Hardware-efficient pipelined programmable FIR filter design

T.-S.Chang and C.-W.Jen

Abstract: With the increasing demand for video-signal processing and transmission, high-speed programmable FIR filters are required for real-time processing. This paper presents a hardware-efficient pipelined FIR architecture with programmable coefficients. FIR operations are first reformulated into multi-bit DA form at an algorithm level. Then, at the architecture level, the (p, q) compressor, instead of Booth encoding or RAM implementation, is used for high-speed operation. Due to the simple architecture, we can easily pipeline the proposed FIR filter to the adder level and save up to half of the cost of previous designs without sacrificing performance. The presented design is useful for bit-parallel input design, which can save 36.7% of the area cost compared with previous approaches.

1 Introduction

Finite impulse response (FIR) filters are important building blocks for various digital signal processing (DSP) applications. Recently, because of the increasing demand for video-signal processing and transmission, high-speed and high-order programmable FIR filters have frequently been used to perform adaptive pulse shaping and signal equalisation on the received data in real time, such as ghost cancellation [1, 2] and channel equalisation [3]. Hence, an efficient VLSI architecture for a high-speed programmable FIR filter is needed.

However, high-speed programmable FIR filters are hard to implement efficiently because of the programmability requirements that result in high implementation costs. The programmability allows the FIR coefficients to be changed at the run time. Therefore, we cannot directly use specific low-cost implementations that depend heavily on the coding and sharing of the constant FIR coefficients, such as canonical signed digit (CSD) representation [4] and subexpression sharing [5]. That is because there is no efficient way to precompute and store the common subexpression of the programmable coefficients or to perform real-time CSD encoding.

Distributed arithmetic (DA), since its introduction by Peled and Liu [6], has been regarded as an efficient method of implementing constant coefficient filters with ROM. When DA is used for programmable filters, RAM has been adopted to replace ROM to store the partial products [6–9]. However, RAM-based programmable filters suffer from a lower convergence rate owing to the extra cycles needed to update the contents of the RAM table. Besides, exponential growth of the RAM size inhibits their use in higher-order filter designs. Hence, RAM-based DA is not well suited for high-speed adaptive and programmable filter applications.

To implement programmable filters, a commonly used approach is to use one multiplier and one accumulator for each tap [1, 10]. However, the cost of multipliers is too high to be applicable for high-order filters. A bit-level approach such as the bit-plane technique [11] pipelines the addition into the bit level. Although it performs at high speed, large hardware costs and long latency make it unsuitable for adaptive or high-order filtering applications.

In comparison, an attractive method is to encode the input signals to reduce the multiplication complexity. The technique of implementing the multiplier of each tap by a modified Booth multiplier has been presented in [1, 12]. However, because of the accumulation loop in each tap, the hardware cost of each tap is still too high. To conquer this problem, another approach [13, 14] combined the Booth encoding and DA formulation (called Booth DA in the rest of the paper) to reorder the computation, such that each tap is accumulation free and we can then reduce the hardware cost of each tap. However, the Booth-encoding approach is often limited to radix-3, since higher-order encoding is often too complex to be implemented [15]. This limits the possibility of attaining higher throughput. Besides, although the Booth encoding circuit can be shared for all taps, Booth selection circuits for different multiples of the multiplicand often occupy a large area.

To reduce the implementation cost of programmable FIR filters, we first reformulate the filter operations into multi-bit DA representations at the algorithm level. With this formulation, all addends will have the same weights that can avoid long-word-length adders in each tap. Multi-bit representation also gives a unified formulation for higher radix architectures. Then in the architecture level, instead of coding input signals, we use (4,2) compressors or multiple-operand adders to reduce the partial product levels in the architecture designs. (4,2) compressors have been shown to be as effective as Booth encoding at reducing the partial product level and have a smaller hardware cost than several other methods [16–18]. A benefit of the (4,2) compressor is that, instead of transmitting the coded input along the filter chain, we can transmit fewer original input samples, which contributes a large delay saving.

