GRAY ENCODED ARITHMETIC OPERATORS APPLIED TO FFT AND FIR DEDICATED DATAPATHS

Eduardo A. C. da Costa[1], Jose C. Monteiro[2] and Sergio Bampi[3]
[1] Universidade Católica dePelotas (UCPel), Pelotas/RS-Brazil; [2]Instituto de Engenharia e Sistemas de Computadores (INESC-ID), Lisboa-Portugal; [3]Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre/RS-Brazil

Abstract: This paper addresses the use of low power techniques applied to FIR filter and FFT dedicated datapath architectures. New low power arithmetic operators are used as basic modules. In FIR filter and FFT algorithms, 2's complement is the most common 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. Each group of m bits uses the Gray code, thus potentially further reducing the switching activity both internally and at the inputs. The multiplier 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. As will be shown, the use of the low power operators with an appropriate choice of coefficients can contribute for the reduction of power consumption of the FIR and FFT architectures. Additionally, a new algorithm for the partitioning and ordering of the coefficients is presented. This technique is experimented in a Semi-Parallel architecture which enables speed-up transformation techniques.

Key words: Hybrid encoding, DSP architectures, power consumption.

Please use the following format when citing this chapter:
A. C. da Costa, Eduardo, Monteiro, Jose, C., Bampi, Sergio 2006, in IFIP International Federation for Information Processing, Volume 200, VLSI-SOC: From Systems to Chips, eds. Glesner, M., Reis, R., Indrusiak, L., Mooney, V., Eveking, H., (Boston: Springer), pp. 281-297.

1. INTRODUCTION

This work focuses on power optimization techniques at the architectural level applied to Digital Signal Processing (DSP) systems[1-4]. DSP applications require high computational speed and, at the same time, suffer from stringent power dissipation constraints[5]. Power consumption in VLSI DSP circuits has gained special attention mainly due to the proliferation of high-performance portable battery-powered electronic devices like cellular phones, laptop computers, etc. In DSP applications, Finite Impulse Response (FIR) and Fast Fourier Transform (FFT) are two of the most widely used algorithms.

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 in the dedicated DSP architectures[6]. These multiplier circuits, named Hybrid array multipliers, use coding as a method of decreasing the switching activity. As observed in this paper, DSP architectures that use the multiplier of[6] are more efficient than those that use the common Booth multiplier. Power savings above 35% are achievable in the FFT architecture using Hybrid array multiplier of[6]. 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 work, the low power arithmetic modules are experimented in dedicated FIR filter and FFT architectures. In the FIR implementations, combinations of Fully-Sequential and Semi-Parallel architectures with pipelined version are explored. For the FFT algorithm, Fully-Sequential and Semi-Parallel architectures with pipelined version are also implemented.

Additionally, we propose an extension to the Coefficient Ordering technique[7] 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 work is organized as follows. In Section 2, we discuss the dedicated FIR filter and FFT realization. An overview of coding for low power is presented in Section 3. Section 4 describes the low power techniques use 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.

2. DEDICATED FIR AND FFT REALIZATION

We present Fully-Sequential and Semi-Parallel FIR filter architectures in the Pipelined form. The Pipelined version is also explored for the Fully-Sequential and Semi-Parallel FFT implementation. 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 lower number of coefficients, the technique shown in this work could be applied to architectures with any coefficient order. However, the results of these more complex architectures are limited by the power estimation tool used in this work[8].

2.1 FIR Filter Datapaths

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 Eq. (1).

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

In the Direct Form FIR filter implementation, in each clock cycle a new data sample and the corresponding filter coefficients are simultaneously, producing considerable glitching at the primary outputs[2].

In our work, we address this problem by implementing an alternative Fully-Sequential architecture, called Pipelined form, as shown in Fig. 1.

The Fully-Sequential architecture is a manner to reduce hardware requirements for the FIR filter algorithm, shown in Fig. 1(a). In the sequential implementation the basic idea is to reduce hardware requirements by re-using as much of the hardware as possible.

