LOGIC AND PHYSICAL SYNTHESIS OF A SINGLE PRECISION FLOATING POINT UNIT

Edson Schlosser, Sidinei Ghissoni and Alessandro Girardi

Federal University of Pampa - UNIPAMPA
Alegrete - RS - Brazil

ABSTRACT

This paper presents the verification and synthesis plan applied to the design of a single precision floating point unit (FPU) that follows the IEEE 754 standard for the representation of binary real numbers. From a proposed RTL architecture we applied logic synthesis, physical synthesis, DRC analysis and verification using Synopsys tools. The development of the architecture was done with emphasis on applications requiring low power consumption and area constraints, such as embedded systems. The operations performed by the FPU are: addition, subtraction, multiplication and division, with emphasis on the reusability of blocks that perform basic functions.

1. INTRODUCTION

The floating point operations in most computer architectures are performed by different hardware components [1]. The component responsible for floating point arithmetic is referred to as Floating Point Unit (FPU).

The FPU in general is incorporated into the processing unit, and is used to accelerate the implementation of different calculations using the advantages of binary arithmetic in floating point representation, providing a considerable increase in computational performance.

The methodology used in this design is based on the IP-Process [2], adopted by the Brazil-IP program, in which this design is inserted. The IP-Process methodology divides the design of an IP-core in four phases: design, architecture, RTL design and prototyping. At the design stage functional and nonfunctional requirements are listed in order to define the project scope and acceptance criteria.

In the architecture phase the building blocks and connections are established, providing the basis for the implementation and verification. In RTL design, the architecture is described in synthesizable blocks. At this stage there is the inclusion of code for functional verification. Finally, the prototyping stage implements the design in a physical device. This paper aims to present the results of logical and physical level synthesis as well as verification for the design of a single precision Floating Point Unit whose architecture, proposed by [3], is shown in fig. 1.

The architecture developed also presents a RS232

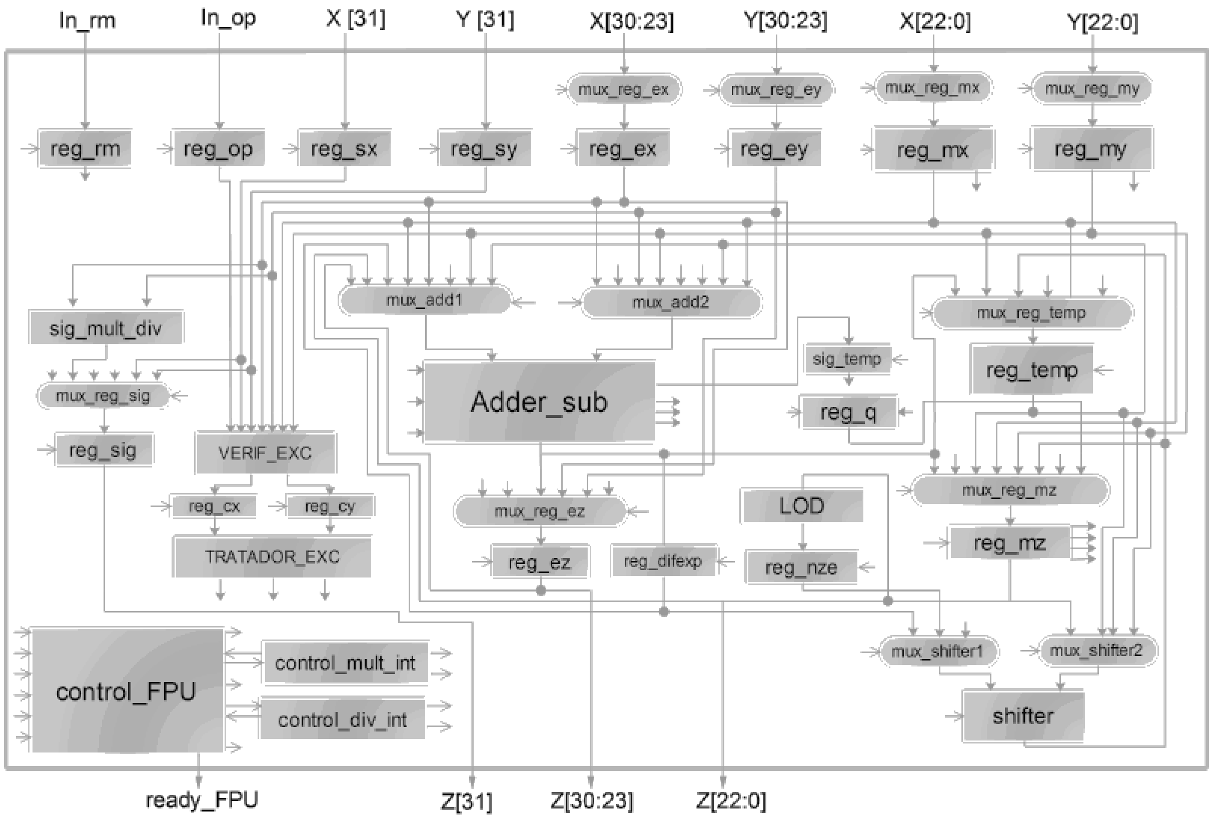


Figure 1 – Architecture Floating Point Unit

interface included, necessary to facilitate the testing and to reduce the number of inputs and outputs of the design.

