

# Timeloop

# Accelergy

Angshuman Parashar  
Yannan Nellie Wu  
Po-An Tsai  
Vivienne Sze  
Joel S. Emer

NVIDIA  
MIT  
NVIDIA  
MIT  
NVIDIA, MIT

## ISCA Tutorial

Hands-on session

May 2020



Massachusetts  
Institute of  
Technology

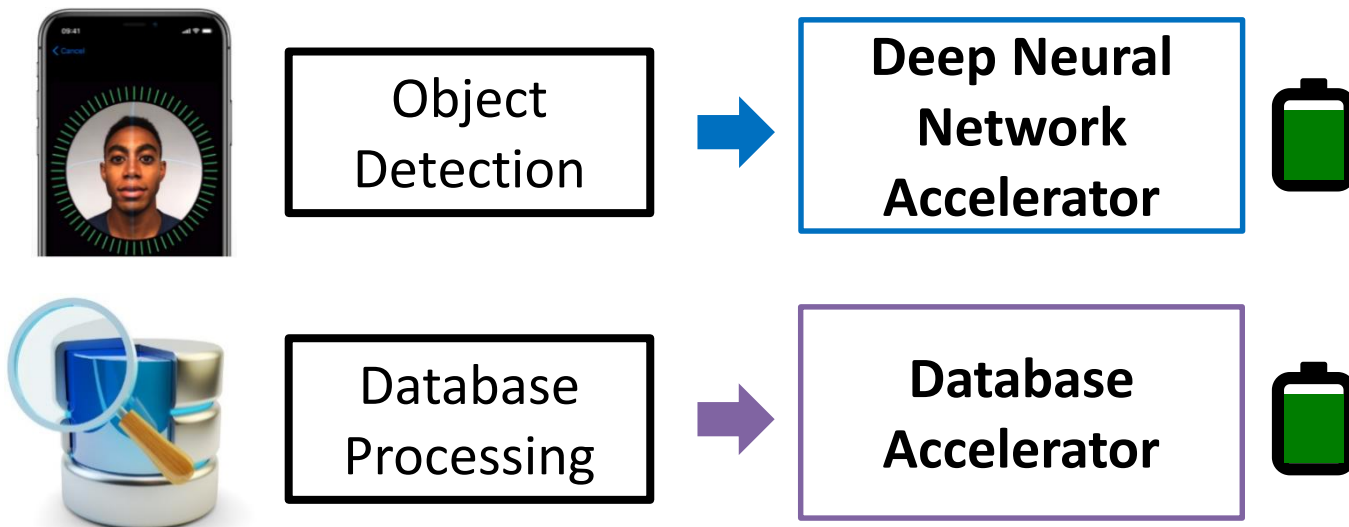


**nVIDIA®**

# Introduction to Accelergy

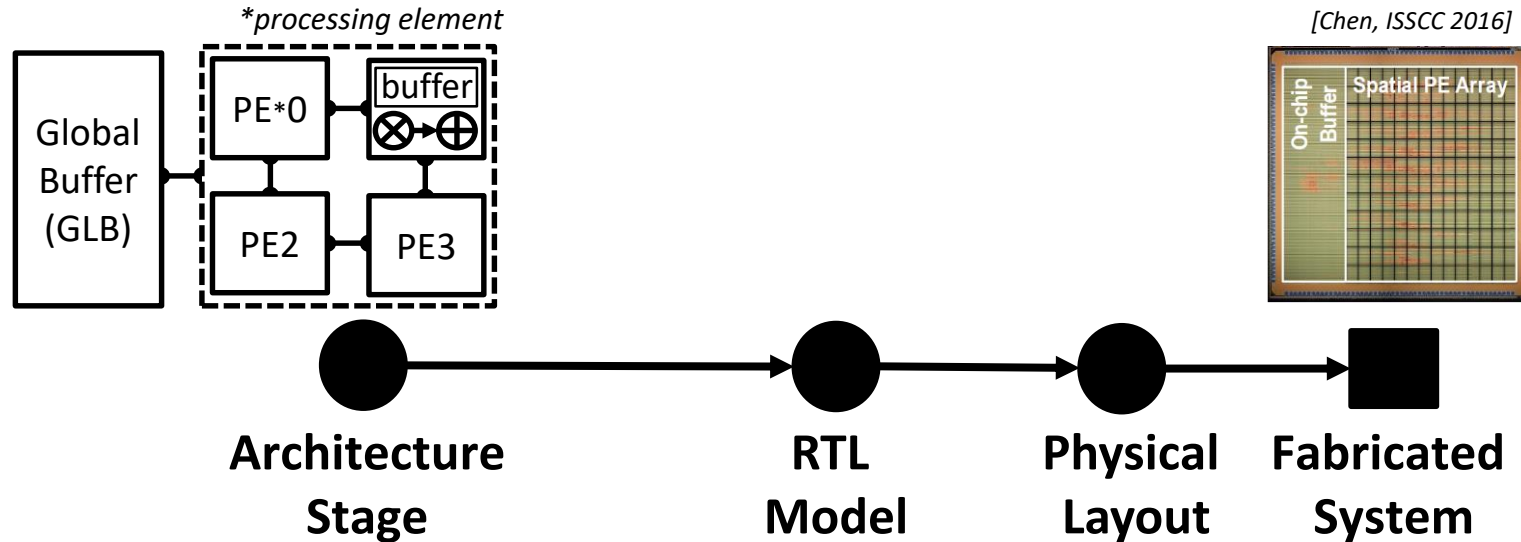
# Domain-Specific Accelerators Improve Energy Efficiency

Data and computation-intensive applications are power hungry

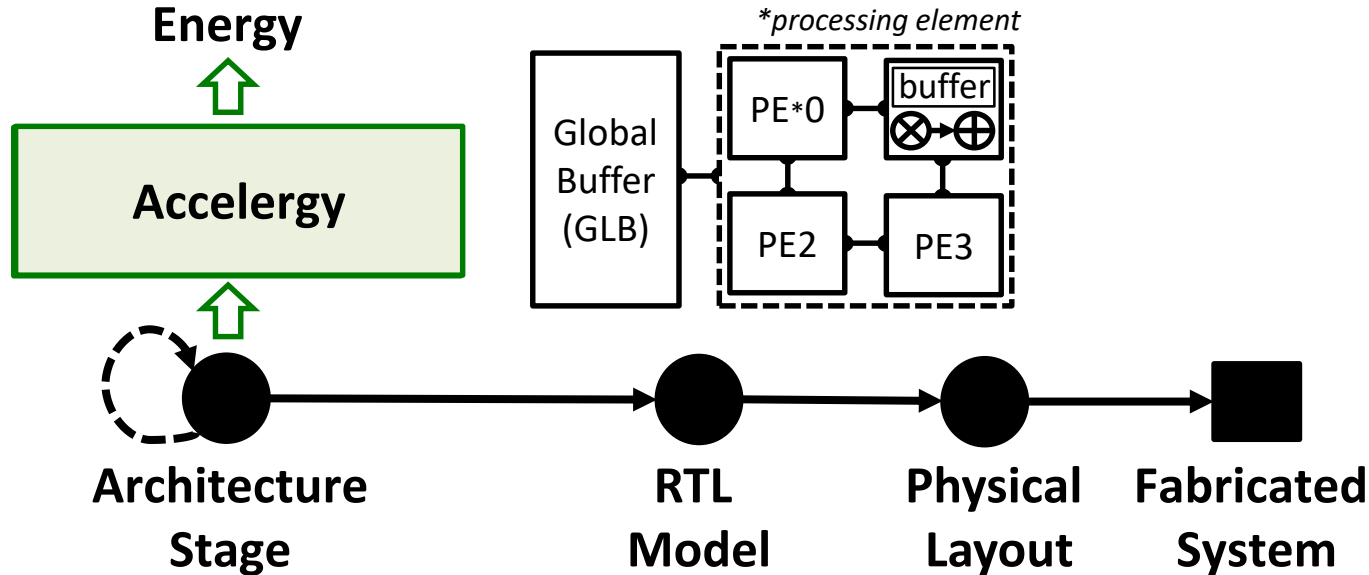


We must quickly evaluate energy efficiency of arbitrary potential designs in the large design space

# From Architecture Blueprints to Physical Systems



# Architecture-Level Energy Estimation and Design Exploration



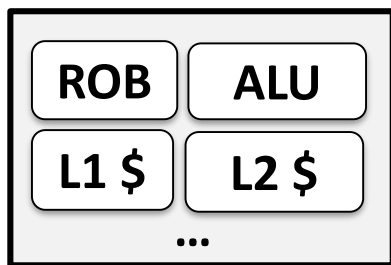
## Fast design space exploration

- Short simulations on architecture-level components
- Short turn-around time for each potential design

# Existing Architecture-Level Energy Estimators

- Architecture-level energy modeling for general purpose processors
  - Wattch[Brooks, ISCA2000], McPAT[Li, MICRO2009], GPUWattch[Leng, ISCA2013], PowerTrain[Lee, ISLPED2015]

CPU/GPU-Centric  
Architecture Model

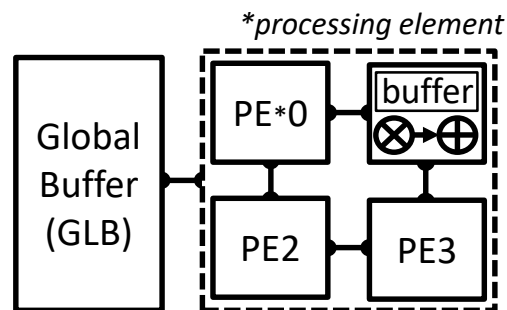
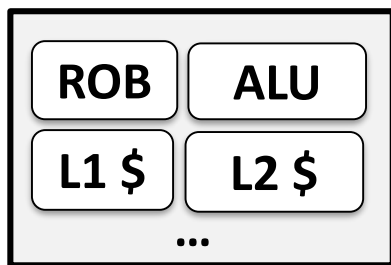


Use a fixed architecture template  
to represent the architecture

# Existing Architecture-Level Energy Estimators

- Architecture-level energy modeling for general purpose processors
  - Wattch[Brooks, ISCA2000], McPAT[Li, MICRO2009], GPUWattch[Leng, ISCA2013], PowerTrain[Lee, ISLPED2015]

CPU/GPU-Centric  
Architecture Model



The fixed template is not sufficient to describe various optimizations in the diverse accelerator design space

