A Composable and Integrable Hardware Compiler for Automated Heterogeneous HW/SW co-design Tool-Chains

Razvan Nane*, Koen Bertels*,[1]

* *Computer Engineering, TU Delft, Mekelweg 4, 2625 CD Delft, The Netherlands*

ABSTRACT

Recent years have showed that the complexity of embedded systems and their architectures is growing rapidly. At the same time, there is a continuous decrease in the time to market. This is due to market demand for more functionality and due to more advanced technologies becoming available and being combined together in new and innovative ways. This complexity growth demands improved tool support. It will not be possible to cost-efficiently engineer high quality systems without extensive use of different tools which work well together. In order to achieve this, we intend to develop a framework for integrating the DWARV c-to-vhdl compiler into a tool chain. Tool integration is a real challenge, and even more so when the tools are being used by engineering communities with different traditions and cultures; just contrast the terminology used by high-level modelling communities of software engineers with that of low-level hardware design communities.

The framework that we develop at TU Delft will make it thus possible to integrate tools in a well-functioning tool chain – with minimal effort – allowing different tools to be interoperable and where one can easily replace one tool with another, similar one. Tools that have the first property we define it as being *vertically integrable* in a tool-chain, whereas the second property we define it as *horizontally integrable*. In this paper we will show how we intend to modify available tools and tool-chains to make these both vertically and horizontally integrable at the hand of three case studies.

KEYWORDS: IP-XACT; Hardware Dependent Software; Design Patterns; Aspects; Meta-modells; Tool Integration; DWARV compiler

[1]E-mail: {r.nane, k.l.m.bertels}@tudelft.nl

1 The SoftSoc Approach

Many advanced consumer products such as mobile phones, set-top-boxes (STB) and digital televisions are based on System on Chip (SoC) solutions consisting of a highly integrated chip and associated software. 95% of SoCs combine hardware IP cores (function specific cores and accelerators) and programmable computing cores (CPU, DSP, ASIP). The integration of the HW IP cores in the SoC requires complementary software for controlling the HW IP. We refer to this software as Hardware Dependent Software (HDS). Designing, building, configuring, integrating and testing of HDS for advanced SoCs has grown to become a huge task. SoftSoC aims at solving the main SoC productivity bottleneck by providing HDS solutions to enable SoC designers to aggregate multiple HW IP with their associated HDS into efficient design [webc].

As the integration tasks become more and more time and resource consuming, it is all the more important to keep the consistency between the different steps of the development process. The concepts and tools proposed by SoftSoC address this specific matter by automating the generation and composition processes. To prove the benefit of this approach, the CE lab at TU Delft works closely with LIACS institute in Leiden. The goal of this cooperation is to integrate DWARV generated IP kernels in the KPN-based ESPAM tool-flow. DWARV [YKB+07] is a c-to-vhdl compiler developed at TU Delft and was designed to work for the Molen machine organization [VGBP03] that relies on a shared memory model, whereas ESPAM [NSD06] from LIACS, is targeting the Kahn Process Network (KPN) computation model which rely on distributed memory.

In order to combine these tools working with different underlying computation models, one will need to redesign/rewrite the IP cores generated by DWARV so that they are ESPAM-compliant. This is a difficult and time consuming task. However, if there was a method to inform DWARV on the special needs of the required generated kernel and this had the possibility to read and understand these requirements, then the integration process could be fully automated. To this purpose we use the IP-XACT schema descriptions [weba], which provide us a standardized way to input requirements about software and what exactly is needed. The same approach can be followed also at the integration side, where the kernels provided by DWARV will be complemented with a schema description describing what was generated and an extra package containing HDS for it, that provides the functionality/interface to use this new piece of hardware. The complete tool-flow is shown in Figure 1a).

Therefore, ESPAM incompatible IP cores can be integrated in the ESPAM tool-flow instead of using only ESPAM compliant IP generator tools. This demonstrates that by using the Softsoc approach, the DWARV tool can become horizontally integrable.

2 The REFLECT Approach

The flexibility of FPGAs allows them to achieve orders of magnitude better performance than conventional computing systems via customization. Programming these, however, is extremely cumbersome and error-prone and as a result their true potential is only achieved at an unreasonably high effort. This problem is partially solved by hardware compilers, e.g. DWARV. These, however, are still in an infant phase and bring only a part of the benefit of using a high level language and associated compiler, i.e. they are used more as a map-