IEE Proceedings online no. 20010726

DOI: 10.1049/ip-cdt:20010726

Paper first received 12th October 1999 and in revised form 20th August 2001

The authors are with the Department of Electronics Engineering, National Chiao-Tung University, 1001 Ta-Hsueh Rd., Hsinchu, Taiwan, Republic of China

2 Algorithm reformulation

Algorithm reformulation can provide a deeper insight into finding an efficient way to implement multiplication in filters, since the addition and multiplication in filter operations are associative. In this Section, we will present the multi-bits DA reformulation of the filter operations. DA is a computation reordering method that operates at the bit level. Considering an N-tap FIR filter with input sequence x_n, output sequence y_n, and coefficient c_i, without loss of generality we can express the FIR formulation with an unsigned fraction as

$$y_n = \sum_{i=0}^{N-1} c_i x_{n-i} = \sum_{i=0}^{N-1} c_i \left(\sum_{j=0}^{W_x-1} x_{n-i,j} 2^{-j} \right) \quad (1)$$

where W_x is the word length of x_n and $x_{n-i,j}$ denotes the jth bit of x_{n-i}. The DA reformulation that decomposes x into the bit level and reorders the summations is

$$y_n = \sum_{i=0}^{N-1} c_i x_{n-i} = \sum_{j=0}^{W_x-1} \left(\sum_{i=0}^{N-1} c_i x_{n-i,j} \right) 2^{-j} \quad (2)$$

To express the formulations as multi-bit DA expressions, we can express x_n as a $\lceil W_x/P \rceil$-digit number with a P-bit digit d_k,

$$x = \sum_{j=0}^{\lceil W_x/P \rceil - 1} d_j 2^{-Pj} \quad (3)$$

where $\lceil\ \rceil$ denotes the smallest integer greater than or equal to the argument. Substituting eqn. 3 into eqn. 1 and reordering the computation, we can obtain a similar formulation as eqn. 2:

$$y_n = \sum_{i=0}^{N-1} c_i x_{n-i} = \sum_{i=0}^{N-1} c_i \left(\sum_{j=0}^{\lceil W_x/P \rceil - 1} d_{n-i,j} 2^{-Pj} \right) = \sum_{j=0}^{\lceil W_x/P \rceil - 1} \left(\sum_{i=0}^{N-1} c_i d_{n-i,j} \right) 2^{-Pj} \quad (4)$$

Comparing with the direct implementation, eqn. 1, DA formulation first adds the partial products of the same weighting and then accumulates the results. This gives a lower word-length requirement in each tap adder. No accumulation is required in the inner summation for the shift-and-add operation, which gives a faster speed and lower hardware cost. These two advantages were first investigated in the Booth DA design [13]. However, with the Booth encoding, extra coded input signal delays and complex Booth encoders will offset the hardware savings of these two advantages. In the next Section, we will present a different architecture design to maximise the hardware advantage.

3 Architecture design

Fig. 1 shows the proposed FIR architecture. For Booth DA architecture, the dotted blocks are comprised of the Booth selector and carry save adders. Table 1 shows the major difference between the two architectures. Both designs show a fully pipelined filter with input signals, X_{in}, of $W_x = 8$ bits and one output, Y_{out}, every four cycles ($W_x/2 = 4$). In the proposed design, we first use the preprocessing unit to convert bit-parallel input signals to digit-serial input (2-bits-at-a-time in this case), then, AND

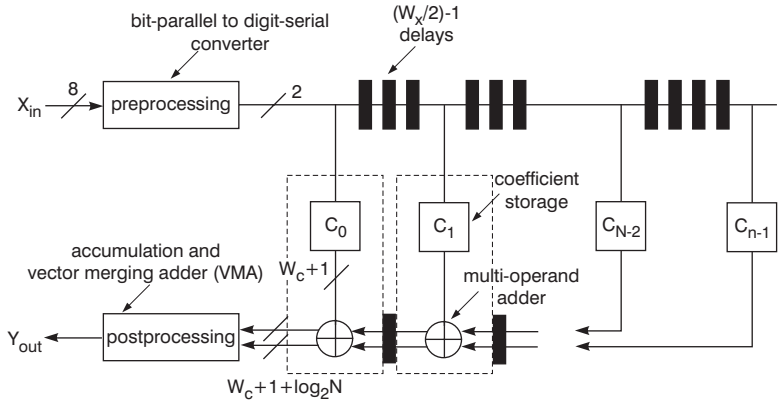

Fig. 1 *The FIR architecture with multi-bit DA for the 2-bits at a time case, where the dotted block is a tap adder*

