Low Power Architectures for FFT and FIR Dedicated Datapaths

Eduardo Costa
UCPel
Pelotas, Brazil

Sergio Bampi
UFRGS
P. Alegre, Brazil

José Monteiro
IST/INESC-ID
Lisbon, Portugal

ABSTRACT

This paper addresses the use of architectural transformations for the low power realization of FIR filter and FFT algorithms on dedicated datapath architectures. We report significant power savings using the propose methodology. New low power arithmetic operators are used as basic modules. In FIR filter and FFT algorithms, 2's complement is a widely used encoding for signed operands. We use a new architecture for signed multiplication, which maintains the pure form of an array multiplier. This architecture uses radix-2^m encoding, which leads to a reduction of the number of partial lines, enabling large gains in performance and power consumption. The proposed architecture is applied to the DSP architectures and compared with the state of the art. Due to the characteristics of the FIR filter and FFT algorithms, which involve multiplications of input data with appropriate coefficients, the best ordering of these operations in order to minimize the power consumption in the implemented architectures is also investigated.

I. INTRODUCTION

This paper focuses on power optimization techniques at the architectural level applied to Digital Signal Processing (DSP) systems [1], [2], [3], [4], [5]. In our work, FIR filter and FFT computations are addressed through the implementation of dedicated architectures, where the main goal is to reduce the power consumption by using transformation techniques.

Since multiplier modules are common to many DSP applications, one of the low power techniques used in this work is the use of efficient multiplier architectures [7] in the dedicated DSP architectures in order to reduce their switched capacitance. As observed in this paper, DSP architectures that use the multiplier of [7] are more efficient than those that use the common Booth multiplier. Power savings above 40% are achievable in the FFT architecture using array multiplier of [7]. This power reduction is mainly due to the lower logic depth in the multiplier circuit, which has a big impact on the reduction of the glitching activity in the FFT architectures.

In this paper, the low power arithmetic modules are experimented in different dedicated FIR filter and FFT architectures. In the FIR implementations, combinations of Fully-Parallel, Fully-Sequential and Semi-Parallel architectures with simple, transposed and pipelined version are explored. For the FFT algorithm, Fully-Sequential and Semi-Parallel architectures with simple and pipelined version are implemented.

Additionally, we propose an extension to the Coefficient Ordering technique [8] that aims at reducing the power dissipation by optimizing the ordering of the coefficient-data product computation. We have used this technique in the FIR and FFT implementations. As will be shown, the manipulation of a set of coefficients can contribute for reducing the power consumption in the dedicated architectures.

This paper is organized as follows. In Section 2, we present the dedicated FIR filter and FFT implementations. An overview of relevant work related to power optimization in FIR filter and FFT realization are shown in Section 3. Section 4 describes the low power techniques used in this work. Performance comparisons between the architectures for the different low power techniques are presented in Section 5. Finally, in Section 6 we discuss the main conclusions of this work.

II. DEDICATED DSP DATAPATH IMPLEMENTATION

We present Fully-Parallel and Fully-Sequential FIR filter architectures, both in three versions: Pipelined, Non-Pipelined and Transposed. The Pipelined and Non-Pipelined version are also explored for the Fully-Sequential FFT implementation. Additionally, we present for both DSP implementations a Semi-Parallel architecture which improves performance over the Fully-Sequential architecture by using duplicated hardware and thus being able to compute two partial products at a time. These different datapath architectures are compared with implementations that are 16-bit wide and use as examples: i) an 8-order FIR filter ii) a 16-point radix-2 common factor FFT with decimation in frequency. As should be emphasized, although we have presented FIR and FFT examples with a low number of coefficients, the techniques described in this work can be applied to architectures with any coefficient order. The limitation is that power estimates for these more complex architectures are more difficult to compute.

