Design of a FFT/IFFT module as an IP core suitable for embedded systems

J. Viejo, A. Millan, M. J. Bellido, J. Juan, P. Ruiz-de-Clavijo, D. Guerrero, E. Ostua, and A. Muñoz
Grupo de Tecnologia Microelectronica
Departamento de Tecnologia Electronica-Universidad de Sevilla
E. T. S. Ing. Informatica, Campus Universitario Reina Mercedes
Sevilla 41012 (SPAIN)
e-mail: julian@dte.us.es, amillan@dte.us.es, bellido@dte.us.es, jjchico@dte.us.es,
paulino@dte.us.es, guerre@dte.us.es, ostua@dte.us.es, amrivera@dte.us.es

***Abstract*— In this work, we have laid the foundations that allow us to accomplish the implementation of a FFT/IFFT module as an IP core. The main objective is to design a configurable optimized core that can be integrated as a standard peripheral of a microprocessor system. Thus, three different methodologies have been compared: VHDL coding, System-level tools at RT level, and System-level tools at macroblock level; in order to propose a general methodology that facilitates the design process as well as allows designers to maintain total control over the module internal architecture.**

I. Introduction

The technological development has produced a significant increase of the integration density that is causing more and more parts of a complete system to be included inside the main core, constituted by a single chip. This chip has been traditionally referred to as Integrated Circuit, but new terms like Integrated System or System on Chip (SoC) have become popular because they better express the fact that the chip can contain not only a part of the system but the whole system itself.

Nowadays, SoC designs are commonly built out of already available parts in the form of Intellectual Property (IP) cores: microprocessors, memory blocks, ethernet controllers, standard input/output devices, etc. Thus, SoC designers typically do a work of integrations of already available parts and design of specific functions, as for example, digital signal processing (DSP) functions.

In this way, in previous works, we have presented the design and implementation of a FFT/IFFT module on FPGA (Field Programmable Gate Array) [1] and ASIC (Application Specific Integrated Circuit) [2]. These implementations were performed by using a methodology based on coding in VHDL (VHSIC Hardware Description Language) [3]. However, it was very important to evaluate the usefulness of the system-level tools provided by the FPGA foundry in order to improve such implementations. Thus, the objectives of the current work were: (a) perform a comparison between the VHDL coding and the system-level tools approach (b) propose a set of steps which summarize the best way to carry out the design process depending on the type of system, and (c) lay the foundations that allow us to design this DSP function as an IP core in the future.

The system-level tools approach allows designers to work at different abstraction levels. So, the first objective actually involved the comparison of three methodologies:

1) VHDL coding (VC): the system architecture is designed at RT (Register Transfer) level and implemented by direct coding in VHDL.
2) System-level tools at RT level (STR): the system architecture is designed at RTL and implemented using the system-level tools provided by the FPGA foundry. Specifically, we have used System Generator for DSP v7.1 from Xilinx [4].
3) System-level tools at macroblock level (STM): the FFT/IFFT module is implemented using the FFT macroblock provided by System Generator (which can also calculate the IFFT). This methodology is very easy to apply but it has an important drawback: the FFT macroblock only supports up to 16-bit data (having 19-bit as the amount necessary to maintain the precision), which produces a remarkable accuracy loss.

The rest of the paper is organized as follows: in the next section the three methodologies used and the main results for both the simulation and the implementation processes are presented, besides results obtained are discussed, in the third section, a general methodology is proposed, in the fourth section, lines of work that are going to be developed in the future are presented, and finally some conclusions are derived.

II. Comparison between methodologies

A. Design methodologies for implementation on FPGA

As we have mentioned previously, our aim was to evaluate the effectiveness of System Generator for DSP. The workflow we have used is conformed of three stages (Fig. 1):

1) Design: this stage differs from one methodology to another and it is explained on the next.
2) Synthesis and implementation: this stage is carried out by using ISE v7.1i from Xilinx. Moreover, the module operation, considering gate delay, is tested by post-P&R (Placement and Routing) simulation through ModelSim v6.0 from Mentor Graphics.

1-4244-0840-7/07/$20.00 ©2007 IEEE.

