Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs

MICRO Tutorial

October 12, 2019

Website: http://accelergy.mit.edu/tutorial.html

Massachusetts Institute of Technology

Accelergy Overview

- An architecture-level energy estimator that can be easily integrated into other design exploration tools
- Accurately and systematically captures the energy consumption of basic building blocks in the design
- Succinctly models diverse and complicated designs
- Achieves 95% accuracy in evaluating a well-know deep neural network accelerator design – Eyeriss [2016 ISSCC]

Accelergy Overview

- **An architecture-level energy estimator that can be easily integrated into other design exploration tools**
- Accurately and systematically captures the energy consumption of basic building blocks in the design
- Succinctly models diverse and complicated designs
- Achieves 95% accuracy in evaluating a well-know deep neural network accelerator design – Eyeriss [2016 ISSCC]

Importance of Energy Estimation

Energy efficiency is the general concern for accelerator designs

Object detection

Scene recognition

Energy-efficient DNN accelerator

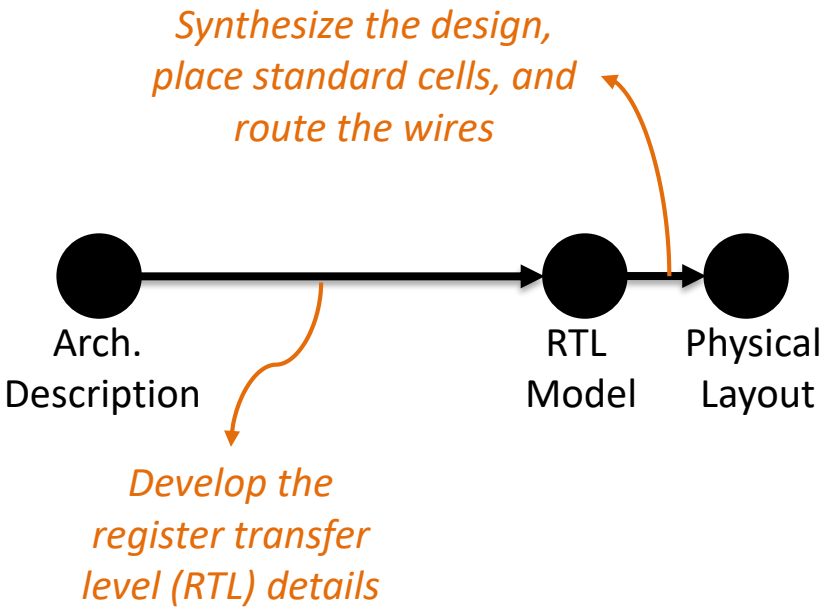
We must quickly evaluate energy efficiency of arbitrary potential designs in the large design space

Energy Estimation and Design Exploration

- **Physical-level energy estimators**

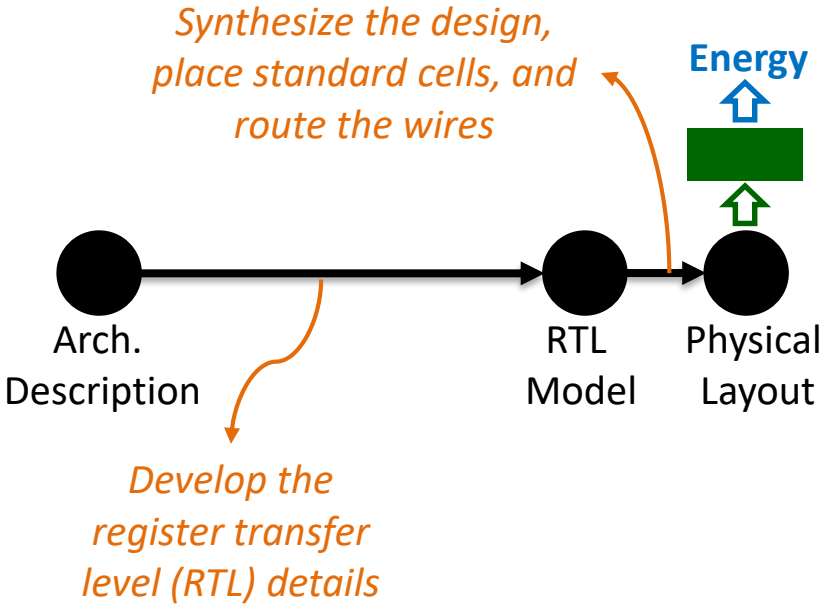
Energy Estimation and Design Exploration

- **Physical-level energy estimators**



Energy Estimation and Design Exploration

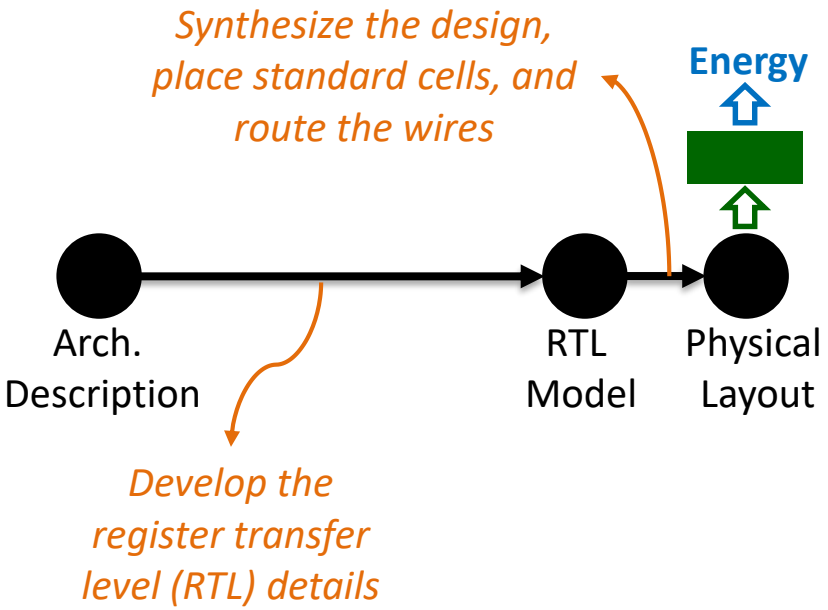
- **Physical-level energy estimators**

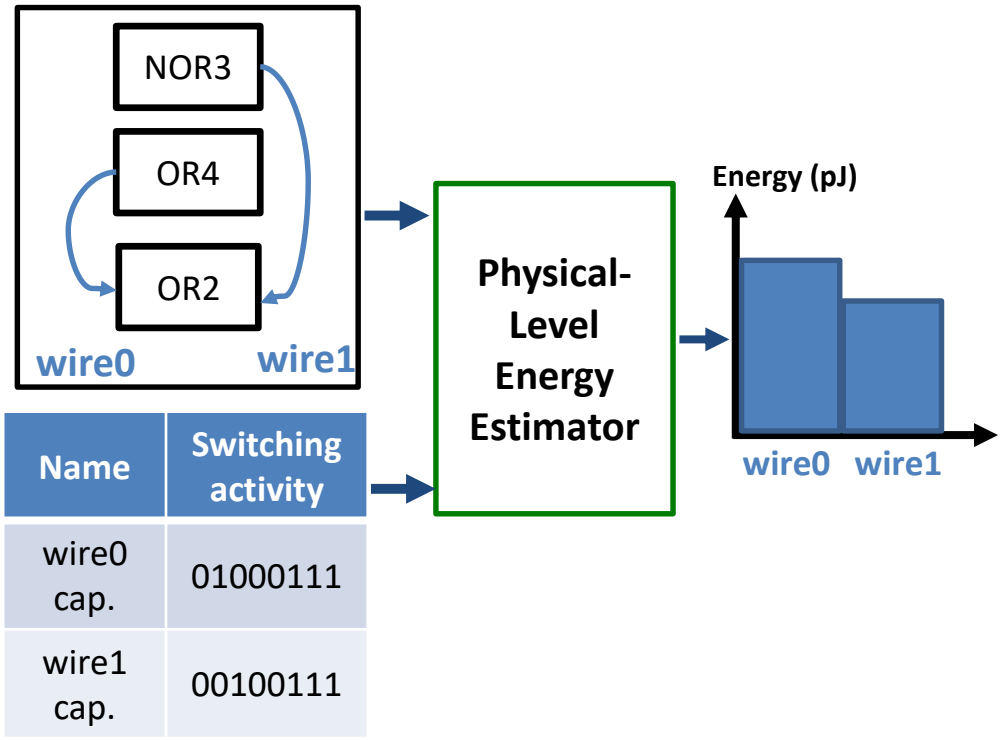


Physical-Level Energy Estimator

Energy Estimation and Design Exploration

- **Physical-level energy estimators**



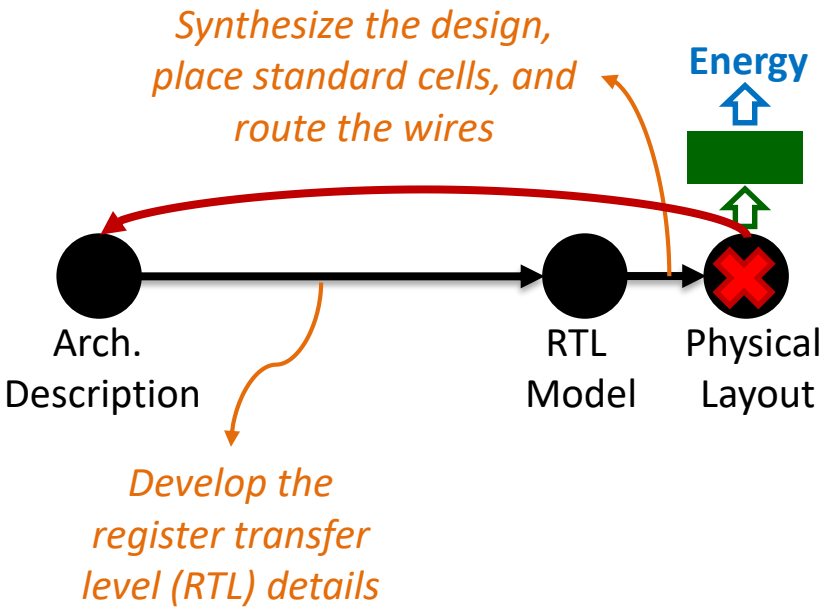


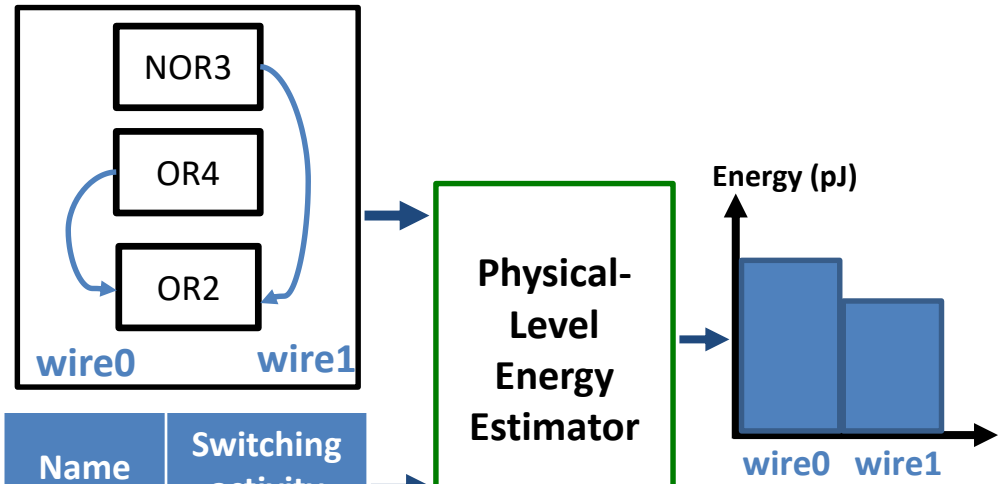
Name	Switching activity
wire0 cap.	01000111
wire1 cap.	00100111

Physical-level simulations

Energy Estimation and Design Exploration

- **Physical-level energy estimators**



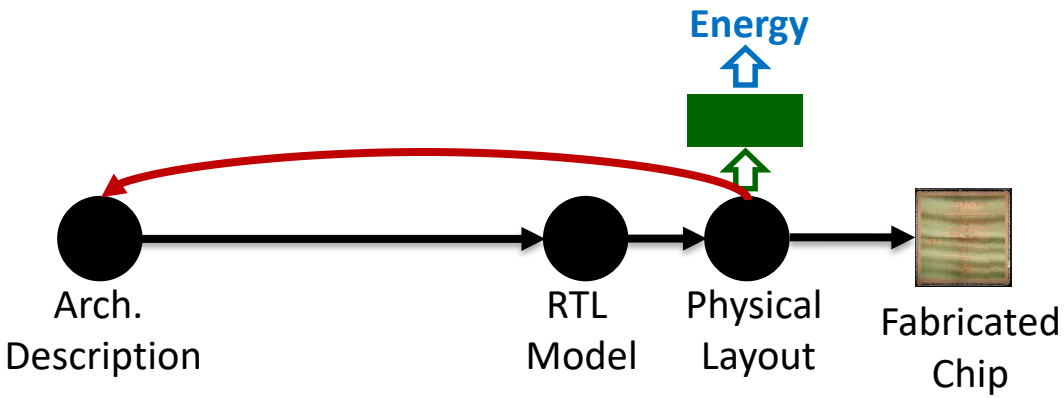


Name	Switching activity
wire0 cap.	01000111
wire1 cap.	00100111

Physical-level simulations

Energy Estimation and Design Exploration

- **Physical-level energy estimators**



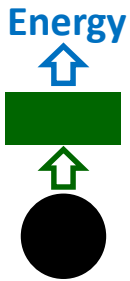
Requires physical layout of the design
Slow design space exploration

Energy Estimation and Design Exploration

- **Architecture-Level Energy Estimators**

Energy Estimation and Design Exploration

- **Architecture-Level Energy Estimators**

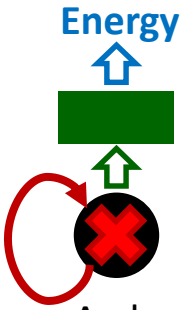


Arch Description

Architecture -Level Energy Estimator

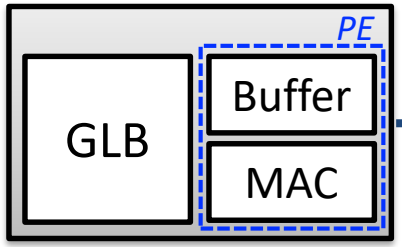
Energy Estimation and Design Exploration

- **Architecture-Level Energy Estimators**



Arch Description

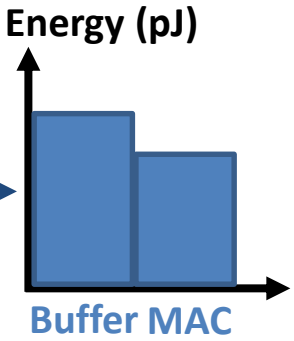
Architecture Description



Comp. name	Action name	Action counts
Buffer	access	800
MAC	compute	400

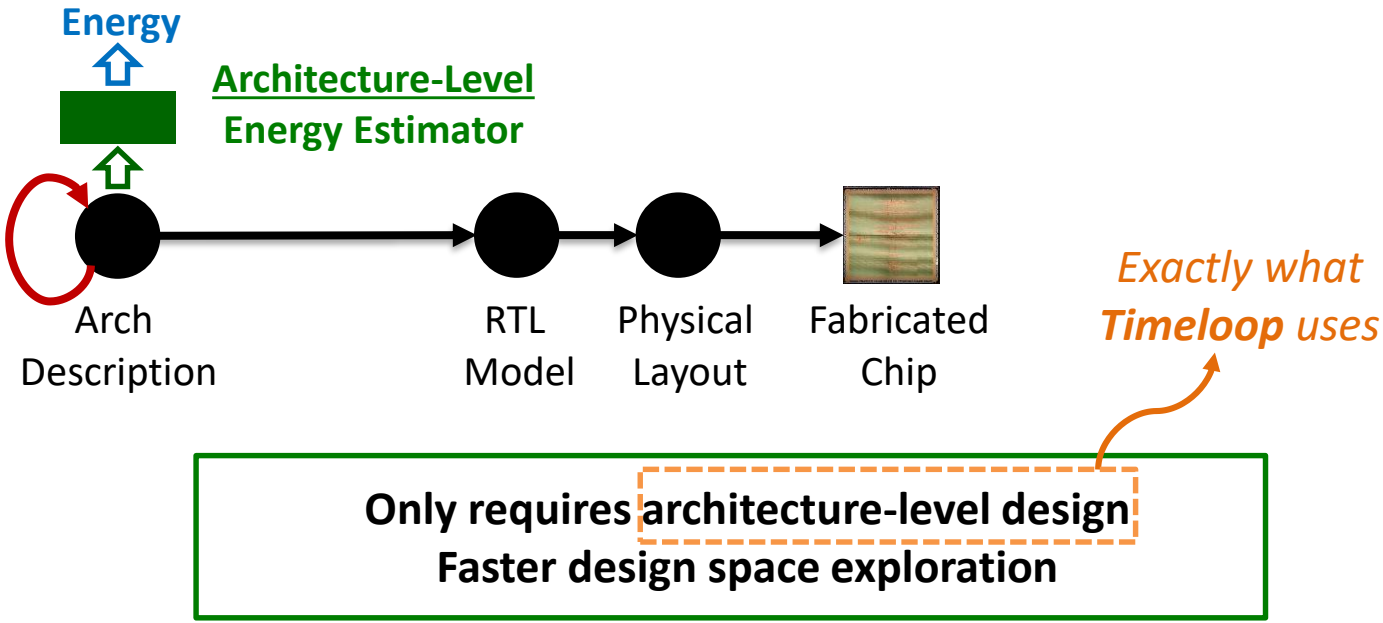
Action Counts

Architecture -Level Energy Estimator



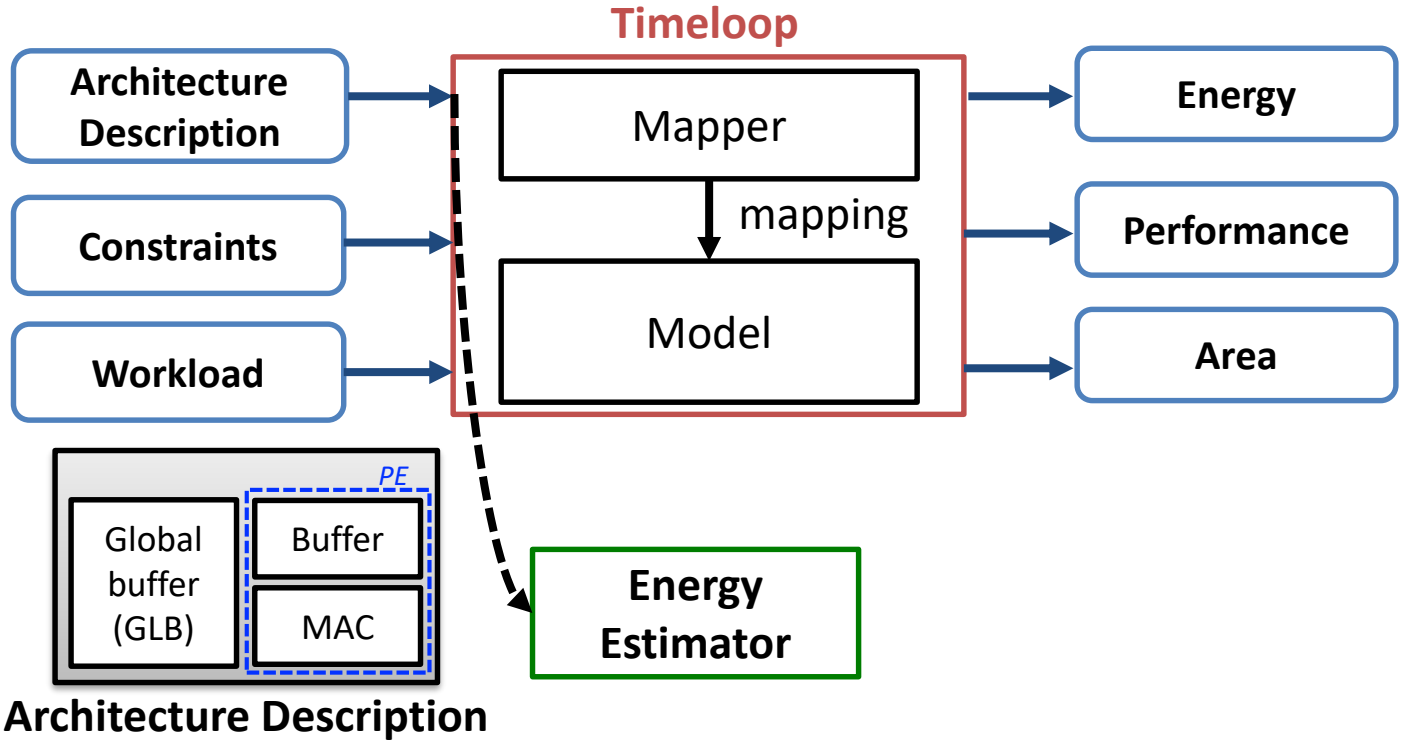
Energy Estimation and Design Exploration

- **Architecture-Level Energy Estimators**



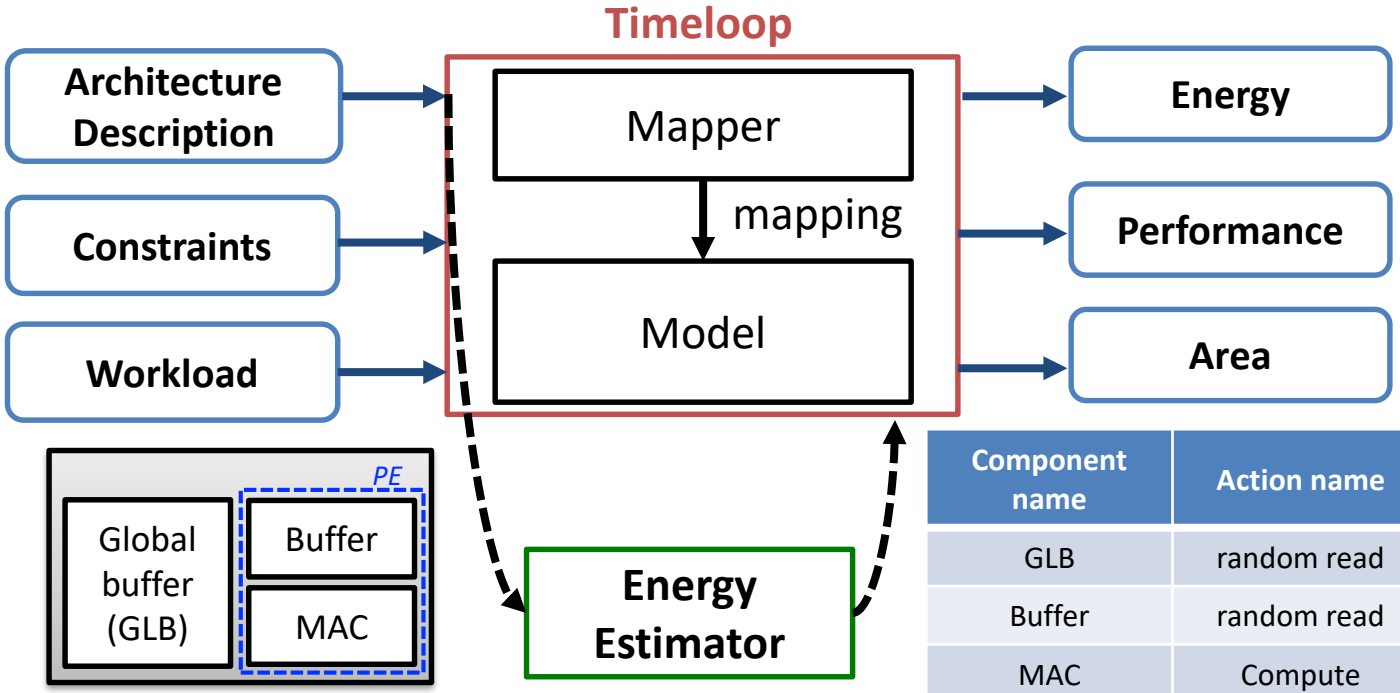
Importance of Energy Estimations

Timeloop requires energy reference tables (ERTs) to evaluate the energy efficiency of a design



Importance of Energy Estimations

Timeloop requires energy reference tables (ERTs) to evaluate the energy efficiency of a design