A. FIR Filter Architectures

FIR filtering is achieved by convolving the input data samples with the desired unit impulse response of the filter. The output $Y[n]$ of an N-tap FIR filter is given by the weighted sum of the latest N input data samples $X[n]$ as shown in Equation 1.

$$Y[n] = \sum_{i=0}^{N-1} H_i X[n-i] \qquad (1)$$

The most direct method of FIR filter implementation is the Fully-Parallel Direct Form, where delay units are used to store

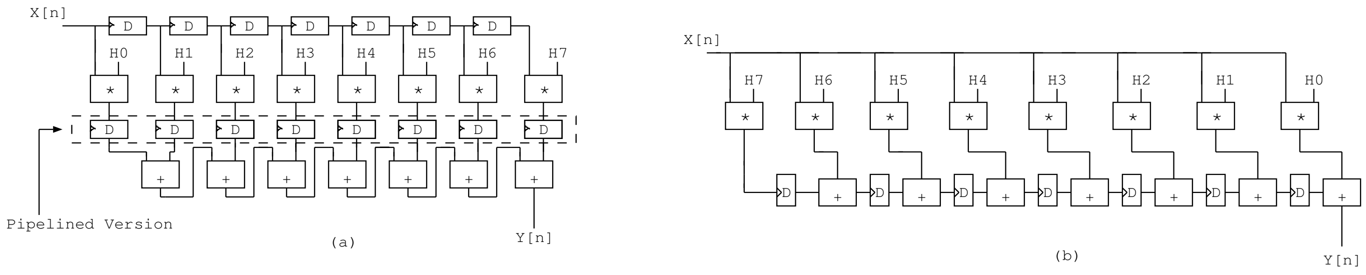


Fig. 1. Datapath of FIR Filter Fully-Parallel Implementations.

previous input samples as shown in Figure 1(a).

In the Direct Form FIR filter implementation, in each clock cycle a new data sample and the corresponding filter coefficient are simultaneously applied to each multiplier. The results of all multipliers are added simultaneously, producing considerable glitching at the primary outputs [3].

In our work, we address this problem by implementing two alternative Fully-Parallel architectures, called Pipelined and Transposed forms, as shown in Figure 1(a) and Figure 1(b) respectively. As can be observed in Figure 1, the Fully-Parallel implementations present a large number of multiplier modules, which can increase area, delay and power consumption in the architectures, depending on the type of operator used. In order to optimize these parameters in the Fully-Parallel architectures, we have used in this work shift-add circuits for the multipliers, which is possibly because one of the inputs is constant.

The second type of datapath implementations considered in this work are Fully-Sequential architectures, as a manner to reduce hardware requirements for the FIR filter algorithm, shown in Figure 2. In the sequential implementation the basic idea is to reduce hardware requirements by re-using as much of the hardware as possible. We have experimented the Direct Form (Non-Pipelined and Pipelined) and Transposed Fully-Sequential implementations, as shown in the Figures 2(a) and 2(b).

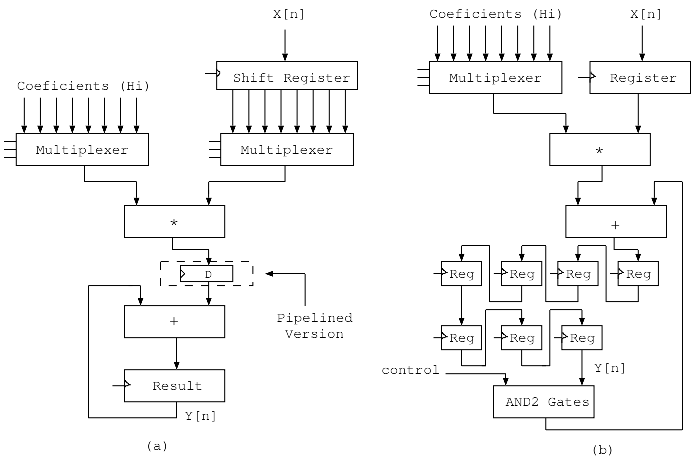


Fig. 2. Datapath of FIR Filter Fully-Sequential Implementations.

In order to enable an intermediate alternative between Fully-Parallel and Fully-Sequential implementations, we have experimented a Semi-Parallel architecture. In this architecture, shown in Figure 3, hardware requirements are duplicated with respect to the Fully-Sequential, allowing two samples to be processed simultaneously. Again, we have constructed three versions of the Semi-Parallel architecture.

