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

*PhD work of Mickael Dardaillon*

Mickaël Dardaillon, Kevin Marquet (Citi), Tanguy Risset (Citi), Jérôme Martin (Cea Leti), Henri-Pierre Charles (CEA List)

June 24th, 2016
E V O L U T I O N  O F  T E L E C O M M U N I C A T I O N  P R O T O C O L S

<table>
<thead>
<tr>
<th>Year</th>
<th>2G</th>
<th>3G</th>
<th>Wi-Fi</th>
<th>Bluetooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2005</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data rate (kbps):
- 2G: 10
- 3G: 1000
- Wi-Fi: 1000000
- Bluetooth: 100

Year:
- 1990
- 1995
- 2000
- 2005
- 2010

Protocols:
- 2G
- 3G
- Wi-Fi
- Bluetooth

Evolution of telecommunication protocols:
- 2G
- 3G
- Wi-Fi
- Bluetooth
EVOlUTION OF TEleCOMMUNICATION PROTOCOLS

<table>
<thead>
<tr>
<th>Year</th>
<th>Data Rate (kbps)</th>
<th>Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>1990</td>
<td>10</td>
<td>2G</td>
</tr>
<tr>
<td>1995</td>
<td>100</td>
<td>2G</td>
</tr>
<tr>
<td>2000</td>
<td>1000</td>
<td>3G</td>
</tr>
<tr>
<td>2005</td>
<td>10000</td>
<td>3G</td>
</tr>
<tr>
<td>2010</td>
<td>100000</td>
<td>4G</td>
</tr>
</tbody>
</table>

- Bluetooth
- Wi-Fi
- Bluetooth
4G LTE-Advanced: Downlink

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

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
</table>
4G LTE-ADVANCED: DOWNLINK

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

- 2048 subcarriers (20 MHz)
- 14 OFDM Symbols
- 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

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

2048 subcarriers (20 MHz)
14 OFDM Symbols

Control
Data
User 1
User 2
User 3
**CONTEXT**

What is an SDR software?
**CONTEXT**

What is an SDR software?

Baseband processing in software
- ZigBee
- ... 
- LTE Advanced

Constraints
- Computing power \( \sim \) GFLOPS
- Reconfiguration time \(< 100\,\mu s\)
- Consumption \(< 500\,mW\)

Architecture independent SDR software
CONTEXT

What is an SDR software?
What is an SDR hardware platform?
CONTEXT

What is an SDR software?
What is an SDR hardware platform?

▸ EVP16?
  ▸ VLIW
  ▸ Vector Processor
**CONTEXT**

What is an SDR software?
What is an SDR hardware platform?

- EVP16?
  - VLIW
  - Vector Processor
- SB3500?
  - DSP
  - Control Processor
What is an SDR software?
What is an SDR hardware platform?

- EVP16?
  - VLIW
  - Vector Processor
- SB3500?
  - DSP
  - Control Processor
- Magali?
  - Configurable Units
  - NoC
CONTEXT

What is an SDR software?
What is an SDR hardware platform?

- EVP16?
  - VLIW
  - Vector Processor
- SB3500?
  - DSP
  - Control Processor
- Magali?
  - Configurable Units
  - NoC
- ...

⇒ No unified hardware platform model for SDR.

Problem Statement: how to program and compile a telecommunication protocol to an heterogeneous MPSoC?
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
Magali SDR

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

CONTEXT
SDR software?

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation
Input Format

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

CONCLUSION
## 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></td>
</tr>
<tr>
<td>GNU Radio</td>
<td></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) [Fradet 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)

[Fradet et al., 12]
- Model of Computation
- Analysis
- Quasi-Static Scheduling
PARAMETRIC DATAFLOW FORMAT (PaDaF)

Actor specification

class Decod: public Actor{
    PortIn<int> in;
    PortOut<int> out;
    ParamIn p;
    void compute() {
        [...]  
        out.push(res, p);
    }
}

Graph specification

Src src;
Decod decod[2];
[...]
for(int i=0; i<2; i++) {
    decod[i].in <= src.out[i];
}
**Front End Implementation**

**SDR Programming Model**
- Propose SPDF for SDR
- C++ input format
- [IWCMC 12, IGI 14]

**Front End**
- Based on LLVM framework
- Derived from SystemC analysis [Marquet et al., 10]
- Static graph structure
- [CASES 14]
OUTLINE

CONTEXT
SDR software?