Table 1: Hardware cost of the Booth DA and the proposed method for FIR designs (coefficient storage and the final VMA are not shown here)

Method	Preprocessing stages	Tap adders			Delays
	Booth encoder	AND gate	Booth selector	full adders	input sample delays + pipeline delays
Booth DA	10	0	$N \times (W_x + 1) \times 11$	$(N-1) \times (W_c + \log_2 N/2 + 1) \times 7$	$(N-1) \times (1/2 \times W_x - 1) \times 3 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 1) \times 2 \times 8$
Proposed (2-bits at a time)	0	$N \times W_x \times 2 \times 2$	0	$(2N-1) \times (W_c + \log_2 N/2 + 1) \times 7$	$(N-1) \times (1/2 \times W_x - 1) \times 2 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 1) \times 2 \times 8$

Note: The gate count of a full adder, delay, AND gate are 7, 8 and 2, respectively [19]

gates and multi-operand adders perform the inner summation of eqn. 4. Finally, all the partial products are propagated through the pipeline registers and accumulated at the final accumulator. As shown in Fig. 1, DA formulation reduces the word length of the tap adders since each tap adder only requires a word length of $(W_c + 1 + \log_2 N)$ bits instead of $(W_c + W_x + \log_2 N)$ bits. Besides, since each tap adder is accumulation free, we can pipeline it.

In the Booth DA design, the partial product is generated through a modified Booth function. The modified Booth function is split into Booth encoder and Booth selection circuits. The Booth encoding circuit is shared with all taps and placed in the preprocessing circuit, which generates 'zero', 'negative' and 'double' signals to select different multiples of the coefficients. The Booth selection circuit is placed at each tap to select the proper multiples of the coefficients, included in the dotted block. Since the Booth encoding circuit is shared with all taps, the Booth encoded signals (three signals: zero, negative and double) instead of input signals (two bits at a time) have to be propagated through the upper delay chains.

The advantage of Booth encoding is that it generates only half of the partial product compared with other designs without Booth encoding. However, the benefit obtained is at the expense of increased hardware complexity. Indeed, Booth encoding implementation requires the hardware for the encoding and for the selection of the partial products $(0, Y, -Y, 2Y, -2Y)$.

On the other hand, reducing the number of partial products by half can be attained by one level of AND gates and one row of (4,2) compressors. The major advantage of the use of this cell is that it allows a highly regular layout. The 2-to-1 reductions of the (4,2) compressor cells can also efficiently reduce the height of the partial product tree as the Booth function, since the delay from input operand to sum output is two XOR delays and the delay from carry-in to all output is one XOR delay. The interconnection of the two full adders should be carefully chosen such that the delay from any input to the sum output in the (4,2) compressor is no greater than the 3 XOR gate delays. Therefore, instead of the Booth function that takes up more area, we use a compressor to implement the addition. Comparing the multi-bit DA with the Booth DA, we can find that the multi-bit DA has a simple preprocessing stage, fewer input sample delays and a simple tap adder design.

4 Hardware comparisons

4.1 Comparison with the Booth DA design

The Booth DA design is a digit-serial approach. An optimised Booth encoding circuit costs about 10 gates count [1] and the Booth selection circuit costs 11 gates. Therefore in the Booth DA design, the hardware items include:

1 Booth encoder + N Booth selectors
+ $(N - 1)$ tap adders
+ $2(N - 1)$ delays for pipelining the tap adder path
+ input sample delays + final VMA
+ coefficient storage

The detailed hardware items and the associated gate area are shown in Table 1. All these area costs in this paper are based on a 0.6 µm CMOS SPDM cell library [19]. For fair comparison, the pipeline method of the Booth DA and the proposed design use the cut-set retiming method to achieve fully pipelined designs. The sign extension of the tap adders for both designs also use the sign-extension method proposed in the Booth DA design.

