Axilog: Language Support for Approximate Hardware Design

Amir Yazdanbakhsh Divya Mahajan Bradley Thwaites
Jongse Park Anandhavel Nagendrakumar Sindhuja Sethuraman
Kartik Ramkrishnan Nishanthi Ravindran Rudra Jariwala
Abbas Rahimi Hadi Esmaeilzadeh Kia Bazargan

Georgia Institute of Technology University of Minnesota UC San Diego

Georgia Institute of Technology
Alternative Computing Technologies (ACT) Lab

DATE 2015

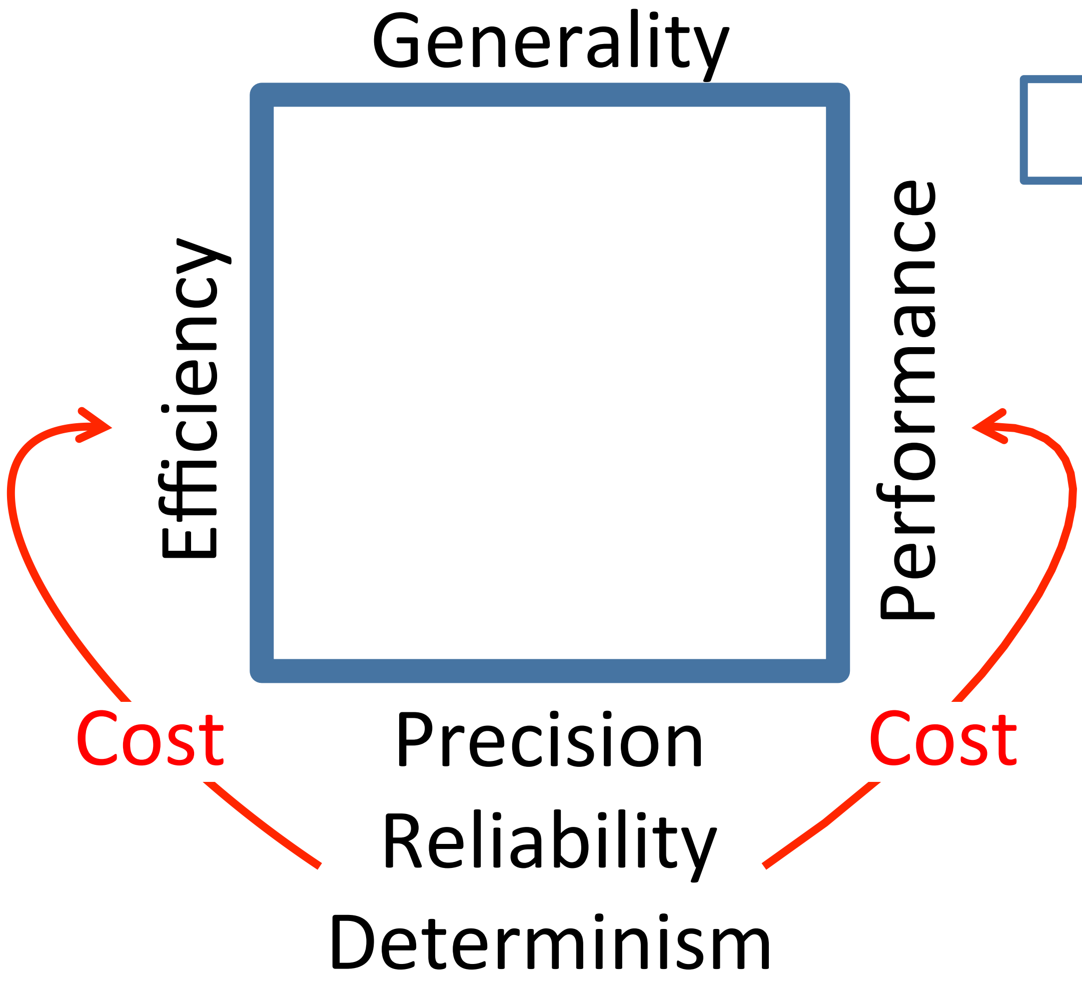
Approximate computing

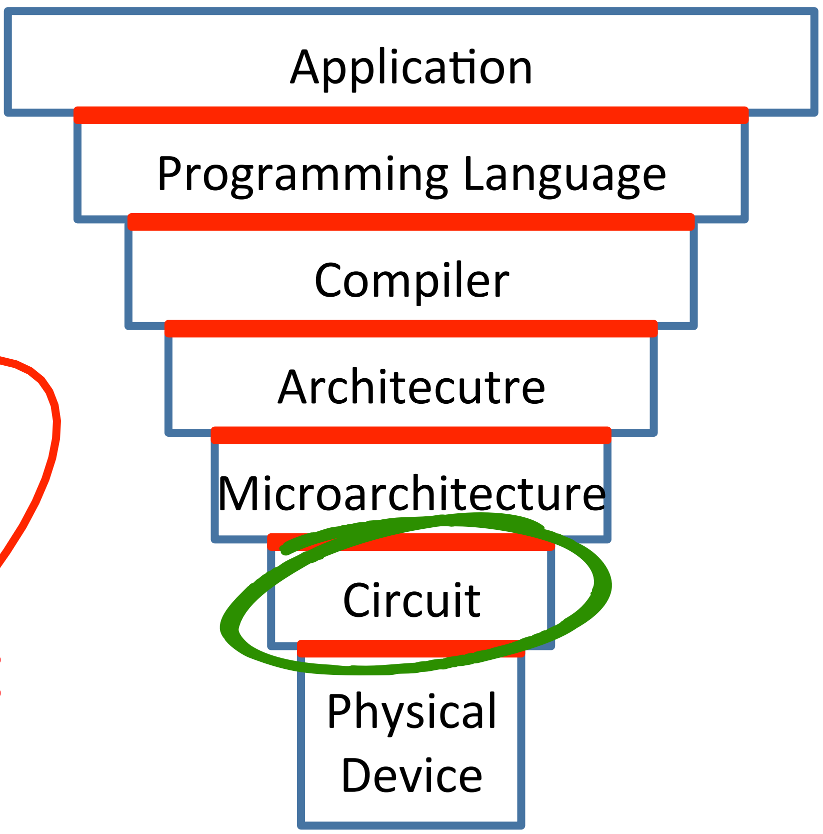
Embracing error

- Relax the abstraction of ***near-perfect*** accuracy in ***general-purpose*** computing/communication/storage
- Allow errors to happen during computation/ communication/storage
  - Improve resource utilization efficiency
    - Energy, bandwidth, capacity, …
  - Improve performance
- Build ***acceptable*** systems from ***intentionally-made unreliable*** software and hardware components
- Avoid ***overkill*** and ***worst-case*** design

Avoiding Worst-Case Design

Approximate Computing





Goals

Design the **first** HDL for:

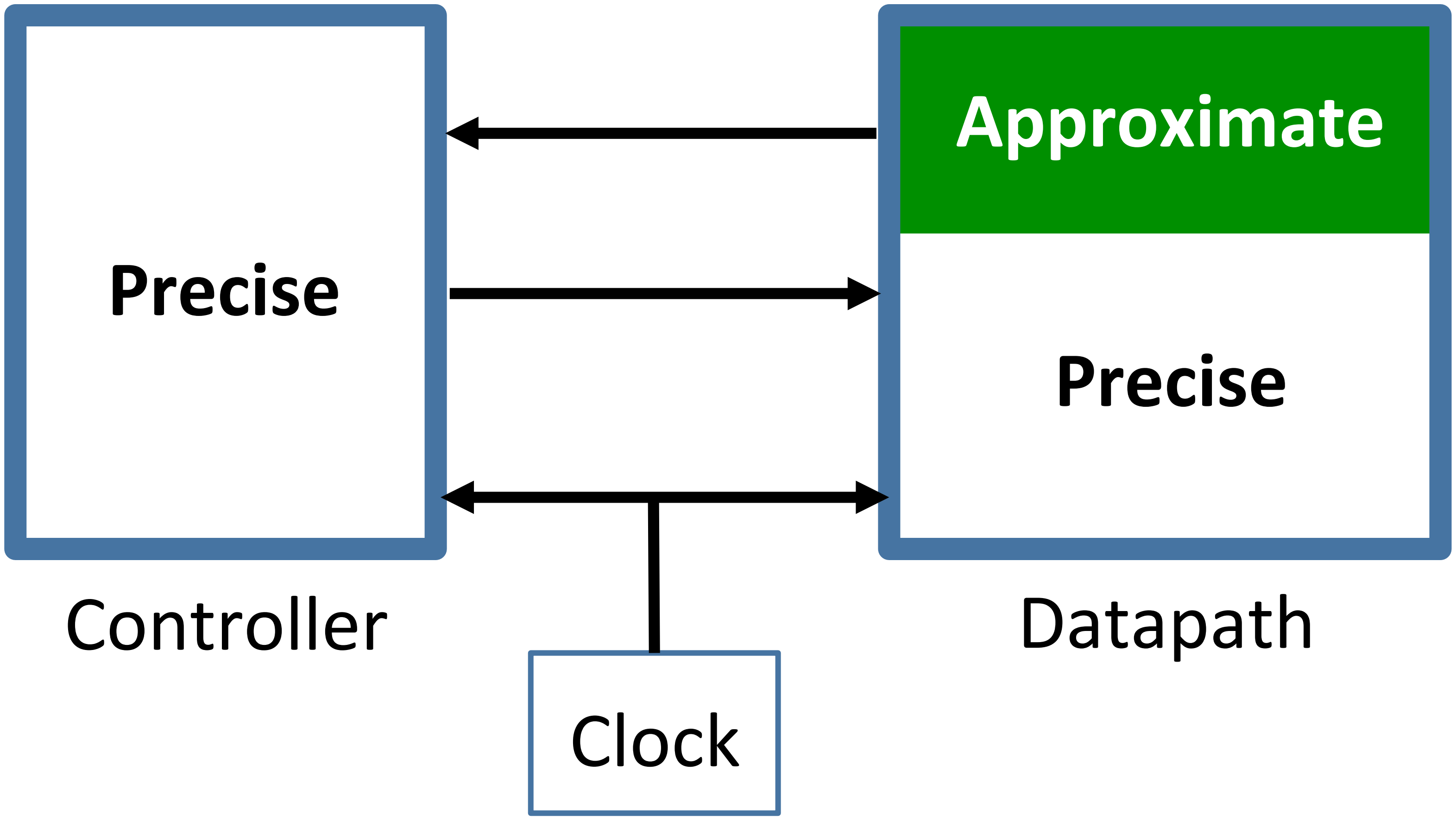
1) Approximate HW **Design**
2) Approximate HW **Reuse**
3) Approximate **Synthesis**

