

THE DEVELOPMENT OF FPGA ARCHITECTURES FOR PRIMITIVE OPERATOR DIGITAL FILTER IMPLEMENTATIONS

T Arslan, H I Eskikurt and D H Horrocks¹

I. INTRODUCTION

Primitive Operator Filter (POF) design technique is a methodology for the realisation of low complexity digital filters through a significant reduction in the number of arithmetic operations [1,2]. Typically a POF is realised using signal flow graph indicating operations such as shifts and additions.

Multipliers are a requirement of most conventional digital filter architectures. High speed digital filtering tasks require fast multipliers which are costly in terms of chip area. For this reason POF design techniques can be utilised to replace multipliers with more primitive operators such as adders, subtractors, and shifters [1].

In recent years, Field Programmable Gate Arrays (FPGAs) have proved to be a popular method of implementation for many applications because of their programmability, reduced development costs, which makes them ideal for rapid product development and prototyping. An FPGA typically consists of Configurable Logic Blocks (CLBs), which are used to implement logic functions, an interconnection matrix and a control unit. Most general purpose FPGAs such as those from XILINX, ACTEL and ALGOTRONIX can be used to implement a wide range of logic functions and are not optimised for specific tasks such as DSP and digital filtering. Due to the wide range of DSP applications and their computationally intensive nature, there is a clear need for FPGAs which are DSP-specific. This need is clearly exemplified with the recent research work in this area [3-5].

This paper reports on current research work being carried out in Cardiff School of Engineering on the development of FPGAs specific for Primitive Operator Filter implementation. The paper proposes two modified CLB architectures, termed CALB (Configurable Arithmetic Logic Block), that are specifically designed for POF implementation and compares these in terms of speed and granularity with a number of practical FIR filter examples.

II. PRIMITIVE OPERATOR FILTERS

Area and speed are important factors in hardware implementation of most digital filters. For this reason a number of design techniques have emerged which aim to optimise area while maintaining throughput. Most of these techniques aim to reduce the complexity of the multiplication operations within the filter design [6-14]. For example savings can be made using multiplier re-coding techniques [6-9].

A characteristic of the direct form structure for an FIR filter can be given by,

$$y[n] = \sum_{i=0}^n h[i].x[n-i] \quad , i = 0 \dots n \quad (1)$$

¹The authors are with the Cardiff University of Wales, School of Engineering.

where $h[i]$ are the coefficients of the filter, $x[n-i]$ are the filter inputs and $y[n]$ is the output of the filter. This is represented diagrammatically in Figure 1, where the broken line encloses the multiplication block of the filter. The output signal of the multiplication performed at stage i of the filter is given by,

$$w_i[n] = x[n] \cdot h[i] \quad , i = 0 \dots n \quad (2)$$

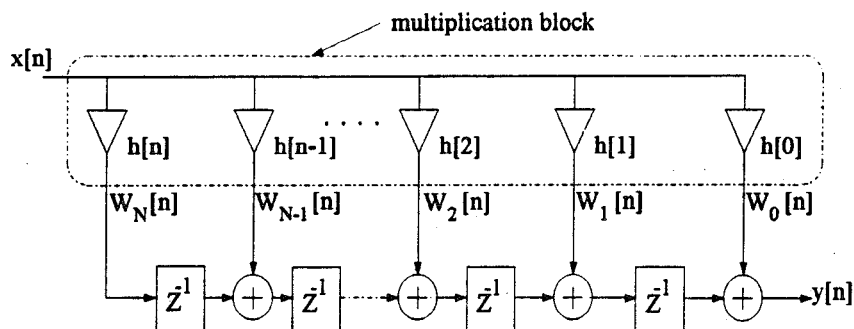


Figure 1- Transposed FIR filter.

