Aalborg Universitet

Fast System-Level Design of Wireless Applications

Le Moullec, Yannick; Christensen, Søren Skovgaard; Chenpeng, Wen; Koch, Peter; Bilavarn, Sébastien

Published in:
Proceedings of the International Symposium on Wireless Personal Multimedia Communications (WPMC 2005)

Publication date:
2005

Document Version
Accepted author manuscript, peer reviewed version

Link to publication from Aalborg University

Citation for published version (APA):
Le Moullec, Y., Christensen, S. S., Chenpeng, W., Koch, P., & Bilavarn, S. (2005). Fast System-Level Design of Wireless Applications. In *Proceedings of the International Symposium on Wireless Personal Multimedia Communications (WPMC 2005)* Center for TeleInFrastruktur (CTIF), Aalborg Universitet. http://www.wpmc-archives.com/site/user_login.php?page_no=1&menu_kind=ref&ref_mode=keyword&ref_head=&ref_year=&within_key=&dtl_ref_title=&dtl_ref_keyword=&dtl_ref_author=moullec&dtl_ref_belong=&dtl_start_year=2004&dtl_end_year=2005&menu_state=inp&essay_id=679

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: December 06, 2025

Fast System-Level Design of Wireless Applications

Yannick Le Moullec[1], *Søren Skovgaard Christensen*[1,2], *Wen Chenpeng*[2],
Peter Koch[1,2], *Sébastien Bilavarn*[3]

[1]Center for Embedded Systems (CISS), [2]KOM department, Aalborg University
DK-9220 Aalborg Ø, Denmark
[3]Signal Processing Institute, EPFL
CH-1015 Lausanne, Switzerland

moullec@cs.aau.dk, http://www.ciss.dk

***Abstract*—This article presents a new methodology to speed-up the development cycle of wireless applications. This methodology is composed of three steps 1) algorithm design with Matlab, 2) algorithmic-level characterization and parallelism exploration from the C language with "Design Trotter" and 3) hardware synthesis from the Handel-C language with "DK Design Suite". Few hardware design experience background is required in this flow since the parallelism information provided by Design-Trotter is extremely useful to develop the Handel-C description of the application. The proposed methodology has been used to explore the design space of a RAKE algorithm and to synthesize it onto a FPGA. The experimental results illustrate how designers can rapidly converge from the specification phase to a high performance solution.**

Key words: Design Space Exploration, parallelism, metrics, digital signal processing, SoC.

1. INTRODUCTION

Designing wireless DSP systems is more and more challenging because i) their needs in terms of computational power are ever increasing, and ii) the time-to-market factor is more and more critical. Therefore, designers need user-friendly, rapid and efficient design methodologies that can help them to rapidly explore and prototype their applications.

In this paper, a design methodology providing rapid design space exploration and short time-to-prototype for the design of DSP systems is proposed. The foundation of the methodology is a combination of academic and commercial tools, each specialized in a specific aspect of the design flow (fig.1): i) algorithm design with Matlab (MathWorks), ii) algorithmic-level characterization and parallelism exploration with "Design-Trotter, and iii) hardware synthesis with "DK Design Suite" (Celoxica). Compared to a traditional design flow, an explicit design space exploration step (algorithm characterization and parallelism exploration, step 2 in Fig.1) is added. This enables the generation of a system with maximal performances via an optimal exploitation of the parallelism.