Criteria

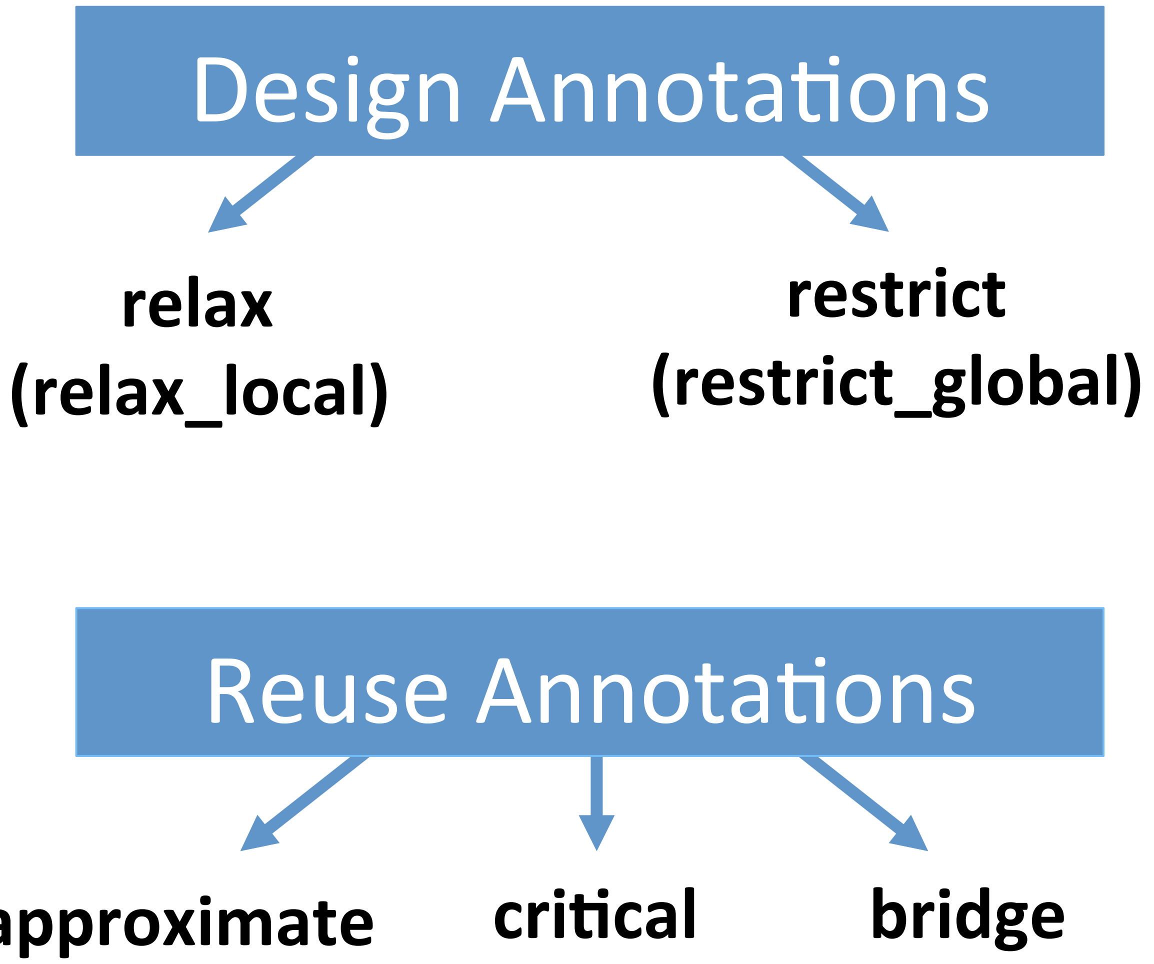
Approximate HDL:

1) **High-level**
2) **Automated**
3) **Backward compatible**
4) **Safety**

Safety in Hardware



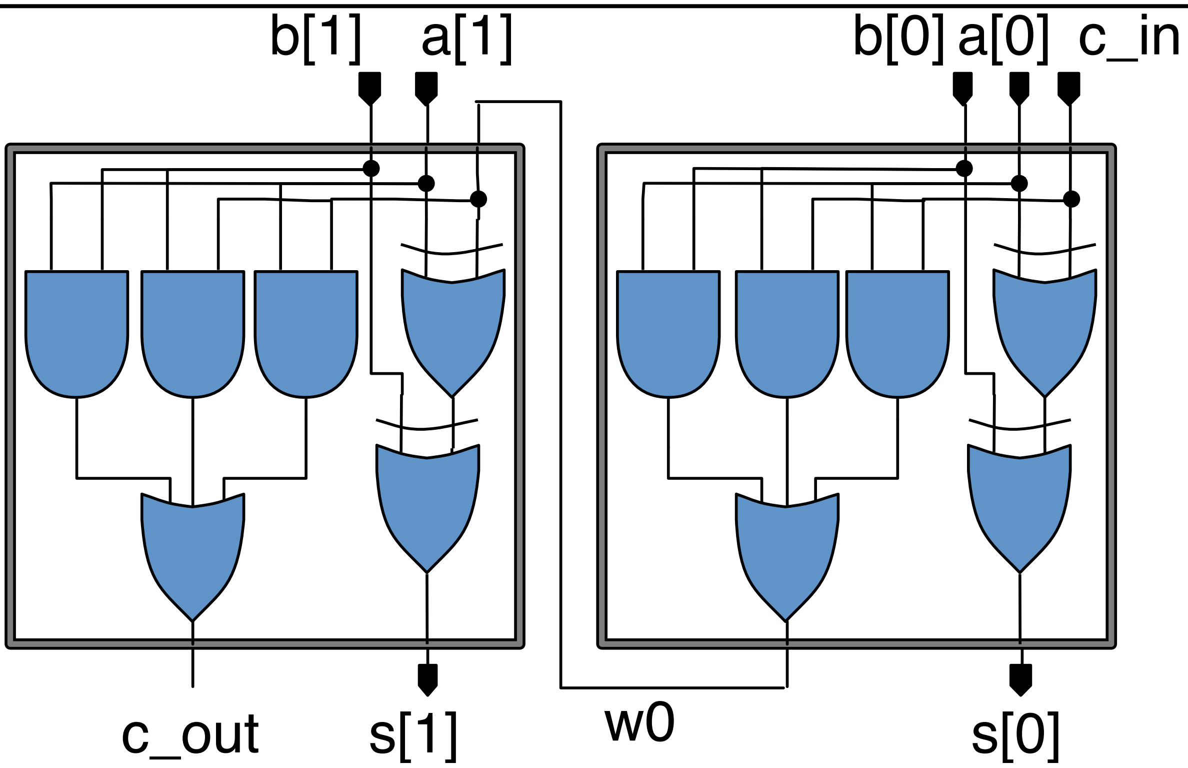
Axilog Annotations



Design Annotations

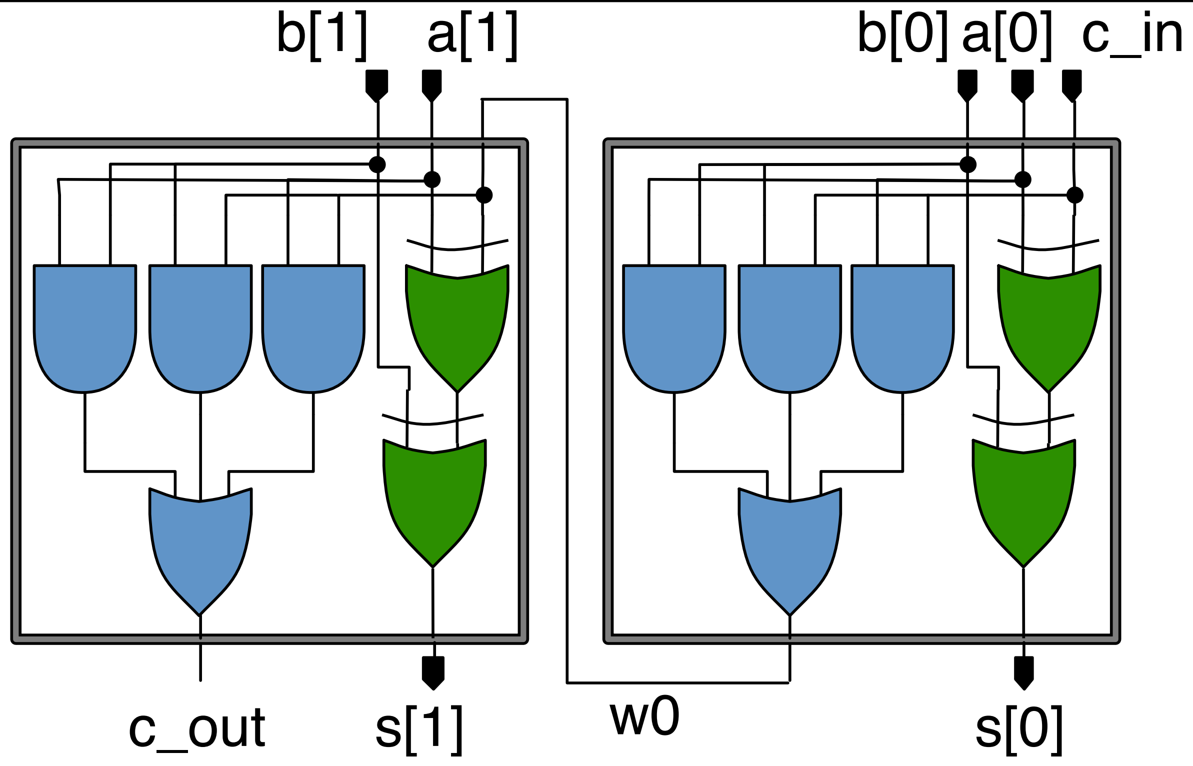
Relaxing Accuracy Requirements

```
module ripple_carry_adder(a, b, c_in, c_out, s)
…
    full_adder f0(a[0], b[0], c_in, w0, s[0])
    full_adder f1(a[1], b[1], w0, c_out, s[1])
    relax (s);
…
```



