Optimal Partition with Block-Level Parallelization in C-to-RTL Synthesis for Streaming Applications

Authors: Shuangchen Li, Yongpan Liu, X.Sharon Hu, Xinyu He, Pei Zhang, and Huazhong Yang

2013/01/23
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
Introduction: Background

- Application complexity increasing
- Well-developed software libraries
- Low speed, high power
- MPSoCs architecture
- Hardware design complexity

How to rapidly design hardware from existing software algorithms?
How to rapidly design hardware from existing software algorithms?

- This challenge is now new. However,
  - Ever increasing design gap
  - Progression of EDA tools

C-to-RTL synthesis can help to tackle this challenge
Introduction: Motivation

• **C2RTL tools are promising**
  – A number of C2RTL tools
    – A DVB-SH Turbo Decoder [8]
    – A face detection system [9]

10 days
3x
Very complicated design
Introduction: Motivation (cont’d)

• **However**, state-of-the-art C2RTL tools suffer from:
  
  - **Low Quality of results (QoR)** for large C programs
  - **System-level optimization** options are limited

<table>
<thead>
<tr>
<th>Design</th>
<th>Bluespec</th>
<th>Catapult-C</th>
<th>Xilinx</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUTs</td>
<td>5863</td>
<td>29549</td>
<td>2067</td>
</tr>
<tr>
<td>FFs</td>
<td>3162</td>
<td>8324</td>
<td>1386</td>
</tr>
<tr>
<td>Block RAMs</td>
<td>42,475,202</td>
<td>4,070,603</td>
<td>1.26x</td>
</tr>
<tr>
<td>Equivalent Gate Count</td>
<td>74267,741</td>
<td>74.296,730</td>
<td>1.26x</td>
</tr>
<tr>
<td>Frequency (MHz)</td>
<td>108.5</td>
<td>91.2</td>
<td>145.3</td>
</tr>
</tbody>
</table>

- **Flatten approach**
- **Hierarchical approach**

**Control and optimization at the system level are needed**
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
Overview: Our work

• **Given**
  – a large C program for a streaming application
  – system constraints (latency, area, …)

• **Determine**
  – how to partition the code into pipelined blocks
  – which blocks should be parallelized

• **The objectives**
  – Improve synthesis result quality
  – Provide more system-level optimization options

[Note: The terms **Partition** and **Parallelization** are highlighted in the text.]
Overview: Design flow

• **STEP 1:**
  - We use eXCite here

• **STEP 2:**
  - Determine partition and parallelization

• **STEP 3:**
  - Synthesize each block with a C2RTL tool

• **STEP 4:**
  - Construct the complete system

- C programs need to be synthesized
- Throughput and Area constraints
- Extract parameters of N functions
- Optimize partition and parallelization
- Block-level parallelization
- Partition
- Synthesize blocks by a C2RTL tool (eXCite)
- Assemble the modules into a single design
- Structure of the final system

- Module 1-1
- Module 1-2
- Module 2
- ...
Overview: An example

- **Given a C program:**
  - In the straight-line style

- **Given constraints:**
  - System throughput and area

- **Partition:**
  - Which functions should be synthesized together as one pipeline stage

- **Parallelization:**
  - Which synthesized modules should be parallelized
Overview: Challenges

• The design space is large:
  – Partition has a great impact on throughput and area
  – Parallelization has a great impact on throughput and area
  – The Pareto optimal solutions

• The importance to simultaneously consider partition and parallelization:
  – The constraints are for the system after both partition and parallelization
  – If optimizing them separately, it is not clear how to apply the constraints to each problem individually

A GSM case

Latency ($r_{\text{all}}^{-1}$) vs. Area ($a_{\text{all}}$) for different partition and parallelization strategies.
### Overview: Related work

<table>
<thead>
<tr>
<th>Application</th>
<th>Input</th>
<th>Target</th>
<th>Partition</th>
<th>Parallelization</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Cong and et al., in DATE2012[12]</td>
<td>Stream</td>
<td>C</td>
<td>FPGA</td>
<td>Manually</td>
</tr>
<tr>
<td>Y. Hara and et al., in IEICE[14]</td>
<td>General</td>
<td>C</td>
<td>FPGA</td>
<td>ILP</td>
</tr>
<tr>
<td>This work</td>
<td>Stream</td>
<td>C</td>
<td>FPGA</td>
<td>Both MILP and Heuristic  (consider simultaneously)</td>
</tr>
</tbody>
</table>

