Methodical Approximate Hardware Design and Reuse

Amir Yazdanbakhsh **Bradley Thwaites**
Jongse Park Hadi Esmaeilzadeh

Georgia Institute of Technology

Outline

Design of approximate modules

Integration and reuse of approximate modules

Safety analysis

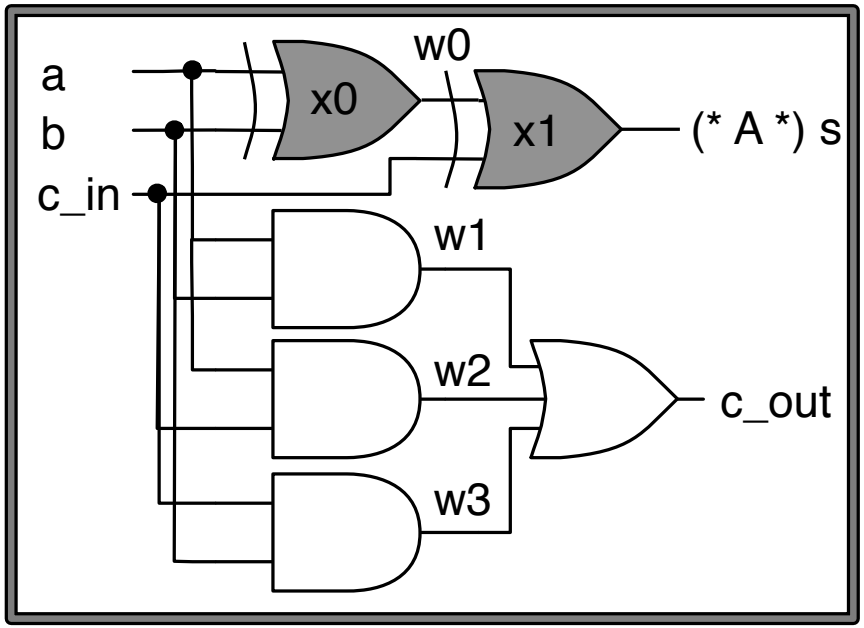
Output quality analysis

Questions

Design Phase

- How does the designer specify what can be approximate?
  - Marking individual gates is burdensome.
  - Mark only output wires as approximate signals.
- Maintain Separation
- Approximation Plan and Interfacing

Approximation Plan

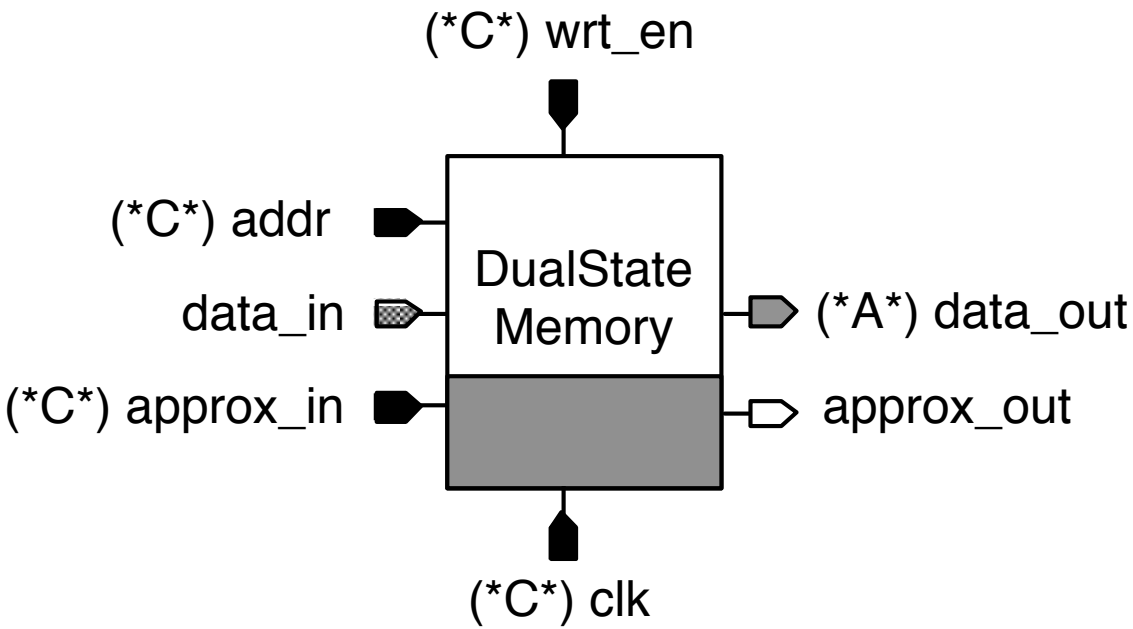



```
module fa(a, b, c_in, c_out, s);
    input a, b, c_in;
    output c_out;
    (*A*) output s;
    wire w0, w1, w2, w3;

    xor x0(w0, a, b);
    xor x1(s, w0, c_in);

    and u2(w1, a, b);
    and u2(w2, a, c_in);
    and u2(w3, b, c_in);
    or  u4(c_out, w1, w2, w3);
endmodule
```