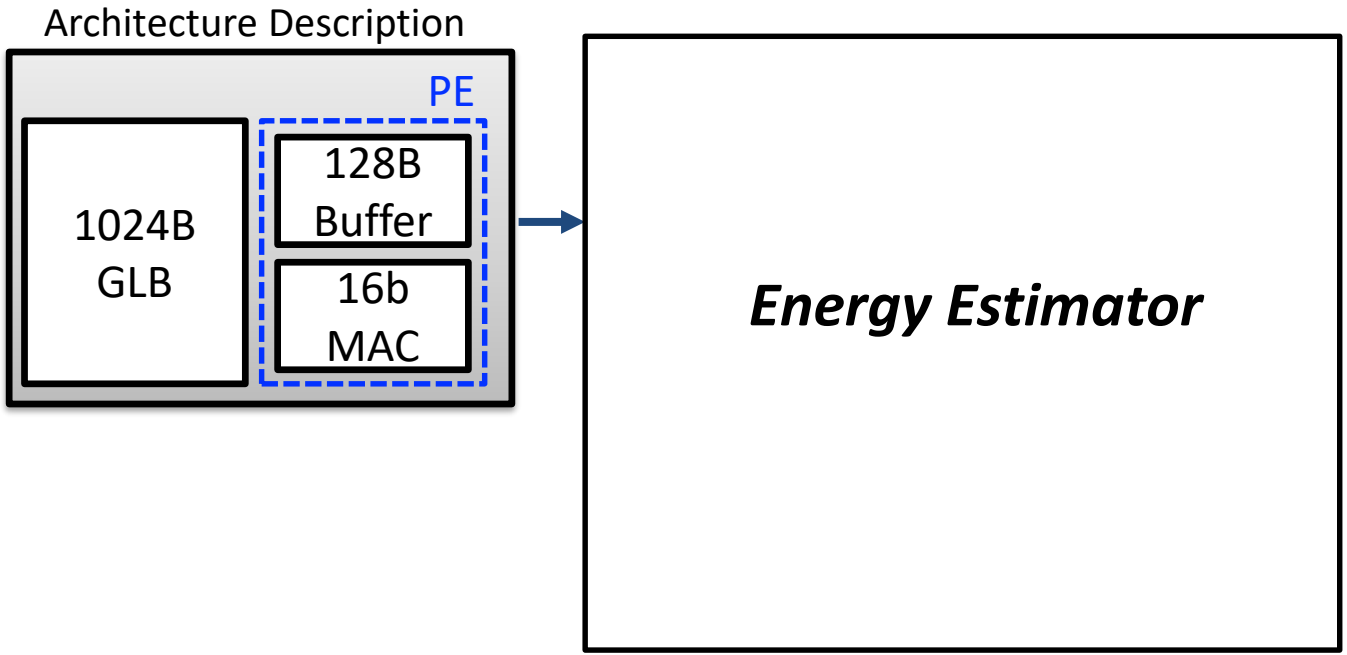


Component name	Action name	Energy /action
GLB	random read	100pJ
Buffer	random read	10pJ
MAC	Compute	5pJ

ERTs

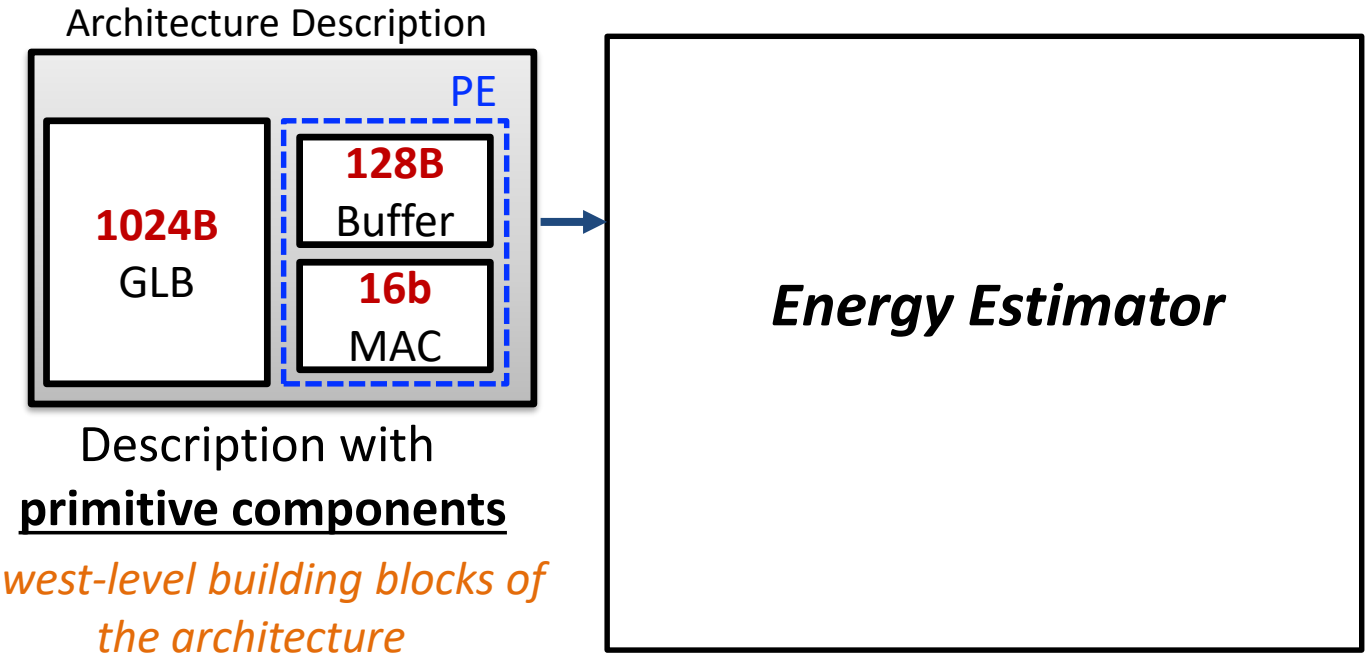
Existing Architecture-Level Energy Estimators

- **Design-Specific Accelerator Estimators: Aladdin**[ISCA2014], **fixed-cost**[Asilomar2017]



Existing Architecture-Level Energy Estimators

- **Design-Specific** **Accelerator Estimators**



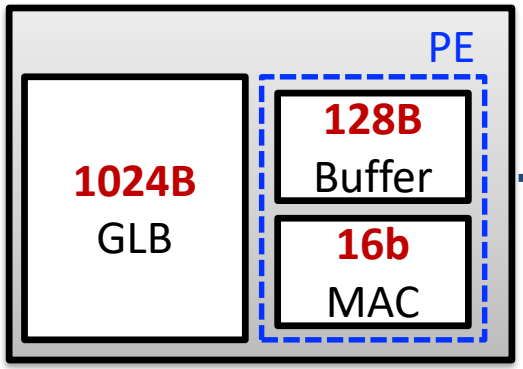
Description with **primitive components**

Lowest-level building blocks of the architecture

Existing Architecture-Level Energy Estimators

- **Design-Specific** Accelerator Estimators

Architecture Description



Description with **primitive components**

Lowest-level building blocks of the architecture

Energy Estimator

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	access	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

Coarse-Grained Actions

Existing Architecture-Level Energy Estimators

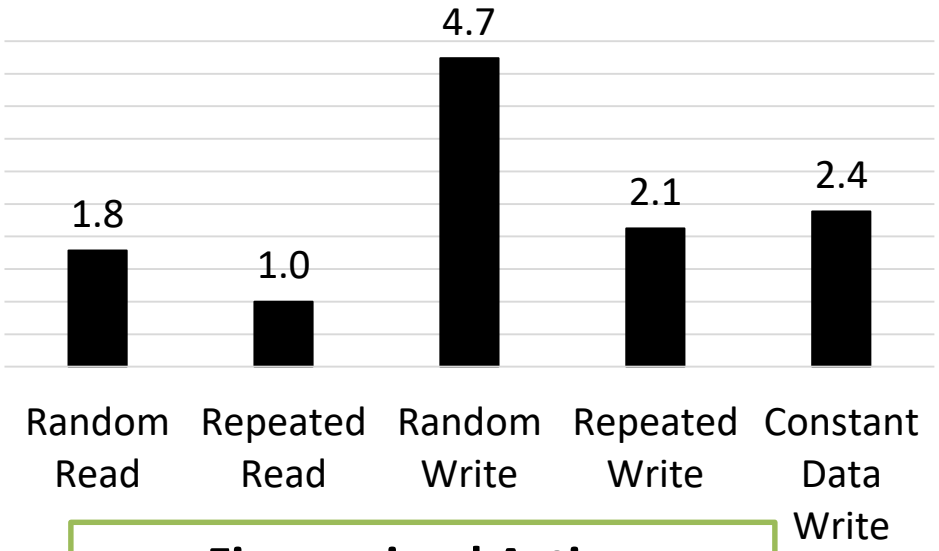
- **Design-Specific** Accelerator Estimators

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	**access**	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

Coarse-grained Action

Energy-Per-Actions of a Register File (normalized to idle)



Fine-grained Action

Existing Architecture-Level Energy Estimators

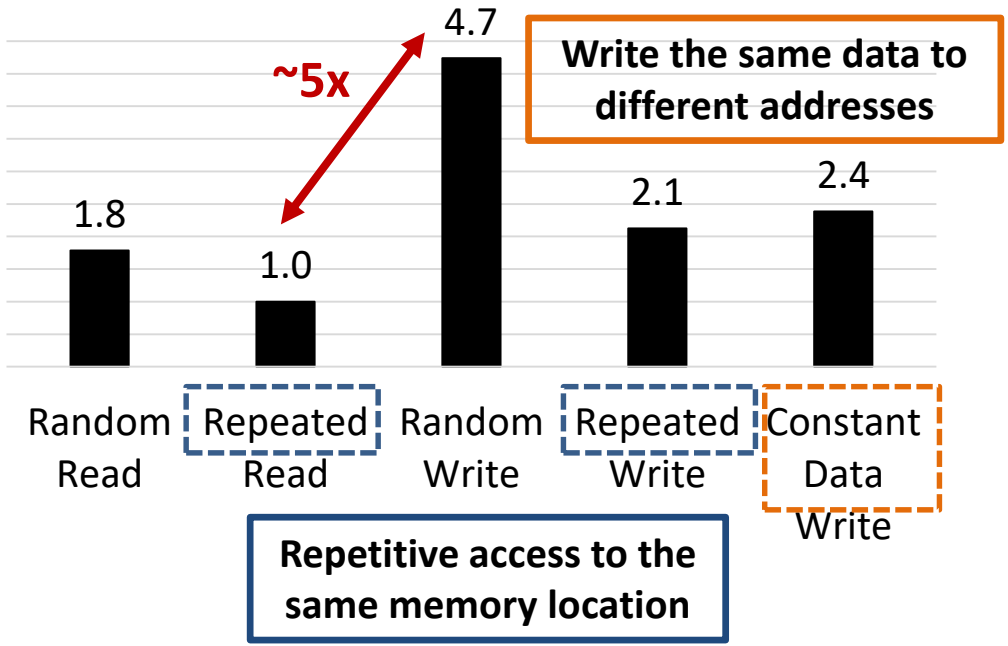
- **Design-Specific** Accelerator Estimators

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	**access**	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

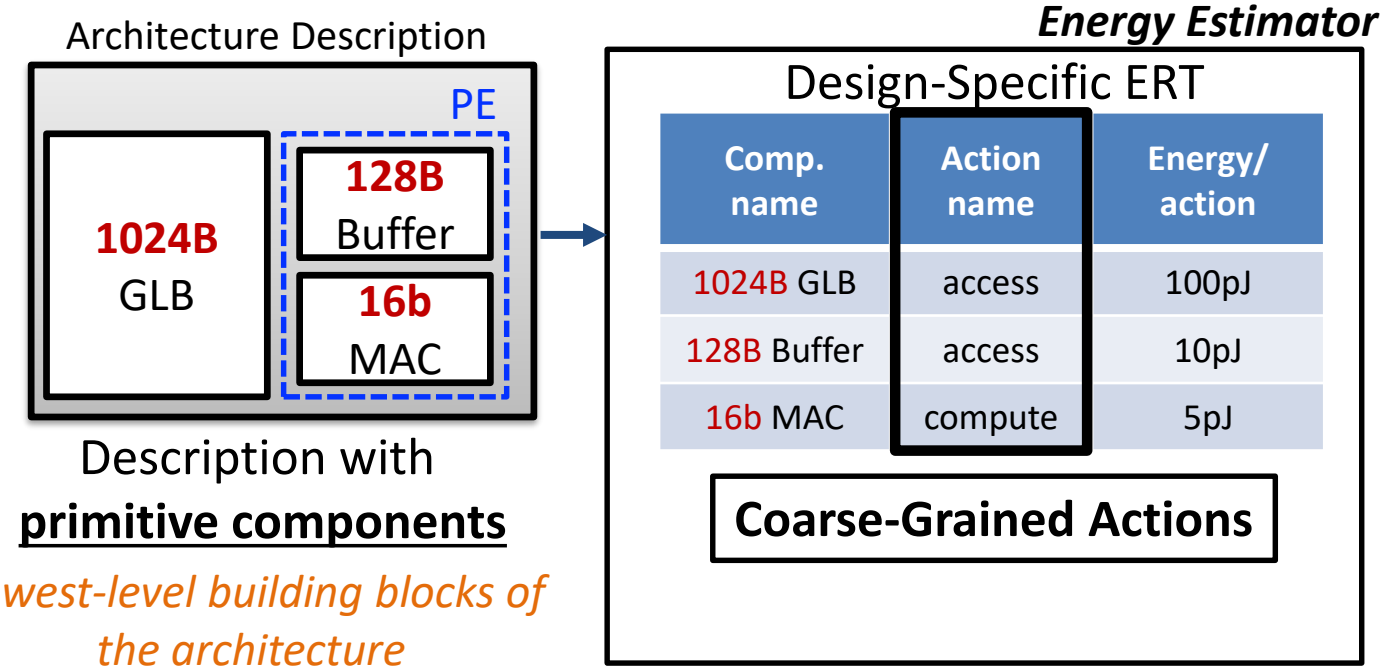
Coarse-grained estimations reduce estimation accuracies

Energy-Per-Actions of a Register File (normalized to idle)



Existing Architecture-Level Energy Estimators

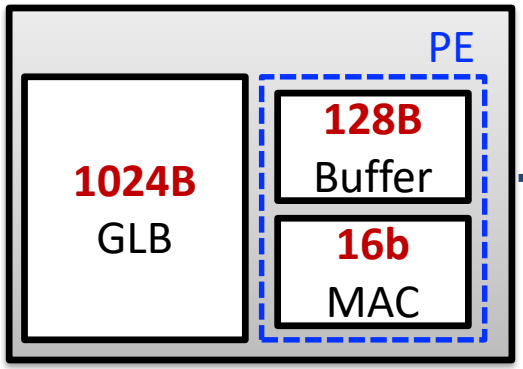
- **Design-Specific** Accelerator Estimators



Existing Architecture-Level Energy Estimators

- **Design-Specific** Accelerator Estimators

Architecture Description



Description with **primitive components**

Lowest-level building blocks of the architecture

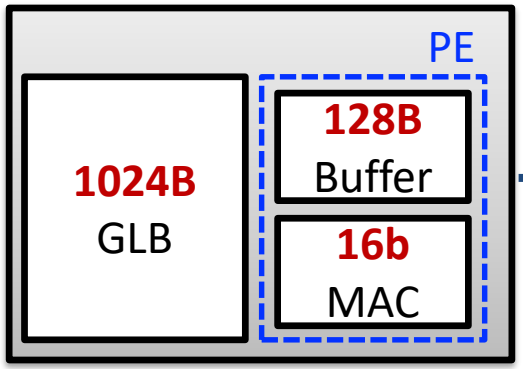
Energy Estimator

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	access	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

Existing Architecture-Level Energy Estimators

- **Design-Specific** Accelerator Estimators



Comp. Name	Action Name	Action Counts
1024B GLB	access	10
128B Buffer	access	800
16b MAC	compute	400

Action Counts

Energy Estimator

Design-Specific ERT

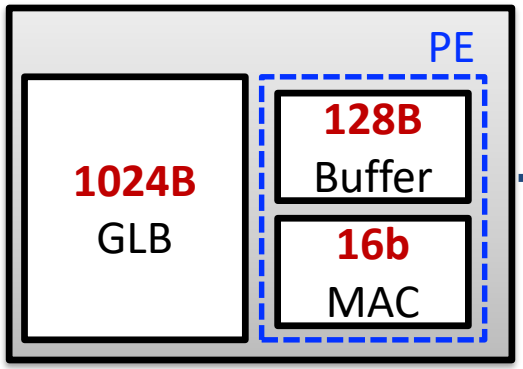
Comp. name	Action name	Energy/ action
1024B GLB	access	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

Energy Calculator

Existing Architecture-Level Energy Estimators

- **Design-Specific** Accelerator Estimators

Architecture Description



Comp. Name	Action Name	Action Counts
1024B GLB	access	10
128B Buffer	access	800
16b MAC	compute	400

Action Counts

Energy Estimator

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	access	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

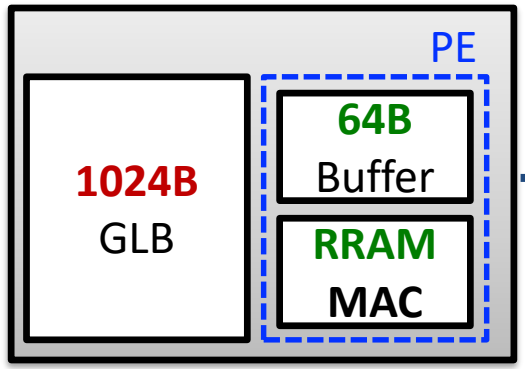
Energy Calculator

Energy Estimations

Name	Energy
128B Buffer	1000pJ
1024B GLB	8000pJ
16b MAC	2000pJ

Existing Architecture-Level Energy Estimators

- **Design-Specific** Accelerator Estimators



Comp. Name	Action Name	Action Counts
1024B GLB	Access	10
64B Buffer	Access	800
RRAM MAC	Compute	400

Action Counts

Energy Estimator

Design-Specific ERT

Comp. name	Action name	Energy/ action
1024B GLB	access	100pJ
128B Buffer	access	10pJ
16b MAC	compute	5pJ

Energy Calculator

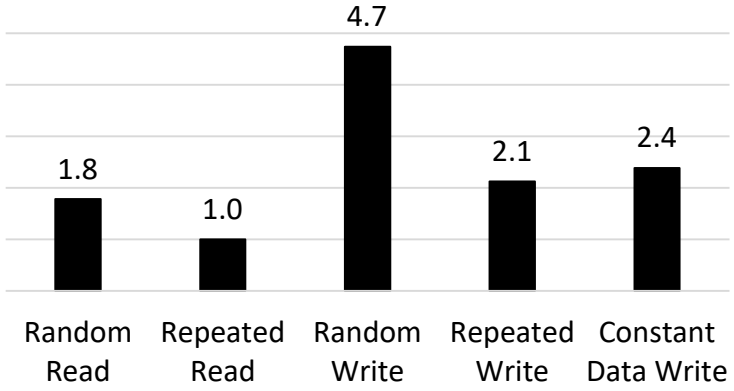
Not generalizable to other designs

Accelergy Overview

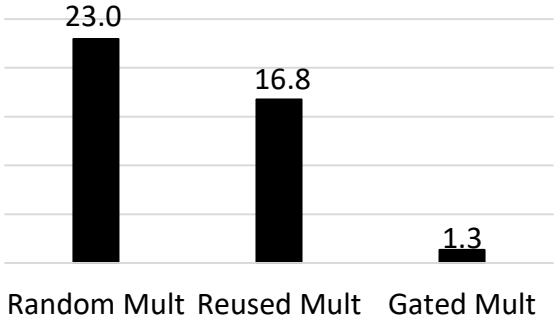
- An architecture-level energy estimator that can be easily integrated into other design exploration tools
- **Accurately and systematically captures the energy consumption of basic building blocks in the design**
- Succinctly models diverse and complicated designs
- Achieves 95% accuracy in evaluating a well-know deep neural network accelerator design – Eyeriss [2016 ISSCC].

Accelergy: Primitive Component Energy Estimation

- **Accurate estimation for primitive components with a primitive component library* that defines**
  - **Necessary hardware attributes to accurately characterize each component**
  - **Necessary fine-grained actions that improve energy estimation accuracy**



Fine-grained memory action types



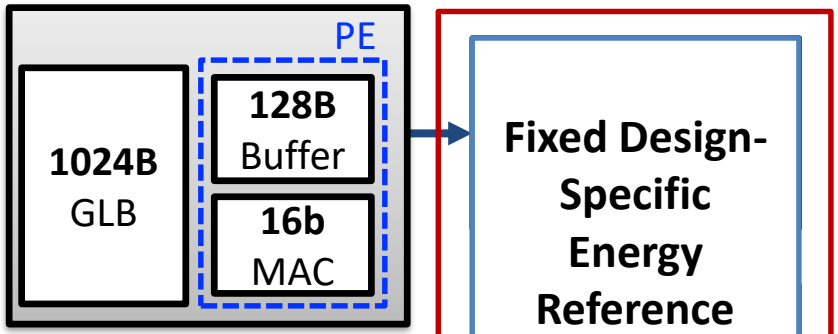
Fine-grained multiplier action types

*open-source at http://acelergy.mit.edu

Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs

Architecture Description

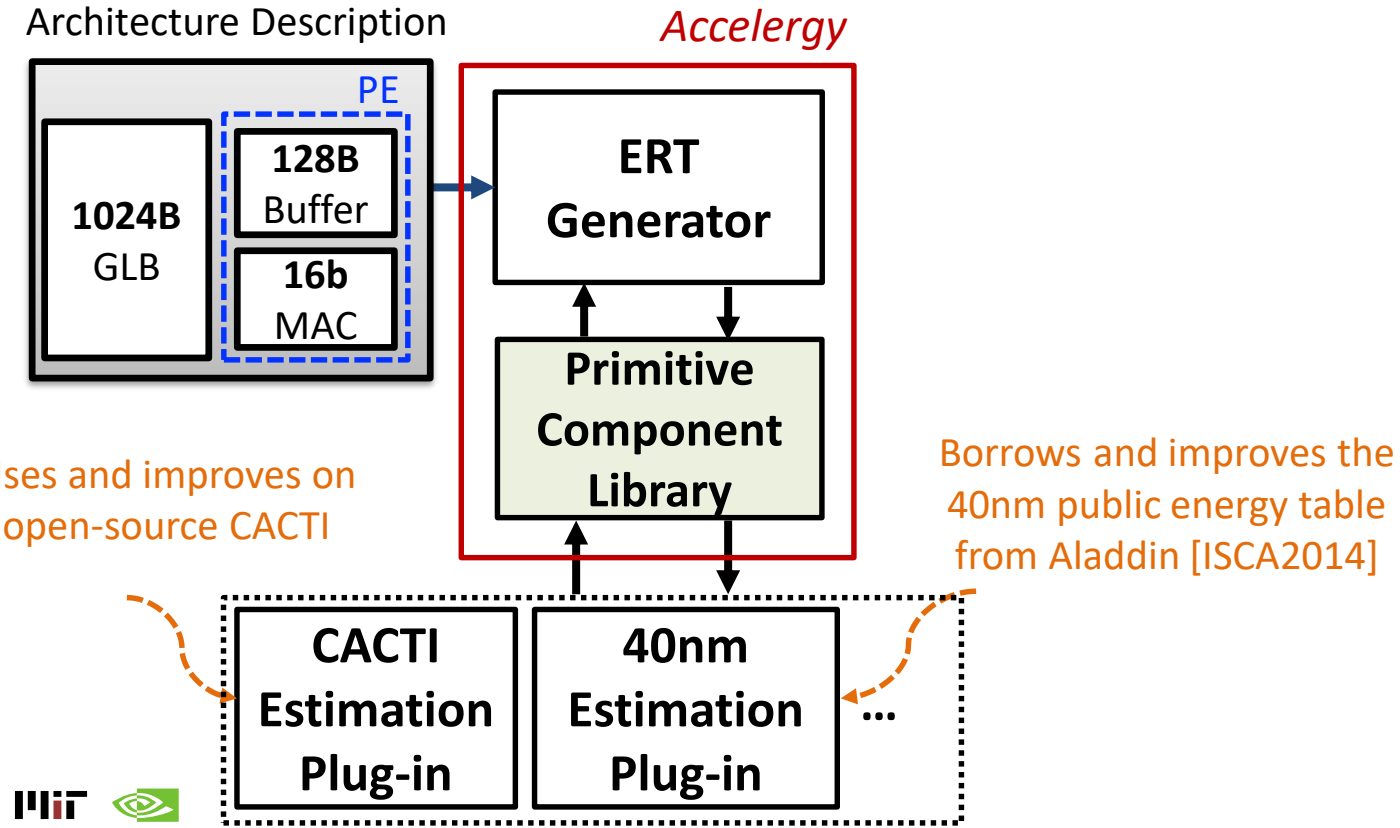


Estimator

Fixed Design-Specific Energy Reference Table (ERT)

Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



Accelergy: Primitive Component Energy Estimation

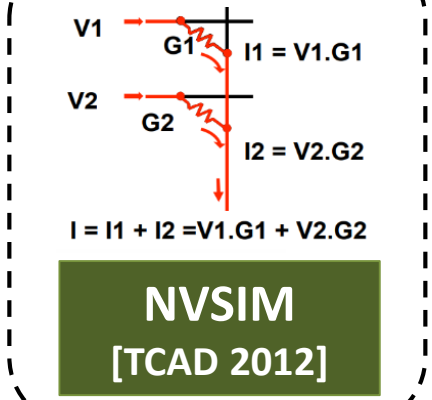
Use energy estimation plug-ins to characterize primitive components