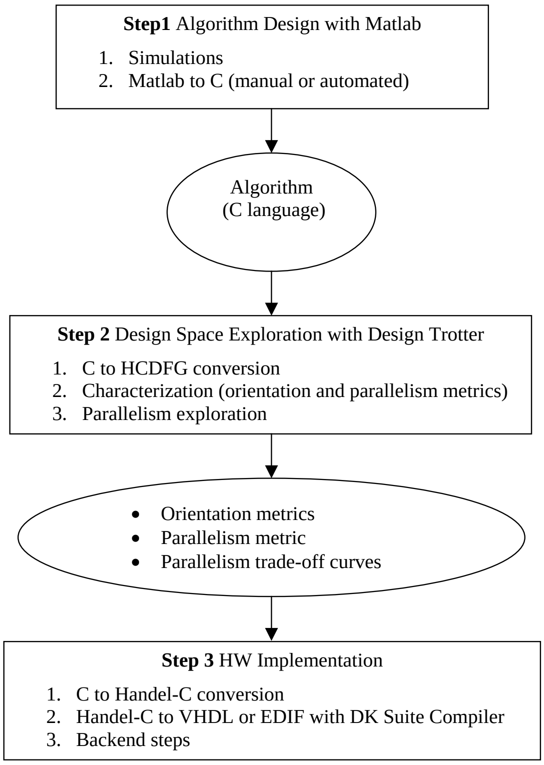

Step1 Algorithm Design with Matlab
1. Simulations
2. Matlab to C (manual or automated)
Algorithm
(C language)
Step 2 Design Space Exploration with Design Trotter
1. C to HCDFG conversion
2. Characterization (orientation and parallelism metrics)
3. Parallelism exploration
Orientation metrics
Parallelism metric
Parallelism trade-off curves
Step 3 HW Implementation
1. C to Handel-C conversion
2. Handel-C to VHDL or EDIF with DK Suite Compiler
3. Backend steps

Fig.1 Our methodology for rapid development of DSP applications

The rest of the paper is organized as follows: section 2 describes the different steps of the design flow. Section 3 presents some experimental results illustrating how designers can rapidly converge from the specification

This is the authors' version of a paper that has been published in the proceedings of the Eight International Symposium on Wireless Personal Multimedia Communications, Aalborg, Denmark, 18-22 September 2005.

phase to the final synthesis of the system onto a FPGA. Section 4 concludes the paper and presents some directions for future work.

2. STEPS OF THE DESIGN FLOW

The proposed methodology is presented in Fig.1. The following sub-sections describe the three steps of the design flow.

2.1. Step 1: design of the algorithm

Step 1 deals with the design of the DSP algorithm. In this step, new ideas (new algorithms, new features, improvements, etc.) are explored. Those new ideas are then typically verified by means of simulations. For this purpose, Matlab from MathWorks is used. During this step, the algorithm is specified, simulated and refined until the desired performance factor values are obtained. Typical performance factors are the SNR (Signal to Noise Ratio), BER (Bit Error Rate), the required number of bits, etc. Moreover those performance numbers can be evaluated for several scenarios (single vs. multi-user, multi-path channel, etc.). Once the theoretical performances of the algorithm have been obtained, a central step must be performed before to actually implement the system into hardware. This design space exploration step is used to characterize and explore the algorithm parallelism, as described in sub-section 2.2.

2.2. Step 2: design space exploration

Our academic tool, Design-Trotter, is used in Step 2 to characterize and explore the algorithm parallelism. The input language of Design-Trotter is a large subset of the C language. The conversion from Matlab to C can be performed manually or automatically using tools such as [1].

2.2.1. Sub-step 2.1: C to HCDFG

Sub-step 2.1 converts the C source code into a HCDFG, namely a Hierarchical Control Data Flow Graph [2]. The HCDFG model is the internal representation of Design-Trotter and includes DFGs, multidimensional data representation, control structures, hierarchy etc. More details about the HCDFG model can be found in [2] and [3].

2.2.2. Sub-step 2.2: characterization

In Sub-step 2.2 an abstracted characterization (without any architectural assumptions) of the algorithm is performed. It relies on the computation of three metrics: a) COM (Control Orientation Metric) which expresses the ratio of control oriented operations; b) MOM (Memory Orientation Metric) which expresses the ratio of memory oriented operations and c) Gamma which expresses the potential parallelism of the algorithm. These metrics guide the designer for implementation decisions such as HW/SW pre-partitioning. For each metric a general formula is given; the formulas for the HCDFG constructs (CDFG, HCDFG, etc.) are detailed in [3].

