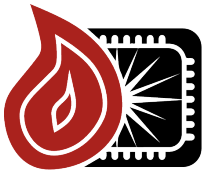
# Tutorial Outline

| Time | Topic |
|---|---|
| 8:30 am – 9:00 am | Introduction |
| **9:00 am – 10:00 am** | **Pre-RTL Simulation Framework: Aladdin** |
| 10:00 am – 10:30 am | Break |
| 10:30 am – 11:00 am | Workload Characterization Tool: WIICA |
| 11:00 am – 12:00 pm | CAD & Benchmarks: HLS & MachSuite |
| 12:00 pm – 2:00 pm | Lunch |
| 2:00 pm – 3:00 pm | Embedded Keynote Talk: Mark Horowitz (Stanford) |
| 3:00 pm – 3:30 pm | Accelerator Selection Tool: Sigil |
| 3:30 pm – 4:00 pm | Break |
| 4:00 pm – 5:00 pm | Hands-on Exercise |



HARVARD UNIVERSITY



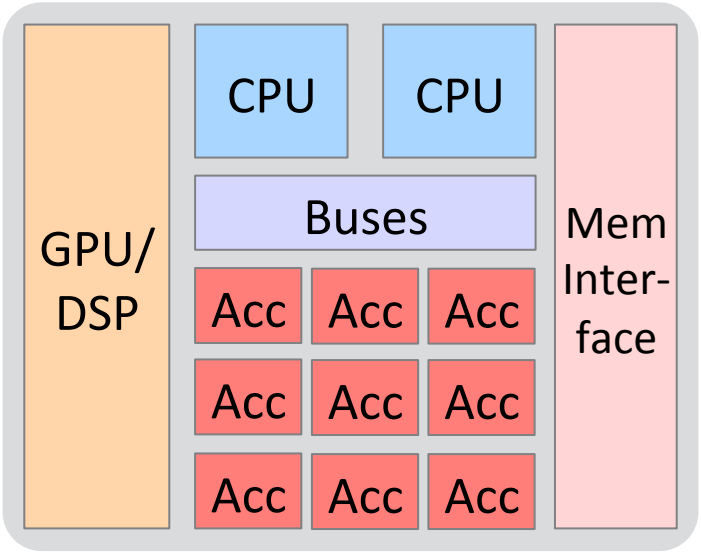
# ALADDIN

## A Pre-RTL, Power-Performance Accelerator Simulator

Enabling Large Design Space Exploration of Customized Architectures

Yakun Sophia Shao, Brandon Reagen,
Gu-Yeon Wei, David Brooks
Harvard University

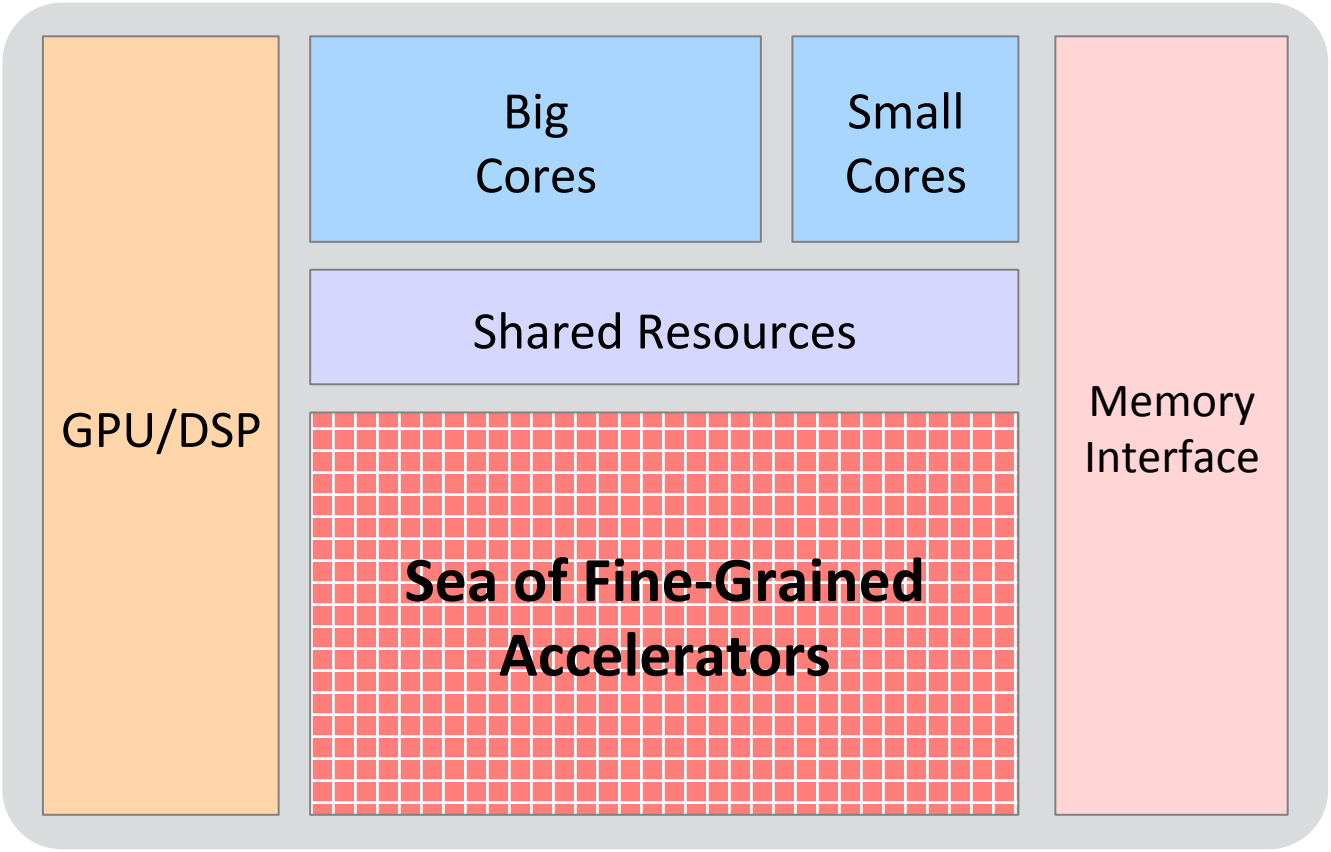
# Today's SoC






HARVARD
UNIVERSITY

# Future Accelerator-Centric Architectures

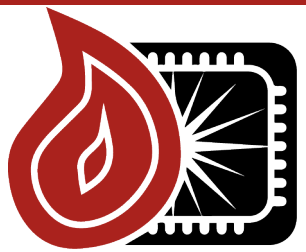



How to decompose an application to accelerators?
How to rapidly design lots of accelerators?
How to design and manage the shared resources?

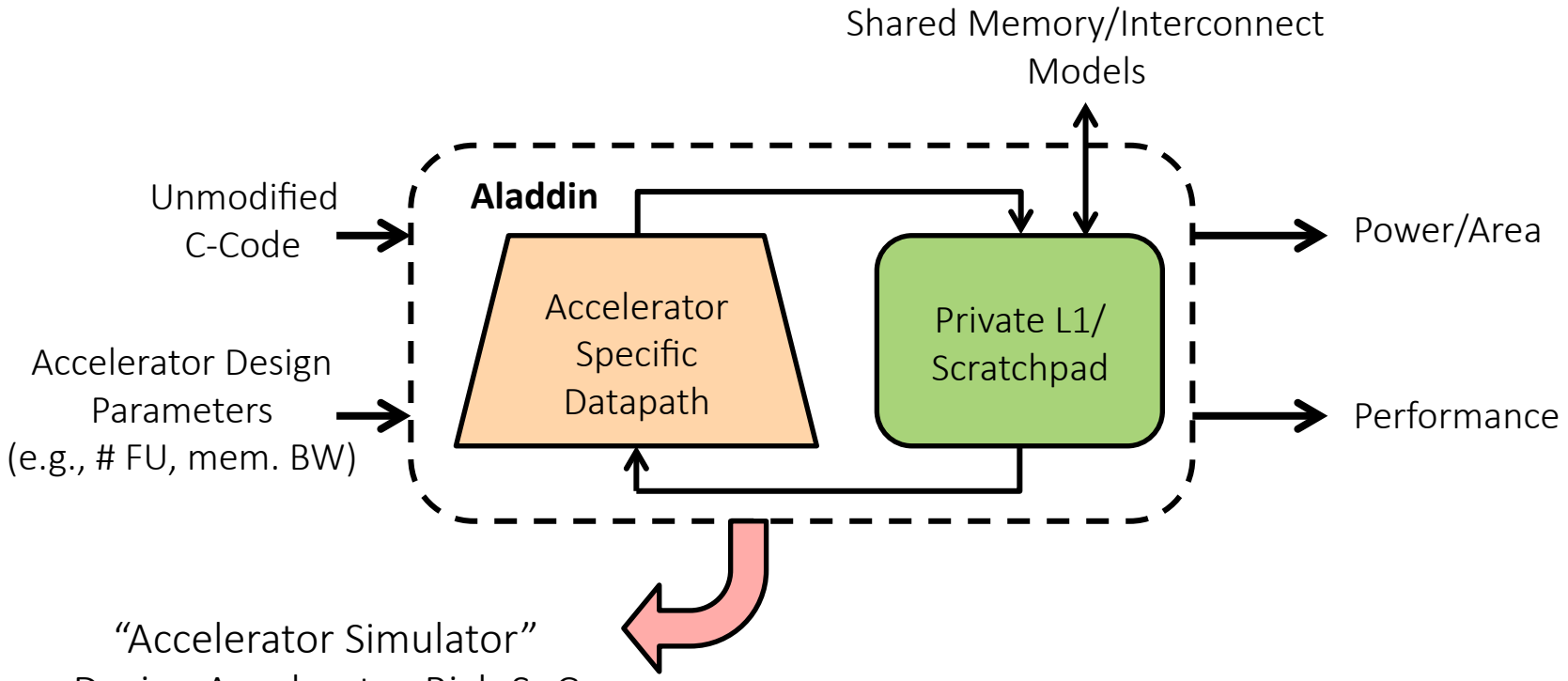
↑ Flexibility
↓ Design Cost
↑ Programmability