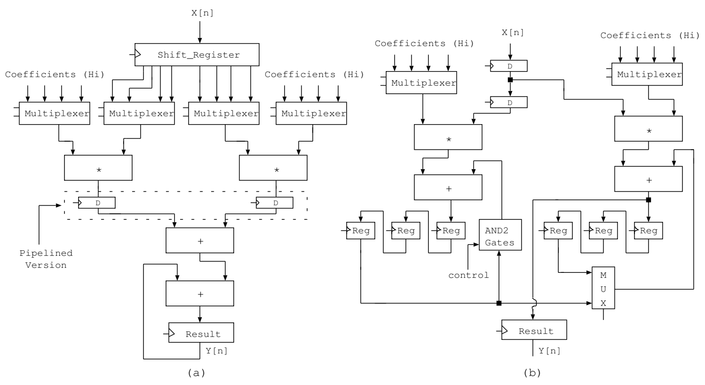


Fig. 3. Datapath of FIR Filter Semi-Parallel implementations.

B. FFT Architectures

The FFT algorithm allows for an efficient computation of the Discrete Fourier Transform (DFT) [10]. The hierarchical computational blocks in the FFT structure are stages, groups, and butterflies. Each stage requires the computation of groups, and each group requires the computation of butterflies. The butterfly plays a central role in the FFT computation. For the common factor FFT algorithm with decimation in frequency, the butterfly allows the calculation of complex terms according to Equations 2 and 3.

$$C_{complex} = A_{complex} + B_{complex} \tag{2}$$

$$D_{complex} = (A_{complex} - B_{complex}) \times W_{complex} \tag{3}$$

As can be observed in the equations above, one complex addition, one complex subtraction and one complex multiplication are involved in the butterfly block. The arithmetic operators for the complex operation are shown in the Figure 4 for a Fully-Sequential FFT implementation. In this figure, the arithmetic operators present in the butterfly block, enable the calculation of the real and imaginary parts. The results of these calculation are stored in appropriate register banks shown in the left side and right side of the Figure 4 for the real and imaginary parts respectively. The set of multiplexers shown in this figure select the appropriate values to be stored in the register banks. Several modules of ROM are required for the storage of twiddle factors.

We have omitted these modules to minimize the complexity of Figure 4.

The presence of a large number of multiplier operators in the FFT architecture leads to a significant amount of glitching in a transform computation. Thus, we have implemented a pipelined version with the insertion of registers at the multiplier outputs, as shown using the dotted lines in the Figure 4.

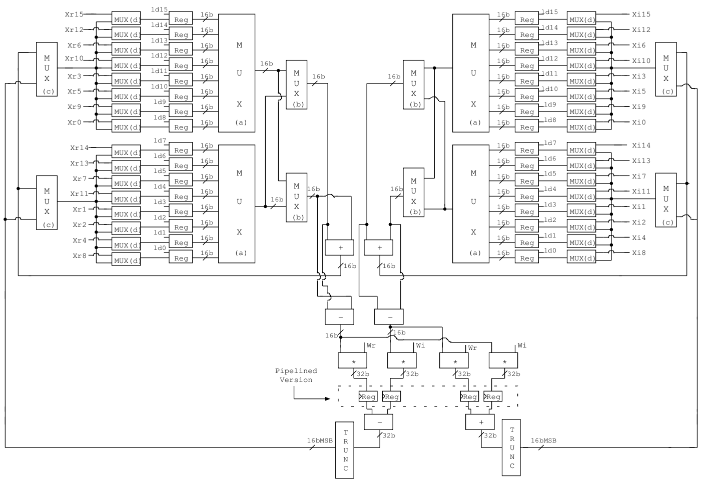


Fig. 4. Datapath of FFT Fully-Sequential Implementations.

In a 16-point FFT algorithm, it is not viable to produce a Fully-Parallel version due to the large number of arithmetic operators present in the butterfly block (32 butterflies operating simultaneously implies 512 arithmetic operators). On the other hand, in a Fully-Sequential implementation 32 real and 32 imaginary terms are performed in the butterfly (4 stages with 8 butterfly). Thus, 33 clock cycles are necessary for a full calculation in the FFT architecture (1 cycle for the 16 point load and 32 cycle for a transform computation in the butterfly).

In order to speed-up the FFT calculation, we have implemented a Semi-Parallel architecture, presented in Figure 5. In this architecture, hardware requirements in terms of arithmetic operators are duplicated with respect to the Fully-Sequential, because two butterfly are used and two transforms can be performed simultaneously. Thus, the full transform calculation is performed using half of the cycles used in the Fully-Sequential version. Again, we have implemented two versions of the Semi-Parallel architecture (Direct and Pipelined form), as shown in Figure 5.

