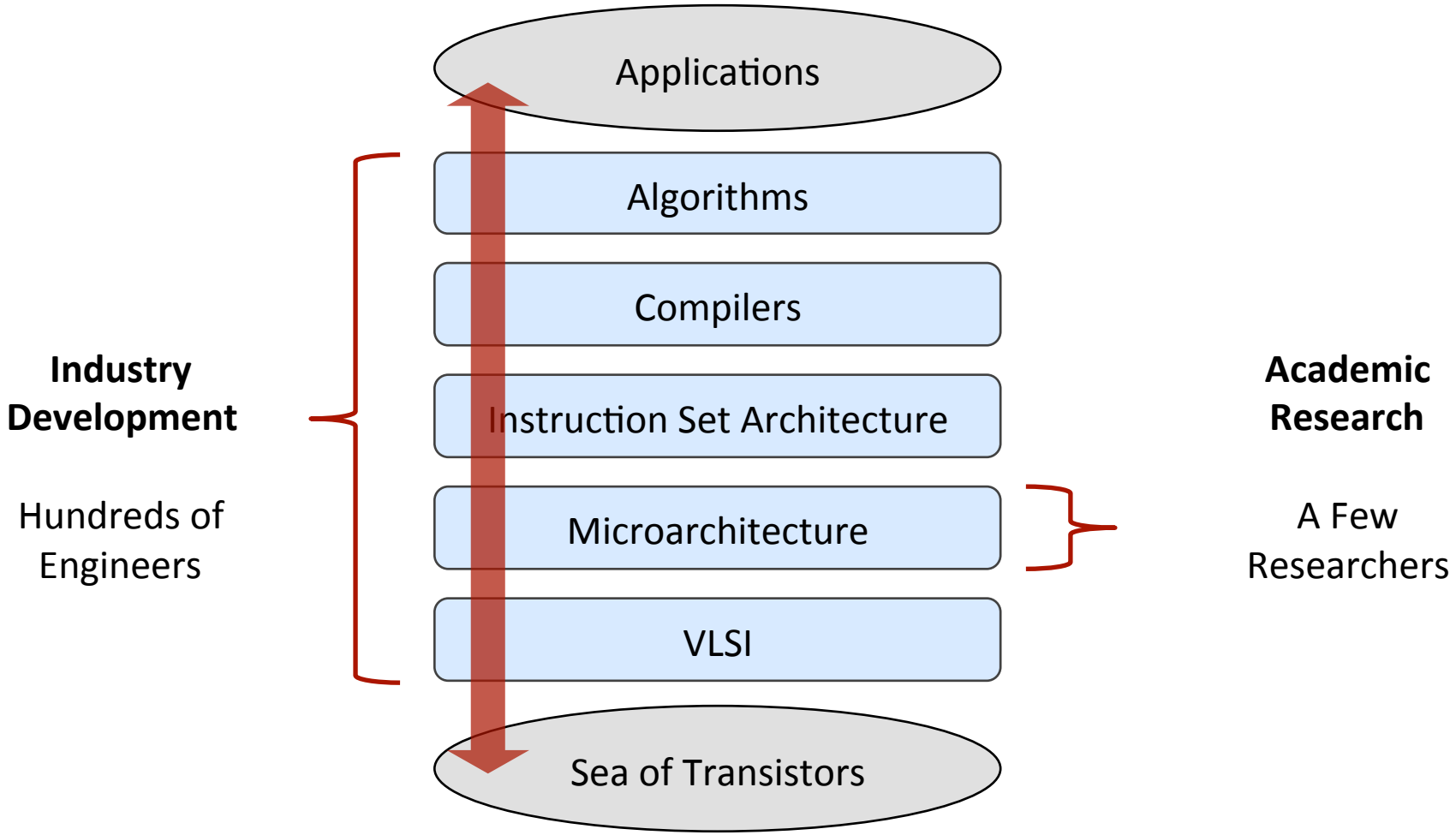
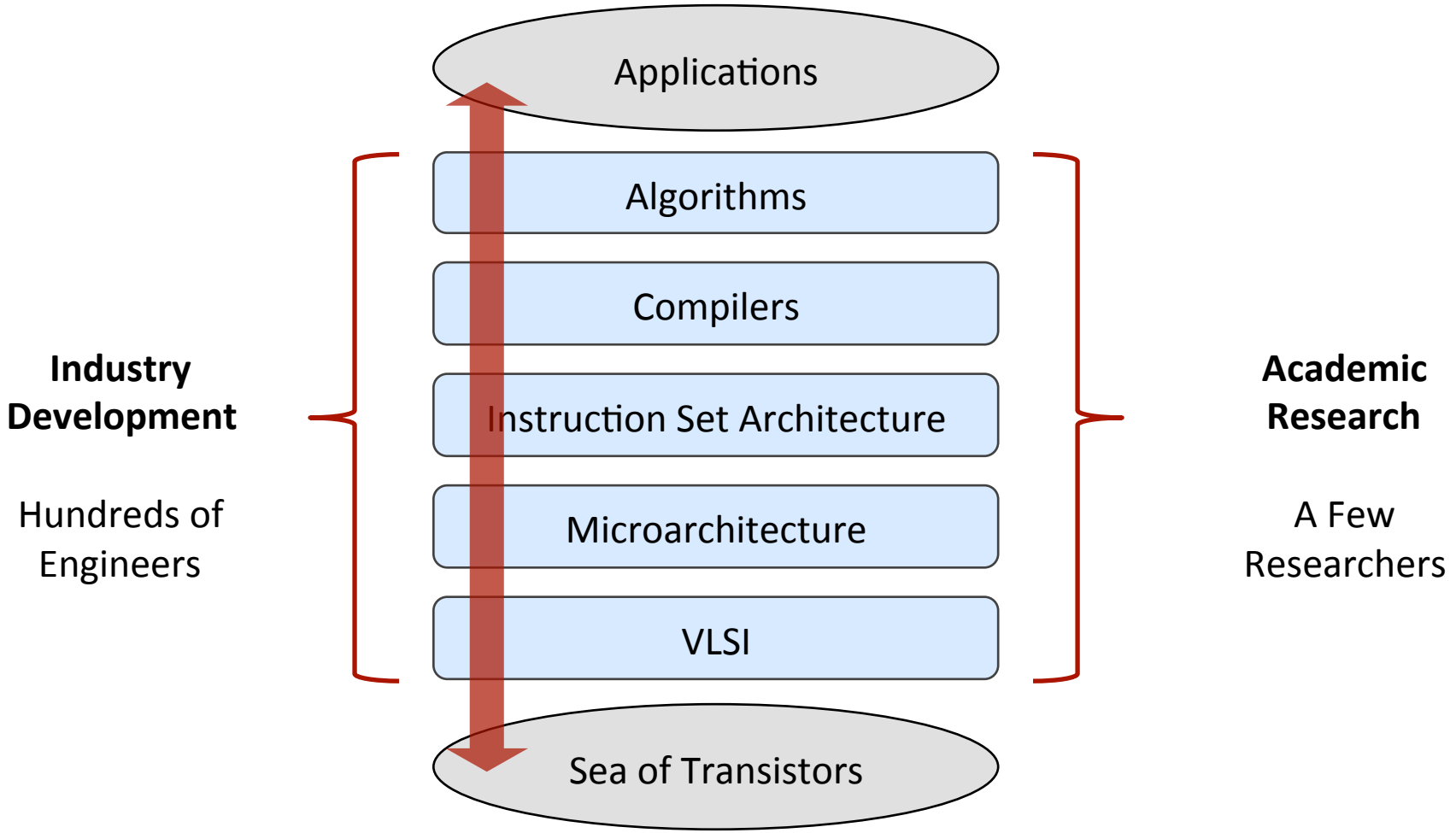
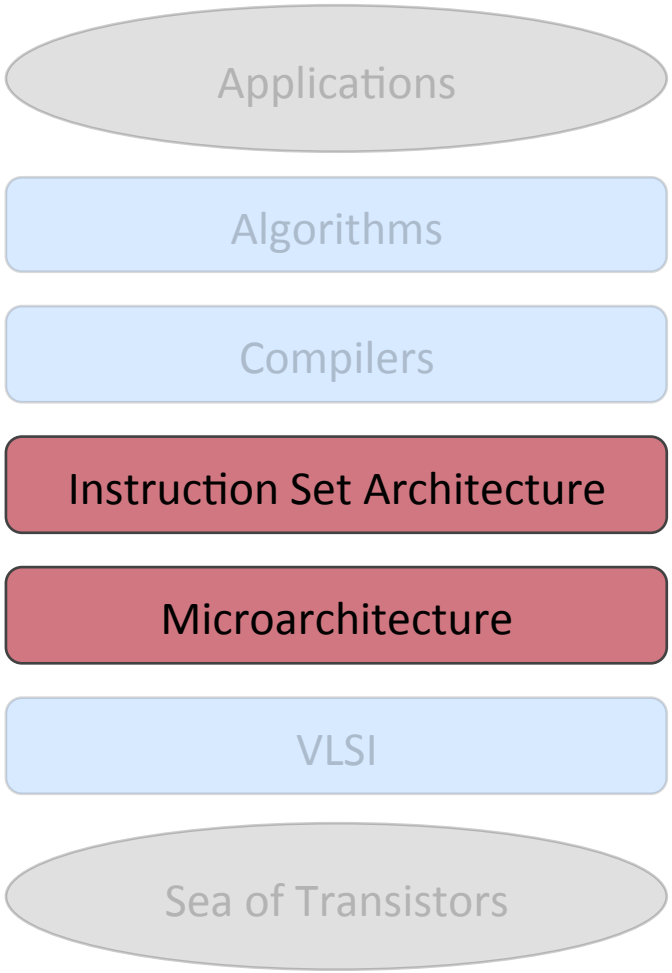
Computer Architecture Research Abstractions



