Platform-Agnostic Dataflow-to-Hardware Design Flow for Reconfigurable Systems

Francesca Palumbo\textsuperscript{1}, Claudio Rubattu\textsuperscript{1,2}, \textbf{Carlo Sau}\textsuperscript{3}, Luigi Raffo\textsuperscript{3} and Maxime Pelcat\textsuperscript{2}

\textsuperscript{1}University of Sassari, IDEA Lab
\textsuperscript{2}University of Rennes, INSA
\textsuperscript{3}University of Cagliari, DIEE, EOLAB

Naples, 20-22 June 2018
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems
• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues
• CGRA Composition and Management
  • High Level Synthesis
• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA
• Final Remarks
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems

• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues

• CGRA Composition and Management
  • High Level Synthesis

• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA

• Final Remarks
Embedded Systems (real-time computing systems with a dedicated functionality), often working in constrained and portable contexts, are pervading the market.

Source: Transparency Market Research
Embedded Systems (real-time computing systems with a dedicated functionality), often working in constrained and portable contexts, are pervading the market.

High performance required to achieve real-time behaviour. Due to the constrained environment and to the portability an extreme execution efficiency (resources, power consumption) is mandatory.

source: Transparency Market Research
Embedded Systems are becoming connected and capable of interacting with Environment, the User and the System itself, to adapt their behaviour according to changing requirements.

source: CISCO ISBG

![Connected Devices Chart]

<table>
<thead>
<tr>
<th>Year</th>
<th>Devices per Person</th>
<th>Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010</td>
<td>1.84</td>
<td></td>
</tr>
<tr>
<td>2021</td>
<td>6.58</td>
<td>+258%</td>
</tr>
</tbody>
</table>

source: IC Insights

![Sensor/Actuators Devices Chart]

<table>
<thead>
<tr>
<th>Year</th>
<th>Devices</th>
<th>Growth</th>
</tr>
</thead>
<tbody>
<tr>
<td>2014</td>
<td>12.4 B</td>
<td></td>
</tr>
<tr>
<td>2018</td>
<td>25.5 B</td>
<td>+106%</td>
</tr>
</tbody>
</table>
Embedded Systems are becoming **connected and capable of interacting** with Environment, the User and the System itself, to **adapt** their behaviour according to changing requirements.

Adaptivity pushes farther **FLEXIBILITY** need of the systems that have to provide several working points while, at the same time, exhibiting a **SHORT RESPONSE TIME**.

*source: CISCO ISBG*  
*source: IC Insights*
Emerging needs for modern embedded systems:

- high performance
- execution efficiency
- flexibility
- short response time
Emerging needs for modern embedded systems:
• high performance
• execution efficiency
• flexibility
• short response time
• minimum time to market
Emerging needs for modern embedded systems:

- high performance
- execution efficiency
- flexibility
- short response time
- minimum time to market
Emerging needs for modern embedded systems:

- high performance
- execution efficiency
- flexibility
- short response time
- minimum time to market
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems
• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues
• CGRA Composition and Management
  • High Level Synthesis
• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA
• Final Remarks
Background
CGRA Design From Applications to Architecture

Imperative
High-Level Specifications

```c
void horizontal_filter_line(pixel *dst, pixel *src, int mx, int width)
{
    int i, j;
    int16_t acc;
    const int8_t *filter = ff_hevc_qpel_filters[mx - 1];
    for (j = 0; j < width-FILTER_LENGTH+1; j++) { dst[j] = src[j];
    }
    for (i = 0; i < FILTER_LENGTH; i++) {
        acc += filter[i] * src[j+i];
        dst[j] = av_clip_pixel(acc>>6);
    }
    vertical_filter_line_intermediate(dst, temporary_image, my, stride, width);
}
```
Background
CGRA Design From Applications to Architecture

Functional
High-Level Specifications

computational actor

1:1 applications → architecture

coarse grained architecture

processing unit

2:1 applications → architecture

coarse grained reconfigurable architecture

A → C → D

A → C → E

A → E → D
Background
CGRA Design Open Issues

Coarse Grained Architecture

Applications → Architecture

Computational Actor

Processing Unit

Coarse Grained Reconfigurable Architecture

1:1

N:1
Background
CGRA Design Open Issues

Functional
High-Level Specifications

A → B → C → D

computational actor

1:1

applications → architecture

A → C → D

coarse grained architecture

processing unit

2:1

coarse grained reconfigurable architecture

A → E → D

SB

E → C → D

SB

B → D
Functional
High-Level Specifications

CGRA development flow requires **design automation** to face:
- resource sharing and runtime configuration management
- high-level specification to hardware description language (HDL) mapping
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems
• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues
• CGRA Composition and Management
  • High Level Synthesis
• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA
• Final Remarks
CGRA Composition and Management

