MAPS: An Integrated Framework for MPSoC Application Parallelization

Rainer Leupers, Weihua Sheng

Software for Systems on Silicon
RWTH Aachen University

1st Workshop on Mapping of Applications to MPSoCs
June 16th 2008, St. Goar

Institute for Integrated Signal Processing Systems
Agenda

MAPS - MPSoC Application Programming Studio

- MAPS Project Overview
- MAPS-TCT Framework
- Summary/Outlook
MAPS Project Overview

Multi-App Project

- **App1**
  - C
  - Hard RT
  - DSP
  - || 2, X 3

- **App2**
  - Threaded C
  - Soft RT
  - RISC
  - || 1

- **App3**
  - Mixed
  - No RT
  - No Pref.
  - || 2, X 1

- **App**
  - Specification
  - RT Constraints
  - Pref. PE class
  - Concurrency

**Task Partitioning**

**Spatial, Temporal Mapping**

**Scheduling**

**Coarse Arch. Model**

**Fine Arch. Model**

**MPSoc C Code-gen (CC*, + OS/OSIP services, etc.)**

**Virtual Platform/Real Hardware Board**
A practical MPSoC software development tool suite

- Sequential C (input) → “threaded C” (output)
- Powerful analysis tools for providing rich feedback to the programmers
  - Static dependence analysis
  - Dynamic profiling
- Powerful clustering method for extracting coarse-grain parallelism
  - Weighted Statement Control Data Flow Graph (WSCDFG): annotates dynamic profiling information on CDFG
  - Coupled Block (CB): subgraph of WSCDFG that is schedulable and tightly coupled by data dependence
  - Constrained Agglomerative Hierarchical Clustering (CAHC): iterative clustering for building coarser graphs
Weighted Statement Control/Data Flow Graph (WSCDFG)

- **Definition:** **WSCDFG** is a directed graph defined by \( G = (V, CE, DE, CW, DW, N) \), where:
  - \( V \): IR statement nodes
  - \( CE \): set of control flow edges
  - \( DE \): set of data flow edges
  - \( CW \): weights (count) of control edges
  - \( DW \): weights (amount of data, e.g. bytes) of data edges
  - \( N \): weight of IR statement nodes (cost table)

\[
\begin{align*}
n_1 &= \text{(Cost of add)} \times 8 \\
n_2 &= \text{(Cost of jump)} \times 8 \\
n_3 &= \text{(Cost of assign)} \times 4 \\
n_4 &= \text{(Cost of assign)} \times 4 \\
n_5 &= \text{(Cost of assign)} \times 4 \\
n_6 &= \text{(Cost of assign)} \times 4 \\
\end{align*}
\]

\[
\begin{align*}
cw_{12} &= 8 \\
cw_{13} &= 4 \times 4 \\
cw_{24} &= 4 \\
cw_{26} &= 4 \\
cw_{34} &= 4 \\
cw_{41} &= 3 \\
cw_{43} &= 5 \\
\end{align*}
\]

- Control-flow edge: \( v_4 \rightarrow v_3 \)
- Data-flow edge: \( v_3 \rightarrow v_2 \)
- Control-flow edge: \( v_1 \rightarrow v_4 \)

The graph represents the flow of control and data between statements, with weighted edges indicating the cost and data flow associated with each.
Parallelism Discovery: Granularity - Coupled Block (CB)

- Coarse-grained Task Parallelism in C
- Based on Compiler IR (Intermediate Representation)
- Suitable Granularity of Codes as Atomic code blocks
  \[ \text{CB (Coupled Block)} \]

CB Design Criteria
- Tightly coupled by data-dependence
- Schedulable

\[
\begin{align*}
    v_2: & \text{ if (d) goto } v_3; \\
    v_3: & \text{ d = b; } \\
    v_4: & \text{ d = a; }
\end{align*}
\]

Weighted Statement Control/Data-flow Graph (WSCDFG)

\[
\forall v_i \in V', \quad \frac{w_1 \cdot \sum_{c_{w_{ij}} \in CW} c_{w_{ij}} + w_2 \cdot \sum_{d_{w_{ij}} \in DW} d_{w_{ij}}}{D(v_1, V')} > T
\]

CB Example in WSCDFG
Parallelism Discovery: CB Generation – CAHC Algorithm

- Optimal generation of CB: Heuristics needed
- Based on Density Boundary Scan algorithm (DBSCAN) for knowledge discovery in data bases