Computer Architecture Research Abstractions



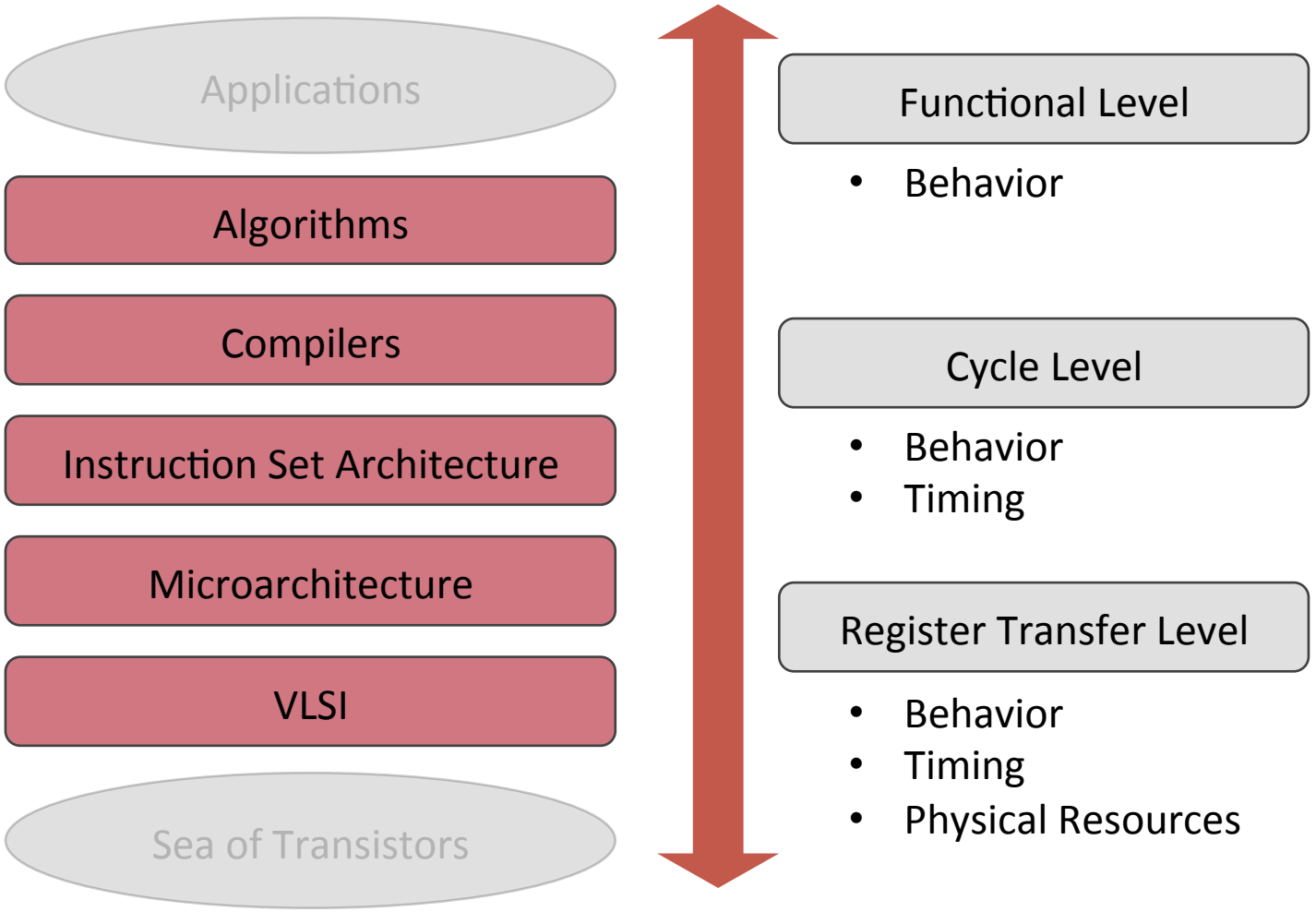
Computer Architecture Research Methodologies