Multi Dataflow Composer Tool

Structural Profiler

Dynamic Power Manager

Co-Processor Generator

Constraint Driven Optimisation

Functional Complexity
Time to Market: Design & Mapping Automation

Fast Integration and Prototyping

Power Efficiency

http://sites.unica.it/rpct/
CGRA Composition and Management

MDC front-end

<table>
<thead>
<tr>
<th>SB</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>α</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>β</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>γ</td>
<td>x</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>

multi-dataflow
CGRA Composition and Management

**DATAPATH MERGING PROBLEM**

NP-complete solved with Moreano’s heuristic method:


<table>
<thead>
<tr>
<th></th>
<th>α</th>
<th>β</th>
<th>γ</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>x</td>
<td>x</td>
<td>1</td>
</tr>
</tbody>
</table>
CGRA Composition and Management

- CGRA substrate
- MDC back-end
- HDL components library
- not fully automated
- hardware communication protocol
- sel0
- sel1
- sel2
- configurador
- ID
- CGR substrate
- not fully automated
- HDL components library
- hardware communication protocol
- sel0
- sel1
- sel2
- configurador
- ID
- CGR substrate
CGRA Composition and Management
High Level Synthesis

Xronos HLS

target specific

Synflow

Vivado™ HLS

target independent

Altera® SDK for OpenCL™

functional

Stratus™ HLS

imperative

Catapult®
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems
• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues
• CGRA Composition and Management
  • High Level Synthesis
• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA
• Final Remarks
Results: Functional vs. Imperative HLS for CGRA
Test Case: HEVC Fractional Pixel Interpolators

example N=4
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

- **y1 = c0*x0 + c1*x1 + c2*x2 + c3*x3**
- **y2 = c0*x1 + c1*x2 + c2*x3 + c3*x4**
- **y3 = c0*x2 + c1*x3 + c2*x4 + c3*x5**

Example N=4:

<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>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
</tr>
<tr>
<td>40</td>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
</tr>
<tr>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
</tr>
<tr>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td>61</td>
<td>62</td>
<td>63</td>
</tr>
</tbody>
</table>

→ **Horizontal Filtering (N-1 cols)**
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

- Functional vs. Imperative HLS for CGRA
- Test Case: HEVC Fractional Pixel Interpolators
- Example: N=4
- y1 = c0x0 + c1x1 + c2x2 + c3x3
- y2 = c0x1 + c1x2 + c2x3 + c3x4
- y3 = c0x2 + c1x3 + c2x4 + c3x5

Serial horizontal 8-taps FIR

Horizontally filtered rows
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

Example N=4

\[
y_1 = c_0 x_0 + c_1 x_1 + c_2 x_2 + c_3 x_3 \\
y_2 = c_0 x_1 + c_1 x_2 + c_2 x_3 + c_3 x_4 \\
y_3 = c_0 x_2 + c_1 x_3 + c_2 x_4 + c_3 x_5 \\
z_9 = c_0 y_1 + c_1 y_9 + c_2 y_{17} + c_3 y_{25} \\
z_{10} = c_0 y_2 + c_1 y_{10} + c_2 y_{18} + c_3 y_{26} \\
z_{11} = c_0 y_1 + c_1 y_9 + c_2 y_{17} + c_3 y_{25}
\]
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

![Diagram of HEVC Fractional Pixel Interpolators]

- **Serial Horizontal 8-taps FIR**
- **Horizontally filtered rows**
- **Parallel Horizontal 8-taps FIR**
- **Final normalization step**

**Example N=4**

- $y_1 = c_0 x_0 + c_1 x_1 + c_2 x_2 + c_3 x_3$
- $y_2 = c_0 x_1 + c_1 x_2 + c_2 x_3 + c_3 x_4$
- $y_3 = c_0 x_2 + c_1 x_3 + c_2 x_4 + c_3 x_5$
- $z_9 = c_0 y_1 + c_1 y_9 + c_2 y_17 + c_3 y_25$
- $z_{10} = c_0 y_{10} + c_1 y_{19} + c_2 y_{27} + c_3 y_{35}$
- $z_{11} = c_0 y_1 + c_1 y_9 + c_2 y_{17} + c_3 y_{25}$

- **Horizontal Filtering (N-1 cols)**
- **Vertical Filtering (N-1 rows)**
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

<table>
<thead>
<tr>
<th># tap</th>
<th>energy</th>
<th>quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 (legacy)</td>
<td>😞</td>
<td>☺</td>
</tr>
<tr>
<td>5</td>
<td>😞</td>
<td>☹</td>
</tr>
<tr>
<td>3</td>
<td>☺</td>
<td>😞</td>
</tr>
</tbody>
</table>
Results: Functional vs. Imperative HLS for CGRA
Test Case: HEVC Fractional Pixel Interpolators