2.2.2.1 Control Orientation Metric (COM)

COM indicates the ratio of control operations (i.e., tests that cannot be eliminated at compile time) in CDFGs and HCDFGs. The general formula of COM is the ratio between i) the number of non-deterministic test operations and ii) the total number of operations including processing operations, tests and accesses to the global memory. COM values are bounded in the interval [0;1]. The more COM gets close to 1, the more a function is control dominated, involving complex control structures that are not well suited for hardware implementation. It also indicates that using very long pipelined processors is not efficient for such functions.

2.2.2.2 Memory Orientation Metric (MOM)

MOM indicates the ratio of memory accesses in a graph. The general formula for MOM is the ratio between i) the number of global memory accesses and ii) the number of global memory accesses plus the number of processing operations. Its value is bounded in the interval [0;1]. High MOM values indicate that processing operations are applied to “new” data (i.e., fresh input data, as opposed to data computed previously that might reside in a local memory). The more MOM gets close to 1, the more the function is data-transfer oriented. If hard time constraints have to be met, high performance memories are required (large bandwidth, dual-port memory, etc.), as well as an efficient use of memory hierarchy and data locality.

2.2.2.3 Criticity metric

The criticity is defined by the metric Gamma such as the ratio between i) the number of processing and memory accesses operations and ii) the critical path. The critical path of a DFG is defined as the longest chain of sequential operations (expressed in cycle number). The critical path for a function (i.e., for an H/CDFG) is computed hierarchically by combining the critical path of its sub-parts. Gamma indicates the average parallelism available at a specific hierarchy level. A function with a high Gamma value can benefit from an architecture offering high parallelism capabilities. On the other hand, a function with a low Gamma value has a rather sequential execution. In that case the acceleration of this function can be made via temporal parallelism (e.g., long pipeline), depending on the value of the COM metric. From a consumption point of view, a function with a high parallelism offers the opportunity to reduce the clock frequency by exploiting the spatial parallelism.

2.2.3. Sub-step 2.3: parallelism exploration

Sub-step 2.3 performs the exploration of the algorithm parallelism. Its principle is to schedule the algorithm onto a generic (virtual model) of the architecture (that can reflect HW and/or SW devices) for many time constraints (expressed in clock cycles at this level of abstraction). The different time constraints express as many possible solutions: for a given time constraint, a certain quantity of operations has to be executed simultaneously, which implies that a sufficient quantity of operators has to be available on the architecture. The solutions generated by Design-Totter are represented with a convenient 2D graphical form using trade-off curves. They represent the number of required operators vs. the number of clock cycles. Thus, they provide accurate estimations to evaluate the acceleration potential of a HW implementation [4]. Before scheduling, the designer can specify the types of operators to use, possibly corresponding to a given implementation device. This flexibility offers him the possibility to create new architectures or to reflect existing ones. The scheduling technique is very fast [5], thus the designer can rapidly explore and prune a very large design space. By referring to the metrics and parallelism exploration results provided by step 2, the designer rapidly evaluates the needs of the application and can choose/build the most appropriate architecture. Parallelism exploration results provide key information on the location and the amount of parallelism that will allow for an optimal implementation of the functions to be mapped onto hardware.

2.3. Step 3: FPGA implementation

In the proposed methodology, the hardware implementation is performed in Step 3. This step is based on the DK-Suite tool that enables Handel-C to FPGA synthesis.

2.3.1. Sub-step 3.1: C to Handel-C

The “C to FPGA” concept in DK-Suite requires a Handel-C version of the algorithm; this involves that the designer must manually express parallelism using the "par" statement. Moreover, to enable resource sharing the designer have to use functions to define the operators (fine or coarse grain) that can be shared. This is where the parallelism trade-off curves generated by Design-Trotter are used: by referring to the schedule details provided by those curves, the designer can easily find out i) where parallelism opportunities are located, ii) which resources should be shared and iii) reflect those options in the Handel-C code by means of "par" statements and functions respectively. This step is not yet automated; however it is easily performed by using the GUI of Design-Trotter: the designer only needs to click on the solution from the trade-off curve that he wants to implement; this opens a new window showing the schedule details, for all the hierarchy levels of the algorithm.