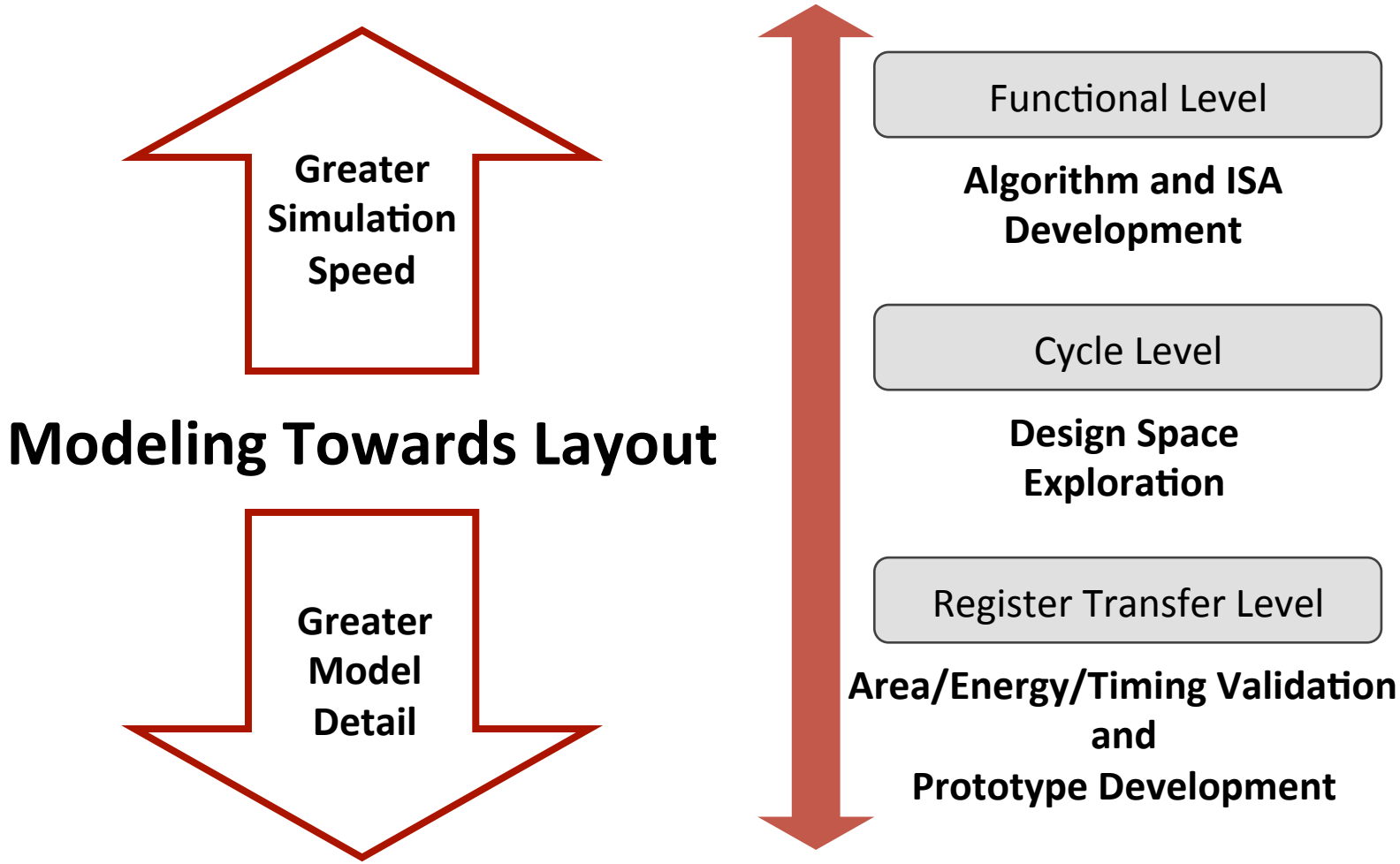
Cycle Level

- Behavior
- Timing

Computer Architecture Research Methodologies



Computer Architecture Research Methodologies

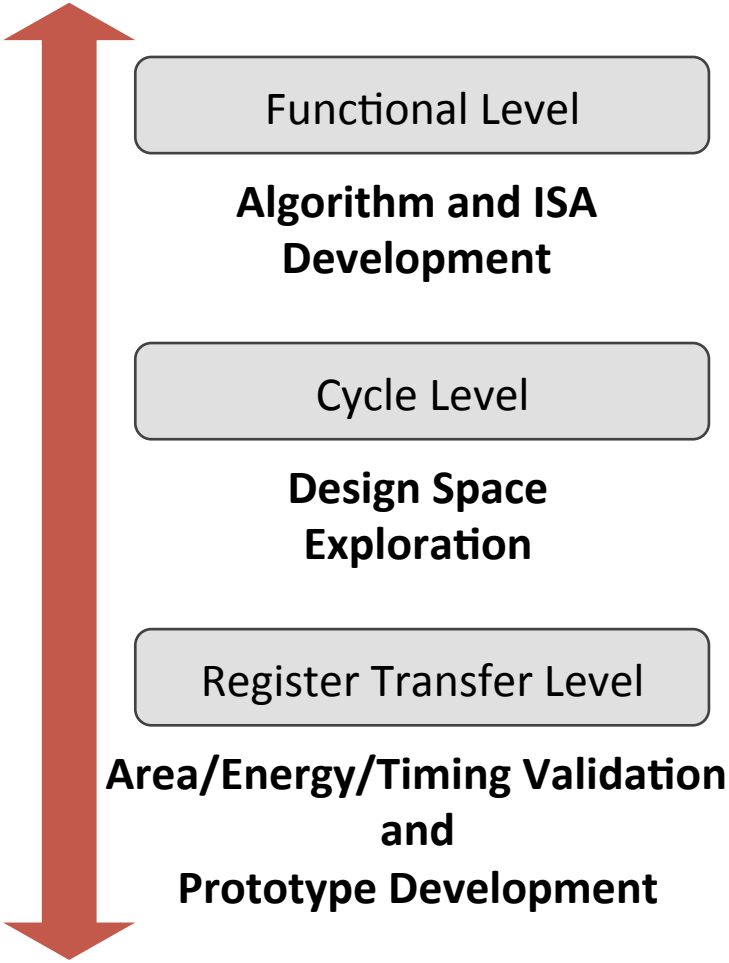


Computer Architecture Research Toolflows

MATLAB/Python Algorithm or
C++ Instruction Set Simulator

C++ Computer Architecture
Simulation Framework
(Object-Oriented)

