OPTIMIZATION OF THE MOVE ARCHITECTURE APPLIED TO DSP UTILIZING BIT-SERIAL MULTIPLIERS

Igor Gavazzi Vazzoler
Departamento de Engenharia Elétrica
Universidade Federal de Santa Catarina
CEP 88040-900 Florianópolis – SC – Brasil
igorgv@grad.ufsc.br

Luigi Carro
Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Sul
CEP 90035-190 Porto Alegre – RS – Brasil
carro@iee.ufrgs.br

ABSTRACT

This work presents the results of a research about the optimization of a single instruction processor (MOVE) applied to DSP. The intended optimization goes both in the way of the occupied area as in the way of processor's performance. The theory is that such optimization can be reached through utilization of bit-serial multipliers working in parallel. The efficiency of the implemented architecture is measured through a simulation of a digital signal filter. The results of this simulation and the technical results of the realized implementation were compared to the results of a previous work about the MOVE architecture that hasn't the bit-serial multipliers. The description of the architecture was developed in VHDL and the synthesizing in FPGA.

1. INTRODUCTION

As important as developing a system that performs a given task is to develop a low-cost and performance-optimized system to perform that task. In digital signal processing, one of the choices to decrease the cost was presented in [1]. Such work showed the possibility of the implementation of digital signal processors based on the MOVE architecture, which have particular features such as simplicity and flexibility. A MOVE architecture processor is a processor that executes only one instruction, move, which copies a data from one memory position to another. The simplicity of this processors results from the fact that its logic is trivial in comparison with the general purpose processor's logic or even with the DSP processor's logic. The MOVE architecture's flexibility results from its high reconfigurability. In this architecture, all the operators are mapped in memory positions and this mapping can be freely changed without producing any direct change in the processor's structure, but just in its programming mode. This way, the system can be adapted to the kind of application where it will be used, and the processor itself stands the same, executing only moves.

This work is organized as follows: section 2 presents the proposed idea to obtain the intended optimization. Section 3 presents the reasoning used to find an equation that gives the great number of bit-serial multipliers to use in the data path. The sections 4 and 5 describes the implementation and the simulation realized to prove that the idea was correct, while the section 6 exposes the results obtained with such practical development. The conclusions and suggests for continuity of this work are in section 7.

2. THE UNDERLYING IDEA TO REACH OPTIMIZATION

The proposed choice in this work to obtain a better performance is to replace the multiplier utilized primarily by a group of bit-serial multipliers working one in parallel with the others. The parallel multiplier utilized in the previous work doesn't depend on clock synchronization to operate and have a relatively small response time, but greater than the memory response time. The bit-serial multipliers depend on clock synchronization to operate and spend several cycles to give a valid response, but each processing step requires a time much smaller than the parallel response times. This way, the entire system can run with a higher frequency. Besides, the bit-serial multiplier occupies an area usually 5 times smaller than the area occupied by the parallel multiplier. This results in resources economy.

In the specific case of using the MOVE architecture applied to DSP, there is need of arithmetic operators in the data path. This fact occurs because the DSP algorithms perform many mathematical operations. The research presented in [1] showed an implementation which had one adder, one subtracter and one parallel multiplier. In that case, the system has its frequency limited by the parallel multiplier within the data path.

3. THE OPTIMUM NUMBER OF MULTIPLIERS

The optimum number of multipliers to add into the data path depends on the architecture's word size. To obtain the best multipliers utilization rate, the developed program must obey to two criteria:

- Read the multiplication's result just after its finish: in order to avoid letting the multiplier inactive, the developed program must read its result as fast as possible after the cycle when this becomes available.
- Make an efficient use of the cycles which precedes the reading of a multiplication result: the instructions between the MOVE instruction which started and read the result must be utilized to start multiplications in the other multipliers.