Module Interfacing



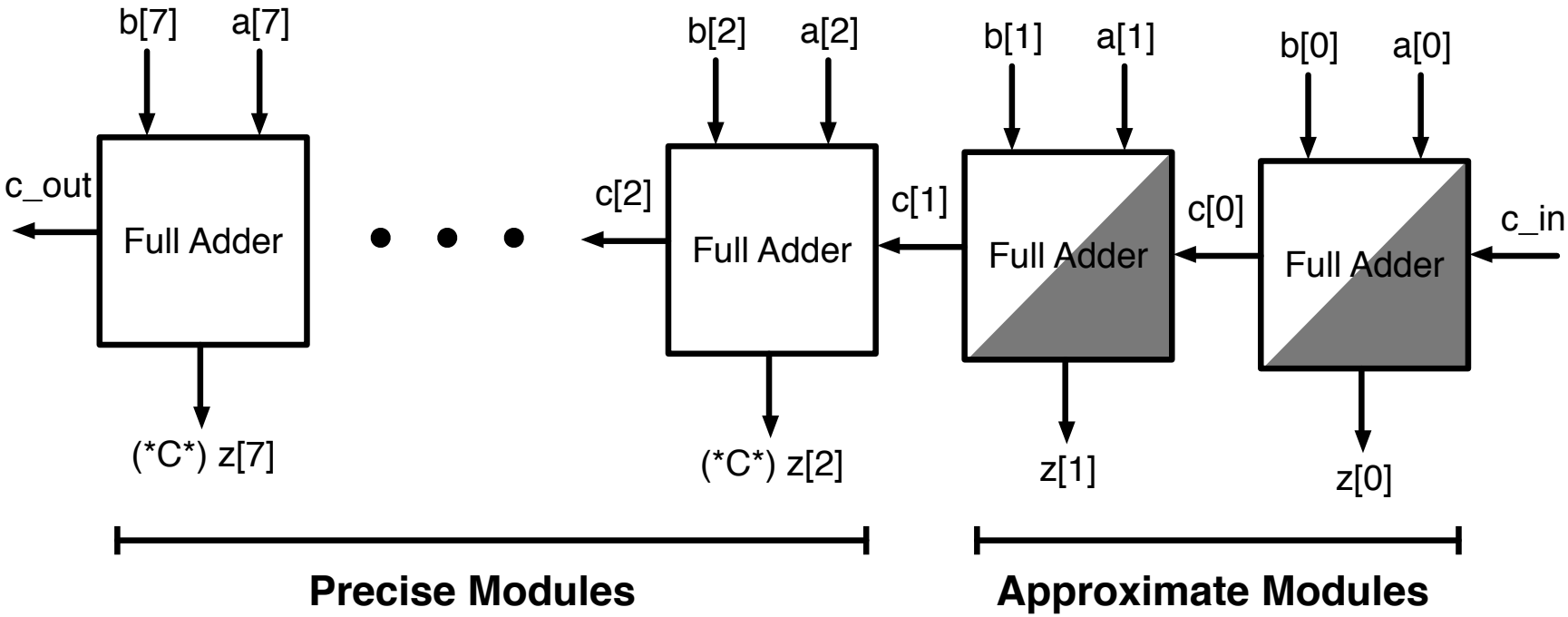

```
module DualStateMemory(
    clk, wrt_en,
    address,
    data_in, approx_in,
    data_out, approx_out);

    (*C*) input clk;
    (*C*) input wrt_en;
    (*C*) input[N-1:0] address;
    input[M-1:0] data_in;
    (*C*) input approx_in;
    (*A*) output[N-1:0] data_out;
    output approx_out;
    ...
endmodule
```