Relaxing Accuracy Requirements

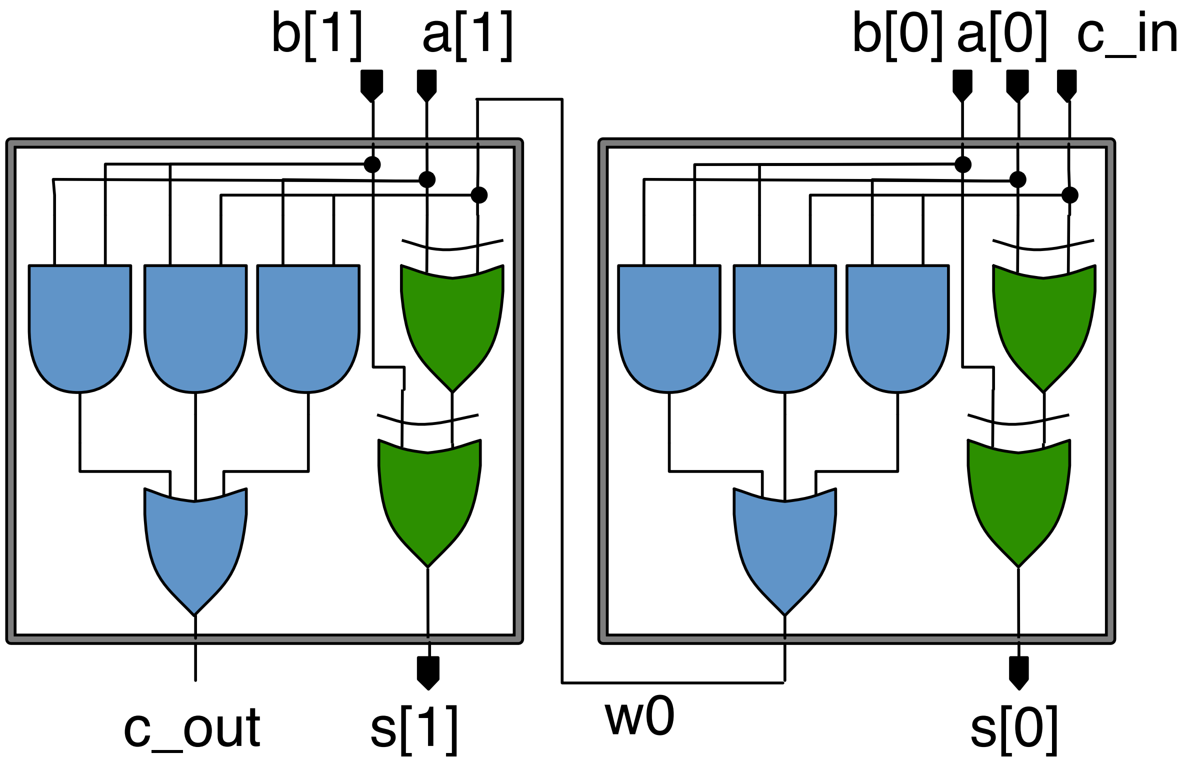
```
module ripple_carry_adder(a, b, c_in, c_out, s)
…
    full_adder f0(a[0], b[0], c_in, w0, s[0])
    full_adder f1(a[1], b[1], w0, c_out, s[1])
    relax (s);
…
```



Scoping Approximation (relax_local)

```
module full_adder
    (a, b, c_in, c_out, s)
…
→ relax_local (s);
…
```

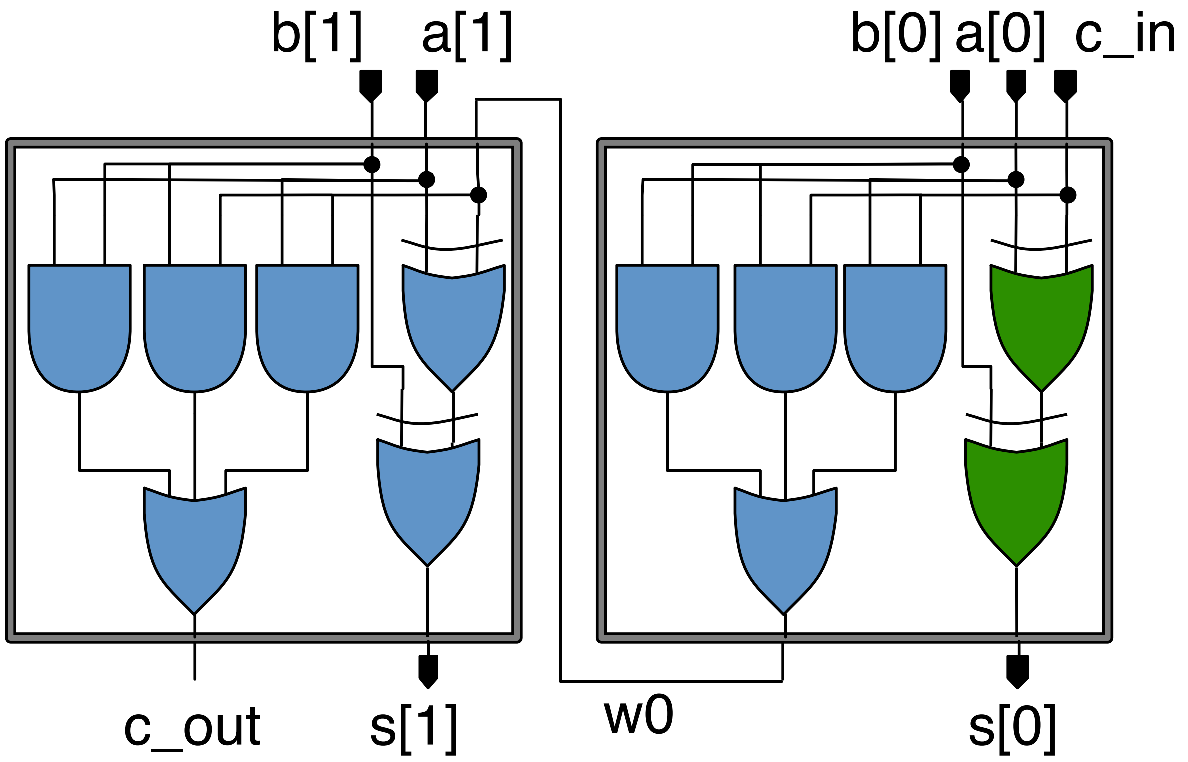
```
…
    full_adder f0 (…)
    full_adder f1(…)
    relax (s[0]);
…
```



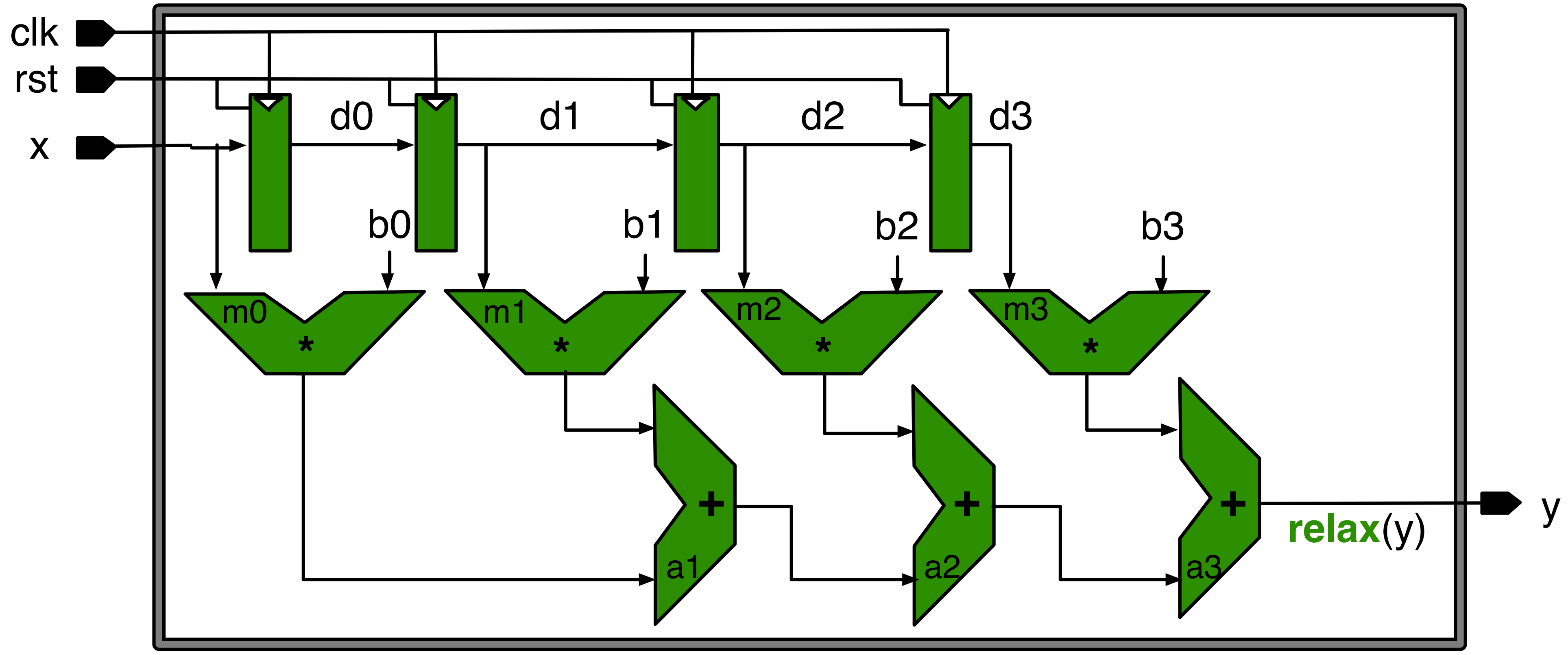
Scoping Approximation (relax_local)

```
module full_adder
    (a, b, c_in, c_out, s)
…
relax_local (s);
…
```

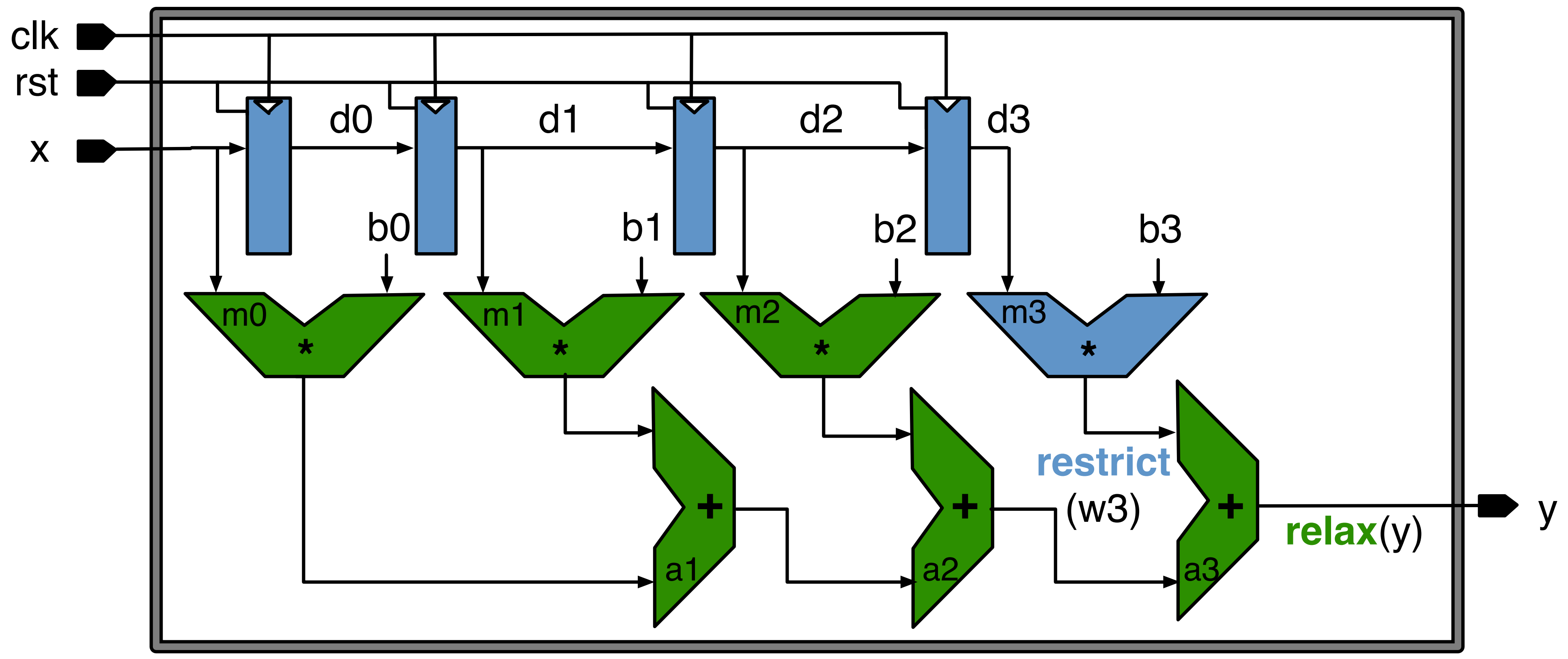
```
…
    full_adder f0 (…)
    full_adder f1(…)
    relax (s[0]);
…
```