HARVARD UNIVERSITY

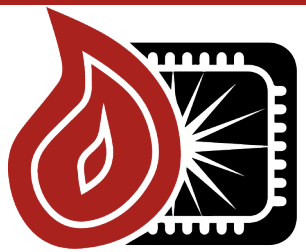


# Aladdin: A pre-RTL, Power-Performance Accelerator Simulator

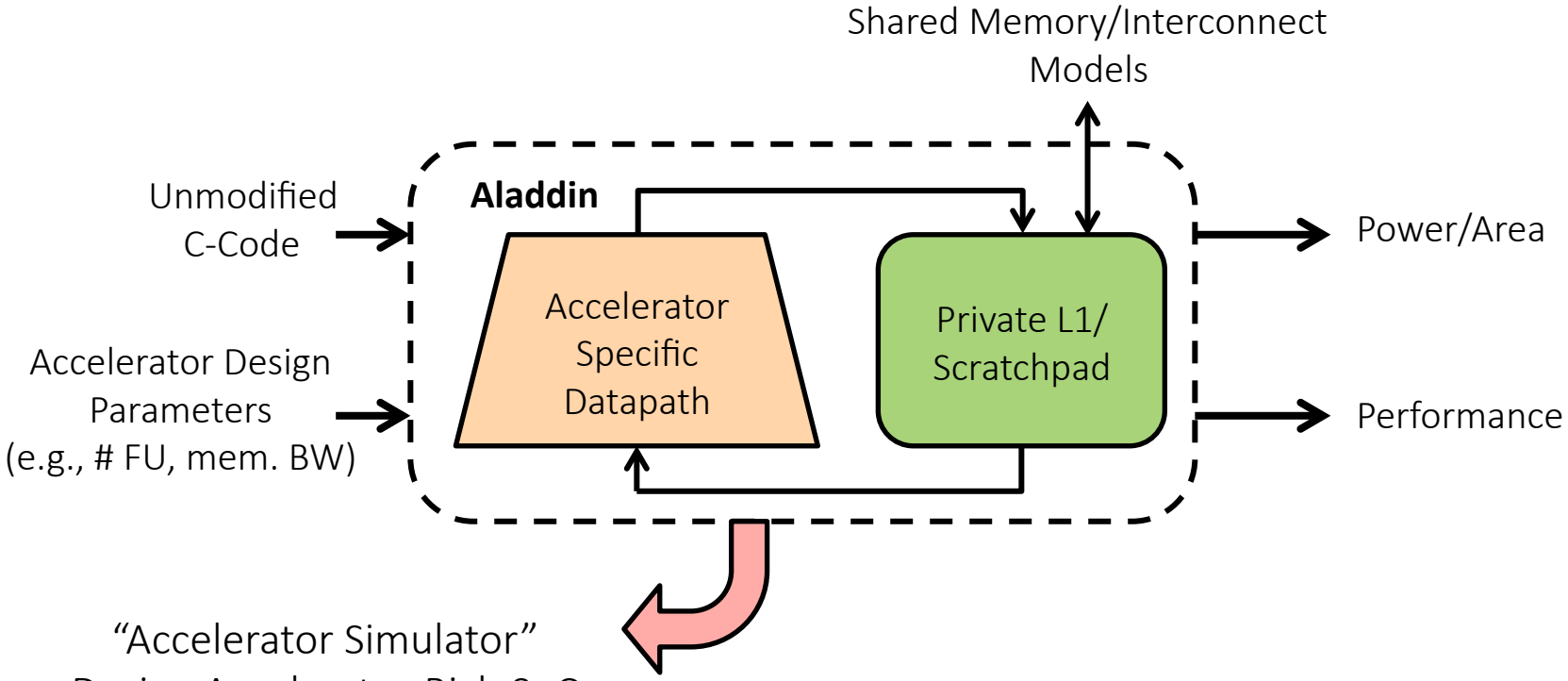





HARVARD UNIVERSITY



# Aladdin: A pre-RTL, Power-Performance Accelerator Simulator

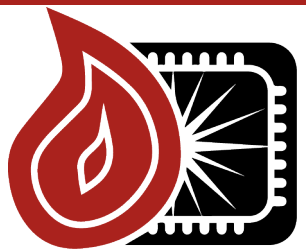



"Accelerator Simulator"
Design Accelerator-Rich SoC Fabrics and Memory Systems

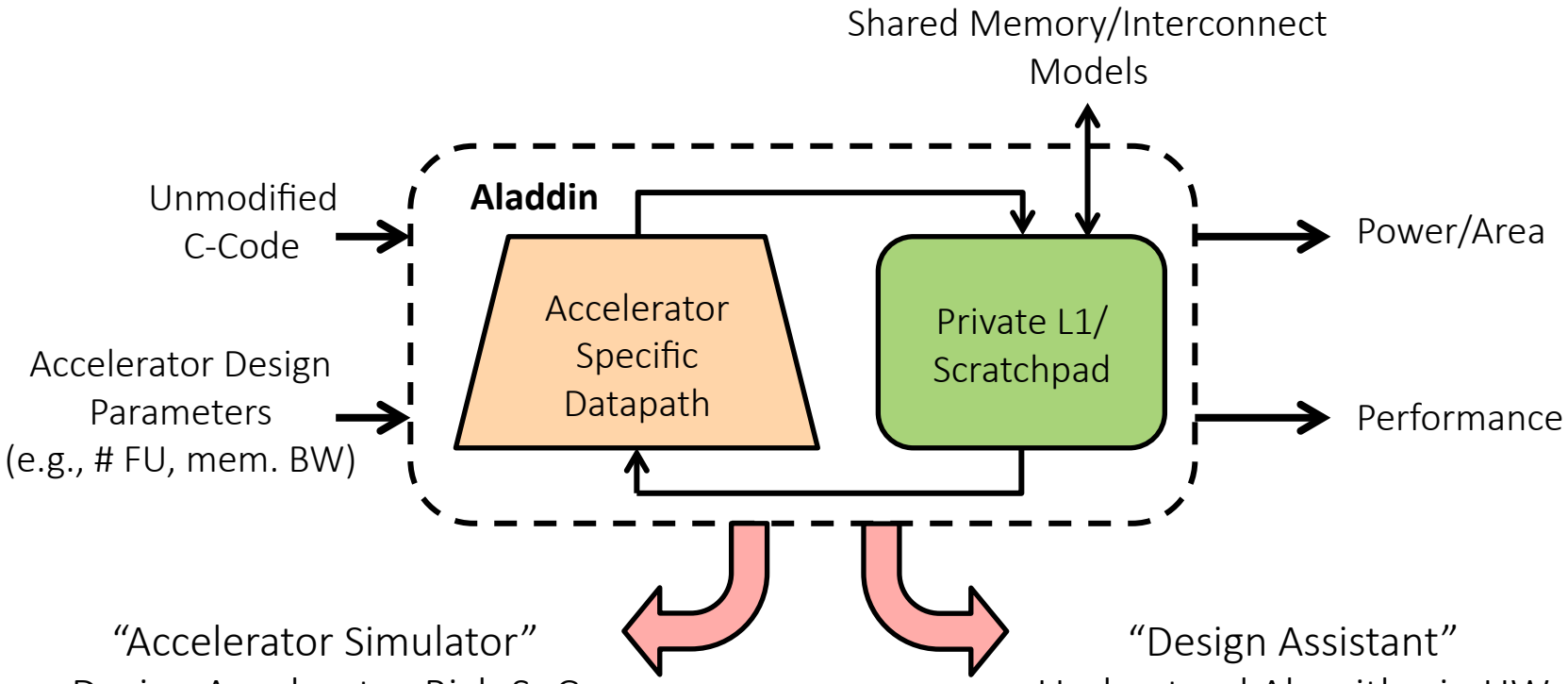
⬆ **Flexibility**
⬆ **Programmability**



HARVARD UNIVERSITY



# Aladdin: A pre-RTL, Power-Performance Accelerator Simulator




"Accelerator Simulator"
Design Accelerator-Rich SoC Fabrics and Memory Systems

⬆ **Flexibility**
⬆ **Programmability**

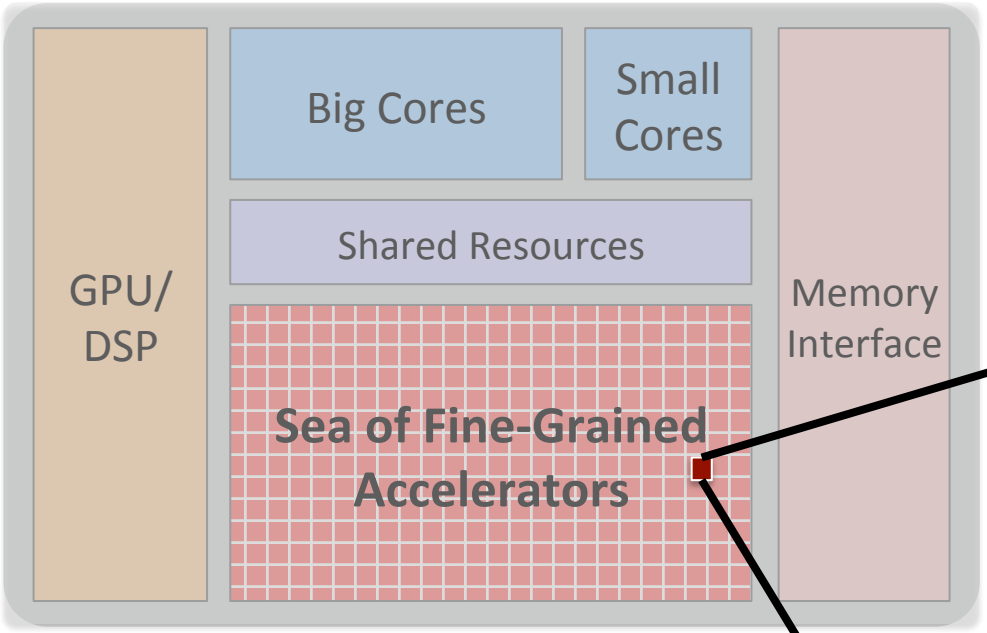
"Design Assistant"
Understand Algorithmic-HW Design Space before RTL

