IMPLEMENTATION OF A FFT/IFFT MODULE ON FPGA: COMPARISON OF METHODOLOGIES

J. Viejo, A. Millan, M. J. Bellido, E. Ostua, P. Ruiz-de-Clavijo, and A. Muñoz

Grupo ID2 (Investigacion y Desarrollo Digital)
Departamento de Tecnologia Electronica-Universidad de Sevilla
E. T. S. Ing. Informatica, Campus Universitario Reina Mercedes
Sevilla 41012 (SPAIN)
Email: {julian, amillan, bellido, ostua, paulino, amrivera}@dte.us.es

ABSTRACT

In this work, we have compared three different methodologies for the implementation of a FFT/IFFT module on FPGA: VHDL coding (VC), System-level tools at RT level (STR), and System-level tools at macroblock level (STM). In terms of resource usage and operation frequency, STM has obtained interesting results, although it has an important restriction about internal data width which produces a mean output error of 2.1%. VC and STR become a more general alternative that yields to a lower mean error (1.0%). Thus, we propose to combine VC and STR in order to facilitate the design process as well as allow designers to maintain total control over the module internal architecture and obtain an efficient structure.

1. INTRODUCTION

This work is included inside the design and implementation of a modem based on the Orthogonal Frequency Division Multiplexing (OFDM) modulation/demodulation system [1]. For both processes of modulation and demodulation, it is required to calculate the Discrete Fourier Transform of the data (in both direct and inverse ways). Thus, a FFT/IFFT module becomes an essential part of the system. The implemented system processes 64-sample complex symbols using 26-bit data (13 bits for the real part and 13 bits for the imaginary one).

In previous works, we have presented the design and implementation of the system on FPGA (Field Programmable Gate Array) [2] and ASIC (Application Specific Integrated Circuit) [3]. These implementations were performed by using a methodology based on coding in VHDL (VHSIC Hardware Description Language) [4]. 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 and (b) propose a set of steps which summarize the best way to carry out the design process depending on the type of system.

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 (Digital Signal Processing) v8.2 from Xilinx [5].

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.

The rest of the paper is organized as follows: in the next section the three methodologies used are described, the third section presents the main results for both the simulation and the implementation processes, in the fourth section these results are discussed and a general methodology is proposed, and finally some conclusions are derived.

2. 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):

This work has been partially supported by the Spanish Government's MEC HIPER project TEC2007-61802/MIC and the Andalusian Regional Government's EXC-2005-TIC-1023 project.