CACTI Estimation Plug-in

40nm Estimation Plug-in

Traditional open-source plug-ins*

Proprietary plug-ins

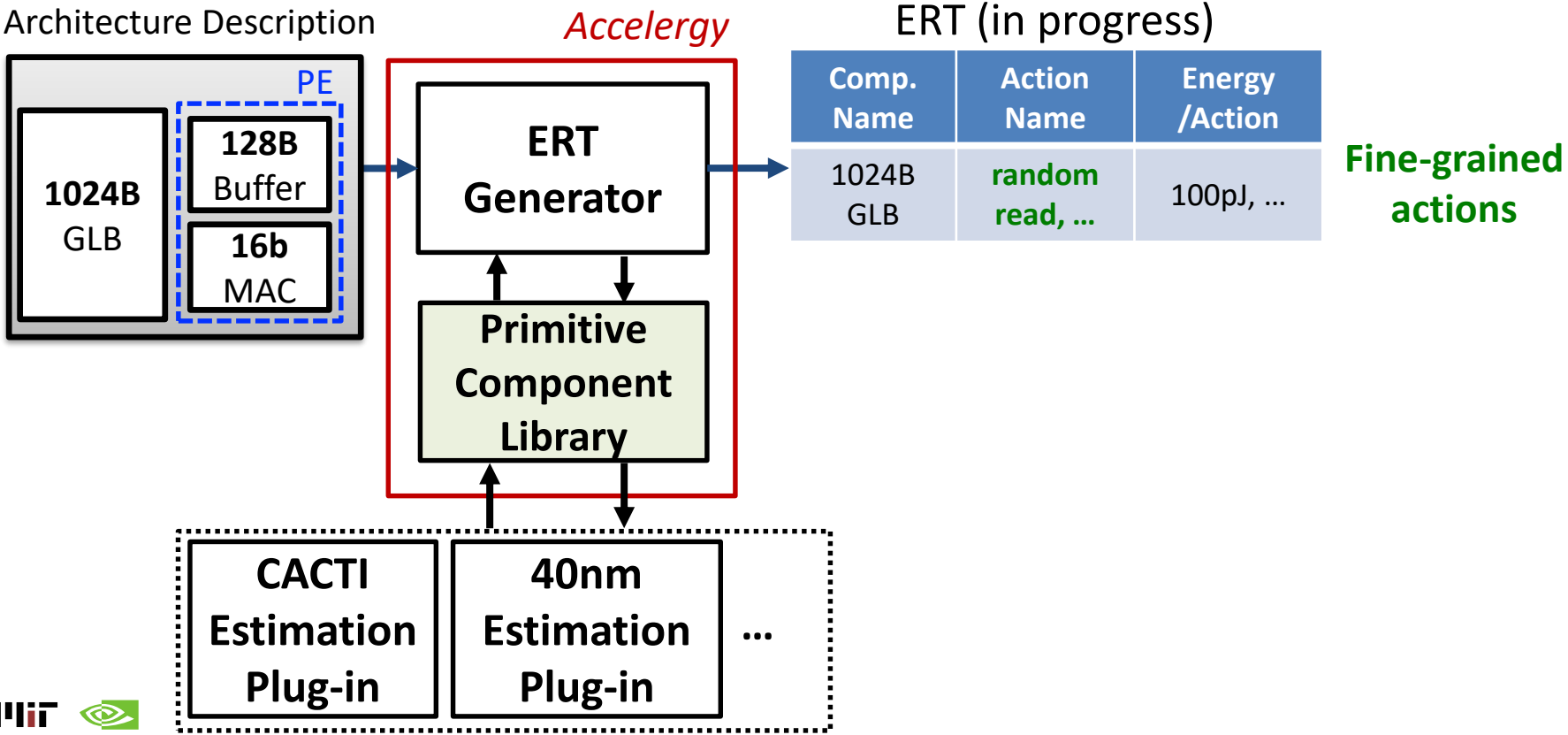


Emerging technology plug-ins

**available at* http://accelergy.mit.edu

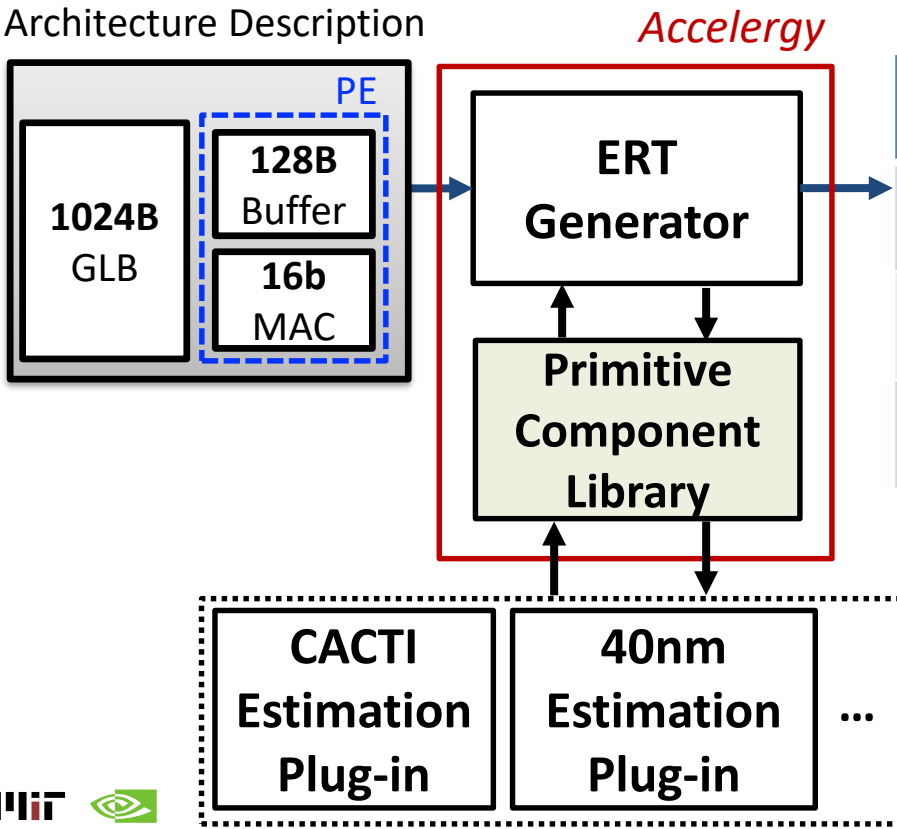
Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



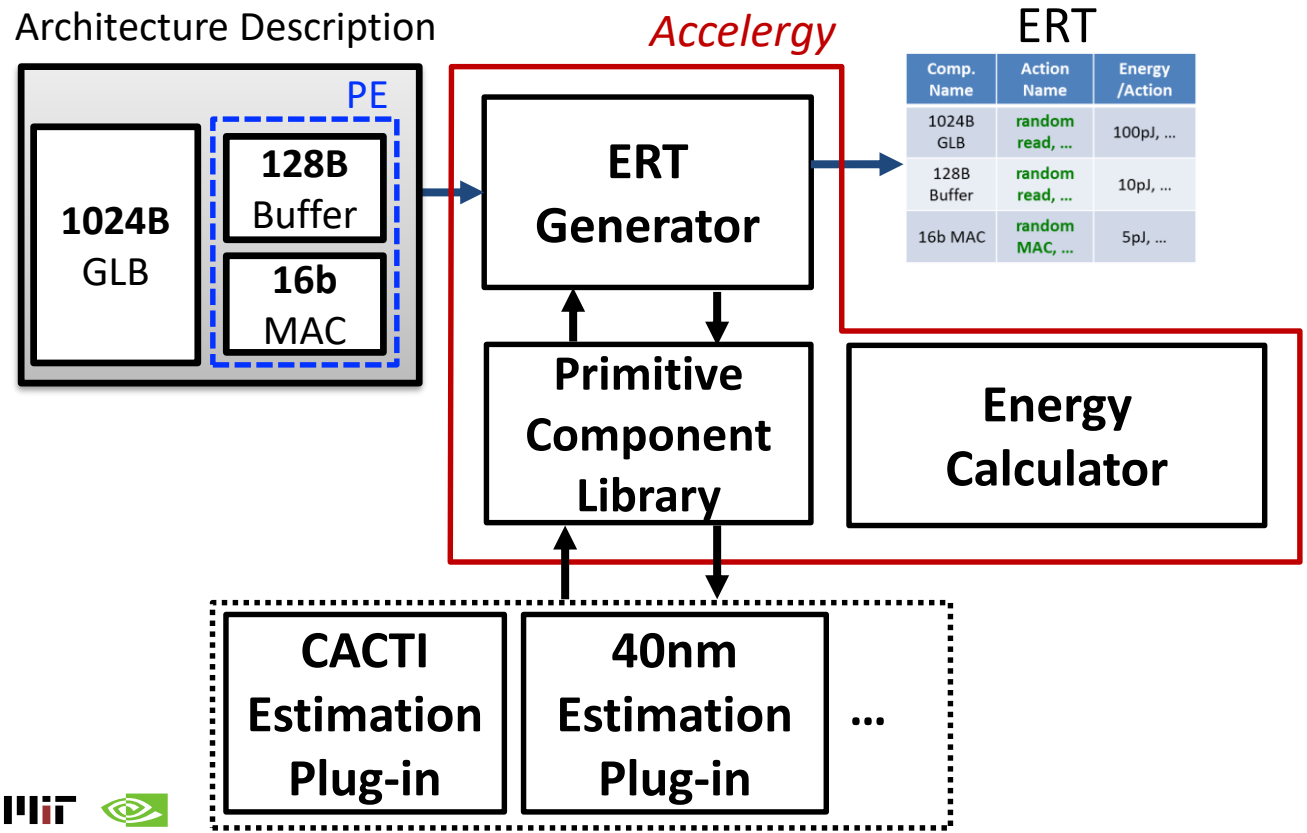
ERT

Comp. Name	Action Name	Energy /Action
1024B GLB	random read, …	100pJ, …
128B Buffer	random read, …	10pJ, …
16b MAC	random MAC, …	5pJ, …

Fine-grained actions

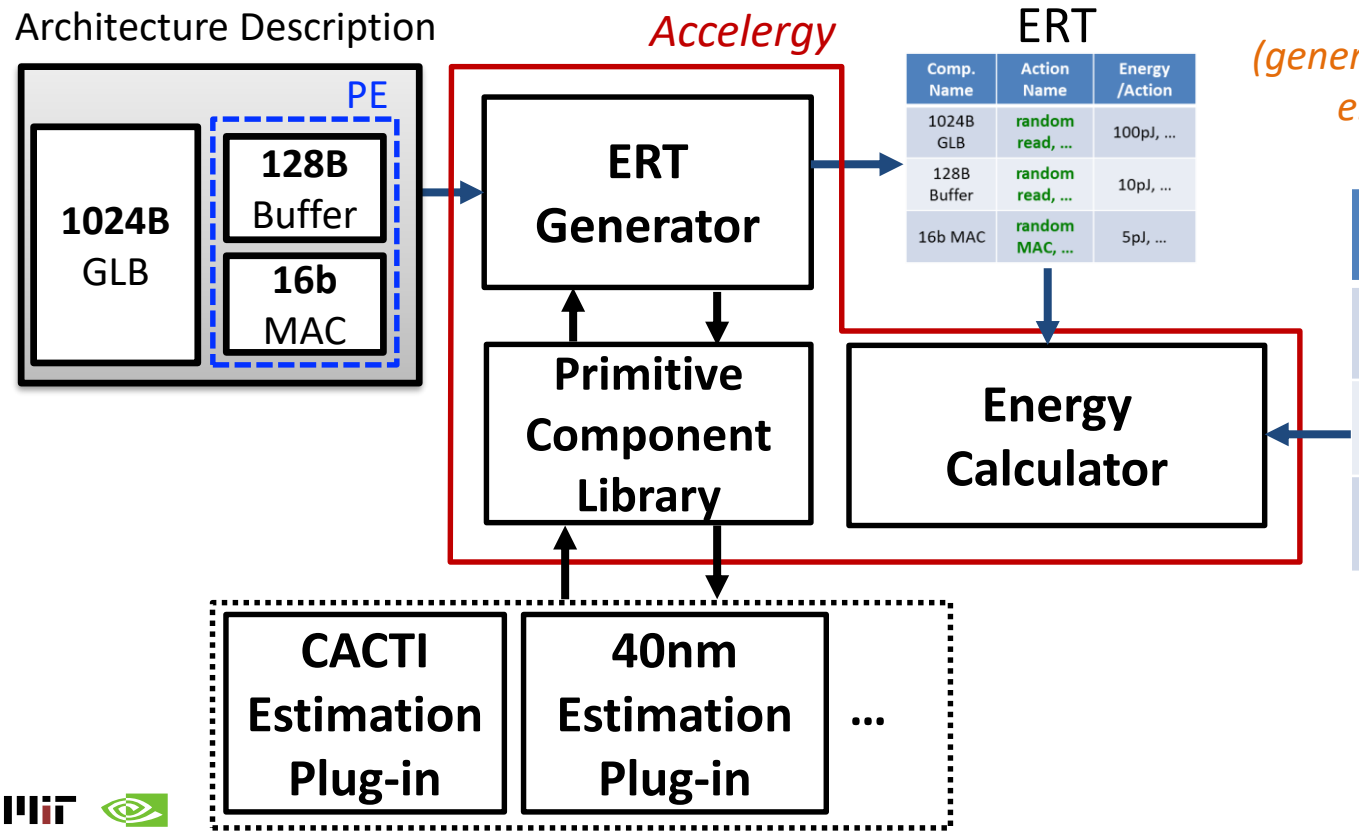
Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



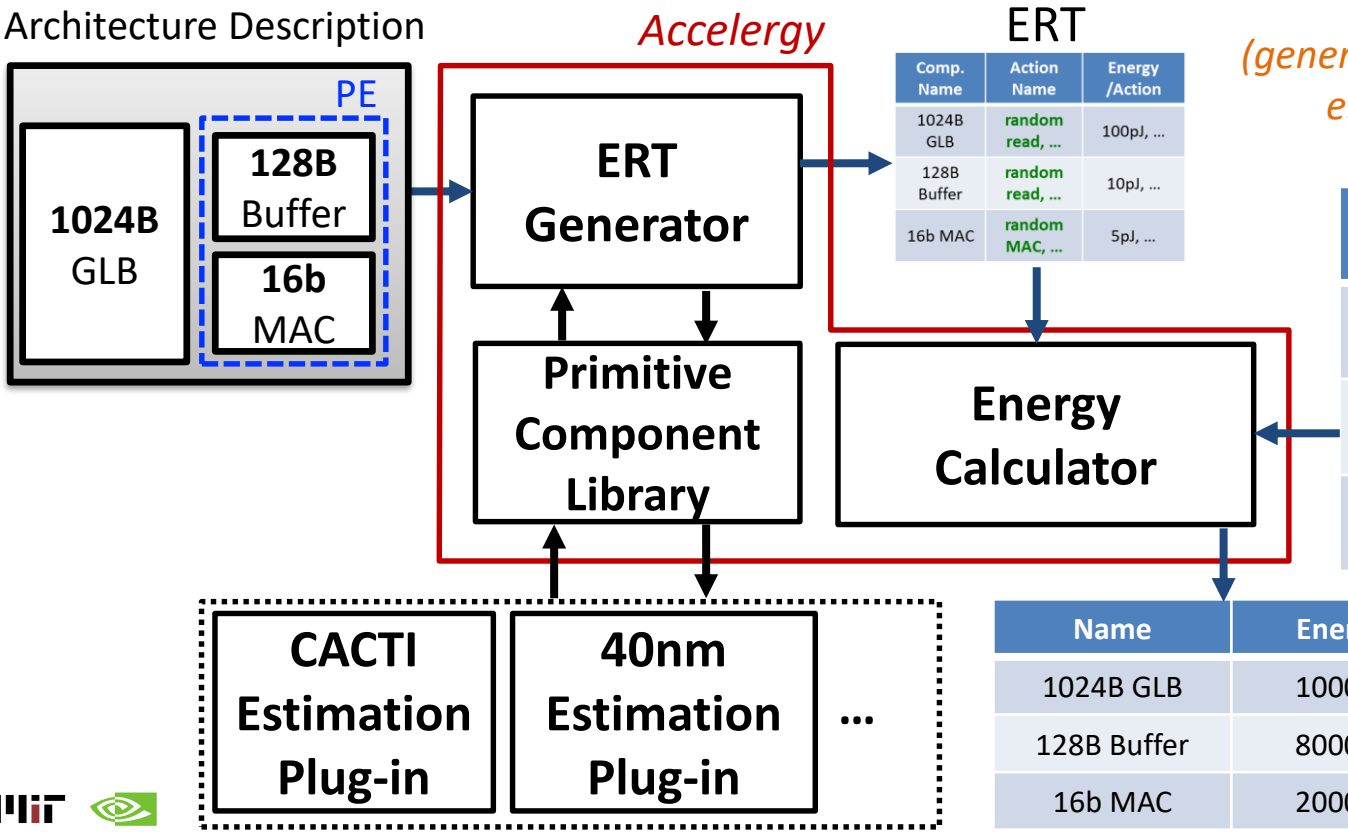
Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



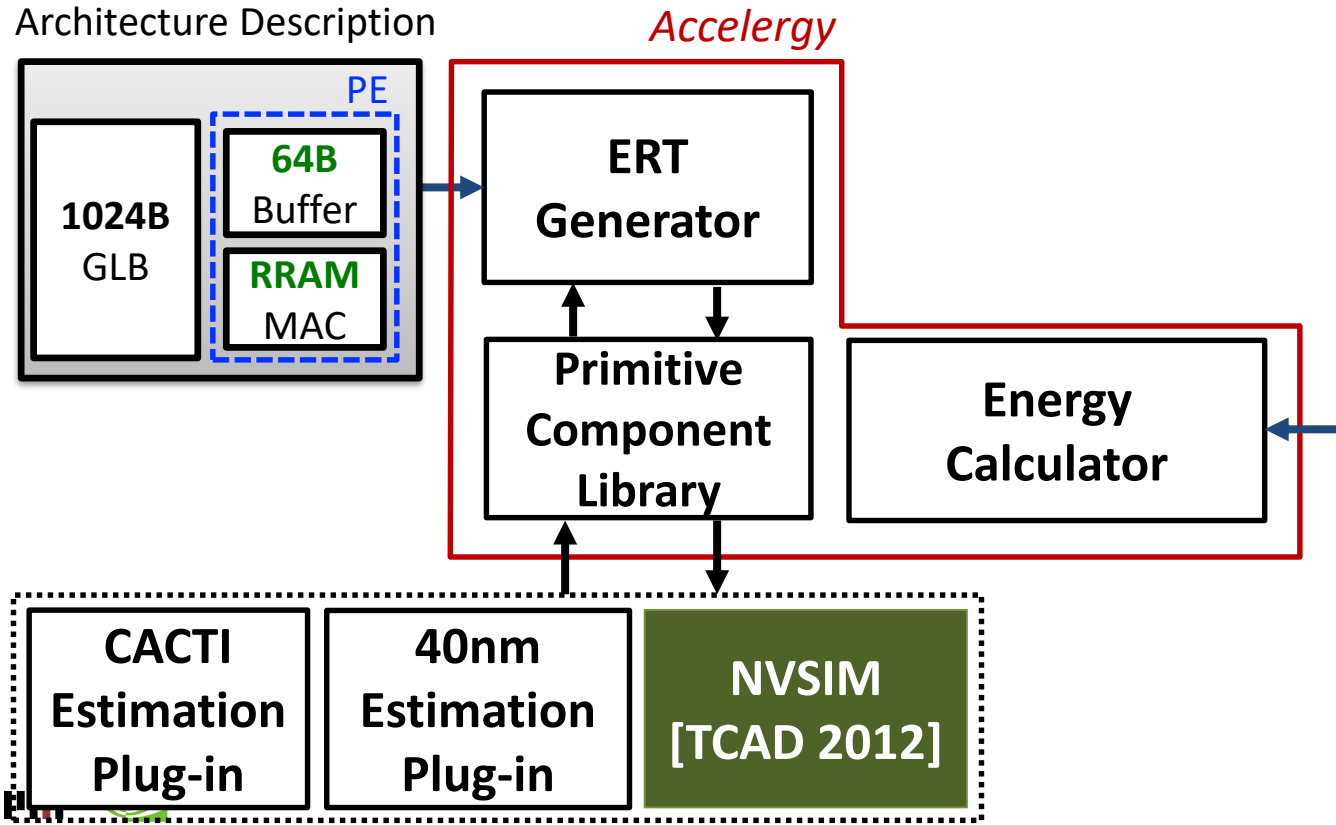
Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs



Accelergy: Primitive Component Energy Estimation

Systematically generates ERTs for primitive component in various designs

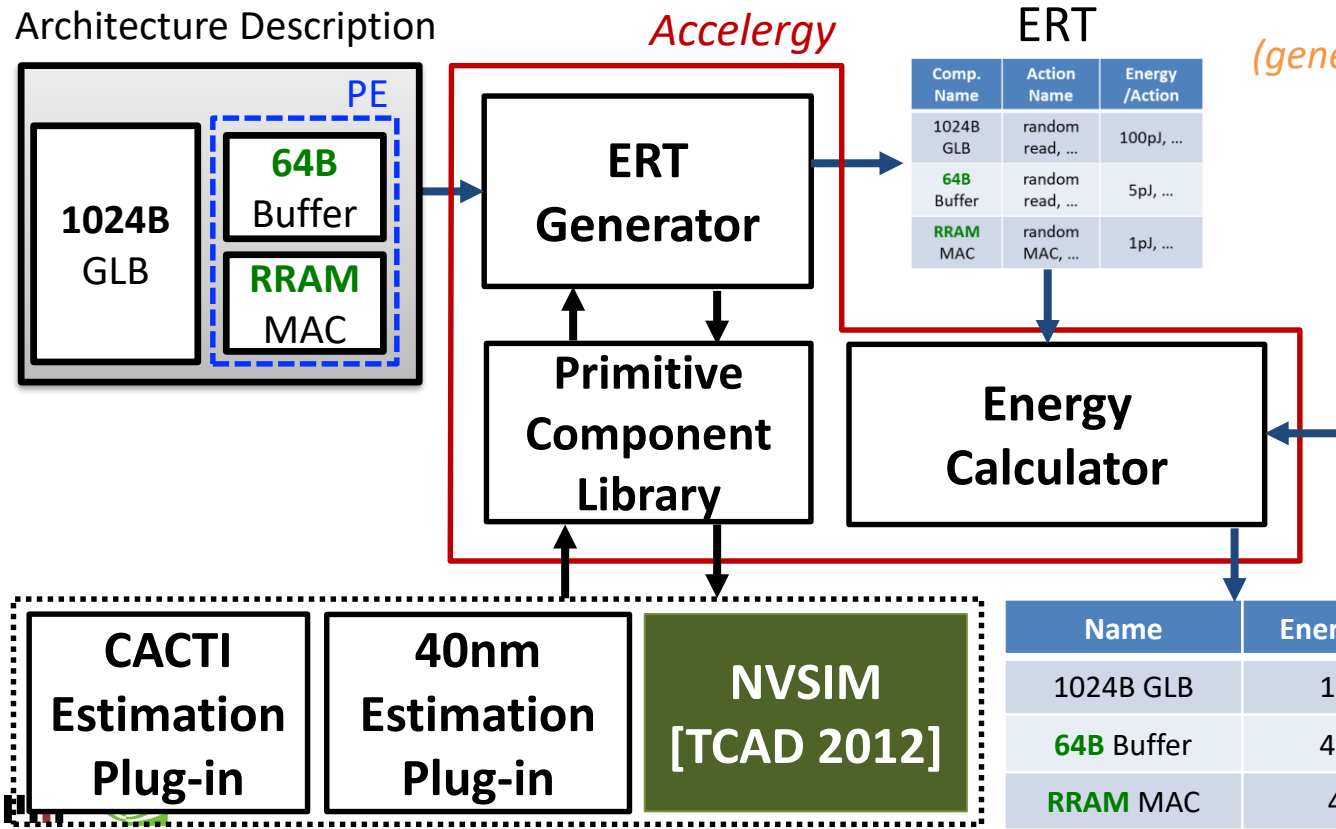


Action Counts

Comp. Name	Action Name	Action Counts
1024B GLB	random read	10
64B Buffer	random read	800
RRAM MAC	random MAC	400

Accelergy: Primitive Component Energy Estimation

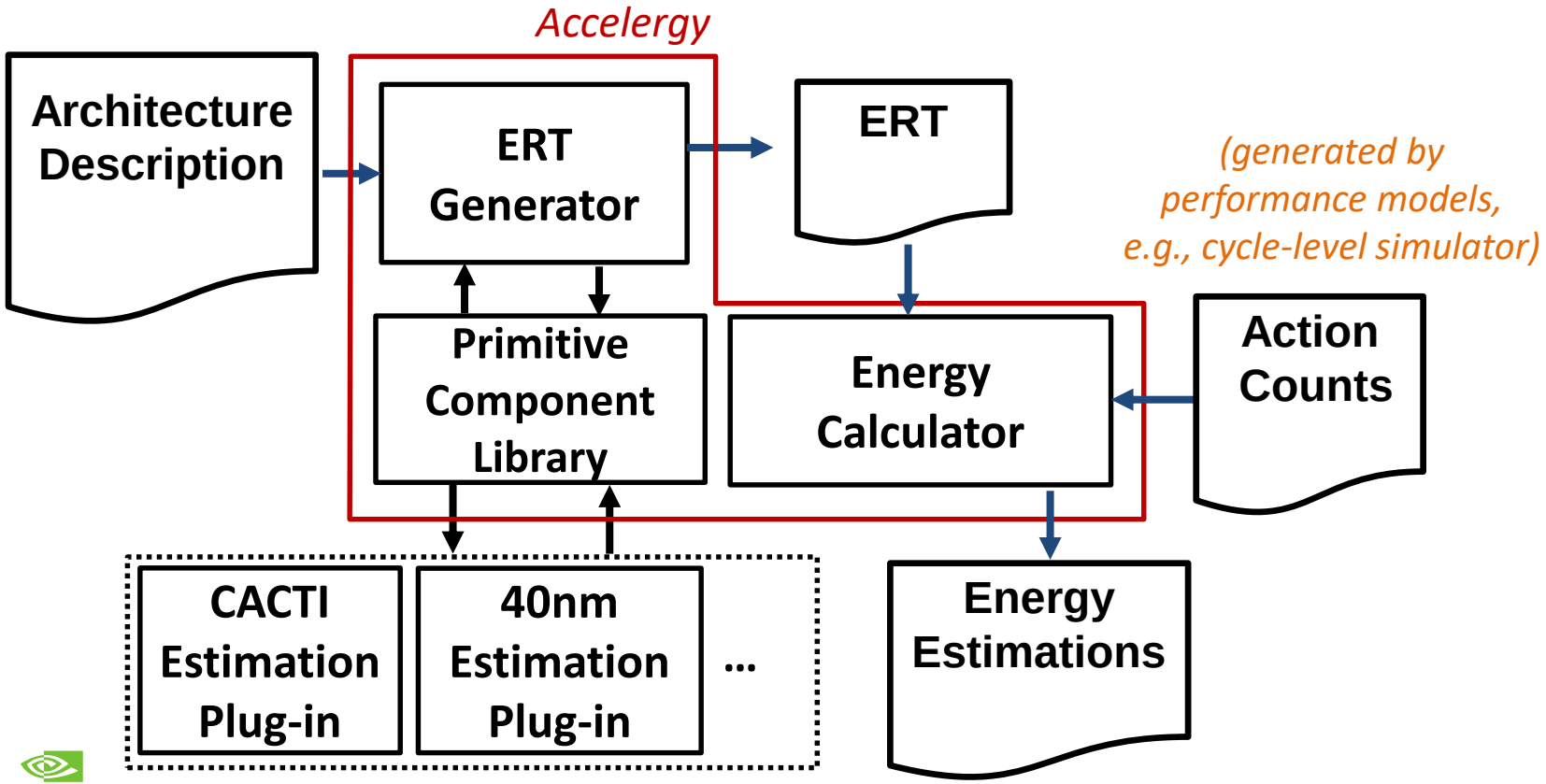
Systematically generates ERTs for primitive component in various designs



How to use Accelergy?