III. RELATED WORK ON DSP OPTIMIZATION

Various architectures have been used in FIR filter and FFT realizations, where implementations in programmable DSP and hardwired architectures are addressed [2], [4], [11]. For applications where the flexibility of the programmable processor is not required, a hardwired implementation is the preferred choice as such an implementation typically results in higher throughput and lower power [8].

For the hardwired implementation, architectural transformations have targeted performance, power and computational complexity [8]. A very efficient technique when targeting low power

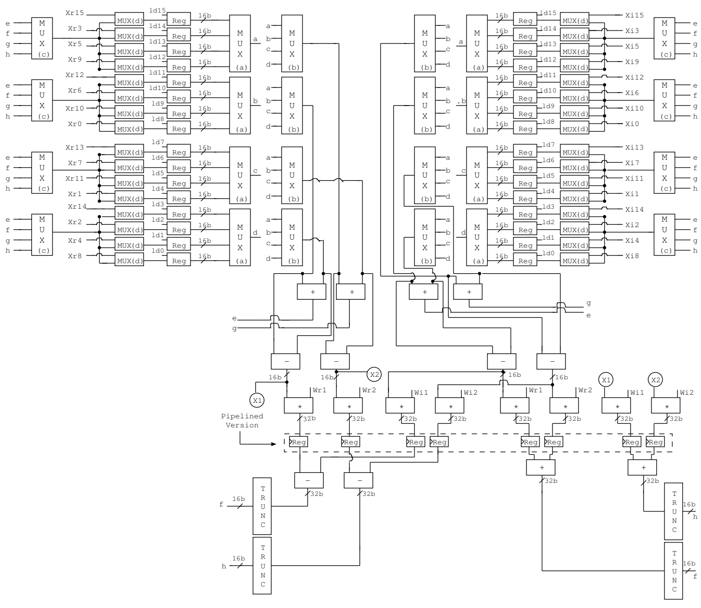


Fig. 5. Datapath of FFT Semi-Parallel Implementations.

consumption is to reduce the supply voltage, resulting in a power reduction proportional to the square of the reduction in the supply voltage. With the same objective, parallel processing and pipelining have been applied to the implementation of FIR filters and FFT architectures [5], [12], [13], [14], [15] as a form of recovering the performance loss due to a lower supply voltages.

The work proposed in this paper will build on some of the transformation approaches mentioned, specially the techniques that target the increase in performance and switching activity reduction. In particular, similar transformations will be essayed on dedicated FIR filter and FFT architectures. In our work, we experiment the use of low power arithmetic operators in the dedicated architectures. The use of pipelining approach in the FIR filter and FFT architectures is also investigated.

In the FIR filter operation, the output is performed by a summation of data-coefficient products. Thus, some techniques called Coefficient Ordering, Selective Coefficient Negation and Coefficient Scaling have addressed the use of coefficient manipulation in order to reduce the switching activity in the multipliers inputs [1], [8]. The main goal of these techniques is to minimize the Hamming distance between consecutive coefficients in order to reduce power consumption in the multiplier input and data bus. The technique is only applied to a Fully-Sequential architecture. In our work an extension of the Coefficient Ordering technique is experimented in the FIR and FFT architectures. The proposed technique can be applied to both Fully-Sequential and Semi-Parallel architectures.

IV. LOW POWER TECHNIQUES

This section presents different low power techniques that will be experimented in the dedicated datapath architectures for DSP. The reduction of switching activity is addressed by using low power arithmetic operators and the manipulation of the filter and FFT coefficients.

A. Low Power Arithmetic Operators

In this section, we summarize the methodology of [7] for the generation of regular structures for arithmetic operators using signed radix-2^m representation.

A.1 2's Complement Radix-2^m Multiplier Architecture

For the operation of a radix-2^m multiplication, the operands are split into groups of m bits. Each of these groups can be seen as representing a digit in a radix-2^m. Hence, the radix-2^m multiplier architecture follows the basic multiplication operation of numbers represented in radix-2^m. The radix-2^m operation in 2's complement representation is given by Equation 4.

$$\begin{aligned} A \times B &= A' \times B' - A' b_{W-1} b_{\frac{W}{m}-1} 2^{W-m} \\ &\quad - a_{W-1} a_{\frac{W}{m}-1} \sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \end{aligned} \qquad (4)$$

This operation is illustrated in Figure 6.