2.3.2. Sub-step 3.2: Handel-C to VHDL or EDIF

Dk-Suite features two design trajectories: from the Handel-C source code it is possible to obtain either a VHDL code or an EDIF netlist. Choosing the VHDL path enables further modifications and optimizations, for example in the generated Finite State Machine. Both approaches are simple and straightforward, requiring a few configuration parameters only. Generic and target-dependant optimizations are available to further improve the design efficiency. The synthesis technology behind DK-suite is described in [6].

2.3.3. Sub-step 3.3: Back-end steps

Once the VHDL or EDIF file has been generated with DK-Suite, the FPGA implementation is performed. If the VHDL option has been chosen, HDL synthesis plus place and route have to be performed. If the EDIF option has been chosen, only place and route is needed.

3. EXPERIMENTAL RESULTS

We have applied the proposed methodology to explore and implement a RAKE algorithm [7]. This algorithm processes vector inner product operations on complex numbers. First of all, the algorithm has been designed and simulated with Matlab. Fig.2 shows 5 simulated BER curves: floating point and 6,8,10,12 bits fixed points representations. The 12 bits curve is almost overlapping the floating-point curve, offering enough performance for this study case. Since floating-point implementations are extremely expensive in terms of hardware, it has been decided to implement the RAKE algorithm with a 12 bits, fixed point, representation.

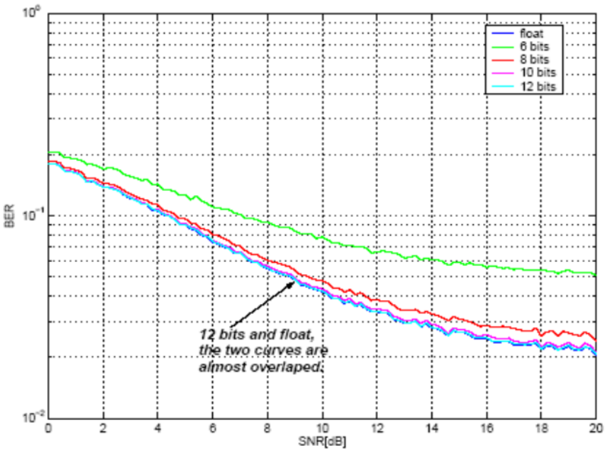


Fig. 2 RAKE receiver BER simulation with Design-Trotter

This is the authors’ version of a paper that has been published in the proceedings of the Eight International Symposium on Wireless Personal Multimedia Communications, Aalborg, Denmark, 18-22 September 2005.

Subsequently the C code has been created, using the Matlab code as a reference. The algorithm has been characterized and its parallelism explored with Design-Trotter. A screenshot of the GUI of Design-Trotter at work is shown in fig.3. Table 1 and table 2 summarize the results generated by Design-Trotter.

Table 1: Rake characterization with Design-Trotter

Level	MOM	COM	Gamma
Main	0.746	0.001	3.417
Outer loop	0.746	0.013	3.417
Inner loop	0.750	0.025	3.303

Table 1 shows the characterization results. Firstly, the metrics indicate that the algorithm is relatively memory oriented (MOM values are larger than 2/3). This is mainly due to the fact that the algorithm accesses a large number of input-data. Secondly, the COM metric values indicate that there are almost no non-deterministic control operations; this is explained by the fact that the loop indices are not data-dependent. Finally, Gamma values indicate that most of the parallelism is concentrated in the inner loop (since Gamma values do not increase significantly for the outer loop and the main function).

Table 2 shows 7 Pareto-like solutions, generated by the exploration feature of Design-Trotter. Each solution describes a possible implementation of the algorithm, where the fastest ones are the most expensive in terms of resources. In this experiment we have chosen to implement the solution executing in 12499 cycles (4 ALUs, 3 Mults, 4 simultaneous memory accesses) onto a Xilinx FPGA.