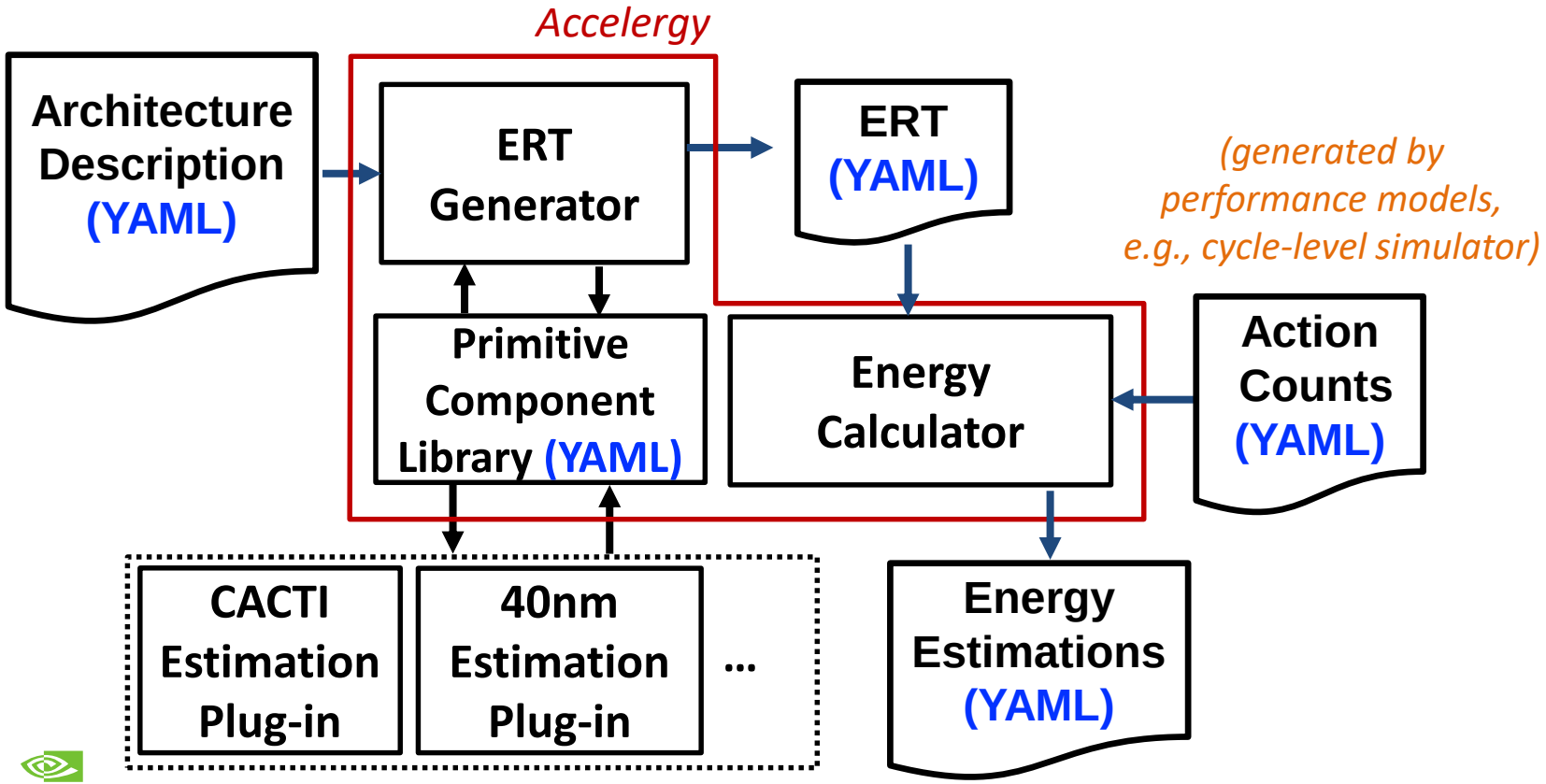
1. Estimate architectures with primitive components

Estimate Architectures with Primitive Components



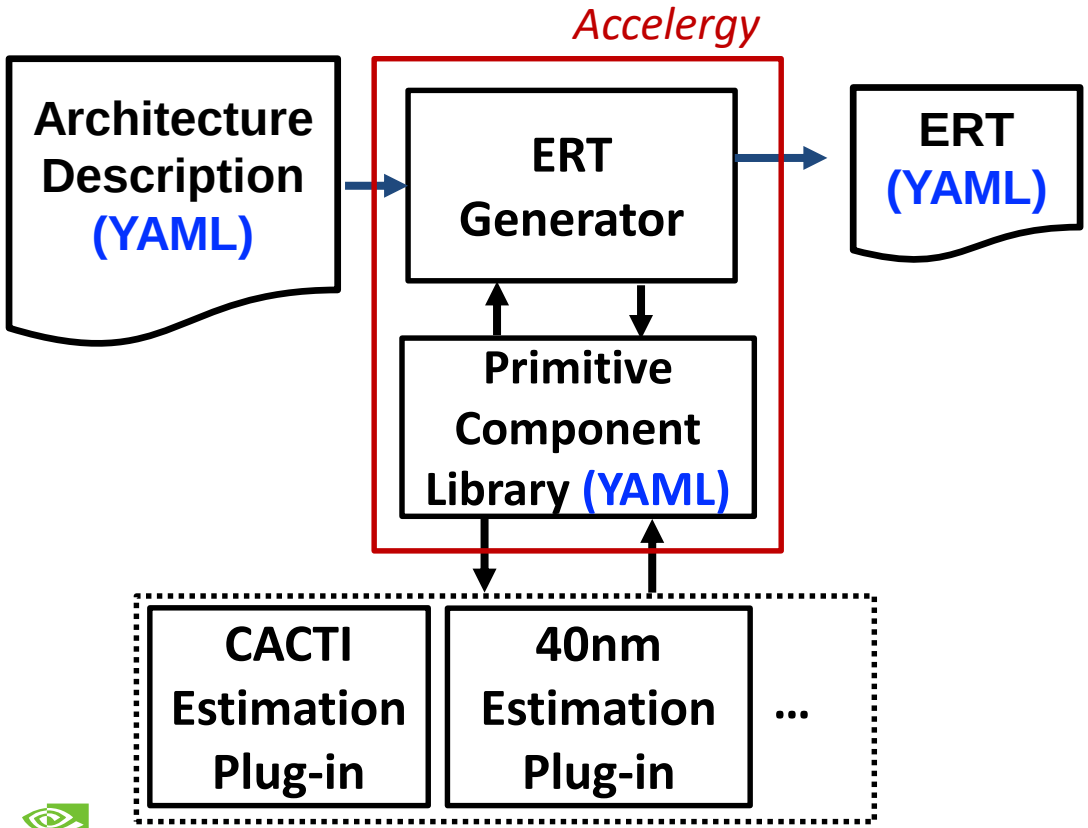
Estimate Architectures with Primitive Components

Accepts inputs and generates outputs in YAML format



Estimate Architectures with Primitive Components

Step1: Generate Energy Reference Tables



Accelergy: Primitive Component Library

- Contains various types of primitive component classes
- Each primitive component class contains attribute names and action names that can be shared by its instances

Object-Oriented Approach
Components are **instances** of component classes

YAML Syntax

```
version: 0.2 # vcs
classes:
- name: bitwise
  attributes:
     technology
     datawidth
  actions:
    - name: process …
- name: intadder
  attributes:
     technology
     datawidth
  actions:
    - name: add …
```

Accelergy: Primitive Component Library

- Contains various types of primitive component classes
- Each primitive component class contains attribute names and action names that can be shared by its instances
- Default attributes are defined

Object-Oriented Approach
Components are **instances** of component classes

YAML Syntax

```
version: 0.2 # vcs
classes:
- name: bitwise
  attributes:
      technology: 40nm
      datawidth: 1 …
 actions:
     - name: process …
- name: intadder
  attributes:
      technology: 40nm
      datawidth: 16 …
  actions:
     - name: add …
```

Accelergy: Primitive Component Library

- Contains various types of primitive component classes
- Each primitive component class contains attribute names and action names that can be shared by its instances
- Default attributes are defined

Object-Oriented Approach
Components are **instances** of component classes

YAML Syntax

```
version: 0.2 # vcs
classes:
- name: bitwise
  attributes:
    technology: 40nm
    datawidth: 16
  actions:
    - name: process …
- name: intadder
  attributes:
    technology: 40nm
    datawidth: 16
  actions:
    - name: add …
```

class

data members

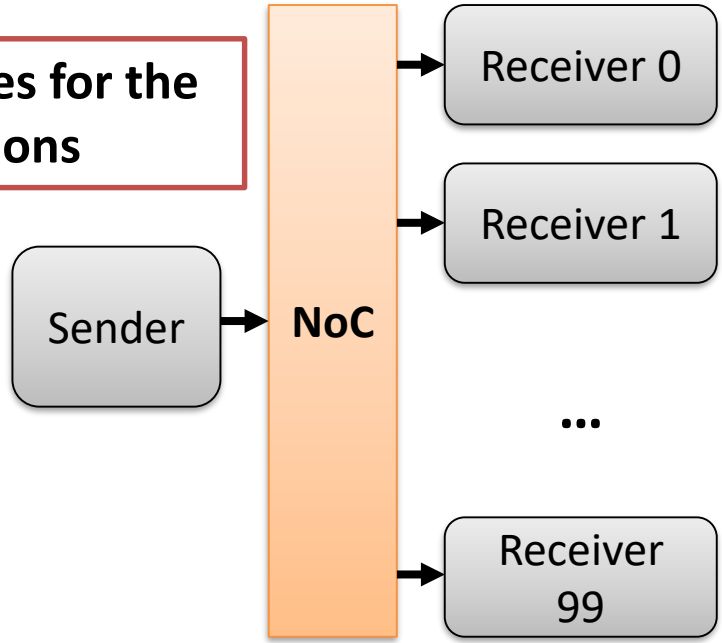
member functions

Accelergy: Primitive Component Library

- **Representation of fine-grained action types**

Action type
Repeated read
Random read
Repeated write
Random write
Repeated data write

We need 100 names for the multicast actions



Need a new name for each fine-grained action?

Tedious

Accelergy: Primitive Component Library

- **Fine-grained actions with runtime arguments**

Original action type	New Action type
Repeated read	read
Random read	
Repeated write	write
Random write	
Repeated data write	

YAML Syntax

```
- name: regfile
  attributes: …
  actions:
    - name: read


    - name: write
```

Accelergy: Primitive Component Library

- **Fine-grained actions with runtime arguments**

Original action type	New Action type	Argument	
		dataΔ	addressΔ
Repeated read	read	0	0
Random read		1	1
Repeated write	write	0	0
Random write		1	1
Repeated data write		0	1

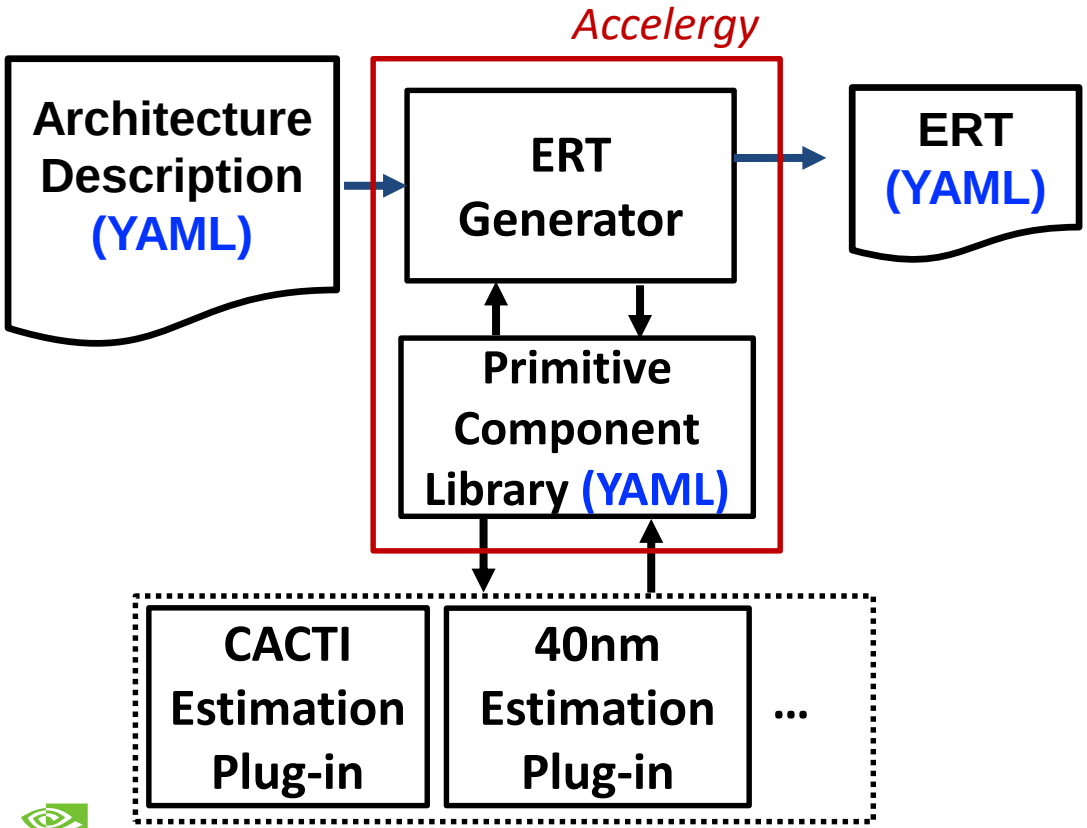
YAML Syntax

```
- name: regfile
  attributes: …
  actions:
    - name: read
      arguments:
        data_delta: 0..1
        address_delta: 0..1
    - name: write
      arguments:
        data_delta: 0..1
        address_delta: 0..1
```

0≤ data_delta ≤1

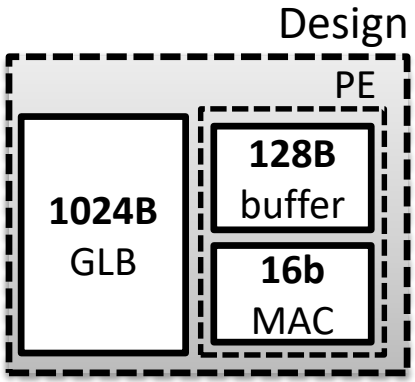
Estimate Architectures with Primitive Components

Step1: Generate Energy Reference Tables



Accelergy: Architecture Description

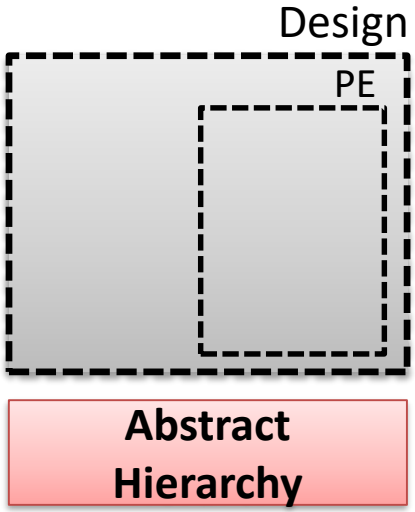
- **Accelergy shares similar architecture description syntax with Timeloop**

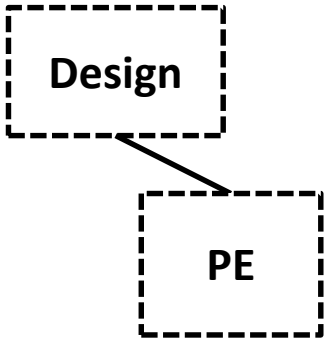


YAML Syntax

Accelergy: Architecture Description

- Accelergy shares similar architecture description syntax with Timeloop





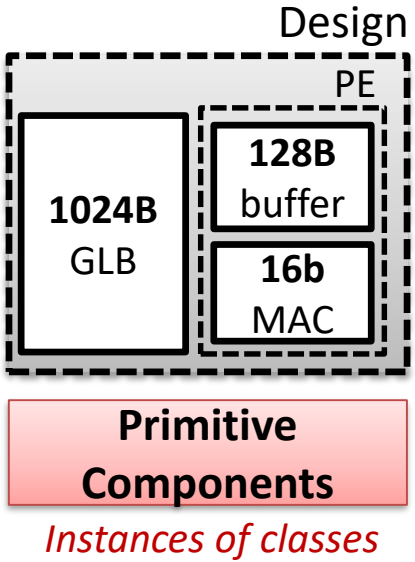
YAML Syntax

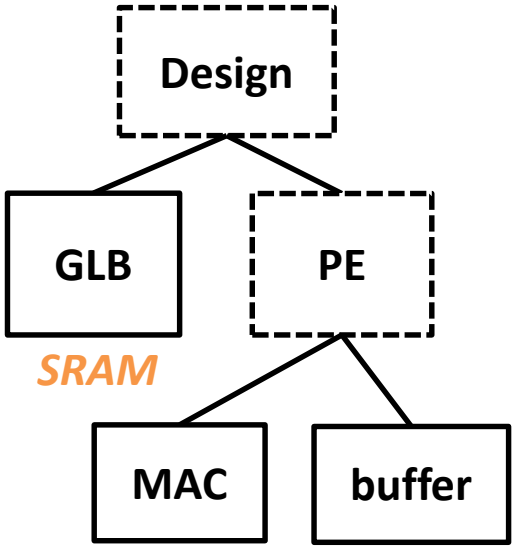
```
subtree:
 - name: Design
   attributes:
     technology: 40nm

   subtree:
   - name: PE
```

Shared Attributes

Accelergy: Architecture Description

- Accelergy shares similar architecture description syntax with Timeloop





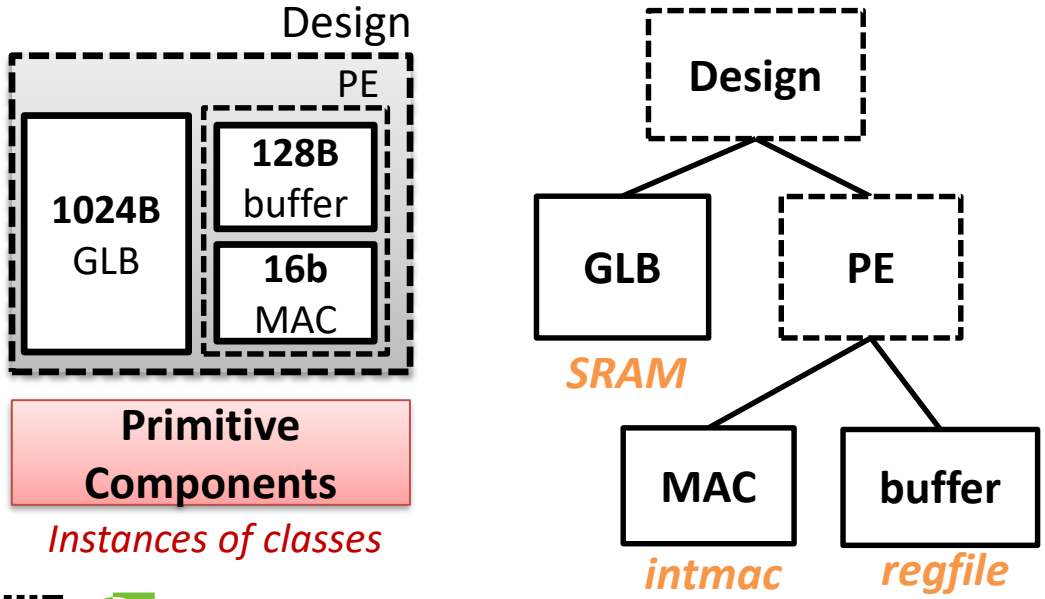
YAML Syntax

```
subtree:
 - name:simple_primitive
 attributes:
    technology: 40nm
    local:
    - name: GLB
      class: SRAM
      attributes:
          width: 32 …
    subtree:
    - name: PE
      local:
        - name: Buffer

        - name: MAC
```

An Instance of the SRAM class

Accelergy: Architecture Description

- Accelergy shares similar architecture description syntax with Timeloop

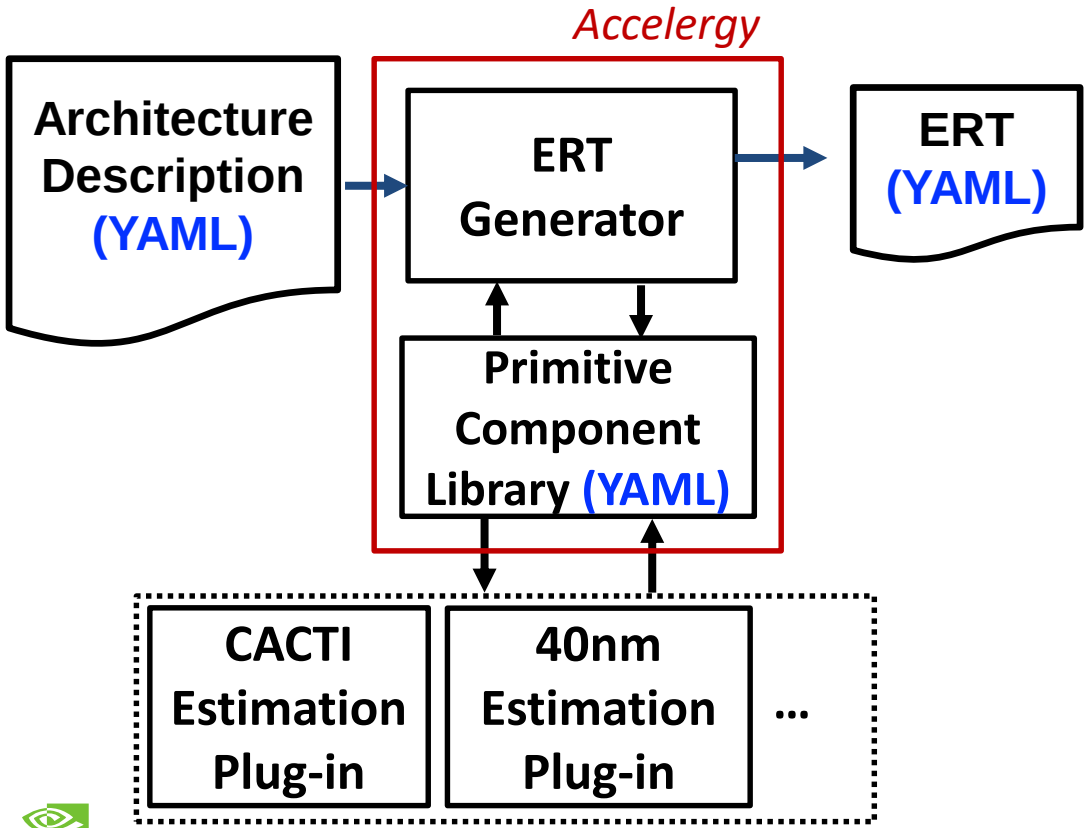


YAML Syntax

```
subtree:
 - name:simple_primitive
 attributes:
    technology: 40nm
    local:
    - name: GLB
      class: SRAM
      attributes:
         width: 32 …
    subtree:
    - name: PE
      local:
       - name: buffer
         class: regfile
       - name: MAC
         class: intmac
```

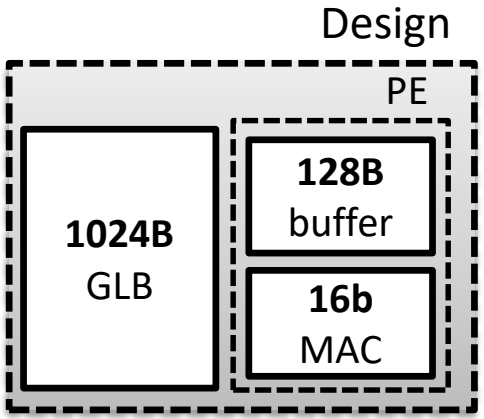
Estimate Architectures with Primitive Components

Step1: Generate Energy Reference Tables



Accelergy: Energy Reference Table (ERT)

- **Contains energy/action values for all the components in the designs**

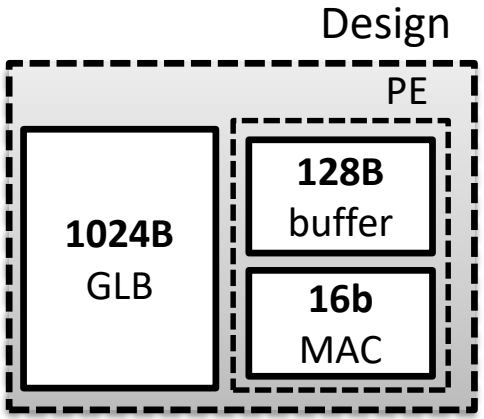


YAML Syntax

```
tables:
- name:Design.GLB

- name: Design.PE.buffer

- name: Design.PE.MAC
```

Accelergy: Energy Reference Table (ERT)

- **Contains energy/action values for all the components in the designs**



YAML Syntax

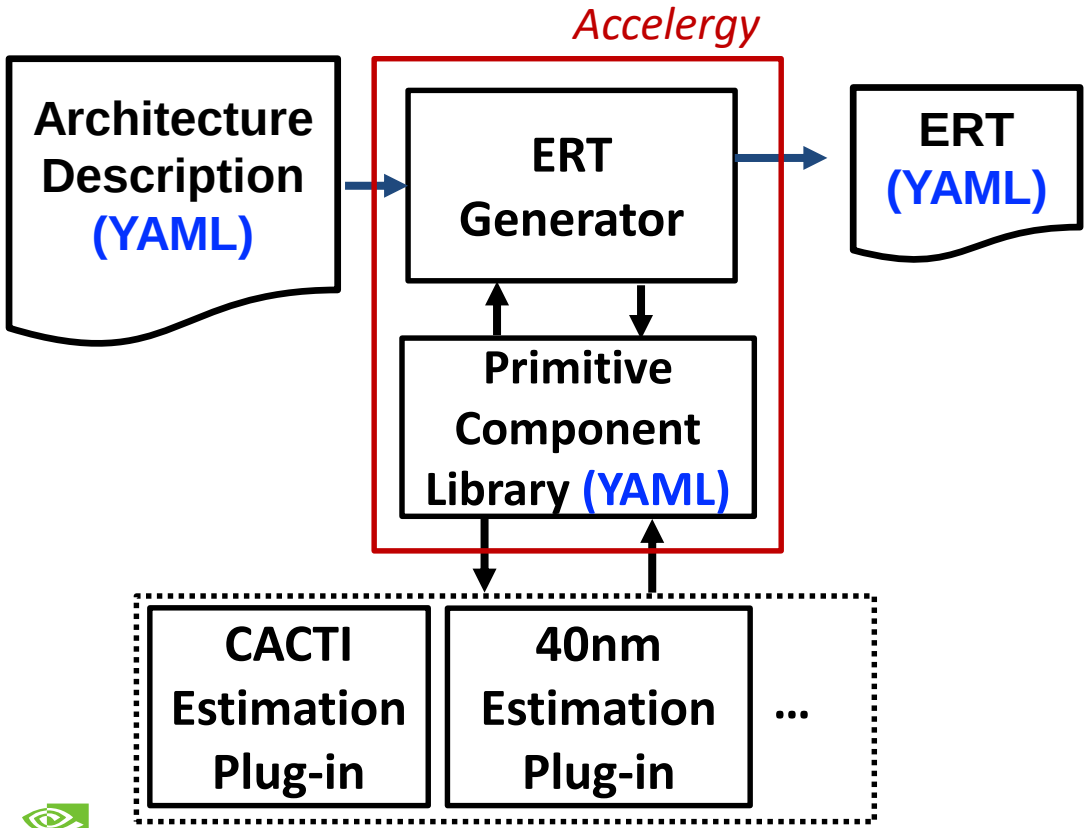
```
tables:
- name:Design.GLB
  read:
    - arguments:
        data_delta: 1
        address_delta: 1
      energy: 100
  …
- name: Design.PE.buffer
  read: …
- name: Design.PE.MAC
  mac_random:
    energy: 5
  …
```

Exercise 1: Generate ERT for Simple Architecture

- **Instructions:**
  - `cd exercises/accelergy/01_primitive_architecute_ERT`
  - `accelergy  input/*.yaml  -o output/ --enable_flattened_arch 1`
  - ERT.yaml, ERT_summary.yaml, and flattened_architecture will be generated in current directory
- **Questions**:
  - Each YAML file contains a top-key that indicates its content. What top-level keys are used to identify the input file content?
  - What information in ERT is used for Timeloop?
  - Examine ERT_summary.yaml, why it's bad to have coarse-grained action type?

