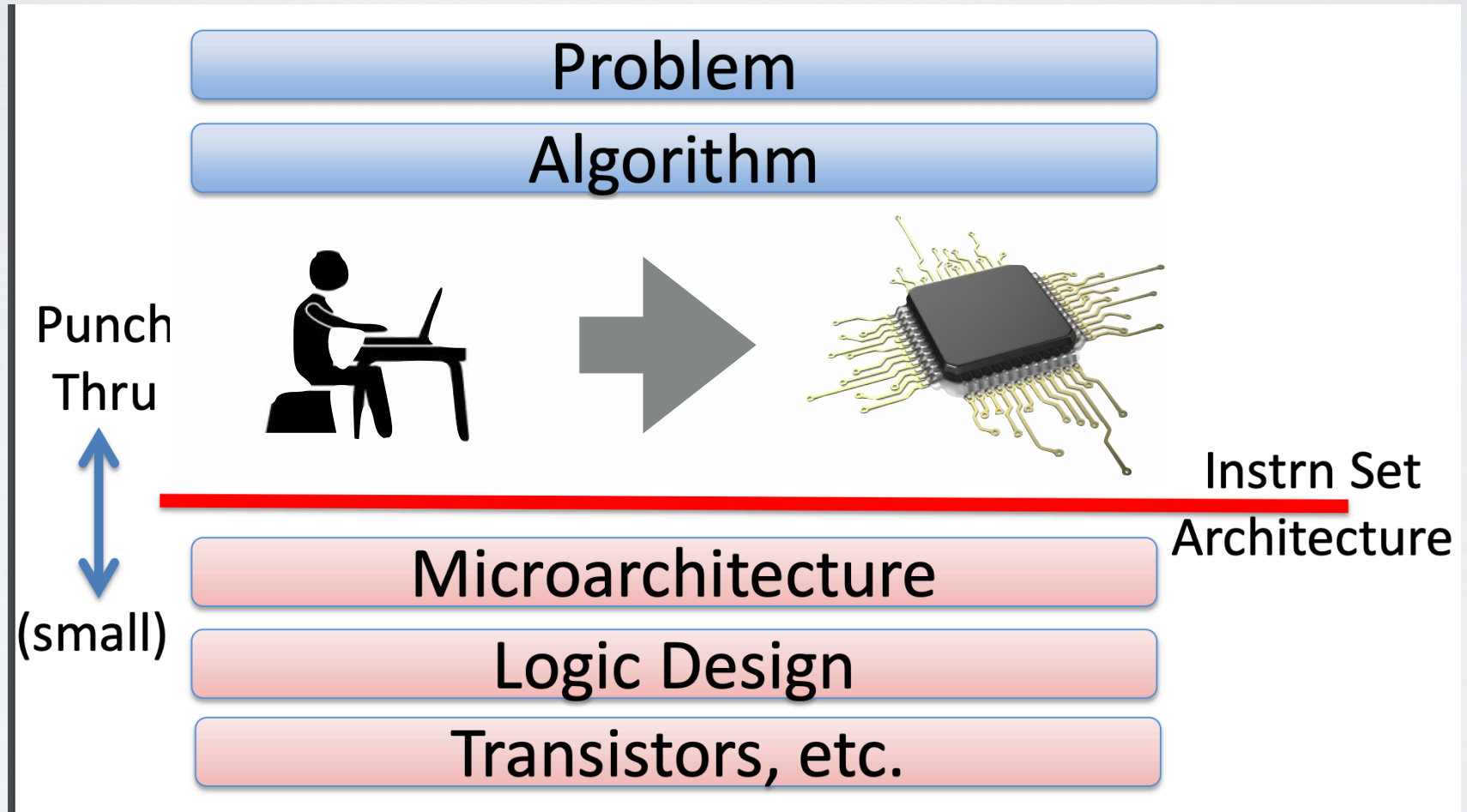


Blast from the recent past

NSF 21st century computer architecture workshop (2013)