Table 2: Rake parallelism exploration with Design-Trotter

Nb cycles	Nb ALUs	Nb Multipliers	NB memory accesses
10211	5	5	5
11795	5	4	5
12499	4	3	4
13203	3	2	4
15315	3	2	3
16723	2	2	3
25611	1	1	1

Subsequently to the characterization and the design space exploration with Design-Totter, the C to Handel-C conversion has been performed. The development of the Handel-C version has been guided by the results provided by Design-Trotter. Those results show where the parallelism is available in the algorithm and are used to choose where to use the 'par' statement and functions in the Handel-C code. Those two features are used to indicate which operations and functions should execute simultaneously and those that should share resources, respectively. Once the Handel-C code has been created we have generated the corresponding VHDL code with DK-Suite. Subsequently, this VHDL code has been synthesized with Xilinx ISE for a Xilinx XC2S200 FPGA. The HDL synthesis results are given in table 3. Finally, the low-level synthesis and place and route steps have been performed, with the same tool. The low-level implementation results are given in table 4.

Table 3: RAKE algorithm HDL synthesis results generated with Xilinx ISE

Macro-statistics	
# adders/subtractor	4
# multipliers	3

Table 4: RAKE algorithm low-level synthesis results generated with Xilinx ISE for a Spartan-II xc2s200 FPGA

Logic Utilization	Used	Available	Utilization
Number of Slices	225	2352	9%
Number of Slice Flip Flops	192	4704	4%
Number of 4 input LUTs	309	4704	6%
Number of bounded IOBs	80	288	27%
Number of GCLKs	1	4	25%

4. CONCLUSIONS AND FUTURE WORK

This paper has described a fast, user-friendly methodology to speed-up the complete development cycle of DSP applications. This methodology is composed of three steps: i) algorithm design with Matlab, ii) algorithmic-level characterization and parallelism exploration with Design-Trotter, and iii)

This is the authors' version of a paper that has been published in the proceedings of the Eight International Symposium on Wireless Personal Multimedia Communications, Aalborg, Denmark, 18-22 September 2005.

FPGA implementation with DK Design Suite. The three successive steps of the flow enable DSP system designers to rapidly converge from the specification phase to the final synthesis of the system onto a FPGA. Firstly the algorithm is designed and simulated with Matlab. Then, as opposed to traditional design-flows, a central step provides an efficient exploration of the algorithm properties. This design space exploration step, performed with our tool Design-Trotter, consists in the algorithm characterization and the exploration of its parallelism. This indicates the designer what are the processing and data-transfer orientations of the algorithm and shows him where the its potential parallelism is available by means of parallelism vs. cycle budget trade-off curves and schedule details. Subsequently, the parallelism information provided by Design-Trotter is used to guide the development of the Handel-C description of the algorithm, and the Handel-C to FPGA synthesis is performed.

Our experiments with the RAKE algorithm have illustrated how designers can easily design, explore and synthesized DSP algorithms. Future works will investigate the use of other HLS tools in order to compare the differences in terms of performance; another point is to verify the level of optimization potential still left to the designer in the backend steps. Furthermore, the automation of the parallelism specification in the Handel-C code to enable parallel execution and resource sharing remains an open research issue that should be investigated.

REFERENCES

[1] MATLAB to C-Code Synthesis. Catalytic-Inc. http://www.catalytic-inc.com/synthesis.html.

[2] Y. le Moullec, J-Ph. Diguet, N. Ben Amor, T. Gourdeaux and J-L. Philippe. Algorithmic Level Specification and Characterization of Embedded Multimedia Applications with Design-Trotter. To appear in Journal of VLSI Signal Processing (Springer), 2005.