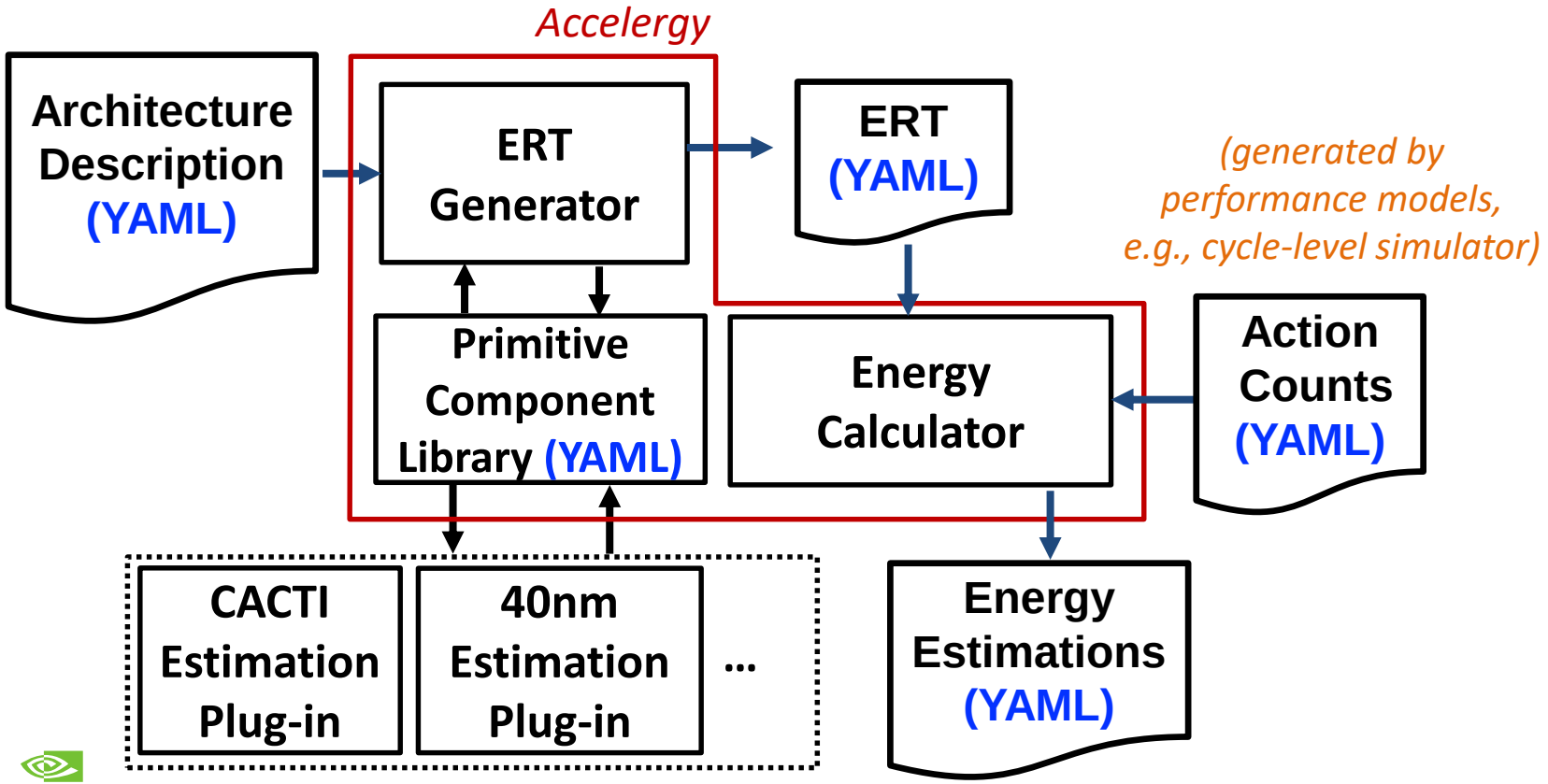
Estimate Architectures with Primitive Components

Step1: Generate Energy Reference Tables



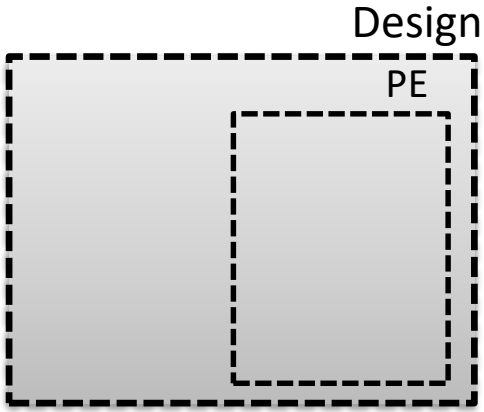
Estimate Architectures with Primitive Components

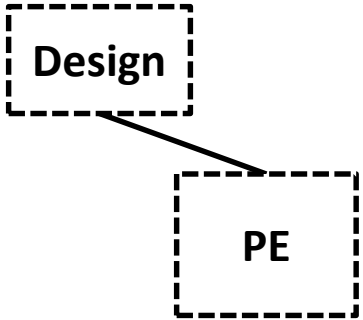
Step2: Generate Energy Estimations



Accelergy: Action Counts

- Action counts are generated using external performance models, such as a cycle-level simulator or Timeloop.



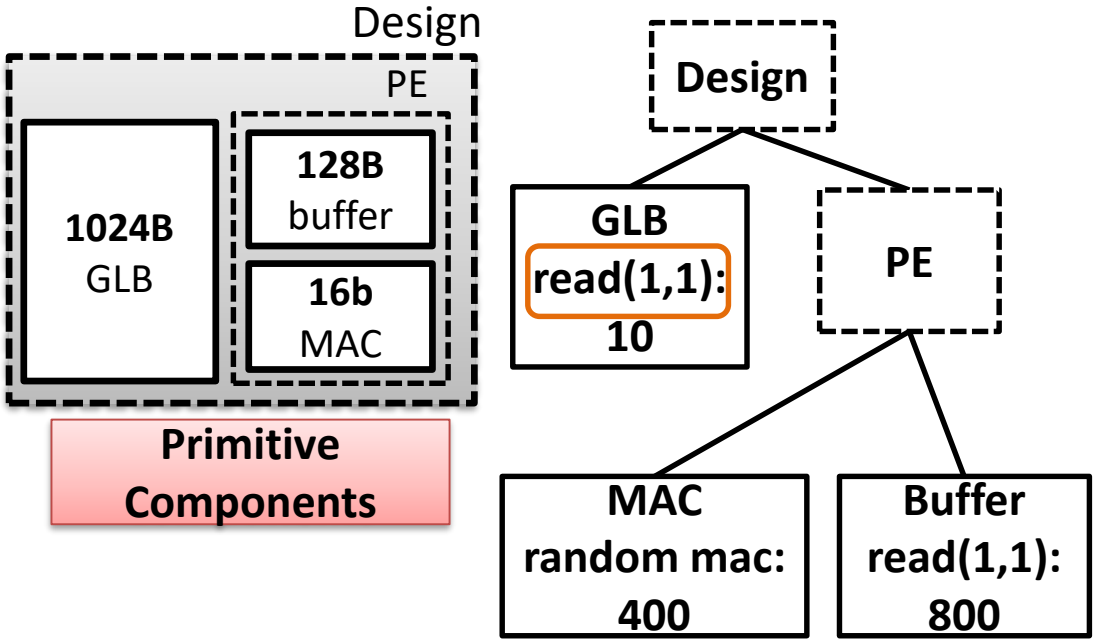


YAML Syntax

```
subtree:
  - name: Design

    subtree:
    - name: PE
```

Accelergy: Action Counts

- Action counts are generated using external performance models, such as a cycle-level simulator or Timeloop.

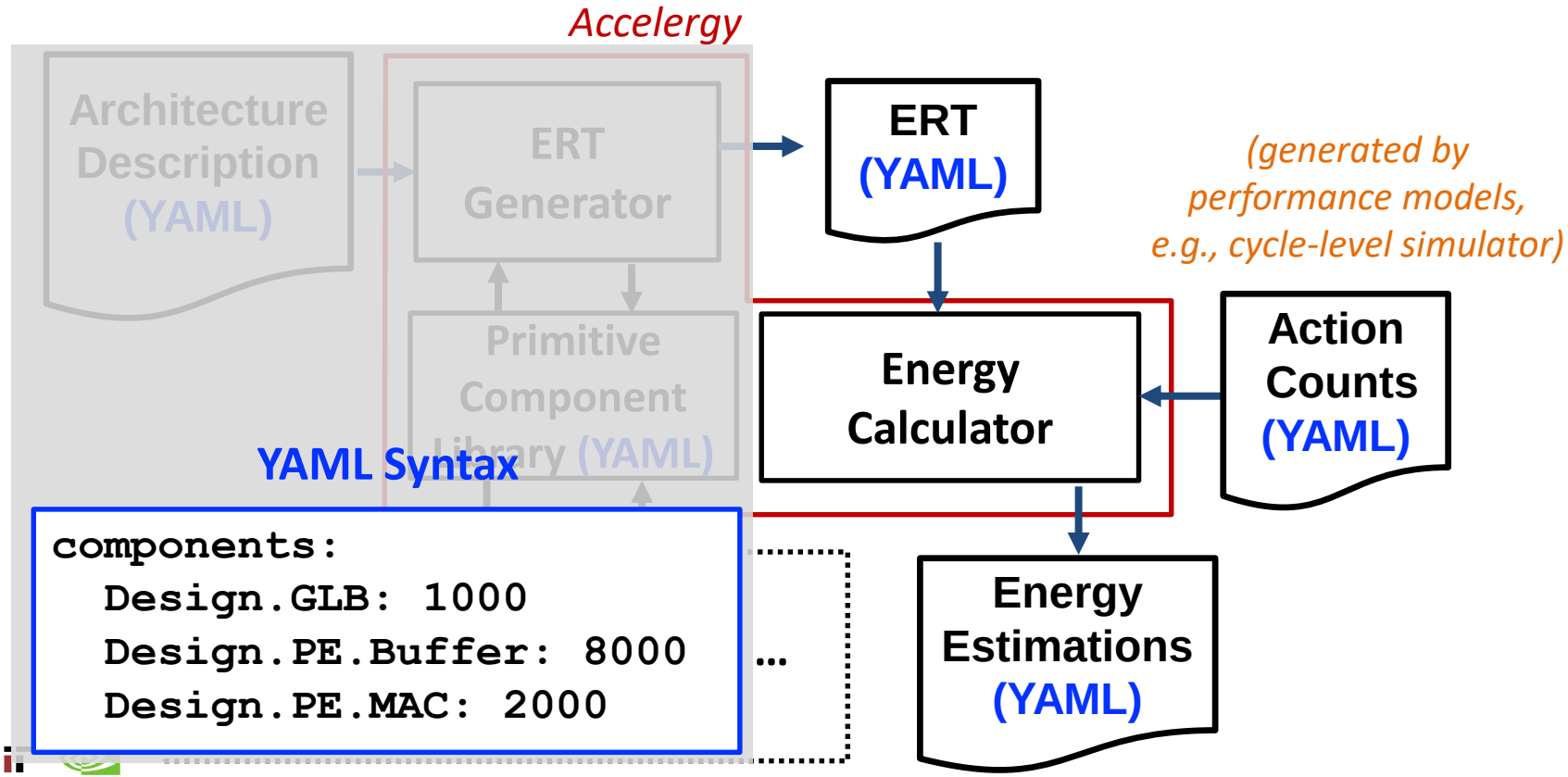


YAML Syntax

```
subtree:
 - name: Design
   local:
   - name: GLB
     action_counts:
       - name: read
         arguments:
           data_delta: 1
           address_delta: 1
         counts: 10
   subtree:
   - name: PE
     local:
      - name: Buffer
        action_counts: …
```

Estimate Architectures with Primitive Components

Accepts inputs and generates outputs in YAML format

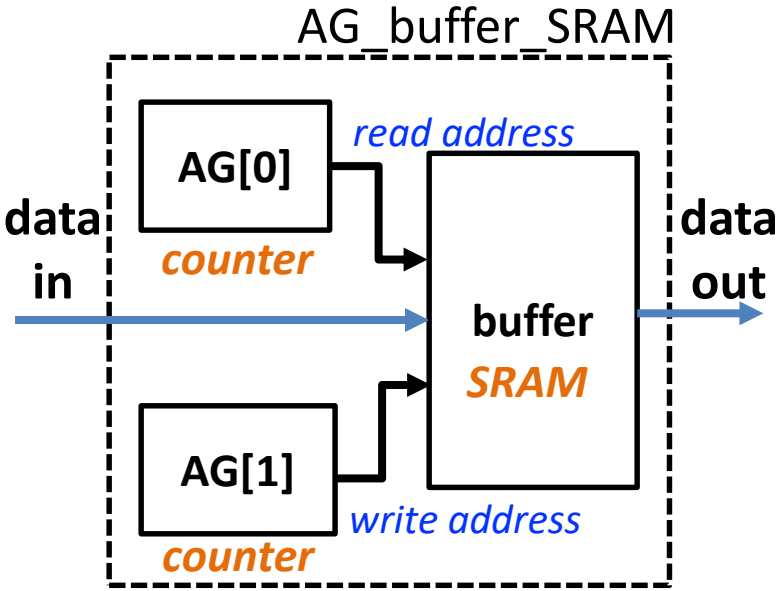


Exercise 2: Generate Energy Estimation for Simple Architecture

- **Instructions:**
  - `cd ../02_primitive_architecute_estimation`
  - `accelergy *.yaml --enable_flattened_arch 1 -o output/`
  - estimation.yaml will be generated in current directory
- **Questions**:
  - Did Accelergy run ERT generator part? Why?
  - Is energy calculation faster than ERT generation?
  - Would adding the architecture description in this directory confuse Accelergy to rerun ERT generator?
  - Why is separating the two parts good?

Modeling Complicated Designs

- **Practical architecture designs involve much more details**
  - **Example: storage units with local address generators (AG)***

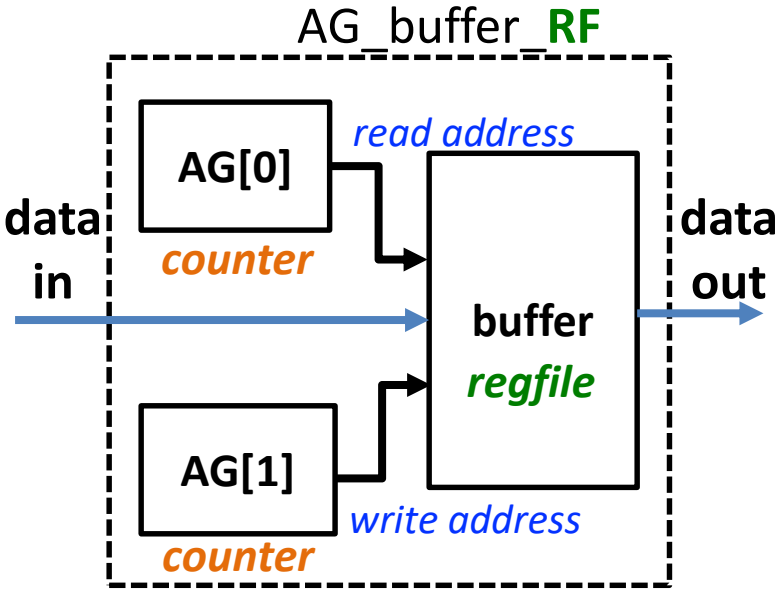


- AG_buffer_SRAM is an abstract hierarchy
- Buffer is an instance of SRAM primitive class
- AGs are instances of counter primitive class

**There are much more complicated storage idioms for explicit data orchestration, e.g., buffet [ASPLOS2019]*

Modeling Complicated Designs

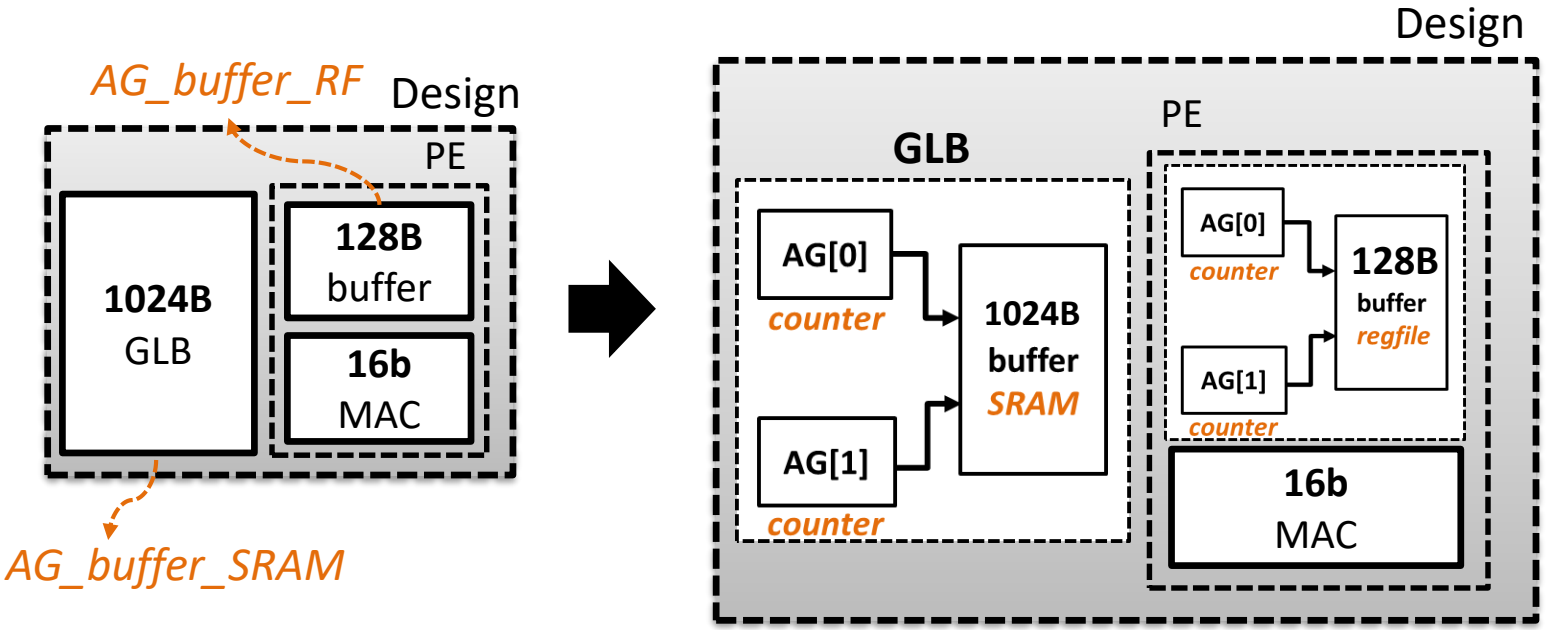
- **Practical architecture designs involve much more details**
  - **Example: storage units with local address generators (AG)**



- AG_buffer_**RF** is an abstract hierarchy
- AGs are instances of counter primitive class
- Buffer is an instance of **regfile** primitive class

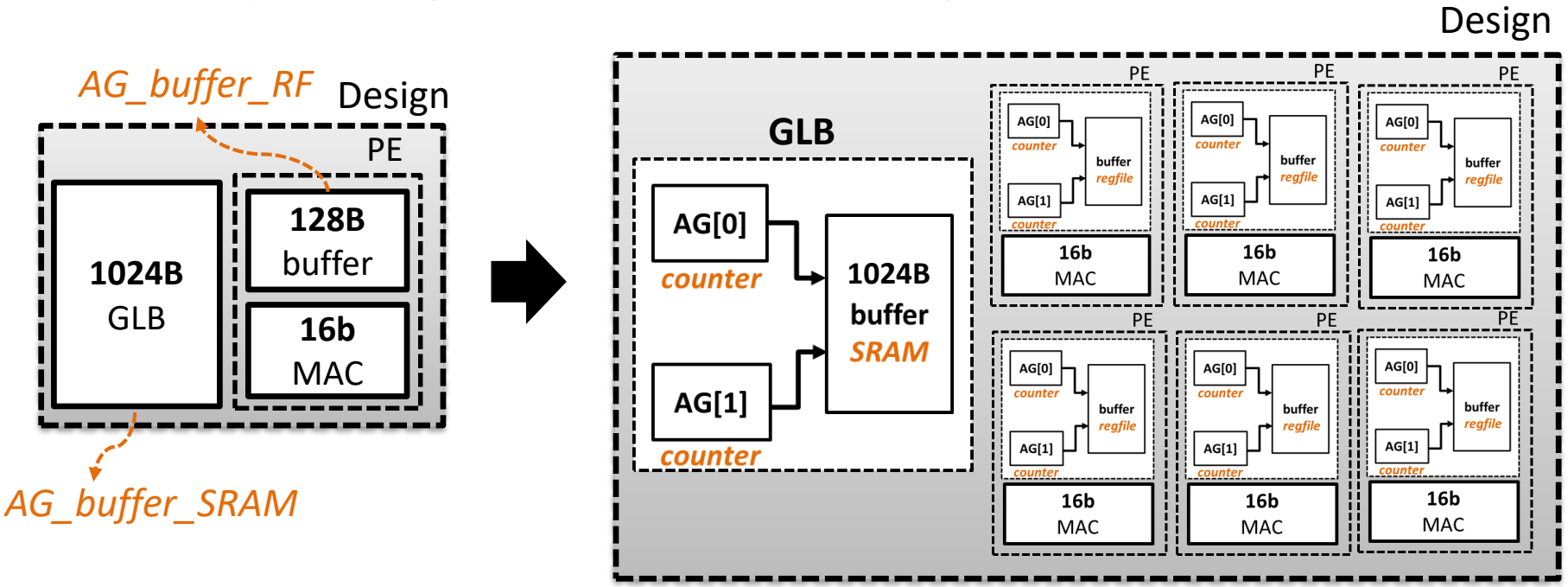
Modeling Complicated Designs

- **Practical architecture designs involve much more details**
  - **Example: storage units with local address generators (AG)**



