Microprocessors Datapath Design: Evaluating Complexity, Performance and Area

Thiago R. B. da Silva Soares
Universidade Federal do Piauí
Departamento de Informática e Estatística
thiagorbss@gmail.com

Laysson Oliveira Luz
Universidade Federal do Piauí
Departamento de Informática e Estatística
layssonoliveir4@gmail.com

Ramon Santos Nepomuceno
Universidade Federal do Piauí
Departamento de Informática e Estatística
ramonn76@gmail.com

ABSTRACT
Despite the wide availability of silicon area, the design of processing cores for multi-core architectures must search for efficiency regarding area, power and performance. For more reasons, a processing cores developed focusing FPGAs implementation must to search such efficiency. This paper presents a case study that aims to analyze the impact of using or not FPGA's embedded multiplier based on multi-core design in FPGA. For this work, VHDL descriptions of two processing cores were developed and analyzed with respect to area and performance. For the performance analysis, a highly multiplication-dependent application was chosen and executed in both processing cores. The paper shows that without FPGA's embedded multiplier it is possible to reach a reduction in area of 23% and increase the frequency in 37%, of course that with a penalty in performance.

Categories and Subject Descriptors
B.2.2 **[Performance Analysis and Design Aids]:** *Simulation – Verification, Worst-case analysis.*

General Terms
Measurement, Performance, Verification.

Keywords
Performace, Multiplication, FPGA, Embedded System.

1. INTRODUCTION
In the design of hardware cores for embedded system, factors such as frequency, power and silicon area are very important. In embedded systems design the higher frequency or the smallest silicon area not necessarily the best choices, the final system cost and power consumption are important factors that significantly impact the decision.

When FPGA is used as the target implementation technology for embedded system the area is measured in terms of FPGA's reconfigurable units usage. Essentially FPGAs consists of an array reconfigurable of basic reconfigurable units, used in the implementation of combinational and sequential logic and specific reconfigurable units, for implementation of multipliers, dividers and others functions. FPGA includes also reconfigurable I/O ports and interconnects. The cost of a single FPGA device is measured by factors including the amount and complexity of these reconfigurable units, and other factors not related with the study presented in this paper, such as manufacturing technology, packaging, and so on.

The use of FPGAs to implement an embedded system that requires multiple computational units implies the necessary analysis of the FPGA resource usage for each of these cores. In this case, depending on the complexity of the project, it makes sense to talk about *chip-area budget*, because the amount of FPGA reconfigurable units is limited and cannot be modified without modifying the FPGA device, family or FPGA provider. Probably any of these options results in increased cost.

This paper makes a comparative study focusing on the FPGA resources usage to implement a processing core and the performance reached, considering the maximum frequency and the number of cycles to perform a computational task, when FPGA's embedded multiplier are used or not.

The paper is organized into four sections. The second describes the processing cores used in the study. Section three presents the experiments and discusses about the results. Section four presents the conclusions and discusses about future work.

2. PROCESSING CORES
For the study presented in this paper two cores were developed and described in VHDL. Each of these cores was described in two versions. The first one uses FPGA's embedded multiplier and the second not.

The first processing core was developed for educational purposes and has a VHDL description based on the proposal presented in [1]. The second processing core has a RISC architecture (*Reduced Instruction Set Computer*) and was originally proposed by [2]. These two processing cores have very different characteristics and this is why they were chosen for this study.

The utilization of different processing cores make possible to analyze how the use or not of FPGA's embedded multipliers influence on the design and performance of a multicycle processing core and a pipelined one. In the next section it is possible to see that the results tables were provided to each processing core individually so that the conclusion should be made considering the differences between the two architectures.

2.1 uDIP
uDIP (UFPI's Didactic Processor) was develop with educational purpose and it have specific resources (hardware and instructions) to be used as processing core in a multi-core architecture. It is a 16-bit Harvard architecture with 32 general-purpose 16-bit registers. Its instruction set includes thirty-five instructions.

The uDIP is a multicycle processor and executes most of its instructions in three cycles, except for the memory access

instructions and the conditional jump instructions (when the condition is satisfied) that are executed in four cycles.

It is a RISC processor capable of addressing up to 16 bits and have two instruction formats, shown in Table 1 and Table 2.

Table 1. Format with immediate value.

15-10	9-0
Opcode	Immediate

Table 2. Format with two registers.

15-10	9-5	4-0
Opcode	Reg1	Reg2

The immediate field can be used as memory address or as operand in logic-arithmetic instructions. The fields reg1 and reg2 addressing one of the 32 registers available in the register file. These registers are used in arithmetic-logic instructions and also for memory access instructions.

To implement the shift instructions, necessary in this study, was used the lpm_clshift Megafunction, a configurable combinational logic shifter provided by Altera.

2.2 Risco

RISCO is a 32-bit processing core with 32 general-purpose registers and three pipeline stages [2]. The instruction set is coded in five formats, and originally did not include instructions for multiplication and division.

A word (instruction) has two bits (T0 and T1) to determine the type of instruction (logical-arithmetic, memory access, jump or subroutine), five bits (C0 to C4) to indicate the operation code (determines which operation will be performed), one bit (APS) to determine the update of the status register, three bits (F1, F2 and SS2) to determine the operands of the instruction, and the following less significant bits to indicate the operands. Figure 1 shows the instruction formats.

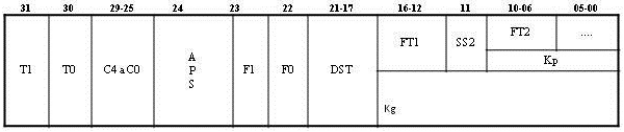

Figure 1. Instruction formats.