In order to speed-up the FIR filter computations, we have experimented a Semi-Parallel architecture. In this architecture, shown in Fig. 1(b), hardware requirements are duplicated with respect to the Fully-Sequential, allowing two samples to be processed simultaneously. Again, we have constructed a Pipelined version of the Semi-Parallel architecture.

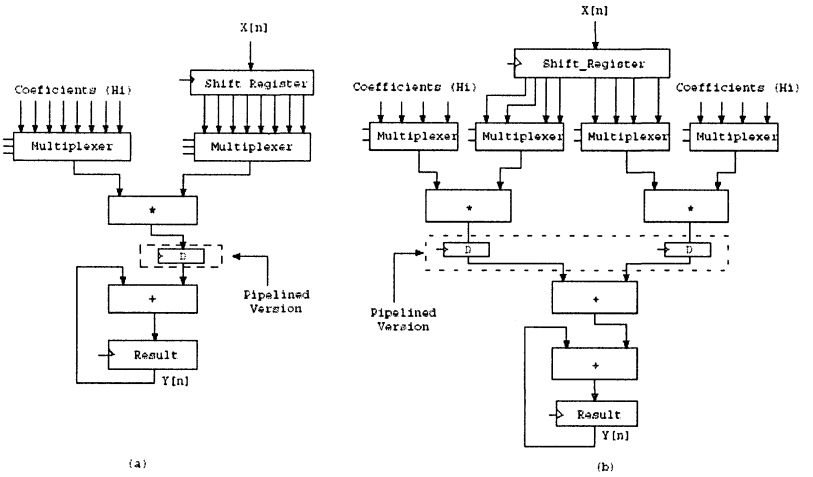

Figure 1. FIR Filter Fully-Sequential and Semi-Parallel Implementations

2.2 FFT Datapaths

The main goal of the FFT algorithms is to compute the Discrete Fourier Transform (DFT) efficiently[9]. 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 Eq. (2) and Eq. (3).

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

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

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 Fig. 2 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 Fig. 2 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 Fig. 2.

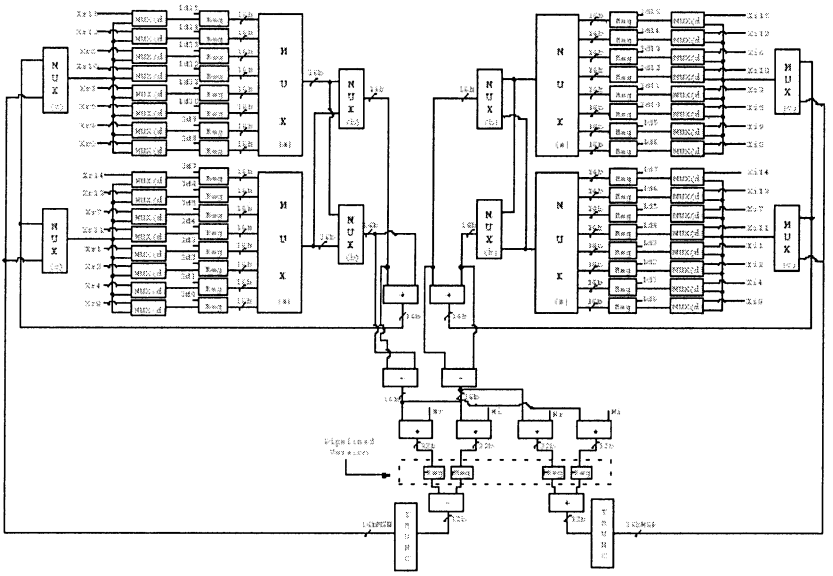

Figure 2. Datapath of FFT Fully-Sequential Implementations

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 Fig. 2.

In a 16-point Fully-Sequential FFT 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 Fig. 3. 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 a Pipelined Semi-Parallel architecture, as shown in Fig. 3.