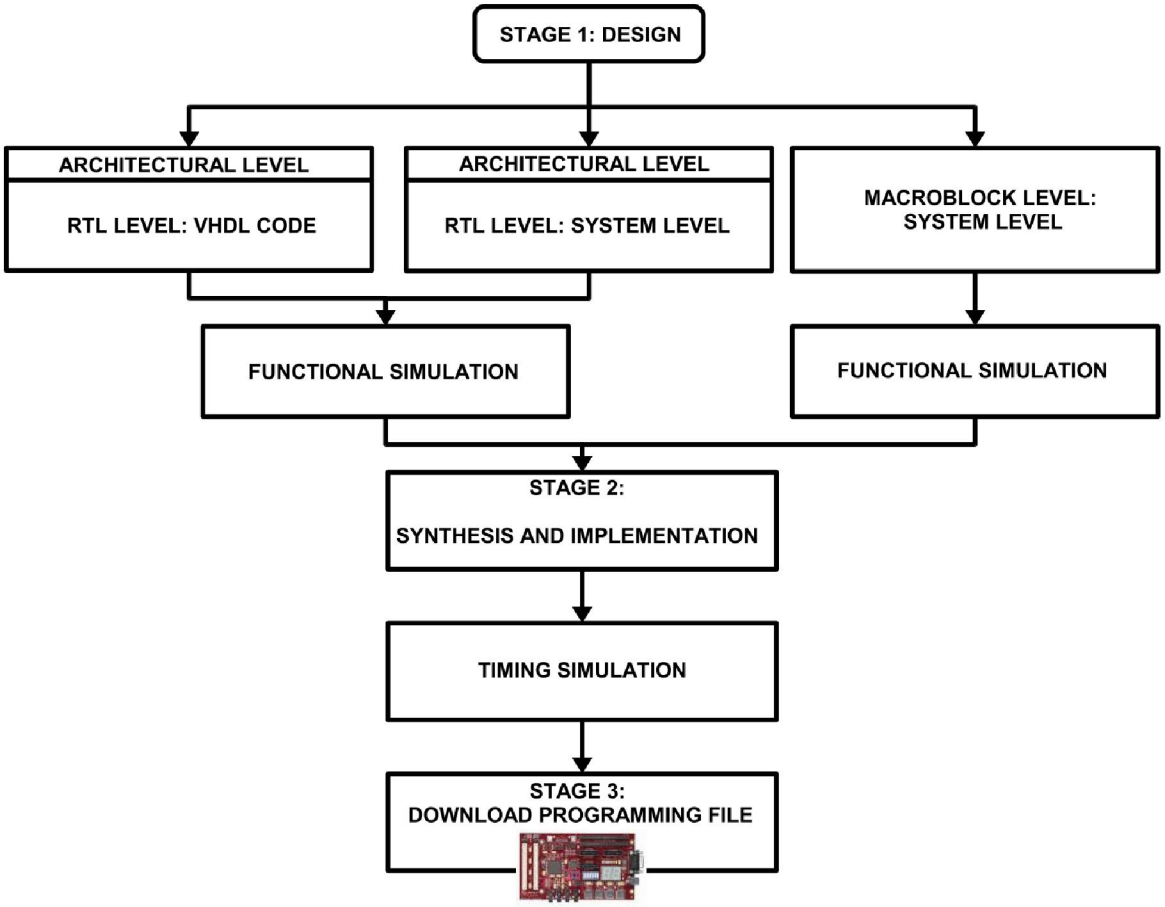

Fig. 1. Design and implementation workflow.

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 v8.2i 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.
3. Programming: finally, the device is configured by using iMPACT v8.2i from Xilinx.

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

2.1. 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 [3] and is depicted in Fig. 2. In this diagram, 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 [6]. Around this main element, there are several components which are described on the next:

- CONJs. These components carry out the conjugate operation of the complex data. Conjugators only operate in the IFFT case.
- RAM. This memory stores the intermediate calculations.
- ROM. This block supplies the twiddle values (precalculated).
- TWIDDLE. The output data of the RADIX-8 component are multiplied by the corresponding twiddle values at this component.
- CONTROL_UNIT. This unit is in charge of controlling the whole process and has been designed as a finite state machine according to the canonic structure described in [7].

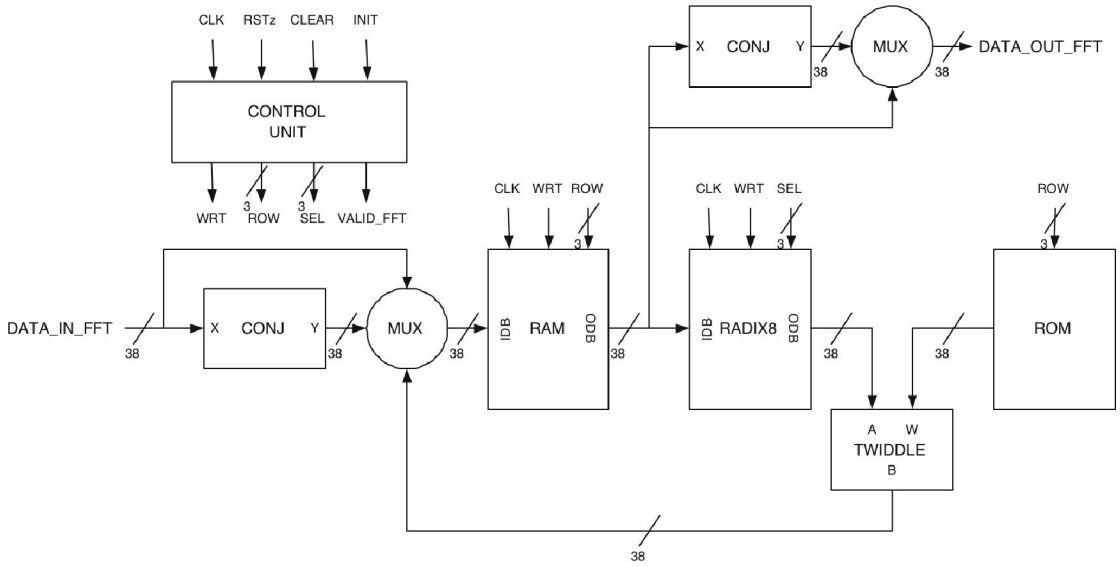

Fig. 2. Internal structure of the FFT/IFFT module.