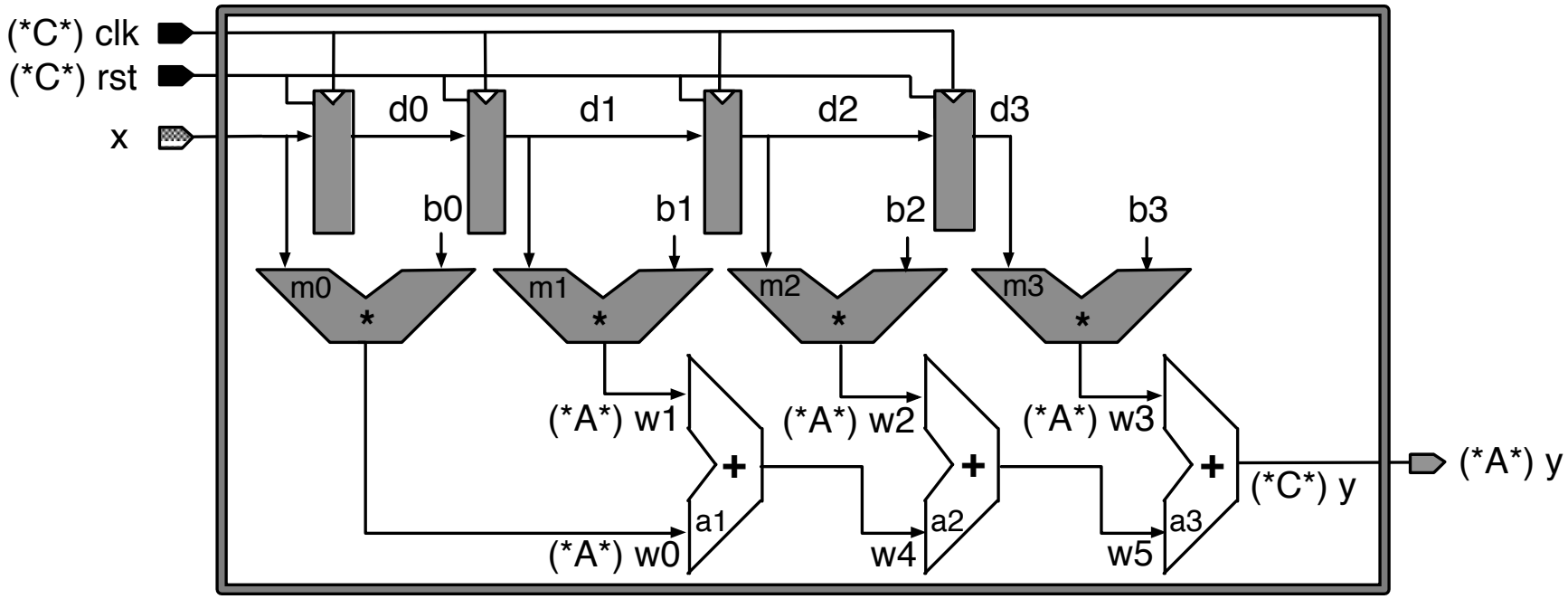
Reuse Phase

- Avoid rewriting modules from scratch
  - Ease of development
- Reuse of IP cores
  - Motivation for innovation and entrepreneurship
- Scalability for very large designs