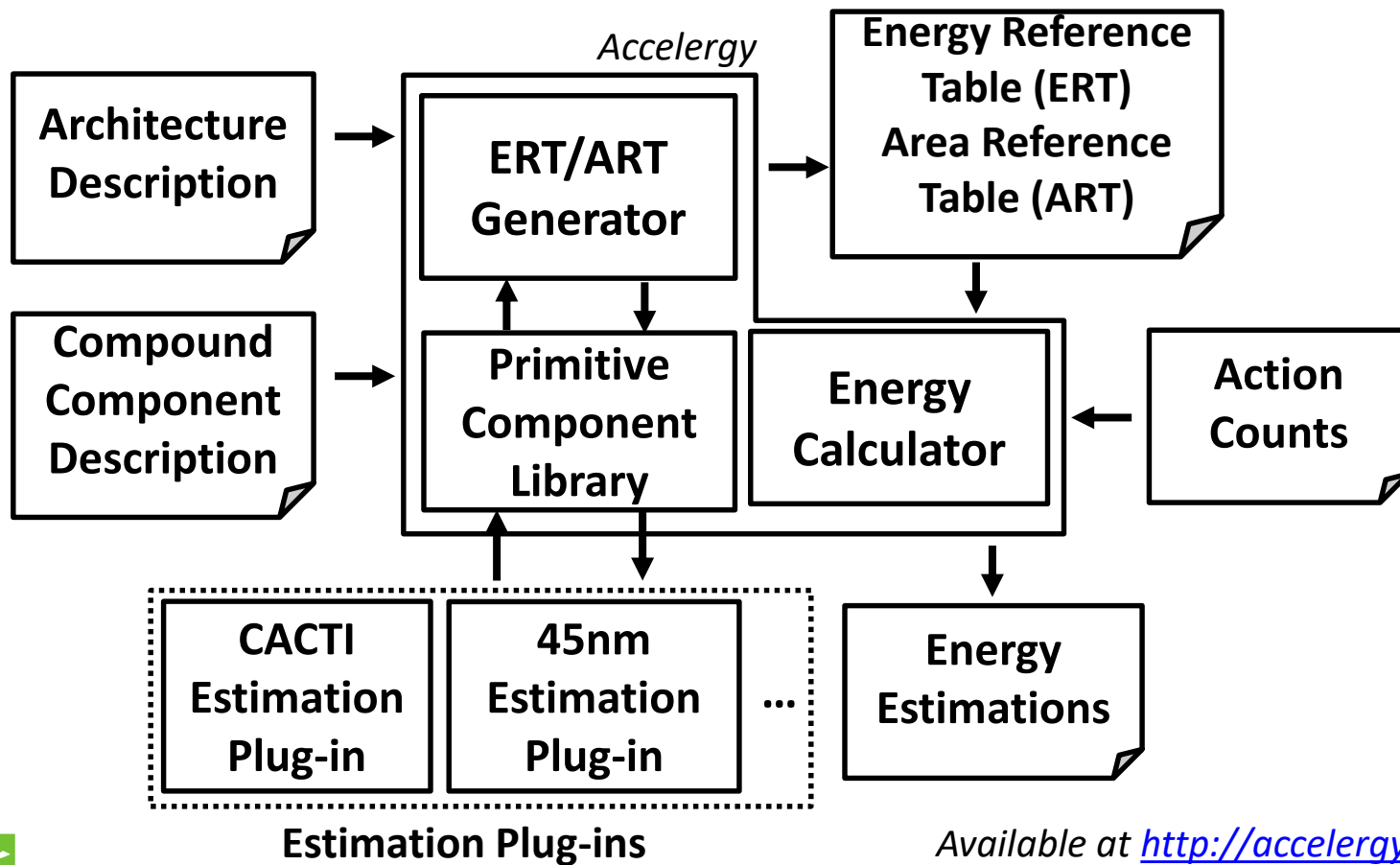
# Accelergy Overview

---

- **Accelergy Infrastructure**

- Performs architecture-level estimations to enable rapid design space exploration
- Supports modeling of diverse architectures with various underlying technologies
- Improves estimation accuracy by allowing fine-grained classification of components' runtime behaviors
- Supports succinct modeling of complicated architectures

# Accelergy High-Level Infrastructure

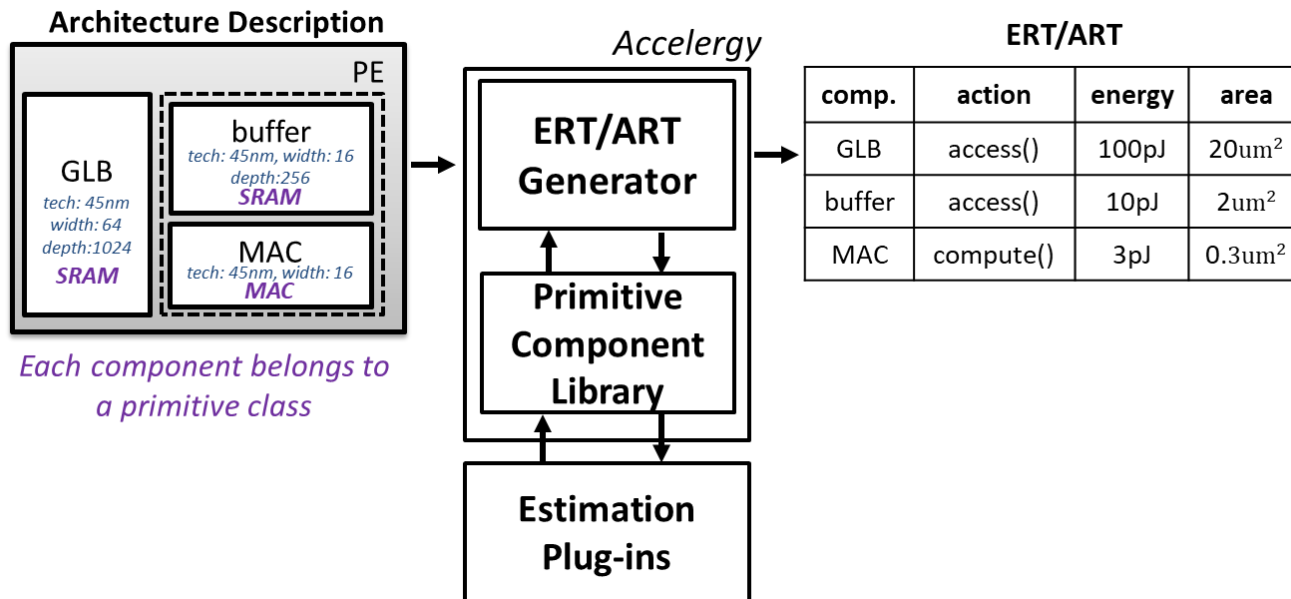


# How to use Accelergy?

- 1. Estimate architectures with primitive components**
2. Estimate architectures with compound components
3. Modeling with various underlying technologies

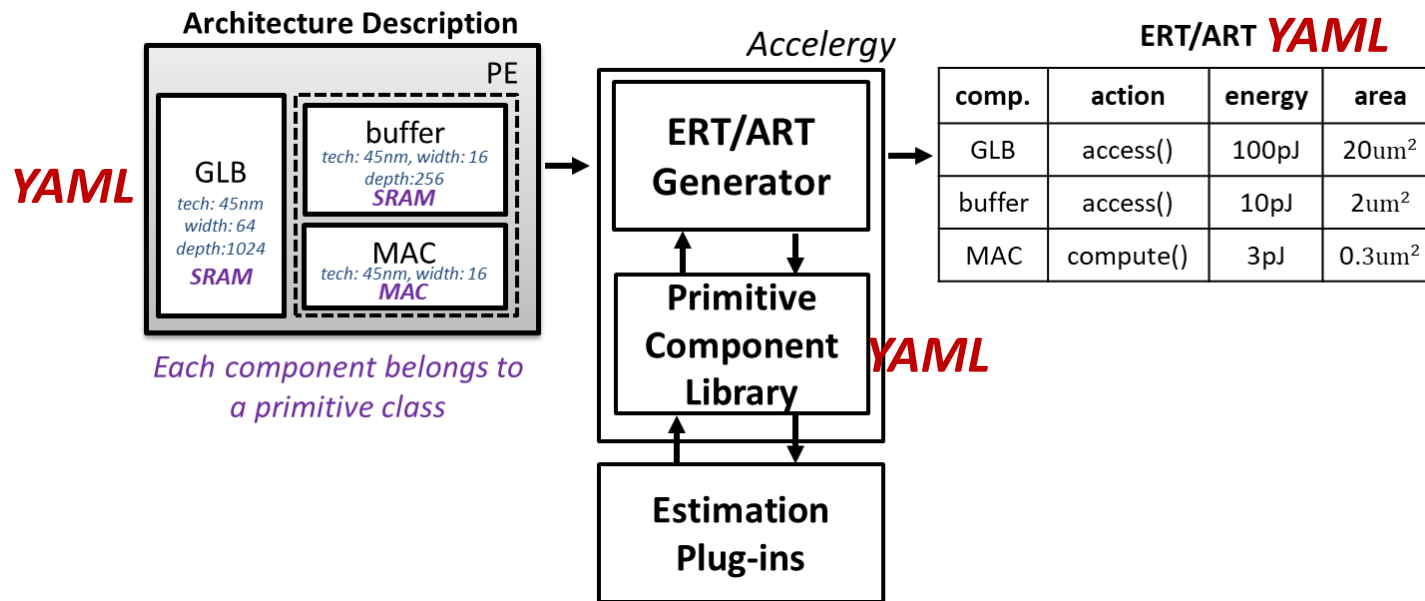
# Accelergy Modeling of a Simple Design

- A simple architecture can be modeled with primitive components
  - Step 01: Energy reference table generation



# Accelergy Modeling of a Simple Design

- A simple architecture can be modeled with primitive components
  - Step 01: Energy reference table generation



# Accelergy Modeling of a Simple Design

- **Primitive Component Library**
  - Describes the following properties of the primitive component classes
    - Hardware attributes

*User-defined  
attributes names*

*Default  
attribute values*

```
version: 0.3
classes:
  - name: bitwise
    attributes:
      technology: 65nm
      datawidth: 16
    actions:
      - name: process
      - name: idle
  - name: intadder
    attributes:
      technology: 65nm
      datawidth: 16
      num_pipeline_stages: 1
    actions:
      - name: add
      - name: idle
      ...
```

# Accelergy Modeling of a Simple Design

- **Primitive Component Library**
  - Describes the following properties of the primitive component classes
    - Hardware attributes
    - Associated actions

*User-defined  
attributes names*

```
version: 0.3
classes:
  - name: bitwise
    attributes:
      technology: 65nm
      datawidth: 16
    actions:
      - name: process
      - name: idle
  - name: intadder
    attributes:
      technology: 65nm
      datawidth: 16
      num_pipeline_stages: 1
    actions:
      - name: add
      - name: idle
      ...
```

*Default  
attribute values*

*User-defined  
action names*

# Accelergy Modeling of a Simple Design

- **Primitive Component Library**

- Describes the following properties of the primitive component classes

- Hardware attributes
    - Associated actions

*User-defined  
attributes names*

- Accelergy comes with a set of primitive component classes by default

- Users can add their own primitive component classes via the *accelergy\_config* file

- Default *accelergy\_config* file generated at:  
~/.config/accelergy/accelergy\_config.yaml

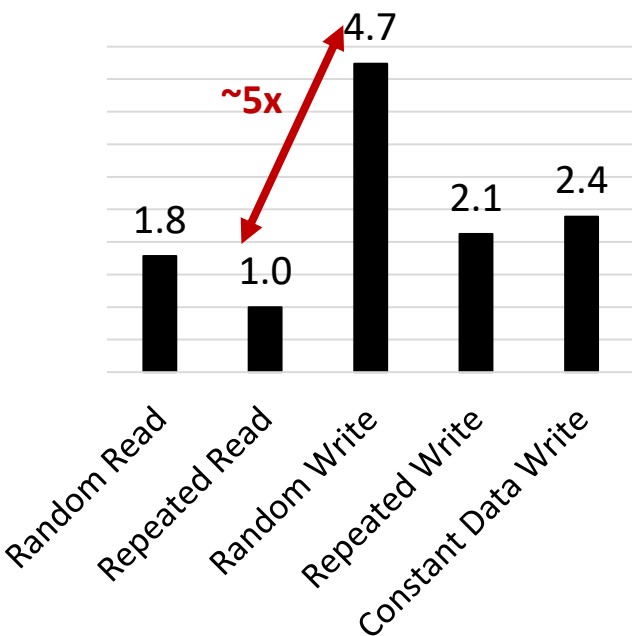
*(more details about the config file in the estimation plug-in section)*

```
version: 0.3
classes:
  - name: bitwise
    attributes:
      technology: 65nm
      datawidth: 16
    actions:
      - name: process
      - name: idle
  - name: intadder
    attributes:
      technology: 65nm
      datawidth: 16
      num_pipeline_stages: 1
    actions:
      - name: add
      - name: idle
      ...
```

*Default  
attribute values*

*User-defined  
action names*

# Actions with arguments



	Action Name	Argument	
		data_delta	address_delta
Repeated read	read	0	0
Random read		1	1
Repeated write	write	0	0
Random write		1	1
Repeated data write		0	1

