Hardware/Software Co-design for Heterogeneous Multi-core Platforms
Koen Bertels
Editor

Hardware/Software Co-design for Heterogeneous Multi-core Platforms

The hArtes Toolchain

Springer
Preface

The hArtes project\textsuperscript{1} was started as an innovative European project (funded by European Union) aiming at laying the foundations of a new holistic approach for the design of complex and heterogeneous embedded solutions (hardware and software), from the concept to the silicon (or B2B, from the brain to bits). The hArtes stands for “holistic Approach to reconfigurable real time embedded systems”. As defined in the Embedded Systems Chapter of the IST 2005-06 Work Programme the objective of the hArtes project is to “develop the next generation of technologies, methods and tools for modeling, design, implementation and operation of hardware/software systems embedded in intelligent devices. An end-to-end systems (holistic) vision should allow building cost-efficient ambient intelligence systems with optimal performance, high confidence, reduced time to market and faster deployment”.

The hArtes project aims to lay the foundation for a new holistic (end-to-end) approach for complex real-time embedded system design, with the latest algorithm exploration tools and reconfigurable hardware technologies. The proposed approach will address, for the first time, optimal and rapid design of embedded systems from high-level descriptions, targeting a combination of embedded processors, digital signal processing and reconfigurable hardware. The project ended with an important scientific and technical contribution that resulted in more than 150 international publications as well as a spin-off company, BlueBee.\textsuperscript{2}

From the application point of view, the complexity of future multimedia devices is becoming too big to design monolithic processing platforms. This is where the hArtes approach with reconfigurable heterogeneous systems becomes vital. As a part of the project, a modular and scalable hardware platforms will be developed that can be reused and re-targeted by the tool chain to produce optimized real-time embedded products. The results obtained will be evaluated using advanced audio and video systems that support next-generation communication and entertainment facilities, such as immersive audio and mobile video processing. Innovations of the hArtes approach include: (a) support for both diagrammatic and textual formats

\textsuperscript{1}www.hartes.org.
\textsuperscript{2}www.bluebee-tech.com.
in algorithm description and exploration, (b) a framework that allows novel algo-
rithms for design space exploration, which aims to automate design partitioning,
task transformation, choice of data representation, and metric evaluation for both
hardware and software components, (c) a system synthesis tool producing near op-
timal implementations that best exploits the capability of each type of processing
element; for instance, dynamic reconfigurability of hardware can be exploited to
support function upgrade or adaptation to operating conditions.

Delft, The Netherlands  

K. Bertels
Acknowledgement

This book describes the outcome of the hArtes project (IST-035143) supported by the Sixth Framework Programme of the European Community under the thematic area Embedded Systems.
Contributors

