GLOBAL OPTIMIZATION FOR COMPOSITIONAL SYSTEMS

F. Zaraket, J. Pape, A. Aziz, M. Jacome, S. Khurshid

November 13, 2007
1. Co-design of Embedded Systems
   - The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
**Overview of Embedded Systems**

- Composition of hardware and software IP modules
  - Communicate with dedicated hardware devices
- Heterogeneous by nature
  - Application specific integrated circuits (ASICs)
  - Field programmable gate arrays (FPGAs)
  - Embedded software running on one or more processors
- Applications: communications, image processing, and automotive electronics
Co-Optimization of Embedded Systems

- Techniques exist to optimize each IP module and the underlying network (Ch+-DAC-95,HwSwCoDesign-02)
- Integrating computing components introduce new opportunities for optimizations
- Need for co-optimization techniques
  - Work across components
  - Work across hardware and software boundaries
CO-OPTIMIZATION OF EMBEDDED SYSTEMS

- Techniques exist to optimize each IP module and the underlying network (Ch+-DAC-95,HwSwCoDesign-02)
- Integrating computing components introduce new opportunities for optimizations
- Need for co-optimization techniques
  - Work across components
  - Work across hardware and software boundaries

IMPORTANT QUESTIONS

- Can software be developed before hardware is committed?
Co-optimization of Embedded Systems

- Techniques exist to optimize each IP module and the underlying network (Ch+–DAC-95, HwSwCoDesign-02)
- Integrating computing components introduce new opportunities for optimizations
- Need for co-optimization techniques
  - Work across components
  - Work across hardware and software boundaries

Important Questions

- Can software be developed before hardware is committed?
- What if new versions of software used hardware that was optimized away?
Co-design of embedded systems
- The Metropolis framework
  - Motivation for co-optimization

Co-optimization using symbolic execution
- Symbolic execution by example
- Constraint detection and propagation

Case Studies
- Switch fabric
- Vision system
The Metropolis Design Framework

[BA+-Computer-03]

- Express embedded systems in Metropolis Meta Model (MMM) netlists
  - MMM extends a subset of the Java programming language
- Separate computation and communication
  - Processes: computing elements
  - Media: communication elements
- Independent of the model of computation (MoC)
  - Similar to the tagged signal model (Ed+-IEEE-97)
ARCHITECTURE OF METROPOLIS

Communication

Architecture

Function

Meta Model

Constraints

Meta Model Compiler

Front End

Abstract Syntax Tree

Back End

Back End

Back End

Back End

Back End

Elaborate

Simulate (SystemC)

Synthesize

Verify (SPIN)

Metro Shell

Compiler

Function

Communication
Co-design of Embedded Systems

1. The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
Optimization opportunities: IP traffic only, dedicated ports

Need for co-optimization techniques
OUTLINE

1. Co-design of embedded systems
   - The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
Co-design of embedded systems
- The Metropolis framework
- Motivation for co-optimization

Co-optimization using symbolic execution
- Symbolic execution by example
- Constraint detection and propagation

Case Studies
- Switch fabric
- Vision system
SYMBOLIC EXECUTION BY EXAMPLE

JUZI/CVC-LITE [Kh+-TACAS-03]

int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}

x = A, y = B
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
    }
Symbolic Execution by Example

int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}
int x, y;

if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}

\[ x = A, y = B \]
\[ A > B \]
\[ \langle A > B \rangle x = A + B \]
\[ \langle A > B \rangle y = A + B - B = A \]
\[ \langle A \leq B \rangle \text{ end} \]
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}

\[ x = A, y = B \]
\[ A >?B \]
\[ \langle A > B \rangle x = A + B \]
\[ \langle A > B \rangle y = A + B - B = A \]
\[ \langle A > B \rangle x = A + B - A = B \]

\[ \langle A \leq B \rangle \text{ end} \]
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}  
\(\langle A \leq B \rangle \) end