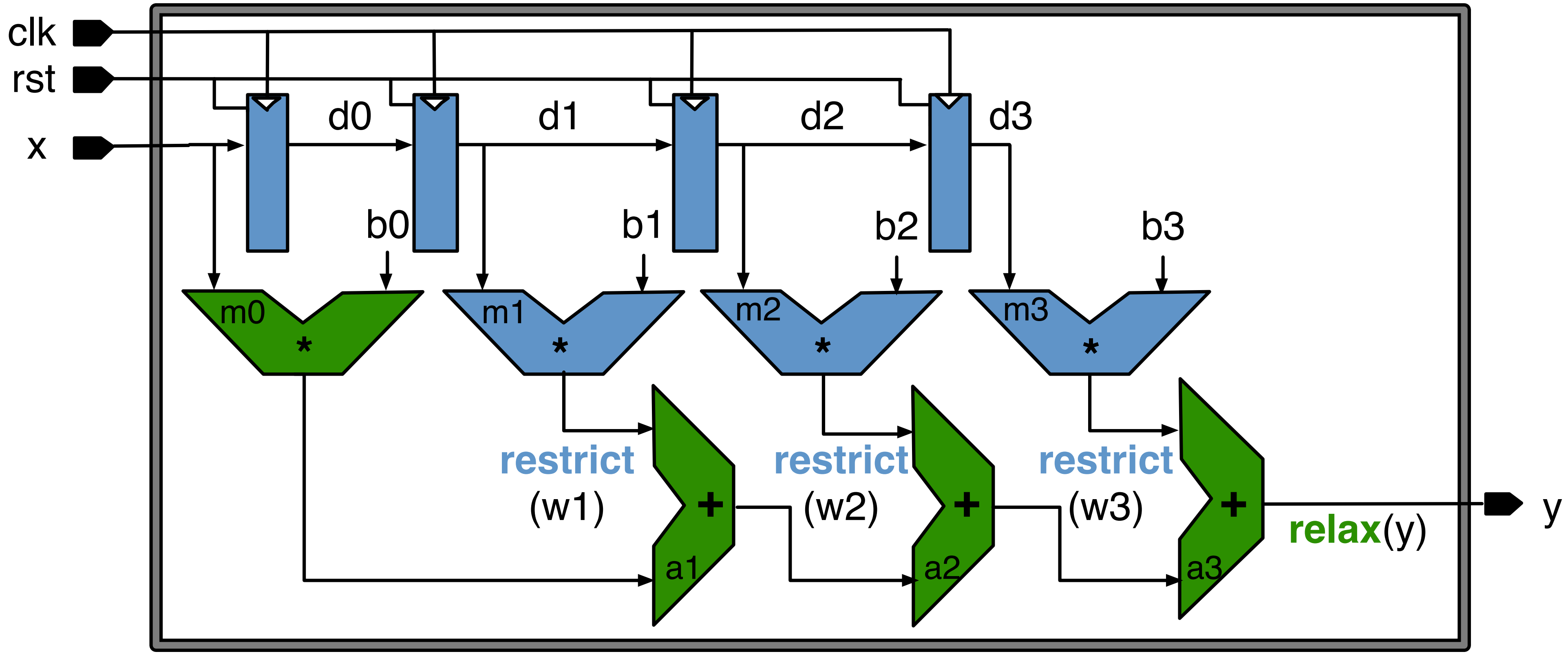
Restricting Approximation



Restricting Approximation

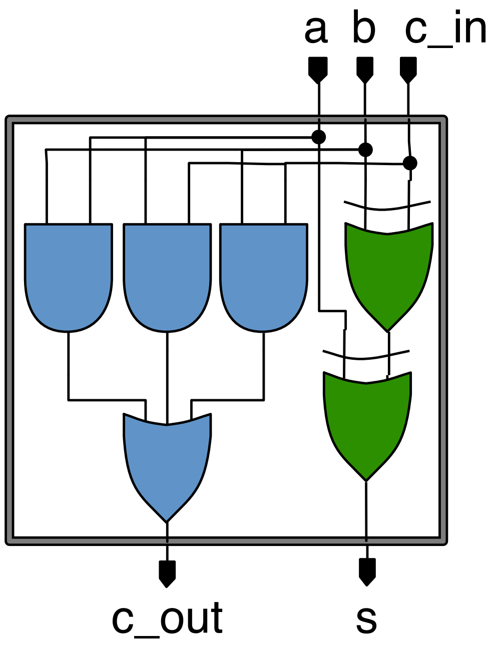


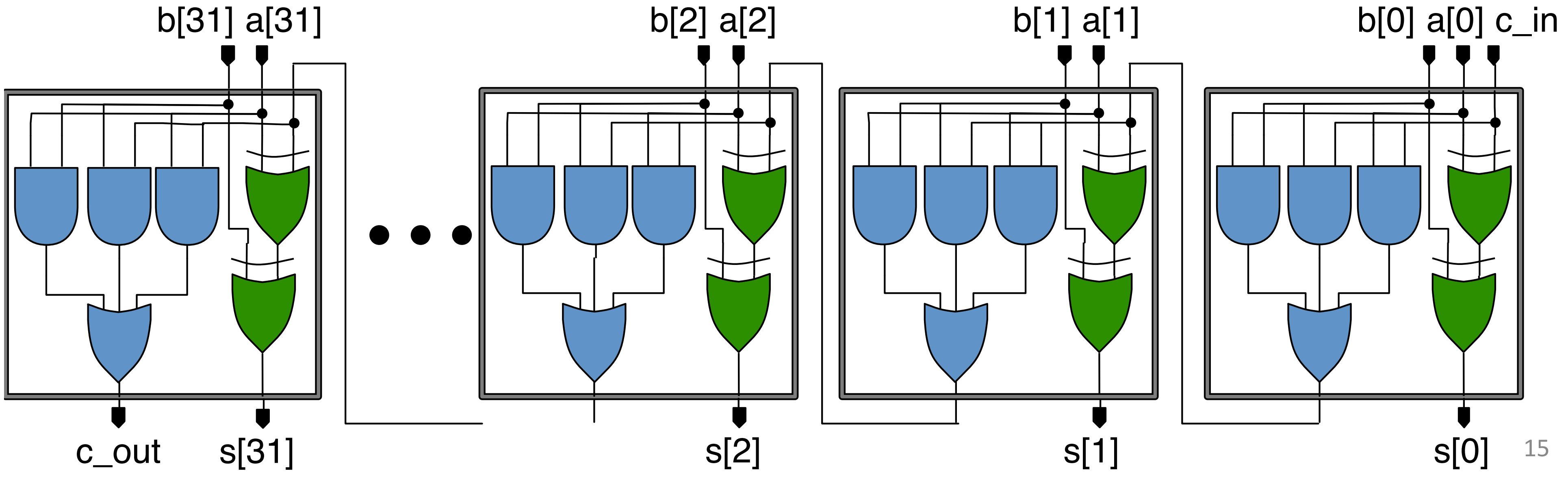
Restricting Approximation



Restricting Approximation Globally

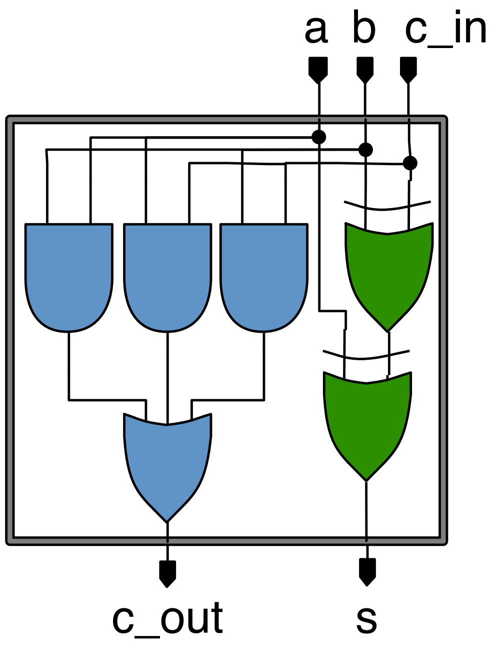
```
module full_adder(a, b, c_in, c_out, s);
    …
    approximate output s;
    relax (s);
    …
endmodule
    …
    restrict_global(s[31:0]);
```