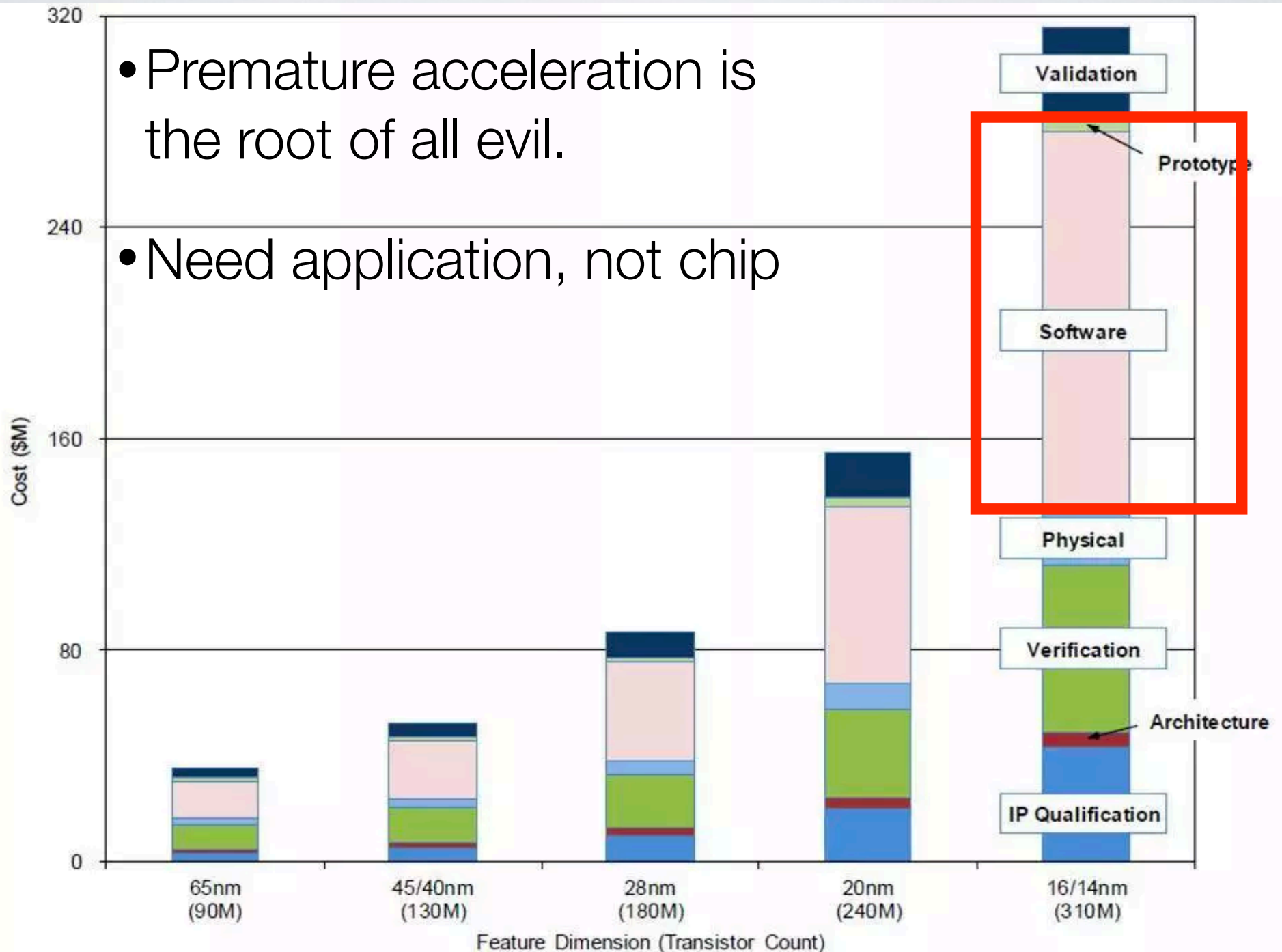


Many researchers eliminated layers entirely



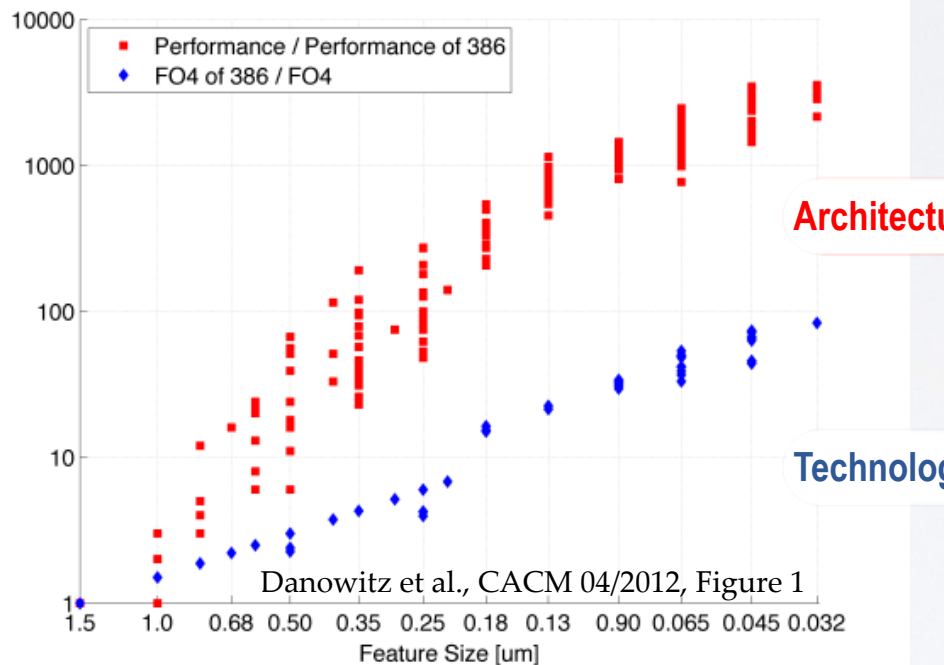
Lesson 1: Its all about the software

- Premature acceleration is the root of all evil.
- Need application, not chip



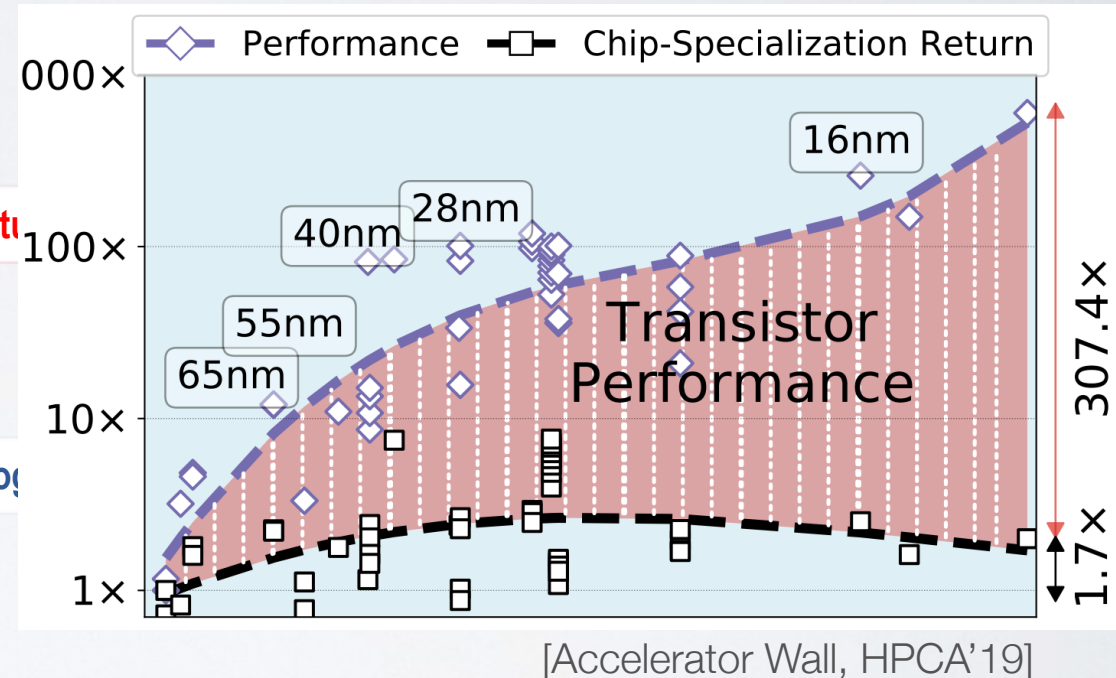
Lesson 2: We are just getting started

Arch innovation Pre-Moore



~50% every two years

Arch innovation Post-Moore



~20% every two years
after initial gain

Goal 1: Always keep the software-to-hardware flow

Goal 2: How do we lower the barrier to specialization?

Goal 3: Enable iterative HW exploration

Goal 4: Full Application (multiple IPs)

Problem

Algorithm

Application

