A High-Level Synthesis Flow for Custom Instruction Set Extensions for Application-Specific Processors

Nagaraju Pothineni
Google, India

Philip Brisk
UC Riverside

Paolo Ienne
EPFL

Anshul Kumar
IIT Delhi

Kolin Paul
IIT Delhi

Asia and South Pacific Design Automation Conference
Taipei, Taiwan R.O.C. January 21, 2010
Extensible Processors

Applications → Compiler → Assembly code with ISEs

Compiler

Instruction Set Extensions

ISE

Fetch → Decode → Execute → Memory → Write-back

I$ → RF → D$ → RF
Compilation Flow

Application source code

Optimized source code rewritten with ISE calls

High-level program optimizations

ISE identification

Rewrite source code with ISEs

Behavioral HDL description of n ISEs

Machine code with ISE calls

Behavioral description of n ISEs

Target-specific compiler

Assembler

Synthesizer

Processor

Generator

Architecture description

Behavioral HDL description of the processor and n ISEs

Optimized source code rewritten with ISE calls

Structural HDL description of the processor and n ISEs
ISE Synthesis Flow

G₁
...
Gₙ
RF I/O Ports
R W
Clock Period Constraint: \( \Lambda \)
\( \Lambda^* = \Lambda \)

I/O-constrained Scheduling

Reschedule to Reduce Area

Decompose each ISE into 1-cycle Ops

Resource Allocation and Binding

Done

Yes

\( \text{Clk period} \leq \Lambda \)

No

\( \Lambda^* = \Lambda^* - \delta \)
I/O-constrained Scheduling

- I/O ports are a resource constraint
  - Resource-constrained scheduling is NP-complete
  - Optimal algorithm [Pozzi and Ienne, CASES ’05]
ISE Synthesis Flow

**G1**

... **Gn**

**RF I/O Ports**

Clock Period Constraint: \( \Lambda \)

\( \Lambda^* = \Lambda \)

**I/O-constrained Scheduling**

**Reschedule to Reduce Area**

**Decompose each ISE into 1-cycle Ops**

**Resource Allocation and Binding**

Done

Yes

\( \text{Clk period} \leq \Lambda \)

No

\( \Lambda^* = \Lambda^* - \delta \)
Reschedule to Reduce Area

• Goal:
  – Minimize area

• Constraints:
  – Do not increase latency or clock period
  – I/O constraints

• Implementation:
  – Simulated annealing (details in the paper)

2 Adders, 1 Multiplier

1 Adder, 1 Multiplier
ISE Synthesis Flow

I/O-constrained Scheduling

Reschedule to Reduce Area

Decompose each ISE into 1-cycle Ops

Resource Allocation and Binding

Done

Yes

Clk period $\leq \Lambda$

No

$\Lambda^* = \Lambda^* - \delta$
Decompose each ISE into Single-Cycle Operations

ISE | After Scheduling | 1-cycle Ops

A → B → C → D → E

A → B → C

A → B → C → D

A → B → D → E

A → D → E

A → D → E
Decomposition Facilitates Resource Sharing within an ISE
ISE Synthesis Flow

G_1, G_n, RF I/O Ports

Clock Period Constraint: \( \Lambda \)

\( \Lambda^* = \Lambda \)

I/O-constrained Scheduling

Reschedule to Reduce Area

Decompose each ISE into 1-cycle Ops

Resource Allocation and Binding

Done

Yes

\( \text{Clk period} \leq \Lambda \)

No

\( \Lambda^* = \Lambda^* - \delta \)
Resource Allocation and Binding

Two 1-cycle ISEs

Minimum cost common supergraph
Maximum-cost weighted common isomorphic subgraph problem

Maximum-cost weighted common isomorphic subgraph problem
Weighted minimum-cost common supergraph (WMCS) problem

Higher cost common supergraph
2-input operation
No multiplexers needed
(Multiplexers required)

Which solution is better?
Depends on the cost of the multiplexers compared to the merged operations!

No multiplexers needed
Higher cost common supergraph
Which solution is better?
Depends on the cost of the multiplexers compared to the merged operations!
Datapath Merging (DPM)

• Old problem formulation
  – Based on WMCS problem (graph theory)
    • NP-complete [Bunke et al., ICIP ’02]
  – Share as many operations/interconnects as possible
    • [Moreano et al., TCAD ’05; de Souza e al., JEA ’05]
  – Optimize port assignment as a post-processing step
    • NP-complete [Pangrle, TCAD ’91]
    • Heuristic [Chen and Cong, ASPDAC ’04]

• **Contribution:** New problem formulation
  – Accounts for multiplexer cost and port assignment
New DPM Algorithms

• ILP Formulation
  – See the paper for details

• Reduction to Max-Clique Problem
  – Extends [Moreano et al., TCAD ’05]
    • Solve Max-Clique problem optimally using “Cliquer”
  – Identify isomorphic subgraphs up-front
    • Merge isomorphic subgraphs rather than vertices/edges
    • (Details in the paper)
Example

New ISE fragment to merge  Partial merged datapath

Vertex mappings:
1. Map v1 onto v4
2. Allocate a new resource r1; map v1 onto r1

Edge mappings:
e1 could map onto: e3, (v4, r3), (r1, v5), (r1, r3)
Must be compatible with vertex mappings!