For the proposed design, we assume two-bits-at-a-time for equal throughput consideration. The hardware required for this case is:

$2N$ AND for partial product
+ $(N - 2)$ tap (4, 2) compressors
+ $2(N - 1)$ delays for pipelining the tap adder path
+ input sample delays + final VMA
+ coefficient storage

The detailed hardware items and the associated area are shown in Table 1.

For a more detailed observation and specific comparisons, Fig. 2 shows the total area ratio plot (proposed design/Booth DA) of Table 1 for cases of 12-bit and 24-bit word length. As shown in the Figure, the multi-bit DA designs use less area compared with the Booth DA designs, saving up to 17% of the area for a 4-tap case. When the tap number grows, the hardware saving becomes less. On the other hand, the hardware saving becomes larger as the word length grows.

The main area saving of the multi-bit DA comes from the input sample delays and tap adders. As shown in Table 1, since we only propagate the input sample with two-bits-at-a-time instead of three Booth encoded signals, we can save $(N - 1)(W_x/2 - 1)$ delays. This part contributes a 10% area reduction to the proposed design. The Booth selector also consumes more gate area than a full adder, so, for each tap adder in the dotted block, two adders costing a (4,2) compressor plus are input AND gate require less area than the hardware cost of one Booth selector and an adder in the Booth DA. This contributes to the area reduction of $(10 + N \times (W_x + 1) \times 11 - N \times W_x \times 4 - (N - 2) \times (W_c + \log_2 N/2 + 1) \times 7)$. The area reduction in tap adder designs is large at small N values, but becomes smaller at large N values. For $N \geq 32$, the area reduction of this part will be negative, which means the area of this part will cost more than the previous design and offset the area saving by the input sample delays. The negative area saving is due to the $(\log_2 N/2)$ factor in the tap adder part, since we keep the full word length of the partial result. Therefore the curve in Fig. 2 has a sharp rise when the tap number is from 4 to 32, but rises slowly when the tap number is greater than 32. Although this reduction is small for the 2-bits-at-a-time case, it will grow when we extend to higher radix designs.

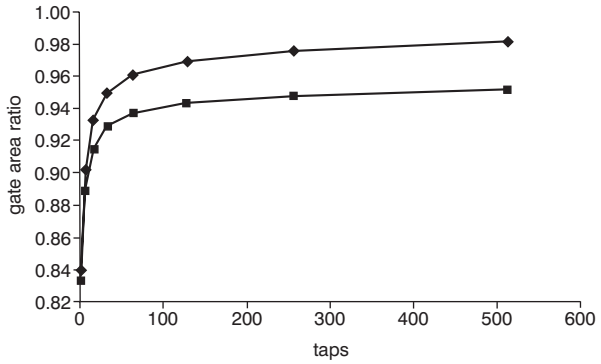

Fig. 2 *Gate area ratio plot (the proposed method with 2-bits-at-a-time/the Booth DA), starting from taps = 4 ($W_x = W_c = 12$ bits or 24 bits is assumed in the Figure)*
–◆– 12 bits –■– 24 bits

The delays of the two designs are comparable. For the Booth DA with the pipeline architecture, the delay is '4-input MUX + AND + FA + DFF setup time'. For 2-bits-at-a-time with the pipeline architecture, the delay is 'AND + 2 × FA + DFF setup time'. Therefore both designs can operate at the same clock rate.

4.2 Extension to higher radix designs

When extending to higher radix designs for higher throughput, the area saving by the multi-bit DA is larger than that at the 2-bits-at-a-time case. The overall architectures of the higher radix designs are similar to those shown in Fig. 1. However, for Booth DA designs, the high radix designs need a high radix Booth encoding and selection circuit and more coefficient storage for different multiples of the coefficients. Table 2 shows the hardware cost of the 4-bit Booth DA designs and the proposed designs and Table 3 shows the hardware cost of the 5-bit Booth DA designs and the proposed designs. Both Tables have the same throughput. To be more specific, Fig. 3 shows the area ratio plots between the proposed designs and the previous Booth DA designs. As shown in the Figure, the area saving can be up to 30% for the $p = 3$ case and up to 50% for the $p = 4$ case (for the 4 taps case). As the tap number grows, the area saving will be 14% for the $p = 3$ case and 35% for the $p = 4$ case. The sharp rise in the area ratio curve is also due to the $(\log_2 N)$ factor in the adder part. The effect of different word lengths (coefficient or input) is quite small, as shown in Fig. 3. However, the difference becomes larger and larger when the tap number grows.

