Compilation of Parametric Dataflow Applications for Software-Defined-Radio-Dedicated MPSoCs

*DREAM seminar*

Mickaël Dardaillon

Research Intern with NOKIA Technologies

January 27th, 2015
EVOLUTION OF TELECOMMUNICATION PROTOCOLS

- **Real-time control**
- **NextGen media**
- **Monitoring & sensing**
- **Multimedia**
- **Mail**
- **Text**
- **Voice**

**Push & pull of technology**

**1G**

**2G**

**3G**

**4G**

**5G**

- **Tactile**
  - 1ms
- **Visual**
  - 10ms
- **Audio**
  - 100ms

**Gigabit experience**

**1ms**

**10ms**

**100ms**
4G LTE-ADVANCED: DOWNLINK

1 frame (10 ms)
1 sub-frame (1 ms)
4G LTE-Advanced: Downlink

- 1 frame (10 ms)
- 1 sub-frame (1 ms)

- 14 OFDM Symbols
- 2048 subcarriers (20 MHz)

- MIMO: 4 × 2 antennas
- LTE throughput: 1.4 Gbps
- LTE-Advanced: 7 Gbps
- Latency: 2 ms
- Power budget: 500 mW
4G LTE-ADVANCED: DOWNLINK

- MIMO: $4 \times 2$ antennas
- LTE throughput: 1.4 Gbps
- LTE-Advanced: 7 Gbps
- Latency: 2 ms
- Power budget: 500 mW
MAGALI SDR

LTE demonstrator
[Clermidy et al., 09]
Power consumption: 231mW
Magali SDR

LTE demonstrator

[Clermidy et al., 09]

Power consumption: 231mW
Magali SDR

LTE demonstrator
[Clermidy et al., 09]
Power consumption: 231mW
LTE demonstrator

[Clermidy et al., 09]

Power consumption: 231mW
Magali SDR

LTE demonstrator
[Clermidy et al., 09]
Power consumption: 231mW
PROBLEM STATEMENT

– How should we program a Cell processor?
PROBLEM STATEMENT

– How should we program a Cell processor?
– Any way you want!

How to **program** and **compile** a telecommunication protocol to an heterogeneous MPSoC?
# Outline

**Context**

Programming Model for SDR
Dataflow Model of Computation

Dataflow Refinement and Buffer Verification
Mapping and Scheduling
Micro-Scheduling

Experimentations on Magali
Code Generation
Experimental Results

Perspectives
# State of the Art in SDR Programming

## Imperative Concurrent

<table>
<thead>
<tr>
<th>Platform</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExoCHI [Wang et al., 07]</td>
<td>OpenMP + C</td>
</tr>
<tr>
<td>BEAR [Derudder et al., 09]</td>
<td>Matlab + C</td>
</tr>
</tbody>
</table>

## Dataflow

<table>
<thead>
<tr>
<th>Platform</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulink</td>
<td>Python + C</td>
</tr>
<tr>
<td>LabView</td>
<td>XML + C</td>
</tr>
<tr>
<td>GNU Radio</td>
<td>UML</td>
</tr>
<tr>
<td>RVC-CAL [Lucarz et al., 08]</td>
<td>XML + C</td>
</tr>
<tr>
<td>DiplodocusDF [Gonzalez-Pina et al., 12]</td>
<td>UML</td>
</tr>
<tr>
<td>MAPS [Castrillon et al., 13]</td>
<td>C like</td>
</tr>
</tbody>
</table>
STATIC DATAFLOW (SDF) [Lee et al., 87]
Phase Approach with Static Dataflow
Dynamic Dataflow (DDF) [Buck, 93]

Kahn Process Network (KPN) [Kahn, 74]
Dynamic Dataflow (DDF) [Buck, 93]

Scenario Aware DataFlow (SADF) [Theelen et al., 06]
Mode Controlled DataFlow (MCDF) [Moreira et al., 12]
Schedulable Parametric DataFlow (SPDF) [Frade et al., 12]
Parameterized and Interfaced dataflow Meta-Model (PiMM) [Desnos et al., 13]
Boolean Parametric DataFlow (BPDF) [Bebelis et al., 13]
Kahn Process Network (KPN) [Kahn, 74]
Schedulable Parametric DataFlow (SPDF)

[Fradet et al., 12]

- Model of Computation
- Analysis
- Quasi-Static Scheduling
**Schedulable Parametric DataFlow (SPDF)**