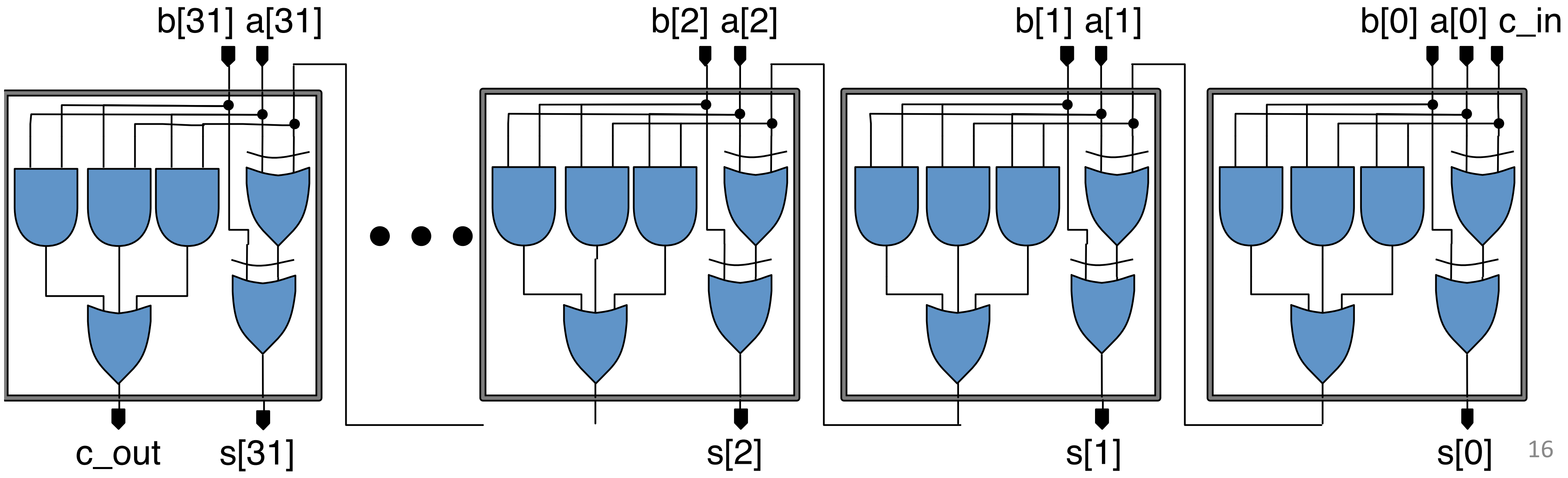




Restricting Approximation Globally

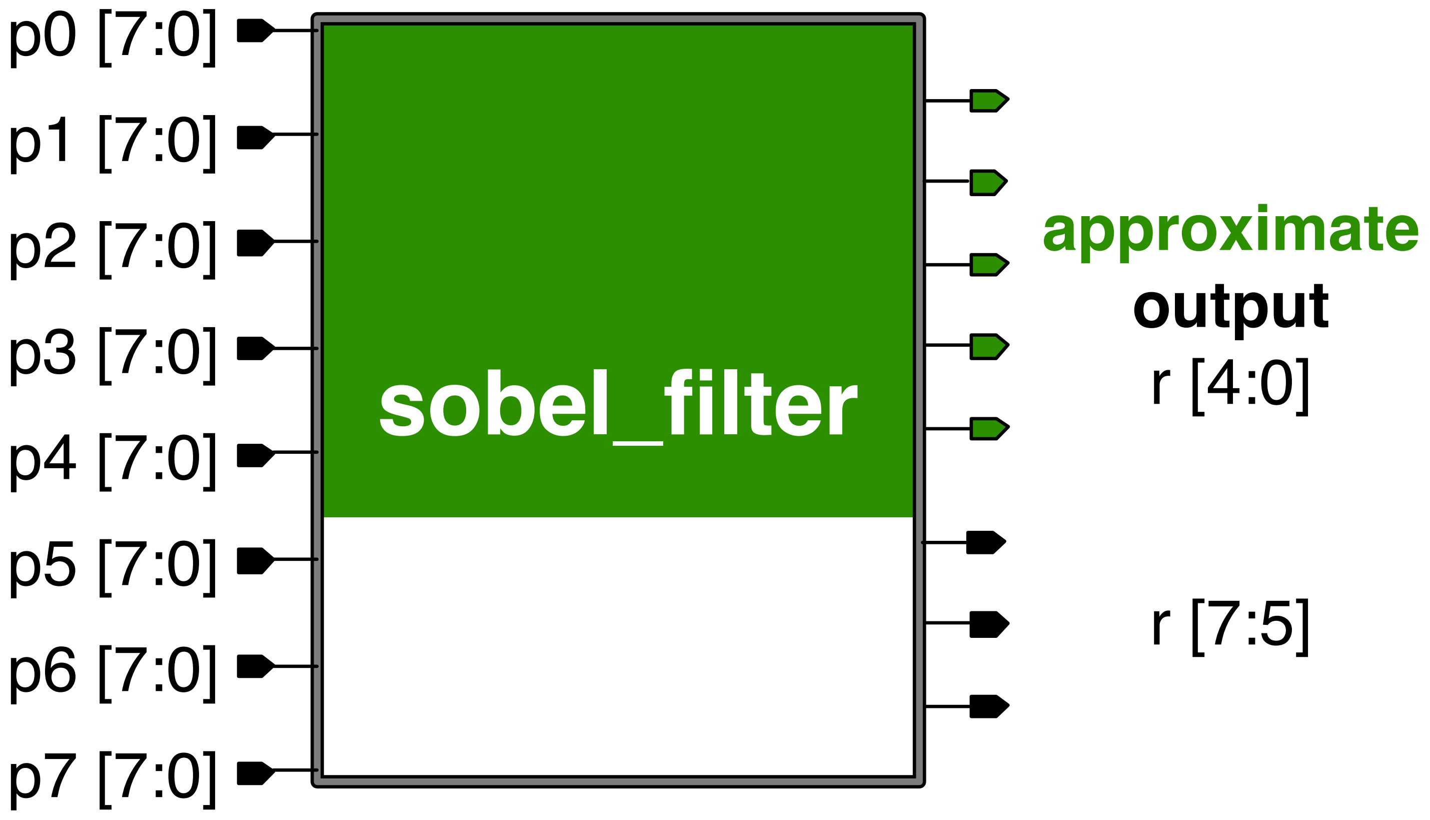
```
module full_adder(a, b, c_in, c_out, s);
    …
    approximate output s;
    relax (s);
    …
endmodule
    …
    restrict_global(s[31:0]);
```

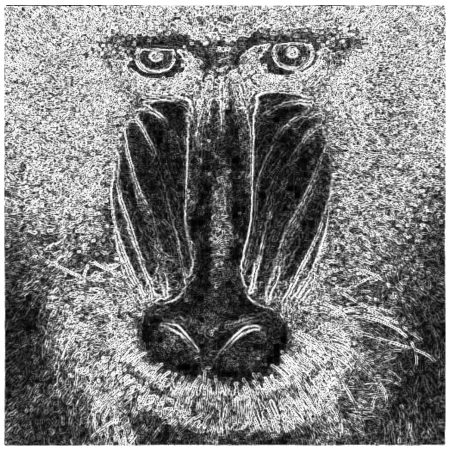




Reuse Annotations

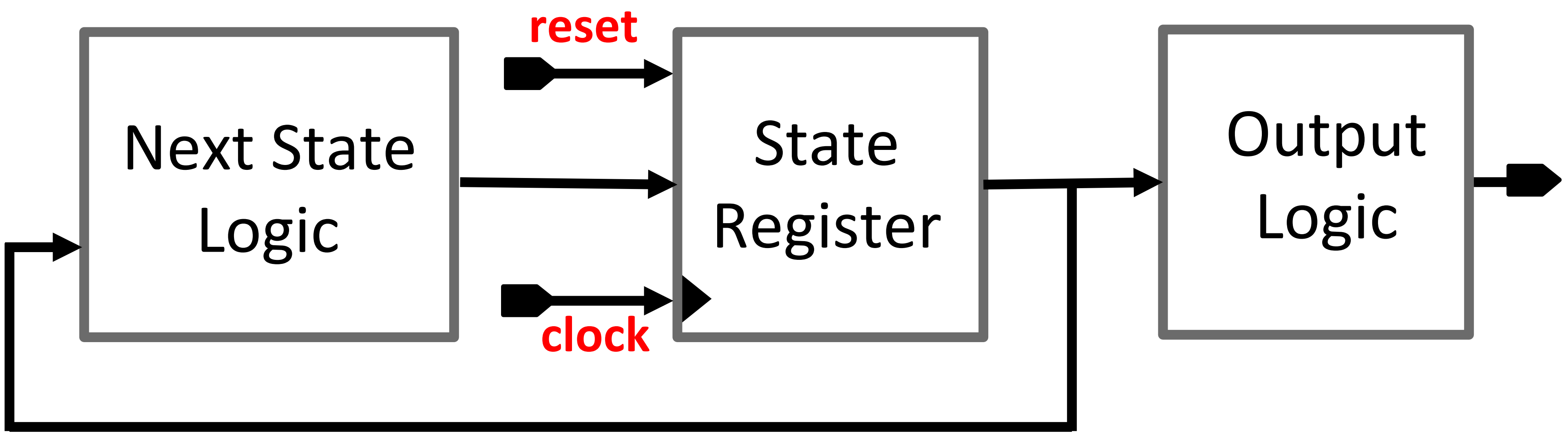
Outputs Carrying Approximate Semantics





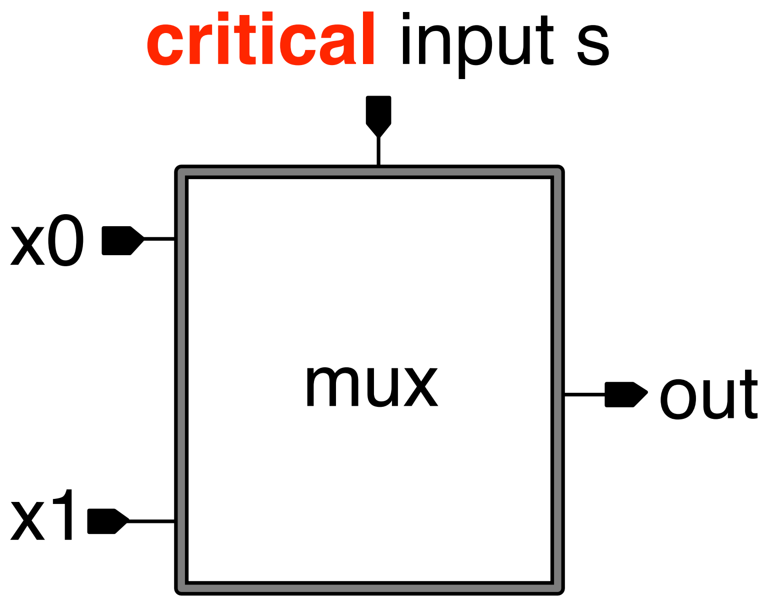
Critical Inputs

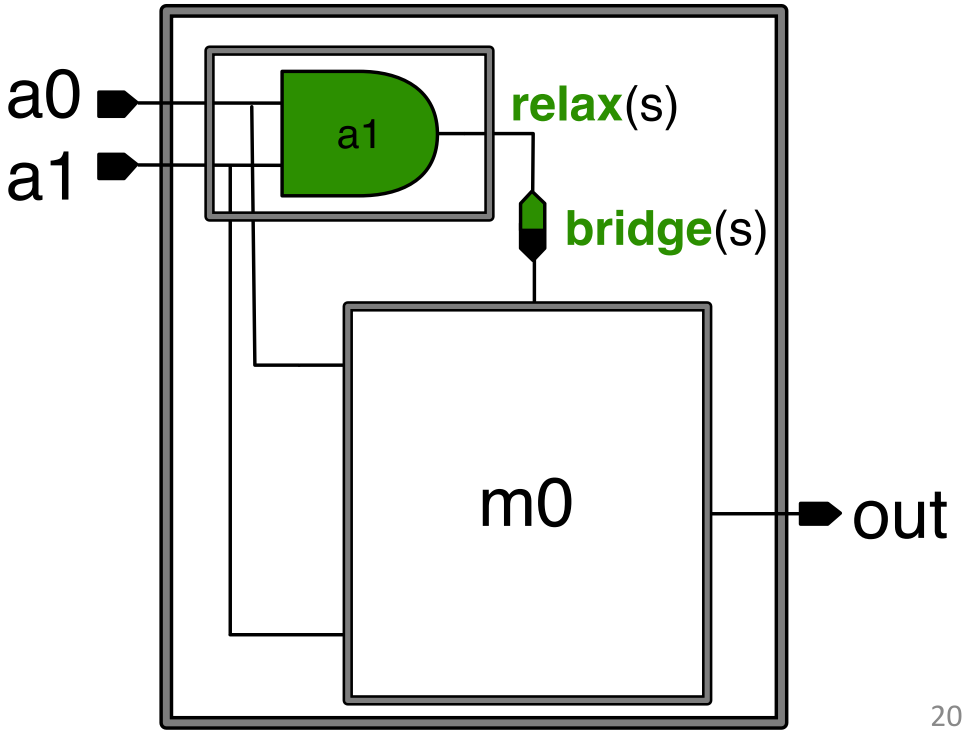
```
...
critical input reset;
critical input clock;
...
```