Anna Antola  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Giovanni Beltrame  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Koen Bertels  Fac. Electrical Engineering, Mathematics & Computer Science, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
Ferruccio Bettarelli  Leaff Engineering, Via Puccini 75, 62016 Porto Potenza Picena, MC, Italy
Jean-Francois Bonastre  Université d’Avignon et des Pays de Vaucluse, 339 Chemin des Meinajaries, 84911 Avignon, France
Francois Capman  Thales Communications, 146 Bd de Valmy, 92704 Colombes, France
Emiliano Capucci  Faital Spa, Via B. Buozzi 12, 20097 San Donato Milanese, MI, Italy
Stefania Cecchi  DIBET—Universita Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
Antonio Cerruto  Atmel Corp., Milan, Italy
Maria Teresa Chiaradia  Politecnico di Bari, Bari, Italy
Emanuele Ciavattini  Leaff Engineering, Via Puccini 75, 62016 Porto Potenza Picena, MC, Italy
Immacolata Colacicco  Università di Ferrara, via Saragat 1, 44100 Ferrara, Italy
Fabrizio Ferrandi  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Jose Gabriel de Figueiredo Coutinho  Imperial College, 180 Queen’s Gate, London SW72A2, UK
Luca Fossati  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Volker Hahn  Fraunhofer IGD, Fraunhoferstraße 5, 64283 Darmstadt, Germany
Gregor Heinrich  vsonix GmbH, Fraunhoferstraße 5, 64283 Darmstadt, Germany; Fraunhofer IGD, Fraunhoferstraße 5, 64283 Darmstadt, Germany
Christoph Jung  Fraunhofer IGD, Fraunhoferstraße 5, 64283 Darmstadt, Germany
Georgi Kuzmanov  TU Delft, Delft, The Netherlands
Ariano Lattanzi  Leaff Engineering, Via Puccini 75, 62016 Porto Potenza Picena, MC, Italy
Marco Lattuada  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Christophe Levy  Université d’Avignon et des Pays de Vaucluse, 339 Chemin des Meinajaries, 84911 Avignon, France
Fabian Logemann  Fraunhofer IGD, Fraunhoferstraße 5, 64283 Darmstadt, Germany
Yi Lu  TU Delft, Delft, The Netherlands
Wayne Luk  Imperial College, 180 Queen’s Gate, London SW72A2, UK
Giacomo Marchiori  Università di Ferrara, via Saragat 1, 44100 Ferrara, Italy
Thomas Marconi  TU Delft, Delft, The Netherlands
Roel J. Meeuws  TU Delft, Delft, The Netherlands
Andrea Michelotti  Atmel Corp., Milan, Italy
Yuet Ming Lam  Imperial College, 180 Queen’s Gate, London SW72A2, UK
Alberto Morea  Politecnico di Bari, Bari, Italy
Zubair Nawaz  TU Delft, Delft, The Netherlands
Raffaele Nutricato  Politecnico di Bari, Bari, Italy
S. Arash Ostadzadeh  TU Delft, Delft, The Netherlands
Lorenzo Palestini  DIBET—Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
Paolo Peretti  DIBET—Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
Francesco Piazza  DIBET—Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
Christian Pilato  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Andrea Primavera  DIBET—Università Politecnica delle Marche, Via Brecce Bianche 1, 60131 Ancona, Italy
Mojtaba Sabeghi  TU Delft, Delft, The Netherlands
Contributors

**Donatella Sciuto**  Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy

**Kamana Sigdel**  TU Delft, Delft, The Netherlands

**Vlad Mihai Sima**  TU Delft, Delft, The Netherlands

**Simon Thabuteau**  Thales Communications, 146 Bd de Valmy, 92704 Colombes, France

**Tim Todman**  Imperial College, 180 Queen’s Gate, London SW72A2, UK

**Romolo Toppi**  Faital Spa, Via B. Buozzi 12, 20097 San Donato Milanese, MI, Italy

**Raffaele Tripiccione**  Università di Ferrara, via Saragat 1, 44100 Ferrara, Italy

**Yana Yankova**  TU Delft, Delft, The Netherlands
List of Figures

Fig. 1.1 Growth of the ES industry (EU report) ............... 2
Fig. 1.2 Proportion of software cost in total cost of end product (EU report) 3
Fig. 1.3 Market for multi-core CPU’s in ES .......................... 3
Fig. 1.4 The hArtes tool chain ...................................... 5
Fig. 1.5 Mapping Code on various platforms .................... 6

Fig. 2.1 The hArtes toolchain ........................................ 11
Fig. 2.2 NU-Tech graphical user interface ......................... 36
Fig. 2.3 NU-Tech RealTime Watch .................................. 37
Fig. 2.4 NU-Tech Profiling information .......................... 38
Fig. 2.5 GAETool ..................................................... 39
Fig. 2.6 GAETool Eclipse Wizard .................................... 40
Fig. 2.7 Hgprof Tool Flow ............................................. 42
Fig. 2.8 Interaction of tools, scripts, and intermediate results in the Quipu modeling approach .......................... 44
Fig. 2.9 Predicted versus measured number of flip-flops on a Virtex-4 LX200 using the Quipu prediction model. These are validated predictions using leave-one-out validation ................................ 47
Fig. 2.10 PandA framework ............................................. 50
Fig. 2.11 PandA automatic parallelization .......................... 51
Fig. 2.12 Example of program in original format (a) and in the corresponding SSA-form (b) ............... 52
Fig. 2.13 Example of Control Flow Graph of basic blocks .............. 54
Fig. 2.14 Control dependence graph (a), data dependence graph (b), anti-dependence graph (c) .............................. 57
Fig. 2.15 Program dependence graph .................................. 59
Fig. 2.16 Nodes belonging to different loops are clustered in different subgraphs ............................................. 60
Fig. 2.17 Phases of the parallelism extraction algorithm .................... 61
Fig. 2.18 A task graph compliant with the fork/join programming model (on the left) and one not compliant ............................... 62
Fig. 2.19 Fork (a) and Join (b) node sharing ................................ 65
The task mapping approach is comprised by four main modules: (1) an automatic C guidelines verification process that automatically determines if the code conforms to the restrictions of the task mapping process, (2) a filtering engine that determines which processing elements can support each task, (3) a task transformation engine which produces one or more implementations of a given task that can potentially exploit the capabilities of the processing elements in the system, and (4) the mapping selection process which assigns a processing element implementation to each task of the application.

