C for Process Networks

Stefan Schürmans, Weihua Sheng, Anastasia Stulova, Jeronimo Castrillon

3rd Workshop on Mapping of Applications to MPSoCs, Schloss Rheinfels, June 29th 2010
Outline

Introduction

MPSoc Application Programming Studio

C for Process Networks

Generated Code

Targets & Applications

Summary & Outlook
Embedded Systems use MPSoCs

- **Embedded Systems Requirements**
  - High Computational Performance
  - High Energy Efficiency

Fujitsu, MPSOC 2009

SoC Consumer Portable Design Complexity Trends (ITRS 2007)

⇒ **Parallel Hardware: MPSoCs**
Programming Languages for Embedded Systems

- Dominant Embedded Systems Programming Language: C
  - Existing legacy software
  - New projects
    - Programmers use language they know

Source: Embedded.com (EE Times)
Mismatch between HW and SW

Parallel Hardware

Sequential Software

- RISC
- VLIW
- ASIP
- DSP
- MPSoC

Mismatch

Possible Solutions:
- Partition sequential software
- Write parallel software e.g. Process Networks (PN)

Tool Support required
MAPS (MPSoC Application Programming Studio) Approach

Applications Specification
• Support both sequential C or parallel models (process networks)
• Multi-App Scenarios (Real-Time constraints, interference)

Application Partitioning
• High-level transformations to expose more parallelism
• Semi-automatic Parallelism extraction

MPSoC Backends
• Code generation for MPSoCs
• Native Hardware/Virtual Platform System Simulation

Efficient Mapping/Scheduling
• Spatial/Temporal Mapping
• Efficient/Fast Multi-task Run-time processing
MAPS Tool Flow

Sequential Application (C)

Process Network Application (Language?)

Arch. Model

Applications

Multi-applications

- App. concurrency graph (ACG)
- Real-time constraints
- PE class preferences

Analysis

- Parsing
- Profiling & Tracing
- Seq. Partitioning

Mapping & Scheduling

- Mapping
- Scheduling
- Performance Estimation

Multi-app. Analysis

Run-time Info

- OS Primitives
- Low level APIs

Execution

- Real Silicon / Virtual Platforms

MVP simulator
- **Process Network (PN)**
  - C code inside processes

  *Merge C + PN into a single language*

- **C for Process Networks (CPN)**
  - Minimal language extension to C
  - New keywords
    - Processes
    - Channels
    - Channel Accesses
CPN Example: Fibonacci Numbers

```plaintext
__PNsdf  Add __PNIn(int u, int v)
    __PNOut(int sum) {
        __PNLoop { sum = u + v; }
    }

__PNkpn  Delay __PNIn(int i)
    __PNOut(int o)
    __PNParam(int first) {
        __PNOut(o) { o = first; }
        while (1) {
            __PNIn(i) __PNOut(o) { o = i; }
        }
    }

__PNsdf  Print __PNIn(int wnd : 2) {
    __PNLoop {
        printf("%d %d\n", wnd[0], wnd[1]);
    }
}

__PNChannel int a, b, c;

__PNProcess add = Add __PNIn(b, c) __PNOut(a);
__PNProcess delay1 = Delay __PNIn(a) __PNOut(b) __PNParam(1);
__PNProcess delay2 = Delay __PNIn(b) __PNOut(c) __PNParam(0);
__PNProcess print = Print __PNIn(c);
```
Outline

- Introduction
- MPSoC Application Programming Studio
- C for Process Networks
- Generated Code
- Targets & Applications
- Summary & Outlook
Translation of CPN to Message Passing

CPN Code

```cpp
void fft(int i[64], int o[64]);

__PNkpn FFT __PNin(int i : 64)
    __PNout(int o : 64) {
        __PNin(i) __PNout(o) {
            int i[64], o[64];
            recv(ch_i, i, 64);
            fft(i, o);
            send(ch_o, o, 64);
            fft(i, o);
        }
    }
```

C Code for Message Passing

```c
... fft(i, o); ...
... fft(i, o); ...
... fft(i, o); ...
```

Diagram:

```
ch_i  recv  fft(i, o)  send  ch_o
     |     |       |     |
     |     |       |     |
     |     |       |     |
     |     |       |     |
```
Translation of CPN for Shared Memory Systems

- **FIFO Channel in global Shared Memory**
  - Cyclic buffer
    - Part filled with data, free part
  - Windowed Access (WA)
    - Parts being written and read
  - Multiple Readers (MR)
    - Replication of reader information
Translation of CPN for Shared Memory Systems

CPN Code

```cpp
void fft(int i[64], int o[64]);

__PNkpn FFT __PNin(int i : 64)
    __PNout(int o : 64) {
        __PNin(i) __PNout(o) {
            fft(i, o);
        }
    }
}

C Code

for Shared Memory FIFO

```
- **Pthreads**
  - Execution of application on host

- **MAPS Virtual Platform (MVP)**
  - Early Exploration of Mapping & Scheduling

- **CoWare/Synopsys ATM**
  - Abstract Virtual Platform
  - Based on Virtual Processing Unit (VPU)

<table>
<thead>
<tr>
<th>Target</th>
<th>Shared Memory</th>
<th>Process</th>
<th>Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pthreads</td>
<td>Yes</td>
<td>Host Thread</td>
<td>Sh. Mem. (WA + MR)</td>
</tr>
<tr>
<td>MVP</td>
<td>Yes</td>
<td>MVP Task</td>
<td>Sh. Mem. (WA + MR)</td>
</tr>
<tr>
<td>ATM</td>
<td>No</td>
<td>VPU Task</td>
<td>System C FIFO</td>
</tr>
</tbody>
</table>
Task Graph

Source \rightarrow R2B \rightarrow DCT \rightarrow Q \rightarrow ZZE \rightarrow RLE \rightarrow Encoder

Source \rightarrow R2B \rightarrow DCT \rightarrow Q \rightarrow ZZE \rightarrow RLE \rightarrow Encoder

Sink \rightarrow B2R \rightarrow DCT \rightarrow IQ \rightarrow ZZD \rightarrow Decoder

Sink \rightarrow B2R \rightarrow DCT \rightarrow IQ \rightarrow ZZD \rightarrow Decoder

Parameters

Input

Output

18

Difference (amplified 16x)

Same Results on all 3 Targets

JEVP Encoder / Decoder
MMSE Processing in MIMO Receiver

Task Graph

Input

Test Vectors

Reference

Expected Results

Output

Results

*) on all 3 Targets
Outline

Introduction

MPSoC Application Programming Studio

C for Process Networks

Generated Code

Targets & Applications

Summary & Outlook
Summary & Outlook

- **MPSoC programming requires tool support**
  - MAPS: MPSoC Application Programming Studio

- **CPN to write PN applications**
  - Minimal extension to C
  - Concise and expressive

- **Code Generation**
  - Portable to different targets
    - Use special features of target → Efficiency

- **Outlook:**
  - Code generation for real hardware platforms
  - Evaluate additional applications
Thank You!

Questions?