Bridging Approximate Wires to Critical Inputs

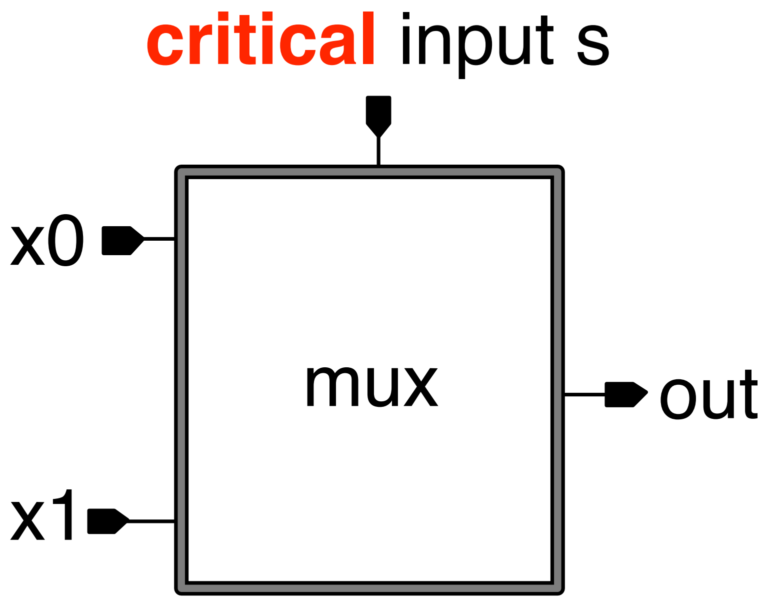
```
 …
and a1(s, a0, a1);
relax (s);
bridge (s);
multiplexer m0(s, a0, a1, out);
 …
```

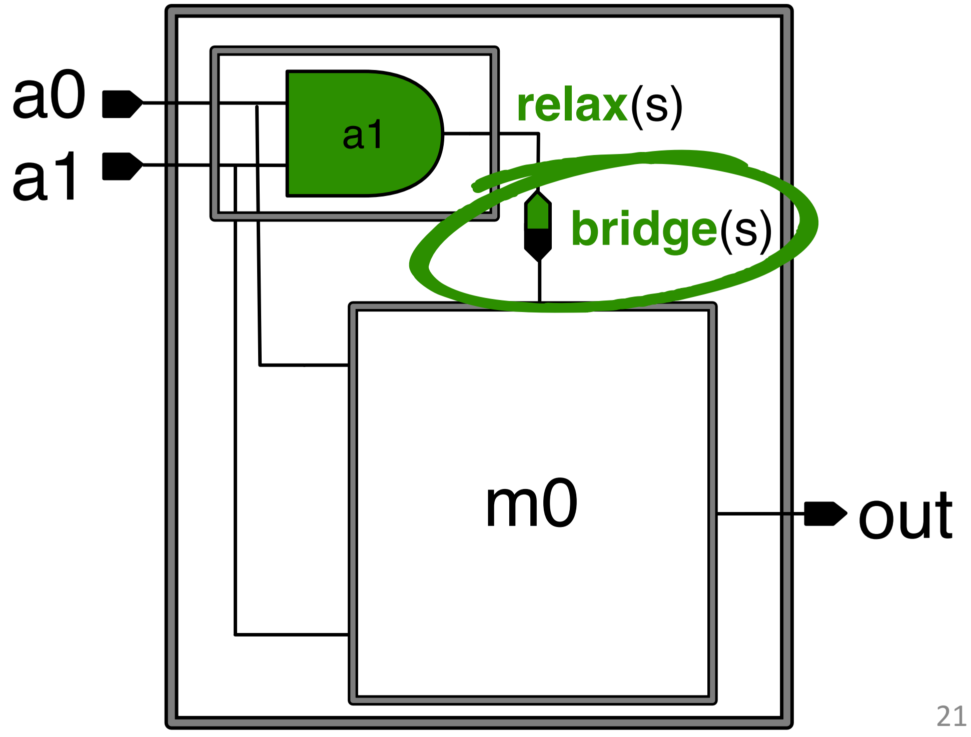




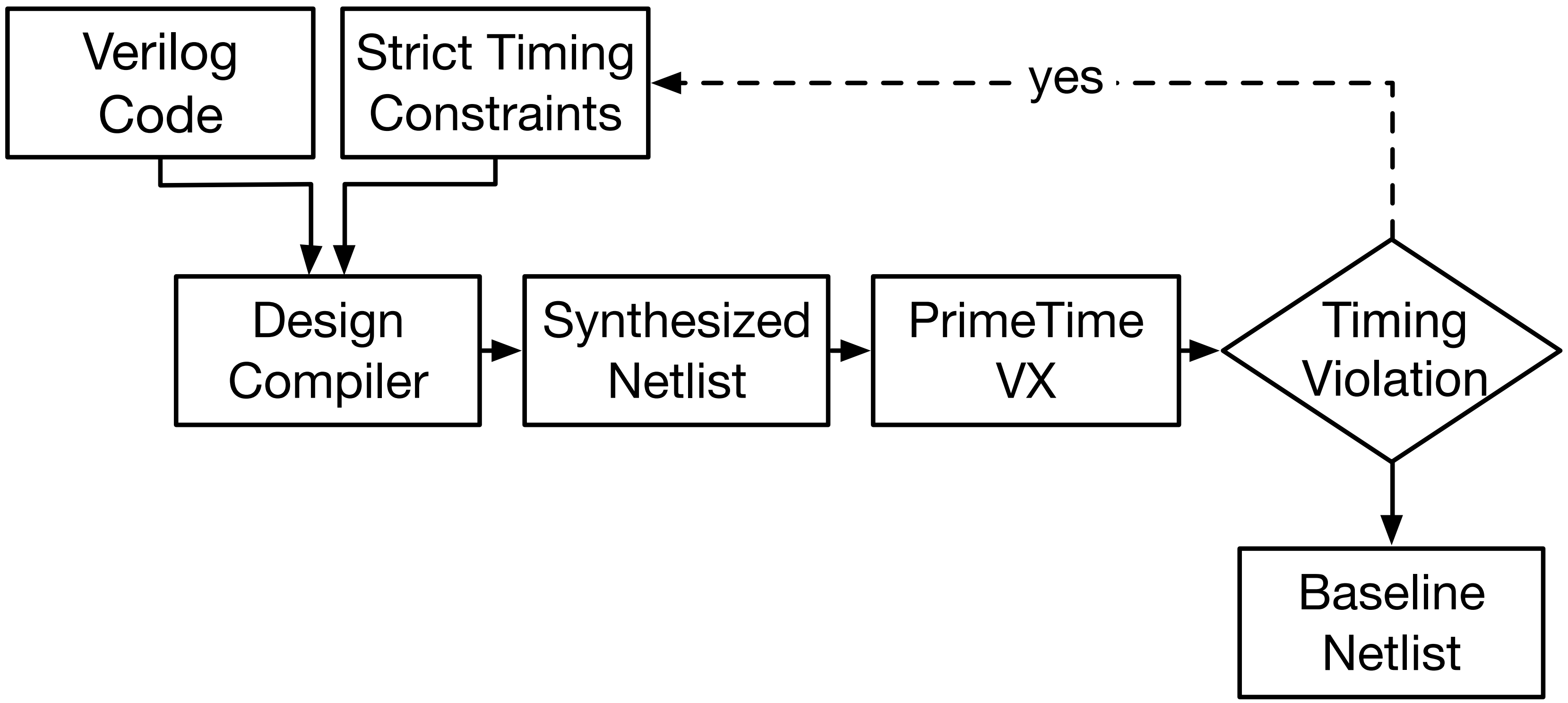
Bridging Approximate Wires to Critical Inputs

```
  ...
and a1(s, a0, a1);
relax (s);
bridge (s);
multiplexer m0(s, a0, a1, out);
  ...
```