An overview of the task transformation engine. The task transformation engine receives as input a task and additional parameters such as the processing element that we wish to target, and generates a set of implementations. The set of transformations to be applied to each processing element is provided by the user. The implementations of transformations are stored as shared libraries for ROSE transformations, and as text files for CML-based transformations. A CML description consists of three sections: the pattern to match, the matching conditions, and the resulting pattern (Listing 2.1).

Starting with the original code for the application that models the vibration of a guitar string, we explore ways of using seven different transformations to attempt to improve the run time and memory usage. Much of the speedup comes from simplifying the code and making iteration variables integer, while the remainder comes from caching to prevent repeat memory access and removing a constant assignment from the loop body. The caching also enables one array to be eliminated (about 33% reduction in memory usage), possibly at the expense of performance.

An overview of the mapping selection process.

Searching for the best mapping and scheduling solution using multiple neighborhood functions. The solid arrows show the moves generated by different neighborhood functions. The dotted arrows denote the best move in each iteration of the search. PE: processing element, tk: task.

A description of C guidelines automatically detected by hArmonic, and a screenshot of its graphical user-interface.

Example of MOLEN.

Molen without parallelism.
Fig. 2.33 Molen with parallelism expressed with OpenMP syntax ........ 91
Fig. 2.34 The DWARV toolset ........................................ 92
Fig. 2.35 Representation of the linking process in the hArtes toolchain ... 99
Fig. 2.36 Harte framework .............................................. 101
Fig. 3.1 The Molen machine organization ............................ 113
Fig. 3.2 The Molen programming paradigm ........................... 114
Fig. 3.3 Top level architectural structure of the hArtes Hardware Platform. The system has two independent heterogeneous and configurable processors that communicate among each other and with an audio I/O subsystems; the latter supports several ADAT channels. 117
Fig. 3.4 Detailed block diagram of the Basic Configurable Element (BCE) of the hArtes Hardware Platform. The BCE is the basic building block of the platform, supporting several processing architectures. One or more BCEs work in parallel to support an hArtes application ............................................. 118
Fig. 3.5 Picture of the hArtes Hardware Platform. The two BCEs use two daughter boards each, one for the D940HF processor and one for the FPGA based infrastructure. These daughter-boards are at the center of the mother board. The ADAT interfaces and several standard I/O connectors are clearly visible at the top and at the bottom of the picture, respectively ................................. 120
Fig. 4.1 Prototype setup used for the telepresence application ........... 127
Fig. 4.2 Telepresence scenario: functional overview ........................ 128
Fig. 4.3 Multi-channel beamforming: Listening in two directions ........ 129
Fig. 4.4 Wave-field synthesis: Matching the original and synthesized fields 130
Fig. 4.5 Generic structure of the array processing algorithms ............. 131
Fig. 4.6 Structure of the beamforming algorithm ........................ 132
Fig. 4.7 Structure of the wave-field synthesis algorithm .................. 133
Fig. 4.8 Overlap-Save fast convolution for a filter block $H(z)$. Square brackets indicate frame indices, interleaved signals are denoted $x_{12}$ etc., sizes of data blocks are to scale .............................. 135
Fig. 4.9 Multitap ring buffer with wrap-around output for a delay line $D(z) = z^{-M}$ ................................................................. 136
Fig. 4.10 Main loop C-language structure of the generic algorithm in Fig. 4.5 ................................................................. 137
Fig. 4.11 Implementations of audio array processing algorithms .......... 139
Fig. 4.12 Exemplary benchmark of memory transfer (hHP DMA read from shared memory to DSP-internal memory) .............................. 141
Fig. 4.13 Embedded platform benchmark based on the generic array processing algorithm in Fig. 4.5. Left: optimal path (D-MEM represents shared memory, FPGA used for hHP audio I/O), top-right: memory transfers on DEB and hHP (linear coefficients: DSP cycles at 80MHz = $a + Nb$). Bottom-right:
Performance figures for filter operation (cycles vs. frame-length, internal/external overlap refers to storing the previous OLS frame in DSP internal memory or shared memory) and multiply-add (cycles = $a + Nb$).