Middleware / Compiler

Operating System

Microarchitecture

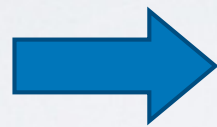
Logic Design

Transistors, etc.

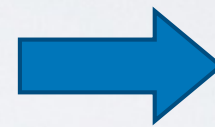
**Duh! Don't ISAs
do that?**

ISA-based Design

Programming model



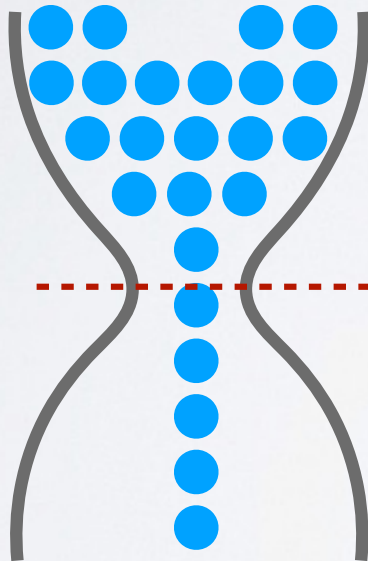
ISA



Hardware



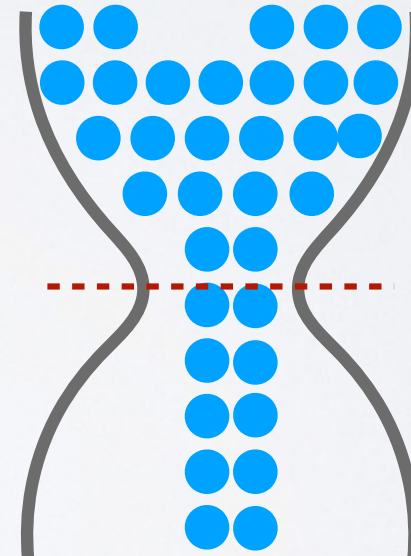
Software \rightarrow C/C++



ISA

Hardware — Gen 1

Software \rightarrow C/C++



ISA w/
SIMD

Hardware — Gen 2

ISA-based Design



Defines only “what” - operational semantics.

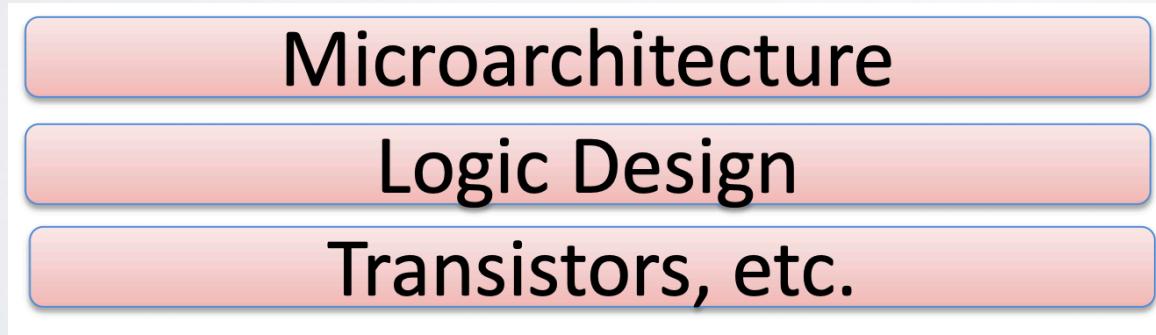
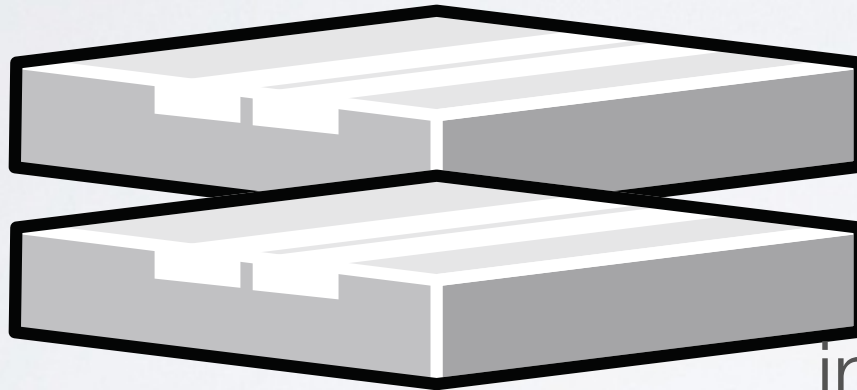
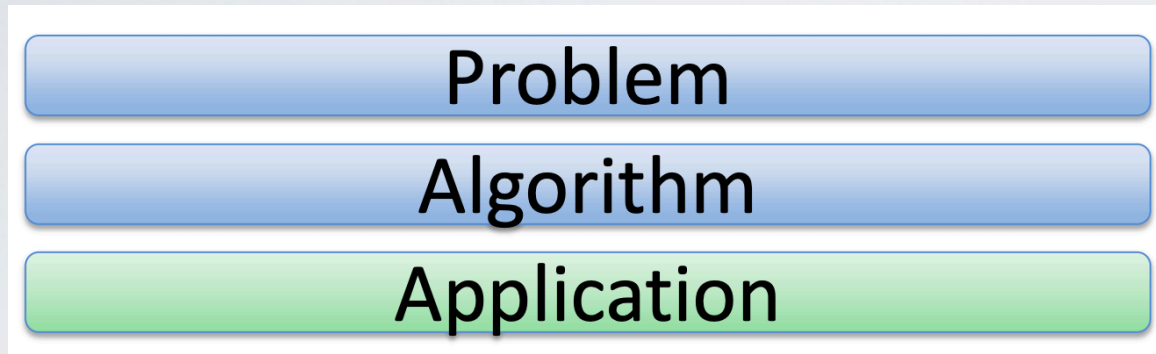
Not precise enough to create hardware