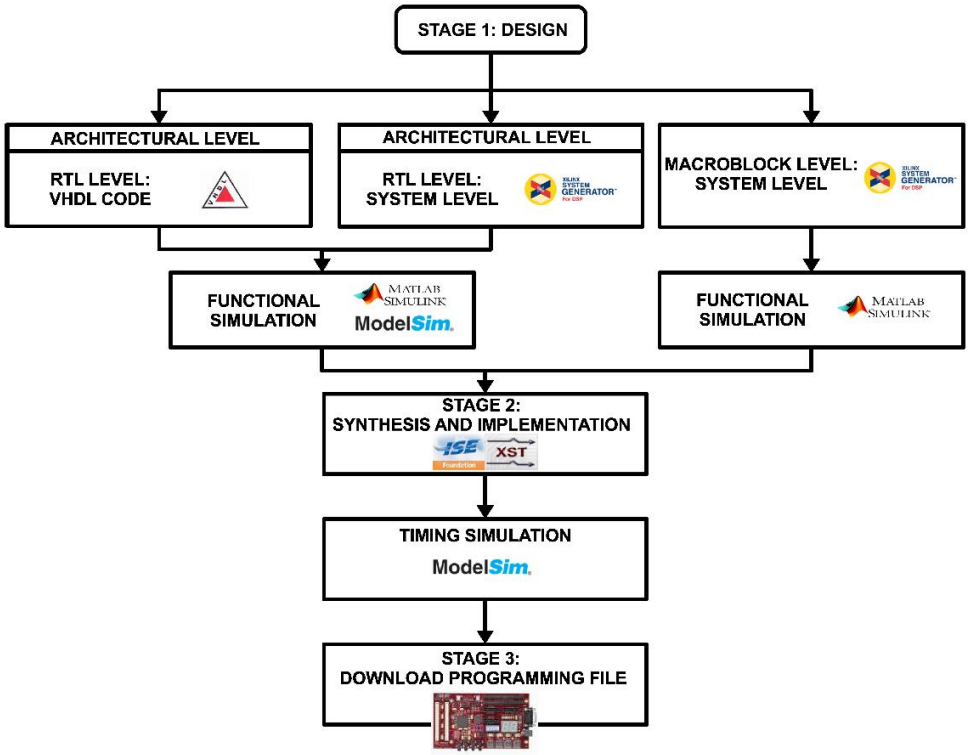

Fig. 1. Design and implementation workflow.

3) Programming: finally, the device is configured by using iMPACT v7.1i from Xilinx.

Because the design stage depends on the methodology used, a detailed explanation of each case is presented on the next subsections.

1) VHDL coding methodology: The VC methodology consists of developing the FFT/IFFT module at RT level using VHDL language and following the usual methodology for digital system design (based on a control unit and a data path). The architecture of the module has been suited to the algorithm presented in [2]. In this architecture, the most important component is the RADIX-8 butterfly that performs an 8-element DFT in a parallel way and its implementation follows the structure proposed in [5]. A more detailed explanation on the structure and functioning of the VHDL implementation can be found in [1],[2].

Also, verification of such designs is usually carried out by using a VHDL simulator like ModelSim. However, another alternative has been explored in this work: the VHDL code has been simulated using Simulink and ModelSim together (HDL co-simulation). That is possible through the System Generator's BLACK BOX block which allows designers to import VHDL code into a System Generator design. In this type of verification, System Generator's blocks are simulated through Simulink and the VHDL blocks (black boxes) are simulated through ModelSim.

2) System-level tools at RTL methodology: The STR methodology described in this work consists of designing the FFT/IFFT module using System Generator for DSP: a tool developed by Xilinx that facilitates the design and implementation of DSP functions on its FPGAs. This tool is a software platform integrated within Matlab and Simulink, from The MathWorks, and allows the design of DSP systems using the Xilinx BlockSet [7]. Also, it handles the automatic generation of VHDL code; synthesizable on Xilinx FPGAs.

Within this methodology, the same module architecture employed in the VC case has been applied. In this case, the data path has been built with the System Generator's blocks and the same control unit has been added to the design through the BLACK BOX block. Most of the system simulation is carried out using Simulink while ModelSim has been used to simulate the VHDL code (HDL co-simulation). Such simulation is mandatory in order to take into account those blocks, which are only available as black boxes.

3) System-level tools at macroblock level methodology: The STM methodology consists of modeling the FFT/IFFT module using the System Generator's FFT block. In this methodology, it is only necessary to design an interface that adapts the input/output signals of the FFT block to the module interface. In this case, the functional simulation is totally carried out using Simulink.

B. Design results

In this section, simulation and hardware implementation results obtained for the three methodologies are described in some detail.

1) Simulation results: In order to check that the designs work correctly, the following simulation process has been

TABLE I
SIMULATION RESULTS.

	VC	STR	STM
Clock cycles	292	292	341
Mean error	0.59%	0.59%	5.09%

carried out. At the first stage, designs have been verified using Simulink and ModelSim. For the generation of the input stimuli, the Source Blockset of Simulink has been employed. So, the input signal (DATA_IN_FFT) has been generated with Matlab and exported to Simulink through the From-Workspace block. This input is conformed by 64 complex numbers where real and imaginary parts are values from –1 up to 1. At the second stage, once the simulation has been finished, the To-Workspace block has allowed us to compare the results with the ones provided by the FFT Matlab function (the From/To-Workspace blocks provide an interface between Simulink and Matlab). In order to estimate the results accuracy, the duration of the whole calculation as well as the relative error of output values have been measured (Table I).