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation
Input Format

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

CONCLUSION
SDF Scheduling

Iteration vector:

\( (\text{Src}; \ \text{Decod}_1; \ \text{Ctrl}; \ (\text{Decod}_2)^{10}; \ (\text{Sink})^5) \)
SPDF Scheduling [Fradet et al., 12]

Iteration vector:

\[
\left( \text{Src}; \text{Decod}_1; \text{Ctrl}; (\text{Decod}_2)^{10}; (\text{Sink})^p \right)
\]
SPDF Mapping
SPDF Mapping
SPDF Quasi-Static Scheduling

\[ S(dma1) = (Src) \]
\[ S(arm) = (Ctrl; set(p)) \]
\[ S(demod) = (Decod_1; get(p); (Decod_2)^{10}) \]
\[ S(dma2) = (get(p); (Sink)^p) \]
SPDF Symbolic Execution

\[
S(dma1) = (Src)
\]
\[
S(arm) = (Ctrl; set(p))
\]
\[
S(demod) = (Decod_1; get(p); (Decod_2)^{10})
\]
\[
S(dma2) = (get(p); (Sink)^p)
\]
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

```
Src::compute() {
    [...]
    out[1].push(ctrl, 10);
    for(int i=0; i<10; i++)
        out[2].push(data[i], 10);
}
```
**MICRO-SCHEDULING: AN EXAMPLE**

\[ \mu_S(Src) = \left( push_{Src,D_1}(10); push_{Src,D_2}(10)^{10} \right) \]

\[ \mu_S(D_2) = \left( pop_{Src,D_2}(10); push_{D_2,Sink}(p) \right) \]

\[ \mu_S(Sink) = \left( pop_{D_2,Sink}(1)^{10} \right) \]
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**
- Mapping
- Scheduling
- Buffer Verification (SPIN)

**Micro-Scheduling**
- SPDF model refinement
- Sequential communications

**Buffer Verification**
- Model checking
- Model generation
- [CASES 14]
OUTLINE

CONTEXT
SDR software?

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation
Input Format

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

CONCLUSION
**CODE GENERATION**

- **Graph + LLVM IR**
- **OFDM**
- **DEMOD**
- **TURBO**
- **DSP**
- **DMA**
- **ARM**

**Code Generation**

- **communication code generation**
- **control code generation**

**Magali code (ASM)**
Benchmarks using LTE

OFDM: compilation

Demodulation: communications
BENCHMARKS USING LTE

Parametric Demodulation: parameter

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>Native (hours)</th>
<th>PaDaF C++ (#lines)</th>
<th>PaDaF (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**

```
<table>
<thead>
<tr>
<th></th>
<th>arm</th>
<th>dma1</th>
<th>ofdm1</th>
<th>dma3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>25 µs</td>
<td>37 µs</td>
<td>16 µs</td>
<td>21 µs</td>
</tr>
</tbody>
</table>
```

**Distributed**

```
<table>
<thead>
<tr>
<th></th>
<th>arm</th>
<th>dma1</th>
<th>ofdm1</th>
<th>dma3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>25 µs</td>
<td>74 µs</td>
<td>25 µs</td>
<td>23 µs</td>
</tr>
</tbody>
</table>
```
## 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 (μs)</th>
<th>Generated (μs)</th>
<th>Optimized (μs)</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
- [ComPAS 14, CASES 14]
OUTLINE

CONTEXT
SDR software?

PROGRAMMING MODEL FOR SDR
Dataflow Model of Computation
Input Format

DATAFLOW REFINEMENT AND BUFFER VERIFICATION
Mapping and Scheduling
Micro-Scheduling

EXPERIMENTATIONS ON MAGALI
Code Generation
Experimental Results

CONCLUSION
CONCLUSION

▶ A complete research experience between Telecommunication, SoC programming and Compilation.
▶ Mickael Dardaillon is currently working at National Instruments (Austin) on the compilation Parametric Dataflow in LabView-FPGA (mickael.dardaillon@gmail.com)
▶ CEA has stopped the activities on Magali (and is, in general, less involved in telecommunication chips because of ST-microelectronics strategy).
▶ There are many open questions:
  ▶ How to program FPGA-based SDR machines?
  ▶ How to handle fast dynamic reconfiguration in heterogenous MP-SoC?
QUESTION?

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)

Programming Model
- PaDaF
- Front End

Micro-Scheduling
- Buffer verification
- Model checking

Experimentations
- Magali Back End
- LTE experiments
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

On data manipulation
  - 50% of telecom. protocol
  - Complexity abstraction
PERSPECTIVES

On dataflow programming

On heterogeneous MPSoC

On data manipulation