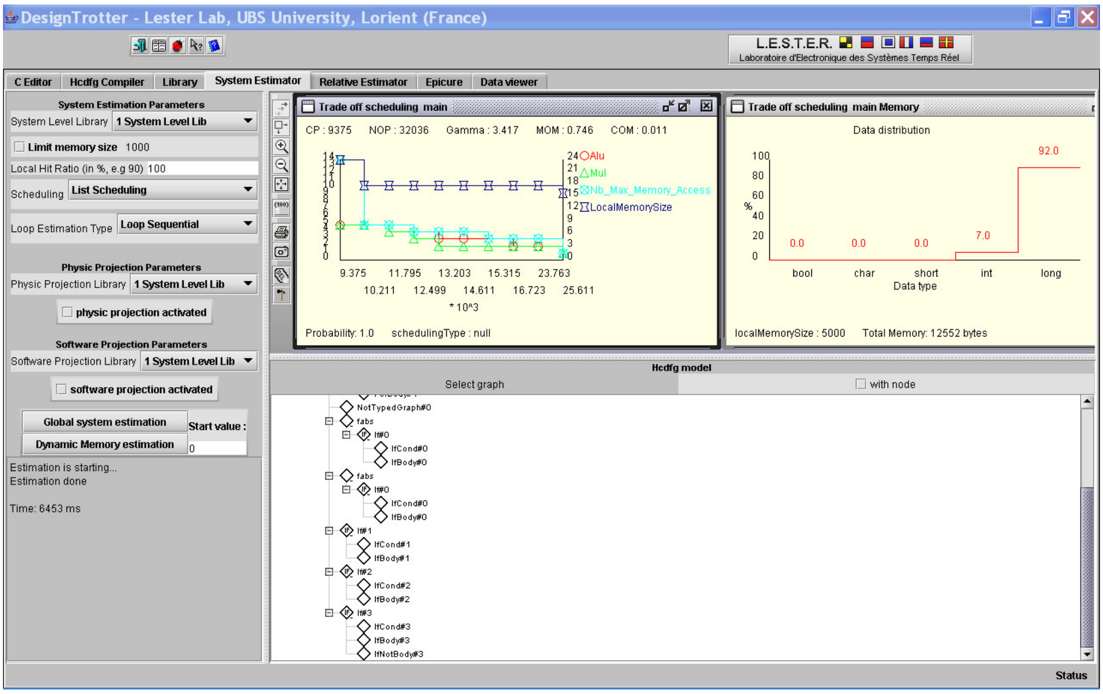
[3] Y. le Moullec, N. ben Amor, J-Ph. Diguet, J-L. Philippe, and M. Abid. Multi-granularity Metrics for the Era of Strongly Personalized SoCs. In Design Automation and Test in Europe Conference (DATE03), Munich, Germany, March 2003.

[4] S. Bilavarn, G. Gogniat, J-L. Philippe and L. Bossuet, Fast Prototyping of Reconfigurable Architectures from a C Program, IEEE International Symposium on Circuits and Systems, Bangkok, Thailand, May 2003.

[5] Y. le Moullec, P. Koch, J-Ph. Diguet, and J-L. Philippe. Design-Trotter: Building and Selecting Architectures for Embedded Multimedia Applications. In International Symposium on Consumer Electronics (ISCE03), Sydney, Australia, December 2003.

[6] Dk design suite datasheet. www.celoxica.com.

[7] Søren S. Christensen and Tarik Vaizovic. Analysis and Implementation of Linear Receivers for DS-CDMA Signals. Master's thesis, Aalborg University, Denmark, 2002.


DesignTrotter - Lester Lab, UBS University, Lorient (France)

System Estimation Parameters
System Level Library: 1 System Level Lib
☐ Limit memory size 1000
Local Hit Ratio (in %, e.g 90) 100
Scheduling: List Scheduling
Loop Estimation Type: Loop Sequential

Physic Projection Parameters
Physic Projection Library: 1 System Level Lib
☐ physic projection activated

Software Projection Parameters
Software Projection Library: 1 System Level Lib
☐ software projection activated

Start value : 0

Estimation is starting...
Estimation done

Time: 6453 ms

CP : 9375 NOP : 32036 Gamma : 3.417 MOM : 0.746 COM : 0.011

Probability: 1.0 schedulingType : null

localMemorySize : 5000 Total Memory: 12552 bytes

Fig. 3 Characterization and parallelism exploration with Design-Trotter

This is the authors' version of a paper that has been published in the proceedings of the Eight International Symposium on Wireless Personal Multimedia Communications, Aalborg, Denmark, 18-22 September 2005.