- A somewhat related line of work is mapping C programs to MPSoCs (software mapping):
  - Blocks (or tasks) can be assigned to the same processor
  - The processor area is given
Overview: Our Contribution

• A novel MILP based formulation
  – Find a partition and parallelization solution with maximum throughput or minimum area while satisfying a given area or throughput constraint, respectively

• An efficient heuristic algorithm
  – Overcome the scalability challenge facing the MILP formulation

• Validation of the proposed methods
  – Developing FPGA based accelerators for seven streaming applications
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
MILP-Based Solution: Formulation

• **Given function parameters (Para)**
  – Area, throughput … of each function

• **Determine \((x_n)\)**
  – Which functions should be clustered to form blocks
  – Which blocks should be parallelized

• **Objective:**
  – min. Area \((a_{all}(x_n, Para))\) or max. Throughput \((r_{all}(x_n, Para))\)

• **Subject to:**
  – Area constraints \((a_{all}<A_{req})\)
  – Throughput constraints \((r_{all}>R_{req})\)
  – Connectivity constraints
MILP-Based Solution: Variable

- We use \( \{x_n\} \in \mathbb{Z} \) to represent partition and parallelization:
  - Partition: If \( x_n = 0 \): \( F_n \) and \( F_{n+1} \) are in the same block
  - Parallelization: If \( x_n \neq 0 \): The parallelism degree of block with \( F_n \) is \( x_n \)

- We also use \( \{y_{i,j}\} \in \text{Binary} \) to represent partition
  - \( y_{i,j} = 1 \) means \( F_i, F_{i+1}, \ldots, F_j \) are clustered

\[ \begin{align*}
  x_n & \in \{0, 2, 1, 16, 0, 1, 0, 3\} \\
\end{align*} \]
MILP-Based Solution: Details

- To calculate throughput $r_{all}(x_n, Para)$:

  $$r_{all} \leq r_{i,j} \text{ if } y_{i,j} = 1 \quad (1)$$

  $$r_{i,j} = \begin{cases} 
  x_j y_{i,j} / T_{i,j} & x_j y_{i,j} < P_{i,j} \\
  1 / \max\{T_{i,j}^{in}, T_{i,j}^{out}\} & \text{otherwise} 
  \end{cases} \quad (2)$$

- To calculate area $a_{all}(x_n, Para)$:

  $$a_{all}^{le/mem} = a_{fifo}^{le/mem} + \sum_{i=1}^{N} \sum_{j=1}^{N} \left((x_j - 1)O_{i,j}^{le/mem} + x_j A_{i,j}^{le/mem}\right) y_{i,j} \quad (3)$$

- Connectivity constraints:

  $$\begin{cases} 
  \sum_{i=1}^{i=n} y_{i,n} \leq x_n \\
  x_n = 1 \text{ when } \sum_{i=1}^{i=n} y_{i,n} = 0 \\
  \sum_{i=1}^{i=j-1} y_{i,j-1} = \sum_{i=j}^{i=N} y_{j,i} \quad \forall j \in [2, N] \\
  \sum_{i=1}^{i=j} y_{i,j} + \sum_{i=j}^{i=N} y_{j,i} - y_{j,j} \leq 1 \quad \forall j \in [1, N] 
  \end{cases} \quad (4, 5, 6)$$
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
Heuristic Solution: Overview

• **Motivation:**
  – MILP is not scalable
  – Bad feasible regions may incur long running time even when $N$ is small

• **Consider partition and parallelization separately (constructive algorithm):**
  – Parallelization before partition *to increase throughput*: \textit{Incx()}
  – Partition for the given parallelization *to reduce area*: \textit{Clust()}
  – Implement \textit{Incx()} and \textit{Clust()} in a backtracking iterative way
Heuristic Solution: Algorithm

- **Incx()**: Parallelization before Partition to *increase throughput*
- **Clust()**: Partition for the given Parallelization to *reduce area*
Heuristic Solution: Algorithm (cont’d)

• \textit{Incx()}, Parallelization before Partition:
  – Increase the parallelization degree of the bottleneck function