May not capture hints from the domain

Hardware Microbrewery ?



The MicroArchitectural IR (μ IR) Flow

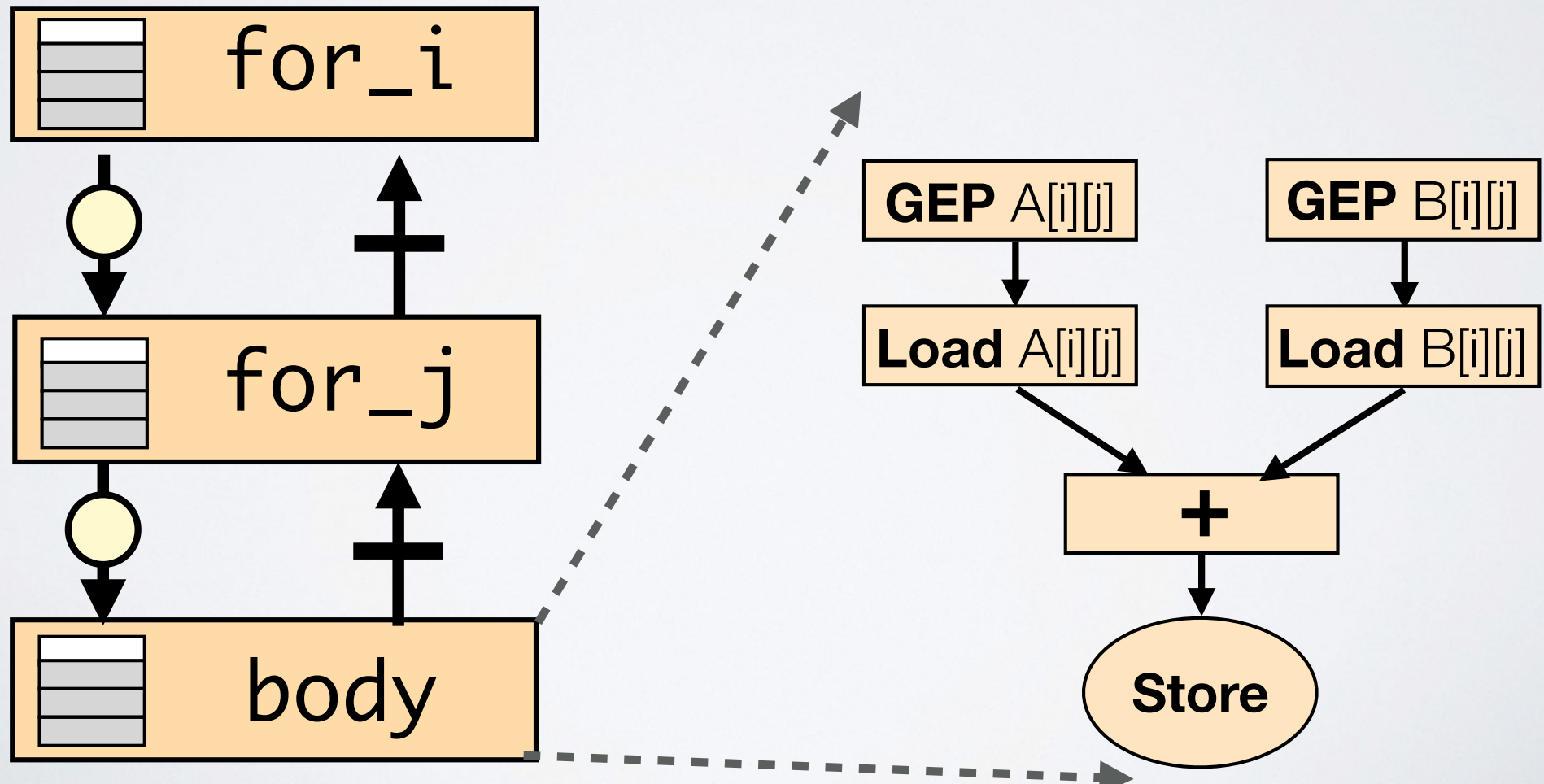


Extensible
software
and hardware
intermediate representation

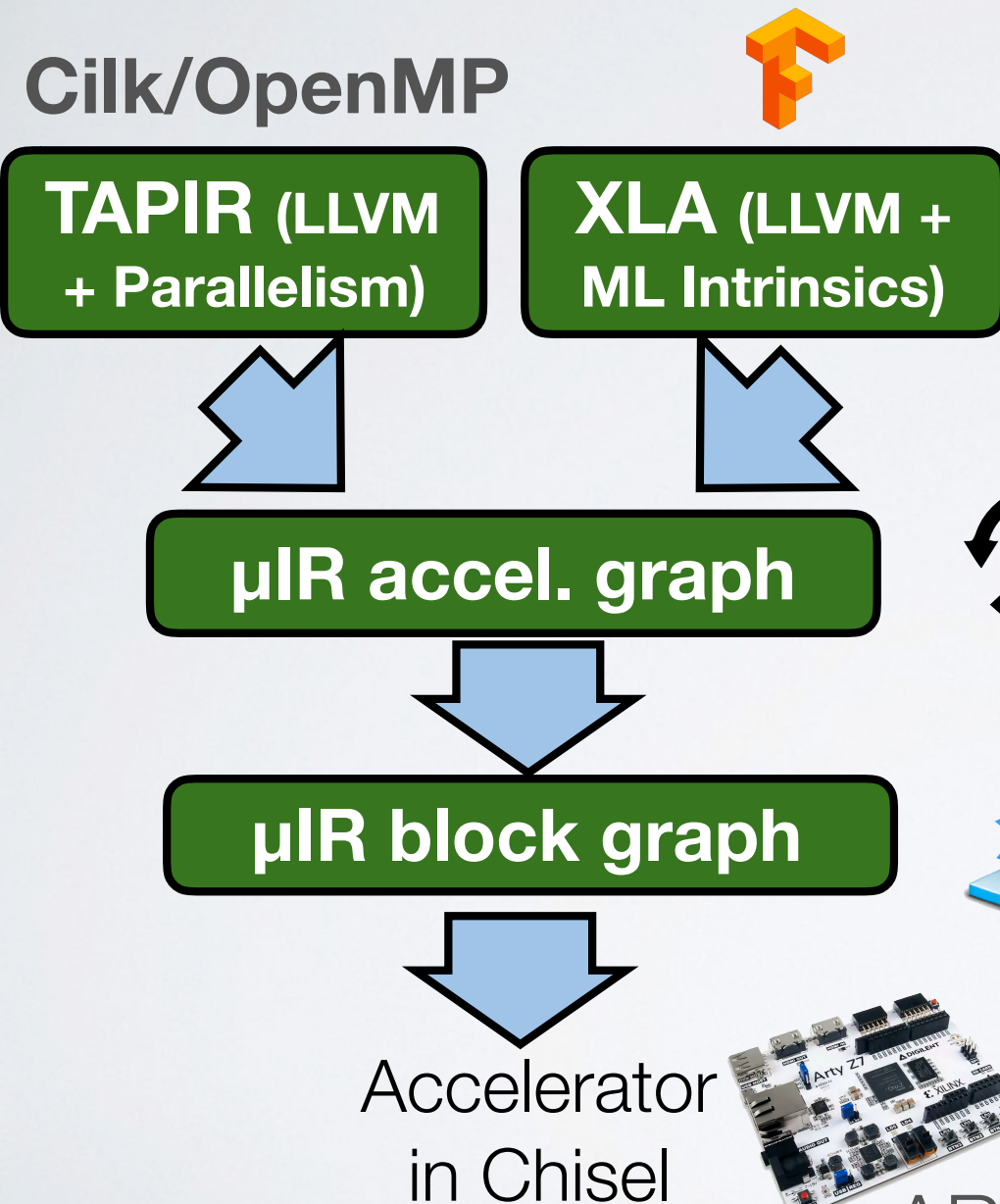
μIR on 1 slide

```
parallel_for(i = 0 until n)  
  parallel_for(j = 0 until n)  
    c[i][j] = a[i][j] + b[i][j];
```

Hierarchical Data + Control Dynamic Graph



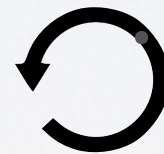
MicroArchitectural IR (μ IR)



- Target: Spatial Hardware (FPGAs today...)

- High-level-synthesis

- match-up multiple DSL IRs