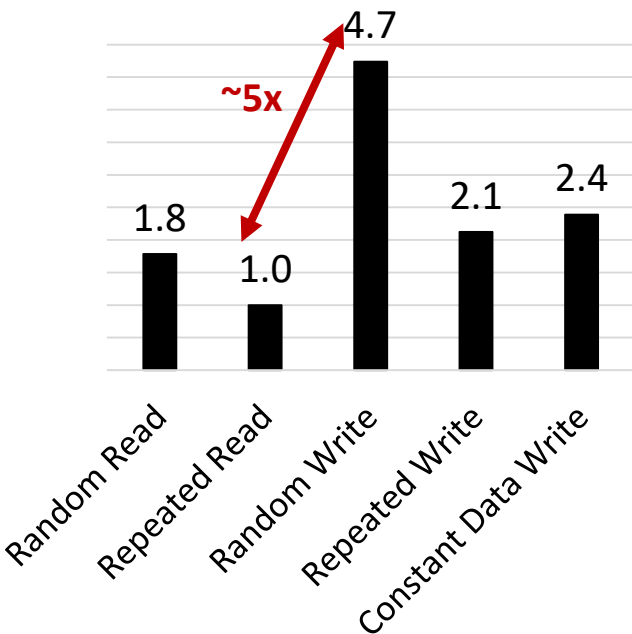


How much does  
data wires switch?  
0: idle, 1: active



How much does  
address wires switch?  
0: idle, 1: active

# Actions with arguments



	Action Name	Argument	
		data_delta	address_delta
Repeated read	read	0	0
Random read		1	1
Repeated write	write	0	0
Random write		1	1
Repeated data write		0	1



How much does  
data wires switch?  
0: idle, 1: active



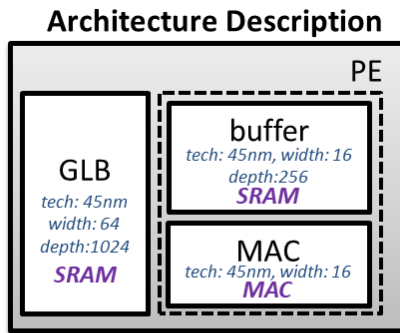
How much does  
address wires switch?  
0: idle, 1: active

```

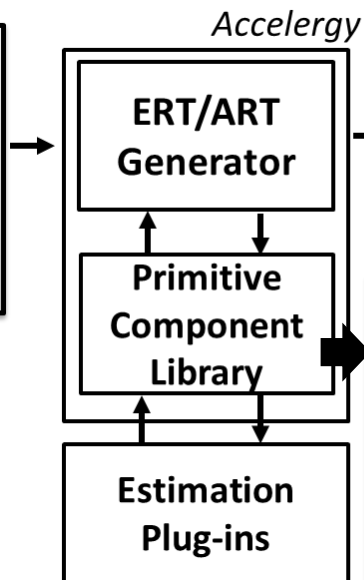
...
- name: regfile
  attributes:
    technology: 45nm
    width: 16
    depth: 1
    n_ports: 2
  actions:
    - name: read
      arguments:
        data_delta: 0..1
        address_delta: 0..1
    - name: write
      arguments:
        data_delta: 0..1
        address_delta: 0..1
    - name: idle
  
```

# Accelergy Modeling of a Simple Design

## Architecture Description



*Each component belongs to a primitive class*



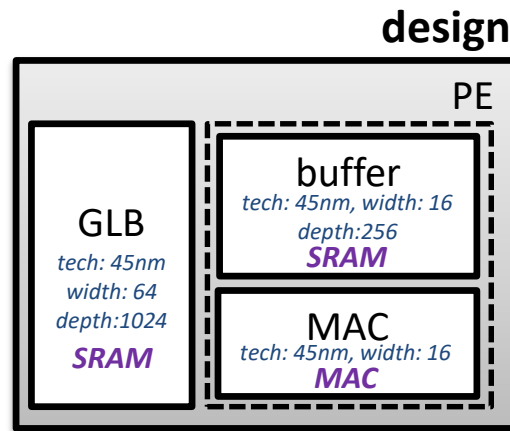
## ERT/ART

comp.	action	energy	area
GLB	access()	100pJ	20um <sup>2</sup>
buffer	access()	10pJ	2um <sup>2</sup>
MAC	compute()	3pJ	0.3um <sup>2</sup>

```
version: 0.3
classes:
- name: bitwise
  attributes:
    technology: 65nm
    datawidth: 16
  actions:
    - name: process
    - name: idle
- name: intadder
  attributes:
    technology: 65nm
    datawidth: 16
    num_pipeline_stages: 1
  actions:
    - name: add
    - name: idle
...
```

# Accelergy Modeling of a Simple Design

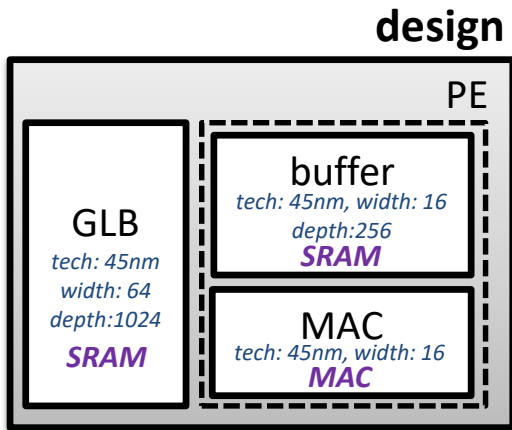
- Architecture Description
  - Describes the following properties of the components in the architecture
    - Hierarchical relationships
    - Component classes
    - Hardware attributes



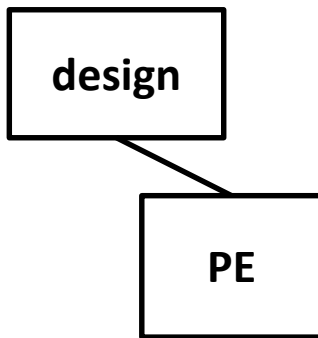
Architecture Description

# Architecture Description with Primitive Components

- Hierarchical represented using a tree structure



Architecture Description

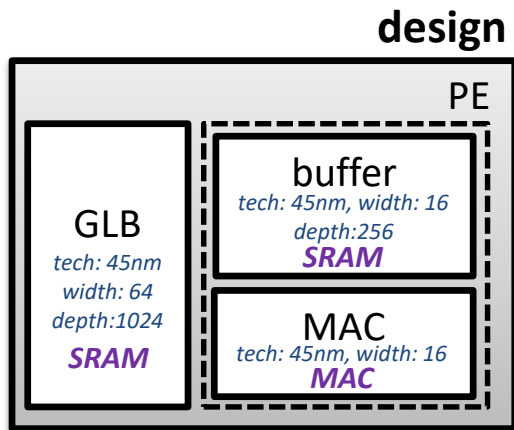


Architecture Tree

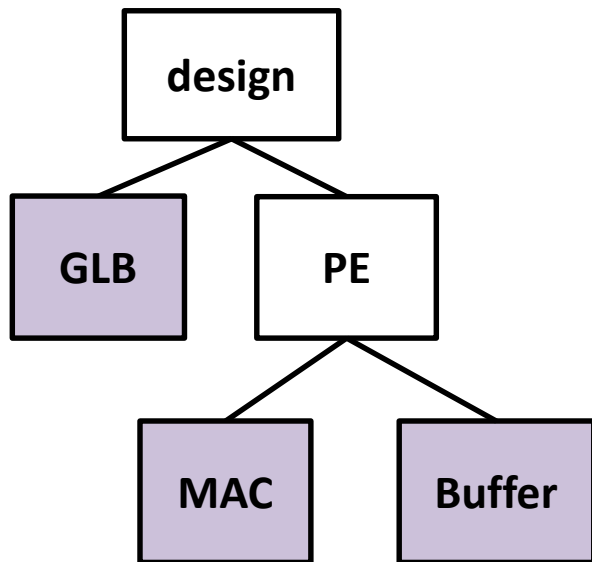
```
architecture:
version: 0.3
subtree:
  - name: design
    attributes:
      technology: 45nm
    local:
      - name: GLB
        class: SRAM
        attributes:
          width: 64
          depth: 1024
      subtree:
        - name: PE
          local:
            - name: buffer
              class: SRAM
              attributes: ...
            name: MAC
              class: MAC
              attributes: ...
```

# Architecture Description with Primitive Components

- Hierarchical represented using a tree structure



Architecture Description

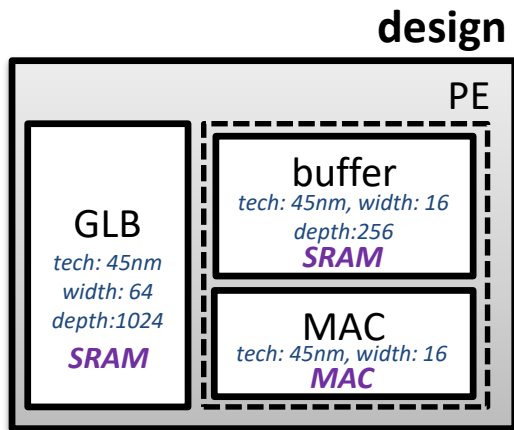


Architecture Tree

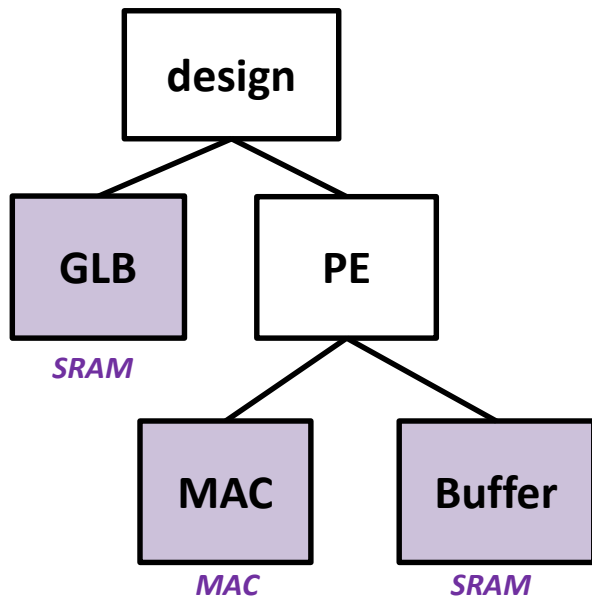
```
architecture:
version: 0.3
subtree:
  - name: design
    attributes:
      technology: 45nm
    local:
      - name: GLB
        class: SRAM
        attributes:
          width: 64
          depth: 1024
        subtree:
          - name: PE
            local:
              - name: buffer
                class: SRAM
                attributes: ...
              - name: MAC
                class: MAC
                attributes: ...
```