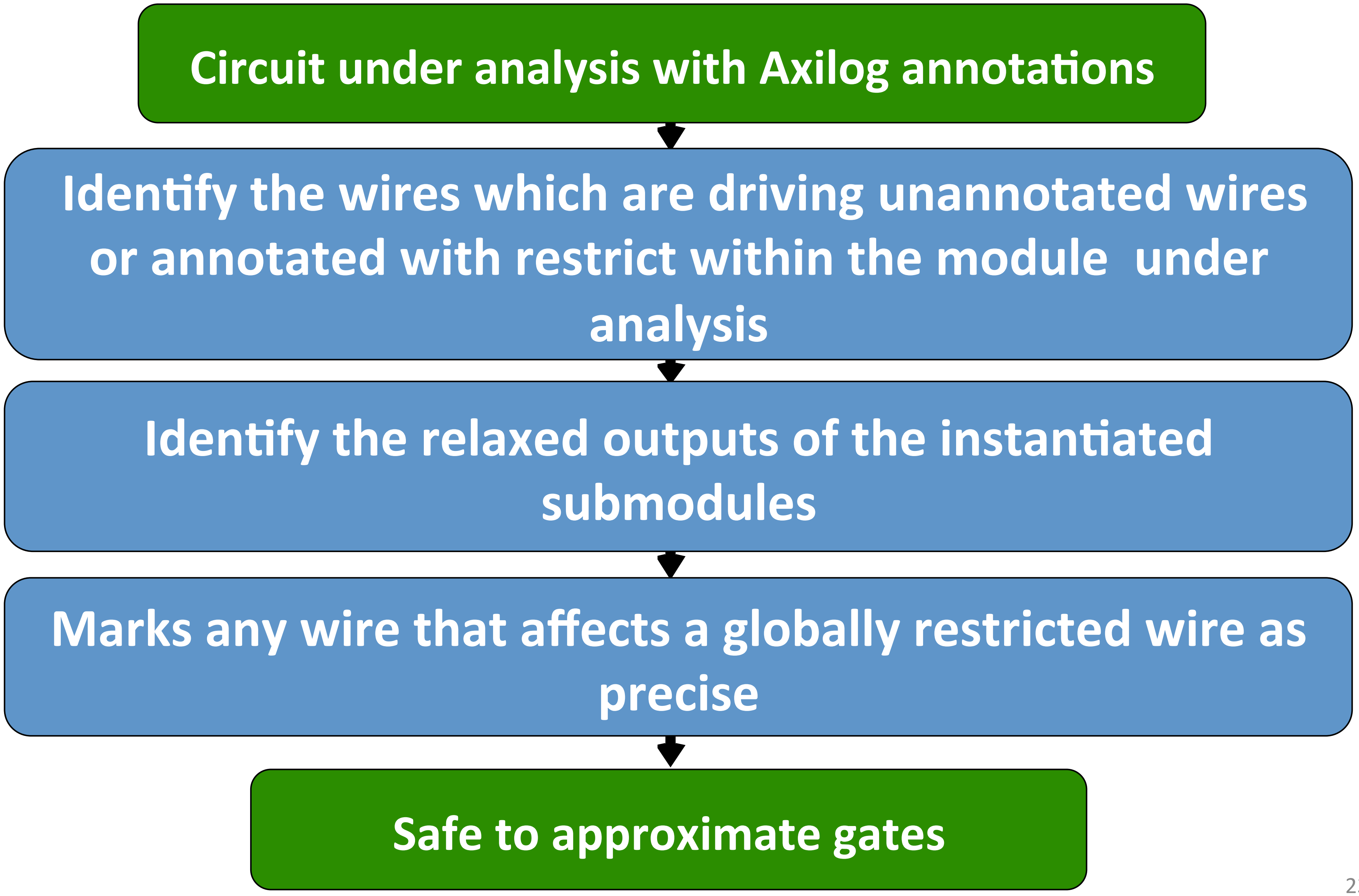


Baseline Synthesis Flow



Highest frequency with minimum power and area

Relaxability Inference Analysis



Circuit under analysis with Axilog annotations

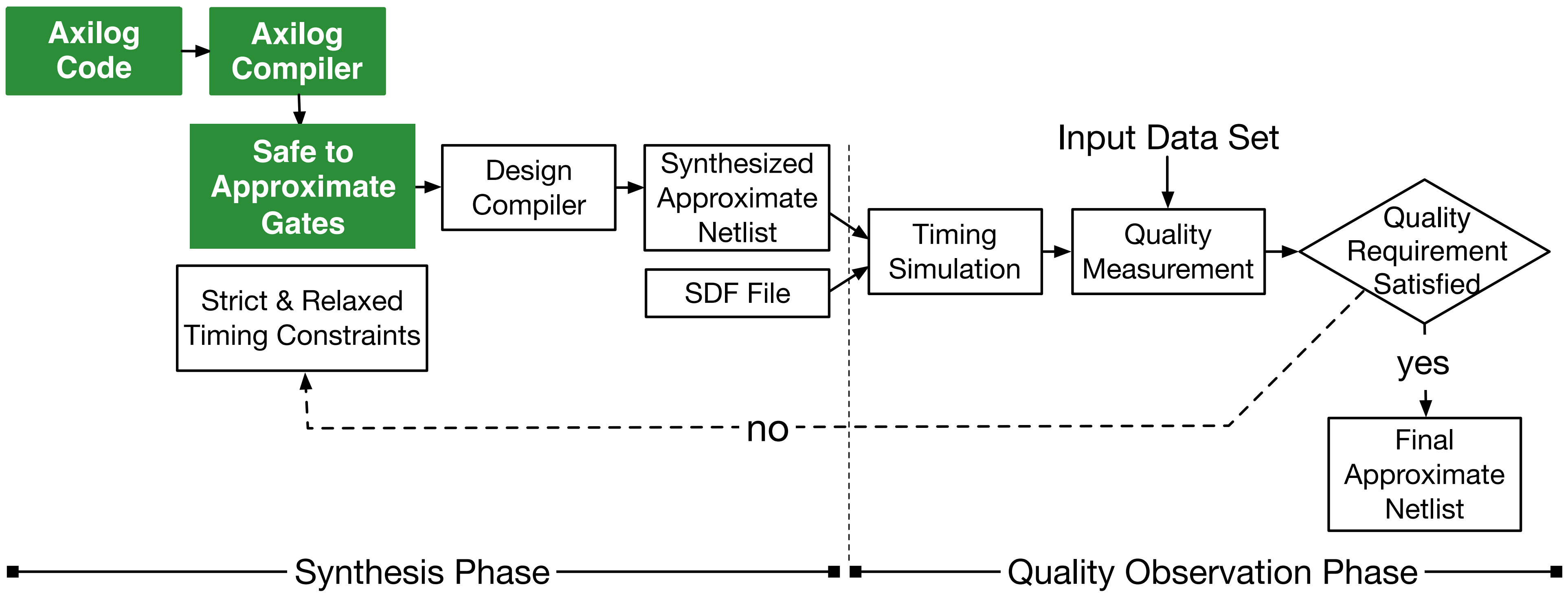
Identify the wires which are driving unannotated wires or annotated with restrict within the module under analysis

Identify the relaxed outputs of the instantiated submodules

Marks any wire that affects a globally restricted wire as precise

Safe to approximate gates

Approximate Synthesis Flow



Axilog Code
Axilog Compiler
Safe to Approximate Gates
Strict & Relaxed Timing Constraints
Design Compiler
Synthesized Approximate Netlist
SDF File
Input Data Set
Timing Simulation
Quality Measurement
Quality Requirement Satisfied
yes
no
Final Approximate Netlist
Synthesis Phase
Quality Observation Phase

Measurements

Tools for Synthesis and Energy Analysis

- Synopsys Design Compiler
- Synopsys Primetime

Timing Simulation with SDF back annotations

- Cadence NC-Verilog

Standard Cell Library

- TSMC 45-nm multi-V_t
- Slowest PVT corner (SS, 0.81V, 0C) for baseline results

Benchmarks

Arithmetic Computation, Signal Processing, Robotics, Machine Learning, Image Processing

FIR
# lines: 113
Signal Processing

Annotations
Design: 6
Reuse: 5

Sobel
# lines: 143
Image Processing

Annotations
Design: 6
Reuse: 3

Brent-Kung
# lines: 352
Arithmetic Computation

Annotations
Design: 1
Reuse: 1

Kogge-Stone
# lines: 353
Arithmetic Computation

Annotations
Design: 1
Reuse: 1

K-means
# lines: 10,985
Machine Learning

Annotations
Design: 7
Reuse: 3

Wallace Tree
# lines: 13,928
Arithmetic Computation

Annotations
Design: 5
Reuse: 3

ForwardK
# lines: 18,282
Robotics

Annotations
Design: 5
Reuse: 4

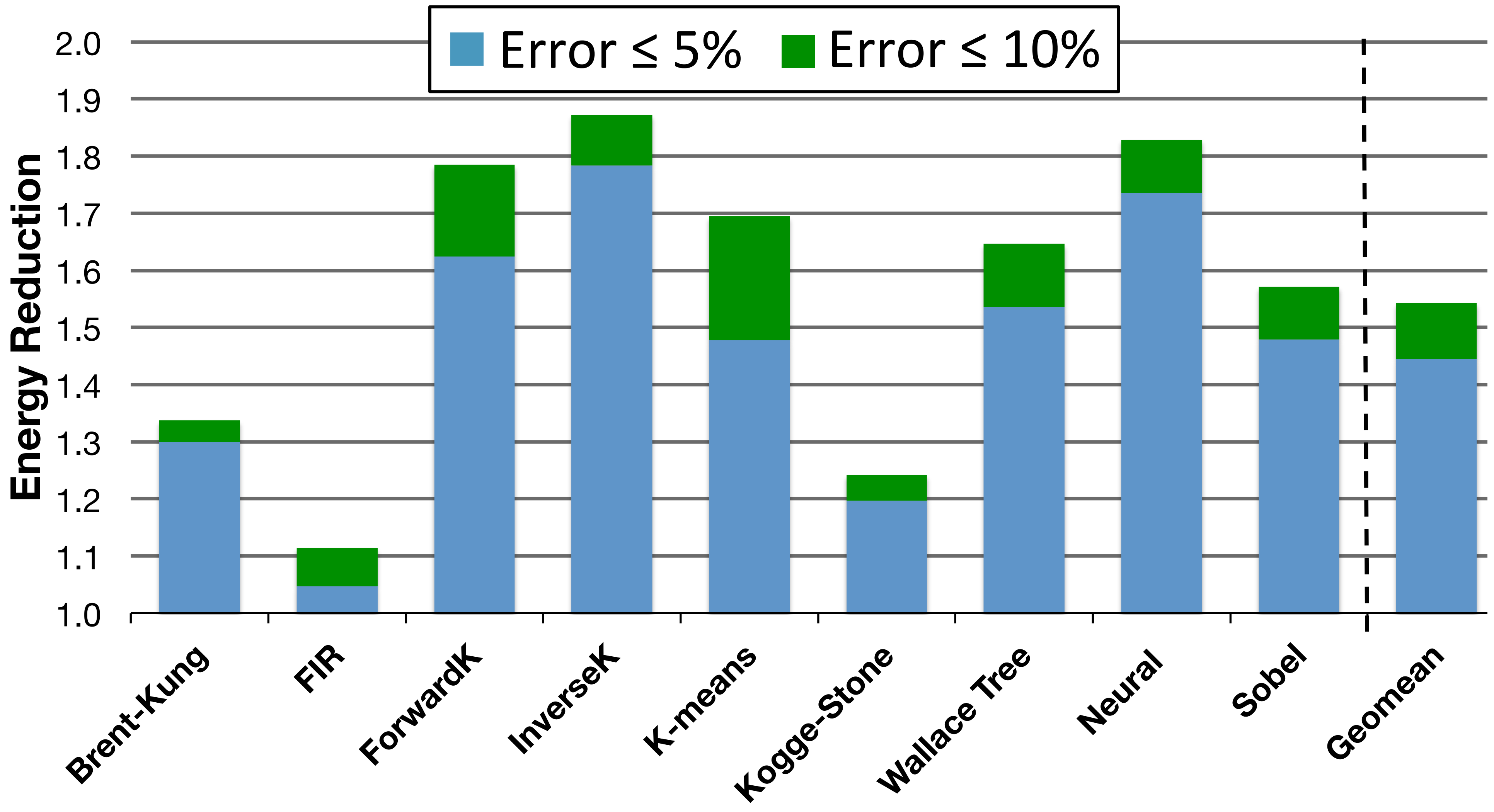
Neural Network
# lines: 21,053
Machine Learning