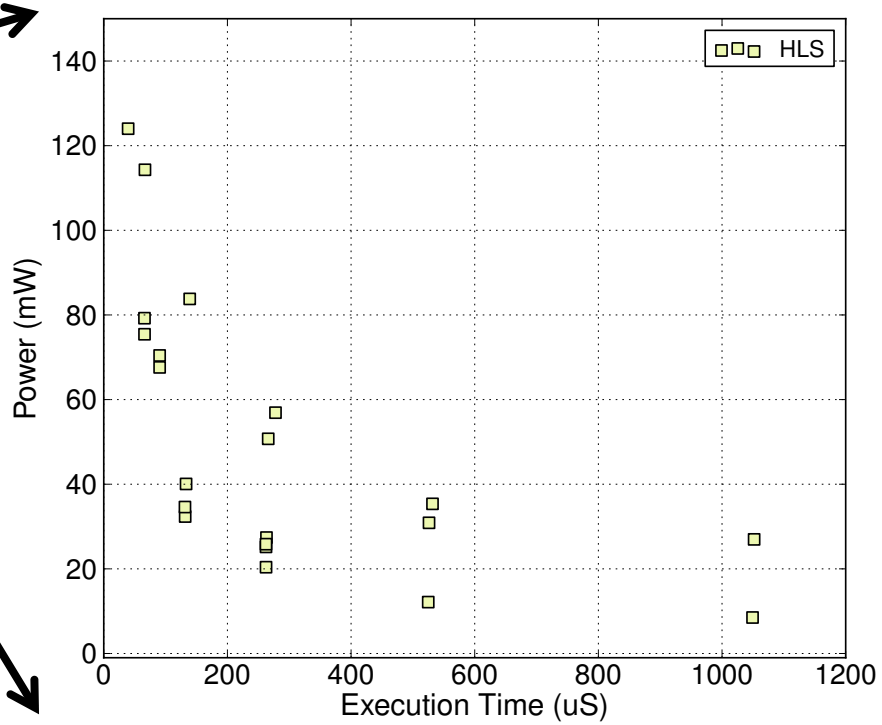
⬇ **Design Cost**



HARVARD UNIVERSITY

# Future Accelerator-Centric Architecture

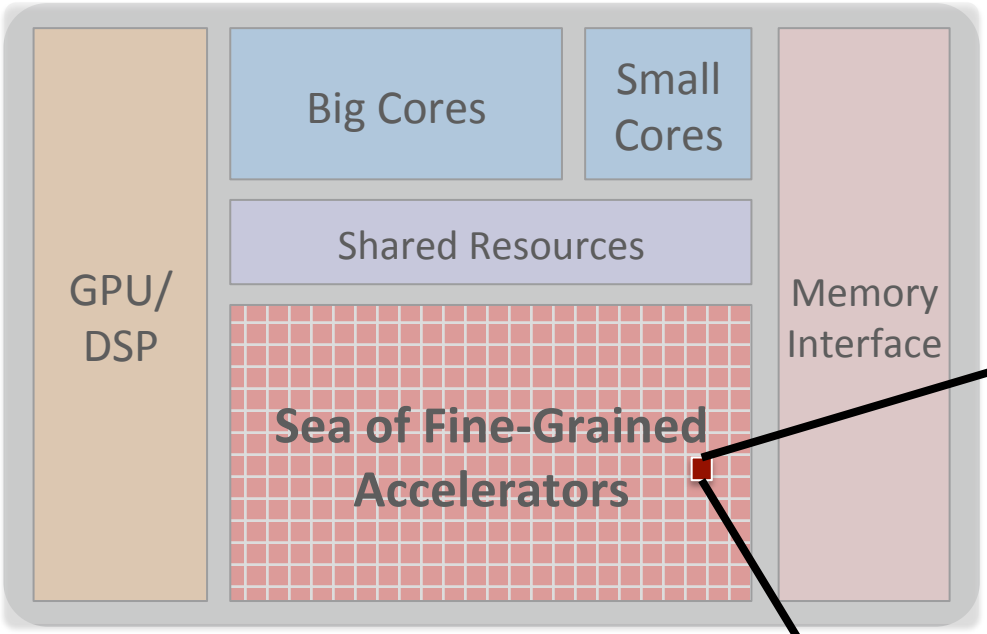


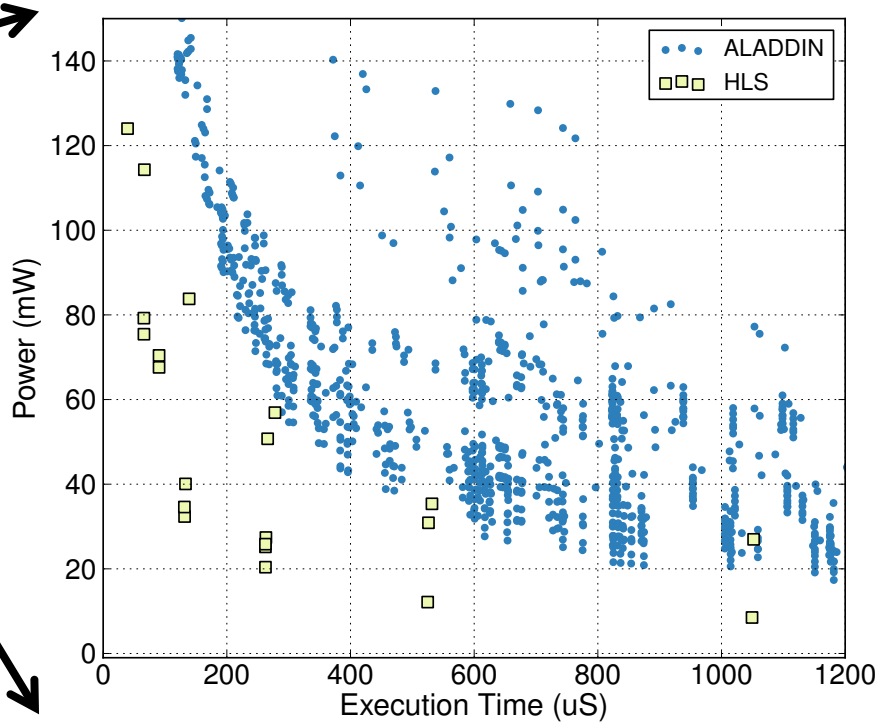






HARVARD
UNIVERSITY

# Future Accelerator-Centric Architecture



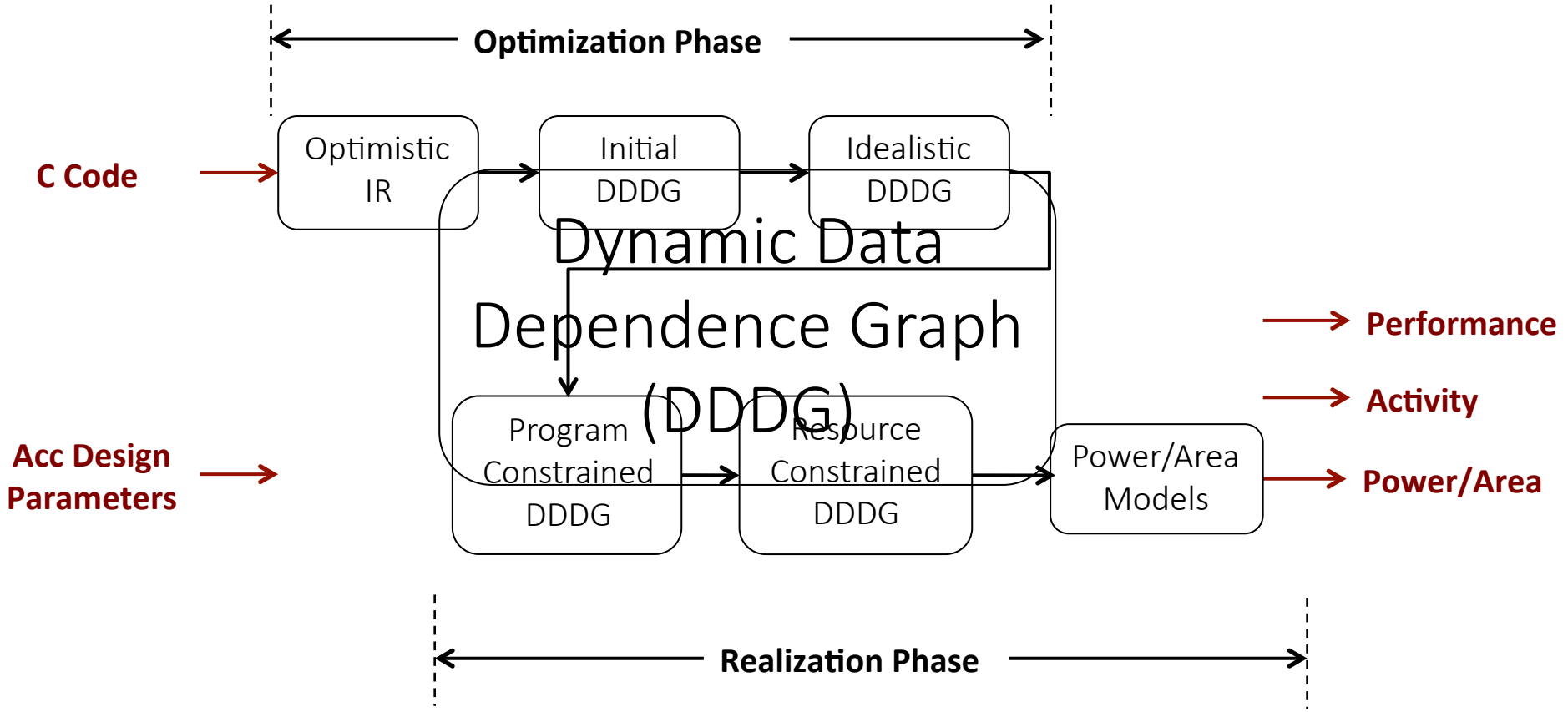




**Aladdin** can **rapidly** evaluate **large** design space of accelerator-centric architectures.



HARVARD UNIVERSITY

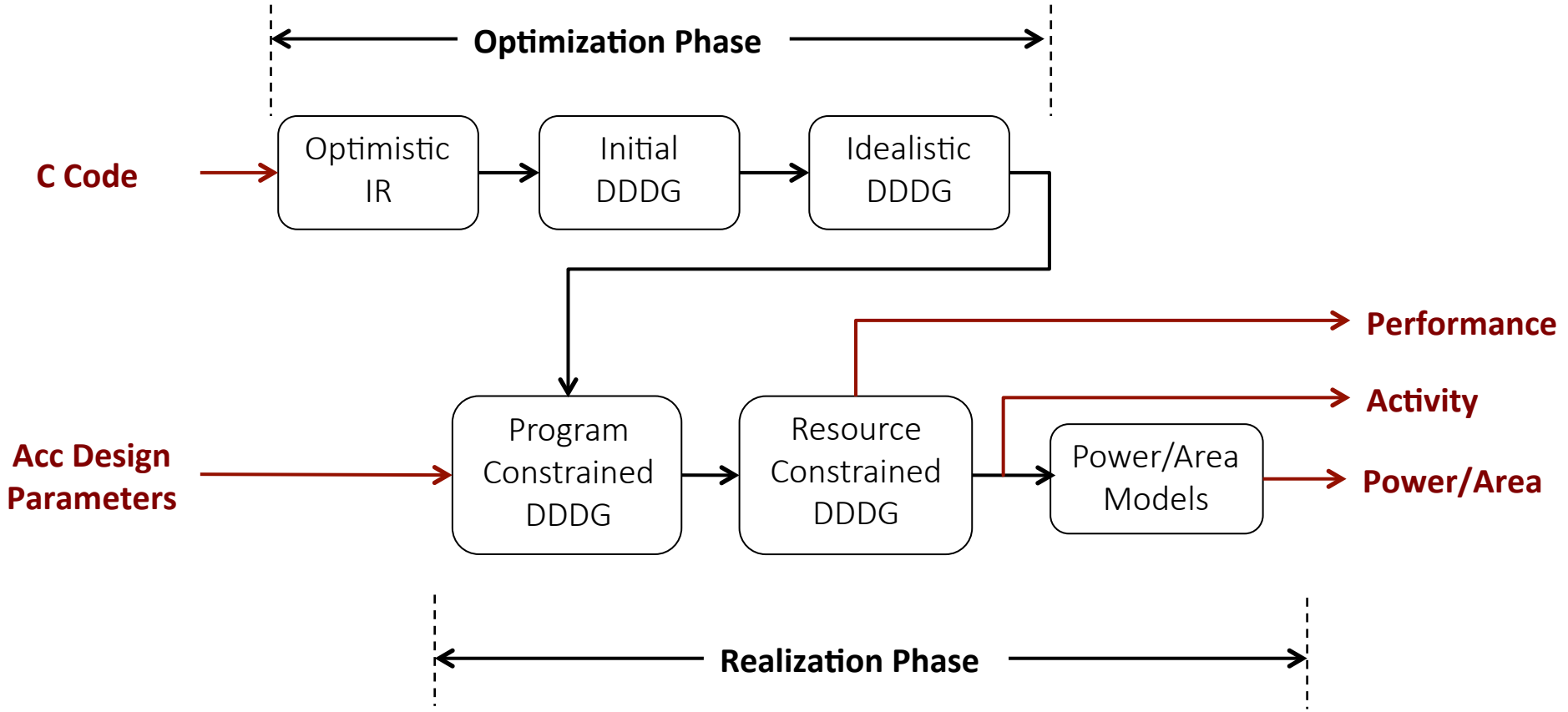
# Aladdin Overview






HARVARD UNIVERSITY

# Aladdin Overview






HARVARD UNIVERSITY

# Aladdin is NOT

- An HLS flow:
  - No RTL is generated.
  - High-level *estimates* of power and performance;
  - Aladdin uses fully dynamic analysis to expose algorithmic parallelism for unmodified HLL codes;
- Limit of ILP study:
  - "optimistic but realistic" DDDG is constructed to model accelerators.



HARVARD UNIVERSITY

# From C to Design Space

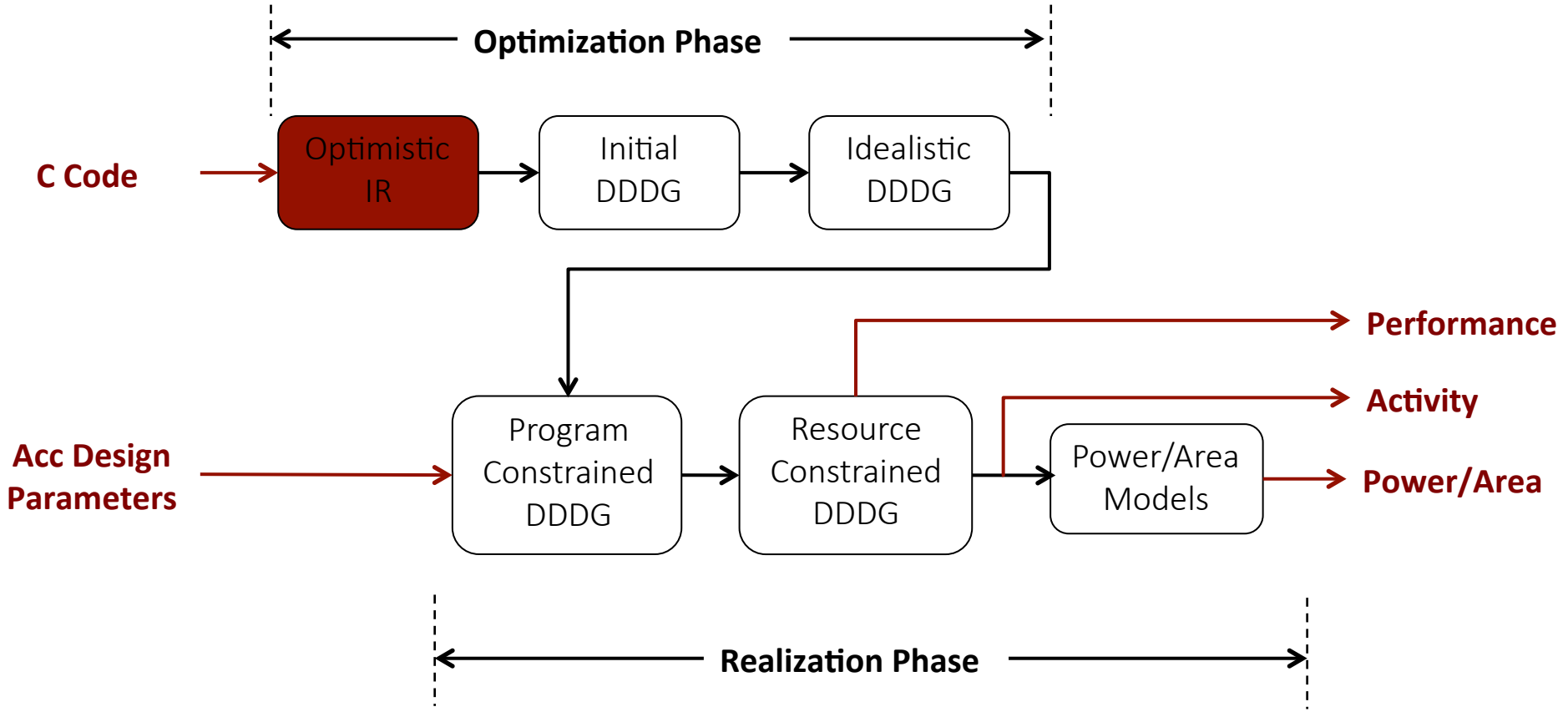
C Code:

```
for(i=0; i<N; ++i)
  c[i] = a[i] + b[i];
```



HARVARD UNIVERSITY

# Aladdin Overview



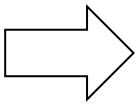



HARVARD UNIVERSITY

# IR Dynamic Trace

C Code:

```
for(i=0; i<N; ++i)
  c[i] = a[i] + b[i];
```



```
0.  r0=0  //i = 0
1.   r4=load (r0 + r1) //load a[i]
2.   r5=load (r0 + r2) //load b[i]
3.   r6=r4 + r5
4.   store(r0 + r3, r6) //store c[i]
5.   r0=r0 + 1  //++i
6.   r4=load(r0 + r1) //load a[i]
7.   r5=load(r0 + r2) //load b[i]
8.   r6=r4 + r5
9.   store(r0 + r3, r6) //store c[i]
10.  r0 = r0 + 1  //++i
...
```



HARVARD UNIVERSITY

# Optimistic IR

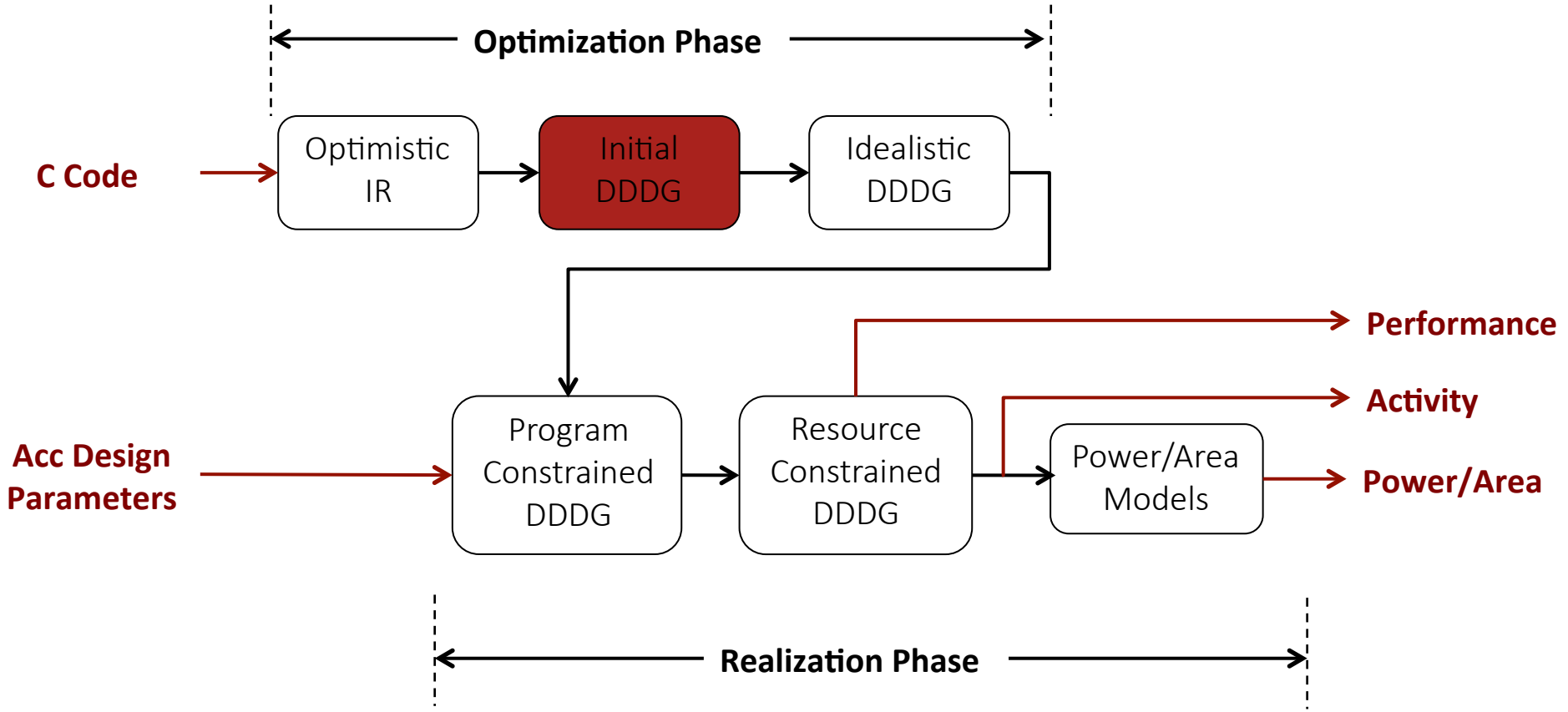
- LLVM IR
- High-level IR:
  - Machine- and ISA-independent
- Features:
  - Unlimited Registers
  - Simple Opcodes: add, mul, sin, sqrt
  - Only load/store access memory

*Shao, et al., ISA-Independent Workload Characterization and Implications for Specialized Architecture, ISPASS, 2013*



HARVARD UNIVERSITY

# Aladdin Overview






HARVARD UNIVERSITY

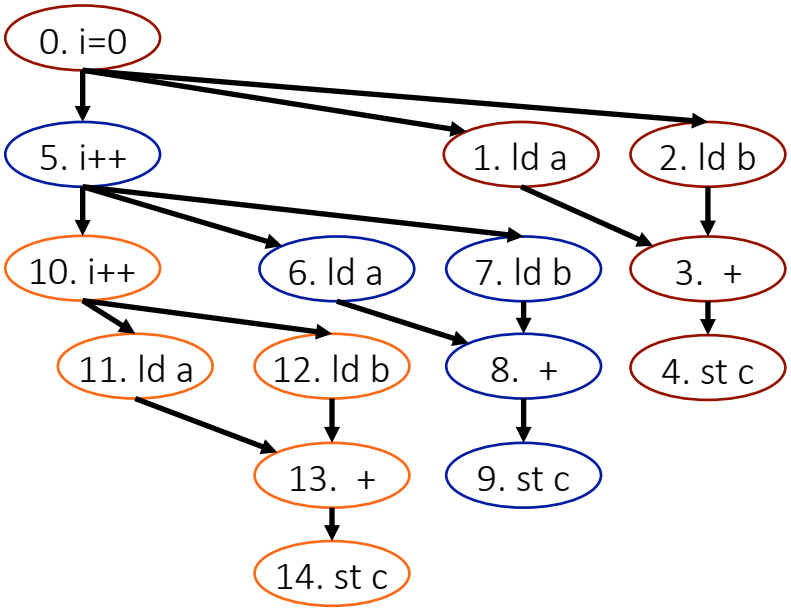
# Initial DDDG

C Code:

```
for(i=0; i<N; ++i)
  c[i] = a[i] + b[i];
```

IR Trace:

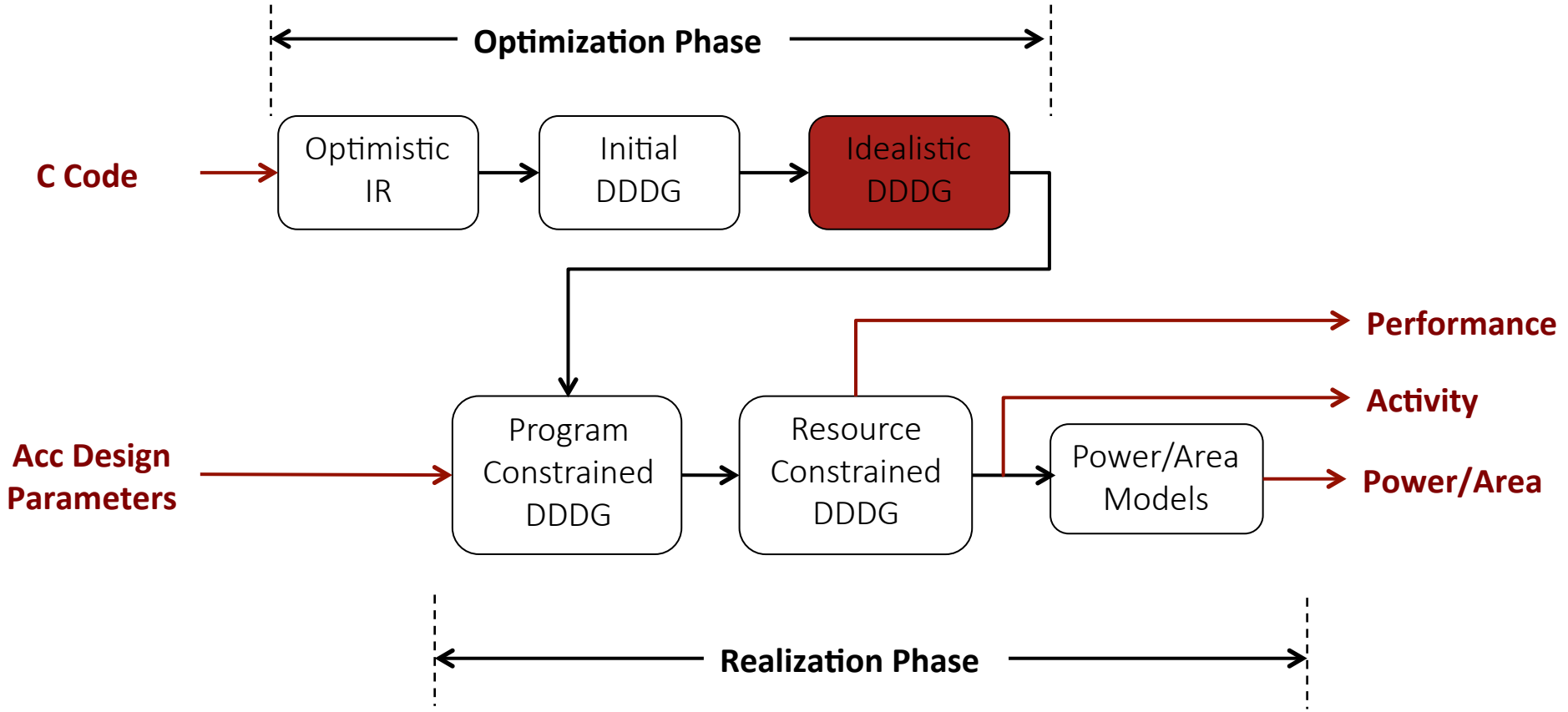
```
0.  r0=0  //i = 0
1.  r4=load (r0 + r1) //load a[i]
2.  r5=load (r0 + r2) //load b[i]
3.  r6=r4 + r5
4.  store(r0 + r3, r6) //store c[i]
5.  r0=r0 + 1  //++i
6.  r4=load(r0 + r1) //load a[i]
7.  r5=load(r0 + r2) //load b[i]
8.  r6=r4 + r5
9.  store(r0 + r3, r6) //store c[i]
10.r0 = r0 + 1  //++i
...
```