This paper is organized as follows: In Section 2 is presented the synthesis procedure of the floating point unit. The results of implementations are showed in section 3. Finally in Section 4, is presented some conclusions of this work and ideas for future work.

2. SYNTHESIS PROCEDURE OF THE FLOATING POINT UNIT

Starting from a RTL code prototyped in FPGA and validated, we performed the logic synthesis and physical design and verification using Synopsys tools. In this design we used the design kit XH 0.35 from X-FAB [4]. Synthesis steps will be presented in the following sections.

2.1. Logic Synthesis

With the SystemVerilog description of the FPU architecture and the functionality of the blocks, we translated it into gate level using standard cells methodology. The mapping allows the choice of the size of logic gates, which is crucial in the final performance of the circuit. According to [5], the mapping technology is one of the most critical steps and has the greater impact on the final result, because it defines the construction and performance of the circuit, as well as power consumption.

In this methodology pre-designed cells are stored in libraries, and the layout and specifications are available for use in integrated circuit design. In this design we used libraries available from X-FAB for the technology XH 0.35.

Making use of the tool Design Compiler, the mapping can be done automatically with some configurations. The flow using to perform the logic synthesis can be seen in Figure 2.

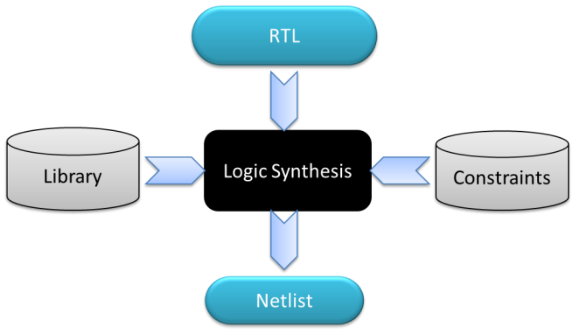


Figure 2 - Flow of logic synthesis using the Design Compiler tool.

The tool takes as input the RTL circuit description, as well as archives of technology XH0.35 and the constraints imposed by the designer. After mapping defined for the technology, the tool generates a synthesized design netlist containing the logical level.

2.2. Physical Synthesis and DRC Analysis

The physical synthesis generates the layout of the chip, producing the mask patterns and getting the physical representation of the integrated circuit to be prototyped.

Using the IC Compiler tool we performed the physical synthesis flow, following some steps until the generation of the GDSII file that contains the final design to be sent to the foundry for prototyping. The design flow of IC Compiler tool can be seen in Figure.3.

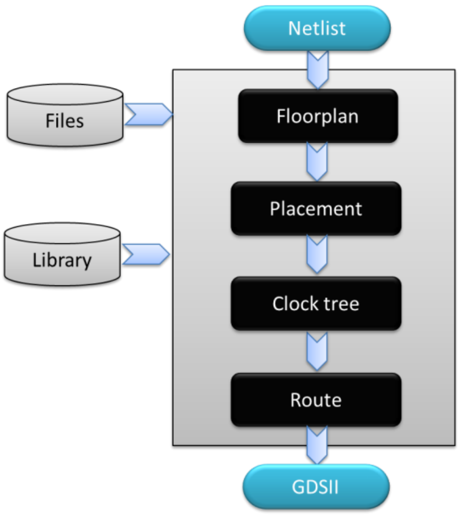


Figure 3 - Flow of physical synthesis using the IC Compiler tool.

The IC Compiler tool needs three input files for the synthesis.

The first file must contain the environmental conditions and constraints attached to the project, determined by the designer. This information is similar to those presented to the Design Compiler and are organized in a file with SDC extension obtained in the logic synthesis stage.

The second file is the netlist generated by Design Compiler, including I/O pads. According to [6] the input and output pads have an additional circuit for ESD protection. Also, the supply voltage pins (VSS and VDD) are included. We inserted four pairs of VDD and VSS in order to feed of rings around the pads, serving as a source for the I/O pads, and also to provide power to the core logic cells. They was inserted as a pair of voltage source on each side of the chip.

The third file the tool needs is the position of I/Os around the chip. An integrated circuit design may be core or pad limited. The design of the FPU described in this paper is core limited, because the data input and output is serial.