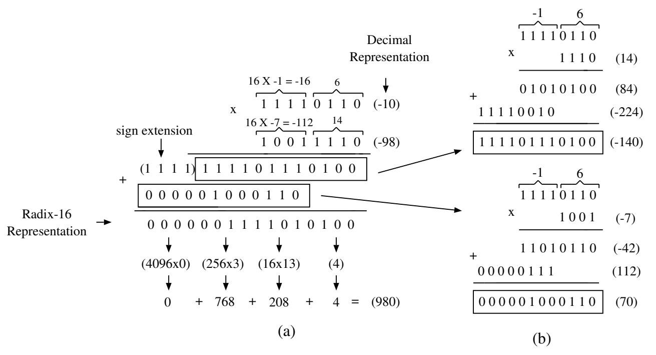


Fig. 6. Example of a 2's complement 8-bit wide radix-16 multiplication.

For the $W - m$ least significant bits of the operands unsigned multiplication can be used. The partial product modules at the left and bottom of the array need to be different to handle the sign of the operands.

For this architecture, three types of modules are needed. `Type I` are the unsigned modules used in the previous section. `Type II` modules handle the m-bit partial product of an unsigned value with a 2's complement value. Finally, `Type III` modules that operate on two signed values. Only one `Type III` module is required for any type of multiplier, whereas $2\frac{W}{m} - 2$ `Type II` modules and $(\frac{W}{m} - 1)^2$ `Type I` modules are needed. We present a concrete example for $W = 8$ bit wide operands using radix-16 ($m = 4$) in Figure 7.

A.2 Modified Booth Multiplier

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in [16]. In this architecture it is possible to reduce the number of partial products by encoding the two's complement multiplier. In the circuit the control signals (0,+X,+2X,-X and -2X) are generated from the multiplier operand for each group of 3-b as shown in the example of Fig. 8 for a 8 bits wide operation. A multiplexer produces the partial product according to the encoded control signal.

Common to both architectures is that at each step of the algorithm two bits are processed. However, the basic Booth cells

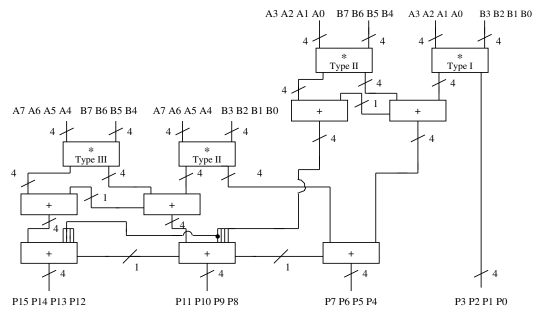
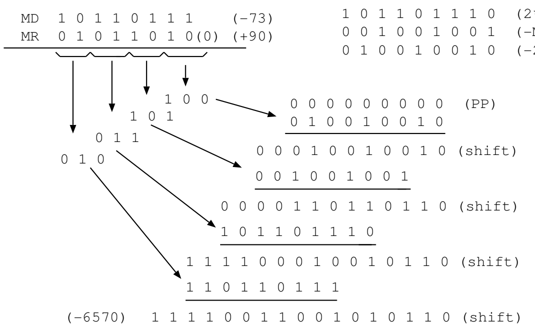


Fig. 7. Example of a 8-bit wide 2's complement radix-16 array multiplier.



MD 1 0 1 1 0 1 1 1 (-73)
MR 0 1 0 1 1 0 1 0(0) (+90)
1 0 1 1 0 1 1 1 0 (2*MD)
0 0 1 0 0 1 0 0 1 (-MD)
0 1 0 0 1 0 0 1 0 (-2*MD)
1 0 0
1 0 1
0 1 1
0 1 0
0 0 0 0 0 0 0 0 0 (PP)
0 1 0 0 1 0 0 1 0
0 0 0 1 0 0 1 0 0 1 0 (shift)
0 0 1 0 0 1 0 0 1
0 0 0 0 1 1 0 1 1 0 1 1 0 (shift)
1 0 1 1 0 1 1 1 0
1 1 1 1 0 0 0 1 0 0 1 0 1 1 0 (shift)
1 1 0 1 1 0 1 1 1
(-6570) 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 1 0 (shift)

Fig. 8. Example of a 8-bit wide Modified Booth multiplication.

are not simple adders as in the proposed array multiplier, but must perform addition-subtraction-no operation and controlled left-shift of the bits on the multiplicand. Besides taking more area, this complexity also makes it more difficult to increase the radix value in the Booth architecture.