There are three different ways to use the immediate: Kpe, an 11-bits immediate, which is made an extension of the most significant bit, Kgl, with 16 bits, in which the extension is made in the same way as before, and Kgh, in which the lower 16 bits become more significant and the less significant 16 bits are filled with 0.

The uses of the immediate field are in Table 3.

Table 3. Using the immediate field.

F1	F2	SS2	Operation (DST <= A op B)
0	0	0	DST <= FT1 op FT2
0	0	1	DST <= FT1 op Kpe
0	1	X	DST <= R0 op Kgl
1	0	X	DST <= DST op Kgh
1	1	X	DST <= DST op Kgl

For this study three new instructions were added to the ISA: *mult, div e jpar*. The instruction *mult* performs integer multiplication, the *div* perform integer divisions and the *jpar* is a conditional jump based on the least significant bit (jump if zero). To the RISCO's instruction set it was also included inter-processors communication instructions. The inter-processor communication instructions are I/O-like instruction and allow that processors send or receive data through inter-processors communication ports.

3. EXPERIMENTS AND RESULTS

Both cores were implemented in VHDL and synthesized with Quartus II software for Cyclone III - EP3C16F484C6 device. To realize the necessary execution tests, a multiplication of two 8x8 matrices was developed and performed on the processors. Obviously this application proposes a worst case study because it uses intensively multiplication operations.

For the tests in which the processing cores had no hardware multipliers, an algorithm was developed using logical shifts and additions to perform the multiplication. The algorithm uses four general-purpose registers: Rx and Ry to be multiplied, Ra, used to control the amount of shifting operations and Rb, which is initialized with zero and get the final result.

The multiplication of Rx and Ry is performed as follows: if the least significant bit of Rx is 1 then Rb = Rb + Ry, after Rx is shifted one bit to the right and Ry is shifted one bit to the left. Otherwise (the least significant bit of Rx is 0), only the shift is performed. This procedure is repeated until all bits from Rx are analyzed.

The multiplications were made with 16 shifts in both processors and the values used in the matrices do not generate results higher than 255. Thus the matrix multiplication performed on both processing cores are comparable.

To get the desired results for the comparisons, the data collection was done in four steps:

Step 1: Synthesis of uDIP with hardware multipliers and execution of the matrix multiplication using the *mult* instruction. Synthesis and performance results are show in Table 4.

Step 2: Synthesis of uDIP without hardware multipliers and execution of the matrix multiplication using the shift-add algorithm. Synthesis and performance results are show in Table 5.

Step 3: Synthesis of RISCO with hardware multipliers and execution of the matrix multiplication using the *mult* instruction. Synthesis and performance results are show in Table 6.

Step 4: Synthesis of RISCO without hardware multipliers and execution of matrix multiplication using the shift-add algorithm. Synthesis and performance results are show in Table 7.

3.1 Results

The results of Tab. 4 and 5 shows a small reduction in the amount of logic elements used when the hardware multipliers was not used. The second column of Tab. 5 shows that FPGA's *9-bit Multiplier* was not used. This means a reduction in the use of FPGA resources, which can result in reduced energy

consumption. Without the hardware multipliers the maximum frequency increased approximately in 36%.

Despite the advantages obtained by reducing the use of FPGA resources and increasing the frequency, the number of cycles required for the execution of the matrix multiplication increased considerably. The percentage increase is approximately 736%. This increase is due to the multiplication algorithm used.

In Tab. 6 and 7, related to RISCO, the amount of logic elements decreased considerably and the 9-bit multiplier also was not used. However, the increase of frequency was not high as uDIP's and the number of cycles required for the execution of the matrix multiplication, in RISCO, has also increased. The percentage increase is, approximately, 629 %.

Table 4. uDIP with the multipliers.

Total logic elements	9-bit Multiplier	Clock's max frequency	Cycles
1.429 / 15.408 (9%)	4 / 112 (4%)	91.76 MHz	41081

Table 5. uDIP without the multipliers.

Total logic elements	9-bit Multiplier	Clock's max frequency	Cycles
1.382 / 15.408 (9%)	0 / 112 (0%)	126.01 MHz	343417

Table 6. Risco with the multipliers.

Total logic elements	9-bit Multiplier	Clock's max frequency	Cycles
5.367 / 15.408 (35%)	6 / 112 (5%)	77.81 MHz	48988

Table 7. Risco without the multipliers.

Total logic elements	9-bit Multiplier	Clock's max frequency	Cycles
4.124 / 15.408 (27%)	0 / 112 (0%)	83.31 MHz	356959

4. CONCLUSION AND FUTURE WORKS

This study demonstrated the advantages and disadvantages of using or not FPGA's embedded multiplier in the design of processing cores based in FPGA.

In the multicycle architecture, uDIP, the increase in frequency was quite satisfactory, despite the low reduction of logic elements and the number of cycles have increased greatly. In the pipelined architecture, RISCO, the increase in frequency was low, the number of cycles also increased greatly, but the number of logic elements decreased considerably.

4.1 Future works

As future work it is necessary to analyze the processing core's performance for more application (real applications). These application needs to have different profiles considering the amount of multiplication required to the execution. Also, it is necessary to analyze the energy consumption. Finally the processing cores will be integrated into a multi-core architecture and more execution tests must be executed.

5. REFERENCES

[1] Hamblen, J. O., Hall, T. S. and Furman, M. D, 2005, Rapid Prototyping of Digital Systems, Ed Springer.

[2] Junqueira, A. A. 1993, RISCO – 32-bit Microprocessor RISC CMOS, Master's Degree Thesis, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, RS.