High-Level synthesis based on Xilinx Vivado for hardware accelerators

Adrián Domínguez Hernández, Antonio Núñez Ordóñez, Pedro Pérez Carballo
SICAD Division
Institute for Applied Microelectronics - IUMA
Las Palmas de Gran Canaria, Spain
adominguez@iuma.ulpgc.es

Abstract – **This work describes key concepts in the implementation of a real time event processor using high-level synthesis methodology based on Xilinx Vivado HLS. The design flow starts from a SystemC functional model and has been refined using high-level synthesis methodology to RTL microarchitecture. The process is guided with performance measurements (latency, cycle, time, power, resource utilization) with the objective of assuring the quality of the final system.**

The results show that Vivado HLS provides improvements in the use of resources despite the difficulties to handle certain aspects of system descriptions languages as SystemC.

Finally, some recommendations about high-level synthesis with Xilinx Vivado HLS are given.

High level synthesis, FPGA, Vivado, Synplify, SystemC, abstraction, methodology

I. INTRODUCTION

Due to the growing complexity of SoC design is necessary to increase the level of abstraction from which its design is captured. At present, is desirable to create algorithmic descriptions that capture the functionality of the design in a high level language such as C/C++, SystemC and SystemVerilog. These specifications are transformed by synthesis techniques in the corresponding high-level RTL description [1][2][3].

The use of ESL methodologies facilitates the design of hardware accelerators that leverage implicit parallelism of hardware implementation to accelerate the execution of key cores of the application for its real-time behavior [1][2][3].

The event processing systems are typical examples of real-time systems. In them there is an arrival rate of events which must be processed in a available time. The operation of the event processing systems can be divided into the following stages: capture and event filtering, events processing and action generating [4].

The main idea of this work is the evaluation of the high-level synthesis methodology proposed by Xilinx Vivado HLS for comparison in terms of methodology and quality of the results obtained with Cadence CtoS [5][6].

II. DESIGN FLOW AND METHODOLOGY

The design flow is shown in Figure 1. Starts from a verified and functional SystemC code and performs the necessary adaptations to support the SystemC subset of Xilinx Vivado HLS. At each stage, a functional check is performed to verify that the functionality is not affected by the changes [7].

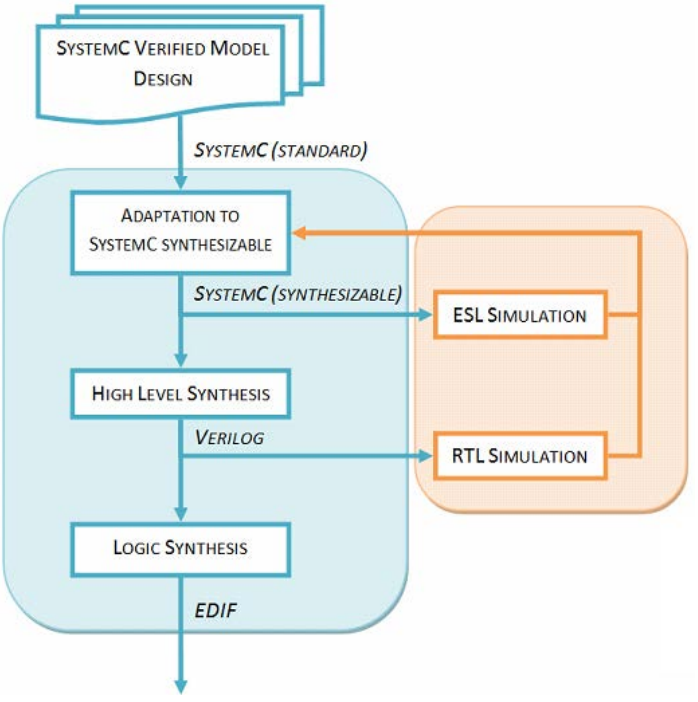

Figure 1. Proposed designflow.

III. RESULTS AND FLOW COMPARISON

Figure 2 shows the resource usage after logic synthesis. The resource usage are expressed in FFs and LUTs. The improvement of the results after logic synthesis is about 60% for resource usage. Moreover, Figure 3 shows the timing results. In

this case it is shown a remarked improvement in the clock period, increasing from 131.7 MHz to 221.3 MHz, as is also shown in Table1.