In [7], it is reported that the radix-4 array multiplier can be significantly more efficient in terms of power consumption than the Modified Booth multiplier, allowing for about 50% power savings.

B. Coefficient Manipulation

Coefficient ordering can be used as a technique for low power because, in a FIR filter computation, the summation operation is both commutative and associative. Thus, the filter output is independent of the order of computing the coefficient product [8]. Coefficient ordering is used in [8] as a technique for low power, where all coefficients are ordered in a Fully-Sequential circuit so as to minimize the transitions in the multiplier input and bus. In our work we have experimented an extension of this technique in a Semi-Parallel architecture, where the hardware is duplicated and coefficients are partitioned into groups of coefficients. Thus, the problem is related to finding the best coefficient partition by calculating the minimum Hamming distance between the coefficients that fall in each partition.

The pseudo-code presented in Figure 9 describes an algorithm that optimizes the partitioning and ordering of the coefficients. In this algorithm, the cost function is calculated for all the combinations over the coefficients. For the FIR and FFT architectures used in this work, the total number of permutations is still reasonable. However, for a higher number of coefficients this exhaustive algorithm is less attractive due to the time necessary

```
for all permutations of coefficients H(0-7){
    partition1=Hamming((H[0],H[1]) + (H[1],H[2]) +
            (H[2],H[3]) + (H[3],H[0]));
    partition2=Hamming((H[4],H[5]) + (H[5],H[6]) +
            (H[6],H[7]) + (H[7],H[0]));
    cost function = partition1 + partition2;
    if (cost function < minimum found) {
        save current partition;
        minimum=cost function;
    }
}
```

Fig. 9. Pseudo-code of the algorithm for the generation of coefficient partitioning and ordering.

to process the large number of combinations. In this case, an heuristic algorithm should be used to get as near as possible to the optimal solution.

V. RESULTS

In this section, we discuss the impact of the proposed low power techniques on dedicated pipelined FIR filter and FFT architectures. Area, delay and power consumption for each architecture are presented. Area is given in terms of the number of literals. Delay values were obtained in SIS environment [17] using the general delay model from the `mcnc` library. This parameter defines the minimum clock period. Power results were obtained with the SLS tool [9] using the general delay model. For the power simulation, we have applied a random pattern signal with 10,000 input vectors represented in 2's complement. For power consumption comparisons, we chose to compute the power dissipation per sample for the FIR filter and the power dissipation per transform for the FFT.

A. Application of the Low Power Arithmetic Operator

In this section, we present results on use of the array (m=2) arithmetic operators of Section IV-A in the FIR and FFT architectures. Area, minimum clock period and power consumption are investigated and compared to the architectures with Modified Booth operator.

A.1 Area

Table I presents area results for FIR filter and FFT architectures using the array (m=2) and Modified Booth operators. As can be observed in this table, there is significant area difference between the architectures with these operators. The Fully-Sequential and Semi-Parallel architectures which use the array multiplier operators present slightly more area. This due to the fact that array multipliers require more area than Booth circuits.

TABLE I. Area results for the pipelined architectures.

Architecture		Operators		Difference(%)
		Booth	Array	Array vs. Booth
Fully-Sequential	FIR	6427	7329	+14.0
	FFT	24099	29867	+23.9
Semi-Parallel	FIR	10569	12437	+17.7
	FFT	46000	53246	+15.7

A.2 Minimum Clock Period

Although FIR filter and FFT architectures with the array operators present higher area, these architectures permit a lower clock period than the architectures with Booth operators, as shown in Table II. This reduction occurs because in the Fully-Sequential and Semi-Parallel, both for the FIR and FFT architectures, the multiplier circuit is present in the critical path (Figures 2, 3, 4 and 5). For this arithmetic operator, the circuit has a lower delay value [7].

TABLE II. Minimum Clock Period results in ns for the pipelined architectures.