Annotations
Design: 4
Reuse: 3

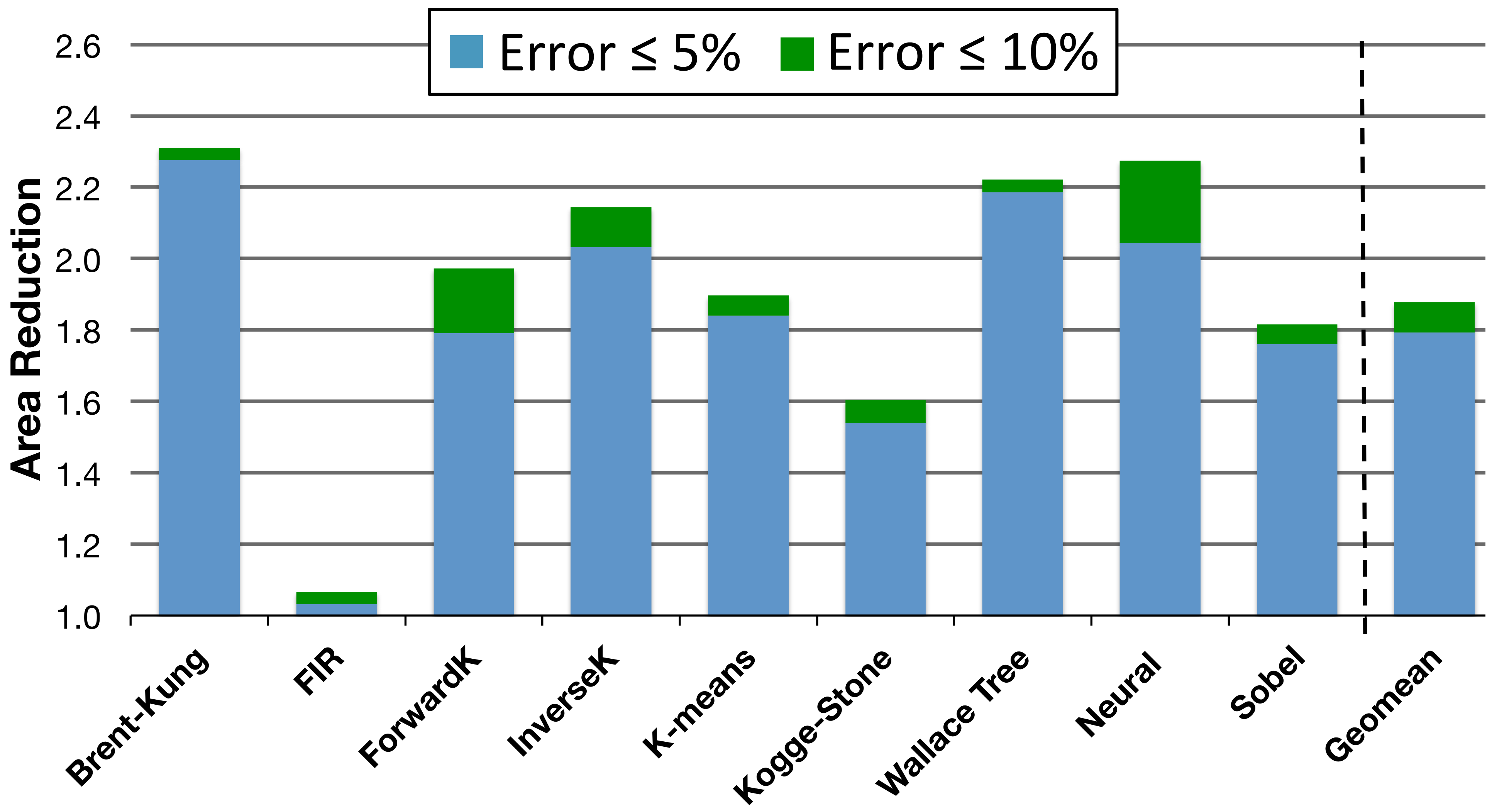
InverseK
# lines: 22,407
Robotics

Annotations
Design: 8
Reuse: 4

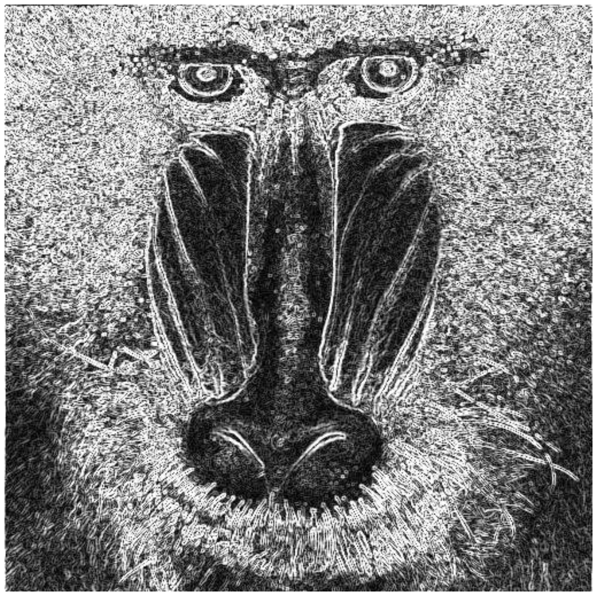
Energy Reduction

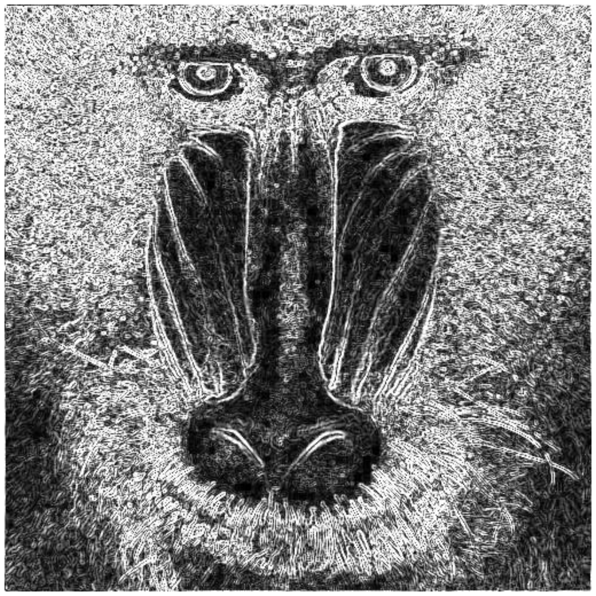


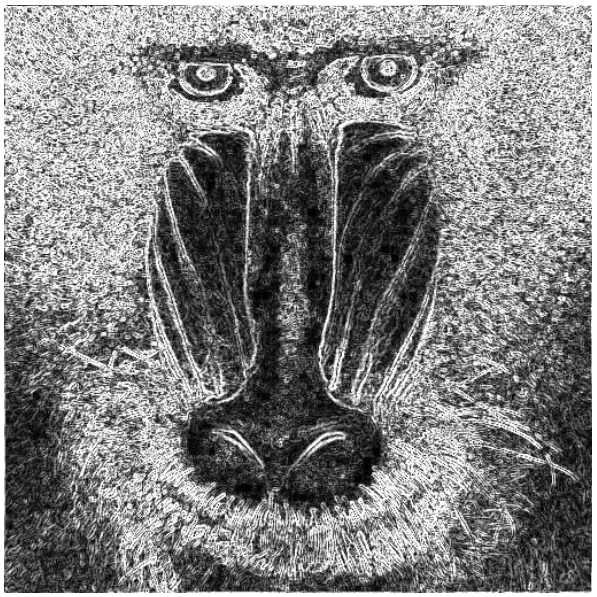
Area Reduction



Output Quality Degradation in Sobel

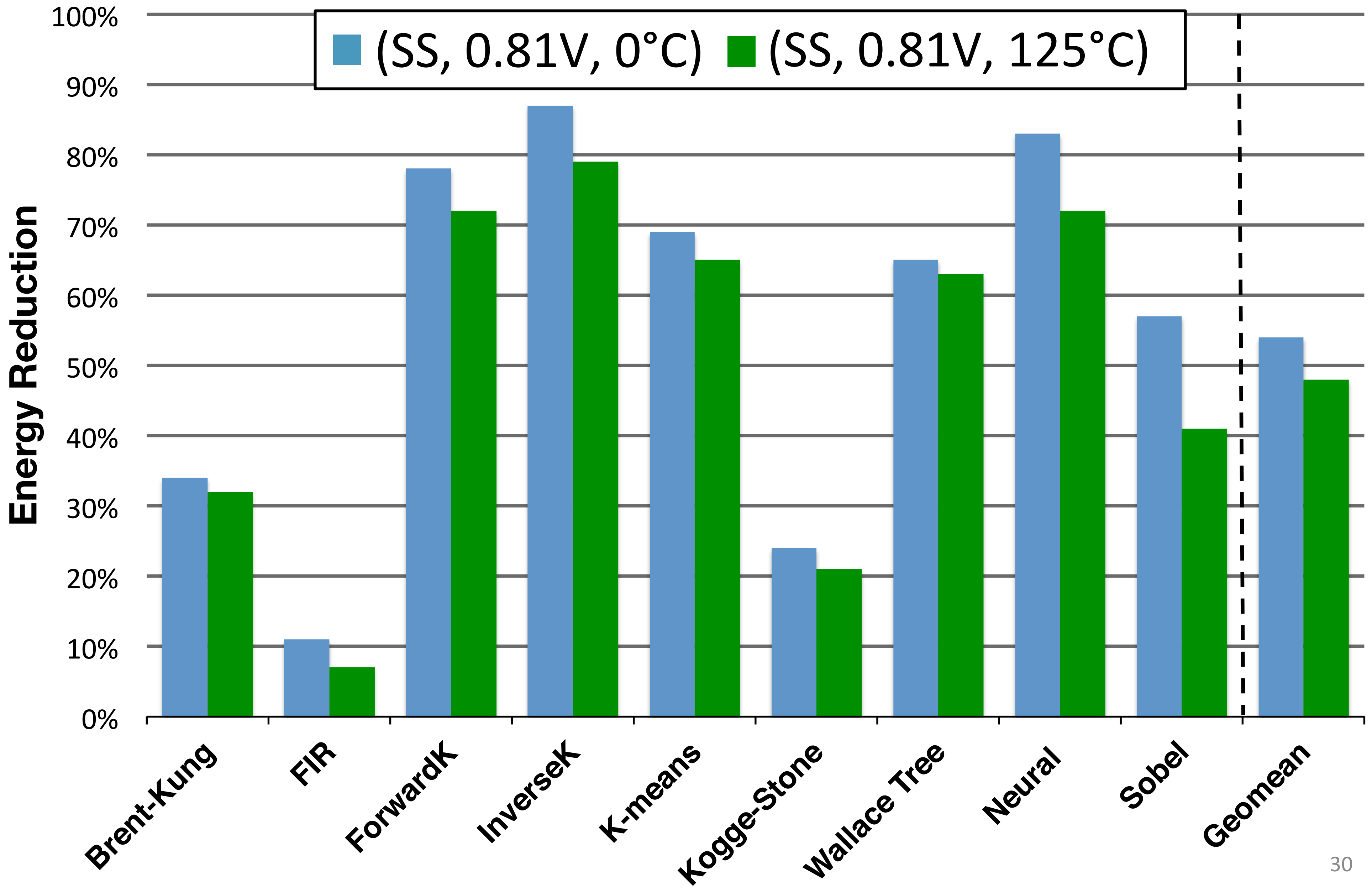

0% Quality Loss


5% Quality Loss


10% Quality Loss

10% Quality loss is nearly indiscernible to the eye yet provides 57% energy savings

Energy Reduction for Different PVT Corners



Axilog

First HDL for Approximation

- Design
- Reuse
- Automation
- High-level
- Backward-compatibility
- Safety

Energy Savings	Area Reduction	Code Annotations
54%	1.9×	2-12

http://www.act-lab.org/artifacts/axilog