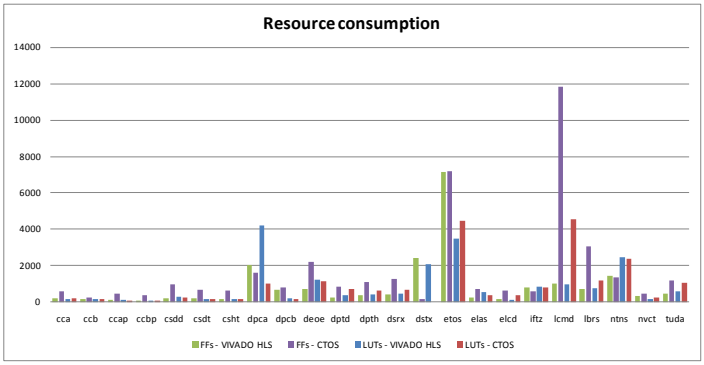

Figure 2. Logic synthesis resource results.

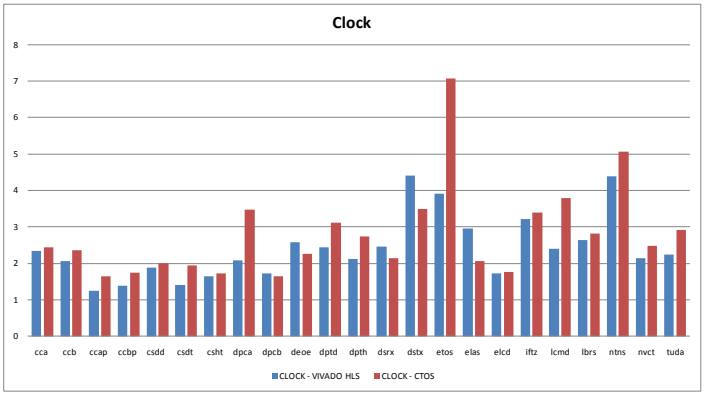

Figure 3. Logic synthesis timing results.

Table 1 shows the results of the complete processor. The improvement in resource usage in some cases is greater than 60% and more than 40% improvement in the operating frequency.

Table 1.Resource results for both design flows.

	XilinxVivado HLS flow	Cadence CtoS flow	Difference
LUT	12.443	25.774	51,72%
BRAM	45	58	22,41%
FF	15.127	44.635	66,11%
DSP	26	38	31,58%
Frecuencia	221.3	131.7	40,49%

IV. CONCLUSIONS

In this paper, we have followed the high-level synthesis design approach to obtain a RTL description ready to hardware implementation from an application described in a system description language like SystemC.

The results obtained with the design flow based on Xilinx Vivado HLS improve significantly with respect to the Cadence CtoS design flow despite of the difficulties encountered during development, as adapting interfaces, using directives and the necessary adjustments to make code synthesizable.

During the development of this work, some conclusions about system development tools based on high-level synthesis have been reached:

- The need to adapt the source code to a synthesizable subset makes difficult the task of re-synthesizing the code with each new tool. For designs where is necessary a design update and implementation of a complete design flow from time to time, the task of synthesis with any new and enhanced tool can be complex.
- Vivado HLS does not support certain aspects of SystemC and treats it as a class of C++, instead of a systems design language. This makes the designer's specifications in some cases are not synthetized successfully as is the case of input/output interfaces.
- Using directives in Vivado HLS to guide the synthesis to the desired result often does not yield good results, especially if SystemC is used.
- Vivado HLS is targeting a higher level of abstraction and is designed for quickly time-to-market.
- The tools of high-level synthesis are constantly evolving, allowing increasing the abstraction level in the design of electronic hardware systems. The tendency of such tools in the future indicates that hardware design systems will increasingly abstract, for example, in the communication step with the TLM libraries.

V. REFERENCES

[1] T. Grötker, *System Design with SystemC.* Boston: Kluwer Academic Publishers, 2002.

[2] M. Fingeroff, *High-Level Synthesis: Blue Book.* S.L.: Xlibris Corporation, 2010.

[3] P. Coussy, A. Morawiec, *High-Level Synthesis from Algorithm to Digital Circuit.* Springer, 2008.

[4] S. Levi, A. K. Agrawala. *Real-Time System Design.* Universidad de Michigan: McGraw-Hill Pub. Co., 1990.

[5] C. C.-T.-S. Compiler, «Cadence C-To-Silicon Compiler» [Online]. Available: http://www.cadence.com/products/sd/silicon_compiler/pages/default.aspx. [Last Access: March 2014].

[6] X. V. D. Suite, «Xilinx Vivado Design Suite» [En línea]. Available: http://www.xilinx.com/products/design-tools/vivado. [Último acceso: Marzo 2014].

[7] «Design + System Drivers Update» ITRS Public Conference, [Online]. Available: http://www.itrs.net/Links/2012Winter/1205%20Presentation/DesignSD_12052012.pdf. [Last Access: April 2014].