Architecture		Difference(%) Array vs. Booth
Fully-Sequential	FIR	-5.8
	FFT	-6.3
Semi-Parallel	FIR	-5.9
	FFT	-8.7

A.3 Power Dissipation

The array multiplier applied in this work and the Modified Booth present reduced power consumption values because of the reduction of the number of partial product lines. In Table III we present the power per sample values for the Fully-Sequential and Semi-Parallel FIR architectures in the pipelined version, using the array multiplier (m=2) and the Modified Booth multiplier.

TABLE III. FIR architecture - Power per sample (μW).

Architecture	Modified Booth	Array m=2	Difference(%) Array vs. Booth
Fully-Sequential	215.4	155.7	-27.7
Semi-Parallel	188.6	136.9	-27.4

As can be observed in Table III, with the use of the array multiplier power per sample savings above 27% are achievable in the Fully-Sequential and Semi-Parallel FIR architectures. This occurs because multiplier circuits are the main responsable for the power consumption in the FIR architectures and the array multiplier consumes less power due to the simplest structure and smaller critical path and delay values.

FFT architectures also have multiplier circuit in the critical path, as can be observed in Figures 4 and 5. For the FFT structure, the higher number of multiplier circuits in the butterfly produces a great amount of glitching activity. Thus, with the use of the array multiplier, the FFT architectures become significantly more efficient presenting close to 40% less power consumption per transform, as shown in Table IV. This power reduction is mainly due to the lower logic depth of the array multiplier structure, which has a big impact on the reduction of the amount of glitching in the FFT circuits. It has been mentioned that these array multipliers alone may save up to 50% in power when compared to the Modified Booth multiplier [7].

TABLE IV. FFT architecture - Power per transform (mW).

Architecture	Modified Booth	Array m=2	Difference(%) Array vs. Booth
Fully-Sequential	156.6	96.0	-38.7
Semi-Parallel	144.8	81.6	-43.6

B. Application of Coefficient Manipulation

The coefficients ordering algorithm presented in [8] reorders the coefficients in a Fully-Sequential architecture in order to reduce the switching activity in the multiplier inputs. In Table V we show the power per sample results after using this algorithm in the Pipelined Fully-Sequential FIR filter architecture with array (m=2) operator. In this table, it is also shown the power per sample results after applying the ordering algorithm to the Semi-Parallel architecture.

TABLE V. FIR architecture - Power per sample (μW).

	Original Coefficients	Manipulated Coefficients	Difference(%) Manip. vs. Orig.
Fully-Seq	155.7	156.5	+0.5
Semi-Par	136.9	132.5	-3.2

As shown in Table V, there is no significant power per sample reduction in the FIR architectures for the set of coefficients used in this work. Moreover, the Fully-Sequential architecture that uses this technique imposes a bottleneck where 8 clock cycles are required for each sample. On the other hand, in the Semi-Parallel architecture the performance is improved since the hardware can be operated at half of clock cycles. Thus, we have experimented the application of our ordering and partitioning algorithm in this architecture. As can be observed in Table V, for the set of coefficients used, the Semi-Parallel architecture with ordering and partitioning presents more power per sample reduction compared to Sequential architecture with ordering algorithm. This technique has the potential to be significantly more effective in a set of coefficients with higher correlation.

The manipulation techniques that have been applied to the Fully-Sequential and Semi-Parallel architectures show that the correlation between coefficients can reduce the switching activity in the multipliers input. In the FFT algorithm this aspect becomes more significant due to a higher number of coefficients used in all the stages of the FFT. Thus, we have a higher opportunity for saving power by the manipulation of coefficients. Table VI shows the power per transform results by the application of the manipulation technique in the Pipelined Fully-Sequential and Semi-Parallel FFT architectures with the array multiplier.

TABLE VI. FFT architecture - Power per transform (mW).

	Original Coefficients	Manipulated Coefficients	Difference(%) Manip. vs. Orig.
Fully-Seq	96.0	85.4	-11.0
Semi-Par	81.6	66.4	-18.6

The higher level of opportunity of applying the ordering algorithm at each stage of the FFT contributes for a significant power per transform reduction when compared to FIR architectures as can be compared in Tables V and VI. In a Semi-Parallel architecture, the coefficients are partitioned into $\frac{N}{4}$ groups at each FFT stage. The aspect of applying the ordering technique in a smaller group of partitioned coefficients increase the proximity between the coefficients. Thus, the Semi-Parallel architecture presents a higher power per transform reduction compared to the Sequential architecture as can be observed in Table VI.