INPUT FUNCTIONAL SPECIFICATIONS (CAPH dataflows)

3-tap

5-tap

8-tap

MULTI-FUNCTIONAL DATAFLOW SPECIFICATION (composed by MDC)

switching module
Results: Functional vs. Imperative HLS for CGRA

Test Case: HEVC Fractional Pixel Interpolators

<table>
<thead>
<tr>
<th></th>
<th>REGS</th>
<th>LOGIC</th>
<th>DSP</th>
<th>Latency</th>
<th>Fmax</th>
</tr>
</thead>
<tbody>
<tr>
<td>functional</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MDC+CAPH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>imperative</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vivado HLS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ pragmas</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target: Xilinx XC7VX485T Virtex-7 FPGA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results: Target Specific vs. Independent HLS for CGRA
Test Case: Sobel and Prewitt edge detectors

\[
G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \times A
\]

\[
G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \times A
\]

\[
G = \sqrt{G_x^2 + G_y^2}
\]

\[G \geq 80\]
Results: Target Specific vs. Independent HLS for CGRA
Test Case: Sobel and Prewitt edge detectors
## Results: Target Specific vs. Independent HLS for CGRA
### Test Case: Sobel and Prewitt edge detectors

<table>
<thead>
<tr>
<th>FPGA*</th>
<th>MDC+CAPH</th>
<th>MDC+XRONOS</th>
<th>XRONOS vs CAPH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Altera</td>
<td>Xilinx</td>
<td>Altera</td>
</tr>
<tr>
<td>REG</td>
<td>1484</td>
<td>780</td>
<td>-</td>
</tr>
<tr>
<td>LOGIC</td>
<td>1047</td>
<td>2347</td>
<td>-</td>
</tr>
<tr>
<td>RAM</td>
<td>15</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>DSP</td>
<td>36</td>
<td>36</td>
<td>-</td>
</tr>
<tr>
<td>MAX FREQ [MHz]</td>
<td>105,80</td>
<td>93,69</td>
<td>-</td>
</tr>
<tr>
<td>EXEC TIME [cck]</td>
<td>15340</td>
<td>15340</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASIC**</th>
<th>MDC+CAPH</th>
<th>MDC+XRONOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AREA [kGE]</td>
<td>466,90</td>
<td>-</td>
</tr>
<tr>
<td>Max Freq [MHz]</td>
<td>399.04</td>
<td>-</td>
</tr>
</tbody>
</table>

*target: Xilinx XXX Zynq-7 FPGA  **target: TMSC 45 nm CMOS technology
Results: Target Specific vs. Independent HLS for CGRA
Test Case: Sobel and Prewitt edge detectors
Outline

• Introduction
  • Embedded Moving Toward IoT and CPS
  • Computing Architectures for Embedded Systems
• Background
  • CGRA Design From Applications to Architecture
  • CGRA Design Open Issues
• CGRA Composition and Management
  • High Level Synthesis
• Results
  • Functional vs. Imperative HLS for CGRA
  • Target Specific vs. Independent HLS for CGRA
• Final Remarks
Final Remarks

• Modern **embedded systems**, pushed into the IoT and CPS era, have lots of colliding requirements.

• **Coarse-Grained Reconfigurable Architectures** (CGRAs) could be a suitable solution to meet these requirements.

• The development of CGRAs require **HLS support** to complete design automation
  
  • **Functional HLS outperforms imperative HLS** for this specific kind of architectures

  • **Target specific HLS** is more efficient on its target, but it lacks in code compactness and readability
Thanks To ...

EU Commission for funding the **CERBERO** (*Cross-layer mOdEl-based fRamework for multi-oBjective dEsign of Reconfigurable systems in unceRtain hybRid envirOnments*) project as part of the H2020 Programme under grant agreement No 732105.

**Coordinator:**
Michal Masin (IBM), michaelm@il.ibm.com

**Scientific Coordinator:**
Francesca Palumbo (UniSS), fpalumbo@uniss.it

**Innovation Manager:**
Katiuscia Zedda (Abinsula), katiuscia.zedda@abinsula.com

**Dissemination-Communication Manager:**
Francesco Regazzoni (USI), francesco.regazzoni@usi.ch
Platform-Agnostic Dataflow-to-Hardware Design Flow for Reconfigurable Systems
Francesca Palumbo, Claudio Rubattu, Carlo Sau, Luigi Raffo and Maxime Pelcat

Alghero (Italy) – September 25-30, 2017
Designing Cyber-Physical Systems – From concepts to implementation
Distinguished lecturers: Alberto Sangiovanni-Vincentelli, XXX
Application deadline: July XXth, 2018
http://www.cpsschool.eu/

Naples, 20-22 June 2018