As stated previously, the major area difference comes from the input sample delays and the Booth selector. When the radix increases, the gate structure of the Booth selector circuits becomes more and more complex and consumes too large an area. A simpler multi-operand adders design

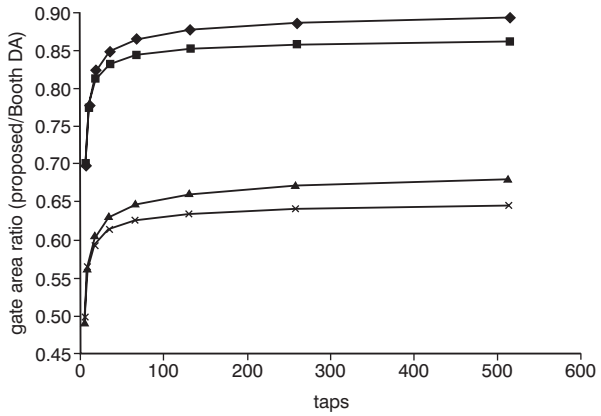

Fig. 3 *Area ratio plot between the proposed p-bits-at-a-time and modified Booth DA methods, starting from taps = 4 ($W_x = W_c = 12$ bits or 24 bits is assumed in the Figure)*

–◆– 12 bits ($p = 3$)
–■– 24 bits ($p = 3$)
–▲– 12 bits ($p = 4$)
–×– 24 bits ($p = 4$)

contributes a portion of the area saving. The area saving of the input sample delays and the coefficient storage is a major part of the area saving when the radix increases. That is because the high radix Booth DA has to transfer the encoded Booth signals rather than the original input samples. In the Booth DA, the generation of odd multiples also requires extra hardware cost. More precisely, take the case of a filter of order $N = 32$ as an example: Table 4 shows the hardware cost. As shown in the Table, the total area saving is 36503.5, of which 75.9% of the area reduction is contributed by the delay savings and the other 24.1% of the area reduction is due to the simpler compressor structure.

Table 2: Hardware costs of the radix-4 Booth DA and the proposed 3-bits-at-a-time method for FIR designs (final VMA is not included here)

Method	AND gate	Booth encoder and selector	Full adders	Input sample delays + pipeline delays + coefficient storage
Radix-4 Booth DA	0	$19 + N \times (W_x + 2) \times 19$	$(N-1) \times (W_c + \log_2 N/2 + 2) \times 7$	$(N-1) \times (1/3 \times W_x - 1) \times 5 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 2) \times 2 \times 8$ $+ N \times (W_c + 2) \times 2 \times 8$
3-bits at a time	$N \times W_x \times 3 \times 2$	0	$(3N-5) \times (W_c + \log_2 N/2 + 2) \times 7$	$(N-1) \times (1/3 \times W_x - 1) \times 3 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 2) \times 2 \times 8$ $+ N \times W_c \times 8$

Table 3: Hardware costs of the radix-5 Booth DA and the proposed 4-bits-at-a-time method for FIR designs (final VMA is not included here)

Method	AND gate	Booth encoder and selector	Full adders	Input sample delays + pipeline delays + coefficient storage
Radix-5 Booth DA	0	$39 + N \times (W_x + 3) \times 39$	$(N-1) \times (W_c + \log_2 N/2 + 3) \times 7$	$(N-1) \times (1/4 \times W_x - 1) \times 9 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 3) \times 2 \times 8$ $+ N \times (W_c + 3) \times 4 \times 8$
4-bits at a time	$N \times W_x \times 4 \times 2$	0	$(4N-6) \times (W_c + \log_2 N/2 + 3) \times 7$	$(N-1) \times (1/4 \times W_x - 1) \times 4 \times 8$ $+ (N-1) \times (W_c + \log_2 N + 3) \times 2 \times 8$ $+ N \times W_c \times 8$