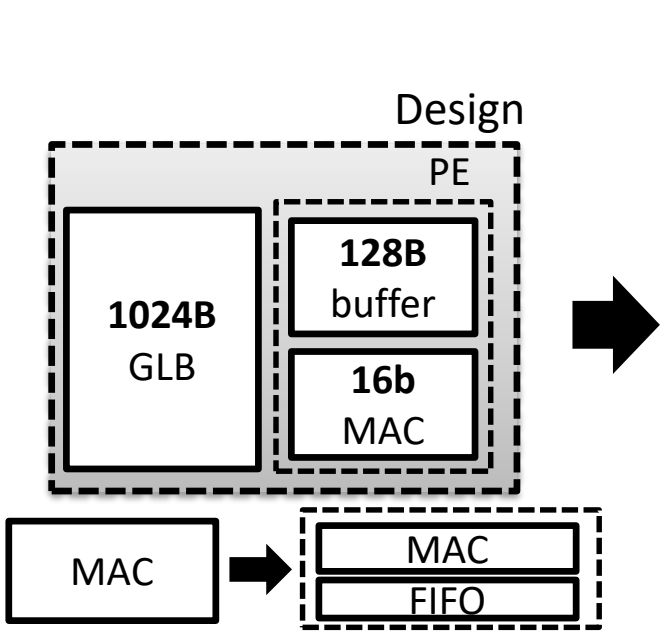
Modeling Complicated Designs

- **Practical architecture designs involve much more details**
  - **Example: storage units with local address generators (AG)**

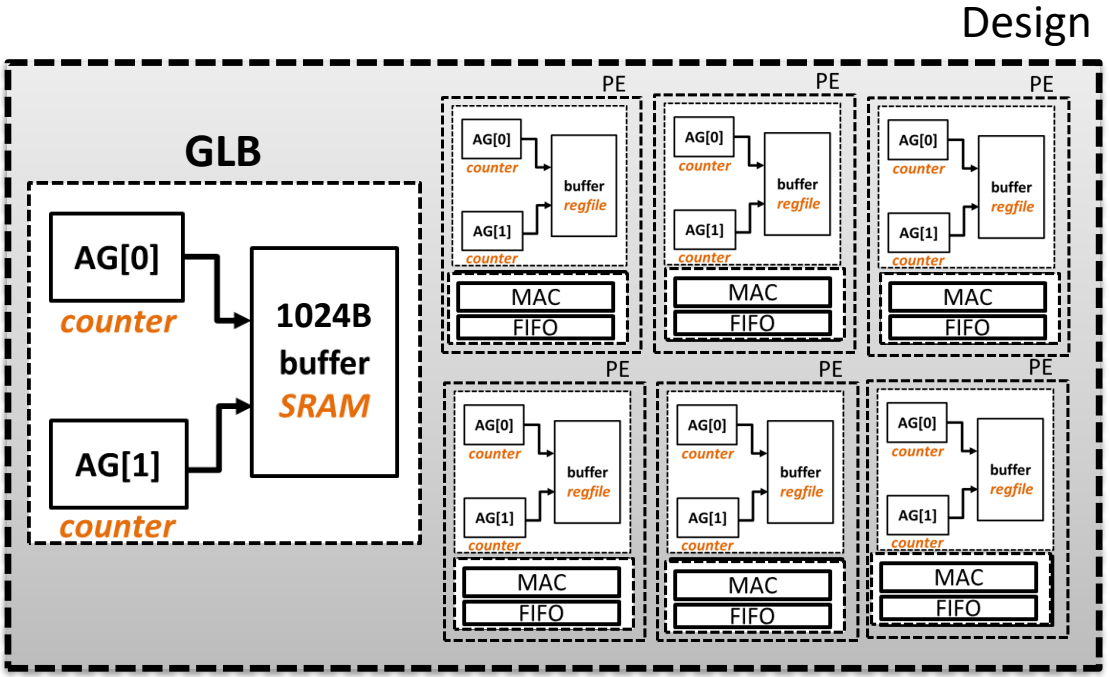


Modeling Complicated Designs

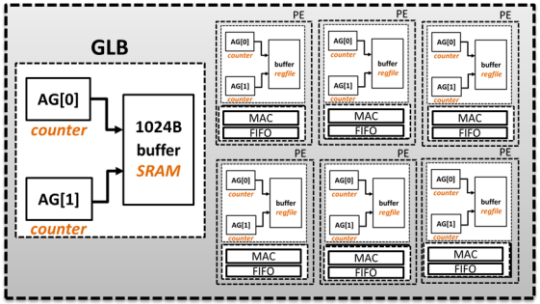
- **Practical architecture designs involve much more details**
  - **Example: storage units with local address generators (AG)**



Add output FIFO to improve MAC performance



Modeling Complicated Designs



Accelergy

ERT Generator

ERT (YAML)

Primitive Component Library (YAML)

Energy Calculator

Action Counts (YAML)

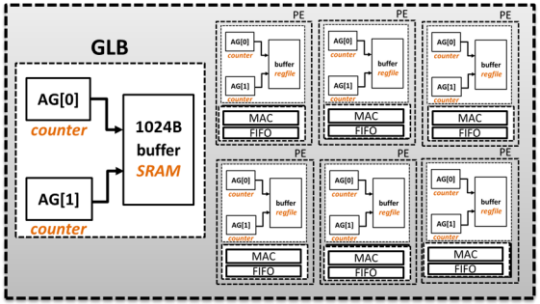
CACTI Estimation Plug-in

40nm Estimation Plug-in

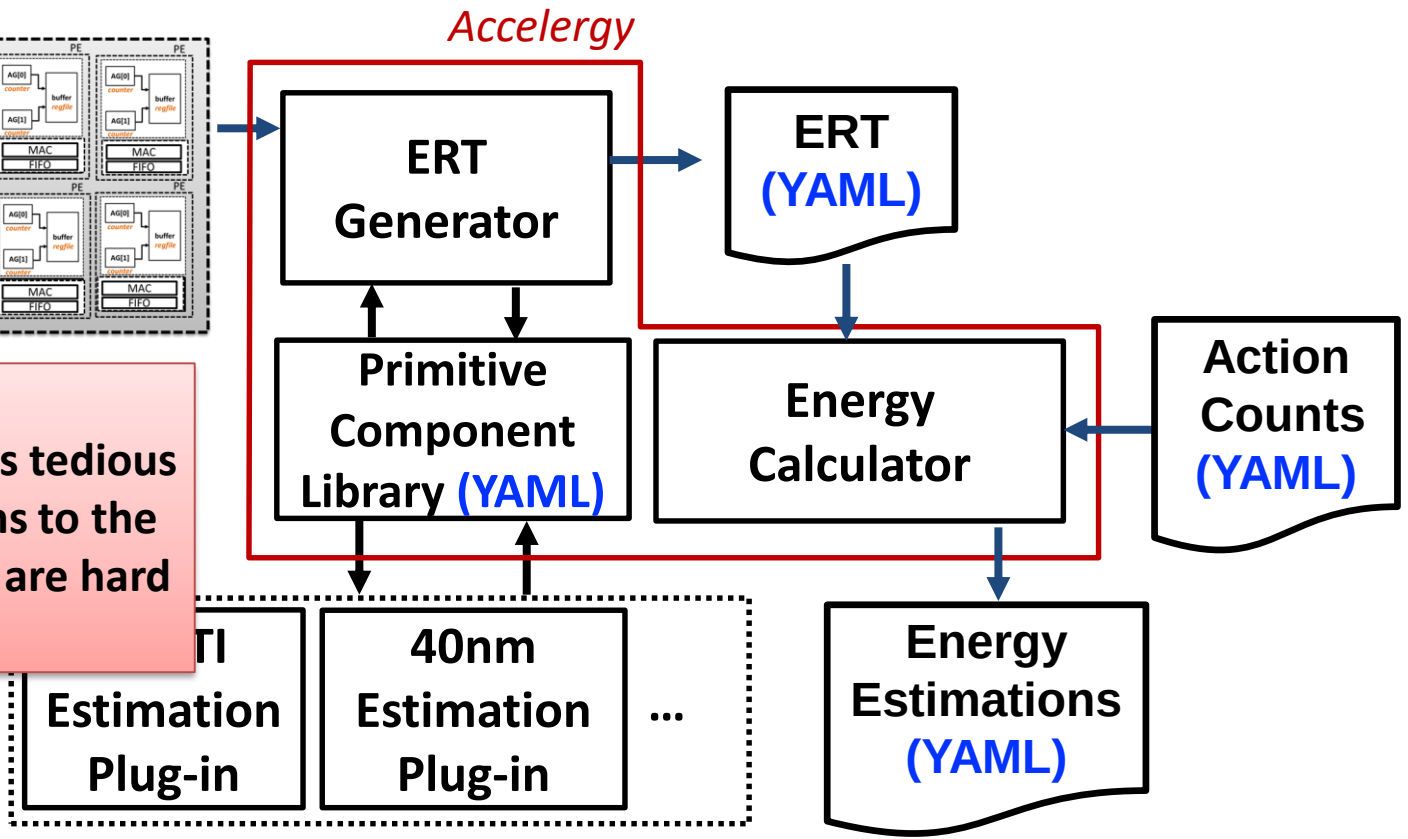
...

Energy Estimations (YAML)

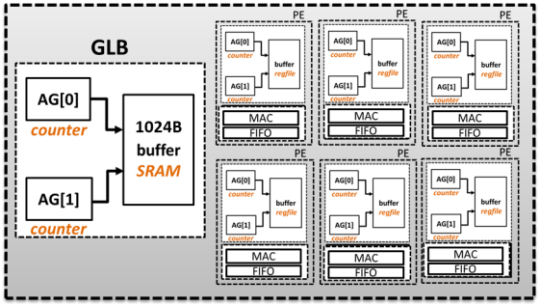
Modeling Complicated Designs



- Architecture description is tedious
- Modifications to the architecture are hard to make



Modeling Complicated Designs



- Architecture description is tedious
- Modifications to the architecture are hard to make

Accelergy

ERT Generator

ERT (YAML)

- Action counts are even more tedious
- Small modification to the architecture requires new action counts

Primitive Component Library (YAML)

Energy Calculator

Action Counts (YAML)

TI Estimation Plug-in

40nm Estimation Plug-in

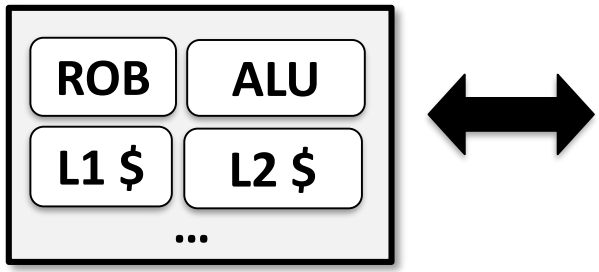
...

Energy Estimations (YAML)

Modeling Complicated Designs

- **Existing work that aims to succinctly model complicated architectures**
  - **Wacttch, McPAT, PowerTrain**

CPU-Centric Architecture Model



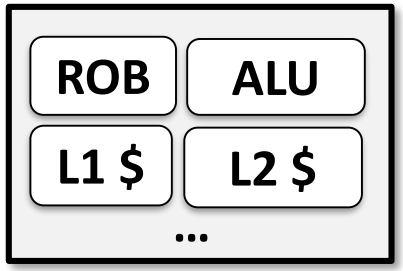
Use a fixed set of compound components to represent model the architecture

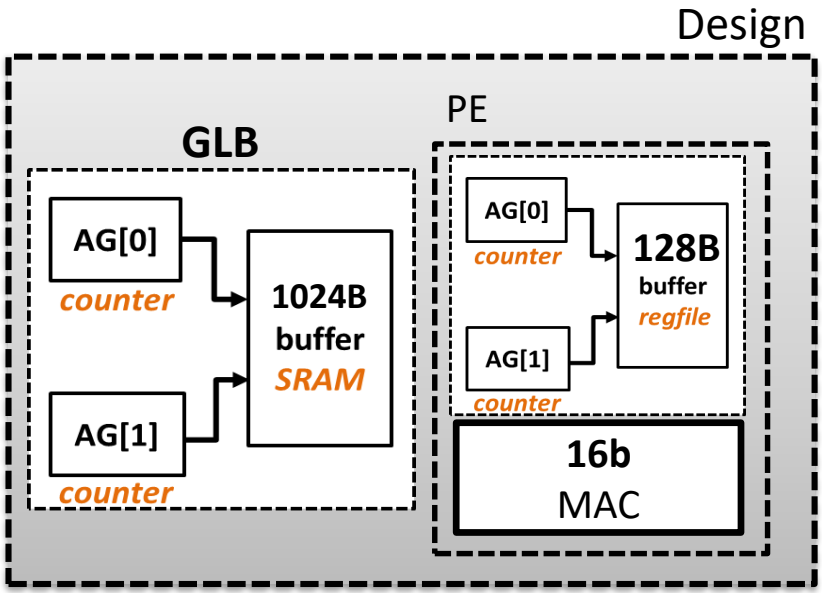
Components that can be decomposed into lower level components

Modeling Complicated Designs

- **Existing work that aims to succinctly model complicated architectures**
  - **Wacttch, McPAT, PowerTrain**

CPU-Centric Architecture Model



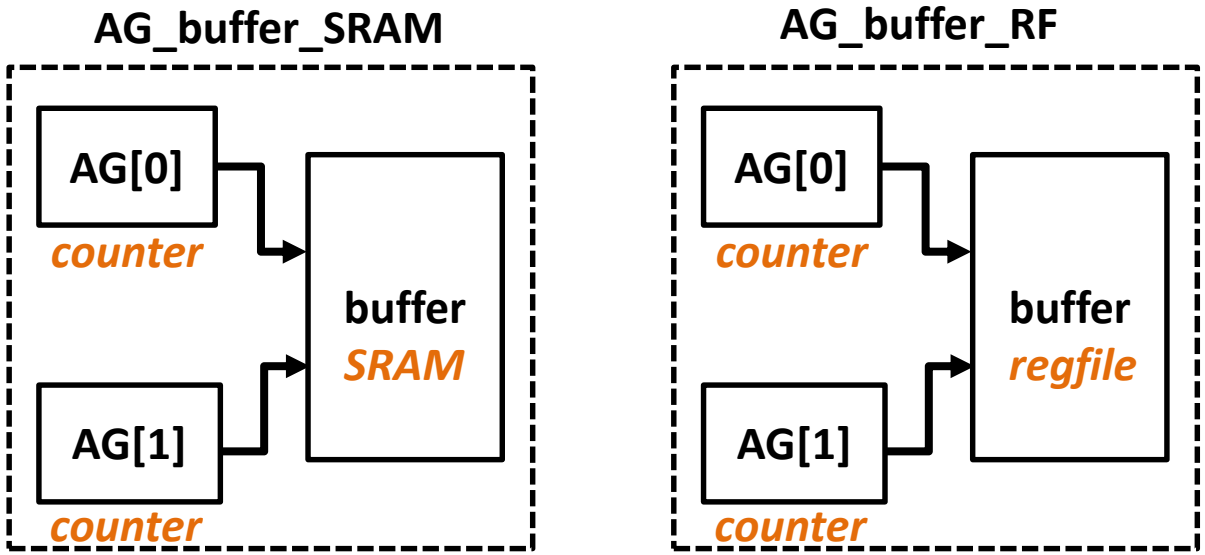


The fixed set of compound components is not sufficient to describe arbitrary accelerator designs

Accelergy Overview

- An architecture-level energy estimator that can be easily integrated into other design exploration tools
- Accurately and systematically captures the energy consumption of basic building blocks in the design
- **Succinctly models diverse and complicated designs**
- Achieves 95% accuracy in evaluating a well-know deep neural network accelerator design – Eyeriss [2016 ISSCC].

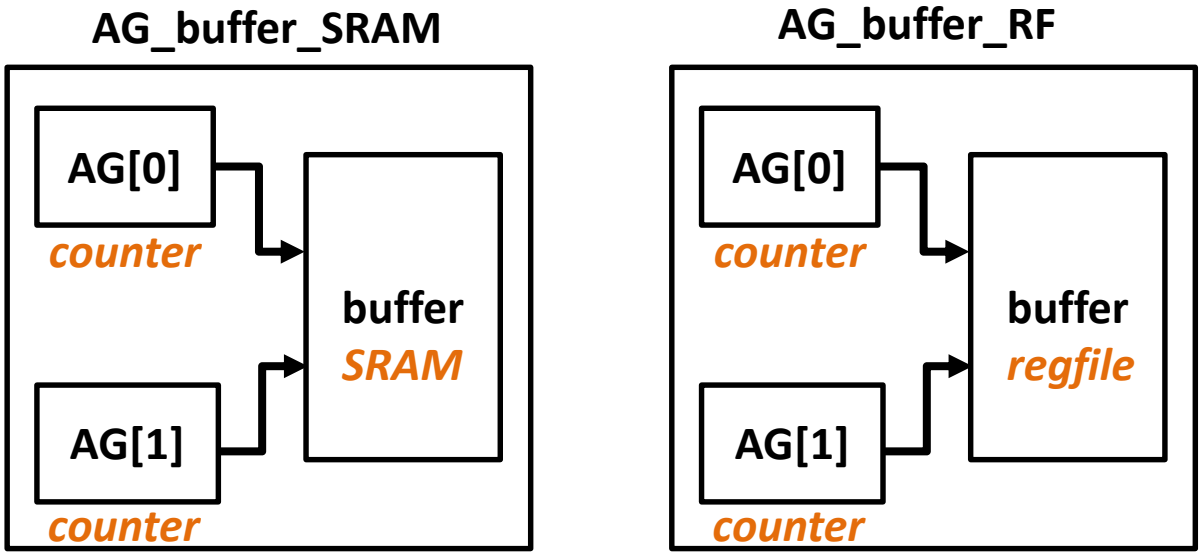
Accelergy: Succinctly Model Arbitrary Designs



Previous: Abstract Hierarchies

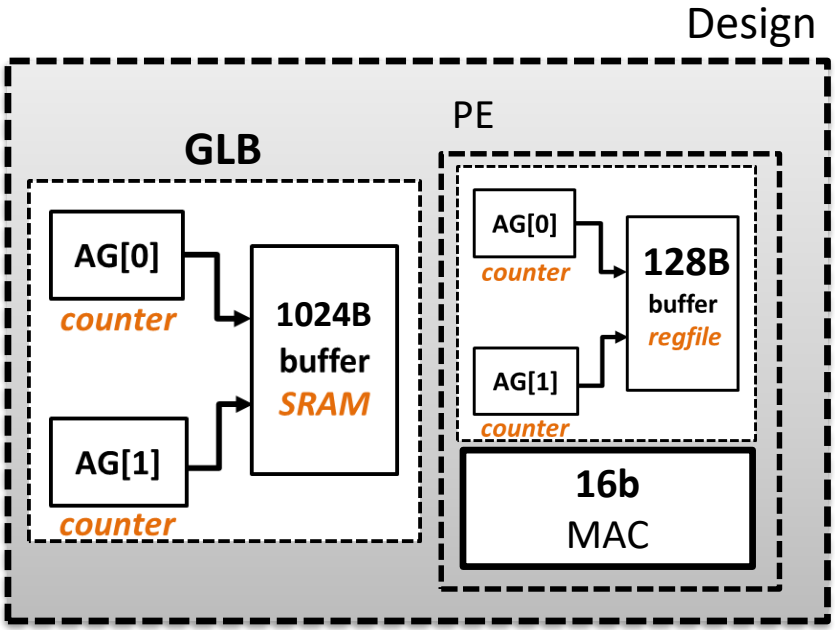
Accelergy: Succinctly Model Arbitrary Designs

- Allows users to describe the design-specific compound components



Now: User-Defined Compound Components

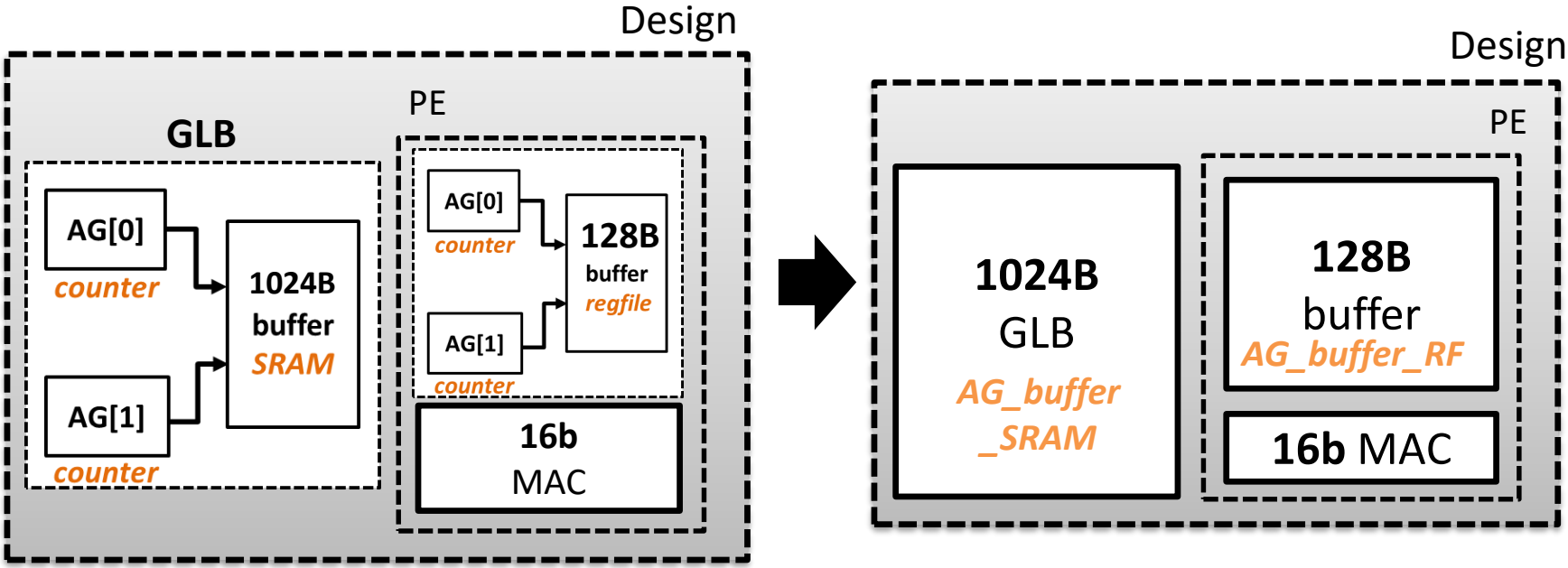
Accelergy: Succinctly Model Arbitrary Designs



Previous: Architecture described with primitive components

Accelergy: Succinctly Model Arbitrary Designs

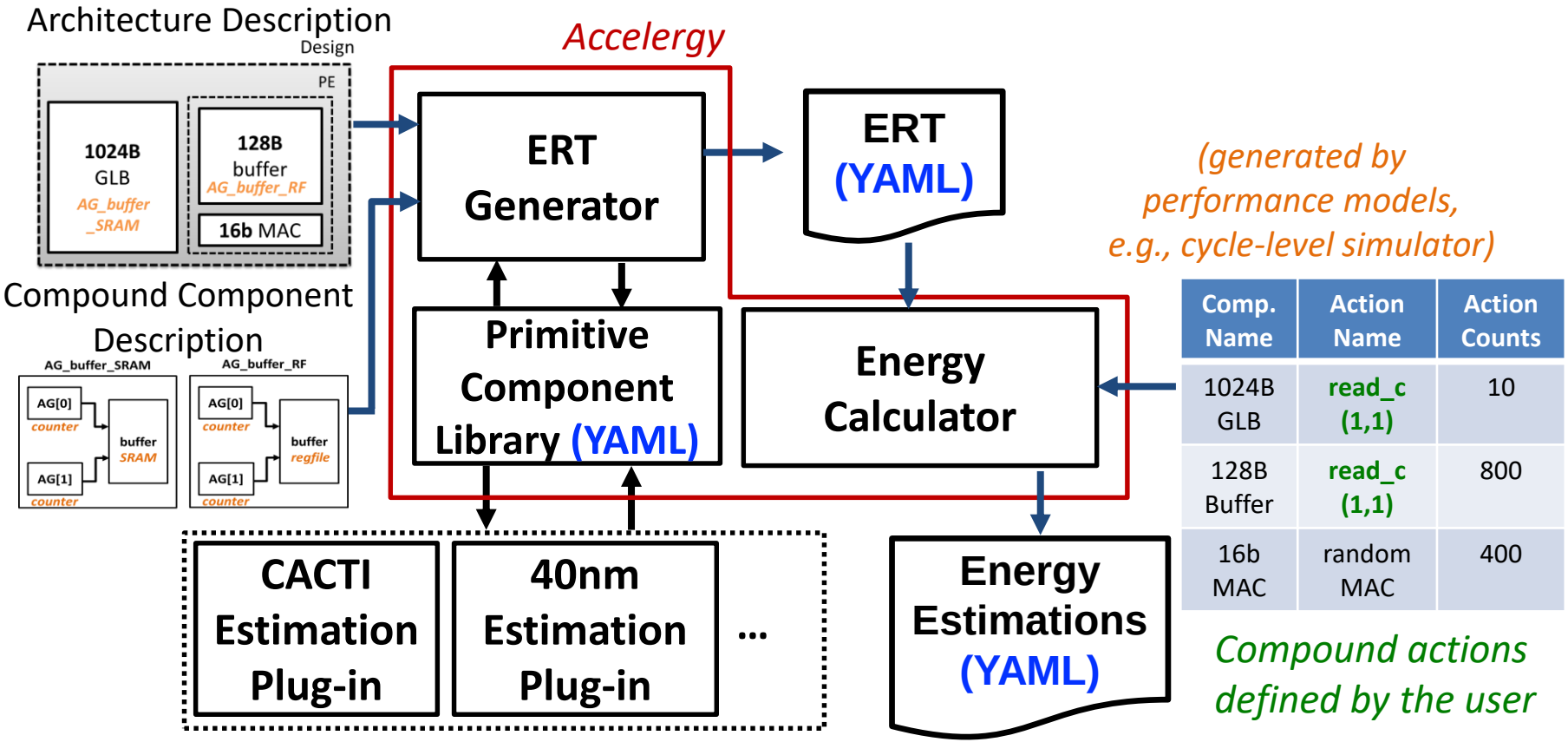
- Allows Architecture to be described with compound components