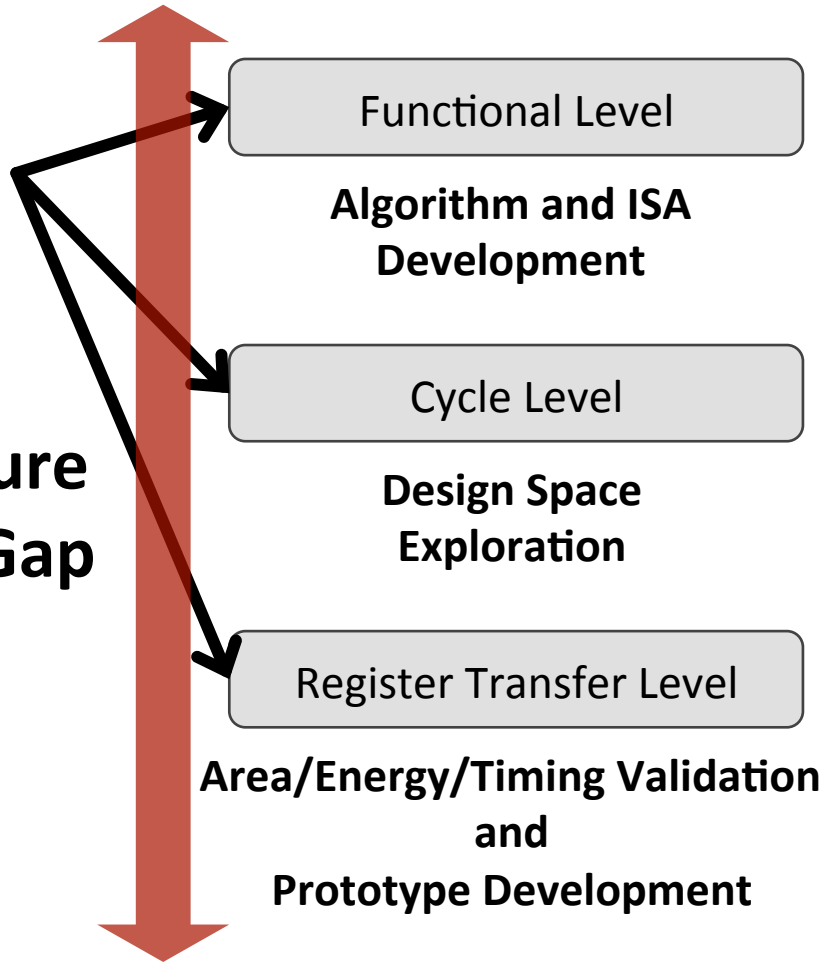
Verilog or VHDL Design with
EDA Toolflow
(Concurrent-Structural)



Computer Architecture Research Toolflows

Different languages, patterns, and tools!

The Computer Architecture Research Methodology Gap

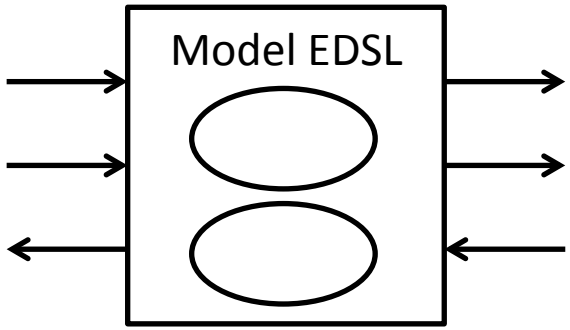


Great Ideas From Prior Frameworks

• **Concurrent-Structural Modeling** (Liberty, Cascade, SystemC)	Consistent interfaces across abstractions
• **Unified Modeling Languages** (SystemC)	Unified design environment for FL, CL, RTL
• **Hardware Generation Languages** (Chisel, Genesis2, BlueSpec, MyHDL)	Productive RTL design space exploration
• **HDL-Integrated Simulation Frameworks** (Cascade)	Productive RTL validation and cosimulation
• **Latency-Insensitive Interfaces** (Liberty, BlueSpec)	Component and test bench reuse

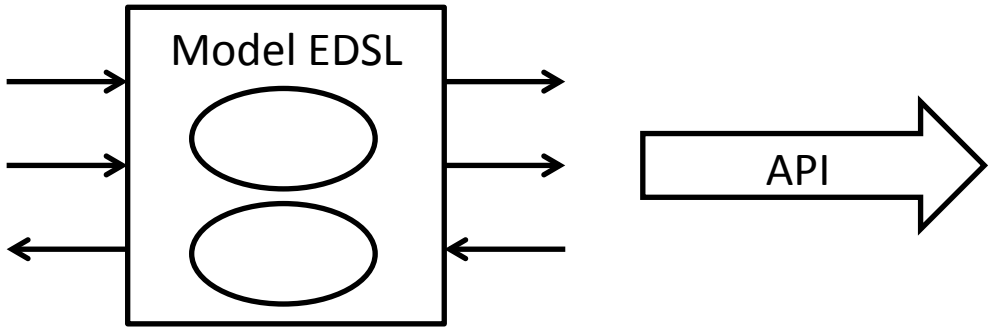
What is PyMTL?

- A Python EDSL for concurrent-structural hardware modeling



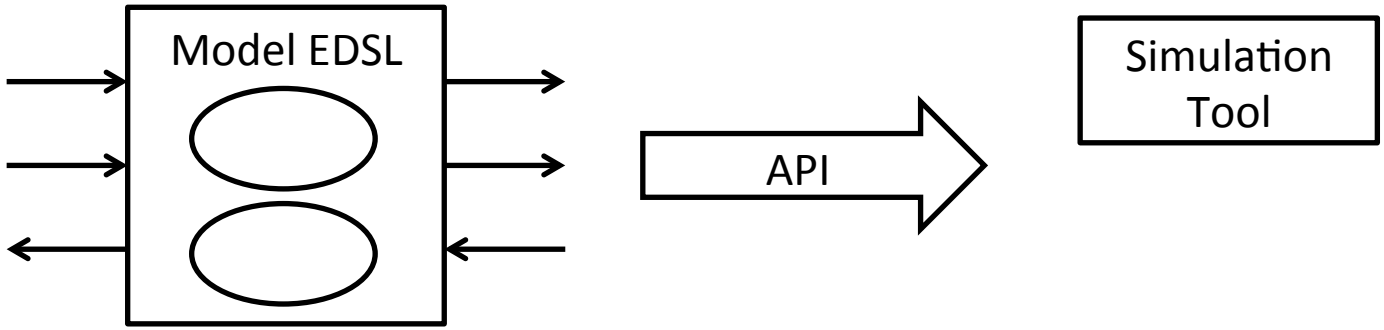
What is PyMTL?