\(x = A, y = B\)
\(A > B\)
\(\langle A > B \rangle \) end

\(\langle A > B \rangle \) end
\(x = A + B\)
\(\langle A > B \rangle \) end
\(y = A + B - B = A\)
\(\langle A > B \rangle \) end
\(x = A + B - A = B\)
\(\langle A > B \rangle \) end
\(B - A > 0\)
\(\langle A > B \rangle \) end
\(x = A + B - A = B\)
\(\langle A > B \rangle \) end
\(B - A > 0\)
\(\langle A > B \rangle \) end
\(x = A + B - A = B\)
\(\langle A > B \rangle \) end
\(B - A > 0\)
\(\langle A > B \rangle \) end
\(x = A + B - A = B\)
\(\langle A > B \rangle \) end
\(B - A > 0\)
\(\langle A > B \rangle \) end
\(x = A + B - A = B\)
\(\langle A > B \rangle \) end
\(B - A > 0\)
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}
int x, y;
if (x > y) {
    x = x + y;
    y = x - y;
    x = x - y;
    if (x - y > 0)
        assert(false)
}
COSE instruments MMM code to perform symbolic execution
1 Co-design of embedded systems
   - The Metropolis framework
   - Motivation for co-optimization

2 Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3 Case Studies
   - Switch fabric
   - Vision system
Constraint Detection

- Use symbolic execution on component level
  - Detect local invariants—constraints
  - Accumulate path conditions
- Annotate ports with detected constraints
- Quality of detected invariants
  - Designer may not know them
  - Designer may not recognize them as useful to optimize other components
**Constraint Propagation**

- Build dependency map between components
- Propagate constraints to other components
  - Start with detected constraints as initial path conditions
  - Use symbolic execution to propagate constraints
MMM AND SYMBOLIC EXECUTION

- Translate MMM into inlined Java
  - MMM is an extension of a Java subset
  - Process, medium, and netlist: class
- Juzi instruments Java code
- CVC-Lite solves and simplifies path conditions
COSE OPTIMIZATIONS

Qualitatively different than those detected by local compiler optimizations

- Eliminate dead code — infeasible path conditions
- Detect range restrictions and re-encode variables
- Detect mutually exclusive executions
  - Target resources sharing and multiplexing
- Annotate MMM with constraints and pass to synthesis tools
  - Apply constant propagation, redundancy removal, and observability don’t care reductions
COSE ARCHITECTURE

Translate to Java

Inline PCs

Juzi: instrument

Juzi: execute

Write optimized MMM

COS ϕ

PCs

Propagate PCs to other modules

Back End

Synthesize

Synthesize

Verify (SPIN)
OUTLINE

1. Co-design of embedded systems
   - The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
**Outline**

1. Co-design of embedded systems
   - The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
MMM FOR Switch Fabric Example
Opportunities: IP traffic only, dedicated ports

Processes: compute schedule, perform transfer, update
**Case Study: Switch Fabric**

**Mixed Traffic, Multiport, 755 Lines of MMM Code**

**IP Traffic Only**

- 4 input ports, 4 output ports, and $8 \times 16$ packet buffers
- 192 minutes and 37K symbolic variables
Case study: Switch fabric
Mixed traffic, multiport, 755 lines of MMM code

Limited output ports
- 4 input ports, 8 output ports, and \(8 \times 16\) packet buffers
- Enabled dropping 4 output ports
- 247 minutes and 61K symbolic variables
1. Co-design of embedded systems
   - The Metropolis framework
   - Motivation for co-optimization

2. Co-optimization using symbolic execution
   - Symbolic execution by example
   - Constraint detection and propagation

3. Case Studies
   - Switch fabric
   - Vision system
Case study: ObjectID Vision System

Flow diagram, 4K lines of C/RTL code

- Labels objects in image with identified names
- Developed for military and medical purposes
  - Deployed for home surveillance applications
Case study: ObjectID vision system
Class diagram, 1255 lines of MMM

- 4 process classes, 3 media classes, and 10 interfaces
Case study: ObjectID vision system

Results: Low resolution capture

Dropped 2 edge detectors in the first iteration
Dropped a segmentation process in the second iteration

15 minutes and 13K symbolic variables
Case Study: ObjectId Vision System

Results: Low Resolution Capture

- Dropped 2 edge detectors in the first iteration
- Dropped a segmentation process in the second iteration
- 15 minutes and 13K symbolic variables
FUTURE...

- Use a difference equation solver instead of CVC-Lite
- Use symbolic execution to optimize linking compilable software modules
Questions?

- Can software be developed before hardware is committed?
- What if new versions of software used hardware that was optimized away?
- ...
SOFTWARE LATENCY QUESTION

Can software be developed before hardware is committed?

- Metropolis supports different design abstraction and implementation refinement levels
- At each refinement level discard COSE optimizations and compute again
What if new versions of software used hardware that was optimized away?

- COSE produces the path conditions it used to optimize the design
  - Can be used as a guide to avoid adding optimized hardware
  - Can be used to undo the optimizations
**Classical Seat Belt Example...**

(a) seat belt controller  
(b) counter  
(c) force key off