# Architecture Description with Primitive Components

- Specification of component classes



Architecture Description

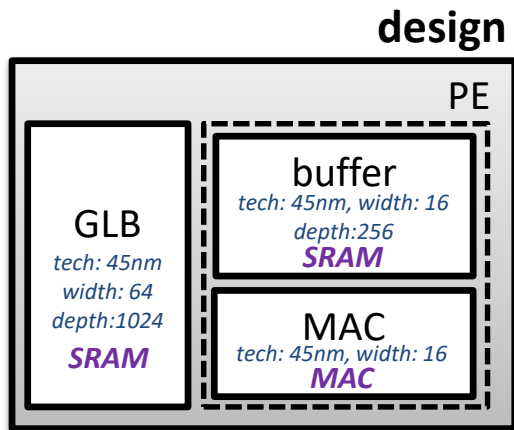


Architecture Tree

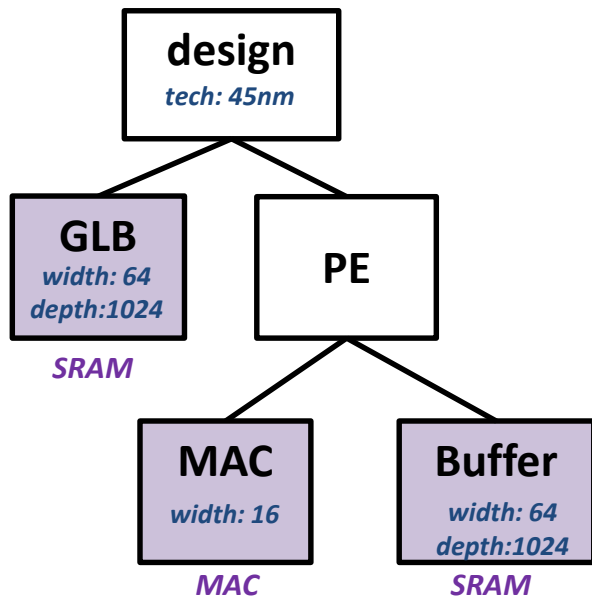
```
architecture:
version: 0.3
subtree:
  - name: design
    attributes:
      technology: 45nm
    local:
      - name: GLB
        class: SRAM
        attributes:
          width: 64
          depth: 1024
      subtree:
        - name: PE
          local:
            - name: buffer
              class: SRAM
              attributes:
                tech: 45nm, width: 16, depth: 256
            - name: MAC
              class: MAC
              attributes:
```

# Architecture Description with Primitive Components

- Hardware attributes defined for each component



Architecture Description



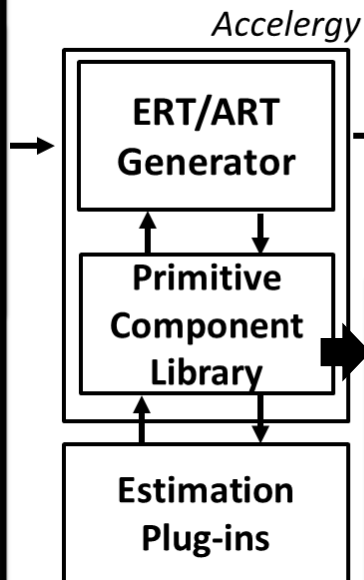
Architecture Tree

```
architecture:
version: 0.3
subtree:
- name: design
  attributes:
    technology: 45nm Global Attributes
  local:
    - name: GLB
      class: SRAM
      attributes:
        width: 64
        depth: 1024
      subtree:
        - name: PE
          local:
            - name: buffer
              class: SRAM
              attributes: ...
            - name: MAC
              class: MAC
              attributes: ...
```

# Architecture Description with Primitive Components

## Architecture Description

```
architecture: ← Top key used to represent the file type
version: 0.3
subtree:
- name: design
  attributes:
    technology: 45nm
  local:
- name: GLB
  class: SRAM
  attributes:
    width: 64
    depth: 1024
  subtree:
- name: PE
  local:
- name: buffer
  class: SRAM
  attributes: ...
- name: MAC
  class: MAC
  attributes: ...
```



## ERT/ART

comp.	action	energy	area
GLB	access()	100pJ	20um <sup>2</sup>
buffer	access()	10pJ	2um <sup>2</sup>
MAC	compute()	3pJ	0.3um <sup>2</sup>

```
version: 0.3
classes:
- name: bitwise
  attributes:
    technology: 65nm
    datawidth: 16
  actions:
- name: process
- name: idle
- name: intadder
  attributes:
    technology: 65nm
    datawidth: 16
    num_pipeline_stages: 1
  actions:
- name: add
- name: idle
...
```

# Accelergy Modeling of a Simple Design

- **Energy Reference Table**
  - List component in a flattened fashion with component names that reflect hierarchy

ERT:

version: 0.3

tables:

- name: design.PE.MAC

actions:

- name: mac\_random

arguments: null

energy: 2.2

- name: mac\_reused

- name: design.PE.buffer

actions:

- name: read

arguments:

address\_delta: 0

data\_delta: 0

energy: 0.006

- name: read

arguments:

address\_delta: 0

data\_delta: 1

energy: 0.144

# Accelergy Modeling of a Simple Design

- **Energy Reference Table**
  - List component in a flattened fashion with component names that reflect hierarchy
  - Describes the energy/action values (pJ) of the actions associated with each component

If an action has arguments, all of the possible combination of argument values are listed

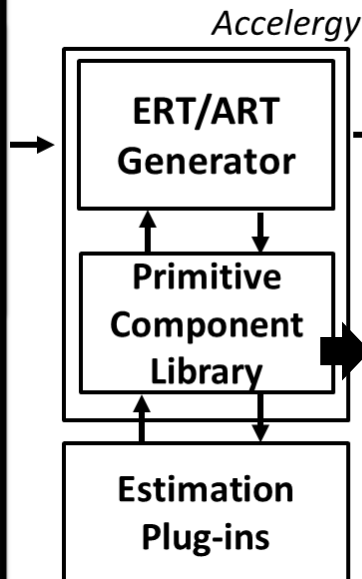
ERT:

```
version: 0.3
tables:
- name: design.PE.MAC
  actions:
  - name: mac_random
    arguments: null
    energy: 2.2
  - name: mac_reused
    ...
- name: design.PE.buffer
  actions:
  - name: read
    arguments:
      address_delta: 0
      data_delta: 0
    energy: 0.006
  - name: read
    arguments:
      address_delta: 0
      data_delta: 1
    energy: 0.144
  ...
```

# Exercise 01: Simple Architecture ERT/ART Generation

## Architecture Description

architecture: **← Top key used to represent the file type**  
version: 0.3  
subtree:  
- name: design  
  attributes:  
    technology: 45nm  
  local:  
    - name: GLB  
      class: SRAM  
      attributes:  
        width: 64  
        depth: 1024  
    subtree:  
      - name: PE  
        local:  
          - name: buffer  
            class: SRAM  
            attributes: ...  
          - name: MAC  
            class: MAC  
            attributes: ...



## ERT/ART

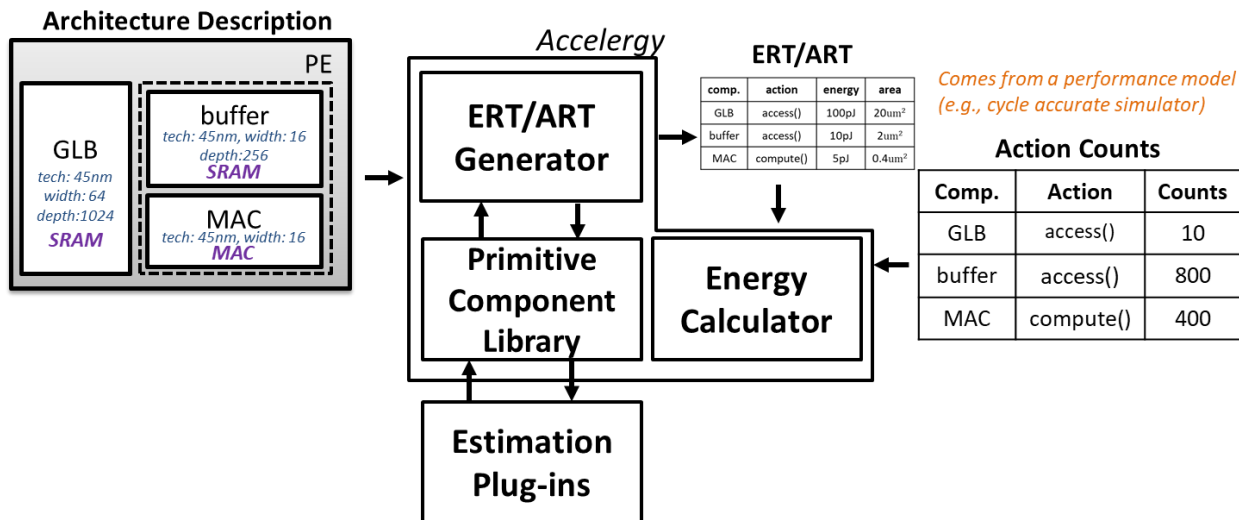
comp.	action	end
GLB	access()	10
buffer	access()	10
MAC	compute()	3

version: 0.3  
classes:  
- name: bitwise  
  attributes:  
    technology: 65nm  
    datawidth: 16  
  actions:  
    - name: process  
    - name: idle  
- name: intadder  
  attributes:  
    technology: 65nm  
    datawidth: 16  
    num\_pipeline\_stages: 1  
  actions:  
    - name: add  
    - name: idle  
  ...

ERT:  
version: 0.3  
tables:  
- name: design.PE.MAC  
  actions:  
    - name: mac\_random  
      arguments: null  
      energy: 2.2  
    - name: mac\_reused  
      ...  
- name: design.PE.buffer  
  actions:  
    - name: read  
      arguments:  
        address\_delta: 0  
        data\_delta: 0  
        energy: 0.006  
    - name: read  
      arguments:  
        address\_delta: 0  
        data\_delta: 1  
        energy: 0.144  
    ...

# Accelergy Modeling of a Simple Design

- A simple architecture can be modeled with primitive components
  - Step 01: Energy reference table generation
  - Step 02: Energy calculation with action counts



# Accelergy Modeling of a Simple Design

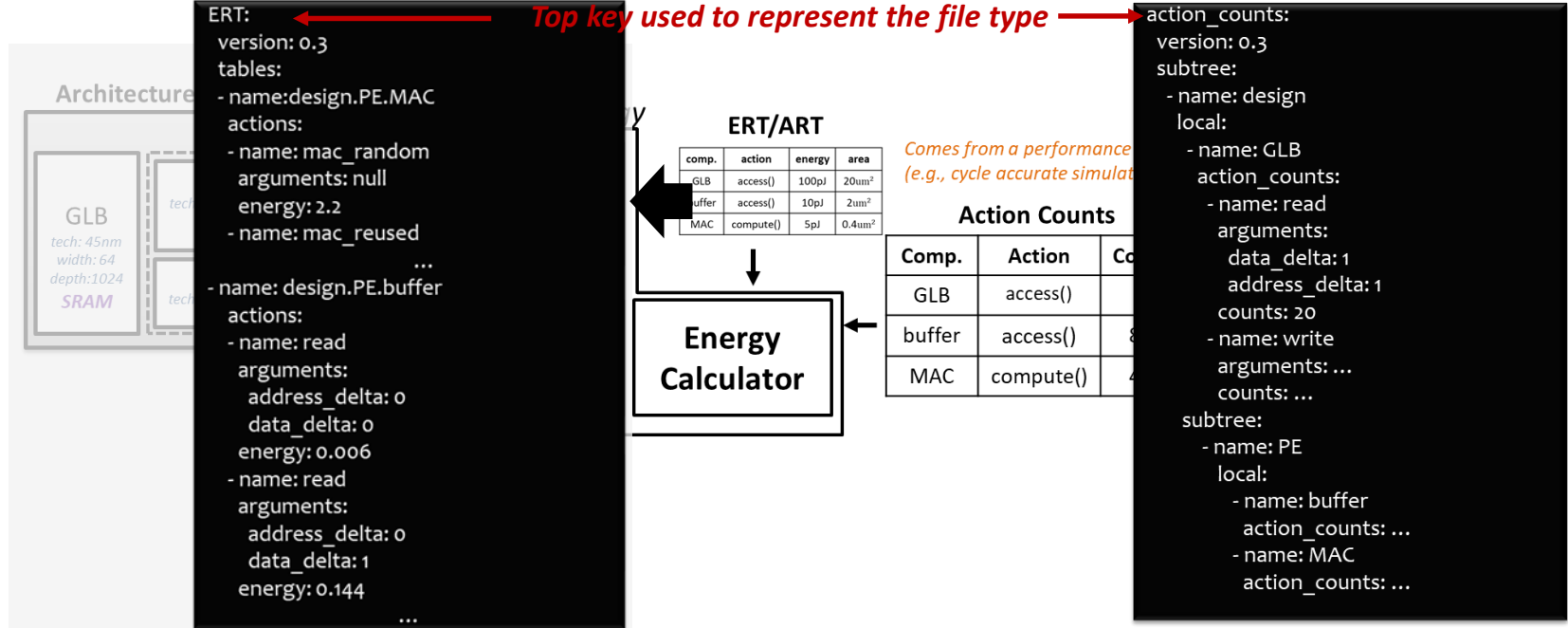
- Action counts
  - List the components in a hierarchical/flattened fashion
  - For each component, describes the number of times each action has occurred during the run of a specific workload