Previous: Architecture described with primitive components

Now: Architecture described with compound components

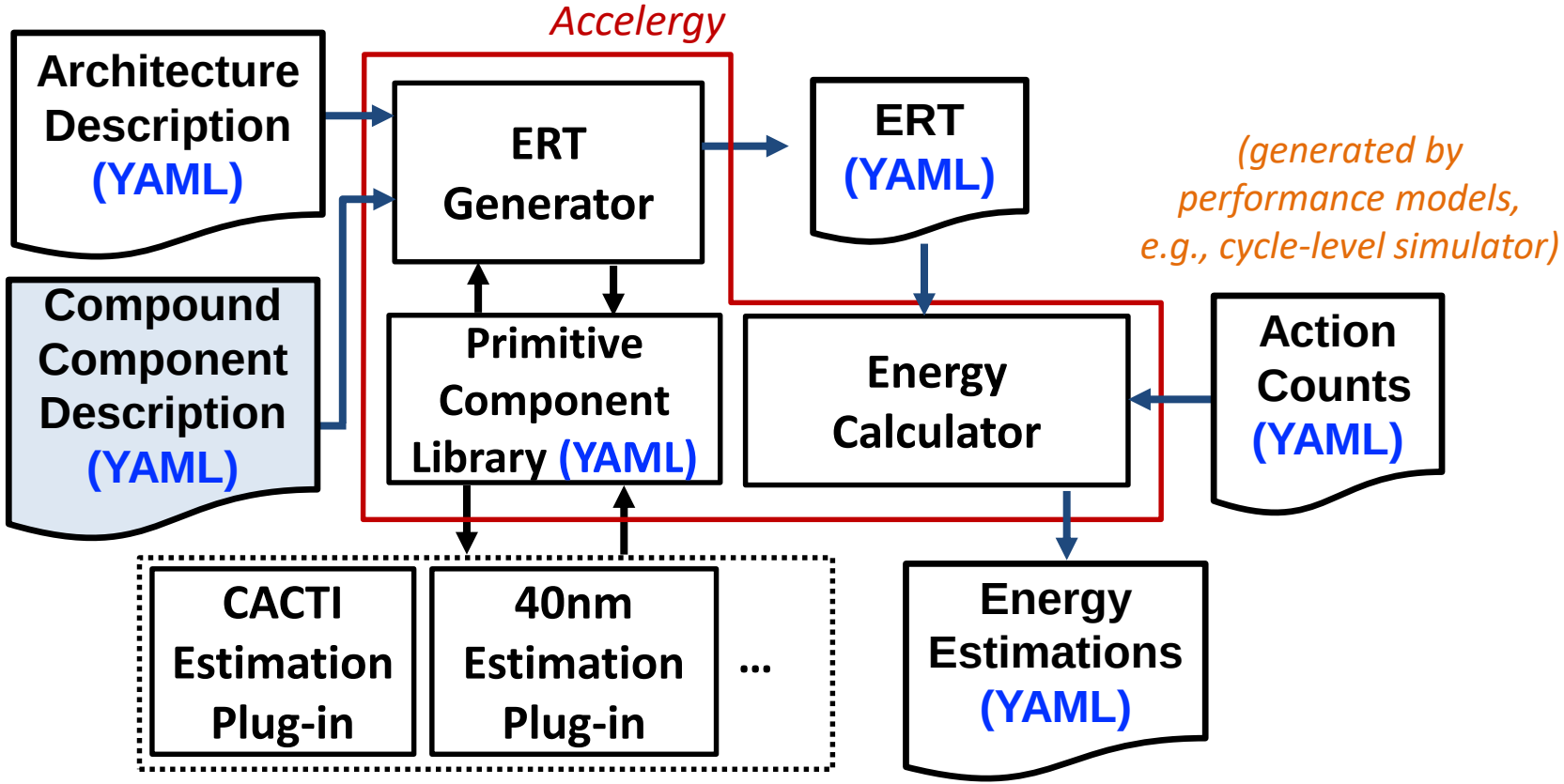
Accelergy: Succinctly Model Arbitrary Designs



How to use Accelergy?

1. Estimate architectures with primitive components
2. **Estimate architectures with compound components**

Accelergy: Succinctly Model Arbitrary Designs



Exercise 3a

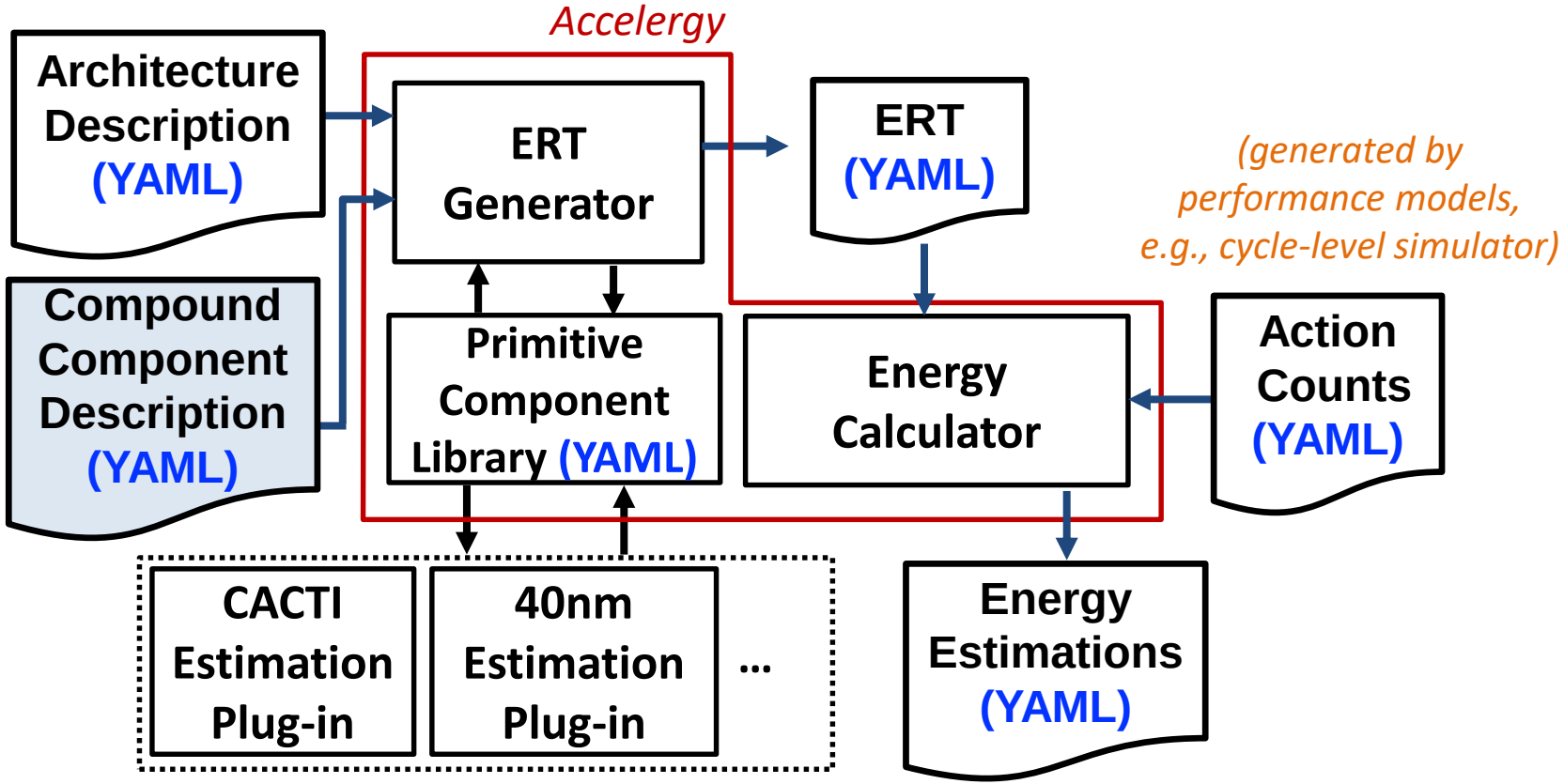
- **Instructions:**
  - `cd ../03_compound_architecute_estimation`
- **Questions:**
  - Does the architecture description with compound components follow the previous general format?
  - Does the action counts follow the previous general format?

Exercise 3a

- **Instructions:**
  - cd ../03_compound_architecute_estimation
- **Questions:**
  - Does the architecture description with compound components follow the previous general format?
  - Does the action counts follow the previous general format?

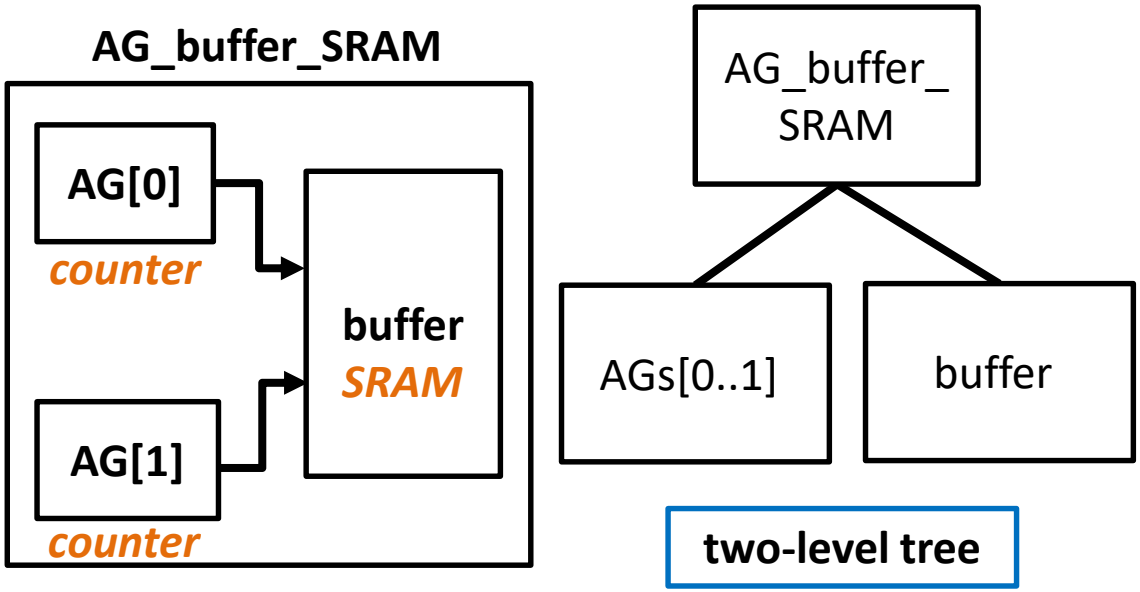
Architecture description and Action counts follow the same general format

Accelergy: Succinctly Model Arbitrary Designs



Accelergy: Succinctly Model Arbitrary Designs

- How to define a compound component?



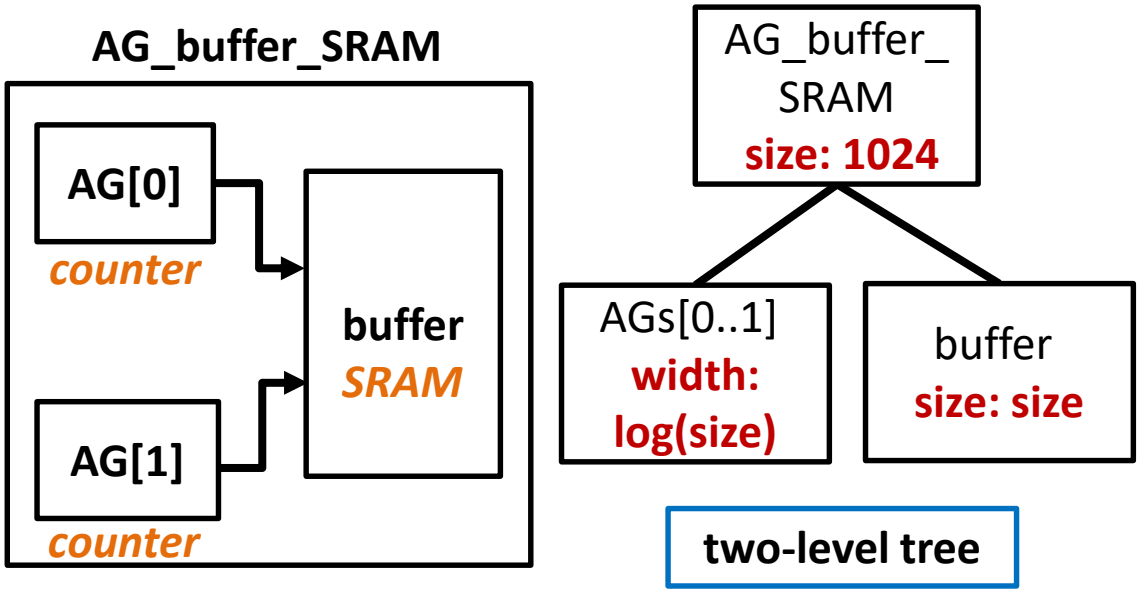
YAML Syntax

```
name:AG_buffer_SRAM


subcomponents:
  - name: buffer
    class: SRAM


  - name: AGs[0..1]
    class: counter
```

Define a set of subcomponents

Accelergy: Succinctly Model Arbitrary Designs

- How to define a compound component?



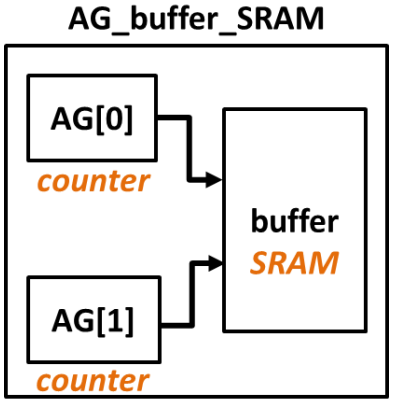
YAML Syntax

```
name:AG_buffer_SRAM
attributes:
  size: 1024 …
subcomponents:
  - name: buffer
    class: SRAM
    attributes:
      size: size …
  - name: AGs[0..1]
    class: counter
    attributes:
      width: log(size)
```

Define a set of attributes

Accelergy: Compound Component Description

- **How to define a compound component?**



AG_buffer_SRAM :
read (addr_delta, data_delta)

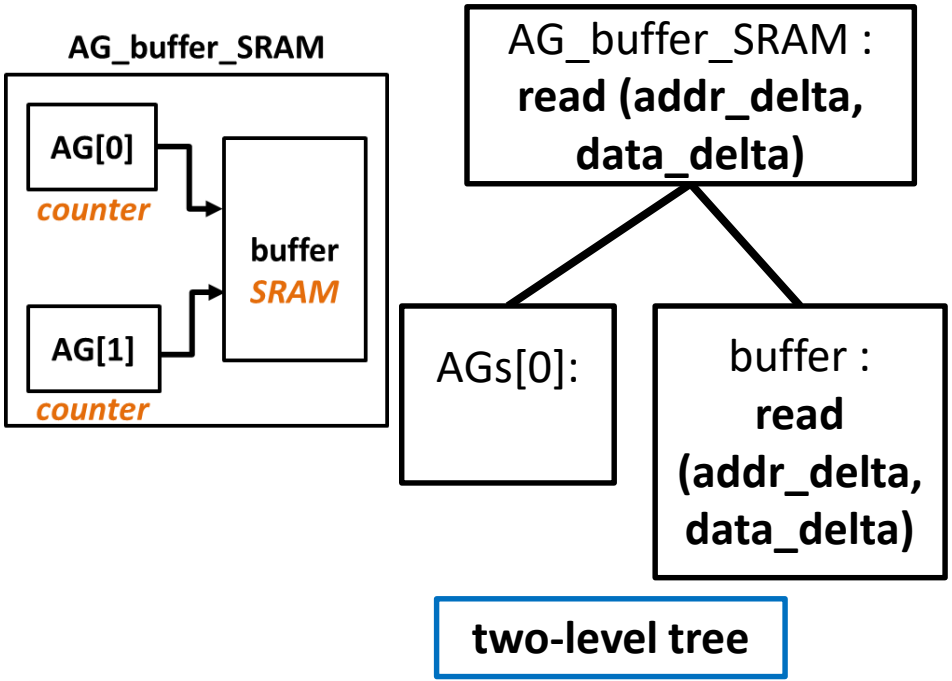
YAML Syntax

```
name: AG_buffer_SRAM
actions:
  - name: read
    arguments:
      addr_delta: 0..1
      data_delta: 0..1
```

Define a set of compound actions

Accelergy: Compound Component Description

- **How to define a compound component?**



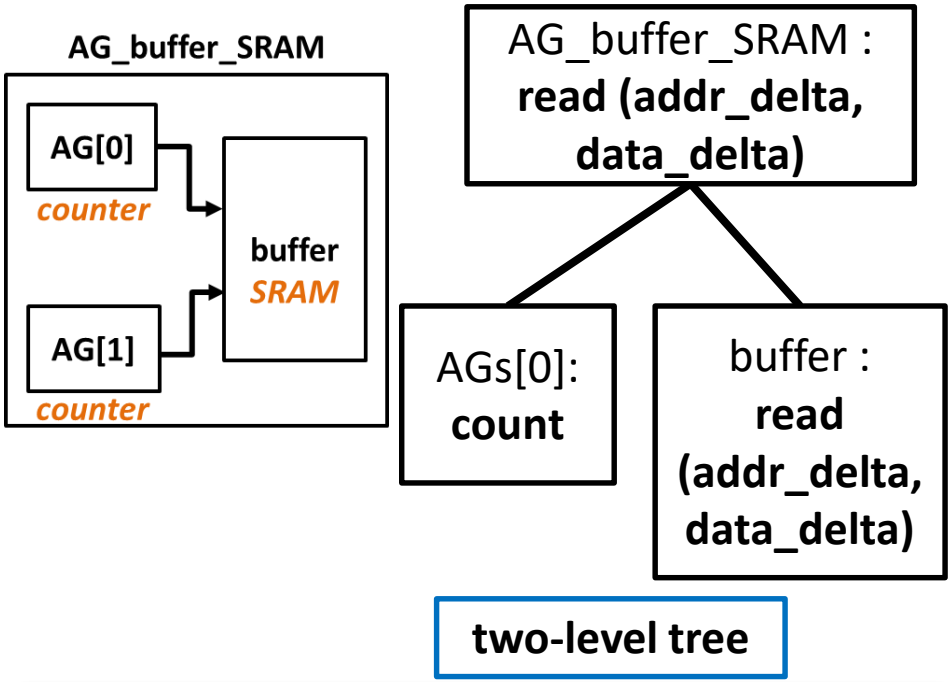
YAML Syntax

```
name: AG_buffer_SRAM
actions:
  - name: read
    arguments:
      addr_delta: 0..1
      data_delta: 0..1
    subcomponents:
      - name: buffer
        actions:
          - name: read
            arguments:
              addr_delta: addr_delta
              data_delta: data_delta
```

Define a set of compound actions

Accelergy: Compound Component Description

- **How to define a compound component?**

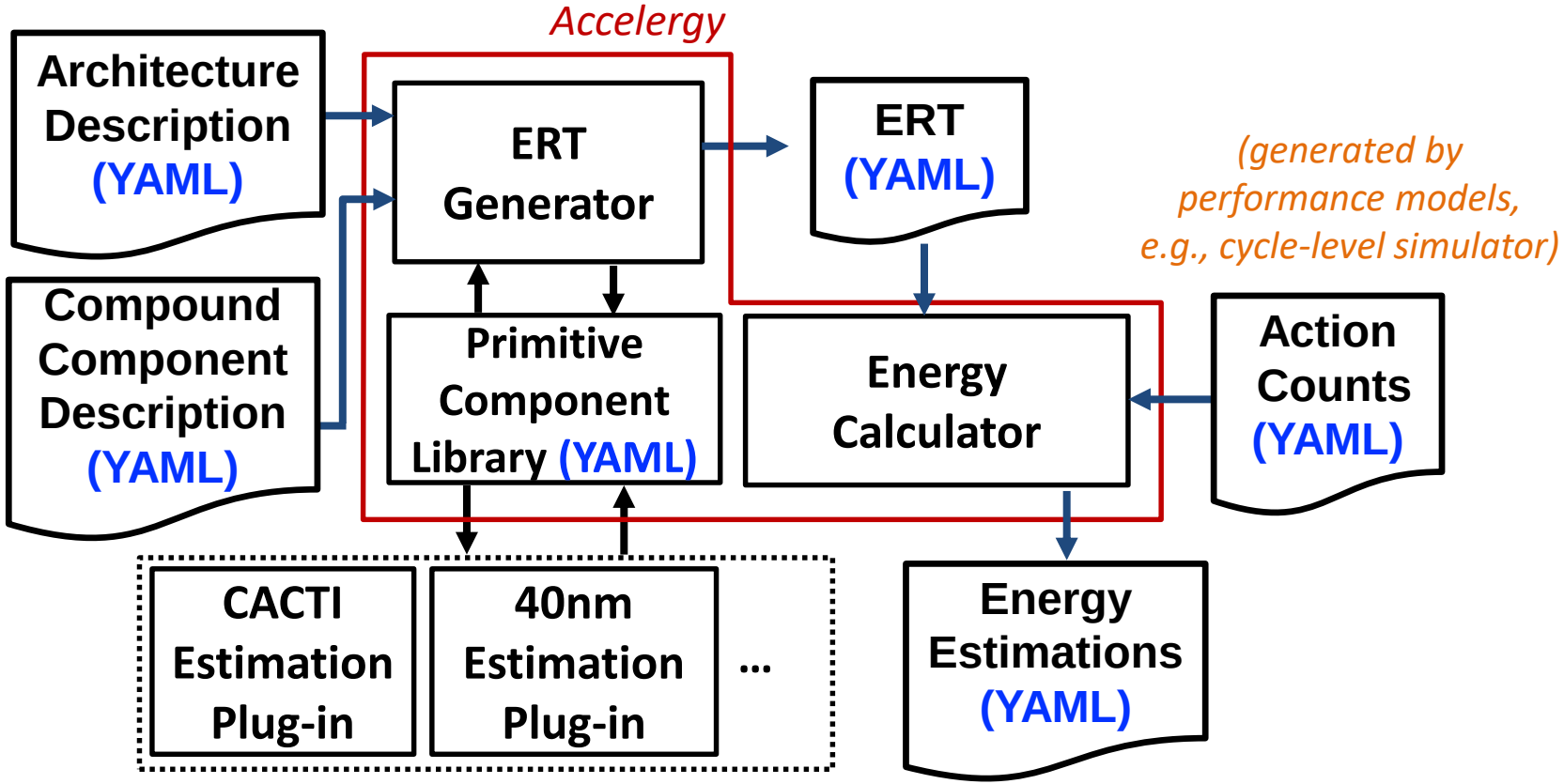


Define a set of compound actions

YAML Syntax

```
name: AG_buffer_SRAM
actions:
  - name: read
    arguments:
      addr_delta: 0..1
      data_delta: 0..1
    subcomponents:
      - name: buffer
        actions:
          - name: read
            arguments:
              addr_delta: addr_delta
              data_delta: data_delta
      - name: AGs[0]
        actions:
          - name: count
```

Accelergy: Succinctly Model Arbitrary Designs



Exercise 3b

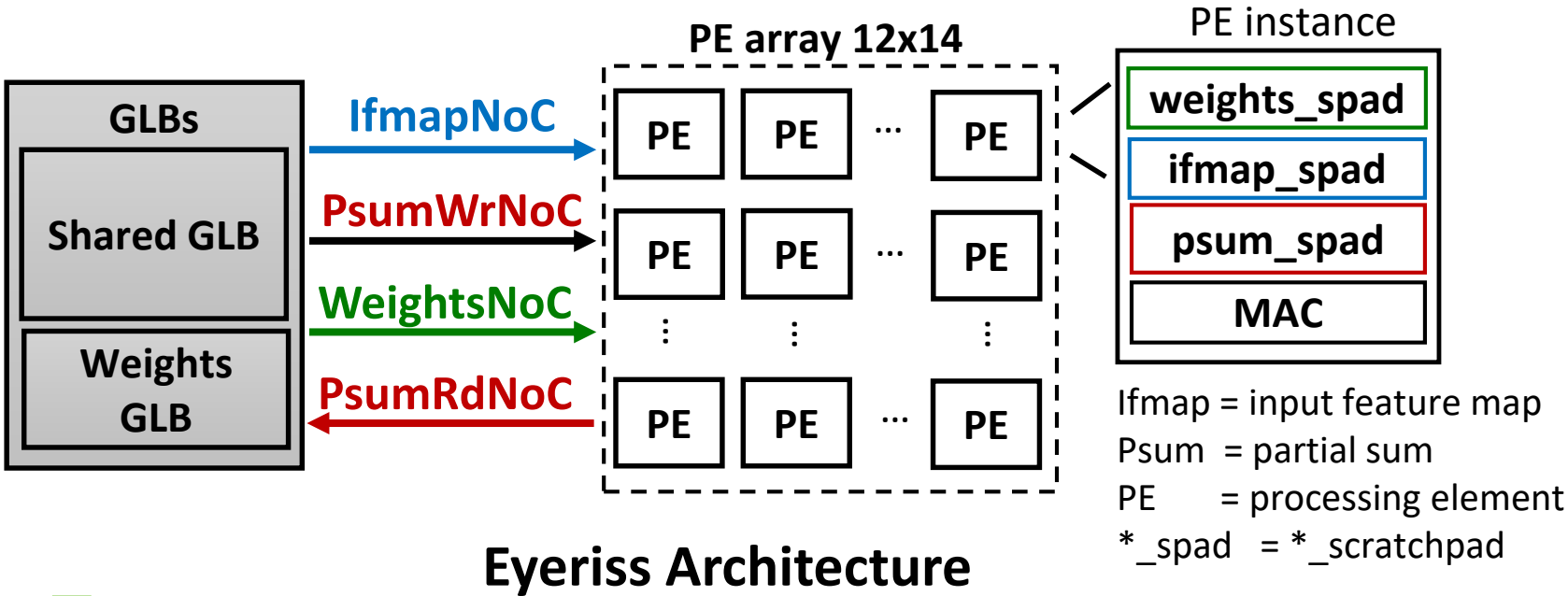
- **Instruction:**
  - `accelergy *.yaml components/*.yaml -o output/`
- **Questions:**
  - Are the ERTs in terms of compound components or primitive components? Why?
  - Compare the current ERT with the previous ERT, what is different?
  - If we change the hardware implementation of a compound component, e.g. add a FIFO to AG_buffer_SRAM, what needs to be changed?