Table 4: A hardware cost example for $N=32$ and $W_x=W_c=24$ bits

Methods	AND gate (T1)	Booth function (T2)	Full adders (T3)	T1+T2+T3	Delays	Total
Radix-5 Booth DA	0	33735	6401.5	40136.5	54680	94816.5
4-bits-at-a-time	6144	0	25193	31337	26976	58313

As to the cycle delay of both designs, the delay of the higher-radix Booth DA is slightly larger than that of proposed multi-bit DA owing to the complex Booth selection circuit. The higher-radix Booth DA also requires extra cycles to generate the odd multiples, which will lower the overall performance.

4.3 Comparison to the bit-parallel input design

The programmable FIR filter design [2] proposed in 1997 also used Booth encoding and DA-like reformulation. In their design, the bit serial input was expanded to a parallel input to get enough throughputs. The proposed design can also be expanded to a parallel input to obtain such throughput. For an 8-tap design with 10-bit coefficient and input samples used for one block design in [2], the hardware cost is:

$$\begin{aligned}&5\times 8\times 11\ \text{(Booth encoder and selector)}\\&\quad+5\times 6\ ((5,5,4)\ \text{compressor}=6\ \text{FA})\\&\quad+5\times 11\ ((3,2)\ \text{compressor}=1\ \text{FA})\\&\quad+5\times 14\ ((4,2)\ \text{compressor}=2\ \text{FA})\\&\quad+17\ ((6,2)\ \text{compressor}=4\ \text{FA})=5\times 8\times 11\times 21\\&\quad+(5\times 6\times 6+11\times 5\times 1+5\times 14\times 2+17\times 4)\times 7\\&\quad=12341\ \text{gates}\end{aligned}$$

Since the design in [2] used a fully parallel input, the Booth encoder cannot be shared for all taps like the design in the Booth DA. The complex Booth encoder and selector cost 9240 gates, which occupies 74.8% of the gate area.

On the other hand, if we use the proposed design for the 8-tap filters with 10-bit coefficients and input samples, the hardware cost is

$$\begin{aligned}&8\times 10\times 10\ \text{(AND gates)}+10\times 6\ ((8,2)\ \text{compressor}\\&\quad=10\ \text{FA})+(13\times 18+18\times 3)\ \text{(FA)}\\&\quad=8\times 10\times 10\times 2+(10\times 6\times 10+13\times 18\\&\qquad+18\times 3)\times 7=7816\ \text{gates}\end{aligned}$$

The proposed design saves 36.7% of the gate area compared with the design in [2]. We do not include input sample delays and other adder circuits since they are the same for both designs. In this design, the main area saving comes from the efficient high-order compressor design that has the same function as the complex Booth-encoder design. These high-order compressor designs can follow the approaches in [16–18]. The only drawback of the high-order compressor is routing irregularity. The irregularity can be reduced by using a (4,2) compressor to construct the high-order compressor [16–18]. When compared to bit-parallel ones or the extended Booth DA, the proposed one is more regular and easily controlled. As to the power and clock distribution, these global nets can be regularly connected in the proposed pipelined architecture. The power consumption of the proposed design is also less than the other design, since the pipeline latch can eliminate the glitch propagation through the addition chain, and the regular compressor design also has a more balanced addition path delay.

5 Conclusion

In this paper, we propose a hardware-efficient pipelined programmable FIR filter design that adopts two techniques: multi-bit DA algorithm formulation and compressor architecture. Due to DA formulation and simpler structures, the delay of the presented design can be easily pipelined to the adder level. As a short summary, the proposed approach has the following advantages. First, it requires less hardware cost with a similar cycle time and latency (W_x/P cycles per input) as the previous Booth DA designs. Second, with DA formulation, it can be easily pipelined to the fast adder level instead of the slow accumulator level. Third, by a multi-bit parallel input, it can provide a design trade-off between the bit-serial one and bit-parallel ones. Fourth, by using (4,2) to construct a (p, q) compressor, the layout complexity can be reduced and easily controlled.