Edge mappings:
e1 Map/Map onto: e3, (v4, r3), (r1, v5), (r1, r3)
Must be compatible with vertex mappings to r1
Compatibility

- Vertex/vertex compatibility
  - Yes
  - No

- Vertex/edge compatibility
  - Yes
  - No
Why Edge Mappings?

– Allocate an edge in the merged datapath
– May require a multiplexer
Port Assignment

- Deterministic for non-commutative operators
- NP-complete for every commutative operator
  - [Pangrle, TCAD ’91]

We want this!

No!
Edge Mappings = Port Assignment

Commutative Operator

Mapping:
\[ e_1: (v_4, v_6, L) \]

Mapping:
\[ e_1: (v_4, v_6, R) \]
Compatibility Graph

• Vertices correspond to mappings
  – Vertex mappings
    • Weight is 0 for vertex \( \rightarrow \) vertex
    • Weight is resource cost for vertex \( \rightarrow \) new resource
  – Edge mappings, including port assignment
    • Weight is 0 if edge exists in merged datapath
    • Weight is estimated cost of increasing mux size by +1 otherwise

• Place edges between compatible mappings
  – Each max-clique corresponds to a complete binding solution
  – Goal is to find max-clique of minimum weight
Compatibility Graph

\[ f_E(e_2) = (r_2, v_6, L) \]
\[ w = A_{\text{mux}}(2) \]

\[ f_E(e_2) = (r_2, v_6, R) \]
\[ w = A_{\text{mux}}(2) \]

\[ f_E(e_1) = (v_4, v_6, L) \]
\[ w = 0 \]

\[ f_E(e_1) = (v_4, v_6, R) \]
\[ w = A_{\text{mux}}(2) \]

\[ f_E(e_1) = (r_1, v_6, L) \]
\[ w = A_{\text{mux}}(2) \]

\[ f_E(e_1) = (r_1, v_6, R) \]
\[ w = A_{\text{mux}}(2) \]

\[ f_E(e_1) = (v_4, r_3, R) \]
\[ w = 0 \]

\[ f_V(v_1) = v_4 \]
\[ w = 0 \]

\[ f_V(v_2) = r_2 \]
\[ w = A(\ ) \]

\[ f_V(v_1) = r_1 \]
\[ w = A(\ ) \]

\[ f_E(e_1) = (v_4, r_3, L) \]
\[ w = 0 \]

\[ f_E(e_1) = (r_1, r_3, L) \]
\[ w = 0 \]

\[ f_E(e_1) = (r_1, r_3, R) \]
\[ w = 0 \]

\[ f_E(e_2) = (r_2, r_3, L) \]
\[ w = 0 \]

\[ f_E(e_2) = (r_2, r_3, R) \]
\[ w = 0 \]
Experimental Setup

• Internally developed research compiler
  – 1-cycle ISEs [Atasu et al., DAC ’03]
  – RF has 2 read ports, 1 write port
  – Standard cell design flow, 0.18μm technology node

• Five DPM algorithms
  – Baseline No resource sharing
  – ILP (Optimal) [This paper]
  – Our heuristic* [This paper]
  – Moreano’s heuristic* [Moreano et al., TCAD ’05]
  – Brisk’s heuristic [Brisk et al., DAC ’04]

* Max-cliques found by “Cliquer”
Moreano's heuristic is sometimes competitive. Brisk's heuristic performed as well as ours for one benchmark. Moreano's heuristic is not always competitive. Brisk's heuristic was NOT competitive for three benchmarks.
Critical Path Delay Increase

Critical Path Delay Increase (%) – Our heuristic

- adpcm
- jpeg
- gsm encode
- des
- sha
- aes
<table>
<thead>
<tr>
<th>Heuristic</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0</td>
</tr>
<tr>
<td>Optimal (ILP)</td>
<td>3-8 hours</td>
</tr>
<tr>
<td>Our heuristic</td>
<td>2-10 minutes</td>
</tr>
<tr>
<td>Moreano's heuristic</td>
<td>~1 minute</td>
</tr>
<tr>
<td>Brisk's heuristic</td>
<td>&lt; 5-10 seconds</td>
</tr>
</tbody>
</table>
Experimental Setup

• Internally developed research compiler
  – Multi-cycle ISEs [Pozzi and Ienne, CASES ’05]
  – RF has 5 read ports, 2 write ports
  – Standard cell design flow, 0.18 μm technology node

• Four versions of our flow
  – Single-cycle ISE (None) (1-cycle)
  – Full flow (200 MHz) (Multi-cycle)
  – No rescheduling (200 MHz) (Multi-cycle)
  – Baseline flow (200 MHz) (Multi-cycle)
    (Resource sharing and binding step disabled)
Single vs. Multi-cycle ISEs

Area savings (%)

- Single-cycle ISE (No clock period constraint)
- Full flow (200 MHz)

0% Baseline flow (200 MHz)

Resource sharing across multiple cycles of the same ISE

Cost of extra registers

ADPCM  AES  DES  GSM  H263  IDCT  JPEG  SHA
Impact of Rescheduling

- Single-cycle ISE (no clock period constraint)
- Full Flow (200 MHz)
- No rescheduling (200 MHz)
Conclusion

• HLS Flow for ISEs
  – RF I/O constraints
    • Min-latency scheduling is NP-complete
  – Requires two scheduling steps
    • Rescheduling is important for area reduction

• Resource allocation and binding
  – Modeled as a datapath merging problem
  – New problem formulation
    • Multiplexer cost
    • Port assignment