Action and argument names must match with those defined in the ERT

```
action_counts:
  version: 0.3
  subtree:
    - name: design
      local:
        - name: GLB
          action_counts:
            - name: read
              arguments:
                data_delta: 1
                address_delta: 1
              counts: 20
            - name: write
              arguments: ...
              counts: ...
          subtree:
            - name: PE
              local:
                - name: buffer
                  action_counts: ...
                - name: MAC
                  action_counts: ...
```

# Exercise 02: Simple Architecture Energy Calculation

- Energy calculation with existing Energy Reference Table



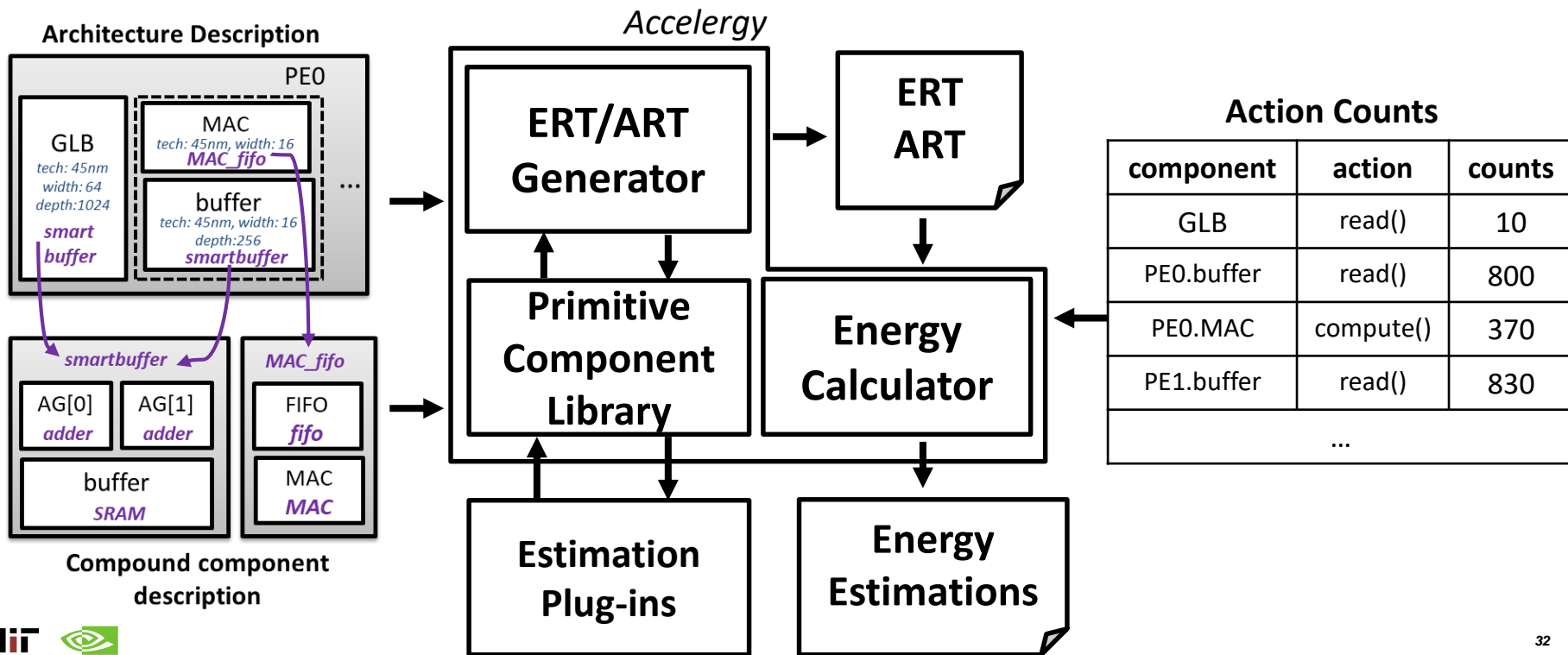
Allows us to quickly iterate through multiple runtime simulation results of various workloads

## How to use Accelergy?

1. Estimate architectures with primitive components
- 2. Estimate architectures with compound components**
3. Modeling with various underlying technologies

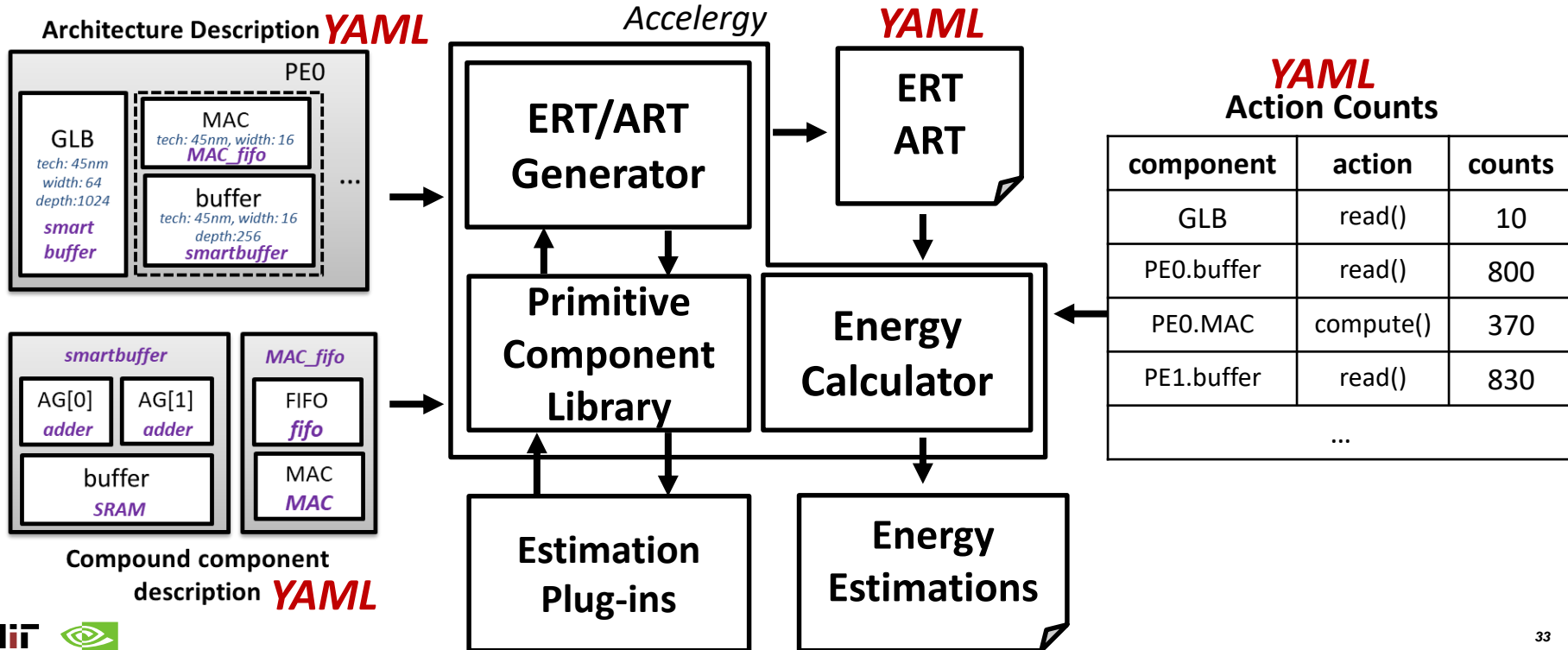
# Accelergy Modeling of Complicated Architectures

- Accelergy is able to succinctly model arbitrary complicated architectures with architecture description of user-defined compound component classes



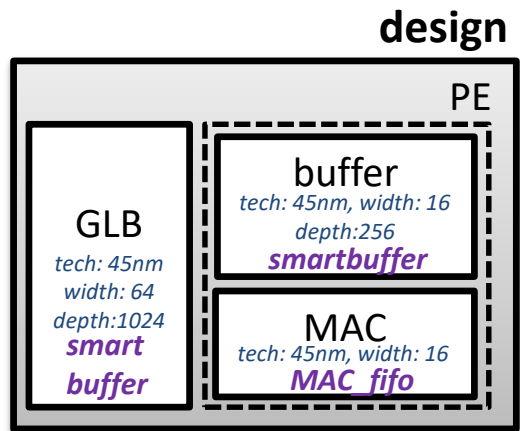
# Accelergy Modeling of Complicated Architectures

- Accelergy is able to succinctly model arbitrary complicated architectures with architecture description of user-defined compound component classes



# Accelergy Modeling of Complicated Architectures

- Architecture Description with user-defined compound component classes



Architecture Description

```
architecture:
version: 0.3
subtree:
- name: design
  attributes:
    technology: 45nm
  local:
    - name: GLB
      class: smartbuffer
      attributes:
        width: 64
        depth: 1024
      subtree:
        - name: PE
          local:
            - name: buffer
              class: smartbuffer
              attributes: ...
            - name: MAC
              class: MAC_fifo
              attributes: ...
```

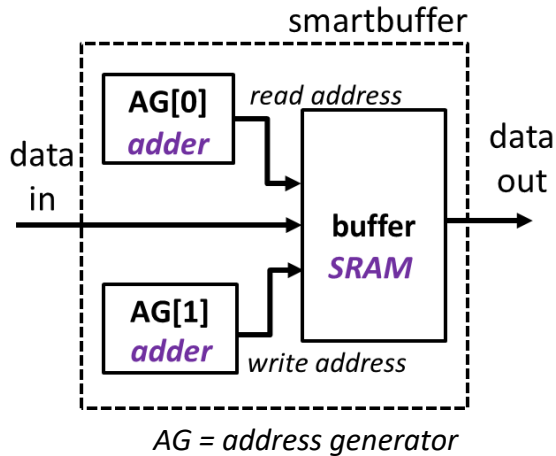
# Accelergy Modeling of Complicated Architectures

---

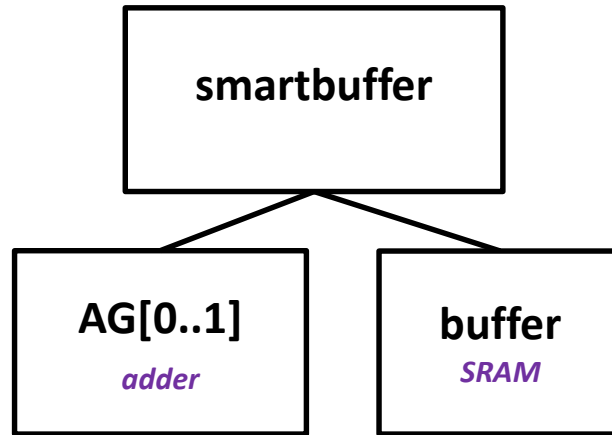
- **Compound component description**
  - **Define compound component hardware implementation**
    - 2-level tree representation of hardware implementations
    - Define hardware attributes for compound component class
  - Define compound actions associated with the compound component class
    - 2-level tree representation of action definition