HARVARD UNIVERSITY

# Aladdin Overview



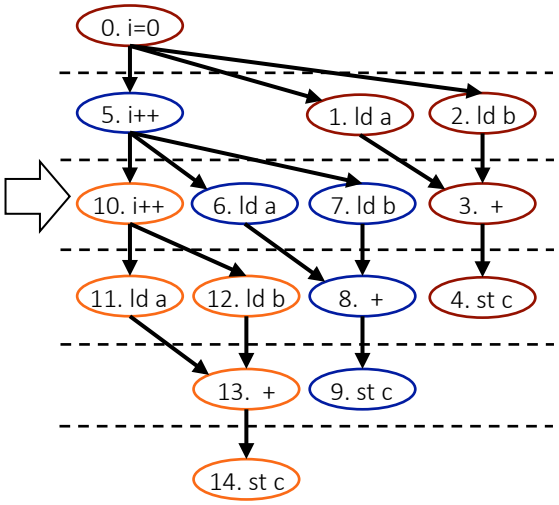


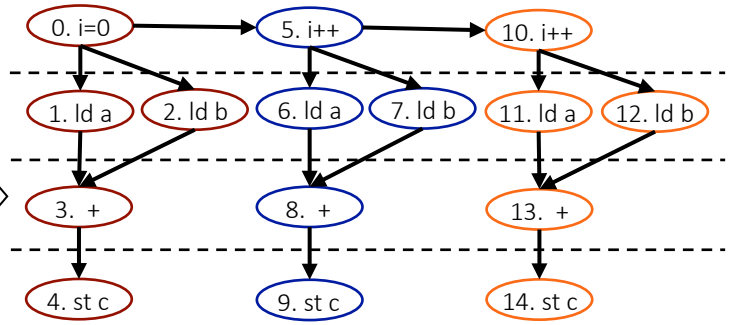

HARVARD UNIVERSITY

# Idealistic DDDG

C Code:
```
for(i=0; i<N; ++i)
  c[i] = a[i] + b[i];
```

IR Trace:
```
0.  r0=0  //i = 0
1.  r4=load (r0 + r1) //load a[i]
2.  r5=load (r0 + r2) //load b[i]
3.  r6=r4 + r5
4.  store(r0 + r3, r6) //store c[i]
5.  r0=r0 + 1  //++i
6.  r4=load(r0 + r1) //load a[i]
7.  r5=load(r0 + r2) //load b[i]
8.  r6=r4 + r5
9.  store(r0 + r3, r6) //store c[i]
10.r0 = r0 + 1  //++i
...
```






HARVARD UNIVERSITY

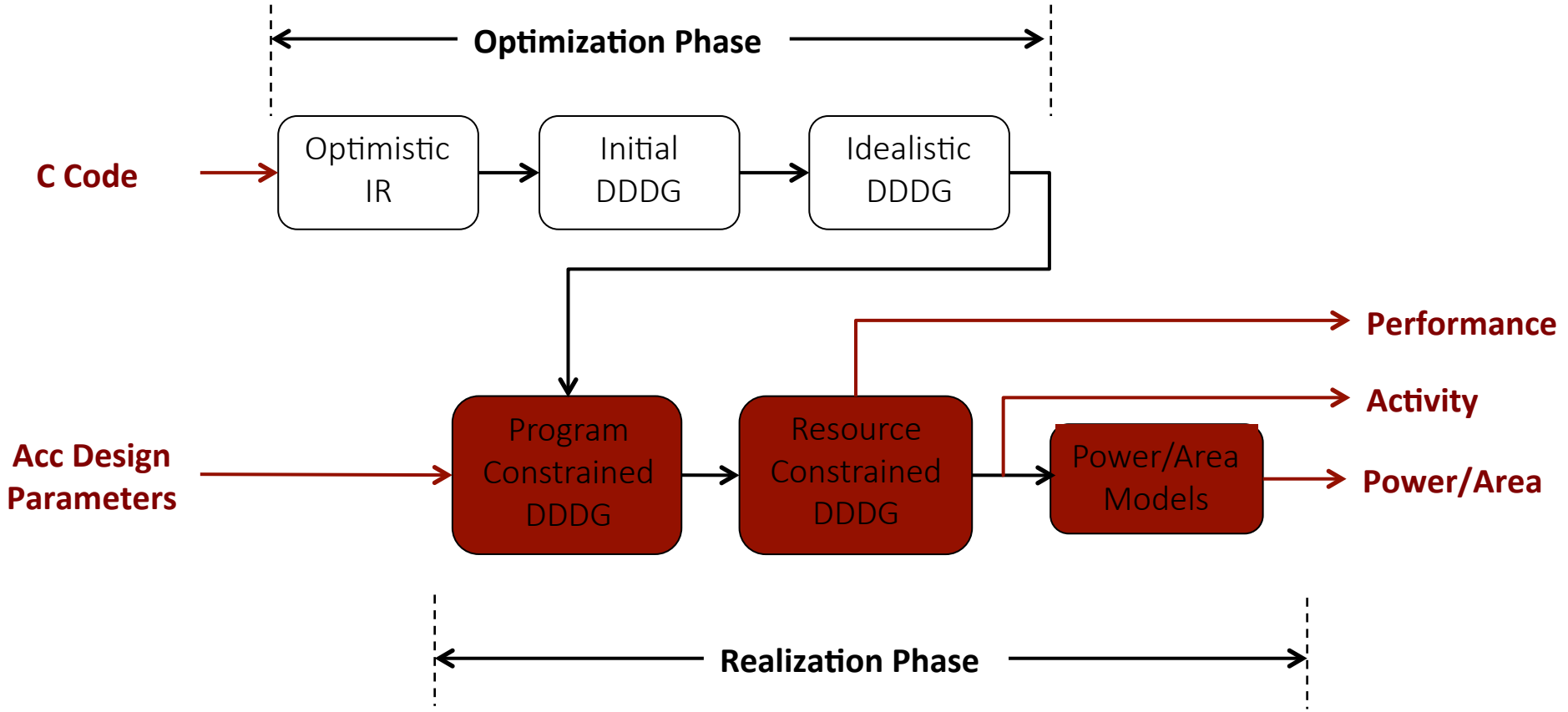
# Idealistic DDDG

- Include application-specific customization strategies.
- Node-Level:
  - Bit-width Analysis
  - Strength Reduction
  - Tree-height Reduction
- Loop-Level:
  - Remove dependences between loop index variables
- Memory Optimization:
  - Memory-to-Register Conversion
  - Store-Load Forwarding
  - Store Buffer
- Extensible
  - e.g. Model CAM accelerator by matching nodes in DDDG



HARVARD UNIVERSITY

# Aladdin Overview



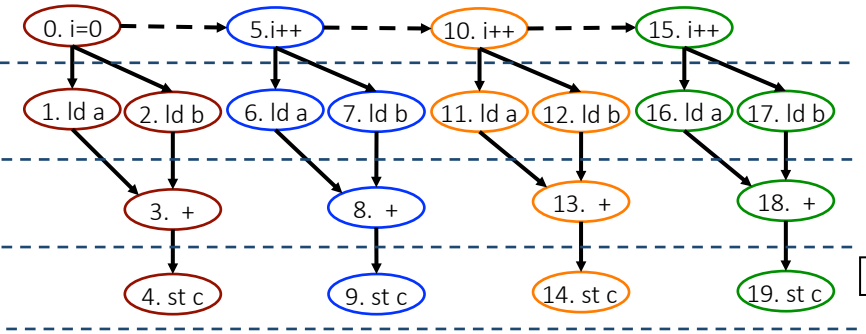


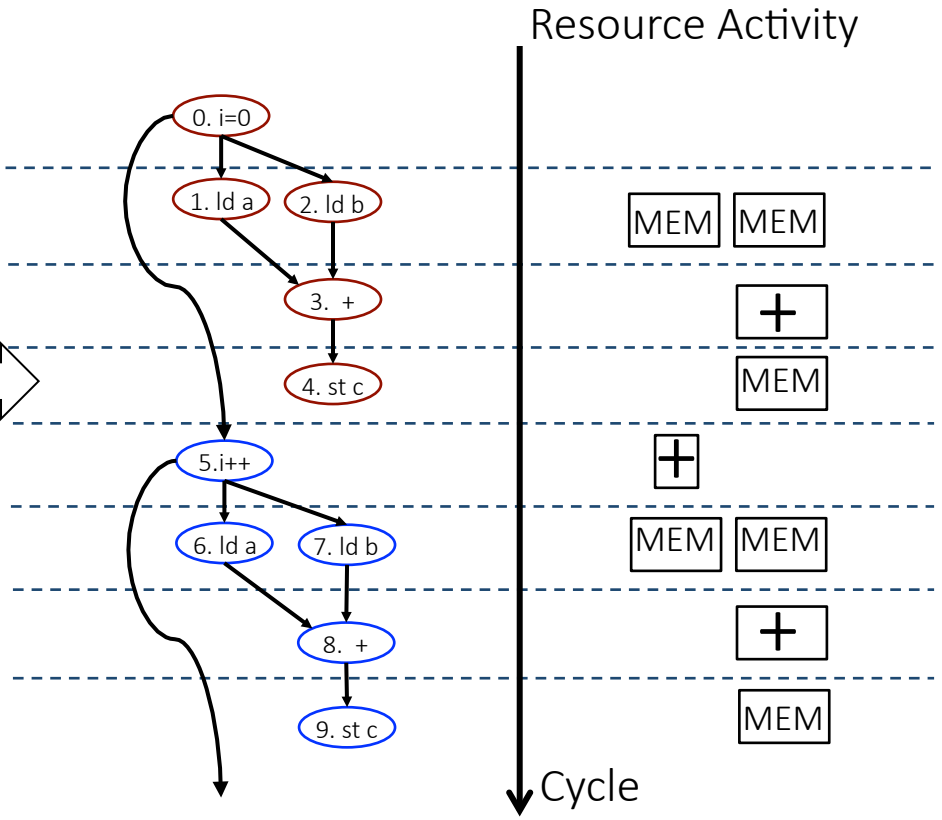

HARVARD UNIVERSITY

From C to Design Space

# One Design

Idealistic DDDG




Acc Design Parameters:

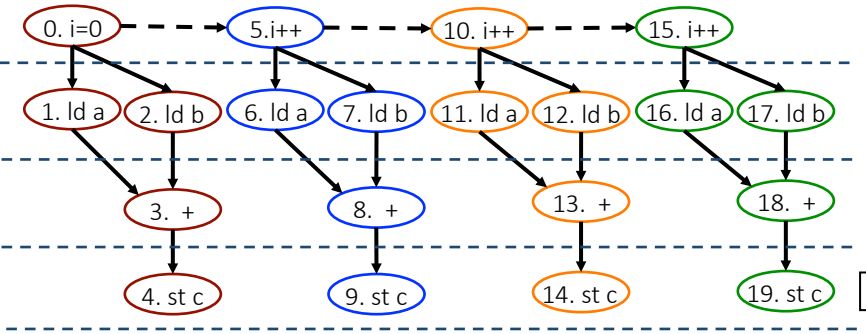
- ✓ Memory BW <= **2**
- ✓ **1** Adder

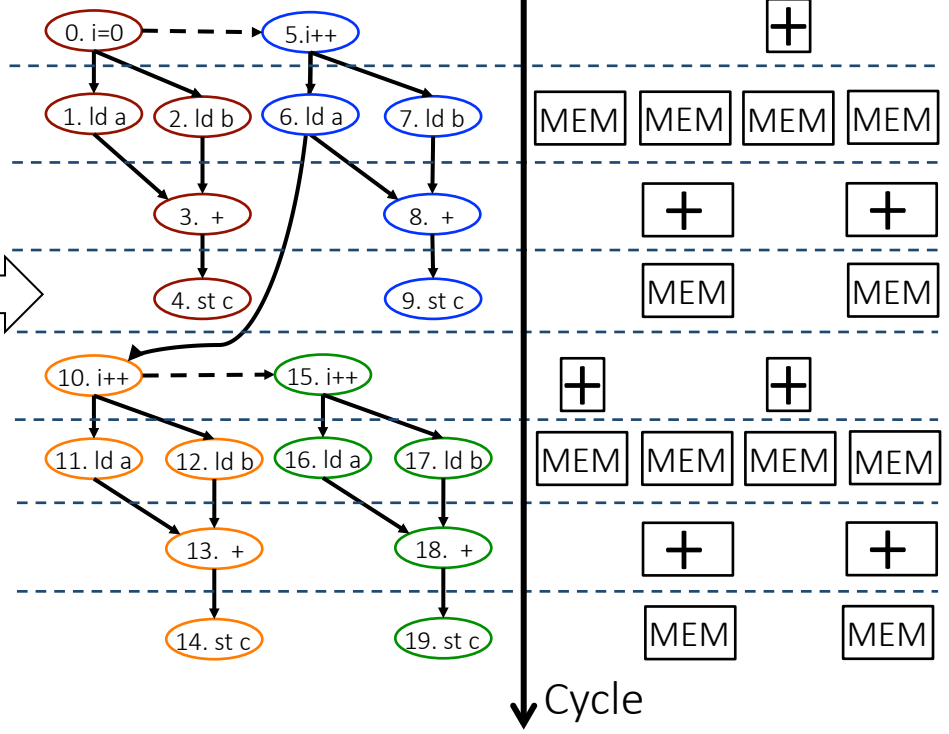


HARVARD UNIVERSITY

# Another Design

Idealistic DDDG




Acc Design Parameters:

- ✓ Memory BW <= **4**
- ✓ **2** Adders



HARVARD UNIVERSITY

# Realization Phase: DDDG->Power-Perf

- Constrain the DDDG with program and user-defined resource constraints
- Program Constraints
  - Control Dependence
  - Memory Ambiguation
- Resource Constraints
  - Loop-level Parallelism
  - Loop Pipelining
  - Memory Ports
  - # of FUs (e.g., adders, multipliers)



HARVARD UNIVERSITY

# Memory Ambiguation

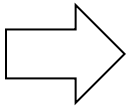
- Idealistic DDDG optimistically removes *all* false memory dependences
- Input-dependent memory accesses cannot be calculated statically.

