DESIGN OF FIXED-POINT EMBEDDED SYSTEMS (DEFIS) FRENCH ANR PROJECT

D. Menard[1]*, R. Rocher*[1]*, O. Sentieys*[1]*, N. Simon*[1]*, L-S. Didier*[2]*, T. Hilaire*[2]*, B. Lopez*[2]*, E. Goubault*[3]*, S. Putot*[3] *F. Vedrine*[3]*, A. Najahi*[4]*, G. Revy*[4]*, L. Fangain*[5]*, C. Samoyeau*[5]*, F. Lemonnier*[6]*, C. Clienti*[6]

[1] University of Rennes I, IRISA/INRIA, 6 rue de kerampont, F-22300 Lannion
[2] University Pierre et Marie Curie, LIP6, 4 place Jussieu, F-75005 Paris
[3] CEA LIST Saclay Nano-INNOV, Point Courrier no 174, F-91191 Gif sur Yvette
[4] Univ. Perpignan Via Domitia, DALI/LIRMM, F-66860, Perpignan,
[5] InPixal, 80 Avenue des Buttes de Cosmes, F-35700 Rennes
[6] Thales TR&T Route Departementale, F-91120 Palaiseau

ABSTRACT

Embedded applications are usually coming with stringent constraints in term of cost, energy consumption and real-time. Consequently, fixed-point arithmetic is mainstream for their implementation into embedded systems. Hence, the main objective of the french ANR project DEFIS is to provide a complete design flow for fixed-point refinement of complex applications. This tool flow is like *the missing link* between high-level application specification tools and low-level implementation.

1. INTRODUCTION

Many embedded systems contain applications integrating mathematical processing. To satisfy the constraints (area, energy consumption, execution time) inherent to embedded systems, fixed-point arithmetic is widely used and preferred. However, applications are designed and simulated using floating-point data types and then implemented into fixed-point architectures. Therefore, reducing time-to-market and development cost strongly requires efficient and fast high-level development tools to automate fixed-point conversion.

The main objective of DEFIS project is to propose new approaches to improve the efficiency of the fixed-point conversion process and to provide a complete design flow for fixed-point refinement of complex applications. The benefit obtained with this design flow in terms of development time and quality of the generated solution will be demonstrated through experimentations. This infrastructure will significantly reduce the design time by automating the fixed-point conversion and it will enable to explore the trade-off between application quality and implementation cost. Moreover, this flow will guarantee and validate the numerical behavior of the resulting implementation. Publications and more details related to this project can be found on the project web site `http://defis.lip6.fr/`.

2. DEFIS INFRASTRUCTURE DESCRIPTION

The general design flow proposed in the DEFIS project is presented in Figure 1. The complete application is described through a dataflow specification and the quality criteria associated to the application are defined. These criteria are used as constraints for the proposed design flow. The application is modeled by a set of blocks interconnected together. Each block can be specified with a C code using floating-point data types. In addition, parameterized IP block can be used to specify specific kernel. The supported IP blocks are Linear-Time Invariant systems or polynomial evaluation. LTI systems correspond to digital filters (FIR, IIR), transforms (FFT, DCT, etc.) and other linear processing. Polynomial evaluation is widely used to implement (elementary) functions.

2.1. System level optimization

The first stage of the fixed-point conversion of the application is the system level optimization. To handle a complete application made-up of different blocks, a hierarchical approach can be used to optimize the fixed-point specification. The aim is to find for each block the numerical accuracy which minimizes the global cost for a given application quality constraint. Then, the fixed-point conversion of each block can be carried-out independently.

2.2. Algorithm level optimization

The second stage corresponding to the algorithm level optimization, seeks to find the best structure, which minimizes the implementation cost. For a parameterized IP block, the available structures are known a priori and, thus, a set of transformations can be applied to select the structure which lead to the best implementation cost for a given numerical accuracy constraint. For C codes, source-to-source transformations are

applied to reorganize the processing inside the code to improve the numerical accuracy and thus decrease the cost.