# Compound Component Description

- 2-level tree representation of hardware implementations



**Compound  
Component Class**

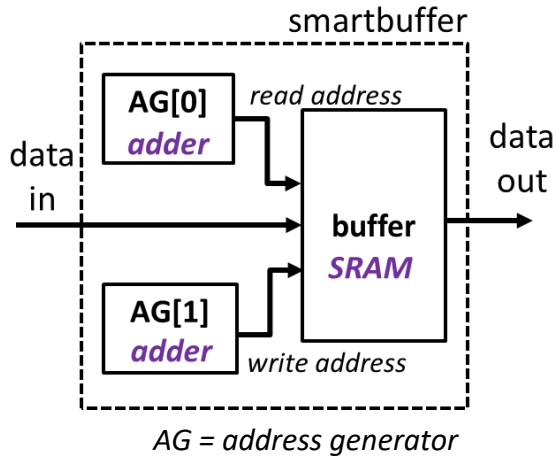


**Compound Component  
Hardware Structure Tree**

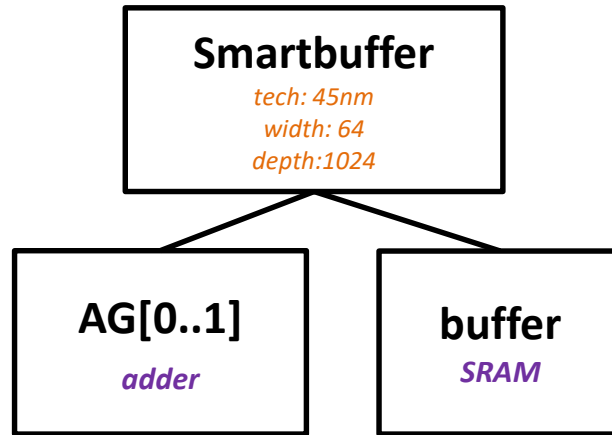
```
name: smartbuffer
attributes:
  technology: 45nm
  width: 64
  depth: 1024
subcomponents:
  - name: AGs[0..1]
    class: adder
    attributes:
      technology: technology
      width: log(depth)
  - name: buffer
    class: SRAM
    attributes:
      technology: technology
      width: width
      depth: depth
  ...
```

# Compound Component Description

- Define hardware attributes for compound component class



Compound  
Component Class

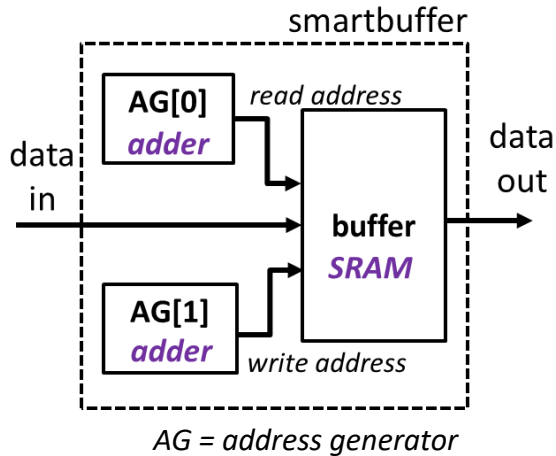


Compound Component  
Hardware Structure Tree

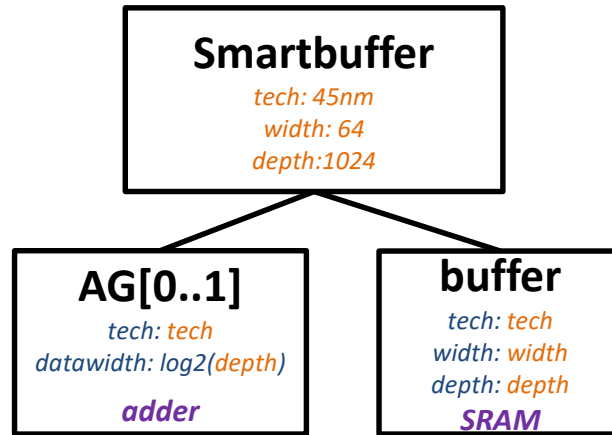
```
name: smartbuffer
attributes:
  technology: 45nm
  width: 64
  depth: 1024
subcomponents:
  - name: AGs[0..1]
    class: adder
    attributes:
      technology: technology
      width: log(depth)
  - name: buffer
    class: SRAM
    attributes:
      technology: technology
      width: width
      depth: depth
  ...
```

# Compound Component Description

- Define hardware attributes for compound component class



Compound  
Component Class

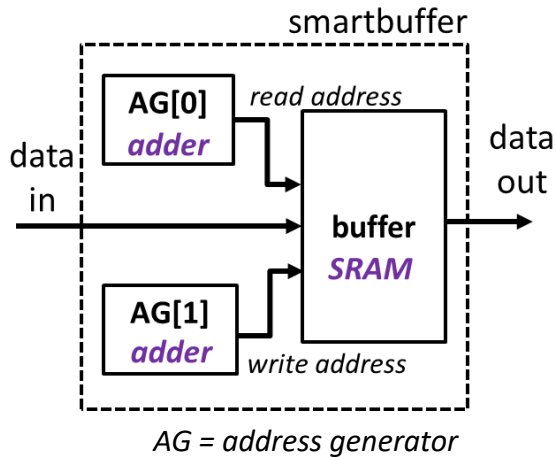


Compound Component  
Hardware Structure Tree

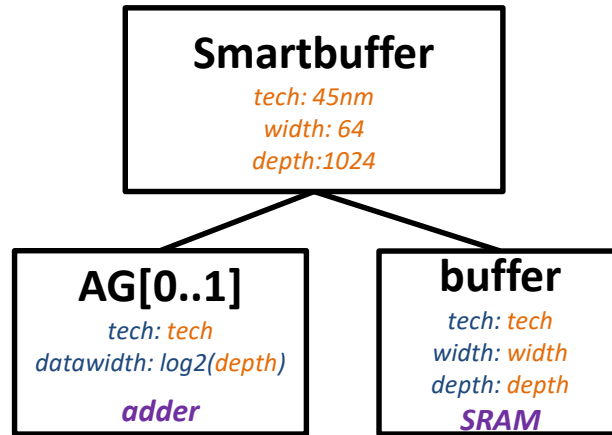
```
name: smartbuffer
attributes:
  technology: 45nm
  width: 64
  depth: 1024
subcomponents:
  - name: AGs[0..1]
    class: adder
    attributes:
      technology: technology
      width: log(depth)
  - name: buffer
    class: SRAM
    attributes:
      technology: technology
      width: width
      depth: depth
  ...
```

# Compound Component Description

- Define hardware attributes for compound component class



Compound  
Component Class



Compound Component  
Hardware Structure Tree

```
name: smartbuffer
attributes:
  technology: 45nm
  width: 64
  depth: 1024
subcomponents:
  - name: AGs[0..1]
    class: adder
    attributes:
      technology: technology
      width: log(depth)
  - name: buffer
    class: SRAM
    attributes:
      technology: technology
      width: width
      depth: depth
  ...
```

attribute mapping

arithmetic computation

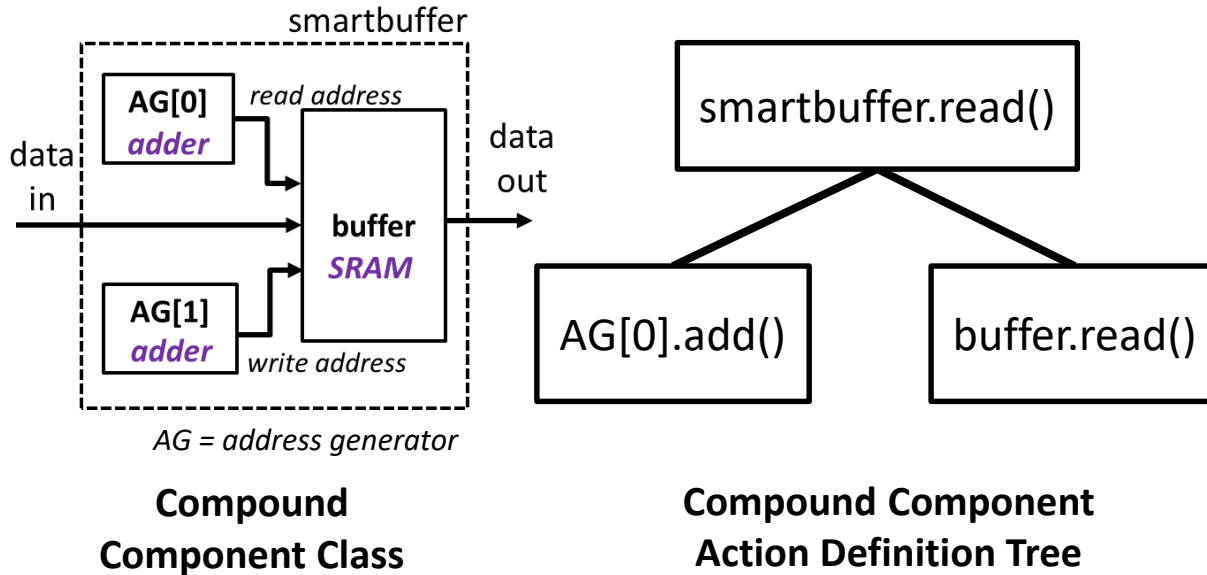
# Accelergy Modeling of Complicated Architectures

---

- **Compound component description**
  - Define compound component hardware implementation
    - 2-level tree representation of hardware implementations
    - Define hardware attributes for compound component class
  - **Define compound actions associated with the compound component class**
    - 2-level tree representation of action definition

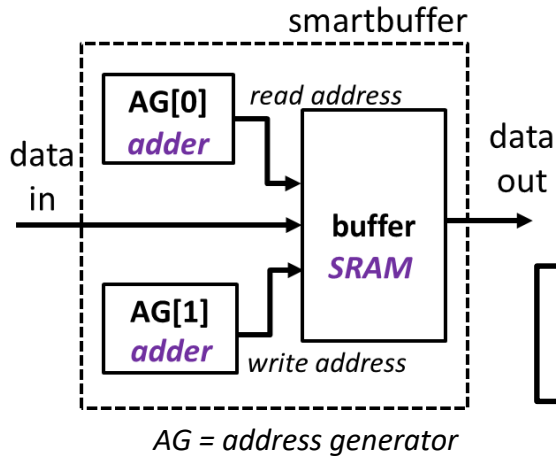
# Compound Component Description

- 2-level tree representation of action definition

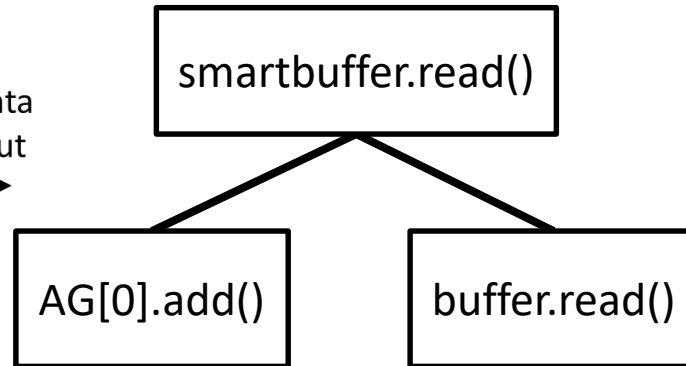


# Compound Component Description

- 2-level tree representation of action definition



**Compound  
Component Class**



**Compound Component  
Action Definition Tree**

```
name: smartbuffer
attributes: ...
subcomponents: ...
actions:
```

```
- name: read
  arguments:
    data_delta: 0..1
    address_delta: 0..1
  subcomponents:
```

```
- name: AG[0]
  actions:
    - name: add
    - name: buffer
  actions:
    - name: read
      arguments:
        data_delta: data_delta
        address_delta: address_delta
    - name: write
```

argument  
mapping

...