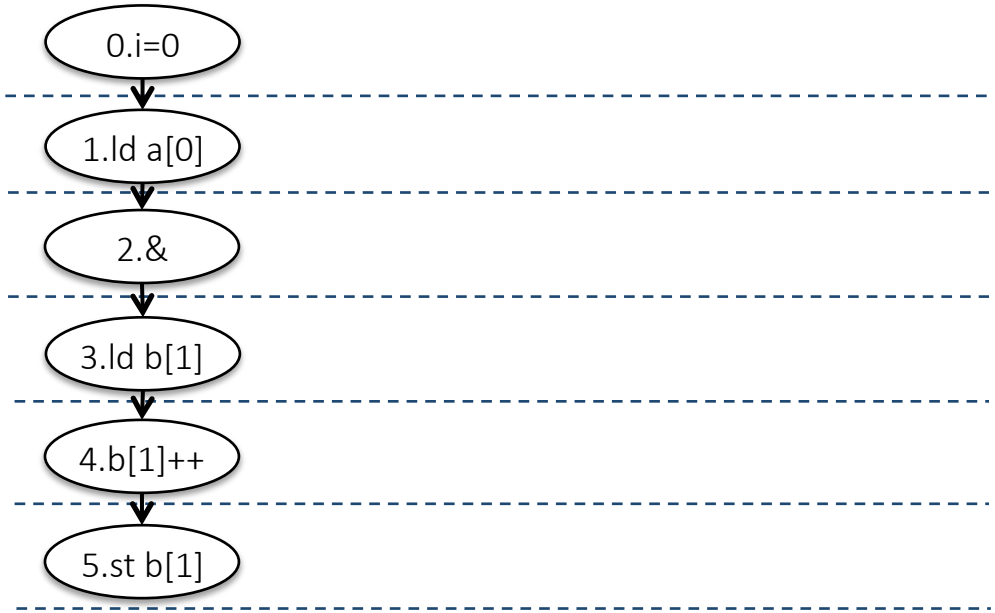


HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```





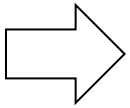

Input:
a[0] = 1;

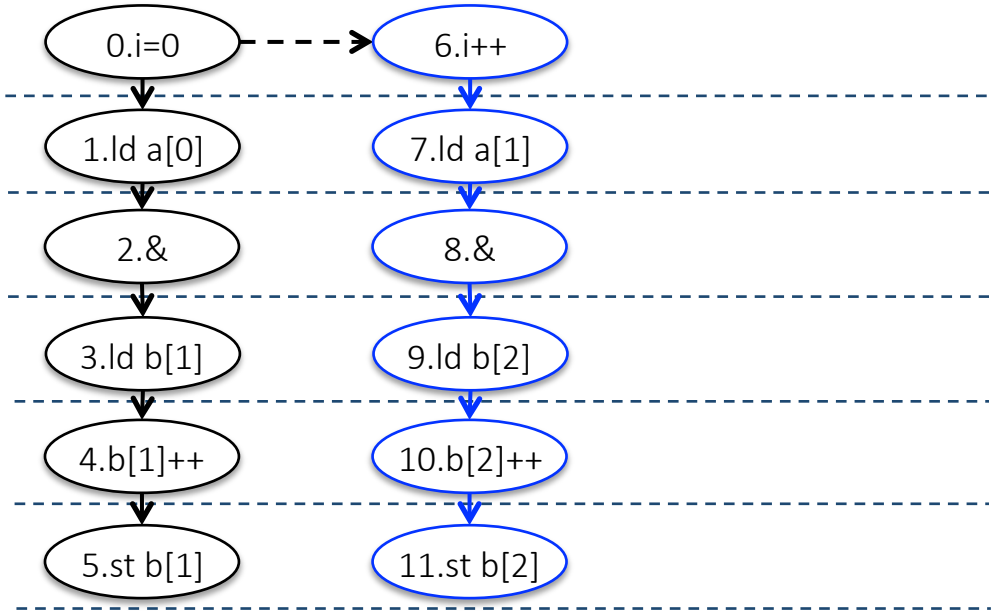


HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```






Input:

a[0] = 1;

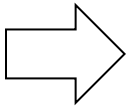
a[1] = 2;

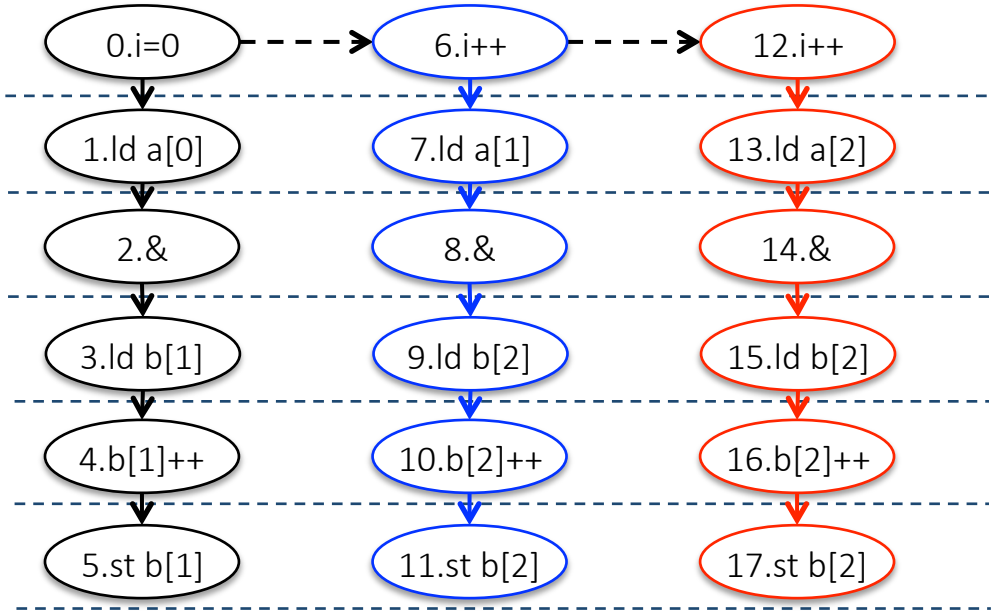


HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```






Input:

a[0] = 1;

a[1] = 2;

a[2] = 2;

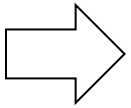
...

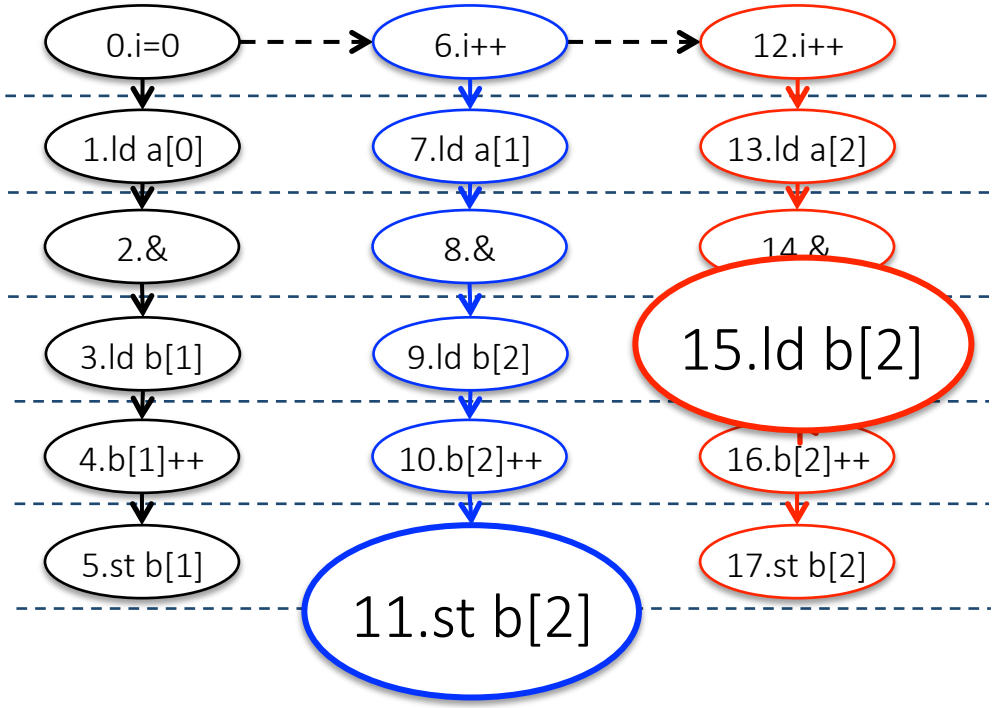


HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```






Input:

a[0] = 1;

a[1] = 2;

a[2] = 2;

…

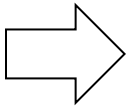


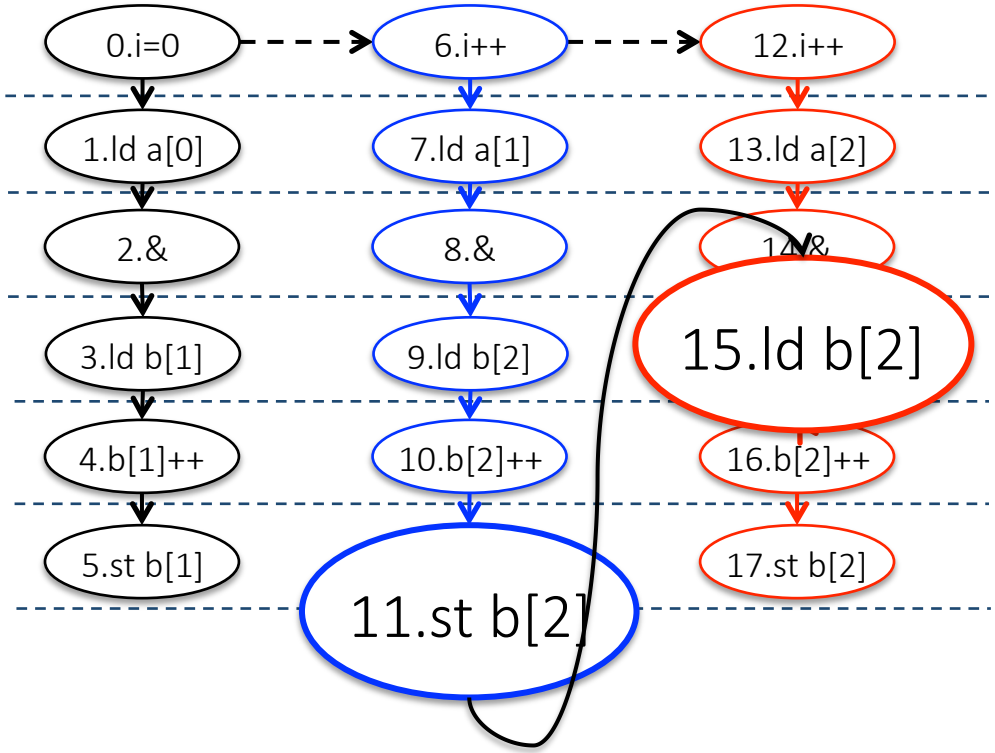
HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```