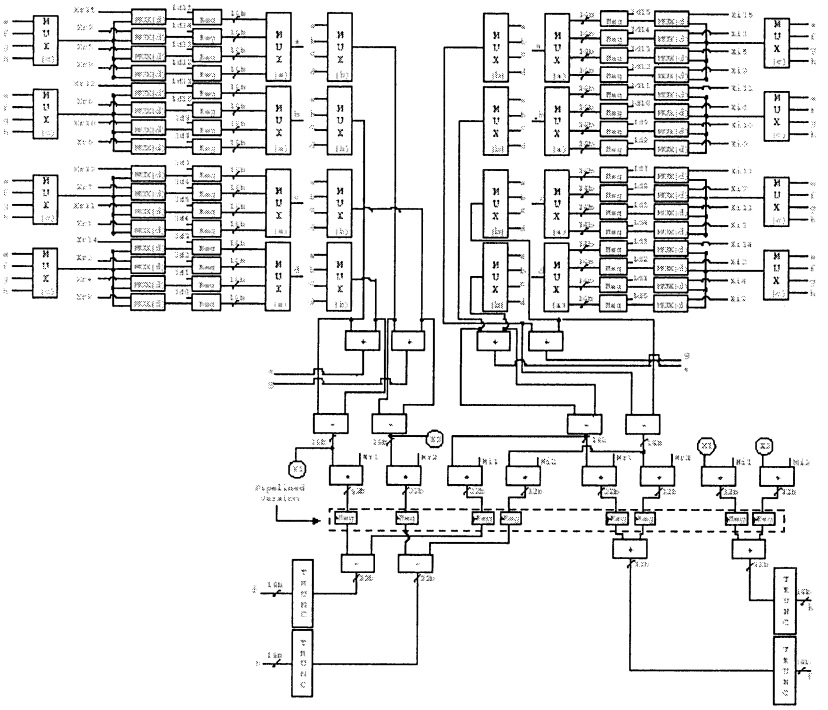

Figure 3. Datapath of FFT Semi-Parallel Implementations

2.3 Related Work on FIR and FFT Realization

Various architectures have been used in FIR filter and FFT realizations, where implementations in programmable DSP and hardwired architectures are addressed[1,3,10]. In case of applications where the flexibility of the programmable processor is not required, hardwired implementation is the preferred choice as such an implementation typically results in higher throughput and low power[7].

For the hardwired implementation, architectural transformations have targeted performance, power and computational complexity[7]. A very efficient technique when targeting low power consumption is to reduce the

supply voltage, resulting in a power reduction proportional to the square of the reduction in the supply voltage. With this objective, parallel processing and pipelining have been applied to the implementation of FIR filters and FFT architectures[4,11-14]as a form of recovering the performance loss due to the lower supply voltages.

The work presented in this chapter 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 pipelined dedicated FIR filter and FFT architectures. In our work, we experiment the use of low power arithmetic operators in the dedicated architectures.

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[15,7]. 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.

3. CODING FOR LOW POWER

Coding has long been used in communication systems to control the statistics of the transmitted data symbols, or in other words, to control the spectrum of the transmitted signal[16].

Low-Power techniques for global communication in CMOS VLSI using data encoding methods are overviewed in[17], where it is shown that such techniques can decrease the power consumed for transmitting information over heavy load communication paths (buses) by reducing the switching activity.

One technique that has been proposed in order to reduce the switching activity on buses is One-Hot Coding[16]. This technique is a redundant coding scheme with a one-to-one mapping between the n-bit data words to be sent and the m-bit data words that are transmitted. The main disadvantage of this technique is related to the wire quantity required, proportional to 2^n.

The Limited-Weight Codes is another technique proposed in order to obtain switching activity reduction on buses[18]. This technique requires

transition signaling in order to reduce the switching activity, since with transition signaling only 1's generate transitions. According to[18], transition signaling is convenient for low-power as it offers a direct way of controlling the bus activity factor simply by reducing the number of logical 1's transmitted over the bus.

The Bus-Invert method as a means of encoding words for reducing I/O power, in which a word may be inverted and then transmitted if doing so reduces the number of transitions[19]. In this method an extra bus line, called *invert* is used.