Filler pads have to be inserted for creating spaces between the pads in order to fill the empty space and maintain the connection of the supply lines of pads. To enter an exact number of pads fillers, it was necessary to know the width and height of the chip, as well as the total number of pads.

All pads fillers must accurately fill the empty space. For this, we perform an adjustment of the width and height of the chip in order to correctly enter the spaces between the fillers, because the width of the pads is fixed. The FPU includes 12 input/output pads, 8 supply pads and 4 corners. The width of these pads can be seen in Table 1.

Table 1 - Width of pads

Pad	Width
Corner	424.2 µm
I/O	89.6 µm
Filler	11.2 µm

To adjust the width and height of the chip it was necessary to check the original size of the chip in order to perform the calculation and adjustment of the exact number of fillers needed between the pads.

The data reported by the tool can be seen in Table 2.

Table 2 - Chip and core information

	Width (µm)	Height (µm)	Area (mm²)
Core	2088.8	2080	4.344
Chip	3131.2	3122.4	9.776
Pad Core	2282.8	2274.	5.191

The calculation of the adjustment was made using Eq.1.

$$\mathrm{nf} = \frac{\mathrm{X} - (\mathrm{nc} * \mathrm{Wc}) - (\mathrm{np} * \mathrm{Wp})}{\mathrm{Wf} * (\mathrm{np} + 1)} \qquad (1)$$

The value obtained in nf must be rounded up to get the exact number of fillers needed between each space. With Eq. 2 we can obtain the new width and height of the chip.

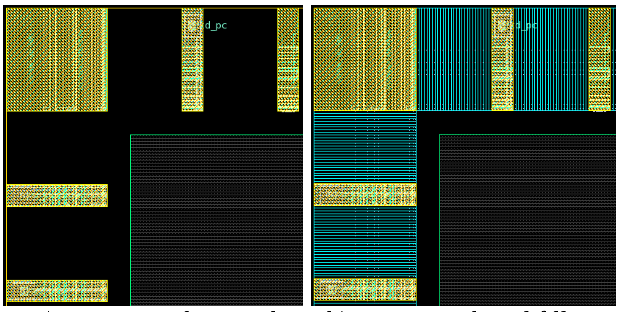
$$\mathrm{X}' = \mathrm{nf} * \mathrm{Wf} * (\mathrm{np} + 1) + \mathrm{np} * \mathrm{Wp} + \mathrm{nc} * \mathrm{Wc} \qquad (2)$$

The variables are the number of fillers per side (nf), the number of pads per side (np), the number of corners per side (nc), the width or height of the chip (X), the new width or height of the chip (X'), the width of the corner (Wc), the width of the pad (Wp) and width of the filler (Wf). We found a width of 3178.0 µm and a height of 3110.8 µm. Then, the width and height of the chip were changed to these values. Fillers were inserted between the spaces. Table 3 presents the new dimensions of the FPU design.

Table 3 - Chip and core sizes after filler insertion

	Width (µm)	Height (µm)	Area (mm²)
Core	2135	2067	4.413
Chip	3178	3110.8	9.886
Pad Core	2329.6	2262.4	5.270

The total area of the chip is about 10mm². Figure 4 presents the filler pads inserted between a corner pad and two pads I/O, noting that the space was completely filled.



a) Layout without pad fillers b) Layout with pad fillers

Figure 4 - Insertion of pad fillers.

The placement order of the pads, as well as the inclusion of power pads and fillers floorplaning defines the overall size of the chip, having a direct influence on the cost of the project.

The next step is the insertion of metal lines for the supply core. These lines are placed in the empty space between pad and core, in the form of metal rings, in order to maintain the distribution of energy to the core. Each side of the ring has a VDD and a VSS line, which are connected to power pads.

In addition to the rings, there are metal lines that cross the core vertically or horizontally. These lines, called straps, are connected to the power rings, and will supply the cells inside the core.

After the step of placing the metal supply lines, the next task is the placement of the standard cells. Here, all cells that were previously generated by the logic synthesis are placed into the core. Time constraints provided by the SDC file are taken into account, in order to the placement algorithm allocate related cells more closely, minimizing routing and delay. The required time constraints are evaluated in this step. However, even with the accounting of timing during the positioning phase, there are effects that cannot be overcome only with the positioning of cells and are analyzed in clock tree synthesis.

An effect that must be analyzed is the clock skew, which is characterized by the arrival of the clock signal at different times for different circuit components. One reason for this phenomenon is the large difference in distance traveled by the clock signal between the clock pad and different cells.

When this occurs, there is a problem with timing in the circuit, prejudicing the data processing. To solve the problem of clock skew, the clock tree generation creates alternative paths and inserts buffers so that the clock reaches the different blocks of the circuit at the same time.