Fig. 4.14 Main results of the hArmonic tool in combination with the synthesis toolbox and the hHP: (a) beamformer and (b) wave-field synthesis.

Fig. 4.15 A description of some of the hArtes constraints automatically derived by the hArmonic tool to ensure that the mapping solutions are feasible on the hArtes platform. This figure shows part of the log generated by hArmonic when processing the beamformer application. A total of 677 mapping constraints are identified for this application.

Fig. 5.1 Enhanced listening experience algorithms.

Fig. 5.2 General approach for speech recognition.

Fig. 5.3 General approach for speaker recognition.

Fig. 5.4 NU-Tech Framework.

Fig. 5.5 NU-Tech integration of single-channel speech enhancement.

Fig. 5.6 Effects of partitioning and mapping on the kernel application.

Fig. 5.7 Main results of the hArmonic tool in combination with the synthesis toolbox and the hArtes hardware platform: (a) Audio Enhancement application and (b) Stationary Noise Filter.

Fig. 5.8 Main features of the code generated by the hArmonic tool for each backend compiler in the synthesis toolbox. In this figure, we focus on the code generated for the ARM processor.

Fig. 5.9 PowerServer final setup in the car trunk.

Fig. 5.10 Loudspeakers position.

Fig. 5.11 Example of a midrange loudspeaker.

Fig. 5.12 Effect of adaptive stereo equalization on car.

Fig. 5.13 Smoothed car impulse response prototype and equalization weights during adaptive equalization.

Fig. 5.14 Stereophonic Echo Cancellation of a music sequence.

Fig. 5.15 Finally, the multi-channel speech enhancement module has been implemented in NU-Tech, using the SRP-PHAT criterion for on-line speaker localization.

Fig. 5.16 Limited search using region-constrained estimation of SRP-PHAT criterion.

Fig. 5.17 Scheme of the Power Server System considering the hartes platform.

Fig. 5.18 Scheme of the overall Audio Enhancement In-Car application developed for the hHp platform.

Fig. 5.19 In Car application performances considering different approaches.

Fig. 5.20 Workload of the overall Audio Enhancement system considering a parallelization of the applications.

Fig. 5.21 (a) Devices workload and (b) Devices memory requirements for the Audio Enhancement system in case of parallelized applications.
Fig. 5.22 In-Car communication performances in terms of time elapses . . 188
Fig. 6.1 Run-time environment ........................................ 194
Fig. 6.2 A sample CCG ............................................. 195
Fig. 6.3 Kernels and memories ........................................ 197
Fig. 6.4 AMMA postcode ............................................ 199
Fig. 6.5 Memory access ............................................. 201
Fig. 6.6 The Molen structure with interrupt support . . . . . . . . . . 202
Fig. 6.7 Simple general operation of Molen .......................... 203
Fig. 6.8 An example of runtime partial reconfiguration instructions . . 204
Fig. 6.9 Integration of the ICAP controller to Molen .................. 205
Fig. 6.10 Models .................................................. 206
Fig. 6.11 Simple example ........................................... 210
Fig. 6.12 Generic expression and its pipeline circuit for example in Fig. 6.11 210
Fig. 6.13 Graph to show speedup and area overhead w.r.t. dataflow . . . 212
Fig. 6.14 Application profiling and system level architecture exploration in the context of hartes toolchain .......................... 213
Fig. 6.15 rSesame framework ....................................... 215
Fig. 6.16 Dynamic profiling framework overview ....................... 216
Fig. 6.17 Sample profiling elements stored by QUAD in an architecture description file ........................................... 217
Fig. 6.18 The architecture of the ReSP system simulation platform .... 219
Fig. 6.19 Organization of the simulated environment including the Routine-Emulation module ................................. 221
Fig. 6.20 Internal structure and working mechanisms of the function routine emulator ........................................... 222
Fig. 6.21 Average execution time on all the 1344 runs for 8 benchmarks and for all the tested number of processors ........................ 223
Fig. 6.22 Execution Time in front of different System Call latencies: 2 PE 2 threads ........................................... 223
Fig. 6.23 Execution Time in front of different System Call latencies: 4 PE 4 threads ........................................... 223
Fig. 6.24 Execution Time in front of different System Call latencies: 8 PE 8 threads ........................................... 224
Fig. 6.25 Execution Time in front of different System Call latencies: 16 PE 16 threads ........................................... 224
Fig. 7.1 Persistence of legacy platforms and applications ................ 230
List of Tables