Input:
a[0] = 1;
a[1] = 2;
a[2] = 2;
…





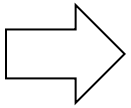


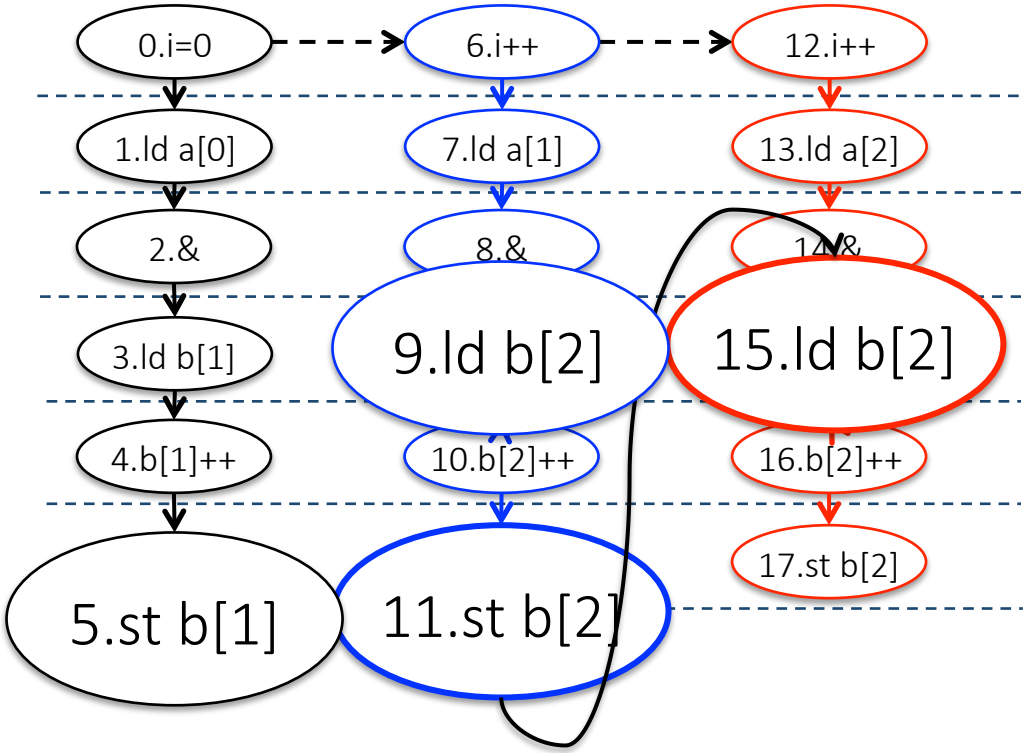

HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```

Input:
a[0] = 1;
a[1] = 2;
a[2] = 2;
…





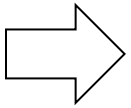


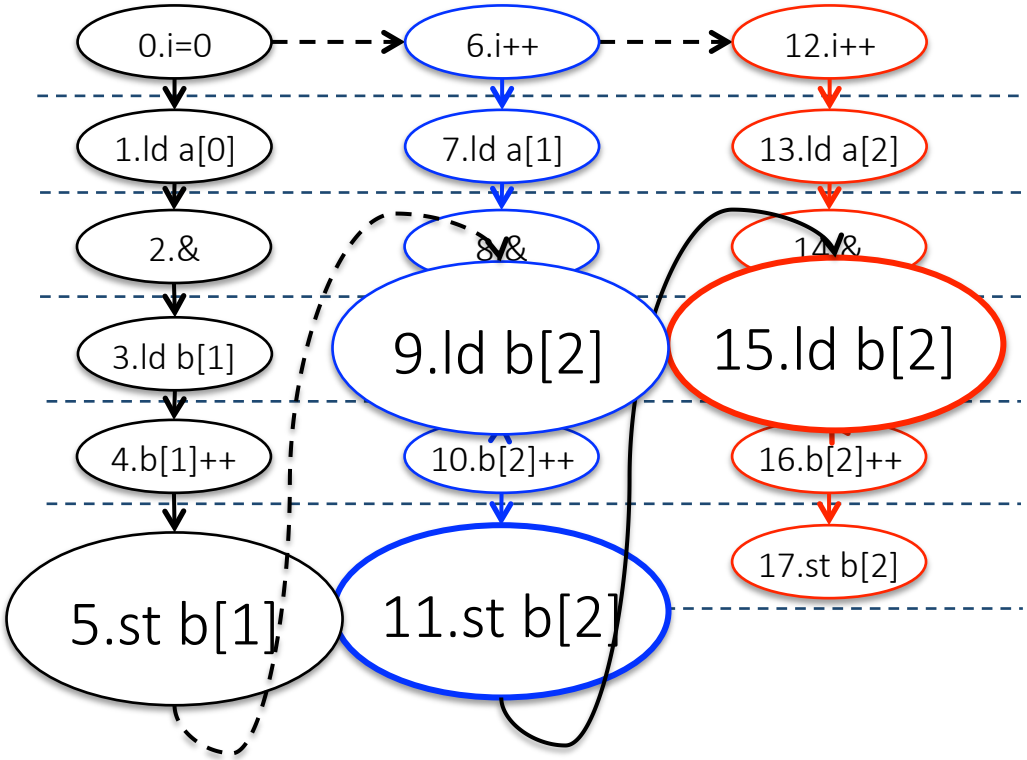

HARVARD UNIVERSITY

# Memory Ambiguation

```
for(i=0; i<N; ++i)
{
  bucket[ a[i] & 0x11 ]++;
}
```

Input:
a[0] = 1;
a[1] = 2;
a[2] = 2;
…

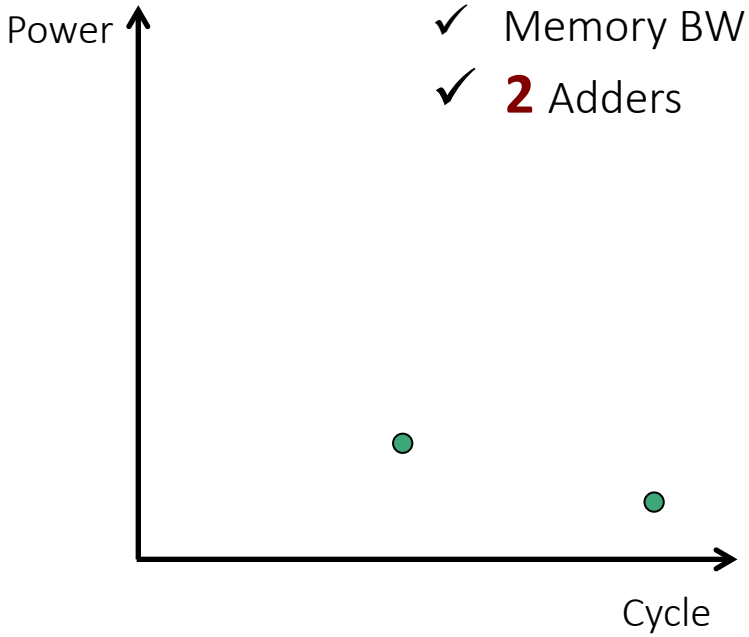







HARVARD UNIVERSITY

# Power-Performance per Design




Acc Design Parameters:

- ✓ Memory BW <= **4**
- ✓ **2** Adders

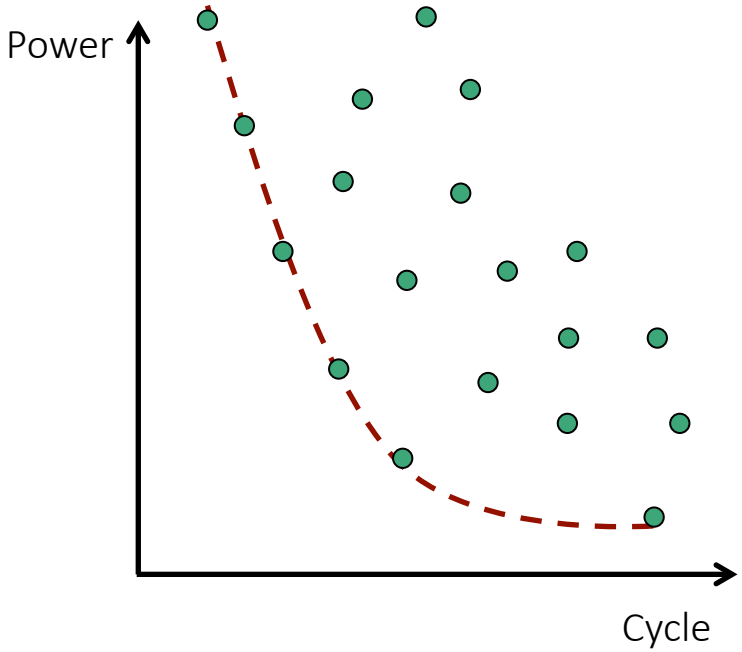
Acc Design Parameters:

- ✓ Memory BW <= **2**
- ✓ **1** Adder



HARVARD UNIVERSITY

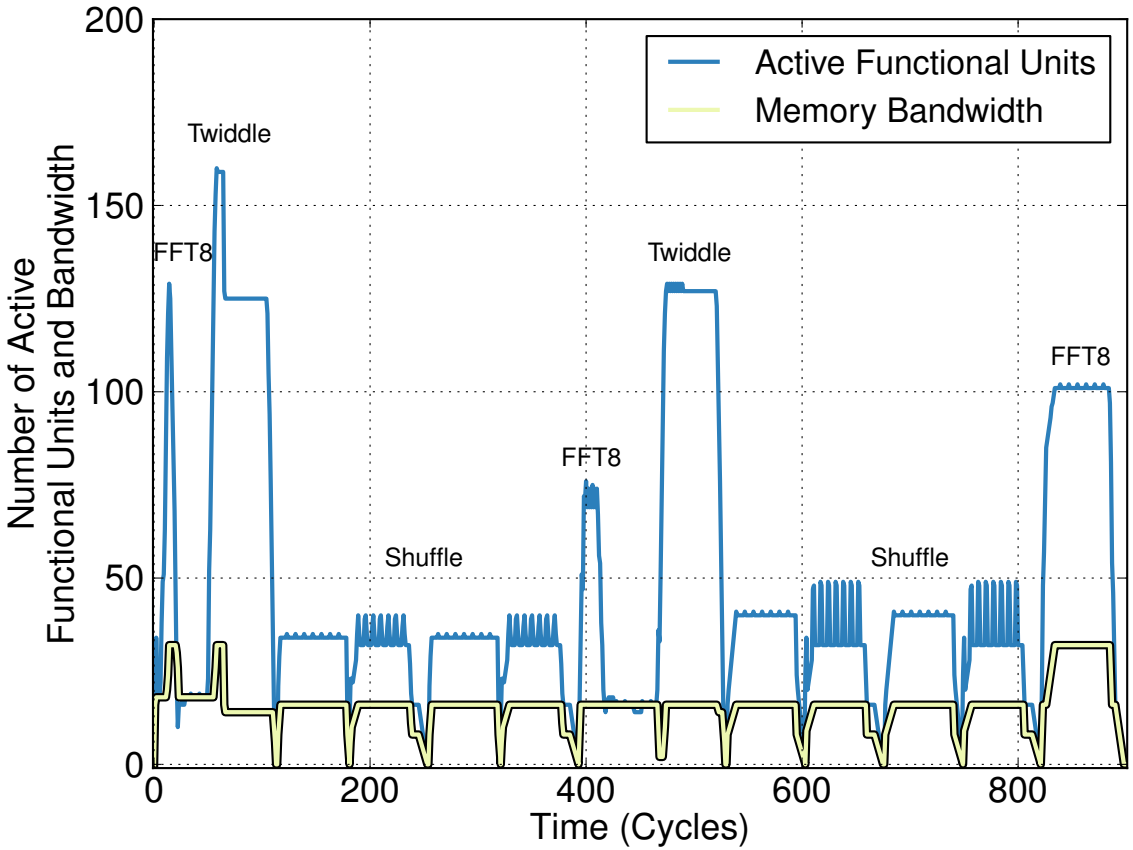
# Design Space of an Algorithm






HARVARD UNIVERSITY

# Cycle-Level Activity






HARVARD UNIVERSITY

# Power Model

- Functional Units Power Model
  - Microbenchmarks characterize various FUs.
  - Design Compiler with 40nm Standard Cell

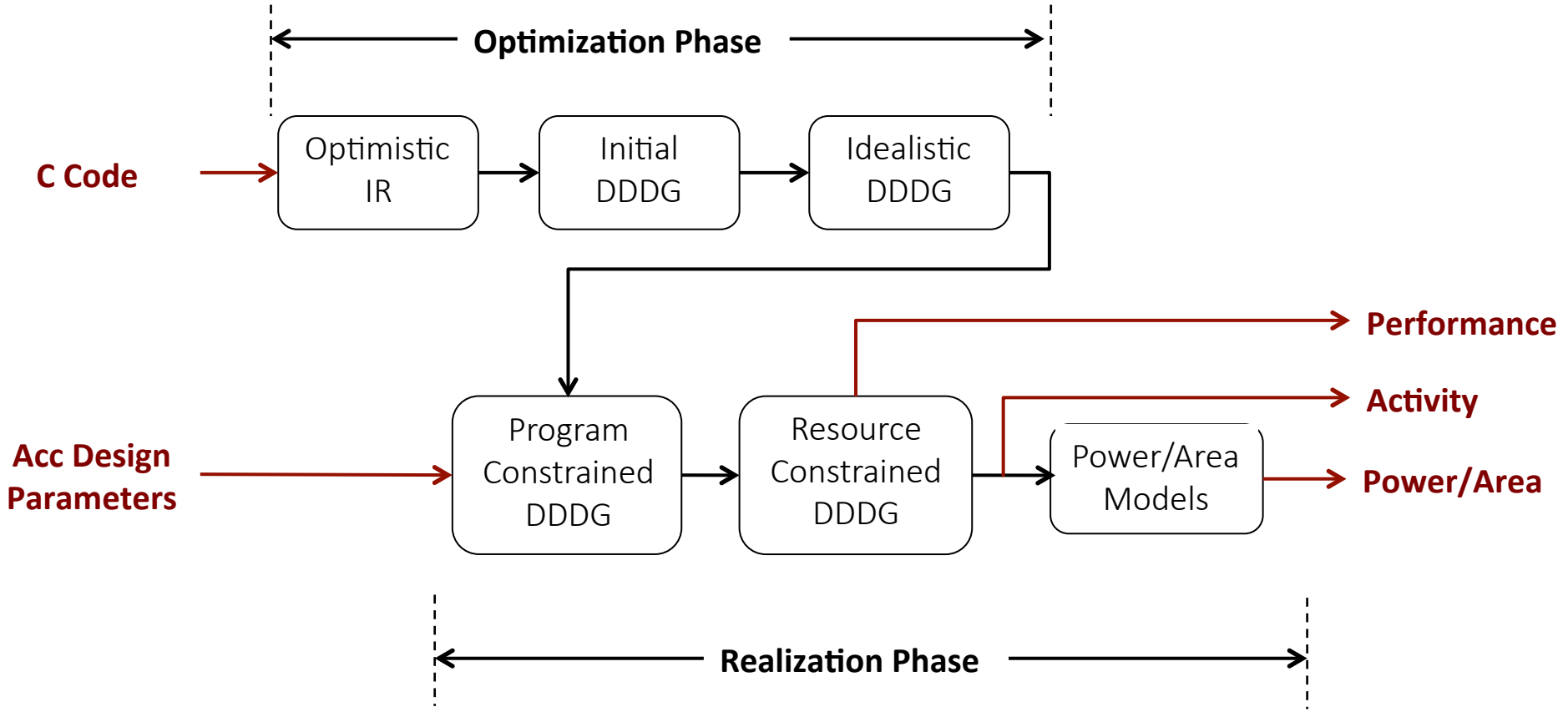
$$Power = \sum_{1<i<N} (activity_i * P_{i_{dynamic}}) + P_{i_{leakage}}$$

- SRAM Power Model
  - Commercial register file and SRAM memory compilers with the same 40nm standard cell library