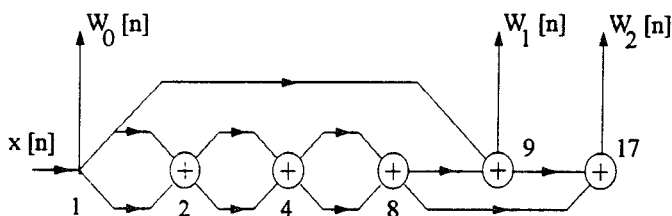


Figure 2- Signal Flow Graph to generate outputs corresponding to the example coefficient set {1,9,17}.

The work in [2,15-17] has demonstrated that the multiplication block of a digital filter may be realised through the application of graph synthesis techniques employing only primitive operations such as addition, subtraction, and bit-wise shift.

An example [1] of a signal flow graph representation of the multiplication block for a filter with coefficients $\{h[0]=1, h[1]=9 \text{ and } h[2]=17\}$ is shown in Figure 2. The formation of $w_0[n]$, $w_1[n]$ and $w_2[n]$ values requires 5 additions. If the same outputs are generated using conventional shift and add multiplication methods [18], the total number of operations will be at least 9.

III. PRIMITIVE OPERATOR FPGA

An FPGA architecture such as the one shown in Figure 3 was found to be suitable for primitive operator filter implementations. A number of CALBs are controlled by a programmable control unit. Each CALB has a corresponding *configuration memory*. Communication is achieved using two main buses: the *control bus* carries control information to the CALBs and memory and the *data bus* is responsible for data transfer between the I/O Buffers, CALBs and *W-memory*. Each CALB is connected to the data bus via tri-state buffers. Since all CALBs are connected in parallel to the data bus, this structure has the advantage of simultaneous data processing by a number of CALBs. Thus if data is available at the inputs of a number of CALBs, then these can operate simultaneously without the need for waiting for other CALBs.

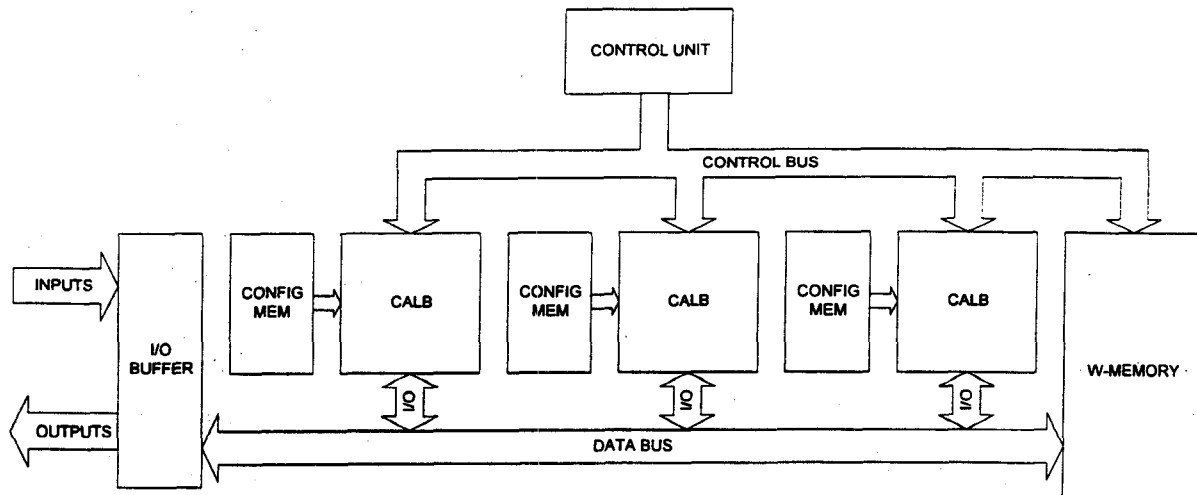


Figure 3- The proposed architecture.

The configuration memories are loaded by configuration data which consists of a set of control signals for configuring the CALBs. After loading the configuration memories, external inputs are supplied to the CALBs in order to be multiplied using primitive operators. The $w_i[n]$ values, are then stored in the *W-memory*. The memory contents are transferred to the main outputs when the read signal is activated by the control unit. Multiplication outputs are now available for other operations.

