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

\[ \text{module } \text{fa}(\text{a, b, c\_in, c\_out, s}); \]
\[ \text{input } \text{a, b, c\_in}; \]
\[ \text{output } \text{c\_out}; \]
\[ (*A*) \text{ output } \text{s}; \]
\[ \text{wire } \text{w0, w1, w2, w3}; \]
\[ \text{xor } \text{x0}(\text{w0, a, b}); \]
\[ \text{xor } \text{x1}(\text{s, w0, c\_in}); \]
\[ \text{and } \text{u2}(\text{w1, a, b}); \]
\[ \text{and } \text{u2}(\text{w2, a, c\_in}); \]
\[ \text{and } \text{u2}(\text{w3, b, c\_in}); \]
\[ \text{or } \text{u4}(\text{c\_out, w1, w2, w3}); \]
\[ \text{endmodule} \]
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

Precise Modules

Approximate Modules
Overriding within a Module
Bridging

DualState Memory

(*C*) clk

(*C*) wrt_en

addr

data_in

(*C*) approx_in

(*C*) clk

(*A*) data_out

approx_out
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
- $\Theta$: Set of precise outputs
- $\Psi$: Set of critical wire overrides
- $Y$: Set of approximate wires overrides

**Output:** $R$: Set of precise wires

```
Initialize $R \leftarrow \emptyset$
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 $w_j \notin R$) then
      $R \leftarrow R \cup w_j$
      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?