VI. CONCLUSIONS

In this work different, dedicated architectures for FIR filters and FFT were implemented. Power optimization techniques were experimented including architectural exploration and coefficient manipulation. Low power arithmetic operators were experimented in the FIR and FFT architectures. Performance comparisons for pipelined architectures using the array (m=2) and Modified Booth operators were investigated and the results showed that, despite higher area shown by the architectures with the array operators, these architectures can present less minimum clock period and power consumption. Due to the characteristics of the FIR and FFT algorithms, which are performed by the product of input data with appropriate coefficients, the best ordering of these coefficients to minimize the power consumption of the implemented architectures was also investigated. The results showed that the FFT architectures can present more power reduction due to the higher opportunity of using the coefficients manipulation technique.

ACKNOWLEDGMENTS

This research was supported in part by the CAPES (Brasilia/Brazil) Institute, Universidade Católica de Pelotas (RS, Brazil) and by the portuguese FCT under program POCTI.

References

[1] M. Mehendale, S. Sherlekar, and G. Venkatesh. Techniques for Low Power Realization of FIR Filters. *Design Automation Conference*, 3(3):404–416, September 1995.

[2] M. Mehendale, S. Sherlekar, and G. Venkatesh. Low-Power Realization of FIR Filters on Programmable DSP's. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(4):546–553, December 1998.

[3] A. Erdogan and T. Arslan. High Throughput FIR Filter Design for Low Power SOC Applications. In 13^{th} *Annual IEEE International ASIC/SOC Conference*, pages 21–24, 2000.

[4] M. Baas. A Low-Power, High-Performance, 1024-Point FFT Processor. *IEEE Journal of Solid-State Circuits*, 34(3):380–387, March 1999.

[5] K. Parhi. Algorithms and Architectures for High-Speed and Low-Power Digital Signal Processing. In *Proceedings of* 4^{th} *International Conference on Advances in Communications and Control*, 1993.

[6] E. Mussol and J. Cortadella. Low-Power Array Multipliers with Transition-Retaining Barriers. In *PATMOS*, pages 227–235, 1995.

[7] E. Costa, J. Monteiro, and S. Bampi. A New Architecture for Signed Radix 2^m Pure Array Multipliers. In *IEEE International Conference on Computer Design*, pages 112–117, September 2002.

[8] M. Mehendale, S. D. Sherlekar, and G. Venkatesh. Algorithmic and Architectural Transformations for Low Power Realization of FIR Filters. In *Eleventh International Conference on VLSI Design*, pages 12–17, 1998.

[9] A. Genderen. SLS: An Efficient Switch-Level Timing Simulator Using Min-Max Voltage Waveforms. In *Proceedings of the International Conference on Very Large Scale Integration*, pages 79–88, 1989.

[10] A. Oppenheim. and R. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Signal Processing Series., 1989.

[11] P. Kumhom, J. Johnson, and P. Nagvajara. Design, Implementation, and Implementation of a Universal FFT Processor. In *13th Annual IEEE International ASIC/SOC Conference*, pages 182–186, 2000.

[12] S. He and M. Torkelson. Design and Implementation of a 1024-point Pipeline FFT Processor. In *IEEE Custom Integrated Circuits Conference*, pages 131–134, 1998.

[13] S. Douglas and et al. A Pipelined LMS Adaptive FIR Filter Architecture without Adaption Delay. *IEEE Trans. on Signal Processing*, 46(3), 1998.

[14] S. Yu and E. Swartzlander. A New Pipelined Implementation of the Fast Fourier Transform. In *Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, pages 423–427, 2000.

[15] K. Muhammad, R. Staszewski, and P. Balsara. Speed, Power, Area, and Latency Tradeoffs in Adaptive FIR Filtering for PRML Read Channels. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 9(1):42–51, February 2001.

[16] I. Khater, A. Bellaouar, and M. Elmasry. Circuit Techniques for CMOS Low-Power High-Performance Multipliers. *IEEE Journal of Solid-State Circuits*, 31:1535–1546, 1996.

[17] E. Sentovich and et al. SIS: A System for Sequential Circuit Synthesis. Technical report, May 1992.