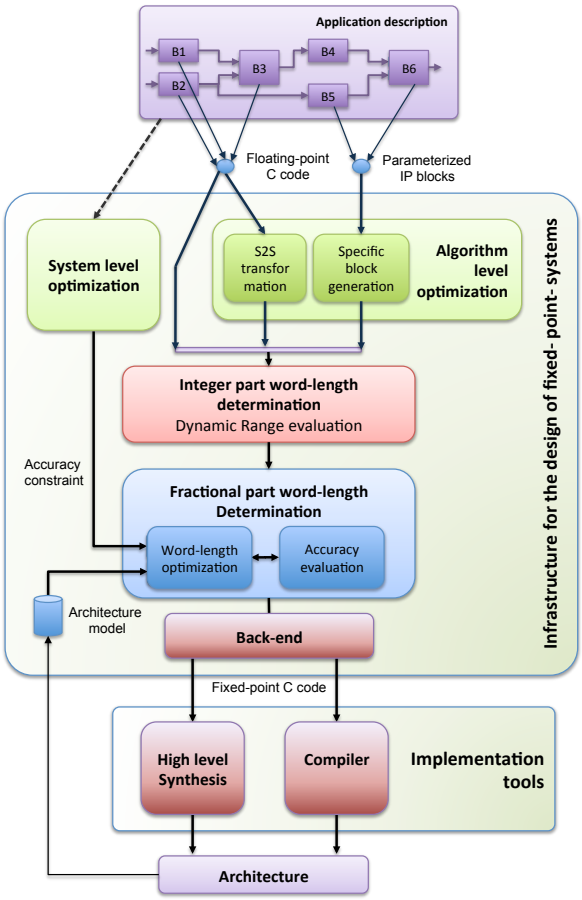

Fig. 1. Design flow of the DEFIS project.

2.3. Fixed-point conversion

The third stage corresponds to the fixed-point conversion of each block and is split into two steps. Firstly, the number of bits for the integer part is determined after the dynamic range evaluation. This number of bits must be minimized while ensuring that no overflow or a limited number of overflow occurs. Secondly, the number of bits for the fractional part is determined. The limited number of bits used to code the data generates an error between the infinite precision and finite precision values. This error must be estimated and controlled. The aim of this part is to optimized the fixed-point specification (word-length optimization) to minimize the implementation cost for a given numerical accuracy constraint. The variables of this optimization problem are the data word-lengths. For a software implementation, the data word-lengths take a limited number of values, which depend on the data types supported by the processor. For a hardware implementation, the architecture is fully defined by the designer. The variables can take any value in a given range (6 to 40 bits for example) and the search space can be huge. Thus, the challenge is to have an efficient approach to evaluate the numerical accuracy (numerical accuracy evaluation), so as to limit the optimization time.

For dynamic range evaluation, classical techniques, based on interval arithmetic or affine arithmetic, overestimate the dynamic range and may lead to pessimistic results. Consequently, several bits of numerical accuracy but only the integer part may be wasted. For many applications targeting a trade-off between performance and implementation cost, a low probability of overflow can be tolerated if the degradation of the application performance is limited. Thus, in this case, the aim is to minimize the integer part word-length (IWL) under application performance constraint.

For numerical accuracy evaluation, analytical methods based on perturbation theory allow a fast estimation of the numerical accuracy. However, only systems made-up of smooth operators are supported. The aim is to extend the class of supported systems by handling unsmooth operators like decision operators. Likewise, static analysis will be extended to support fixed-point operations.

2.4. Implementation

After the fixed-point conversion of each block, a new C code integrating fixed-point data types is generated with the back-end module. This C code is used as input of classical commercial tools (implementation tools) to implement it into the targeted architecture. In the DEFIS project, three architectures are targeted, corresponding to a SIMD accelerator, a processor and a FPGA. For hardware implementation, a FPGA platform is considered and high-level synthesis tool will be used. For software implementation, a VLIW SIMD processor design by Thales is considered.

3. VALIDATION OF THE DEFIS INFRASTRUCTURE

The proposed infrastructure will be tested on classical benchmarks and then validated on two real applications provided by the industrial partners. The two applications are a real time video background/foreground separation computation and a STAP algorithm for radar processing. The results obtained with the infrastructure developed in the DEFIS project will be compared with a reference implementation provided by industrial partners. Each reference implementation has been optimized manually. To analyze the quality of our approach, the gain in terms of development time will be measured and the implementation cost of the two implementations will be compared.

4. ACKNOWLEDGEMENT

This work is supported by the french *Ingenierie Numerique et Securite 2011* ANR program : ANR-11-INSE-0008.