Deriving Resource Efficient Designs Using the REFLECT Aspect-Oriented Approach

(Extended Abstract)

José G.F. Coutinho[1], João M.P. Cardoso[2], Tiago Carvalho[2], Ricardo Nobre[2], Sujit Bhattacharya[3], Pedro C. Diniz[4], Liam Fitzpatrick[5], and Razvan Nane[6]

[1] Department of Computing, Imperial College London, UK
[2] Faculdade de Engenharia da Univ, do Porto, Dep. Eng. Informática, Portugal
[3] Department of Electrical and Electronic Eng., Imperial College London, UK
[4] INESC-Investigação e Desenvolvimento, Lisboa, Portugal
[5] ACE Associated Compiler Experts bv, The Netherlands
[6] Technical University of Delft, The Netherlands
www.reflect-project.eu

In the context of the REFLECT project[1] we have developed an aspect-oriented compilation and synthesis toolchain that aims at facilitating the mapping of applications described in high-level imperative programming languages, such as C, to heterogeneous and configurable computing systems. More specifically, we have designed an aspect-oriented domain-specific language, called LARA[2], that allows programmers to convey application-specific and domain-specific knowledge as a way to capture non-functional concerns. The LARA specifications and the subsequent control of the tools via a code weaver allows a seamless exploration of alternative designs and run-time adaptive strategies, in effect enabling design-space exploration (DSE).

Figure 1 depicts a specific instantiation of the REFLECT aspect-oriented design-flow, which generates resource-efficient Xilinx designs from C kernels and LARA descriptions. This design-flow operates as follows. There are 3 main inputs: (1) C application sources, (2) input parameters that control which and how kernels are synthesized to hardware, and (3) LARA aspects that capture the DSE strategy that derives the final designs. The DSE weaver invokes the Harmonic weaver to compute the word-lengths of variables based on user-provided parameters, such as input ranges and required output accuracy. The results of the word-length analysis are captured as a LARA aspect and passed down to the Reflectc weaver, which controls the CoSy [3] engines. The word-length information contains the precision of each variable, including the minimum number of bits for the integer, fraction and signal that satisfy the target output accuracy. The Reflectc weaver receives the LARA aspect with the word-length information and the C kernel, and uses the weaveshrink CoSy engine [3] to change data-types from floating-point to fixed-point using the specified word-length information. Next, the weaver invokes the DWARV code generator to derive the optimized design in VHDL. Finally, the DSE weaver invokes the Xilinx ISE tools to generate the corresponding hardware design.