HARVARD UNIVERSITY

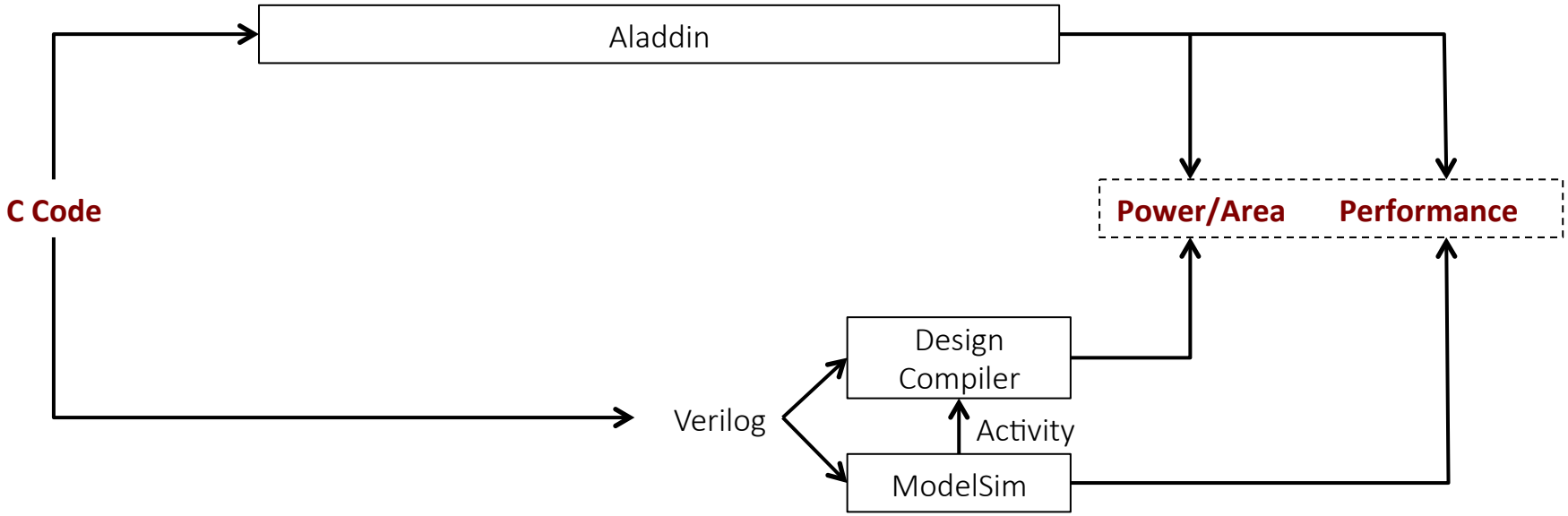
# Aladdin Overview






HARVARD UNIVERSITY

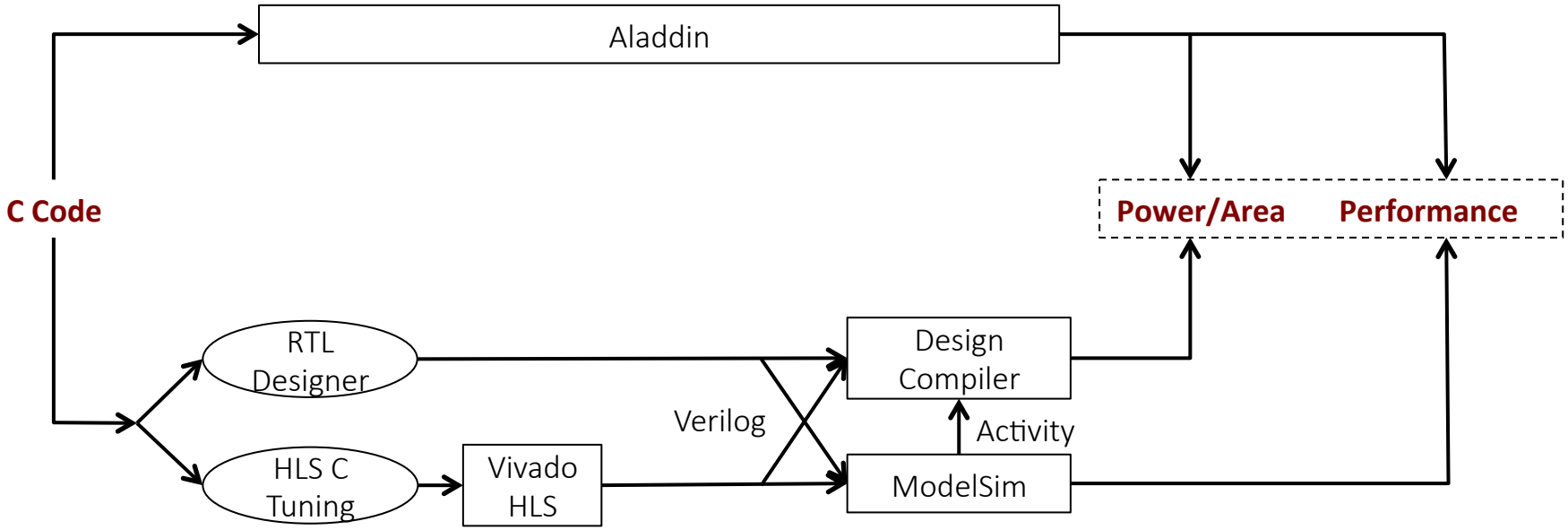
# Aladdin Validation






HARVARD UNIVERSITY

# Aladdin Validation





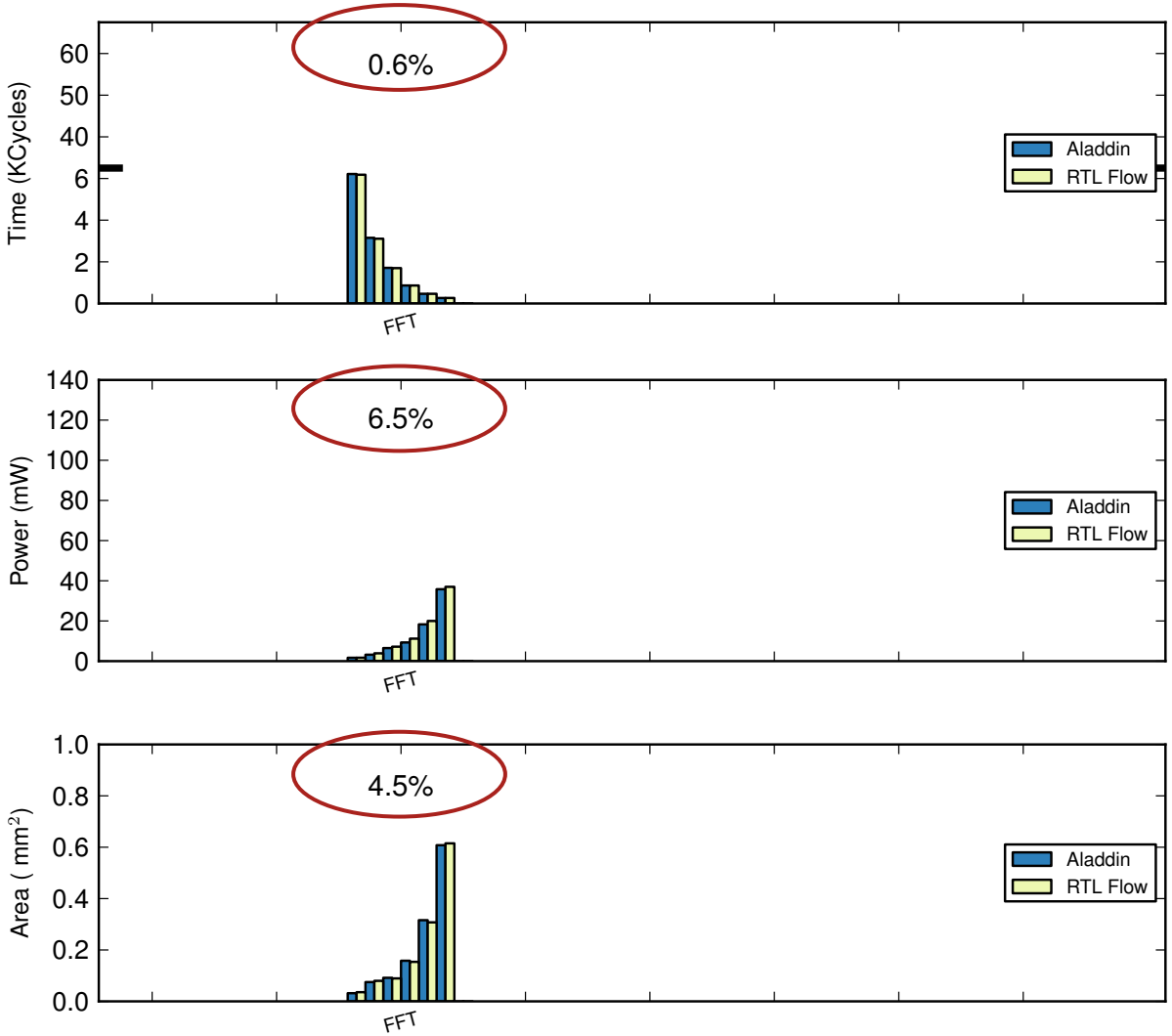

HARVARD UNIVERSITY

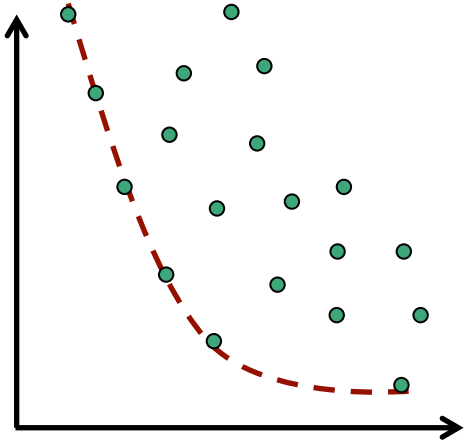
# Validation Benchmarks

| Type | Benchmark | Description | |
|---|---|---|---|
| SHOC Benchmark Suite | MD | Pairwise calculation of the L-J Potential | Optimized HLS Designs |
| | STENCIL | Apply 3x3 filter to an image | |
| | FFT | 1D 512 FFT | |
| | GEMM | Blocked Matrix Multiply | |
| | TRIAD | Single Computation in DOALL loop | |
| | SORT | Radix Sort | |
| | SCAN | Parallel prefix sum | |
| | REDUCTION | Return sum of an array | |
| Proposed Accelerator Constructs | NPU | An individual neuron in a network [MICRO'12] | Hand RTL Designs |
| | Memcached | GET function in Memcached [ISCA'13] | |
| | HARP | Data partition accelerator [ISCA'13] | |