**Constrained Agglomerative Hierarchical Clustering – CAHC**
- Constrained: maintain schedulability of CBs
- Hierarchical: several clustering levels with different granularities
- Agglomerative: build coarser graphs iteratively
MAPS-TCT Framework

MAPS@RWTH Aachen

Parallelization

Sequential C Code

Parallelized C Code

Performance Evaluation

MP-Platform (TCT)

TCT@Tokyo Tech
Tightly-Coupled Thread (TCT) Programming Model

Seamless transition from *sequential C codes*

```
void JPEGtop(){
  for(i = 0; i < imageSizeYPadding;){
    for(ii = 0; ii < 8; ii ++){
      ReadOneLine(fp, i ++);
      // row 0: RGB => Y0/Y1,Cb0,Cr0
      ReadOneLine(fp, i ++);
      // row 1: RGB => Y0/Y1,Cb0,Cr0
      THREAD(Dsamp) {
        DownsampleCbCr(i);
      // Cb0,Cr0 => Cb,Cr
      }
    }
    THREAD(BLKcore) {
      // call the core functions
      int nR = (i - 8 >= imageSizeY); // 2nd row is dummy
      for(j = 0; j < imageSizeX; j += 16){
        int nC = (j + 8 >= imageSizeX); // 2nd col is dummy
        THREAD(Y0) {
          // process Y components
          BLK8x8(&Y0[j],0,&DCy,&state,0);
          BLK8x8(&Y0[j+8],0,&DCy,&state,nC);
        }
        THREAD(Y1) {
          // process Y components
          BLK8x8(&Y1[j],0,&DCy,&state,nR);
          BLK8x8(&Y1[j+8],0,&DCy,&state,nC+nR);
        }
        THREAD(C) {
          // process Cb/Cr components
          BLK8x8(&Cb[j>>1],1,&DCcb,&state,0);
          BLK8x8(&Cr[j>>1],1,&DCcr,&state,0);
        }
      }
    }
  }
}
```

```
void ReadOneLine(...){
  THREAD(cY) {
  }
  THREAD(cCb) {
  }
  THREAD(cCr) {
  }
  THREAD(BUF) {
  }
}
```

```
void BLK8x8(...){
  ...
  THREAD(Q) {
  }
  THREAD(E) {
  ...
    THREAD(W) {
    }
  }
}
```

- Simply insert "thread scope" directly on the sequential C code
- No need to specify concurrent semantics
- Preserves original sequential structure
TCT-MPSoC Prototype Chip

- **TCT Coprocessor (TCoP):** 6-PE array @ 100 MHz
  - Full crossbar interconnect
  - Dedicated comm. module in each PE
- **Host RISC core:** JANUS200 @ 200 MHz (ARMv4T ISA)
  - Can be configured as the 7th PE on the PE array interconnect
Parallelization Experiment: JPEG Encoder

Speedup & efficiency

<table>
<thead>
<tr>
<th>Step</th>
<th>Speedup</th>
<th>No. of PEs</th>
<th>Parallel Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.61x</td>
<td>16</td>
<td>22.58%</td>
</tr>
<tr>
<td>2</td>
<td>5.48x</td>
<td>17</td>
<td>32.3%</td>
</tr>
<tr>
<td>3</td>
<td>5.48x</td>
<td>16</td>
<td>34.3%</td>
</tr>
<tr>
<td>manual</td>
<td>9.43x</td>
<td>19</td>
<td>49.6%</td>
</tr>
</tbody>
</table>
Summary: MAPS work-in-progress

Multi-App Project

App1
- C
- Hard RT
- DSP
- || 2, X 3

App2
- Threaded C
- Soft RT
- RISC
- || 1

App3
- Mixed
- No RT
- No Pref.
- || 2, X 1

App
- Specification
- RT Constraints
- Pref. PE class
- Concurrency

Task Partitioning
Scheduling
Spatial, Temporal Mapping

MPSoc C Code-gen (CC*, + OS/OSIP services, etc.)

Virtual Platform/Real Hardware Board
Outlook

- Multi-App Model
- Better Arch. Model
- OSIP Scheduling
- MPSoC Code-gen

Multi-App Project

App 1
- C
- Hard RT
- DSP
- || 2, X 3

App 2
- Threaded C
- Soft RT
- RISC
- || 1

App 3
- Mixed
- No RT
- No Pref.
- || 2, X 1

App
- Specification
- RT Constraints
- Pref. PE class
- Concurrency

Task Partitioning

Spatial, Temporal Mapping

MPSoC Code-gen (CC*, + OS/OSIP services, etc.)

Virtual Platform/Real Hardware Board
Thank you!