P. Brisk, J.G. de Figueiredo Coutinho, P.C. Diniz (Eds.): ARC 2013, LNCS 7806, pp. 226–228, 2013.

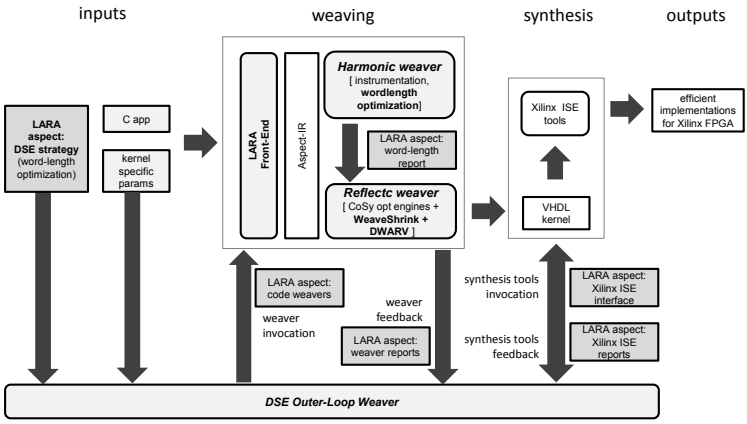

Fig. 1. The REFLECT Aspect-Oriented Design-Flow

Next we present a LARA aspect description that captures a word-length strategy that operates on the design-flow shown in Fig. 1. This aspect receives a list of target accuracies (line 6), and generates a set of FPGA implementations that satisfy the computation requirements of each element of the list, given a set of C sources and a function name (line 4):

```
1  import xilinx_opts, store_results;
2  aspectdef wlot_strategy
3     input // kernel params
4        csource, cflags, kernel_name,
5        input_ranges, // e.g. { x: { min: -1, max: 10} }
6        targetAccs, // e.g. [3e-1, 9e-4, 9e-9],
7        targetVar, maxFreq,
8     end
9     for (t in targetAccs) {
10       var targetAcc = targetAccs[t];
11       run(tool:'harmonic', args:['-aspLARA=wlot_harmonic.lara', ...]);
12       run(tool:'reflectc', args:['wlot.lara','kernel.c','-gen=vhdl']);
13       call xilinx_opts(folder:"VHDL", opt:4, ctrMaxFreq: maxFreq);
14       println("MaxFreq : "+ @design.CCU.maxFreq + " (MHz)");
15       dir = 'wcode' + t;
16       call store_results(dir);
17    }
18 end
```

Table 1 presents the results obtained with the REFLECT design-flow using Xilinx ISE tools [4] on a 3D path planning kernel [1], and targeting a Virtex-5 FPGA-based platform. The results compare the original single precision version of the kernel against 3 automatically derived fixed-point designs using the above aspect strategy with target accuracies of 7.0e-9, 6.0e-4 and 1.0e-1. These target accuracies generated the Q3.29, Q3.13 and Q3.5 fixed-point designs respectively. Qx.y represents a fixed-point representation with x integer bits and y fractional bits. Considering the single floating-point precision design as the reference, we

achieve area savings from 27.54% to 34.09% by dropping the computation accuracy from 5.58e-9 to 9.4e-2. Regarding the power savings, we achieve dynamic power savings from 21.14% to 35.02% by dropping the computation accuracy from 5.58e-9 to 9.4e-2, when running the design at 300 MHz.

Table 1. FPGA resources reported for various data type representations

	accuracy		area		power	
	target accuracy	computation accuracy	slices	reduction slices (%)	dynamic power @300Mhz (mW)	reduction dynamic power @300Mhz (%)
single precision	-	4.2e-7	748	-	142.93	-
Q3.29 (unsigned)	7.0e-9	5.58e-9	542	27.54%	112.72	21.14%
Q3.13 (unsigned)	6.0e-4	3.6e-4	498	33.42%	98.9	30.81%
Q3.5 (unsigned)	1.0e-1	9.4e-2	493	34.09%	92.87	35.02%

These results show a clear trade-off between accuracy and resource utilization, and between accuracy and dynamic power. By decoupling non-functional concerns (expressed in LARA) with functional concerns (implemented in C), we can easily revise the computational requirements and even the strategy itself to automatically derive new solutions. Furthermore, this approach allows the development of compilation strategies that can be re-used and applied to different applications and possibly different target architectures, thus increasing design productivity for the same target as well as code portability across multiple and very distinct target architectures.

Acknowledgements. This work was partially supported by the European Community's Framework Programme 7 (FP7) under contract No. 248976, 257906 and 287804, and UK EPSRC. The authors are grateful to the members of the REFLECT project for their support.

References

1. REFLECT Project (2013), `http://www.reflect-project.eu`
2. Cardoso, J.M.P., Carvalho, T., Coutinho, J.G.F., Luk, W., Nobre, R., Diniz, P.C., Petrov, Z.: LARA: An Aspect-Oriented Programming Language for Embedded Systems. In: Proc. of the ACM Intl Conf. on Aspect-Oriented Software Development (AOSD 2012) (March 2012)
3. ACE CoSy® Compiler Development System (2012), `http://www.ace.nl/compiler/cosy.html`
4. All Programmable Technologies from Xilinx Inc. (2011), `http://www.xilinx.com`