- A Python EDSL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL EDSL



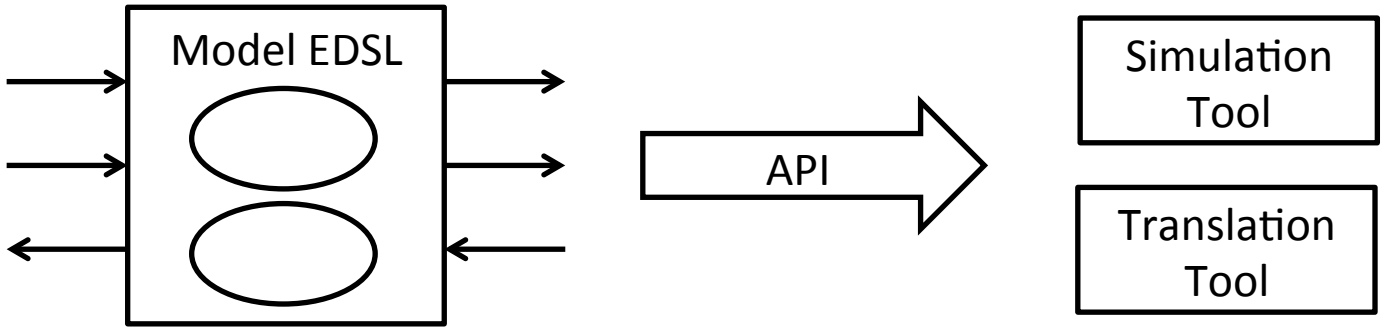
What is PyMTL?

- A Python EDSL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL EDSL
- A Python tool for simulating PyMTL FL, CL, and RTL models



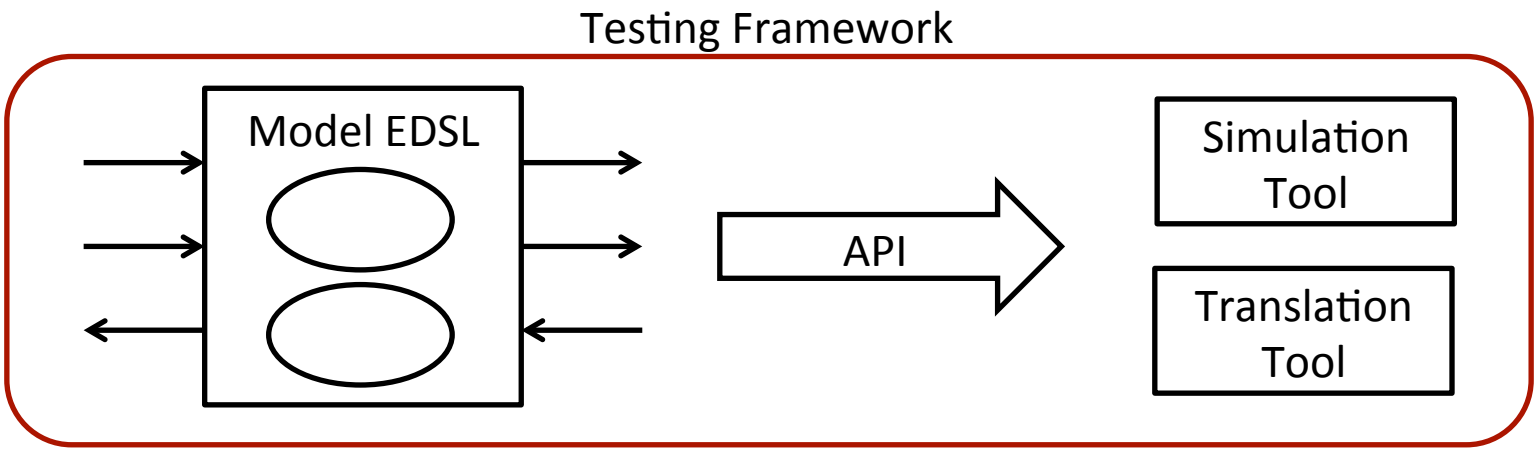
What is PyMTL?

- A Python EDSL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL EDSL
- A Python tool for simulating PyMTL FL, CL, and RTL models
- A Python tool for translating PyMTL RTL models into Verilog

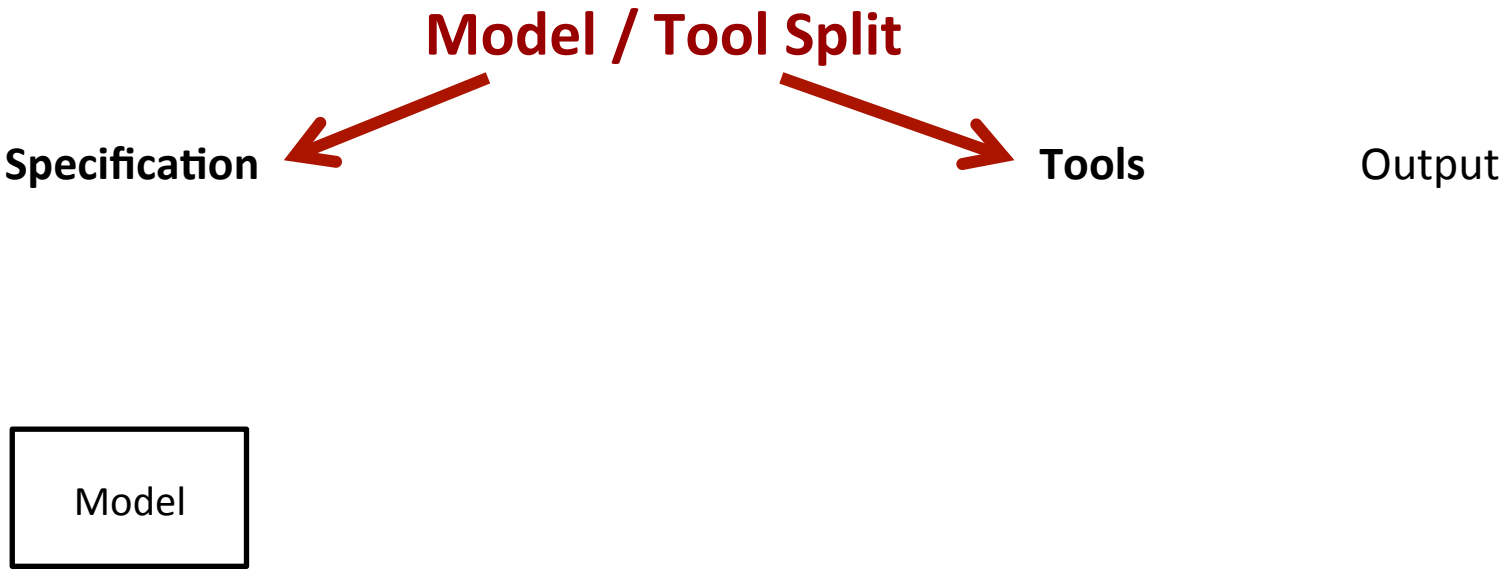


What is PyMTL?