• \textit{Clust()}, Partition under the given Parallelization:
  – Model the blocks and their connections as a graph
  – Convert the problem to a \textit{shortest path problem}

\begin{tikzpicture}
  \node[circle, draw] (B1) at (0,0) {$B_{1,1}$};
  \node[circle, draw] (B2) at (1,0) {$B_{2,2}$};
  \node[circle, draw] (B3) at (2,0) {$B_{3,3}$};
  \node[circle, draw] (A1) at (0,-1) {$A_{1,1}$};
  \node[circle, draw] (A2) at (1,-1) {$A_{2,2}$};
  \node[circle, draw] (A3) at (2,-1) {$A_{3,3}$};
  \node[circle, draw] (B) at (0,-2) {$B_{1,2}$};
  \node[circle, draw] (B') at (1,-2) {$B_{2,3}$};
  \node[circle, draw] (END) at (2,-2) {$\text{END}$};
  \node[circle, draw] (Begin) at (0,1) {$\text{Begin}$};

  \draw[->] (Begin) -- (B1) node[midway, below] {0};
  \draw[->] (B1) -- (B2) node[midway, above] {A_{1,1}};
  \draw[->] (B2) -- (B3) node[midway, above] {A_{2,2}};
  \draw[->] (B3) -- (END) node[midway, above] {A_{3,3}};
  \draw[->] (B1) -- (B) node[midway, above] {A_{1,1}};
  \draw[->] (B) -- (B2) node[midway, above] {A_{1,2}};
  \draw[->] (B2) -- (B') node[midway, above] {A_{2,3}};
  \draw[->] (B') -- (END) node[midway, above] {A_{2,3}};
  \draw[->] (Begin) -- (B') node[midway, below] {0};
  \draw[->] (B1) -- (B') node[midway, below] {21};
\end{tikzpicture}
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
Experiments: Set up

- **7 Benchmark [21]:**
  - ADPCM
  - JPEG encoder/decoder
  - AES encryption/decryption
  - GSM
  - Filter Groups

- **Environment & flow:**
  - C2RTL: eXCite
  - Logic synthesis: Quartus II (cyclone II)
  - Simulation: Modelsim

---

**Our solution:**
- Optimize partition and parallelization

**eXCite C2RTL tool:**
- Implement hardware

**Altera Quartus tool:**
- Area evaluation

**Mentor Modelsim tool:**
- Throughput evaluation
Experiments: Validate proposed method

- Min. area for GSM case

- Heuristic solutions differ from the MILP results by 2.3% on average
Exp.: Validate proposed method (cont’d)

- Min. Area for 7 benchmarks

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Constraints $R_{\text{req}}^{-1}$</th>
<th>MILP vs. Heuristic results</th>
<th>$r_{\text{all}}^{-1}$</th>
<th>$\alpha_{\text{all}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>${x_n}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADPCM</td>
<td>MILP</td>
<td>$200$</td>
<td>${0,2,0,0,1,1}$</td>
<td>197</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${2,0,0,1,1,1}$</td>
<td>164</td>
<td>17513</td>
</tr>
<tr>
<td>AES encryption</td>
<td>MILP</td>
<td>$5000$</td>
<td>${1,1,1,0,1,0,1}$</td>
<td>4678</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${1,1,1,0,1,0,1}$</td>
<td>4678</td>
<td>16463</td>
</tr>
<tr>
<td>AES decryption</td>
<td>MILP</td>
<td>$4500$</td>
<td>${1,0,0,2,0,0,2}$</td>
<td>4325</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${1,2,0,2,2,0,2}$</td>
<td>3867</td>
<td>31036</td>
</tr>
<tr>
<td>JPEG encoder</td>
<td>MILP</td>
<td>$579$</td>
<td>${2,1,1,1,1,1,1,1,0,2}$</td>
<td>355</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${2,1,1,1,1,1,1,0,2}$</td>
<td>355</td>
<td>23944</td>
</tr>
<tr>
<td>JPEG decoder</td>
<td>MILP</td>
<td>$2500$</td>
<td>${0,0,0,0,1,0,0,2}$</td>
<td>5766</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${0,0,0,0,1,2,0,1}$</td>
<td>1670</td>
<td>8502</td>
</tr>
<tr>
<td>GSM</td>
<td>MILP</td>
<td>$1000$</td>
<td>${2,1,0,2,0,2,0,0,1}$</td>
<td>833</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${2,2,0,2,2,0,0,0,1}$</td>
<td>833</td>
<td>18775</td>
</tr>
<tr>
<td>Filter Groups</td>
<td>MILP</td>
<td>$18000$</td>
<td>${0,0,1,0,1,0,2,2,2,0,1,1,1}$</td>
<td>17102</td>
</tr>
<tr>
<td></td>
<td>Heur.</td>
<td>${1,0,0,1,0,1,0,2,2,2,0,1,1,1}$</td>
<td>10372</td>
<td>28994</td>
</tr>
</tbody>
</table>