![Diagram of Schedulable Parametric DataFlow (SPDF)]

[Fradet et al., 12]

- Model of Computation
- Analysis
- Quasi-Static Scheduling
**SDR Programming Model**

- Propose SPDF for SDR
- C++ input format

**Front End**

- Based on LLVM framework
- Derived from SystemC analysis [Marquet et al., 10]
- Static graph structure
OUTLINE

CONTEXT

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

PERSPECTIVES
SPDF Mapping

- **Src** connected to **Decod₁** via **dma1**
- **Decod₁** connected to **Decod₂** via **p**
- **Decod₂** connected to **Ctrl set p[1]** via **p**
- **Ctrl set p[1]** connected to **Sink** via **10**
- **Src** connected to **DEMOD** via **10**
- **DEMOD** connected to **ARM** via **demod**
- **ARM** connected to **DMA** via **arm**
- **DMA** connected to **DEMOD** via **dma1**
- **DMA** connected to **DMA** via **dma2**
SPDF Quasi-Static Scheduling [Fradet et al., 12]

\[ S(dma1) = (\text{Src}) \]
\[ S(arm) = (\text{Ctrl}; \text{set}(p)) \]
\[ S_demod = (\text{Decod}_1; \text{get}(p); (\text{Decod}_2)^{10}) \]
\[ S(dma2) = (\text{get}(p); (\text{Sink})^p) \]
SPDF Symbolic Execution

\[ S(dma1) = (\text{Src}) \]
\[ S(arm) = (\text{Ctrl}; \text{set}(p)) \]
\[ S(demod) = (\text{Decod}_1; \text{get}(p); (\text{Decod}_2)^{10}) \]
\[ S(dma2) = (\text{get}(p); (\text{Sink})^p) \]
SPDF Buffer Sizing

Problem: overestimates buffer size

e.g. Magali
- FFT size: 2048
- Buffer size: 16
SPDF Model Refinement

Idea: model each individual data communication

Micro-Scheduling

```
class Src {
    void compute() {
        // [...]
        out[1].push(ctrl, 10);
        for (int i=0; i<10; i++)
            out[2].push(data[i], 10);
    }
}
```
Micro-Scheduling: an Example

\[
\begin{align*}
\mu S(Src) &= \left(\text{push}_{Src,D_1}(10); \text{push}_{Src,D_2}(10)^{10}\right) \\
\mu S(D_2) &= \left(\text{pop}_{Src,D_2}(10); \text{push}_{D_2,Sink}(p)\right) \\
\mu S(Sink) &= \left(\text{pop}_{D_2,Sink}(1)^{10}\right)
\end{align*}
\]
BUFFER SIZING VERIFICATION

How to verify buffer sizes using micro-schedules?
Buffer Sizing Verification

How to verify buffer sizes using micro-schedules?

Proposed Verification Method
- Based on Model Checking
- Derived from buffer minimization [Geilen et al., 05]

Model
- Schedule
- Buffer sizes
  - Micro-Schedule
  - Parameter values

Model Checker
- SPIN
- Check for deadlocks
MICRO-SCHEDULING IMPLEMENTATION

Front End

- PaDaF (C++)
- C++ Front End (CLang)
- LLVM IR
- Graph Construction
- Graph + LLVM IR

Back End

Micro-Scheduling
- SPDF model refinement
- Sequential communications

Buffer Verification
- Model checking
OUTLINE

CONTEXT

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

PERSPECTIVES
CODE GENERATION

Graph + LLVM IR

OFDM

DEMOD

TURBO

DSP

DMA

ARM code generation

Magali code

ARM code generation

Control code (C)

ARM

Magali code

(ASM)
**Benchmarks using LTE**

**OFDM: compilation**

```
Src    7168 1024
    FFT  1024 1024
    Defram  600 4200
    Sink
```

**Demodulation: communications**

```
Src    1200 1200 1200
    Demap    900 900 900
    Word Deinter 900
    Sink
```

```
Src    1200
    Bit Deinter 900
    Depunct 1353 1353
    Turbo Decod 57
    Sink
```
BENCHMARKS USING LTE

Parametric Demodulation: parameter

[Diagram of a network flow showing various components such as 'Src', 'Bit Deinter', 'Depunct', 'Turbo Decod', 'Word Deinter', 'Sink', with flow rates and parameters like 'dma1', 'dma2', 'dma3', 'dma4', and 'arm'.]
RESULTS: ESTIMATED DEVELOPMENT TIME