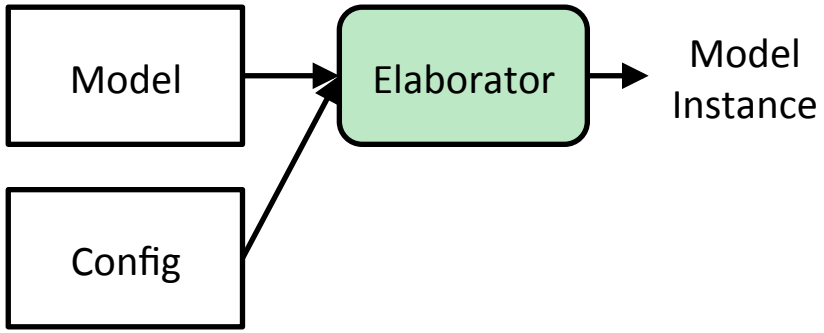
- A Python EDSL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL EDSL
- A Python tool for simulating PyMTL FL, CL, and RTL models
- A Python tool for translating PyMTL RTL models into Verilog
- A Python testing framework for model validation



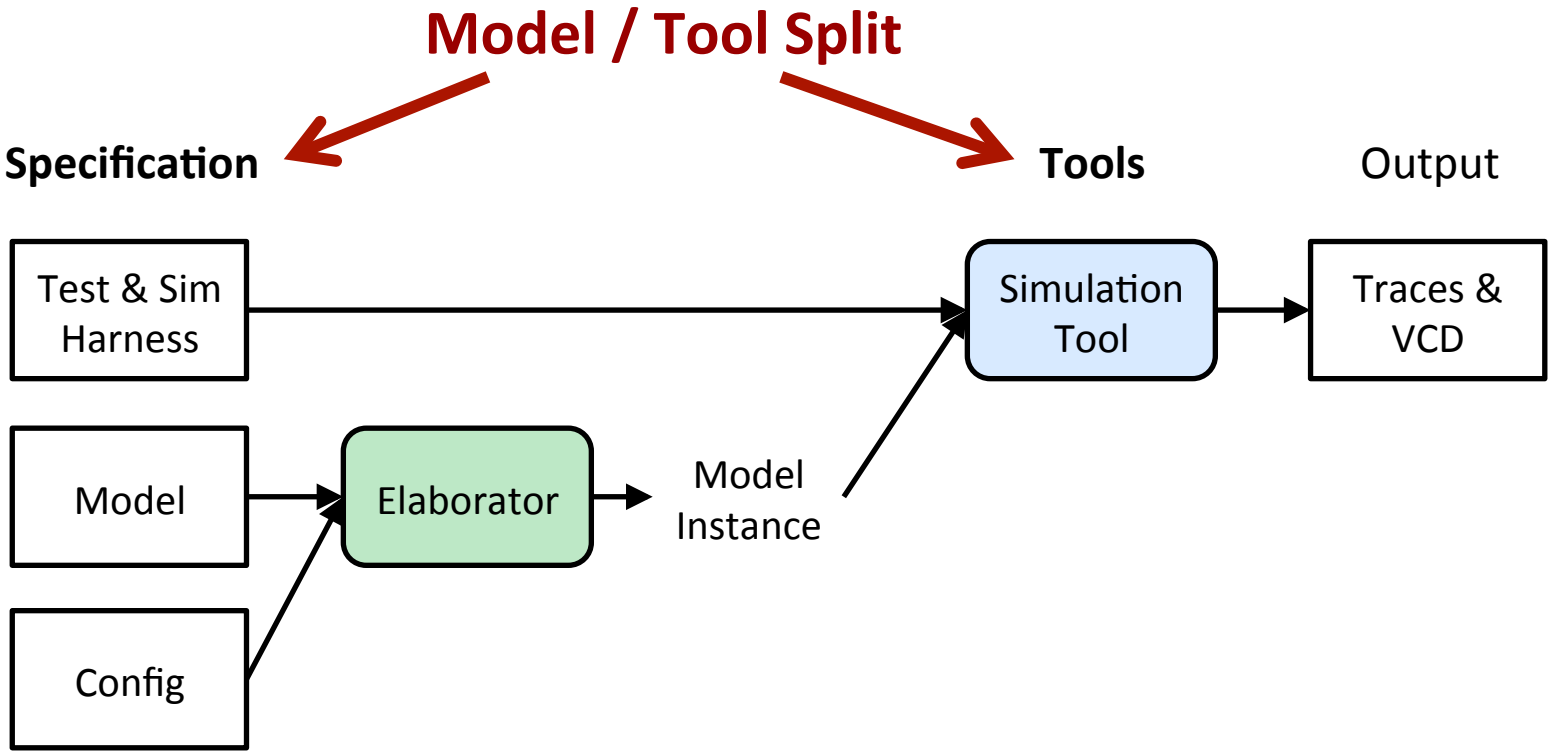
The PyMTL Framework



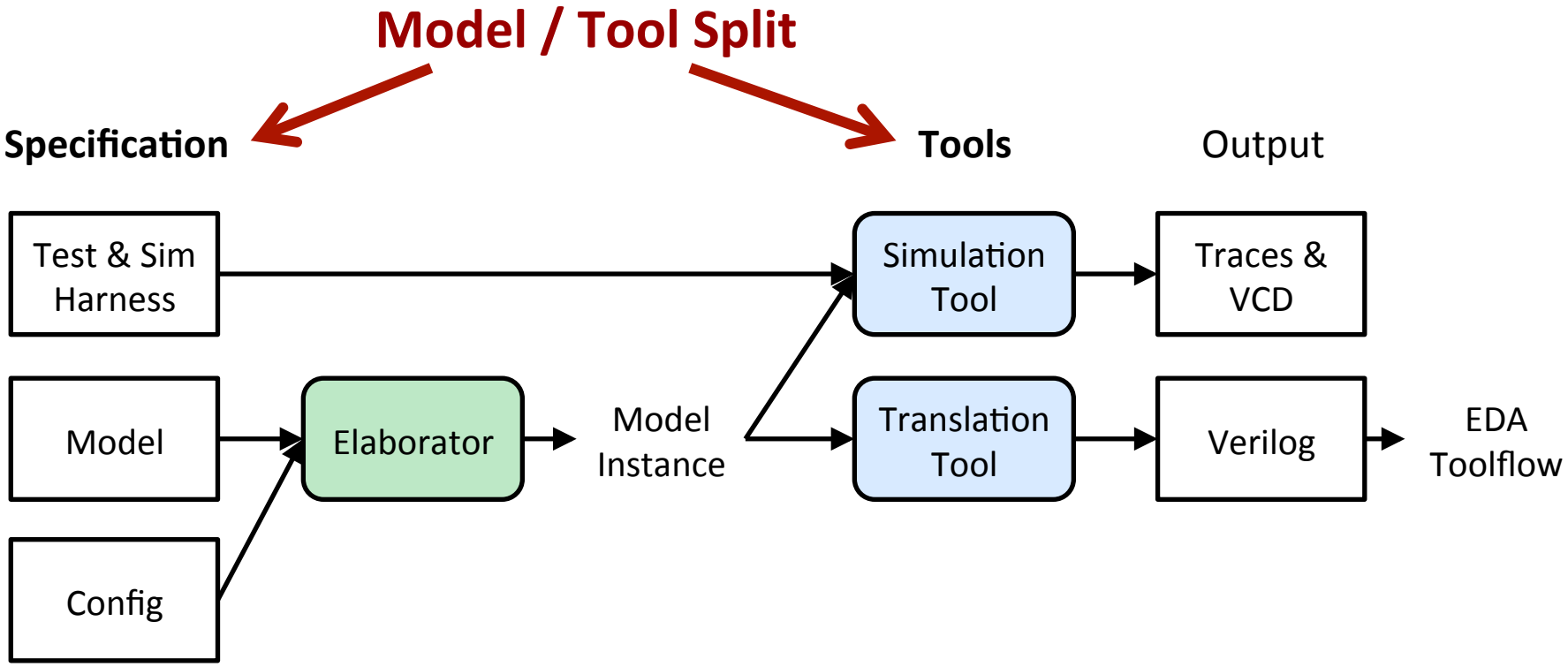
The PyMTL Framework



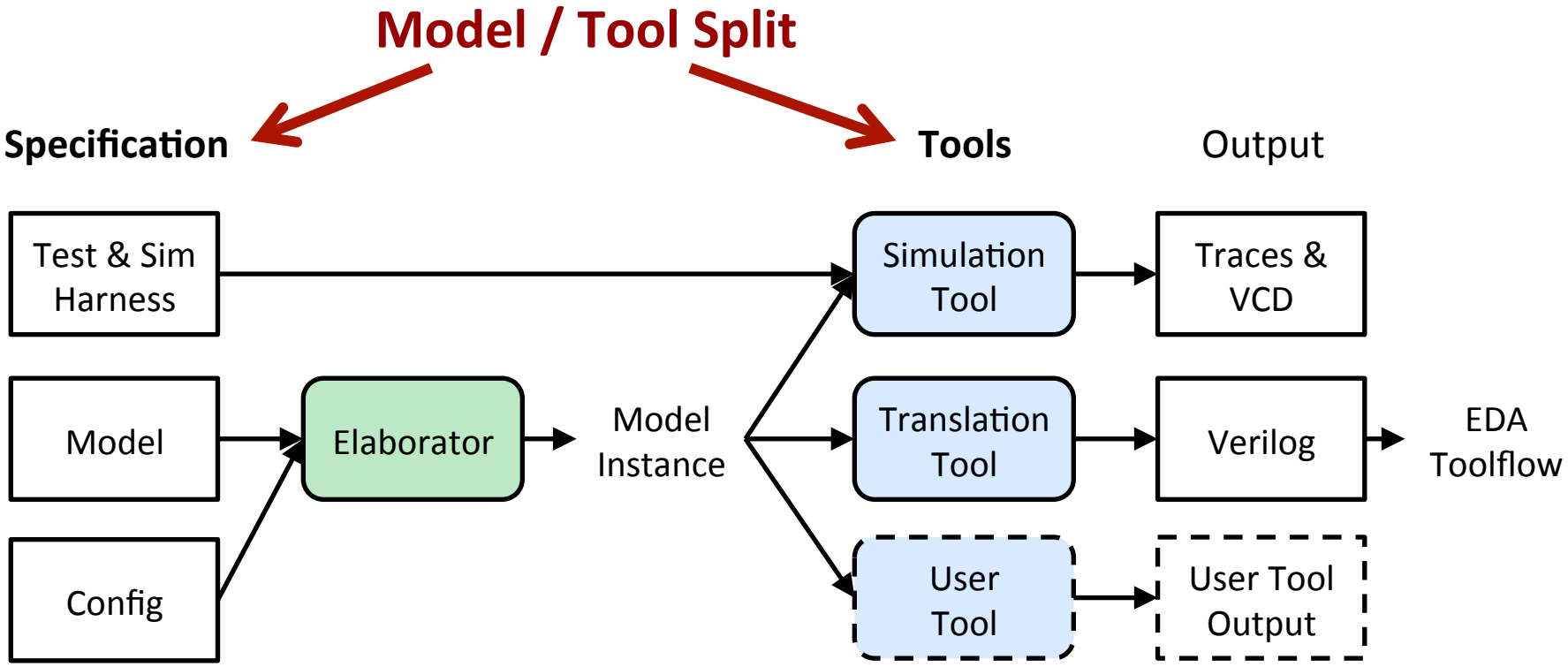
The PyMTL Framework



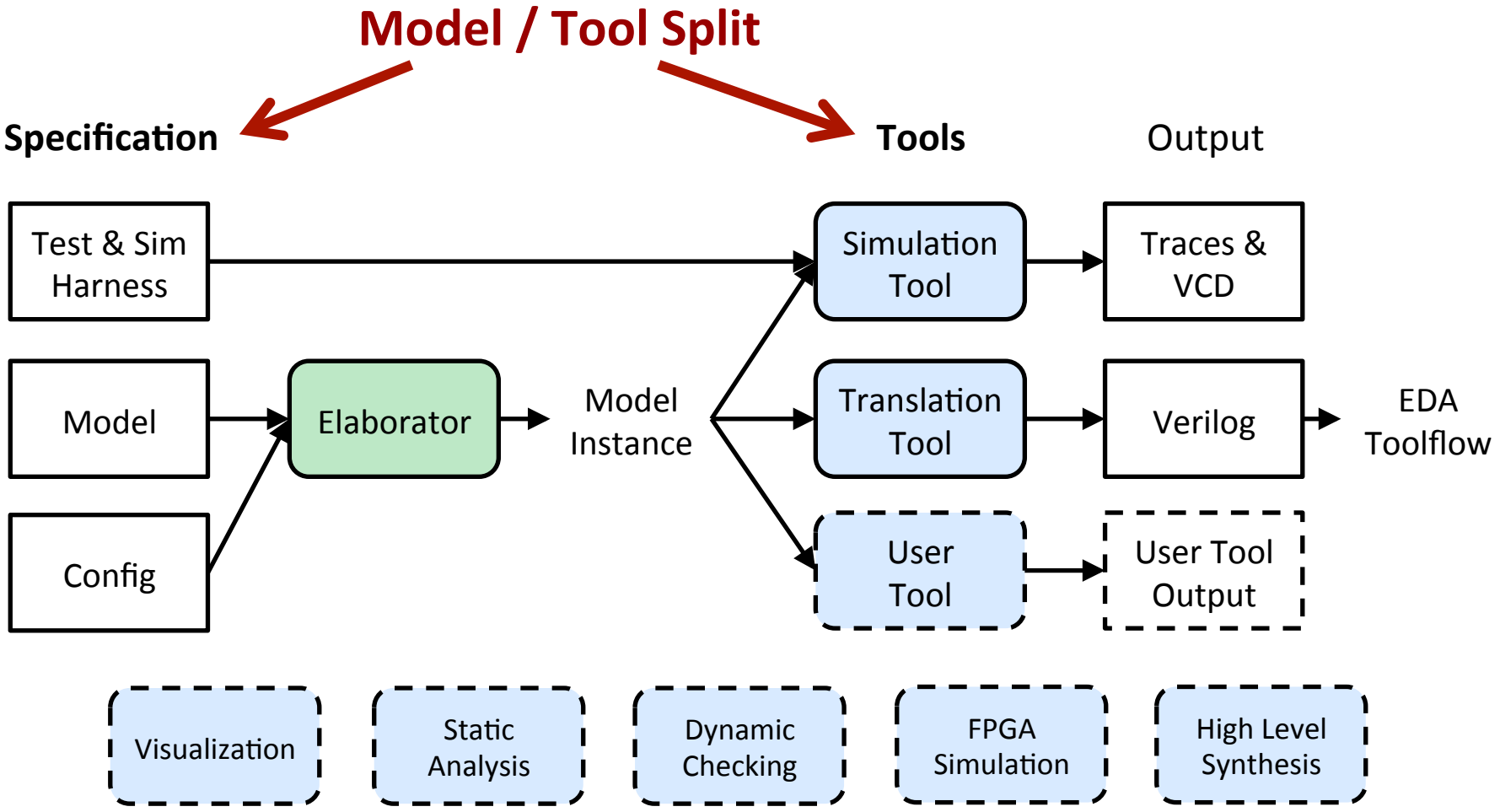
The PyMTL Framework



The PyMTL Framework



The PyMTL Framework



PyMTL101: Traditional FL Model in Python

```
def max_unit( input_list ):
  return max( input_list )
```

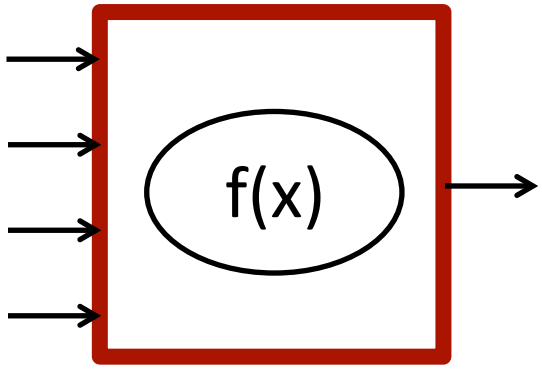
[3, 1, 2, 0] ---> f(x) ---> 3

PyMTL101: FL Model in PyMTL Embedded-DSL

```
def max_unit( input_list ):
  return max( input_list )

class MaxUnitFL( Model ):