The form of the CALB structure has a major impact on the performance of an FPGA. Two CALB structures, CALB1 and CALB2, of differing complexities have been investigated. These are illustrated in Figures 4 and 5 respectively. The number of CALB blocks required for an application depends on the order of the filter.

Both CALB architectures consist of latches, multiplexers, shifters, and adders. The latches are used to store the input data loaded from the data bus. A given configuration memory stores the control signals associated with the multiplexers, adders, and shifters in the corresponding CALB. For the multiplexers, the signals (C1-C12) dictate the direction of paths through the corresponding multiplexer. For the adders, the signals (X1-X4) indicate whether an addition or subtraction (2's complement) is to be performed. Whereas for the shifters a 3-bit signal will allow up to 7-bit shifts. The same applies to the CALB2 architecture.

As an example, the coefficients {1, 9, 17} are implemented using the CALB1 architecture. As could be seen in Figure 4, the first three additions in Figure 2 are replaced by a single shift operation. Hence two additions and a single shift are required. The bold numbers in the figure illustrate the input and output values after each operation.

The same example has been used to illustrate the use of CALB2 in Figure 5. In this case, however, two CALB2 blocks are required. This is due to the simplicity of this architecture which consists of two shifters and an adder only. Again the bold numbers illustrate the input and output values of each operation for the first CALB2. The numbers in italics correspond to the second CALB2.