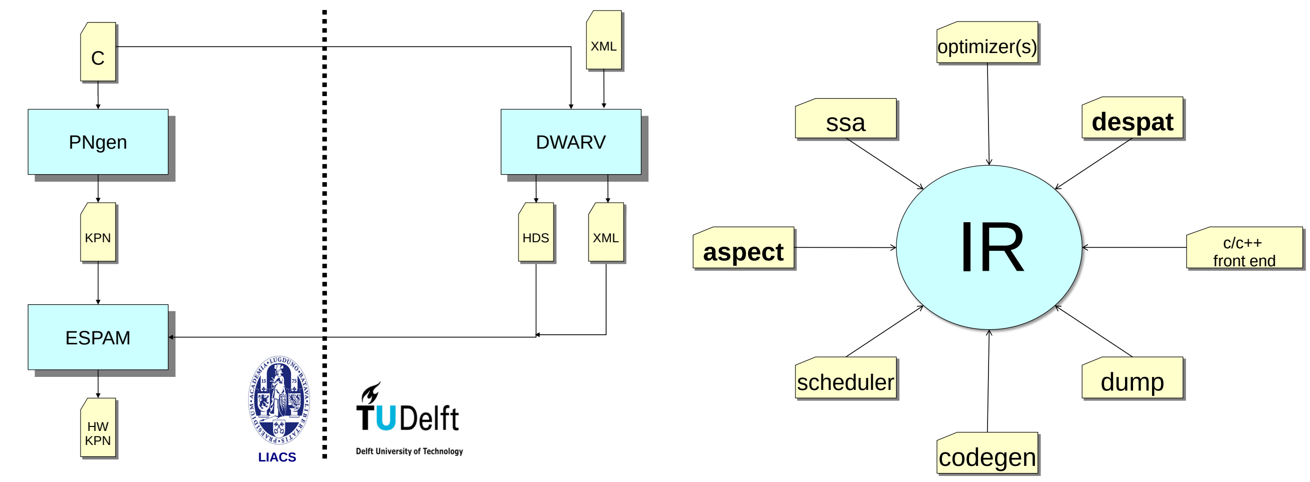

Figure 1: a) DWARV / ESPAM composition; b) DWARV engines

ping tool between high level programming to low level concepts/syntax. The real benefits of using software compilers, e.g. reusing design knowledge and programming in a clean, structured way, are still missing.

REFLECT will develop, implement and evaluate a novel compilation and synthesis system approach for FPGA-based platforms. We rely on Aspect-Oriented Specifications (AOS) and Design Patterns (DP) to convey critical domain knowledge to a mapping engine while preserving the advantages of a high-level imperative programming paradigm in early software development and portability [webb]. In the scope of this project, our efforts consists initially on redesigning the DWARV compiler using the Cosy compiler framework [cw] in order to make the tool modular and extensible, with the goal in mind to easily integrate DP and AOS to reuse design experience and knowledge.

The new DWARV compiler will solve these issues by its inherited modularity from the Cosy framework. To exemplify this, Figure 1b) shows the engines composing the compiler. As we can see in the picture, DWARV will be built from separate entities that will all view and work on the same intermediate representation (IR). By having this internal decoupling of the modules and the IR, engines like *despat* will be able to process various DP and make annotation in the IR, while the *aspect* engine will read these annotations and will weave in the details of the DP used in the final *codegen*. These newly generated cores by the compiler will take then into consideration the design knowledge and experience of the user giving thus the possibility to (re)use abstract information in tool-chains that is not particular to any tool and is described in standard UML. This is equivalent to the horizontally integrable property.

3 The IFEST Approach

The complexity of embedded systems and their architectures is growing rapidly. This is due to market demand for more functionality and due to more advanced technologies becoming available and being combined together in new and innovative ways. This complexity growth demands improved tool support. It will not be possible to cost-efficiently engineer high quality systems without extensive use of different tools which work well together. In order to achieve this, iFEST is proposing to develop a framework for integrating tools into a tool chain.

The work that we will conduct in this respect, that will augment the work explained in sections 1 and 2, consists of making the DWARV tool also vertically integrable. So far we were concerned with making it horizontal integrable, i.e. making DWARV able to replace any tool-chain specific tool. Although this can be achieved, there is still the problem of how to make the rest of the tool-chain integrate and understand how it needs to call DWARV. To solve this problem, we use the IFEST approach in which we shows that by building meta-models of every tool in a tool-chain, information can be saved in an abstract and standardized way (XML) that unrelated tools could use to create and operate a custom tool-flow.

4 Conclusions

In order to cope with the ever increasing hardware capabilities and tool-chains complexity, we forsee a need to cope with tool(-chains) interoperability and interchangeability. This means to create a way to let the various tools interact with each other when their interfaces are different. As highlighted in this paper, this boils down to make the tool models both horizontally and vertically integrable. To do this we need to build meta-models around them at the various phases/steps in the design process. Our efforts at TU Delft focuses mostly towards this goal and the advantages of this approach are *Design time reduction*; *Tool and tool-chain integration*; *Reuse of common knowledge in the form of Design Patterns and Hardware templates*; and *Benefit of standardized ways of structuring the information needed by the tools by using IP-XACT descriptions and/or aspects*.

References

[cw] ACE compiler webpage. http://www.ace.nl/compiler/.

[NSD06] Hristo Nikolov, Todor Stefanov, and Ed Deprettere. Multi-processor system design with espam. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 211–216, New York, NY, USA, 2006. ACM.

[VGBP03] S. Vassiliadis, G. N. Gaydadjiev, K.L.M. Bertels, and E. Moscu Panainte. The molen programming paradigm. In *Proceedings of the Third International Workshop on Systems, Architectures, Modeling, and Simulation*, pages 1–10, July 2003.

[weba] IP-XACT webpage. http://www.spiritconsortium.org/tech/refs/.

[webb] REFLECT webpage. http://www.reflect-ist.org/.

[webc] SoftSoc webpage. http://www.softsoc.org/.

[YKB$^+$07] Y. D. Yankova, G.K. Kuzmanov, K.L.M. Bertels, G. N. Gaydadjiev, Y. Lu, and S. Vassiliadis. Dwarv: Delftworkbench automated reconfigurable vhdl generator. In *In Proceedings of the 17th International Conference on Field Programmable Logic and Applications (FPL07)*, pages 697–701, August 2007.