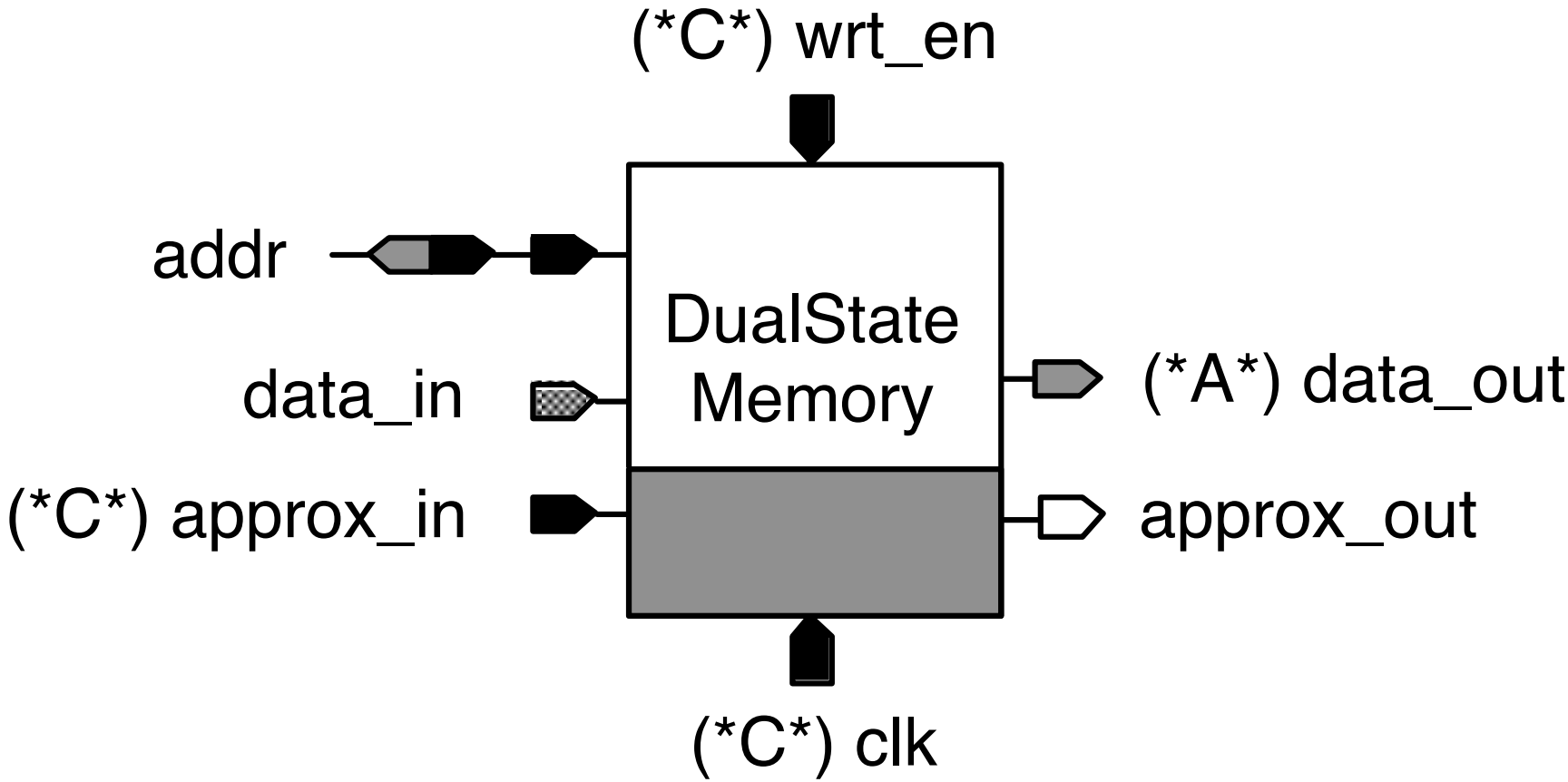
Overriding


b[7]
a[7]
c_out
Full Adder
(*C*) z[7]
b[2]
a[2]
c[2]
Full Adder
(*C*) z[2]
b[1]
a[1]
c[1]
Full Adder
z[1]
b[0]
a[0]
c[0]
Full Adder
c_in
z[0]
Precise Modules
Approximate Modules