# Exercise 03: Architecture with Compound Components

```

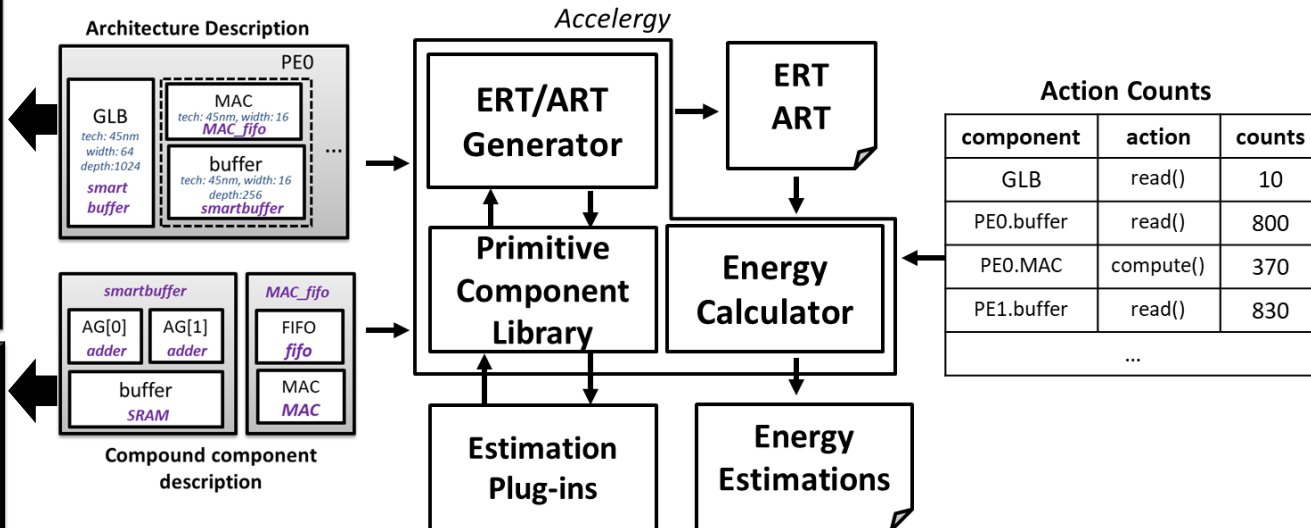
architecture:
version: 0.3
subtree:
- name: design
  attributes:
    technology: 45nm
  local:
    - name: GLB
      class: smartbuffer
      attributes:
        width: 64
        depth: 1024
    subtree:
      - name: PE
        local:
          - name: buffer
            class: smartbuffer
            attributes: ...
          - name: MAC
            class: MAC_fifo
            attributes: ...

```

```

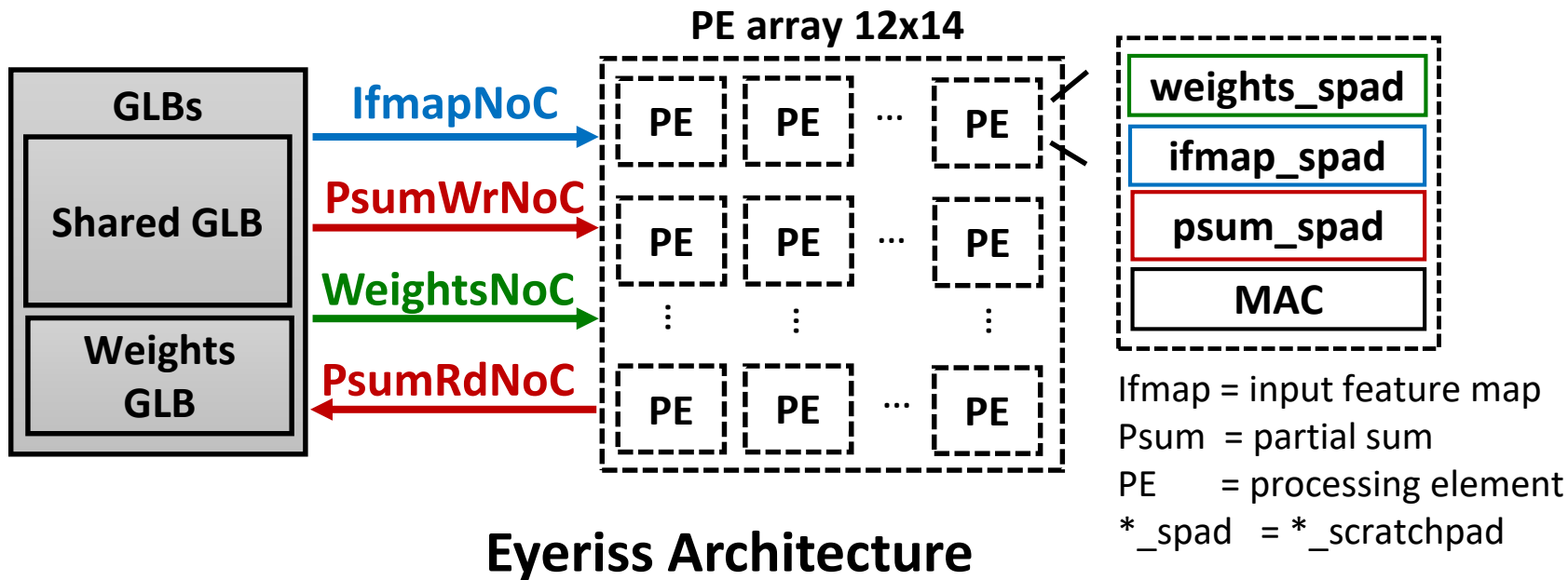
name: smartbuffer
attributes:
  technology: 45nm
  width: 64
  depth: 1024
subcomponents:
- name: AGs[0..1]
  class: adder
  attributes:
    technology: technology
    width: log(depth)
- name: buffer
  class: SRAM
  attributes:
    technology: technology
    width: width
    depth: depth
...

```



# Exercise 04: Example Eyeriss-like Architecture

- High-level Architecture



## How to use Accelergy?

1. Estimate architectures with primitive components
2. Estimate architectures with compound components
3. **Modeling with various underlying technologies**

# Energy Estimation Plug-ins for Various Technologies

---

- **Accelergy automatically locates all the plug-ins according to its config file**

```
version: 0.3
estimator_plug_ins:
  - /usr/local/share/accelergy/estimation_plug_ins
primitive_components:
  - /usr/local/share/accelergy/primitive_component_libs
```

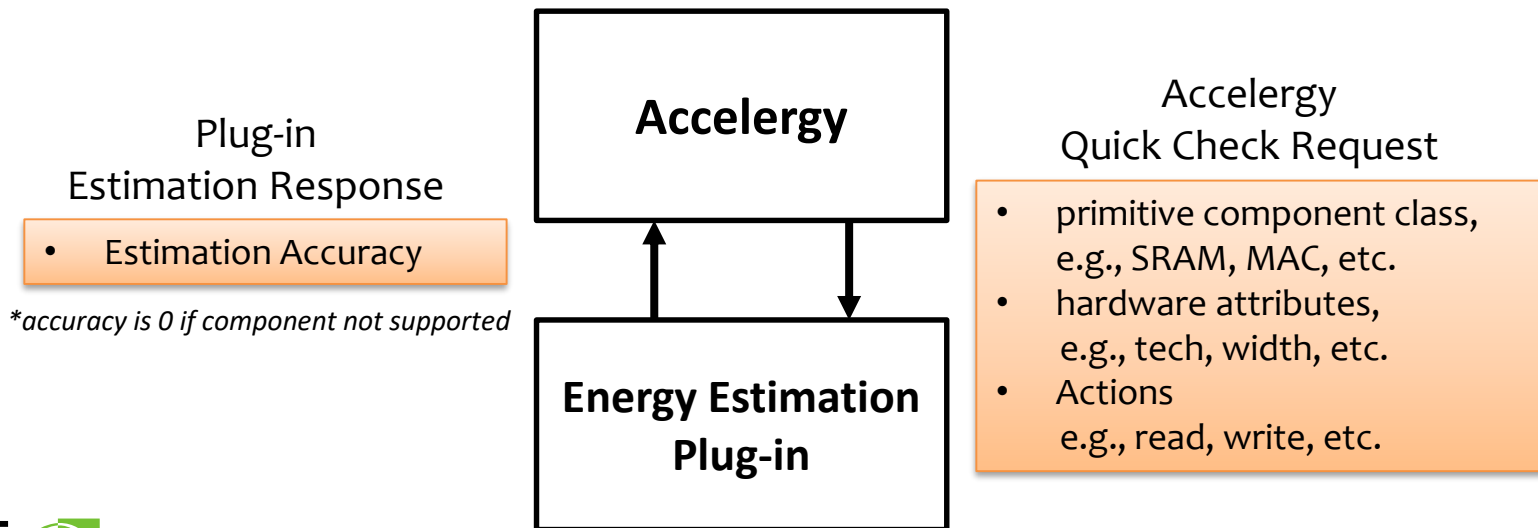
Accelergy Config File

~/.config/accelergy/accelergy\_config.yaml

**Automatically created by the first run of Accelergy**

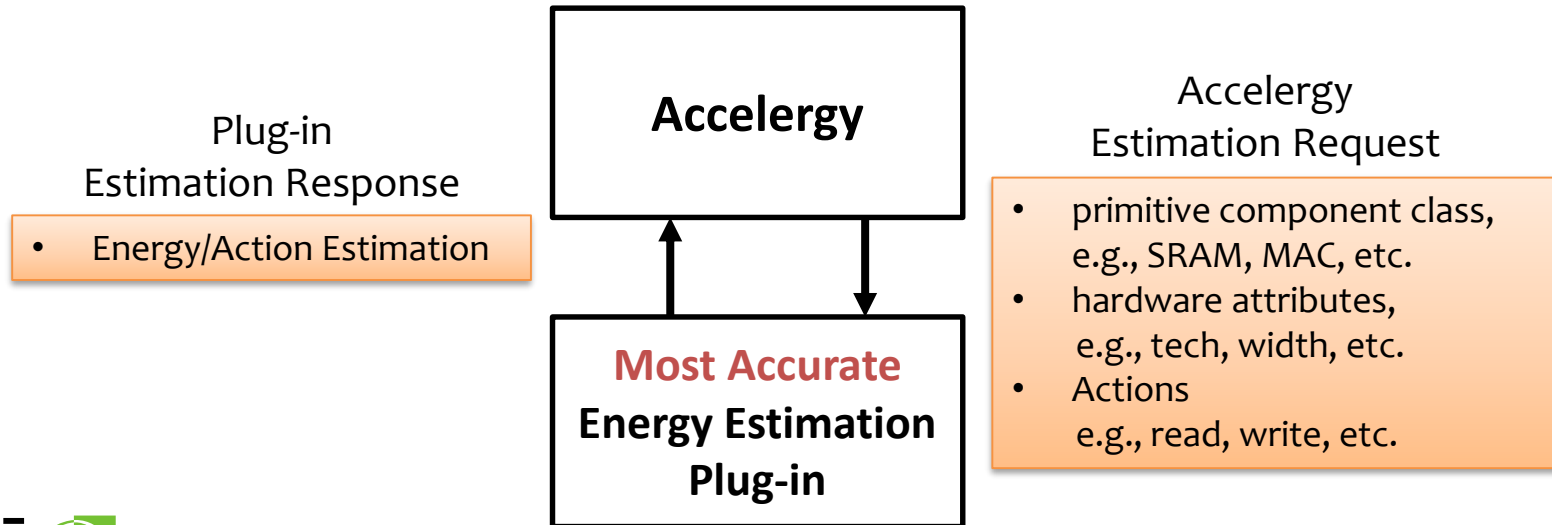
# Energy Estimation Plug-ins for Various Technologies

- Interaction interface between Accelergy and estimation plug-ins
  - Step 1: collect accuracy from estimation plug-ins (quick check)
  - Step 2: pick the most accurate plug-in for estimations (potentially time-consuming estimation)