```

[3, 1, 2, 0] ---> f(x) ---> 3

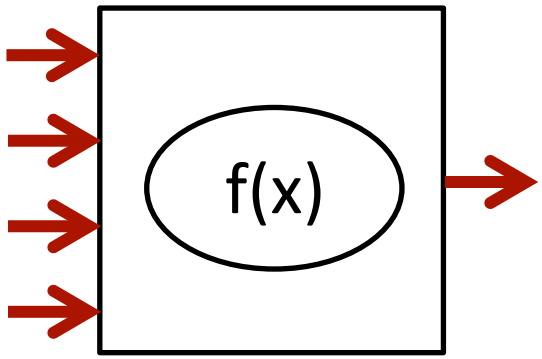


PyMTL101: FL Model in PyMTL Embedded-DSL

```
def max_unit( input_list ):
  return max( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> 3

```
class MaxUnitFL( Model ):
  def __init__( s, nbits, nports ):
    dtype  = Bits( nbits )
    s.in_  = InPort[nports]( dtype )
    s.out  = OutPort( dtype )
```



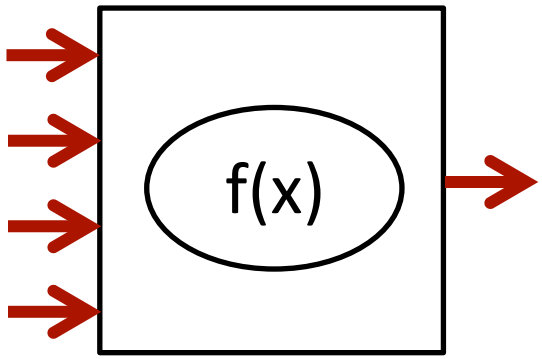
PyMTL101: FL Model in PyMTL Embedded-DSL

```
def max_unit( input_list ):
  return max( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> 3

```
class MaxUnitFL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort[nports]( nbits )
    s.out  = OutPort( nbits )
```



PyMTL101: FL Model in PyMTL Embedded-DSL

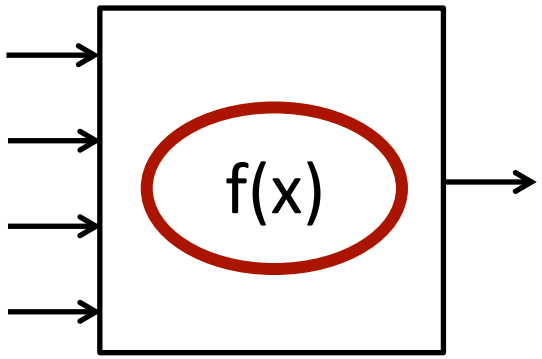
```
def max_unit( input_list ):
  return max( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> 3

```