Compiler Development

- Front-End: 4 man-months
- Back-End: 8 man-months

<table>
<thead>
<tr>
<th>Application</th>
<th>Native C / ASM (#lines)</th>
<th>(hours)</th>
<th>PaDaF C++ (#lines)</th>
<th>(hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDM</td>
<td>150 / 200</td>
<td>40</td>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>Demodulation</td>
<td>300 / 600</td>
<td>160</td>
<td>160</td>
<td>4</td>
</tr>
<tr>
<td>Param. Demod.</td>
<td>500 / 800</td>
<td>480</td>
<td>260</td>
<td>8</td>
</tr>
</tbody>
</table>

Takeaway Message:
Reduces development time
RESULTS: BUFFER VERIFICATION TIME

Evaluation framework
- 2.4 GHz Intel Core i5, 8 GB RAM, OS X 10.9.2.
- SPIN Model Checker

<table>
<thead>
<tr>
<th>Application</th>
<th>States</th>
<th>Transitions</th>
<th>Exec. Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDM</td>
<td>$1.28 \times 10^4$</td>
<td>$2.56 \times 10^4$</td>
<td>0.1</td>
</tr>
<tr>
<td>Demodulation</td>
<td>$2.12 \times 10^6$</td>
<td>$1.07 \times 10^7$</td>
<td>9</td>
</tr>
<tr>
<td>Param. Demod.</td>
<td>$6.07 \times 10^7$</td>
<td>$2.22 \times 10^8$</td>
<td>199</td>
</tr>
</tbody>
</table>

Takeaway Message:
Reduces development time, improves verification
RESULTS: EXECUTION TIME

Evaluation framework

- SystemC TLM based on 65 nm CMOS implementation
- ARM code run on QEMU Virtual Machine

<table>
<thead>
<tr>
<th>Application</th>
<th>Native ($\mu s$)</th>
<th>Generated ($\mu s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDM</td>
<td>149</td>
<td>168 (+13%)</td>
</tr>
<tr>
<td>Demodulation</td>
<td>180</td>
<td>283 (+57%)</td>
</tr>
<tr>
<td>Param. Demod.</td>
<td>419</td>
<td>558 (+33%)</td>
</tr>
</tbody>
</table>

Takeaway Message:
Reduces development time, improves verification
EXECUTION MODEL

Phase Approach

Distributed
EXECUTION MODEL

Phase Approach

Distributed
RESULTS: EXECUTION TIME

Evaluation framework

► SystemC TLM based on 65 nm CMOS implementation
► ARM code run on QEMU Virtual Machine

<table>
<thead>
<tr>
<th>Application</th>
<th>Native ($\mu s$)</th>
<th>Generated ($\mu s$)</th>
<th>Optimized ($\mu s$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFDM</td>
<td>149</td>
<td>168 (+13%)</td>
<td>149 (+0%)</td>
</tr>
<tr>
<td>Demodulation</td>
<td>180</td>
<td>283 (+57%)</td>
<td>180 (+0%)</td>
</tr>
<tr>
<td>Param. Demod.</td>
<td>419</td>
<td>558 (+33%)</td>
<td>288 (-31%)</td>
</tr>
</tbody>
</table>

Takeaway Message:

Reduces development time, improves verification, maintains performances
**Back End Implementation**

- **Front End**
  - PaDaF (C++)
  - C++ Front End (CLang)
  - LLVM IR
  - Graph Construction
  - Graph + LLVM IR

- **Back End**
  - Mapping
  - Scheduling
  - Buffer Verification (SPIN)
  - Code Generation
  - MPSoC Code (ASM)

**Magali Support**
- Computation
- Communication
- Control

**LTE Experimentation**
- Performance close to native
- Buffer verification
- Central controller
OUTLINE

CONTEXT

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

PERSPECTIVES
On dataflow programming

- Compiler
- Runtime
**PERSPECTIVES**

On dataflow programming

On heterogeneous MPSoC
- Future of dedicated platforms
- Development on such platforms
PERSPECTIVES

On dataflow programming

On heterogeneous MPSoC

Publications

- Survey: [Dardaillon et al., IWCMC 12]
- Compilation flow: [Dardaillon et al., CASES 14]

- INSA-Lyon, CITI-Inria
  - Tanguy Risset
  - Kevin Marquet

- CEA Grenoble
  - Jérôme Martin
  - Henri-Pierre Charles