The Transition Coding and Bit Prediction techniques were used in order to reducing the number of transitions observed in data and address buses[20]. The Transition Coding technique indicates that there is a transition on the bus every time the data to be transmitted is a 1 and there is no transition on the bus if the data to be transmitted is a 0. The Bit Prediction technique is used in address buses that exhibit a very high percentage of addresses that are sequential, so that a factor can be used to predict the value of the next data word with reasonably high accuracy.

One of the most promising encodings that can be used to reduce switching activity is the Gray code since only one bit changes between consecutive values. Therefore, for highly correlated signals the switching activity can be reduced significantly[16]. This code has been applied to code address lines for both instruction access and data access to reduce the number of transitions[16].

As presented above, there is a large number of techniques that resort to signal encoding in order to reduce switching activity on buses. These techniques have all been applied to address buses where data is highly sequential. In[6] similar techniques were applied to arithmetic operators that operate directly upon different coded inputs. In this work we have experimented the use of these operators in the dedicated FIR and FFT architectures.

4. 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.

4.1 Low Power Arithmetic Operators

In this section we summarize the methodology of[6] for the generation of regular structures for arithmetic operators using signed radix-2^m Hybrid representation.

4.1.1 2's Complement Radix-2^m Hybrid Multiplier Architecture

The idea of splitting the operands in groups of m-bits and encode each group using the Gray code can be used for operands that operate in 2's complement representation. Table 1 shows the 2's complement Hybrid encoding for 4-bit numbers and m=2.

Table 1. 2's Complement Hybrid Code Representation for m=2

Decimal	Hybrid	Decimal	Hybrid	Decimal	Hybrid	Decimal	Hybrid
0	0000	4	0100	-8	1100	-4	1000
1	0001	5	0101	-7	1101	-3	1001
2	0011	6	0111	-6	1111	-2	1011
3	0010	7	0110	-5	1110	-1	1010

For the operation of a radix-2^m multiplication, the operands are split into group 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 Eq. (4). This operation is illustrated in Fig. 4.

$$AxB = A'xB' - A'b_{W-1}b_{\frac{W}{m}-1}2^{W-m} - a_{W-1}a_{\frac{W}{m}-1}\sum_{j=0}^{\frac{W}{m}-1} b_j 2^{W-m+j} \quad (4)$$

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 Fig. 5. The modules of the architecture are performed by using Hybrid representation.

Moreover, as can be observed in the dotted lines of the Fig. 5, the sign extension is shown in Hybrid representation (1000 for a negative number).