class MaxUnitFL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort[nports]( nbits )
    s.out  = OutPort( nbits )


    @s.tick_fl
    def logic():
```



PyMTL101: FL Model in PyMTL Embedded-DSL

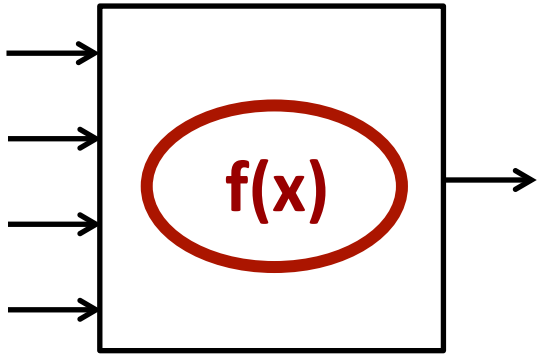
```
def max_unit( input_list ):
  return max( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> 3

```
class MaxUnitFL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort[nports]( nbits )
    s.out  = OutPort( nbits )


    @s.tick_fl
    def logic():
      s.out.next = max( s.in_ )
```



PyMTL101: FL Model in PyMTL Embedded-DSL

```
def sorter_network( input_list ):
  return sorted( input_list )
```

[3, 1, 2, 0] ---> f(x) ---> [0, 1, 2, 3]

```
class MaxUnitFL( Model ):
  def __init__( s, nbits, nports ):

    s.in_  = InPort[nports]( nbits )
    s.out  = OutPort( nbits )


    @s.tick_fl
    def logic():
      s.out.next = max( s.in_ )
```