As we can see, the VC and STR implementations consume a lower amount of clock cycles for the calculation than the STM one (from 341 to 292 cycles). Also, the first two implementations drastically reduce the output error with respect to the STM one (from 5.09% to 0.59%).

2) Hardware implementation results: In order to compare the implementations of the different FFT/IFFT designs, they have been synthesized separately with ISE. In a first approach, they have been implemented on a Virtex XCV1000 FPGA. We have chosen this model because System Generator's FFT macroblock can only be implemented in this type of FPGAs. This device is in the high capacity range of the Virtex family and can allocate one million system gates working at 200 MHz.

In Table II, the implementation results for each design are shown. Two figures of merit are analyzed in this section: hardware resources used and maximum operation frequency. In terms of resource usage, the STM implementation obtains the best results (it saves about 6-7% slices with respect to the other ones). Analyzing the maximum operation frequency, we can see that STM also obtains the best result (62 MHz as opposite of 25-27 MHz).

In a second approach, the VC and STR implementations have been programmed on a Virtex-II XC2V2000 FPGA (as we have mentioned, the STM one can only be programmed on the Virtex XCV1000). This device has all the features necessary to implement DSP functions: two million system gates, 56 embedded multipliers, and 56 Block RAMs.

As in the Virtex XCV1000 case, both VC and STR produce similar results: VC saves a little amount of slices, flip-flops, and LUTs while STR saves four multipliers. Also, they reach an almost equal maximum operation frequency of 40 MHz (Table III).

TABLE II
HARDWARE IMPLEMENTATION RESULTS ON VIRTEX XCV1000.

	VC	STR	STM
Slices	2056 (16%)	1896 (15%)	1125 (9%)
Slice Flip Flops	629 (2%)	656 (2%)	1771 (7%)
4 input LUTs	3405 (13%)	3403 (13%)	1789 (7%)
Bonded IOBs	57 (11%)	57 (11%)	57 (11%)
Block RAMs	2 (6%)	2 (6%)	2 (6%)
MULT18x18	-	-	-
GCLKs	1 (25%)	1 (25%)	1 (25%)
Max. operation freq.	26.91 MHz	25.02 MHz	62.14 MHz

TABLE III
HARDWARE IMPLEMENTATION RESULTS ON VIRTEX-II XC2V2000.

	VC	STR	STM
Slices	1214 (11%)	1276 (11%)	-
Slice Flip Flops	619 (2%)	656 (3%)	-
4 input LUTs	1972 (9%)	2197 (10%)	-
Bonded IOBs	58 (14%)	58 (14%)	-
Block RAMs	2 (3%)	2 (3%)	-
MULT18x18	8 (14%)	4 (7%)	-
GCLKs	1 (6%)	1 (6%)	-
Max. operation freq.	40.02 MHz	40.03 MHz	-

C. Discussion

Firstly, simulation results show that the VC and STR implementations reduce the amount of clock cycles necessary to the calculation from 341 to 292 cycles with respect to STM (Table 1). Also, VC and STR reduce the output relative error in an important way (from 5.09% to 0.59%) with respect to STM. This is due to the internal data width they employ: STM only supports an internal width of up to 16 bits. However, the input data nature makes necessary an internal width of 19 bits in order to maintain accuracy. This is not a problem in VC and STR because designers totally control the internal structure.

Secondly, hardware implementation results show that STM obtains the best results (on Virtex XCV1000 FPGA): in terms of resource usage, it saves about 6-7% slices with respect to VC and STR (Table 2). This is because the FFT macroblock is specifically suited for this FPGA model. In the same way, we can see that STM achieves the highest maximum operation frequency: 62 MHz (compared to the 27 and 25 MHz of VC and STR respectively). However, this result is not so relevant because the module is only part of a complete system and its clock can not be faster than the global one. Thus, it is preferable to perform the calculation in less cycles if the module can reach the work frequency of the whole system.

In the Virtex-II XC2V2000 FPGA case, implementation results are also very similar for both VC and STR because they employ the same architecture (Table 3). Even they reach the same maximum operation frequency: 40 MHz. In this case, we can not compare them to STM because the last one can

not be programmed on this FPGA model.