Solutions are effective with different applications

Heuristic with a difference of 7.5% on average
Experiments: Running time

- Running time:

<table>
<thead>
<tr>
<th>Bench-mark</th>
<th>Objective</th>
<th>Constraints</th>
<th>Time (sec)</th>
<th>Result ($r_{all}^{-1}$, $a_{all}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$R_{req}^{-1}$</td>
<td>$A_{req}$</td>
<td>MILP</td>
</tr>
<tr>
<td>GSM</td>
<td>min $a_{all}$</td>
<td>1000</td>
<td>–</td>
<td>9.089</td>
</tr>
<tr>
<td>(N=10)</td>
<td>max $a_{all}$</td>
<td>3000</td>
<td>17000</td>
<td>37.648</td>
</tr>
<tr>
<td></td>
<td>max $r_{all}$</td>
<td>109000/400000</td>
<td>–</td>
<td>41.135</td>
</tr>
<tr>
<td></td>
<td>max $r_{all}$</td>
<td>–</td>
<td>18900/300000</td>
<td>Failed</td>
</tr>
<tr>
<td>Filter groups</td>
<td>min $a_{all}$</td>
<td>19000</td>
<td>–</td>
<td>355.80</td>
</tr>
<tr>
<td>(N=14)</td>
<td>max $r_{all}$</td>
<td>–</td>
<td>50000</td>
<td>395.47</td>
</tr>
<tr>
<td></td>
<td>max $r_{all}$</td>
<td>–</td>
<td>30000/25000</td>
<td>Failed</td>
</tr>
</tbody>
</table>

Note: With two separated constraints for $A_{req}^{le}$ and $A_{req}^{mem}$, respectively.

- The heuristic solutions are worse by 7.2% on average.
Outline

• Introduction
• Overview
• MILP-Based Solution
• Heuristic Solution
• Experimental Evaluation
• Conclusions and Future work
Conclusions and Future work

• Conclusions:
  – Our work adopts a hierarchical framework with automatic C-code partition and block-level parallelization
  – Both an MILP-based solution and a heuristic solution are proposed
  – Experimental results obtained from seven real applications show that our approaches are effective

• Future work:
  – Extend the solution to C program with feedback
  – Taking power into consideration
Reference

• [1]-[27] is listed in the paper
• [29] “ITRS roadmap on Design” 2011 Edition
THANK YOU!
MILP-Based Solution: Linearization

- Linearize $x_j y_{i,j}$: $z_{i,j} = x_j y_{i,j}$
  
  $$-M y_{i,j} \leq z_{i,j} \leq M y_{i,j}$$

  $$x_j - M (1 - y_{i,j}) \leq z_{i,j} \leq x_j + M (1 - y_{i,j})$$

- Linearize Equation (1):
  
  $$r_{all} \leq r_{i,j} + M (1 - y_{i,j}) \quad \forall 1 \leq i \leq j \leq N$$

- Linearize Equation (2):
  
  $$r_{i,j} \leq \begin{cases} z_{i,j} / T_{i,j} \\ 1 / \max\{T_{i,j}^{in}, T_{i,j}^{out}\} \end{cases}$$

- Linearize Equation (4):
  
  $$\sum_{i=1}^{i=n} y_{i,n} \leq x_n \leq M \cdot \sum_{i=1}^{i=n} y_{i,n} \quad x_n \in \mathbb{N}, \ y_{i,j} \in \text{binary}$$
Exp.: Validate proposed method (cont’d)

- Min. area or Max. throughput for GSM

<table>
<thead>
<tr>
<th>Objective</th>
<th>Constraints</th>
<th>MILP vs. Heuristic result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$R_{\text{req}}^{-1}$</td>
<td>$A_{\text{req}}$</td>
</tr>
<tr>
<td>MILP</td>
<td>min $a_{\text{all}}$</td>
<td>1000</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MILP</td>
<td>min $a_{\text{all}}$</td>
<td>1600</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MILP</td>
<td>max $r_{\text{all}}$</td>
<td>-</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MILP</td>
<td>max $r_{\text{all}}$</td>
<td>-</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MILP</td>
<td>max $r_{\text{all}}$</td>
<td>-</td>
</tr>
<tr>
<td>Heuristic</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Solutions are effective with various settings