It is necessary to obtain an equation that gives this optimum number of multipliers to be placed in the data path. That number must fulfill the two affirmations, so the following reasoning was done: to start a multiplication, two MOVE instructions are needed, one to move the first operand and other to move the second operand to its respective memory positions (the multiplier's inputs). Therefore, four clock cycles are needed to start a

multiplication, since each MOVE instruction takes two cycles to be realized. The number of cycles that the multiplier spends to give a valid result in its output is equal to its word size (p). So, in this time is possible to realize p/2 MOVE instructions. With this number of multiplications is possible to start (p/2)/2 or p/4 multipliers. Adding this number with the multiplier which was initially initialized, it is obtained:

$$Nm = (p / 4) + 1 \quad (1)$$

Where "Nm" is the optimum number of multipliers to instantiate into the data path and "p" is the word size or the number of bits that the multipliers work with.

In the case of the implemented MOVE architecture, where p is equals to 8, Nm is:

Nm = (8/4)+1 = 3

4. IMPLEMENTATION

The development tool utilized in the implementation was the Altera's MAX-PLUS II. This software has a wide library of parameterized functions that were utilized in the data path and in the bit-serial multiplier, just as shift-registers, adders and RAM blocks. In the previous work, the utilized multiplier was from this library, but in the current work the multiplier was developed specially to the implementation.

5. SIMULATION

One aspect of the intended optimization in this work is relative to its performance. To measure the new architecture's performance and make comparisons with the previous results, were utilized a program developed in the previous work, but adapted to the new programming mode. The program is a FIR filter of 10 coefficients. As in the previous work, the architecture was built with a word size equals to 8 bits, therefore, some approximations were made in order to make possible the architecture's performance analysis.

The simulation was done by the simulator of Altera's MAX-PLUS II, and the results were caught through an analysis of the resulting waveform.

6. RESULTS AND COMPARISONS

The table 6.1 presents the technical results of the developed implementation and compares these values with the values obtained in the previous work. These data are relative to the OPERAT block, which contains the bit-serial multipliers.

	Previous work	Current Work
Logic Cells	412	298
Utilized Area	35%	25%
Maximum Frequency	9.14 MHz	20.79 MHz
Number of Multipliers	1	3
Multiplier's response time	69 ns	384 ns

Table 6.1 – Technical comparisons between the two realized works

The table 6.1 shows that in relation to the inherent architecture's cost, the underlying idea was correct. This is evidenced by the reduction of the utilized area. The current work reached a reduction of 27.7%.

The table 6.2 presents the results obtained with the simulation of the FIR digital signal filter in both works.

	Previous work	Current work
Number of instructions	90	90
Number of cycles	180	180
Clock Period	109.4 ns	48.1 ns
Filter's response time	19.69 us	8.65 us
Sample Rate	50.78 KHz	115.61 KHz

Table 6.2 – Performance comparisons

The information in the table 6.2 points to success in the assumption of performance optimization. Despite the bit-serial multiplier be much slower than the parallel multiplier, the group of bit-serial multipliers made the global result better. In this case the optimization is measured through the increase of the filter's sample rate simulated. The new sample rate is 127.7% faster than in the previous implementation.

7. CONCLUSIONS AND FUTURE WORKS

This work reached its goal and showed that the utilization of bit-serial multipliers working one in parallel with the other can increase significantly the MOVE architecture's performance when it's applied to DSP. The work showed a utilized area reduction too, causing a cost decreasing.

An interesting suggestion for continuity of this work is the development of a compiler specific to the MOVE architecture. This compiler would be designed to be a DSP specific compiler or even a high level language, such as C. This way, generic programs would be designed and executed with the MOVE processor, and its performance as a generic purpose processor could be analyzed.

REFERENCES

[1] J. C. B. de Mattos, D. T. Franco, L. Carro, and A. A. Suzim, "MOVE Architecture applied to DSP"

[2] E. E. Fabris, G. A. Hoffmann, A. A. Susim, and L. Carro, "A bit-serial FFT processor"

[3] P. J. Ashenden, *The Student's Guide to VHDL*, Morgan Kaufmann Publishers, San Francisco - USA, 1998.

[4] De Micheli, G., *Synthesis and Optimization of Digital Circuits*, MacGraw-Hill, USA, 1994.

[5] S. W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, San Diego - USA, 1999.

[6] Altera Corporation, *Data Book*. San Jose: Altera Corporation, 1996.