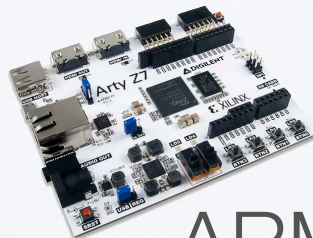
- Design HW iteratively

- changes to graph create optimized HW



- Reusable hardware library

- create different accelerators with common structures



ARM/RISC-V SoC

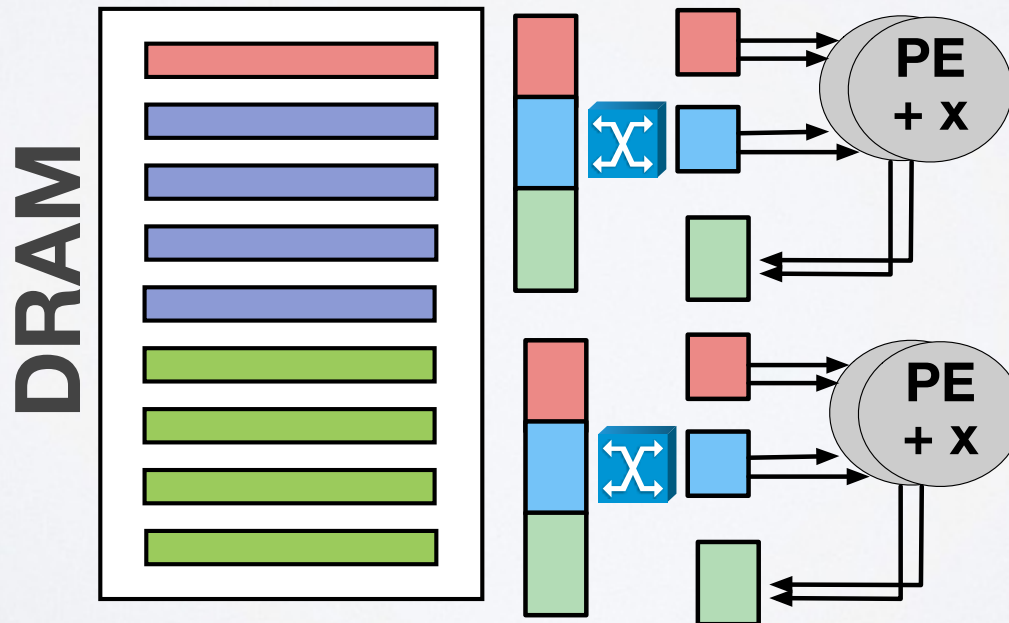
μ IR Goal

- Input: behavior in software

$$\begin{array}{c} \text{weights} \\ \text{red} \end{array} \times \begin{array}{c} \text{input} \\ \text{blue} \end{array} = \begin{array}{c} \text{Output} \\ \text{green} \end{array}$$

```
cilk_for(i=0; I<(M-W); i++)  
  cilk_for(j=0; j<W; j++)  
    output[i] += input[i+j] * weight[j];
```