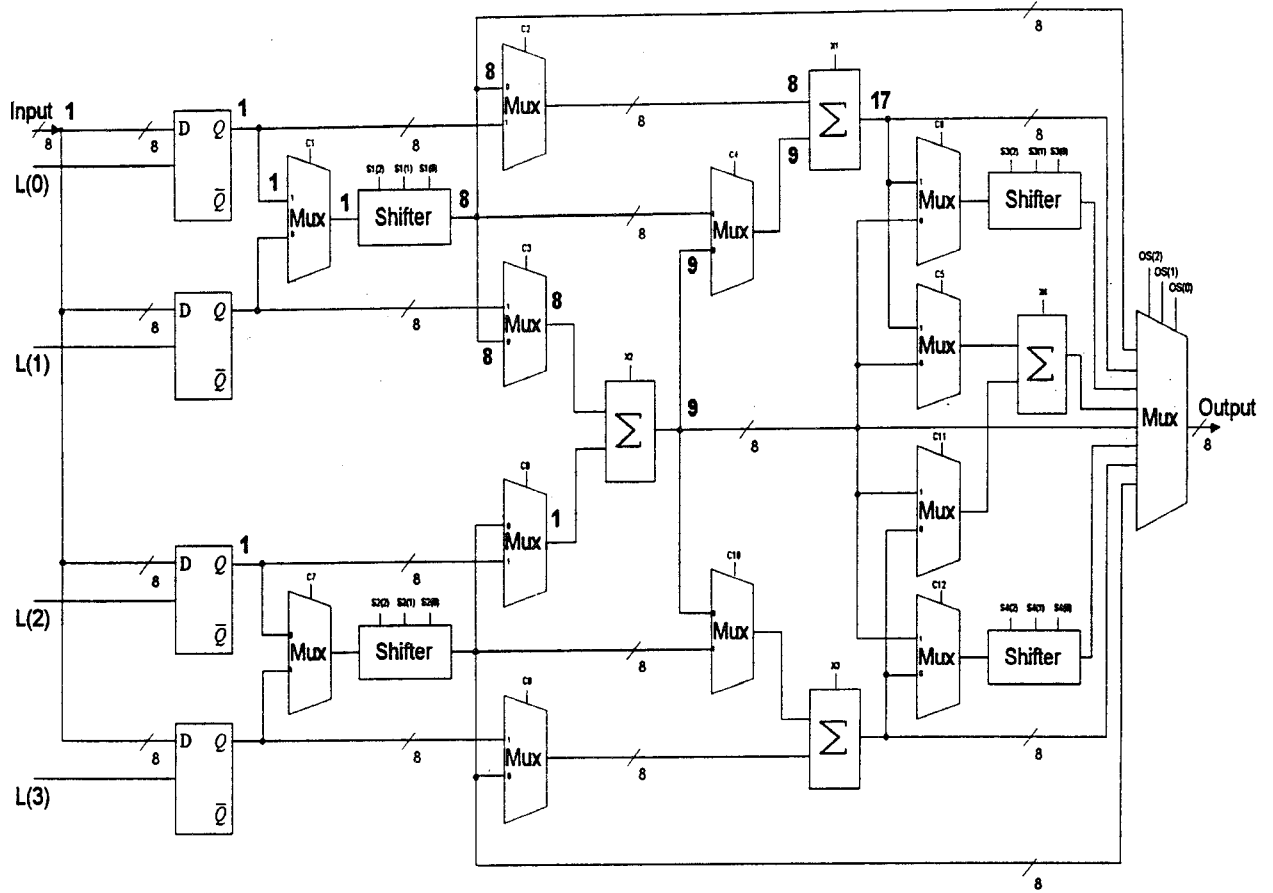


Figure 4- CALB1 architecture.

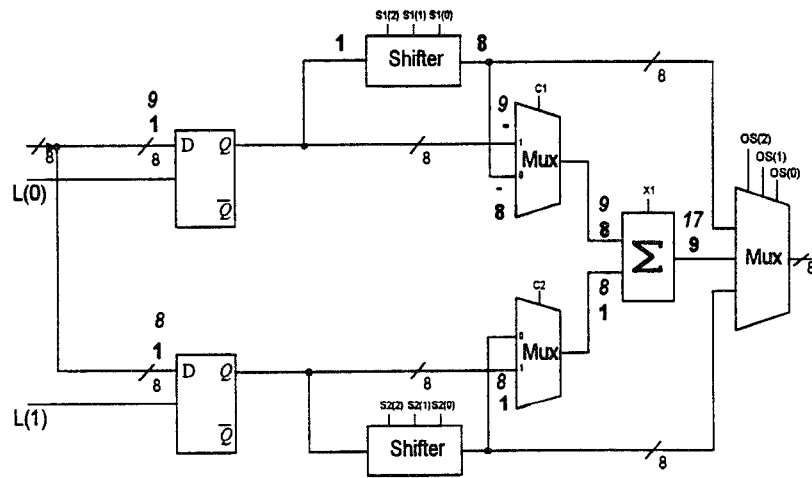


Figure 5- CALB2 architecture.

IV. PERFORMANCE EVALUATION

The two architectures are compared in terms of their operation speed and chip area using a number of examples with different number of coefficients. Table 1 indicates the number of CALBs required for each architecture with each example. The coefficients synthesised by the FPGA are listed in the third column. These exclude identical coefficients caused by symmetry and duplication. The mapping algorithm used is mainly based upon those developed in [1] with some customisation to suit individual CALB architectures where smaller number of primitive operations could be achieved.

Example No.	Filter Degree	No. of Different Coefficients	No. of CALB1 Blocks	No. of CALB2 Blocks
1	5 tap FIR	5	2	4
2	21 tap FIR	11	4	9
3	32 tap FIR	15	3	10
4	41 tap FIR	19	5	12
5	81 tap FIR	28	8	23

Table 1- The number of required CALBs for different examples.

Table 2 compares the two architectures in terms of their operation speed and chip area. In general the CALB1 architecture produces the smallest operation time in comparison with CALB2. The CALB2 architecture, however, is the most area efficient.

Example No.	Operation Time (ns)		Area (sq.mm)	
	CALB1	CALB2	CALB1	CALB2
1	330	330	12.46	7.58
2	575.83	662.14	28.43	19.03
3	558.43	779.5	19.02	22.35
4	756.23	857.70	36.98	25.02
5	1104	1259.4	68.25	50.62

Table 2- CALB performance characteristics.