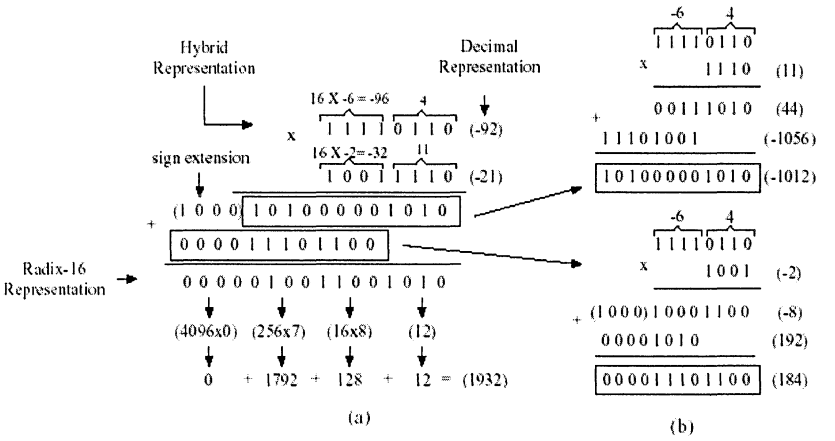

Figure 4. Example of a 2's complement 8-bit wide radix-16 multiplication

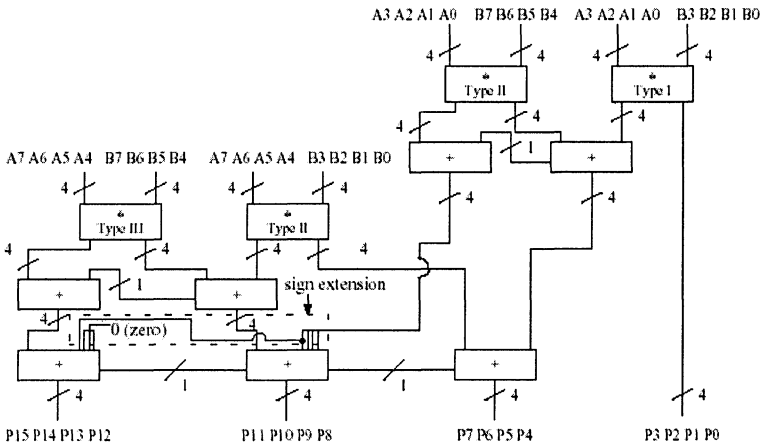

Figure 5. Example of a 8-bit wide 2's complement radix-16 Hybrid multiplier

4.1.2 Modified Booth Multiplier

The radix-4 Booth's algorithm (also called Modified Booth) has been presented in[21]. 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. 6 for a 8 bits wide operation. A multiplexer produces the partial product according to the encoded control signal.

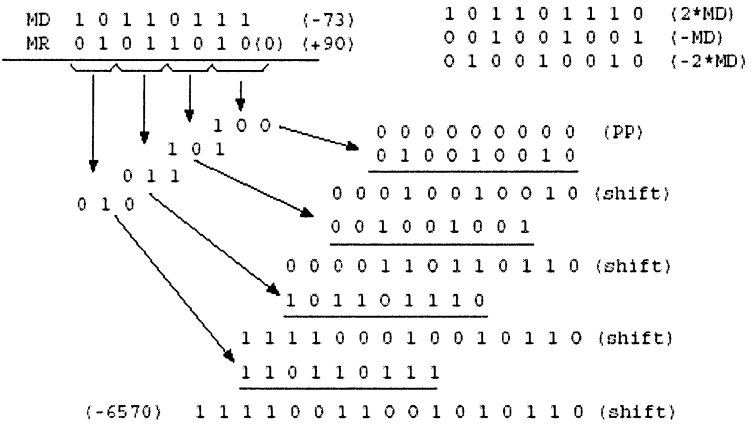

Figure 6. Example of a 8-bit wide Modified Booth multiplication

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

4.2 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, and the filter output is independent of the order of computing the coefficient product[7]. Coefficient ordering is used in[7] 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 data 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 partition for each coefficient by calculating the minimum Hamming distance between the coefficients into each group.

The pseudo-code presented in Fig. 7 describes an example of the algorithm that optimizes the partitioning and ordering of the coefficients. In the example shown in Fig. 7, 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 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.

```
1.  for all permutations of coefficients H(0-7){
2.    partition1=Hamming((H[0],H[1]) + (H[1],H[2]) +
3.              (H[2],H[3]) + (H[3],H[0]));
4.    partition2=Hamming((H[4],H[5]) + (H[5],H[6]) +
5.              (H[6],H[7]) + (H[7],H[0]));
6.    cost function = partition1 + partition2;
7.    if(cost function < minimum found){
8.      save current partition;
9.      minimum=cost function;
10.   }
11. }
```

Figure 7. Pseudo-code of the algorithm for the generation of coefficient partitioning and ordering

5. 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[22] using the general delay model from the *mcnc* library. This parameter defines the minimum clock period. Power results were obtained with the SLS tool[8] 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 have chose to compute the power dissipation per sample for the FIR filter and the power dissipation per transform for the FFT.

5.1 Application of the Low Power Arithmetic Operator

In this section, we present results on use of the Hybrid array (m=2) arithmetic operators of Section 4.1 in the FIR and FFT architectures. Area, minimum clock period and power consumption are investigated and compared to the architectures with Modified Booth operator.

5.1.1 Area

Table 2 presents area results for FIR filter and FFT architectures using the Hybrid 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 Hybrid array multiplier operators present more area. This due to the fact that Hybrid array multipliers require more area than Booth circuits.