# Energy Estimation Plug-ins for Various Technologies

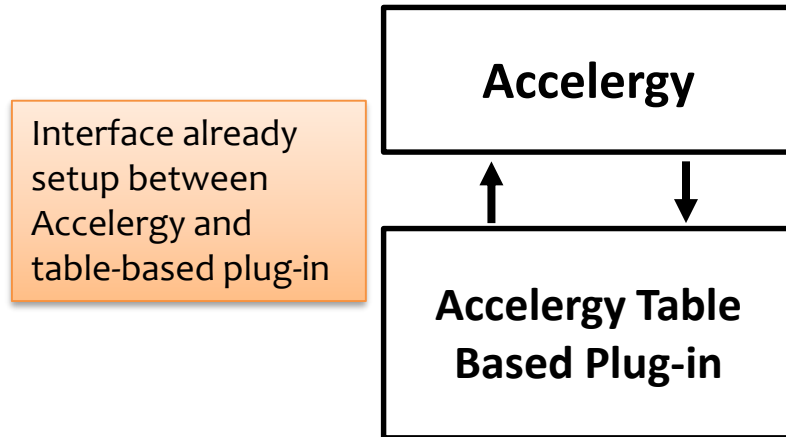
- Interaction interface between Accelergy and estimation plug-ins
  - Step 1: collect accuracy from estimation plug-ins (quick check)
  - Step 2: pick the most accurate plug-in for estimations (potentially time-consuming estimation)



# Energy Estimation Plug-ins for Various Technologies

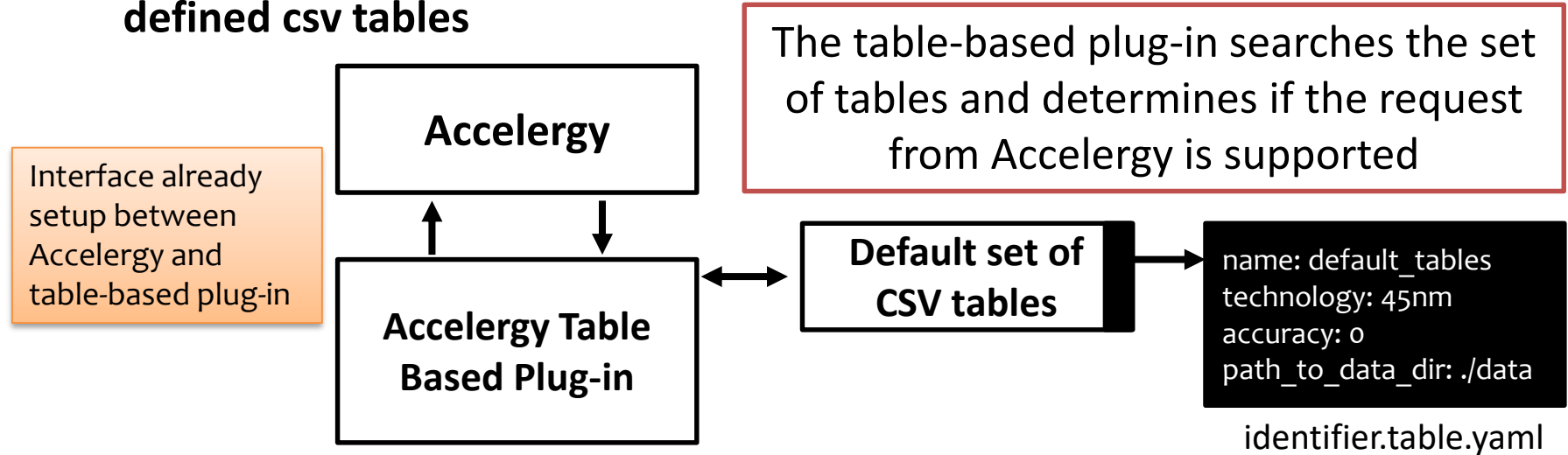
---

- What if none of the open sourced plug-in supports my components?
  - Accelergy provides a table-based-plug-in for easy plug-and-chug of user defined csv tables



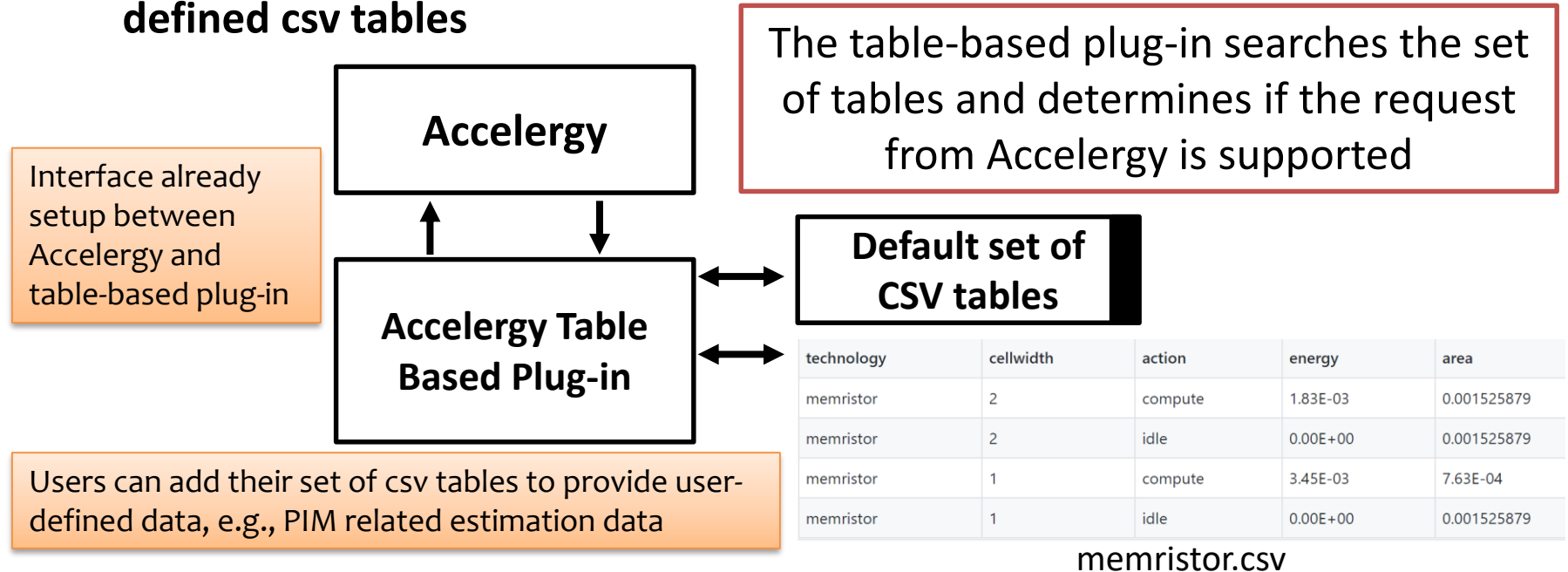
# Energy Estimation Plug-ins for Various Technologies

- What if none of the open sourced plug-in supports my components?
  - Accelergy provides a table-based-plug-in for easy plug-and-chug of user defined csv tables



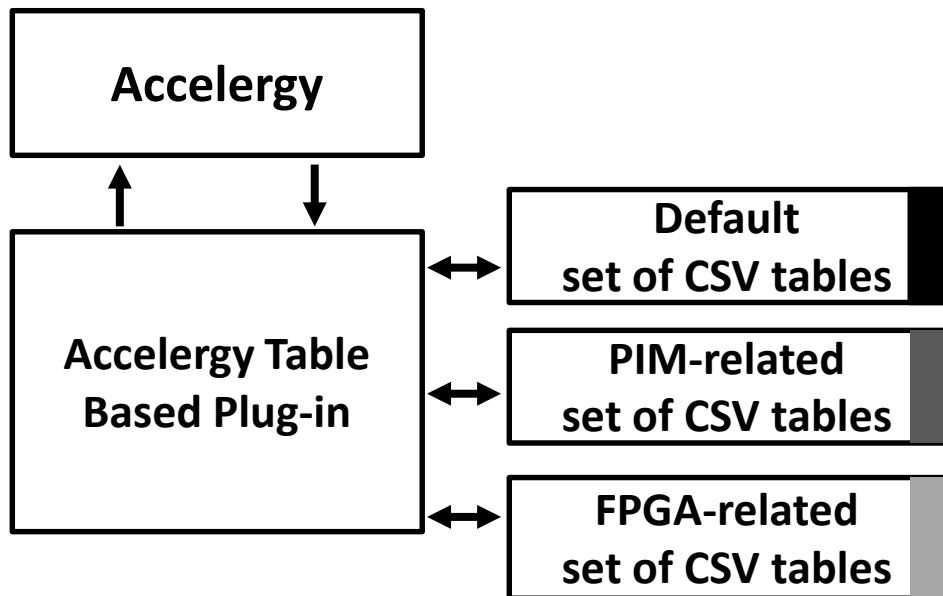
# Energy Estimation Plug-ins for Various Technologies

- What if none of the open sourced plug-in supports my components?
  - Accelergy provides a table-based-plug-in for easy plug-and-chug of user defined csv tables



# Energy Estimation Plug-ins for Various Technologies

- Specifies the roots of the user-defined tables in the Accelergy config file



```
version: 0.3
estimator_plug_ins:
  - /usr/local/share/accelergy/estimation_plug_ins
primitive_components:
  - /usr/local/share/accelergy/primitive_component_libs
table_plug_ins:
  roots:
    - .../accelergy-table-based-plug-ins/set_of_table_templates
    - <path-to-pim-related-csv>
    - <path-to-FPGA-related-csv-root>
```

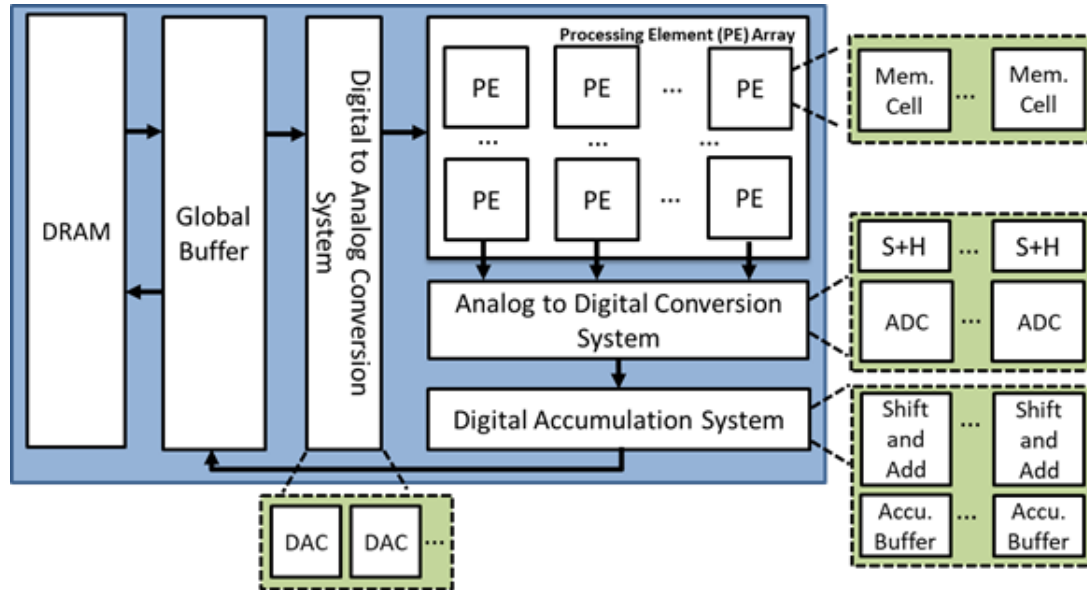
## Accelergy Config File

`~/.config/accelergy/accelergy_config.yaml`

Command to add a root:  
`accelergyTables -r <path-to-pim-related-csv-folder>`

# Exercise 05: Modeling of a Processing in memory based Architecture

- High-level PIM architecture



# Other Exercise/Baselines

---

- **exercises/timeloop+accelergy**
  - mapping exploration with an integer based eyeriss-like architecture
  - mapping exploration with an floating point based eyeriss-like architecture
- **baseline\_designs/**
  - Various popular baseline architectures
  - Example workload specifications