The hardware cost for the 2-bits-at-a-time case can be saved by up to 17% when compared with the previous designs using the Booth DA. This design is especially useful when extended to a parallel input, which can save up to 50% of the hardware cost for 4-bits-at-a-time. Compared with previous fully parallel input designs, the present design can save 36.7% of the area cost.

6 References

1 EDWARDS, B., CORRY, A., WESTE, N., and GREENBERG, C.: 'A single chip ghost canceller'. Proceedings of the IEEE 1992 Custom Integrated Circuits Conference, 1992, pp. 26.5.1–26.5.14
2 CHOI, J.R., JANG, L.H., JUNG, S.W., and CHOI, J.H.: 'Structured design of a 288-tap FIR filter by optimized partial product tree compression', *IEEE J. Solid-State Circuits*, 1997, **32**, (3), pp. 468–476
3 PEARSON, D.J., REYNOLDS, S.K., MEGDANIS, A.C., GOWDA, S., WRENNER, K.R., IMMEDIATO, M., GALBRAITH, R.L., and SHIN, H.J.: 'Digital FIR filters for high speed PRML disk read channels', *IEEE J. Solid-State Circuits*, 1995, **30**, (12), pp. 1517–1523
4 SAMUELI, H.: 'An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients', *IEEE Trans.*, 1989, **CAS-36**, pp. 1044–1047
5 HARTLEY, R.: 'Subexpression sharing in filters using canonical signed digit multipliers', *IEEE Trans.*, 1996, **CAS-43**, (10), pp. 677–688
6 PELED, A., and LIU, B.: 'A new hardware realization of digital filters', *IEEE Trans.*, 1974, **ASSP-22**, (6), pp. 456–462
7 WEI, C.H., and LOU, J.S.: 'Multimemory block structure for implementing a digital adaptive filter using distributed arithmetic', *IEE Proc. G., Electron. Circuits Syst.*, 1986, **133**, pp. 19–26
8 WHITE, S.A.: 'Applications of distributed arithmetic to digital sequence processing: a tutorial review', *IEEE ASSP Mag.*, 1989, **6**, (3), pp. 5–19
9 JONES, D.L.: 'Efficient computation of time-varying and adaptive filters', *IEEE Trans. Signal Process.*, 1993, **SP-41**, (3), pp. 1077–1086
10 SID-AHMED, M.: 'A systolic realization for 2-D digital filters', *IEEE Trans.*, 1989, **ASSP-37**, pp. 560–565
11 REUVER, D., and KLAR, H.: 'A configurable convolution chip with programmable coefficients', *IEEE J. Solid-State Circuits*, 1992, **27**, (7), pp. 1121–1123
12 MOLONEY, D., O'BRIEN, J., O'ROURKE, E., and BRIANTI, F.: 'Low-power 200-Msps, area efficient, five-tap programmable FIR filter', *IEEE J. Solid-State Circuits*, 1998, **33**, (3), pp. 1134–1138

13 LEE, H.-R., JEN, C.-W., and LIU, C.-M.: 'A new hardware-efficient architecture for programmable FIR filters', *IEEE Trans.*, 1996, **CAS-43**, (9), pp. 637–644
14 DAWOUD, D.S.: 'Realization of pipelined multiplier-free FIR digital filter'. Proceedings of the IEEE 1999 Africon Conference, 1999, pp. 335–338
15 MAC SORLEY, O.L.: 'High speed arithmetic in binary computers', *IRE Proc.*, 1961, **49**, pp. 67–91
16 VILLEGER, D., and OKLOBDZIJA, V.G.: 'Evaluation of Booth encoding techniques for parallel multiplier implementation', *Electron. Lett.*, 1993, **29**, (23), pp. 2016–2017
17 SONG, P., and MICHELLI, G.D.: 'Circuits and architecture trade-offs for high speed multiplication', *IEEE J. Solid-State Circuits*, 1991, **26**, (9), pp. 1184–1198
18 OKLOBDZIJA, V.G., VILLEGER, D., and LIU, S.S.: 'A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach', *IEEE Trans. Comput.*, 1996, **45**, (3), pp. 294–306
19 Compass, PASSPORT library, 0.6 micron 5-volt standard cell library, 1996