A more detailed explanation on the structure and functioning of the VHDL implementation can be found in [2, 3].

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.

2.2. 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 [8]. 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.

Table 1. FFT macroblock specifications.

Implementation	Pipelined Streaming I/O
Number of sample points	64
Output ordering	Natural order
Scaling	Unscaled
Rounding mode	Truncation
Phase factor bit width	8

2.3. System-level tools at macroblock level methodology

The STM methodology consists of modeling the FFT/IFFT module using the System Generator's FFT block (Table 1). 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.

3. DESIGN RESULTS

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

3.1. Simulation results

In order to check that the designs work correctly, the following simulation process has been carried out. At the first

Table 2. Simulation results.

	VC	STR	STM
Clock cycles	291	291	262
Mean error	1.0%	1.0%	2.1%

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 the output values have been measured (Table 2).

As we can see, STM implementation consumes a lower amount of clock cycles for the calculation than VC and STR (from 291 to 262 cycles). However, both implementations reduce the output error with respect to the STM one (from 2.1% to 1.0%).

3.2. Hardware implementation results

In order to compare the implementations of the different FFT/IFFT designs, they have been synthesized separately with ISE, and implemented on a Virtex-II XC2V2000 FPGA. This device has all the features necessary to implement DSP functions: two million system gates, 56 embedded multipliers, and 56 Block RAMs.

In Table 3, 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 VC and STR implementations save about 4% slices with respect to STM design. Analyzing the maximum operation frequency, we can see that STM obtains the best result: 122 MHz as opposite of 40 MHz.

4. DISCUSSION

The purpose of this section is to analyze the results obtained and propose a specific methodology.

Firstly, on the one hand, simulation results show that the STM implementation reduces the amount of clock cycles necessary to the calculation from 291 to 262 cycles with respect to VC and STR (Table 2). On the other hand, VC and STR reduce the output relative error from 2.1% to 1.0% with respect to STM. This is due to the internal data width they employ: in the VC and STR implementations, designers fully control the internal structure and intermediate operation accuracy, whereas in the STM case, designers can only configure the input signal accuracy, what increases the output error.

Secondly, hardware implementation results show that VC and STR obtain the best results in terms of resource usage: they save about 4% slices with respect to STM (Table 3). This is because the system architecture designed is specifically suited for area optimization. In other way, we can see that STM achieves the highest maximum operation frequency: 122 MHz compared to 40 MHz of VC and STR. Moreover, if we consider that STM takes less clock cycles to calculate the results, this implementation could be a good alternative whether enough hardware resources are available and results obtained are accurate enough.

Comparing VC and STR, both designs produce similar results, except for the use of embedded multipliers and Block RAMs. Also, they reach an almost equal maximum operation frequency of 40 MHz. This is because both designs implement the same architecture (Table 3).

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 GUI (Graphical User Interface) tool, as for example register sets or decoders. Also, an important drawback is that some blocks can only be implemented on specific FPGA models.

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.

Finally, we have to conclude that STM is an interesting option if the available macroblocks are suited to the specific aims: it offers good results in terms of resource usage and operation frequency (but taking into account the possible data width restriction); besides, it avoids having to design the module, reducing the design time (Table 3). However, we have considered very important to maintain the control over the internal structure (due to the precision aspect), so

Table 3. Hardware implementation results on Virtex-II XC2V2000.

	VC	STR	STM
Slices	1187 (23%)	1188 (23%)	1393 (27%)
Slice Flip Flops	624 (6%)	624 (6%)	2041 (19%)
4 input LUTs	1984 (19%)	2030 (19%)	1380 (13%)
Bonded IOBs	58 (33%)	58 (33%)	57 (33%)
Block RAMs	2 (5%)	4 (10%)	3 (7%)
MULT18x18	12 (30%)	4 (10%)	7 (17%)
GCLKs	1 (6%)	1 (6%)	1 (6%)
Maximum operation frequency	39.53 MHz	40.15 MHz	122.65 MHz
Design, testing, and on-chip verification time	about 60 days	about 40 days	about 5 days

that 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 full control over the module internal architecture and obtain an efficient structure.

5. 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 and maximum operation frequency obtained, the last one has offered good results. However, such implementation has a main drawback: the internal data width cannot be controlled which yields to an important output error. 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.

6. REFERENCES

[1] L. Hanzo and T. Keller, *OFDM and MC-CDMA: A Primer*. Wiley-IEEE Press, 2006.

[2] 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.

[3] 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.

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

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

[6] 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.

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

[8] 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.