HARVARD UNIVERSITY

# Aladdin Validation

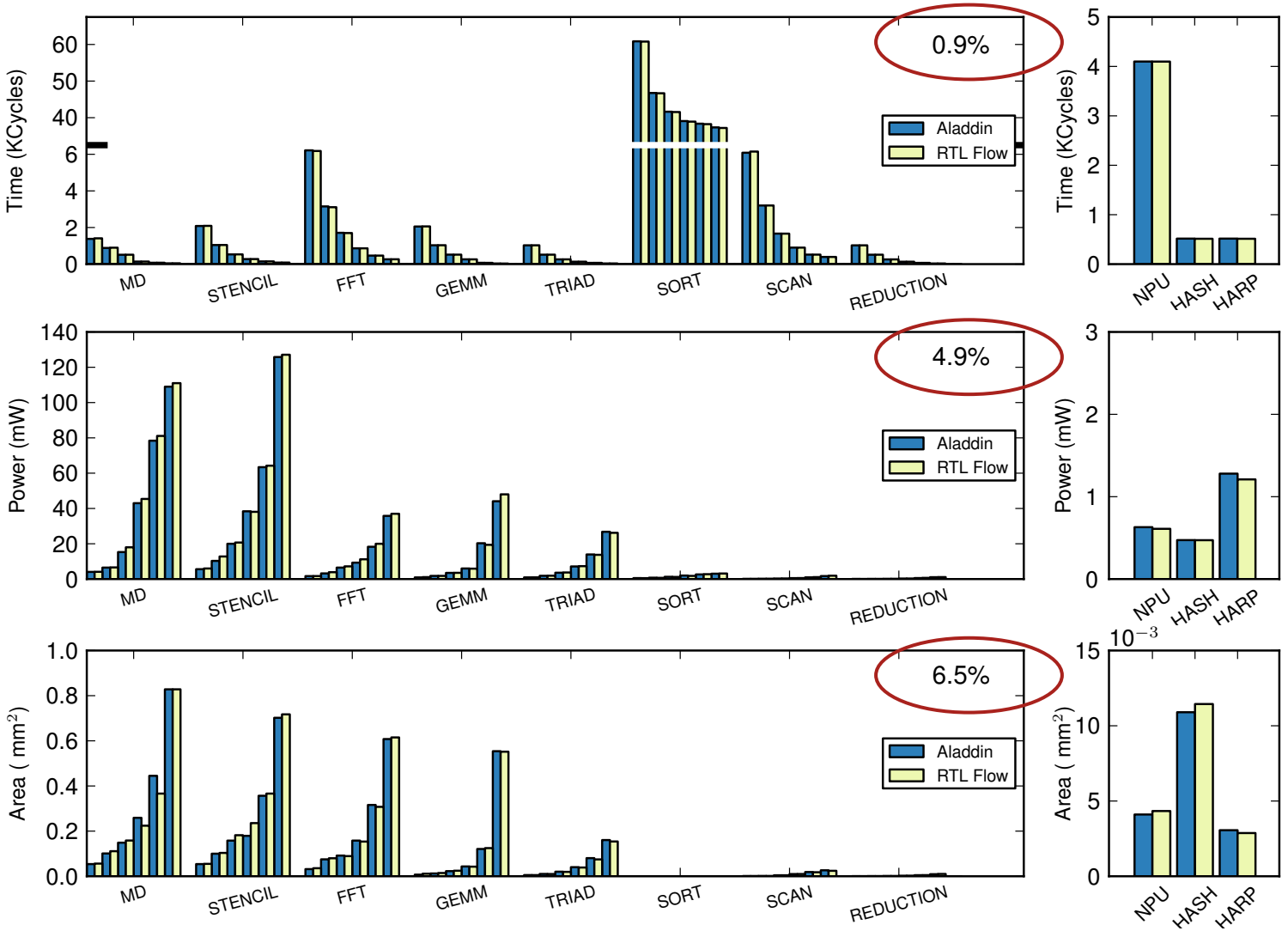




HARVARD UNIVERSITY

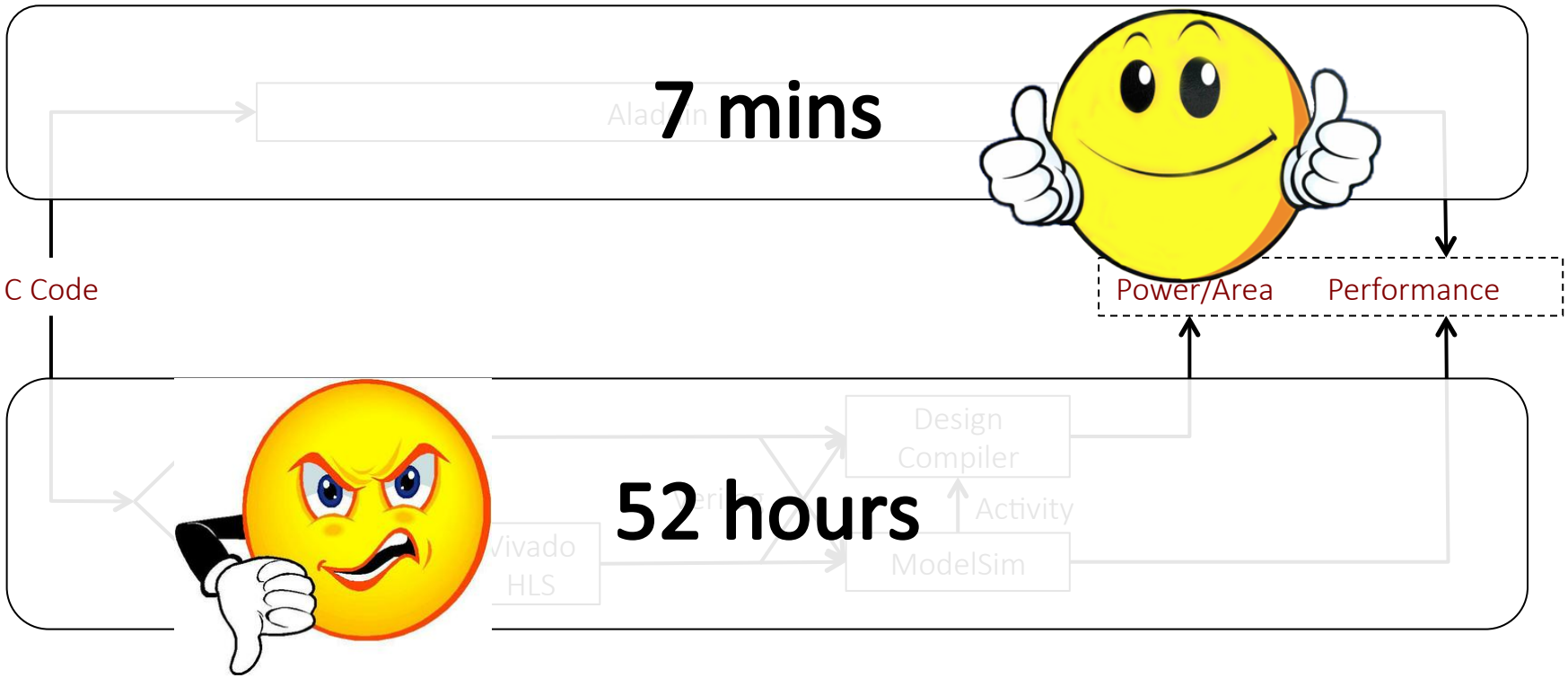
# Aladdin Validation






HARVARD UNIVERSITY

# Aladdin enables rapid design space exploration for accelerators.






HARVARD UNIVERSITY

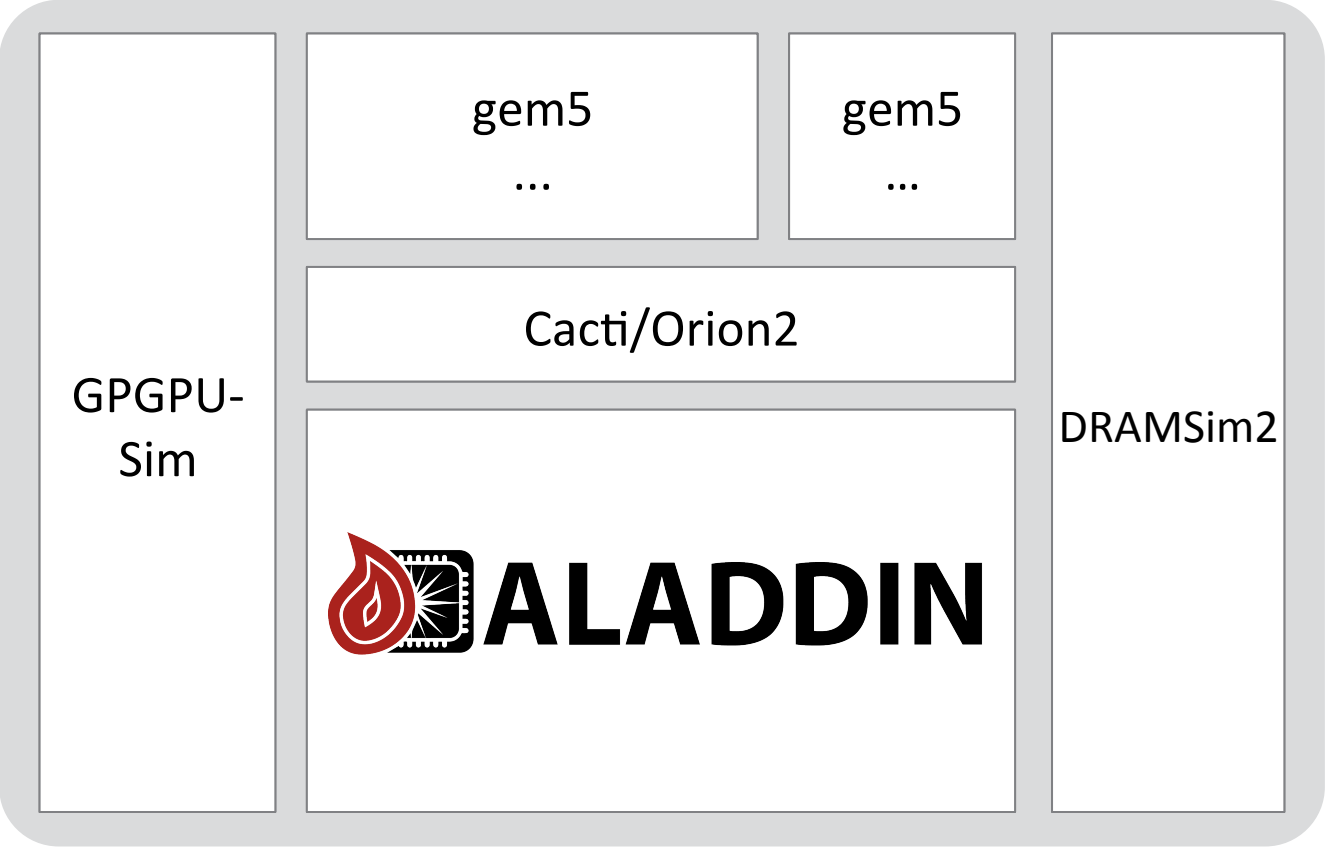
# Limitations

- Algorithm Choices
  - Aladdin generates a design space per algorithm
  - Can use Aladdin to quickly compare the design spaces of algorithms
- Input Dependent
  - Inputs that exercise all paths of the code
- Input C Code
  - Aladdin can create DDDG for any C code.
  - C constructs that require resources outside the accelerator, such as system calls and dynamic memory allocation, are not modeled.



HARVARD UNIVERSITY

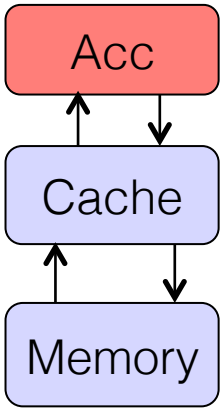
# Aladdin enables pre-RTL simulation of accelerators with the rest of the SoC.

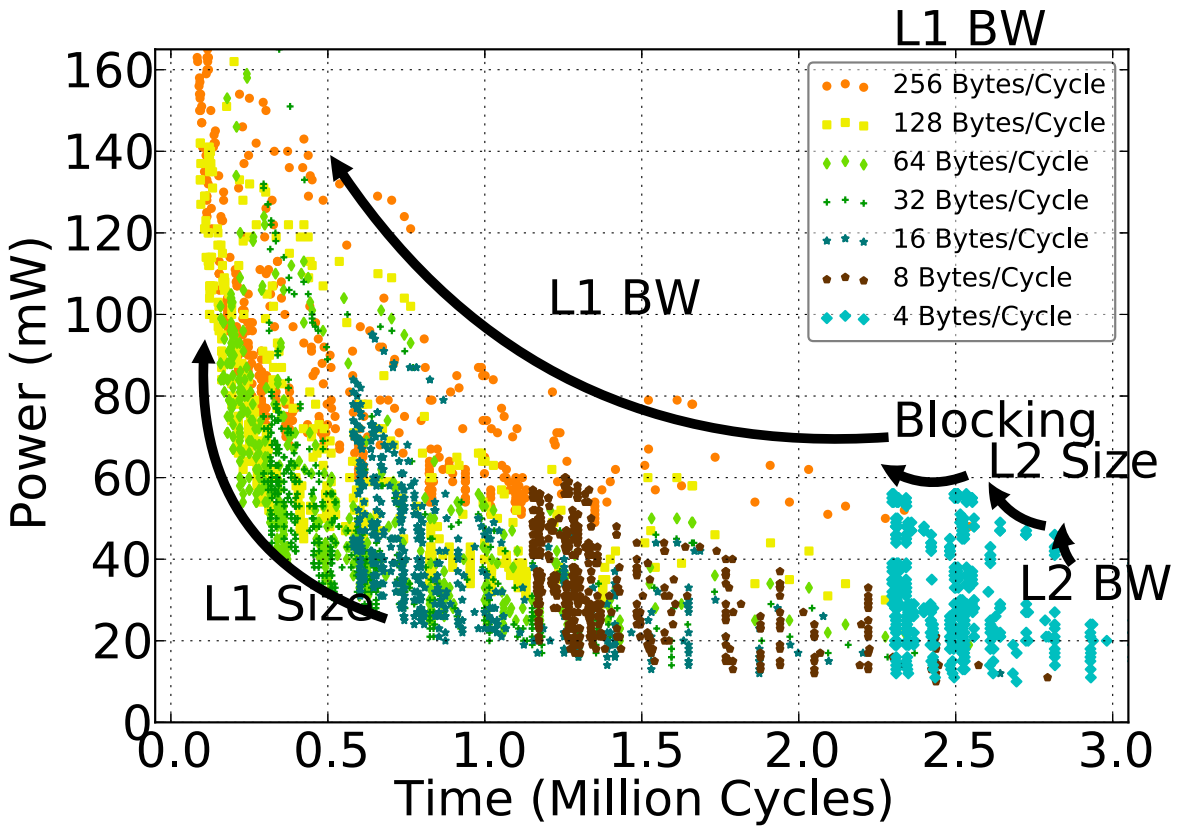





HARVARD UNIVERSITY

# Simulating Accelerator with Memory System using Aladdin

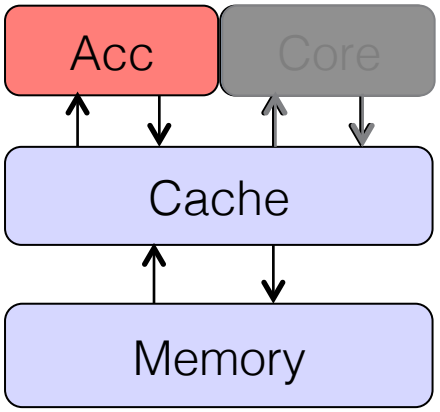


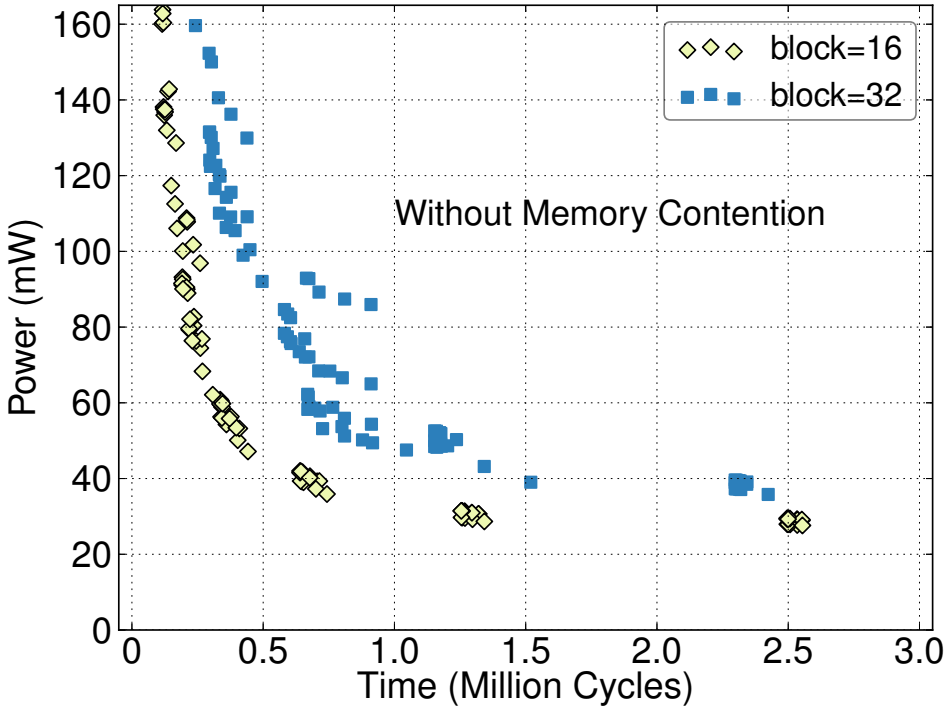





HARVARD UNIVERSITY

# Modeling Accelerators in an SoC-like Environment

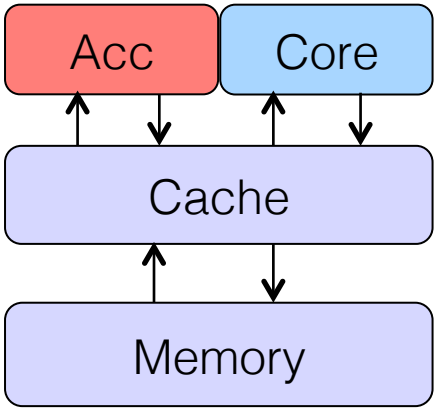


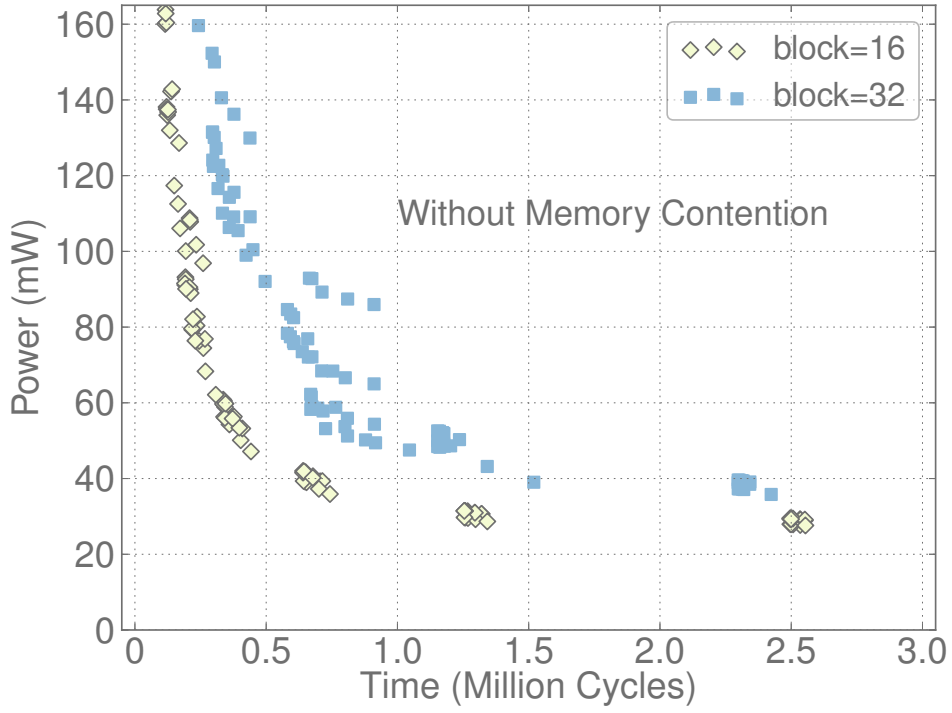



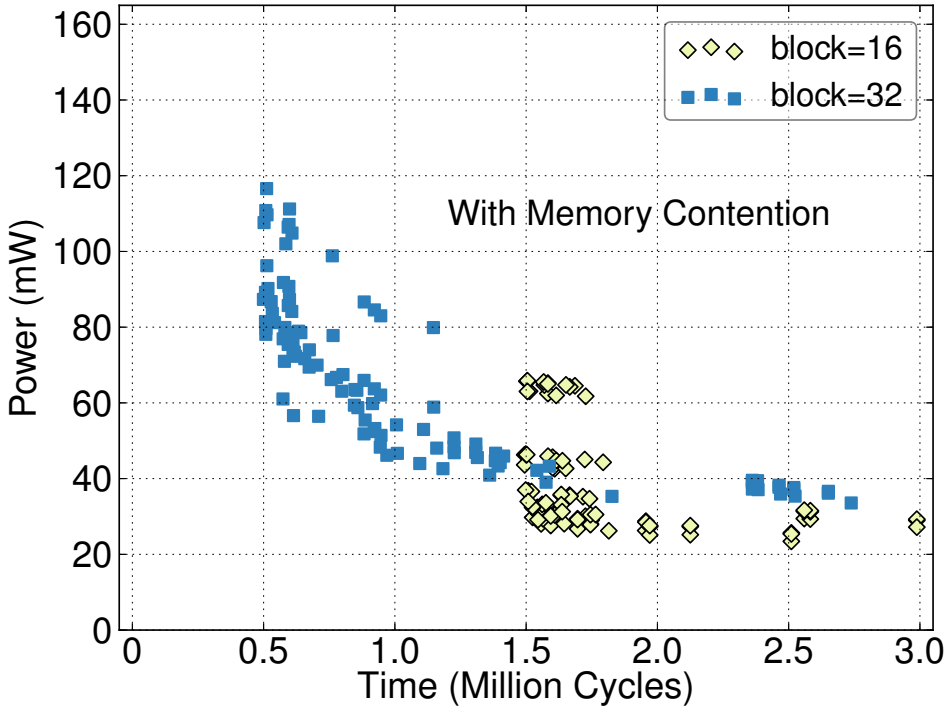



HARVARD UNIVERSITY

# Modeling Accelerators in a SoC-like Environment

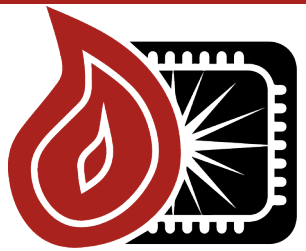





HARVARD UNIVERSITY



# Aladdin: A pre-RTL, Power-Performance Accelerator Simulator

- Architectures with 1000s of accelerators will be radically different; New design tools are needed.
- Aladdin enables rapid design space exploration of future accelerator-centric platforms.
- You can find Aladdin at
  - http://vlsiarch.eecs.harvard.edu/accelerators

# Tutorial References


- Y.S. Shao and D. Brooks, “ISA-Independent Workload Characterization and its Implications for Specialized Architectures,” ISPASS’13.
- B. Reagen, Y.S. Shao, G.-Y. Wei, D. Brooks, “Quantifying Acceleration: Power/ Performance Trade-Offs of Application Kernels in Hardware,” ISLPED’13.
- Y.S. Shao, B. Reagen, G.-Y. Wei, D. Brooks, “Aladdin: A Pre-RTL, Power-Performance Accelerator Simulator Enabling Large Design Space Exploration of Customized Architectures,” ISCA’14.
- B. Reagen, B. Adolf, Y.S. Shao, G.-Y. Wei, D. Brooks, “MachSuite: Benchmarks for Accelerator Design and Customized Architectures,” IISWC’14.
- **http://vlsiarch.eecs.harvard.edu/accelerators**




HARVARD UNIVERSITY