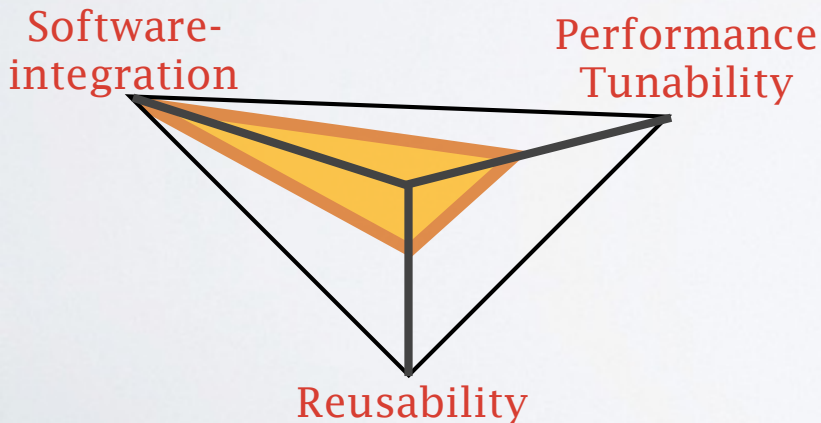
- Output: Modular hardware description



μIR vs Other Approaches

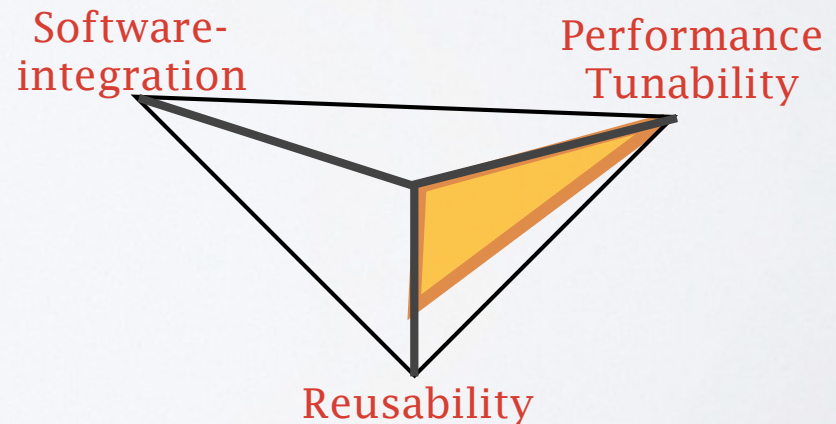
C-to-RTL

```
for(i=0; i<(M-W); i++)  
#pragma HLS LOOP_TRIPCOUNT  
min = c_low max = c_high  
    for(j=0; j<W; j++)  
#pragma HLS PIPELINE II = 1  
#pragma LOOP_UNROLL 2  
    output[i] += input[i+j]*  
                weight[j];
```



Chisel, Spatial

```
class Conv1D[T <: Data] {  
    val io = new Bundle {  
        val in = Input(Valid(dtype))  
        val out = Output(Valid(dtype))  
    }  
    // State machine  
    // Stream interfaces  
}
```

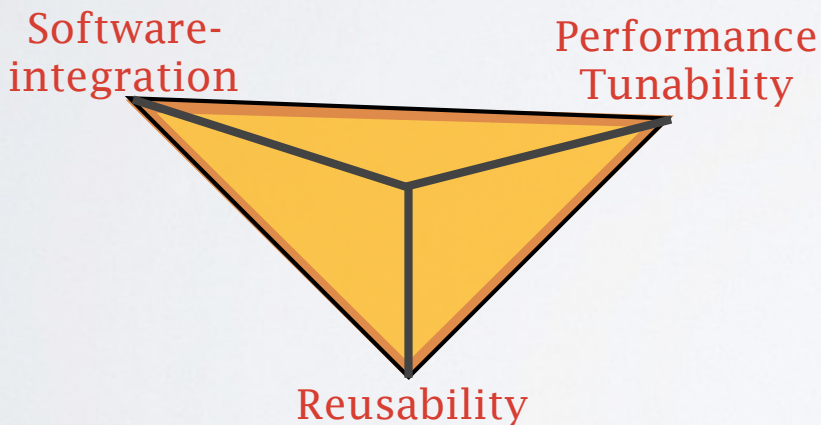


μIR Design Flow

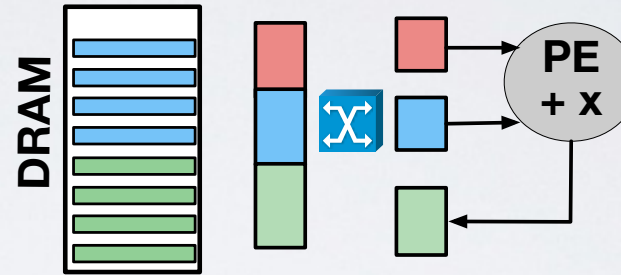
1D Convolution

weights \times input \rightarrow output

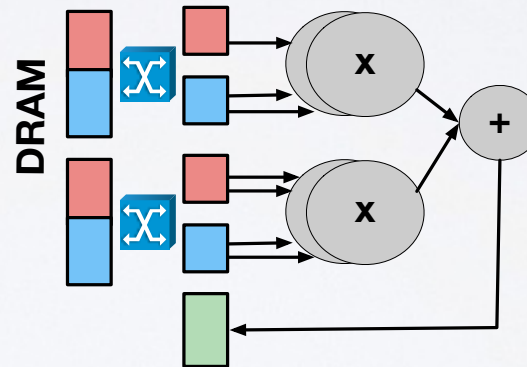
```
cilk_for (i=0; i < (M-W); i++) {  
  cilk_reduce (j = 0; j < W; j++) {  
    output[i] += input[i+j]*  
                  weight[j];  
  }  
}
```