Thirdly, on the one hand, it is remarkable that using system-level tools facilitates the design tasks greatly, allowing designers to focus on the system architecture and reducing the design time in an important way. This is possible because the library provided by the foundry is very optimized for the target programmable chips as well as the available blocks are fully parametrizable: we can decide which ones use on-chip resources (like BRAMs or embedded multipliers) or include different latency configurations (among other available options). This fact allows designers to totally control the way they are implemented and obtain a very efficient structure. On the other hand, System Generator does not count on some blocks that can be easily designed in VHDL, whose design becomes a tedious task by using a GDI (Graphical User Interface) tool, as for example register sets or decoders. Also, an important drawback is that some blocks can be implemented on specific FPGA models (as the FFT macroblock case).

In terms of VHDL coding, we have to remark the great amount of designs available through the Xilinx LogiCORE repository. This makes easier the design process although using such language involves a difficult simulation process due to the tediousness of the input stimuli generation. In this way, we have employed an alternative: simulate the VHDL code by HDL co-simulation.

III. Proposal of a General Methodology

In this section, we propose a specific methodology which allows designers to carry out this type of systems efficiently. In this way, we have to conclude that STM is the best option if the available macroblocks are suited to the specific aims (taking into account the possible data width and FPGA model restrictions) but, as a general methodology, our proposal combines both VC and STR and it can be summarized as follows:

1) Design the module with System Generator at RT level but using VHDL coding for those blocks that are easier to program with such a language (these blocks can be included into the module through the BLACK BOX block).
2) Verify the module by HDL co-simulation. This type of verification employs Simulink to simulate the System Generator blocks and ModelSim to simulate the VHDL ones (black boxes).

Thus, the proposed methodology facilitates the design process greatly as well as it allows designers to maintain total control over the module internal architecture and obtain an efficient structure.

IV. Future Lines of Work

As it was mentioned in the introduction, the main aim of this work is to design the FFT/IFFT module as an IP core, so that it can be used by SoC designers to build embedded systems. Thus, a set of future lines of work are presented next:

1) Firstly, it is necessary to improve the FFT/IFFT module in order to make it fully configurable in terms of data width, symbol length, internal data precision, etc.
2) Secondly, an important objective is to optimize the design, reducing the hardware resources used, increasing the operation frequency, and improving its performance with respect to other existing designs.
3) Finally, in order to facilitate the module usage, it is advisable to add an interface to it that allows its connection to a standard bus (On-Chip Peripheral Bus (OPB), Advanced Peripheral Bus (APB), etc.).

V. Conclusion

We have compared three different methodologies for the FPGA implementation of a FFT/IFFT module: VHDL coding, System-level tools at RT level, and System-level tools at macroblock level. In terms of resource usage, the last one is the best option if the available macroblocks are suited to the specific aims. However, such implementation has two main drawbacks: the internal data width is very restricted (which yields to an important output error) and the available FPGA models are very restricted too (to only one model for the FFT case). Thus, as a general methodology, our proposal is to combine the two first ones (VC and STR) in order to count on the advantages of them both: it facilitates the design process as well as allows designers to maintain total control over the module internal architecture and obtain an efficient structure (which greatly reduces the output error).

The work we have just carried out allows us to approach our next objectives: make the module fully configurable, improve its performance, and add a standard interface to it.

Acknowledgment

This work has been partially supported by the Spanish Government's MEC META project TEC-2004-00840/MIC and the Andalusian Regional Government's EXC-2005-TIC-1023 project.

References

[1] A. Millan, M. J. Bellido, J. Juan, P. Ruiz-de Clavijo, D. Guerrero, and E. Ostua, "Diseño eficiente de un modulo FFT/IFFT sobre FPGA," in *Proc. III Reconfigurable Computing and Applications Conference (JCRA)*, Madrid (Spain), Sept. 2003, pp. 107–114.

[2] A. Millan, M. J. Bellido, J. Juan, P. Ruiz-de Clavijo, D. Guerrero, E. Ostua, and J. Viejo, "Efficient design of a FFT/IFFT-64 module on ASIC," in *Proc. XI Iberchip Workshop (IWS)*, Salvador de Bahia (Brazil), Mar. 2005, pp. 305–306.

[3] P. J. Ashenden, *The Designer's Guide to VHDL*, 2nd ed. Academic Press, 2002.

[4] *Xilinx System Generator for DSP v8.1 User's Guide*, Xilinx Inc., 2005.

[5] T. Widhe, J. Melander, and L. Wanhammar, "Design of efficient radix-8 butterfly PEs for VLSI," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 3, Hong Kong (PRC), 1997, pp. 9–12.

[6] J. M. Berge, A. Fonkoua, S. Maginot, and J. Rouillard, *VHDL Designer's Reference*. Kluwer Academic Publishers, 1992.

[7] J. Hwang, B. Milne, N.Shirazi, and J. Stroomer, "System Level Tools for DSP in FPGAs," in *Proc. XI International Conference on Field Programmable Logic and Applications (FPL)*, Belfast, Northern Ireland (UK), Aug 2001.