An Efficient Performance Improvement Method Utilizing Specialized Functional Units in Behavioral Synthesis

Tsuyoshi Sadakata, and Yusuke Matsunaga
Kyusyu University, Japan
Motivation

- **Specialized Functional Units (SFUs)** (e.g. Multiply-Accumulator) can be designed for specific operation patterns to achieve *shorter delay and/or smaller area* than cascaded basic functional units (e.g. Multiplier & Adder)
- Introducing SFUs into behavioral synthesis can improve synthesis results
- Because SFUs are *less flexible for resource sharing*, utilizing Specialized Functional Units in behavioral synthesis considering performance and area trade-off is a complicated problem
Related Works

• Integer Linear Programming based Methods
  – Landwehr et al, ``Oscar: optimum simultaneous scheduling, allocation and resource binding based on integer programming”, EuroDAC94
  – Marwedel et al., ``Built-in chaining: Introducing complex components into architectural synthesis”, ASPDAC97

Long computational time can be required for large problems

• Heuristic Methods
  – Corazao et al., ``Performance optimization using template mapping for datapath-intensive high-level synthesis”, IEEE Trans. on CAD96
  – Bringmann et al., ``Cross-level hierarchical high-level synthesis”, DATE98

Maximizing performance ignoring the increase of resources
Proposed Method

• A heuristic method utilizing SFUs for a simultaneous Module Selection, Functional Unit Allocation, and Scheduling problem considering performance/area trade-off
  – Constraint: clock cycle time & total functional unit area
  – Objective: minimize # of clock cycles
  – Approach
    1. enumerate several feasible solutions at Module Selection
    2. solve other sub-problems for each solution of Module Selection

• Main Contribution
  Proposal of a novel heuristic Module Selection algorithm to restrict enumerated solutions effectively
Module Selection Sub-Problem

• Enumerate several feasible **Module Set Vectors** satisfying clock cycle time & total functional unit area constraint

\[
msv = (n_1, n_2, \ldots, n_{|FU|})
\]

\(FU\) : a set of functional unit types

\(n_i\) : selected # of \(i\) th functional unit type

\(msv[i]\) : notation for \(i\) th element (\(n_i\))

**Feasible Module Set Vector (FMSV)**

• Synthesis target can be implemented with the msv

• The msv satisfies given constraint

**Inclusion Relation between MSVs**

\(msv\) is included in \(msv'\) \(\iff\)

\[\forall i = 1, 2, \ldots, |FU|, \msv[i] \leq ms'v[i]\]
Proposed Module Selection Algorithm

- Only **maximal FMSVs** are enumerated
  - maximal FMSV: no other FMSV includes the msv

Only FMSVs close to constraint boundary border are enumerated

- maximal FMSVs are divided into several groups based on **unit FMSVs**
  - unit FMSV:
    \[
    msv_{unit}[i] = \begin{cases} 
    0 & (msv_{maximal}[i] = 0) \\
    1 & (msv_{maximal}[i] \geq 1) 
    \end{cases}
    \]

For each group, minimum # of cycles is estimated with only unit FMSV

From a unit FMSV with the best estimated value, constant number of maximal FMSVs are enumerated
Experiment

• Effect of utilizing SFU is evaluated in two ways
  – ALL: a heuristic method that enumerated all maximal FMSVs
  – OUR: a heuristic method with the proposed algorithm

• Synthesis Target
  – bdist2(# of operations: 43, MediaBench:MPEG2 Encoder)
  – fdct(# of operations: 138, MediaBench:JPEG Encoder)

• Functional Unit Library
  – Basic functional units (e.g. adder, multiplier)
  – SFU
    • Carry-Save Adder based construction algorithm for addition based operations (provided by Synopsys Module Compiler)
  – All units were synthesized with Synopsys Module Compiler under maximum delay constraint 3 ns or 6 ns with a cell library for HITACHI 0.18um CMOS process technology provided from VDEC

• Constant number for the enumeration of maximal FMSVs with the proposed algorithm
  – 1,000
Experimental Results

# of clock cycles
(bdist2, clock cycle time constraint: 6ns)

# of clock cycles
(fdct, clock cycle time constraint: 6ns)

The result cannot be obtained without SFUs

OUR with SFUs:
ave. 17.5%, max. 35.7% reduction

The result cannot be obtained without SFUs

OUR with SFU:
ave. 10.4%, max. 15.9% reduction

Computational Time Comparison

ALL with SFUs: max. 7,588 sec (bdist2), max. 8,218 sec (fdct)

OUR with SFUs: max. 149 sec (bdist2), max. 857 sec (fdct)
Conclusion

• An efficient performance improvement method utilizing SFUs is proposed

• Performance improvement under clock cycle time and total functional unit area constraint can be achieved in practical time with the proposed method

• Experimental results show that utilizing specialized functional units has achieved 13.3% on average, maximally 35.7% reduction of # of clock cycles within 15 minutes
Thank you for your attention.