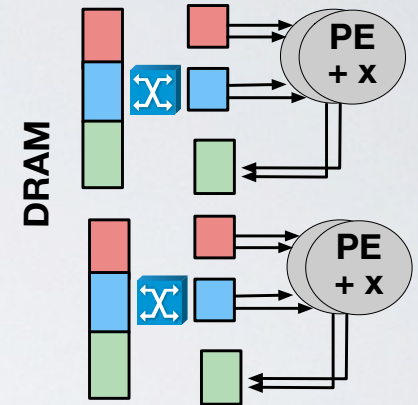
1: Multi-level Buffers



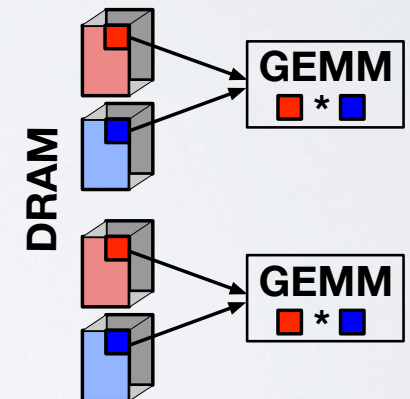
2: Pipeline



3: Parallelism



4: Domain-specific



- **Backend target**
 - currently FPGAs, but really any spatial architecture

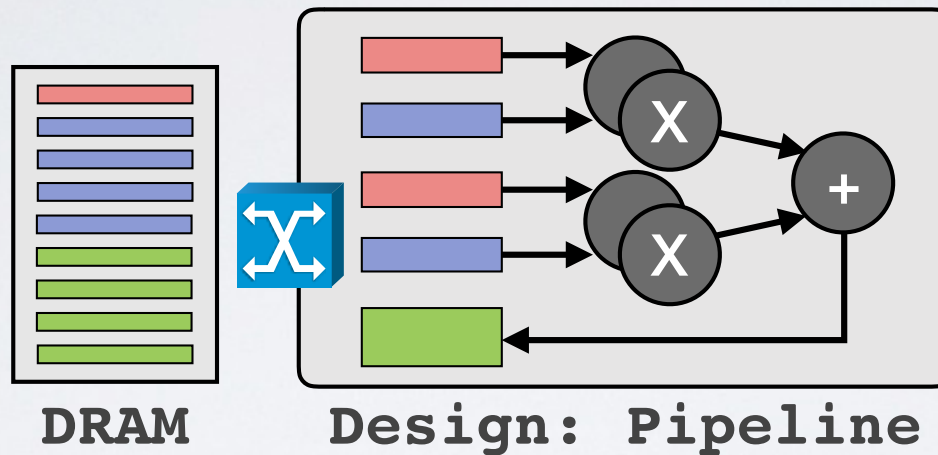
μIR Features

- **Feature 1 : One system for creating hardware and mapping to it.**
 - common information across software lang. and hardware
e.g., data type of ops, spatial data flow,
- **Feature 2: End-to-End System**
- **Feature 3: Automate common passes**
 - Different accelerators can use common optimizations

μIR Features

- **Feature 4: Transformable or iterative designs**
 - HW design evolution and experimentation is automatable
- **Feature 5: Multi-stage generator**
 - inclusion of application and hardware experts
 - Separation of EDA concerns from Architecture and Algorithm

μIR Theme 1: Spatial : hierarchical, heterogeneous and latency-agnostic



- Can include memory ops in the dataflow
 - Multiple memory models (caches, scratchpad, streaming)
- Exposes hardware memory tradeoffs (e.g., FIFOs, RAMs, Shared RAM)
- Hides timing-specific schedule (similar to compiler IRs)

μIR Theme 2: Tasks and Dynamic Schedules in HW

```
cilk_for(i=0 to n) {  
  if(node[i].valid){  
    compute(&node[i]);  
  }  
}
```

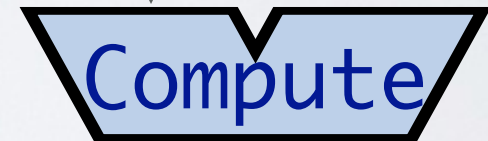
Concurrency control

for_i : if (valid)

Spawn



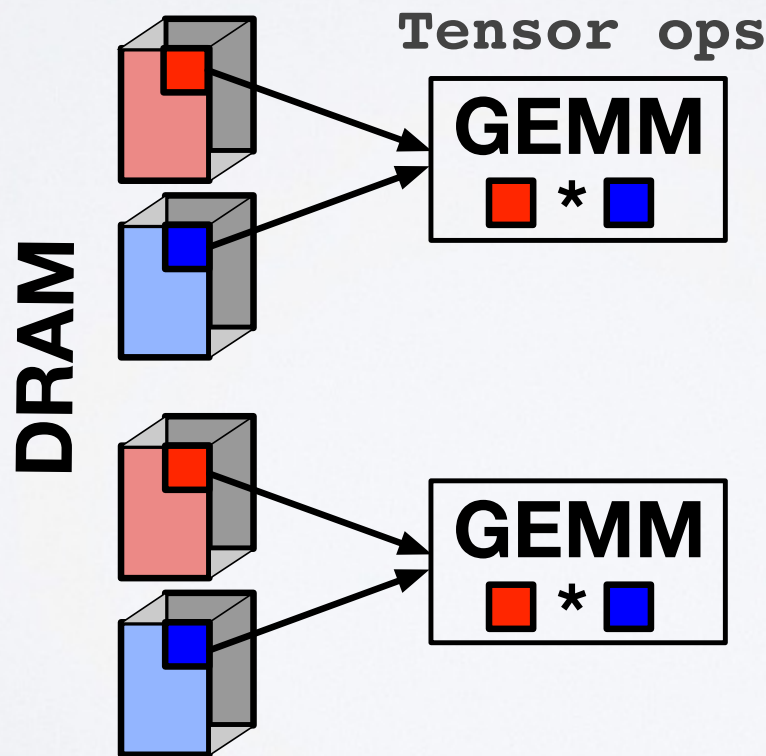
Sync



- Need not statically schedule operations in IR
 - hardware adapts to non-determinism in memory system
- Improves software productivity (can represent rich software patterns. e.g., recursion)