Accelergy Overview

- An architecture-level energy estimator that can be easily integrated into other design exploration tools
- Accurately and systematically captures the energy consumption of basic building blocks in the design
- Succinctly models diverse and complicated designs
- **Achieves 95% accuracy in evaluating a well-know deep neural network accelerator design – Eyeriss [2016 ISSCC].**

Energy Evaluations on Eyeriss

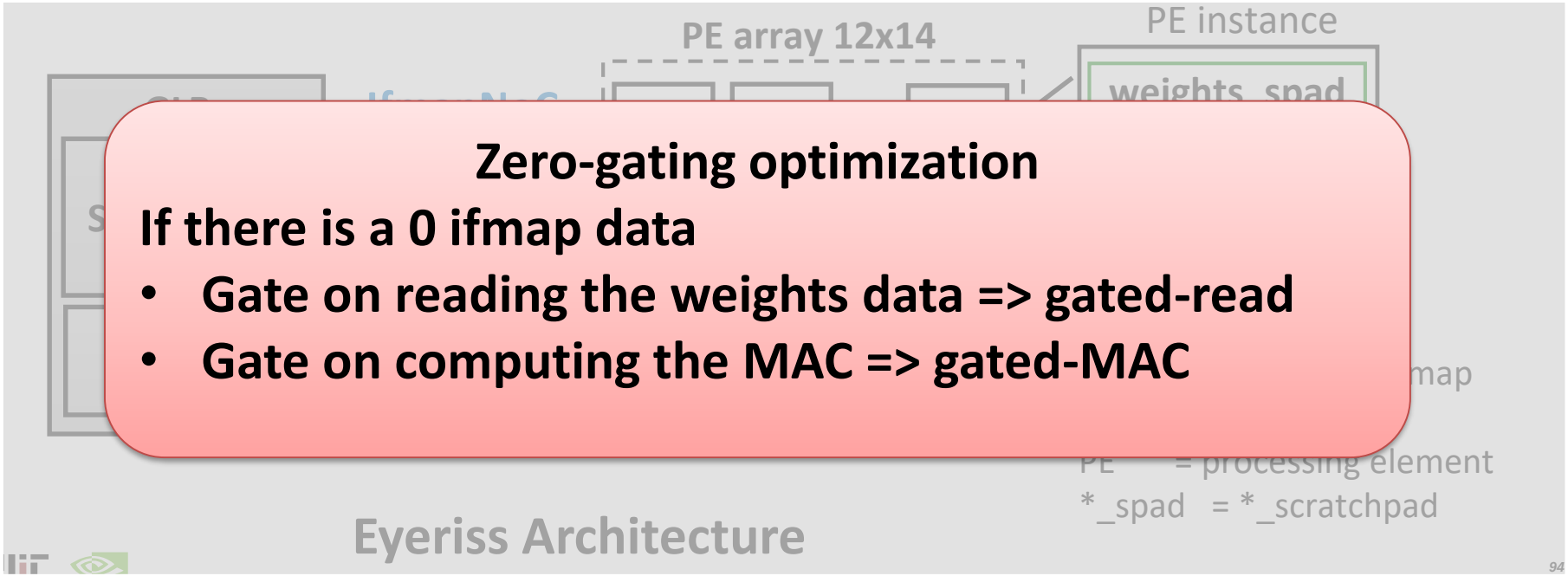
- **Experimental Setup:**
  - **Workload: Alexnet weights & ImageNet input feature maps**
  - **Ground Truth: Energy obtained from post-layout simulations**



Eyeriss Architecture

Energy Evaluations on Eyeriss

- **Experimental Setup:**
  - **Workload: Alexnet weights & ImageNet input feature maps**
  - **Ground Truth: Energy obtained from post-layout simulations**



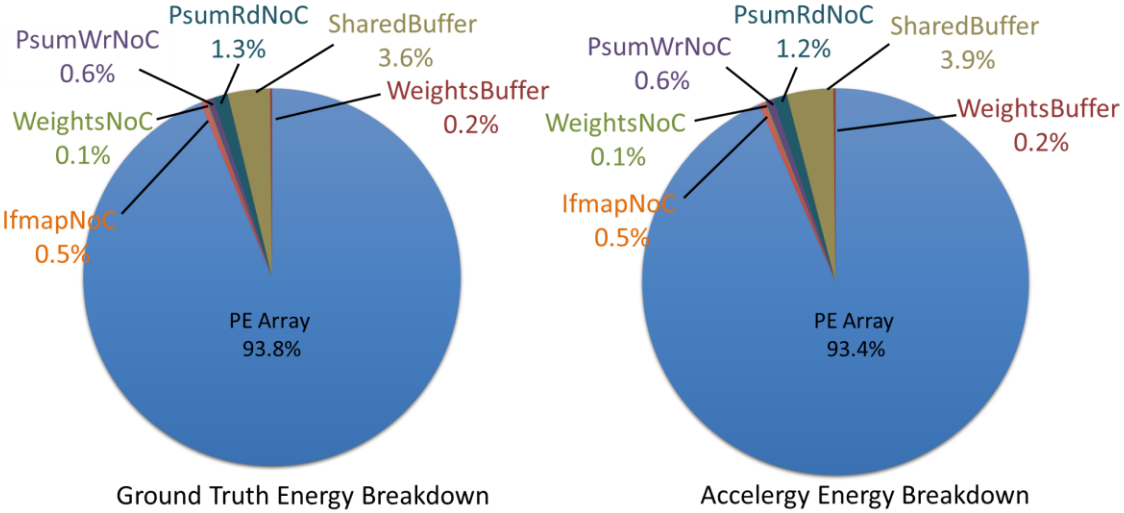
Zero-gating optimization

If there is a 0 ifmap data

- **Gate on reading the weights data => gated-read**
- **Gate on computing the MAC => gated-MAC**

Experimental Results

- **Total energy estimation is 95% accurate of the post-layout energy.**
- **Estimated relative breakdown of the important units in the design is <u>within 8%</u> of the post-layout energy.**



**Total energy might not add up to exact 100.0% due to rounding*

Experimental Results

- **Comparisons with existing work: Aladdin and fixed-cost**

Design-specific energy estimators

Aladdin: *[Shao et al., ISCA2014]*

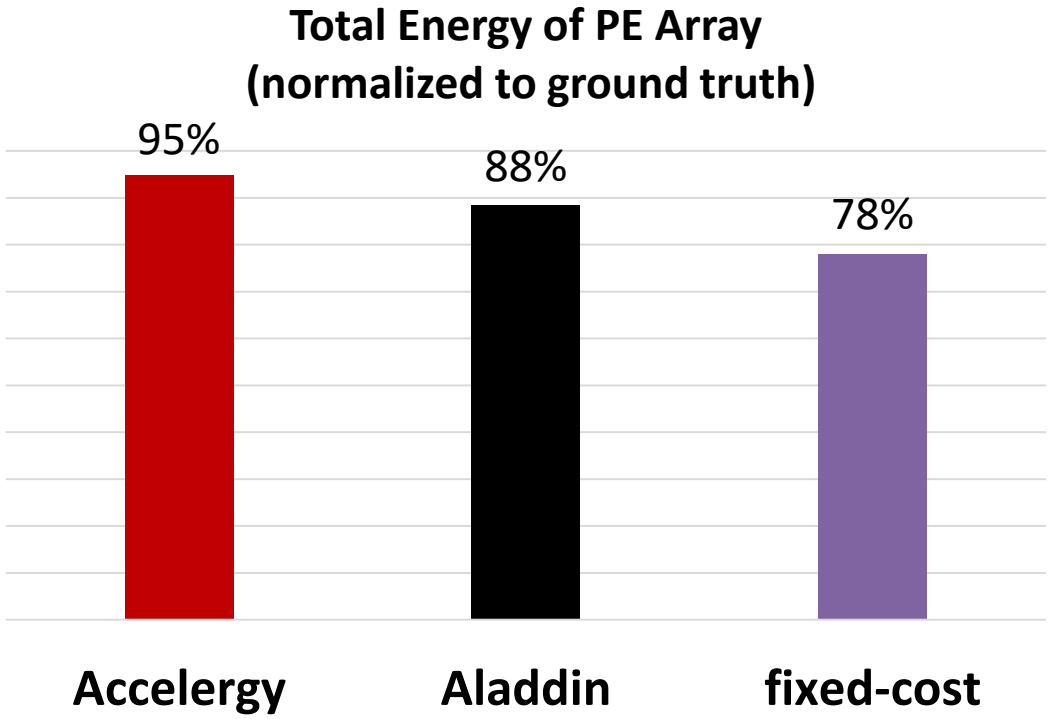
- lack of fine-grained action types, e.g., gated-read

fixed-cost: *[Yang et al., Asilomar2017]*

- Designed specifically for Eyeriss analysis
- lack of fine-grained component characterizations, e.g., all local buffers are characterized as identical components

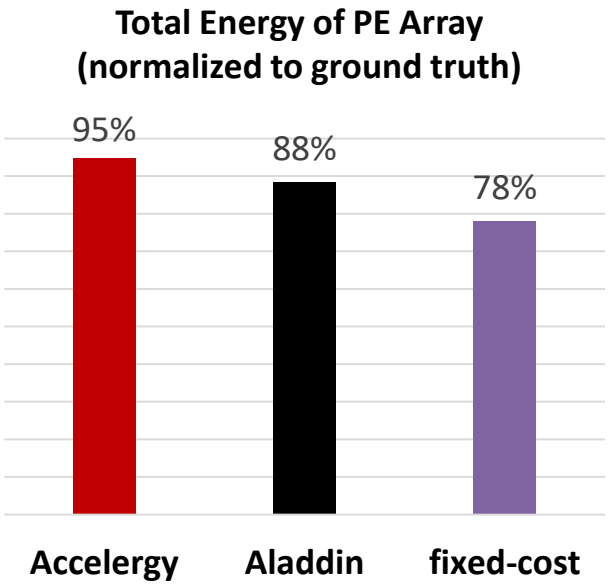
Experimental Results

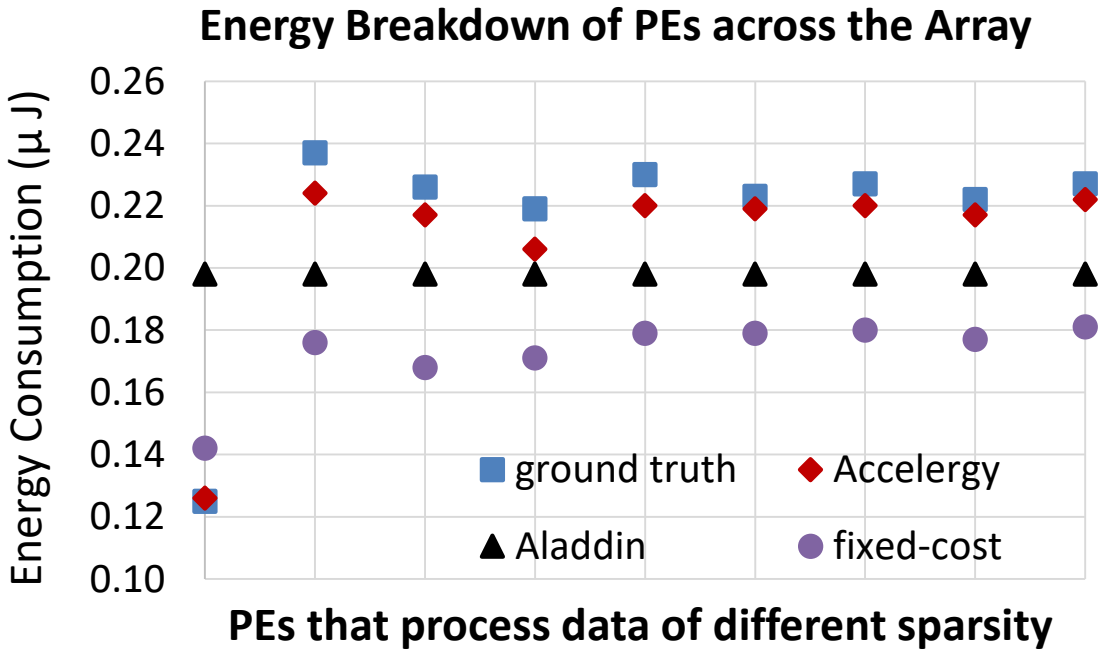
- **Comparisons with existing work: Aladdin and fixed-cost**



Experimental Results

- **Comparisons with existing work: Aladdin and fixed-cost**

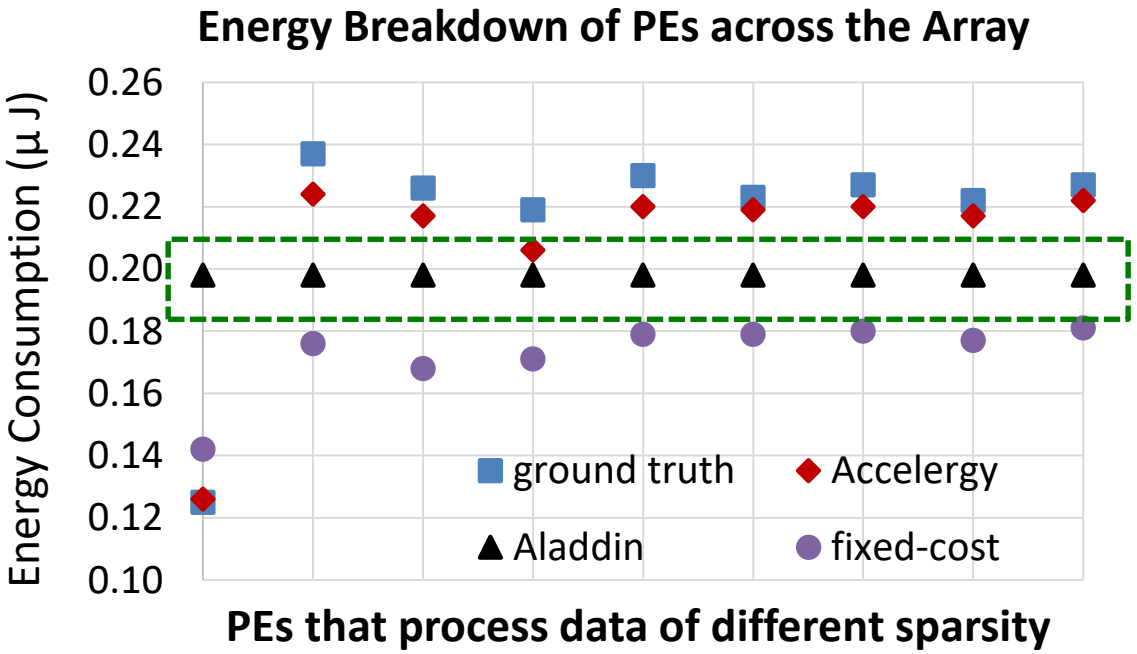




Experimental Results

- **Comparisons with existing work: Aladdin and fixed-cost**

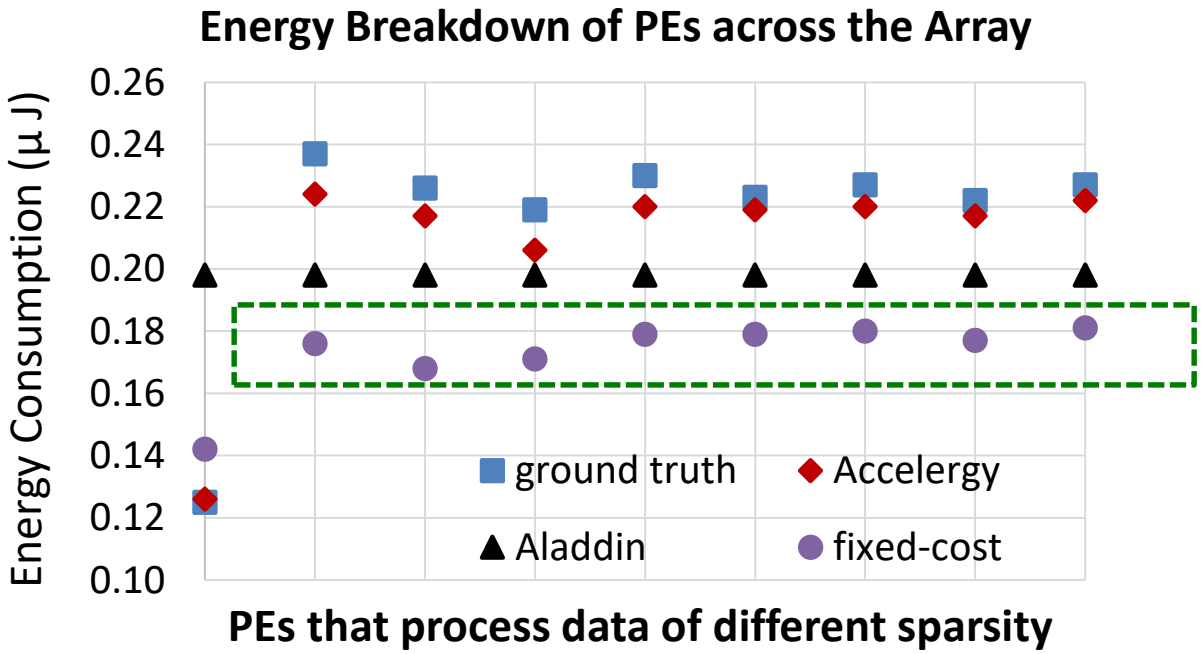
Not sensitive to sparsity as the zero-gating optimization is not recognized



Experimental Results

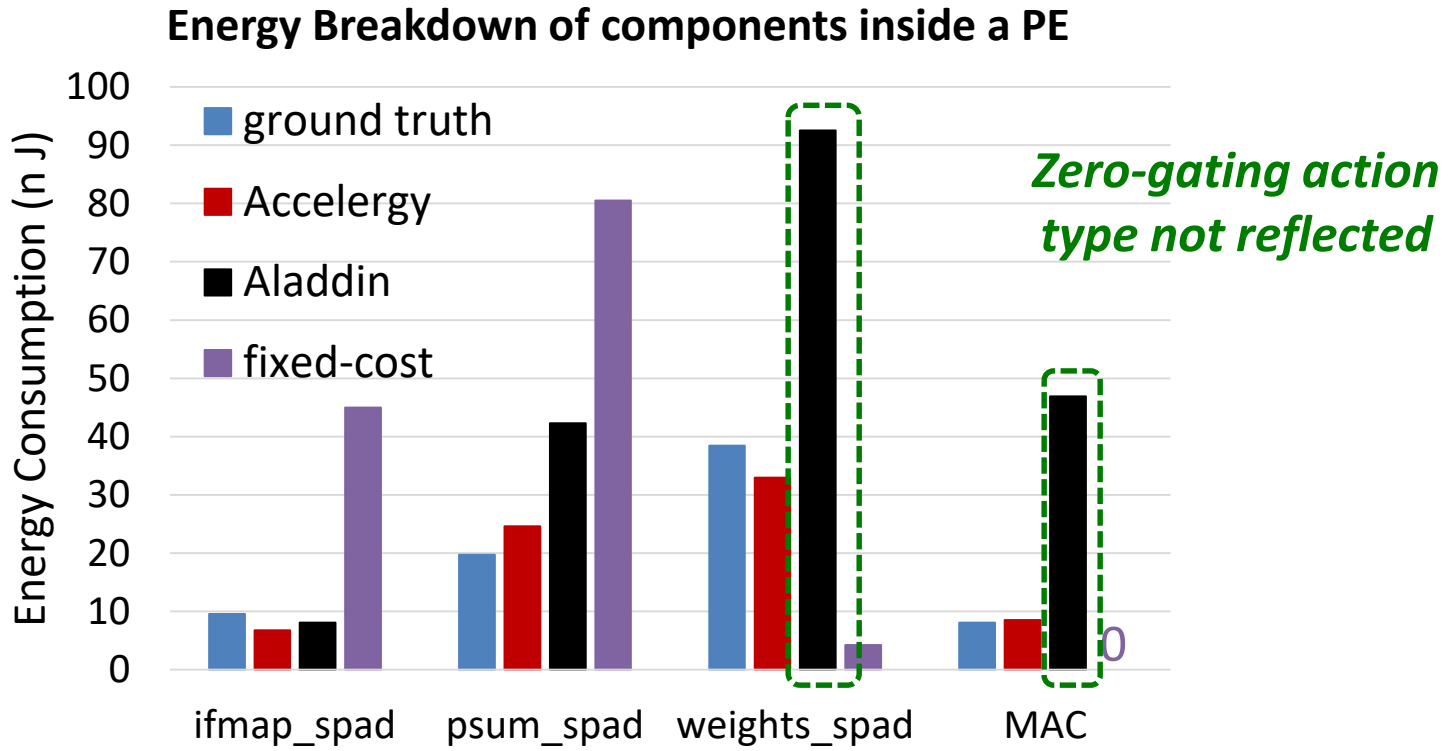
- **Comparisons with existing work: Aladdin and fixed-cost**

Missing components characterizations



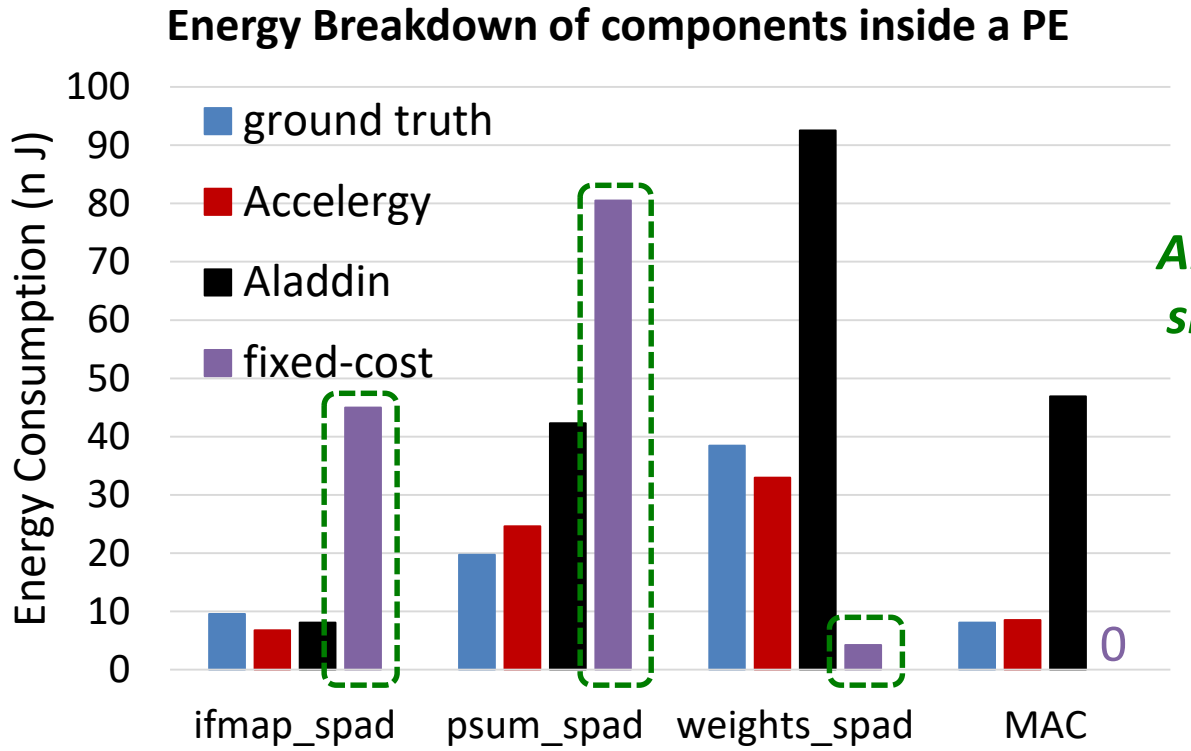
Experimental Results

- **Comparisons with existing work: Aladdin and fixed-cost**



Experimental Results

- **Comparisons with existing work: Aladdin and fixed-cost**



All local scratchpads share the same ERT

Exercise 4

- **Instructions**
  - cd ../04_eyeriss_like
  - Run accelergy *.yaml components/*.yaml –o output/
- **Questions**:
  - What compound components are involved in the architecture?
  - What are the subcomponents of each compound component?
  - What actions in the ERT can help better characterize data-gating?
  - Which components in the PE are sensitive to data sparsity? How do you know?

Free Lab Session

Timeloop + Accelergy

Have fun exploring both tools

- Go to the timeloop + accelergy exercise folder
  - Readme contains instructions for various experiments!
  - Run an mapping exploration with an eyeriss-like architecture that is described by compound components
  - Run an mapping exploration with an floating-point eyeriss-like architecture. Which component's energy increases the most?
  - Try different DRAM types. Modify the architecture description to exploration the energy impact brought by DRAM technologies.