Table 2. Area results for the pipelined architectures

Architectural Alternative		Operators		Difference (%)
		Booth	Hybrid Array	Hybrid Array vs. Booth
Fully-Sequential	FIR	6427	8035	+25.0
	FFT	24099	32435	+34.6
Semi-Parallel	FIR	10569	13785	+30.4
	FFT	46000	58964	+28.2

5.1.2 Minimum Clock Period

Although FIR filter and FFT architectures with the Hybrid array operators present higher area, these architectures permit a slightly lower clock period than the architectures with Booth operators, as shown in Table 3. This reduction occurs because in the Fully-Sequential and Semi-Parallel FIR and FFT architectures the multiplier circuit is present in the critical path (Fig. 1, Fig. 2 and Fig. 3). For this arithmetic operator, the circuit has a lower delay value[6].

Table 3. Minimum Clock Period results in ns for the pipelined architectures.

Architectural Alternative		Operators		Difference (%)
		Booth	Hybrid Array	Hybrid Array vs. Booth
Fully-Sequential	FIR	260.1	254.8	-2.0
	FFT	355.0	342.6	-3.5
Semi-Parallel	FIR	258.1	252.8	-2.1
	FFT	418.2	414.1	-1.0

5.1.3 Power Dissipation

The Hybrid array and the Modified Booth multiplier applied in this work present reduced power consumption values because of the reduction of the number of partial product lines. In Table 4 we present the power per sample values for the Fully-Sequential and Semi-Parallel FIR architectures in the pipelined version, using the Hybrid array multiplier (m=2) and the Modified Booth multiplier.

Table 4. FIR architecture – Power per sample (μW).

Architectural Alternatives	Modified Booth	Hybrid Array m=2	Difference(%) Hybrid Array vs. Booth
Fully-Sequential	215.4	180.7	-16.1
Semi-Parallel	188.6	158.2	-16.1

As can be observed in Table 4, with the use of the Hybrid array multiplier power per sample savings above 16% are achievable in the Fully-Sequential and Semi-Parallel FIR architectures. This occurs because multiplier circuits are the main responsible for the power consumption in the FIR architectures and the Hybrid array multiplier consumes less power due to the simplest structure and smaller critical path and delay values.

Besides the FIR filter, the FFT architectures also have multiplier circuit in the critical path, as can be observed in Fig. 2 and Fig. 3. 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 Hybrid array multiplier, the FFT architectures become significantly more efficient presenting close to 30% less power consumption per transform, as shown in Table 5. 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.

Table 5. FFT architecture – Power per transform (mW).

Architectural Alternatives	Modified Booth	Hybrid Array m=2	Difference(%) Hybrid Array vs. Booth
Fully-Sequential	156.6	110.4	-29.5
Semi-Parallel	144.8	92.8	-35.9

5.2 Application of Coefficient Manipulation

In Table 6 we show the power per sample results after using this algorithm in the Pipelined Fully-Sequential FIR filter architecture with Hybrid array (m=2 and m=4) operators. In this table, it is also shown the power per sample results after applying the ordering algorithm to the Semi-Parallel architecture.

Table 6. FIR architecture – Power per sample (μW)

	Group of Bits	Original Coefficients	Manipulated Coefficients	Difference(%) Manip. vs. Orig.
Fully-Sequential	m=2	180.7	176.9	-3.8
	m=4	228.5	216.0	-12.5
Semi-Parallel	m=2	158.3	151.1	-6.7
	m=4	215.0	201.0	-13.9

As shown in Table 6, there is no significant power per sample reduction in the FIR architectures with m=2 Hybrid array operator for the set of coefficients used in this work. However, for groups of m=4 bits there is a higher correlation between these coefficients and the architectures with m=4 Hybrid array operator present an increasingly power per sample reduction as can be observed in Table 6.

As can be observed in Table 6, for the set of coefficients used in this work, the Semi-Parallel architecture using ordering and partitioning algorithm presents more power per sample reduction compared to Sequential architecture with ordering algorithm. This technique becomes more effective in a set of coefficients with higher correlation (m=4).

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 7 shows the power per transform results by the application of the manipulation technique in the Pipelined Fully-Sequential and Semi-Parallel FFT architectures with m=2 and m=4 Hybrid array multiplier.