μIR Theme 3: Multi-level IR and Generic Dataflow

- Accelerator functionality
 - What ops ? How to create new ops?
- Specializing functionality while keeping system design generic
 - create generic data flow on different op types.
- Extensible IR
 - introduce new ops based on domain and integrate with system



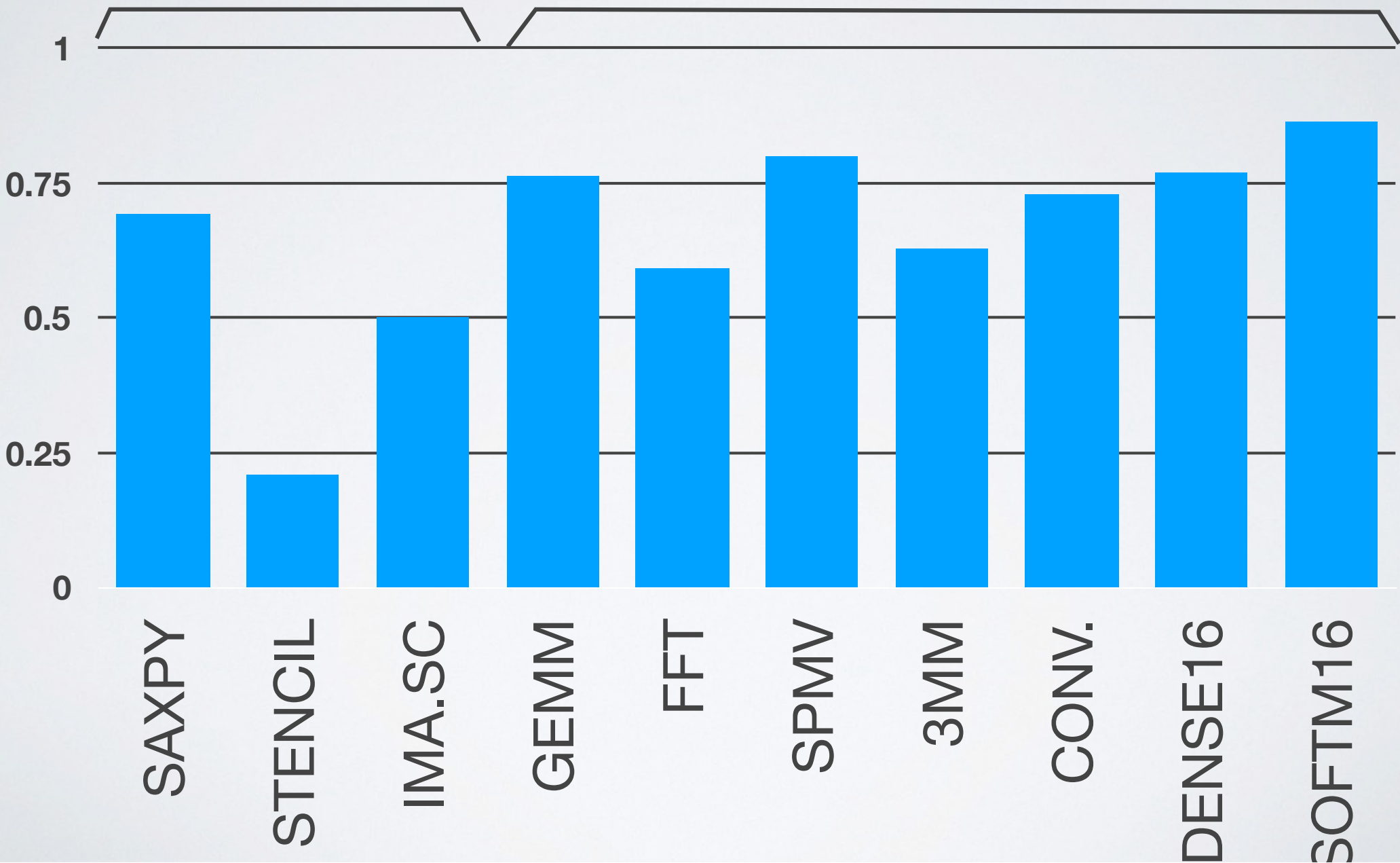
μIR Research Topics

- Multi-level: How many IRs is one too many?
- How to treat memory ?
- Bad software => Bad hardware
- How to iteratively optimize across layers ?
- Generic μIR (our work) vs DSL μIR (e.g., Stanford CoreIR)
- Integration with backend EDA tools (ASIC vs FPGA vs CGRA)

μIR Iterative Optimizations

Banking, Op-Fusion, Tile

Banking, Localization, Op-Fusion



μIR Productivity

μIR

FIRRTL

Optimizations	∂ Nodes	∂ Edges	∂ Nodes	∂ Edges
Increase Task Execution Tiles	4	16	80	120
Add Hierarchical Buffers	6	18	30	70
Pipeline Dataflow	12	8	25	20

Questions

<https://github.com/sfu-arch>

Snehasish Kumar, Dan Lin, Amirali Sharifian (job market),
Naveen Vedula (job market), Steven Margerm, Apala Guha,
Parmida Vahdanthia, Reza Shojabro



NSERC
CRSNG



®

Semiconductor
Research
Corporation