These results are based on CAD simulations using 1 micron ES2 semi-custom libraries.

V. CONCLUSION

An architecture for a primitive operator FPGA is proposed in this paper. The performance of two CALB architectures, CALB1 and CALB2, has been investigated in terms of their operation speed and chip area. The CALB1 architecture offers the advantage of faster speed of operation whereas the CALB2 architecture offers the advantage of reduced chip area.

REFERENCES:

- [1] Bull, D.R. and Horrocks, D.H., "Primitive Operator Digital Filters", *IEE Proceedings-G*, Vol.138, No.3, pp.401-412, June 1991.
- [2] Bull, D.R., "Signal Processing Techniques with Reduced Computational Complexity", *PhD Thesis*, Cardiff University of Wales, 1988.
- [3] Agarwala, M. and Balsara, P.T., "An Architecture for a DSP Field Programmable Gate Array", *IEEE Trans. on VLSI Systems*, Vol.3, No.1, pp.136-141, March 1995.

- [3] Agarwala, M. and Balsara, P.T., "An Architecture for a DSP Field Programmable Gate Array", *IEEE Trans. on VLSI Systems*, Vol.3, No.1, pp.136-141, March 1995.
- [4] Chen, D.C. and Rabaey, J.M., "A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths", *IEEE J. of Solid-State Cir.*, Vol.27, No.12, pp.1895-1904, Dec. 1992.
- [5] Carruthers, C. and Kean, T., "Bipolar CAL Chip Doubles Speed of FPGAs", Edited by Will Moore and Wayne Luk, *FPGAs*, Abingdon EE&CS Books, chapter 2.5, pp.47-53, Abingdon, England, 1991.
- [6] Peled, A., "On the Hardware Implementation of Digital Signal Processors", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, No.1, pp.76-86, 1976.
- [7] Booth, A.D., "A Signed Binary Multiplication Technique", *J. Mech. Appl. Maths.*, Vol.4, pp. 236-245, 1951.
- [8] Capello, P.R. and Steiglitz, K., "Some Complexity Issues in Digital Signal Processing", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, pp.1037-1041, 1984.
- [9] Wierich, A., "Multiplierless Digital Filters with Partial Product Re-use", *Proc. Int. Conf. on Digital Signal Processing*, pp.22-26, Florence, Italy, 1987.
- [10] Peled, A. and Lin, B., "A New Hardware Realisation of Digital Filters", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, Vol. ASSP-22, pp.456-462, 1974.
- [11] Kingsbury, N.G. and Rayner, P.J.W., "Digital Filtering Using Logarithmic Arithmetic", *Electron. Lett.*, Vol.7, No.2, pp.56-58, 1971.
- [12] Tran-Thong and Liu, B., "A Recursive Digital Filter Using DPCM", *IEEE Trans.*, COM-24, pp.2-11, 1976.
- [13] Agarwal, R.C. and Sudhakar, R., "Multiplier-less Design of FIR Filters", *IEEE Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pp.209-212, Boston USA, 1983.
- [14] Saramaki, T., Neuvo, Y. and Mitra, S.K., "Design of Computationally Efficient Interpolated FIR Filters", *IEEE Trans.*, CAS-35, pp.70-87, 1988.
- [15] Bull, D.R. and Horrocks, D.H., "Reduced Complexity Digital Filtering Structures Using Primitive Operations", *Electron. Lett.*, Vol.23, No.15, pp.7699-7771, 1987.
- [16] Bull, D.R. and Horrocks, D.H., *UK Patent Application: GB 2201854A*, 1987.
- [17] Bull, D.R. and Horrocks, D.H., "Primitive Operator Digital Filter Synthesis Using a Shift Biased Algorithm", *IEEE Proc. Intl. Symp. on Circuits and Systems*, pp.1529-1532, Helsinki, Finland, June 1988.
- [18] Bellanger, M., "Digital Processing of Signals", *John Wiley & Sons*, NY, 1985.