Table 2.1 Number of functions in each domain with the main algorithmic characteristics present in each application domain ........ 46
Table 2.2 Comparison of results of Ant Colony Optimization (ACO), Simulated Annealing (SA) and the Tabu Search (TS), along with a dynamic policy ........................................ 64
Table 2.3 Speed-ups for the parallel versions of JPEG and DRM applications as estimated by Zebu and measured on SimIt-ARM and ReSP .................................................. 68
Table 2.4 Performance estimation error on single processing element . 72
Table 2.5 Average and Maximum Speed-up Estimation Error on whole task graph compared with techniques based on Worst Case and Average Case ............................................ 74
Table 2.6 List of filters employed to find mappings ................... 78
Table 2.7 Evaluation of the main hArtes applications ............... 88
Table 2.8 C language support—data types .............................. 95
Table 2.9 C language support—storage ................................... 95
Table 2.10 C language support—expressions ............................ 95
Table 2.11 C language support—statements ............................. 96
Table 2.12 hArtes library components specifications .................. 98
Table 2.13 hArtes library components specifications ................. 98
Table 4.1 Reference implementations of the BF and WFS algorithms ........................................ 140
Table 4.2 Parameters of hArtes implementations of the BF and WFS algorithms ........................................ 143
Table 4.3 Evaluation of the automatic mapping approach on the beamformer and wave-field synthesis applications. The hArtes constraints correspond to the number of automatically derived constraints to make the mapping solution compatible with the hArtes platform. The speedup achieved corresponds to the ratio between the performance of the application running on the ARM and the mapping solution produced by the hArmonic tool . 148
Table 4.4 Beamformer application performance (parameters see Table 4.2) 151
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>WFS application performance (parameters see Table 4.2)</td>
</tr>
<tr>
<td>5.1</td>
<td>Evaluation of the automatic mapping approach on the Audio Enhancement</td>
</tr>
<tr>
<td></td>
<td>application and related kernels (Xfir, PEQ, FracShift, Octave), and the</td>
</tr>
<tr>
<td></td>
<td>Stationary Noise Filter. The speedup corresponds to the ratio between the</td>
</tr>
<tr>
<td></td>
<td>performance of the application running on the ARM and the mapping solution</td>
</tr>
<tr>
<td></td>
<td>produced by the hArmonic tool</td>
</tr>
<tr>
<td>5.2</td>
<td>Average execution time of direct and inverse 1024-pts complex FFT for</td>
</tr>
<tr>
<td></td>
<td>different optimization levels</td>
</tr>
<tr>
<td>5.3</td>
<td>Performances of the martes and DSE approach in terms of workload for each</td>
</tr>
<tr>
<td></td>
<td>algorithm with relation to the direct implementation</td>
</tr>
<tr>
<td>6.1</td>
<td>Memory block scores (in ms)</td>
</tr>
<tr>
<td>6.2</td>
<td>Execution overheads of tracking the memory</td>
</tr>
<tr>
<td>6.3</td>
<td>Status of CCU and the microcode</td>
</tr>
<tr>
<td>6.4</td>
<td>Time and hardware to compute DCT</td>
</tr>
</tbody>
</table>