Overriding within a Module



Bridging


(*C*) wrt_en
addr
data_in
(*C*) approx_in
DualState
Memory
(*A*) data_out
approx_out
(*C*) clk

Approximation Safety Analysis

- Approximation bridge vs. critical wire?
- Deciding final precision of all gates.
- Backward Slicing Algorithm

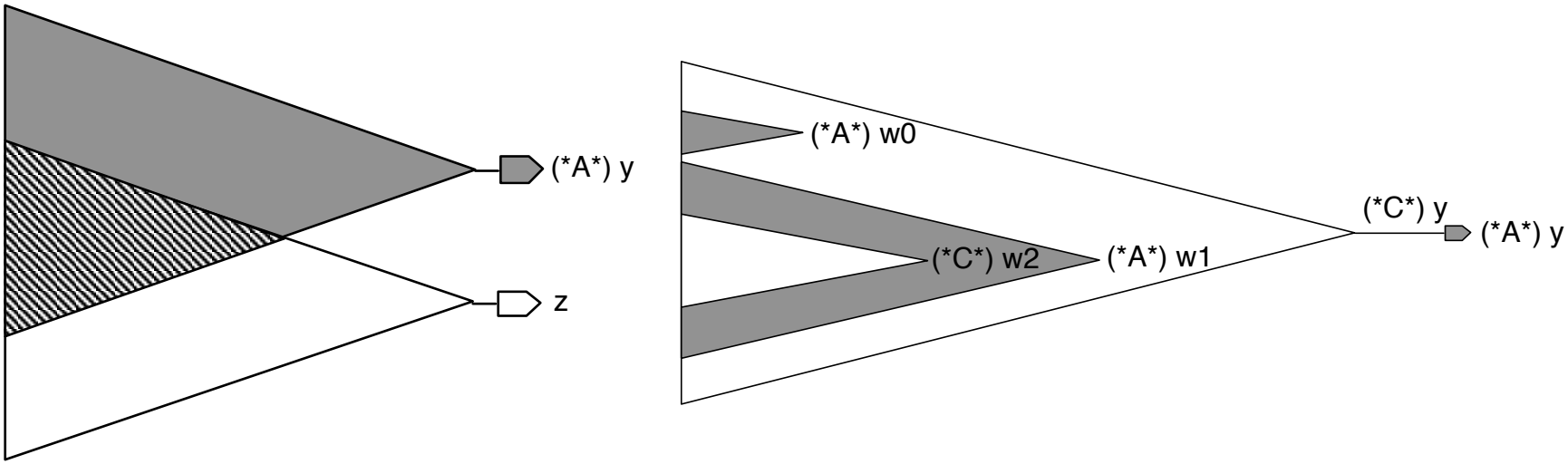
Algorithm 1 Backward slicing to find precise wires.

Inputs: K: Circuit
Θ: Set of precise outputs
Ψ: Set of critical wire overrides
Y: Set of approximate wires overrides

Output: $\Re$: Set of precise wires

Initialize $\Re \leftarrow \emptyset$
Initialize $Q \leftarrow \emptyset$
for each $w_i \in (\Theta \cup \Psi)$ **do**
  enqueue(Q, w_i)
end for
while ($Q \neq \emptyset$) **do**
  $w_i \leftarrow$ dequeue(Q)
  $\Phi \leftarrow$ In K, find input wires of the gate that drives w_i
  for each $w_j \in \Phi$ **do**
    if ($w_j \notin$ Y **and** $w_j \notin \Re$) **then**
      $\Re \leftarrow \Re \cup w_j$
      enqueue(Q, w_j)
    end if
  end for
end while

Cone Analysis



Quality Analysis

- Constraining approximation
- Safety vs. Quality
- (*A: f() < ε*)
- Profiling with test inputs
- Global confidence metric

Questions?