In a Semi-Parallel architecture, the coefficients are partitioned into $N/4$ 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 m=2 and m=4 Semi-Parallel architecture presents a higher power per transform reduction compared to the Sequential architecture as can be observed in Table 7.

Table 7. FFT architecture – Power per transform (mW)

	Group of Bits	Original Coefficients	Manipulated Coefficients	Difference(%) Manip. vs. Orig.
Fully-Sequential	m=2	110.8	91.7	-17.2
	m=4	101.1	91.4	-9.6
Semi-Parallel	m=2	93.7	75.5	-19.4
	m=4	87.7	70.1	-20.1

6. CONCLUSIONS

In this work, 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 Hybrid 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.

7. REFERENCES

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

2. A. Erdogan, and T. Arslan, High Throughput FIR Filter Design for Low Power SOC Applications. In *13th Annual IEEE International ASIC/SOC Conference*, pp. 21-24, (2000).

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

4. K. Parhi, Algorithms and Architectures for High-Speed and Low-Power Digital Signal Processing. In *Proceedings of 4th International Conference on Advances in Communications and Control*, (1993).

5. E. Mussol, and J. Cortadella, Low-Power Array Multipliers with Transition-Retaining Barriers. *PATMOS*, pp. 227-235. 2002.

6. E. da Costa, J. Monteiro, and S. Bampi, A New Architecture for 2's Complement Gray Encoded Array Multiplier. In *Proceedings of the XV Symposium on Integrated Circuits and Systems Design*, pp. 14-19, (2002).

7. M. Mehendale, S. Sherlekar, and G. Venkatesh, Algorithmic and Architectural Transformations for Low Power Realization of FIR Filters. In *Eleventh International Conference on VLSI Design*, pp. 12-17, (1998).
8. 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*, pp. 79-88, (1989).
9. A. Oppenheim, and R. Schafer, *Discrete-Time Signal Processing*. London: Prentice Hall Signal International, (1989).
10. P. Kumhom, J. Johnson, and P. Nagvajara, Design and Implementation of a Universal FFT Processor. *In 13th Annual IEEE International ASIC/SOC Conference*, pp. 182-186, (2000).
11. S. He, and M. Torkelson, Design and Implementation of a 1024-point Pipeline FFT Processor. In *IEEE CICC*, pp. 131.134, (1998).
12. S. Douglas, and et al., 1998, A Pipelined LMS Adaptive FIR Filter Architecture without Adaption Delay. *IEEE Transactions on Signal Processing*, 46(3), (1998).
13. S. Yu, and E. Swartzlander, A New Pipelined Implementation of the Fast Fourier Transform. In *Thirty-Fourth Asilomar Conference on Signals, Systems and Computers*, pp. 423-427, (2000).
14. K. Muhammad, R. Staszewski, and P. Balsara, Speed, Power, Area, and Latency Tradeoffs in Adaptive FIR Filtering for PRML Read Channels. *IEEE Transactions on VLSI Systems*, 9(1):42-51, (2001).
15. M. Mehendale, S. Sherlekar, and G. Venkatesh, Techniques for Low Power Realization of FIR Filters. In *Design Automation Conference*, pp. 404-416, (1995).
16. A. Chandrakasan, and R. Brodersen, *Low Power Digital CMOS Design*. Kluwer Academic Publishers, (1995).
17. M. Stan, and W. Burleson, Low-Power Encodings for Global Communication in CMOS VLSI. *IEEE Trans. on VLSI Systems*, (1997).
18. M. Stan, and W. Burleson, Limited-Weight Codes for Low-Power I/O. *IEEE International Workshop on Low Power Design*, (1997).
19. M. Stan, and W. Burleson, Bus-Invert Coding for Low-Power I/O. *IEEE Transactions on VLSI Systems*, (1995).
20. P. Ramos, and A. Oliveira, Low Overhead Encodings for Reduced Activity in Data and Address Buses. In *IEEE International Symposium on Signals, Circuits and Systems*, pp. 21-24, (1999).
21. 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).
22. E. Sentovich, and et al., SIS: A System for Sequential Circuit Synthesis, Technical report, (1992).