The routing is the next design stage. It must connect all nets inside the circuit.

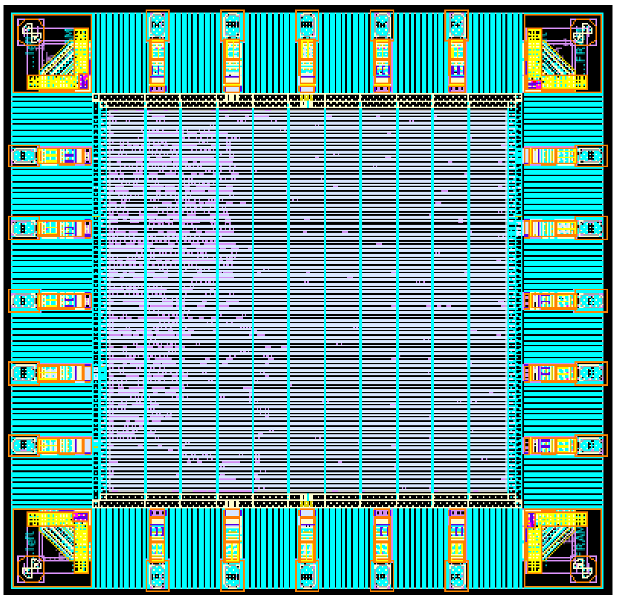


Figure 5 - Layout obtained after physical synthesis.

After the routing process we must perform the DRC and antenna analysis. For the DRC analysis we used Hercules tool, which validate the layout rules from the GDSII file extraction.

The layout obtained after all stages of physical synthesis can be seen in Figure 5.

3. RESULTS

Finishing the logic and physical synthesis, we present some results obtained in tables 4 to 7. The dynamic power consumption of the entire chip is estimated in 23mW.

The maximum frequency of operation is 17MHz and 37% of the chip is occupied by the core cells.

Table 4 – Core area

	Area (mm²)
Combinational area	1.862
Noncombinational area	0.321
Net Interconnect area	0.513
Total cell area	2.183
Total area	2.696

Table 5 – Results of power consumption

	Power consumption
Cell Internal Power	17.136 mW
Net Switching Power	5.906 mW
Cell Leakage Power	3.147 uW
Total Dynamic Power	23.043 mW

Table 6 – Frequency results

Frequency operation	17MHz
Slack	34.08ηs

Table 7 - Area utilization ratios

Cell/Core Ratio	49.477%
Cell/Pad Core Ratio	41.428%
Cell/Chip Ratio	37.056%

4. CONCLUSION

This paper presented the design flow for the development of an FPU applied to embedded devices. Making use of commercial tools from Synopsys, such as Design Compiler for logic synthesis, IC Compiler for physical synthesis, Hercules for DRC analysis and VCS tool for verification tests, the GDSII file is generated for sending to prototyping.

The results presented an operating frequency of 17MHz and a reduced power consumption. The total chip area was approximately 10mm².

As future work, we intend to verify the low power aspect that can be obtained by the reuse of the same architecture in comparison with others designs.

5. REFERENCES

[1] R.V.K Pillai, D. Al-Khalili, A.J. Al-Khalili, “A Low Power Approach to Floating Adder Design”, Proceedings of the 1997 International Conference on Computer Design (ICCD '97); 1997.

[2] M. S. M. Lima, F. S. D. Santos, J. F. B. Silva, E. N. S. Barros. “ipPROCESS: A Development Process for Soft IP-core with Prototyping in FPGA”. In: Forum on Specification and Design Languages (FDL), 2005, Lausanne. Forum on Specification and Design Languages (FDL). Lausanne: EPFL, 2005. p. 487-498.

[3] R. Neves, I. Castro; J. Prates, E. R. Schlosser, D. L. Prediger, S. Ghissoni, A. G. Girardi. “Implementação de uma Unidade em Ponto Flutuante para Operações Aritméticas em FPGA”. In: Iberchip 2010, 2010, Foz do Iguaçu. Anais do Iberchip, 2010.

[4] X-FAB Semiconductor Foundries. “Design Rule Specification XH035 - 0.35 µm Modular CMOS” Document DR_035_03, Release 3.3, September 2009.

[5] Marques F. D. S., Junior O.M., Ribas R. P., Junior L. S. D. R, Reis A. I. Mapeamento Tecnológico no Projeto de Circuitos Integrados Digitais. Cap: 8. Available in: http://www.inf.ufrgs.br/logics/docman/book_ufpel_marques.pdf

[6] G. D. Hachtel, R.K Brayton, A. L. S. Vincentelli.“ Multilevel Logic Synthesis”. Vol. 78, No. 2, Pag: 264-300, February 1990.