Low Power, Area Efficient Digit-Serial FIR Filter using Multiple Constant Multiplications (MCM)

Tiya Reba Stephen
PG Student
Department of ECE
SKCET, Coimbatore

B. Veerasamy
Assistant Professor
Department of ECE
SKCET, Coimbatore

ABSTRACT

Efficient algorithms and architectures are existing for the design of low-complexity bit-parallel multiple constant multiplication (MCM).This operation dominates the complexity of many digital signals processing system. Alternative to this, digit-serial MCM design is available with less complexity. But it is not as much popular as the former one. In this paper, the gate –level area and power of digit-serial MCM design is tried to optimize. So initially from the basic parallel designs, like shift –adds implementation, the common sub-expression elimination and graph-based method are used. From this the efficient one is selected, that is the GB technique and is applied to digit-serial design. Then the newly designed MCM block will be placed to the multiplier block of an FIR filter. Thus comparing to bit-parallel FIR filter design, digit-serial design has 41% of power reduction and 40.5% of area reduction and are independent of data word-length.

Keywords

Digit-serial arithmetic, finite impulse response (FIR) filters, multiple constant multiplications (MCM).

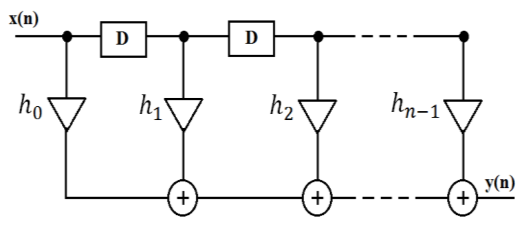
1. INTRODUCTION

Filtering is the most important technique in the signal processing to choose the limited frequency spectrum in some specific frequency band [1-2]. In general, the digital filter is a kind of the generic filter, but used widely. It is necessary to know the design of the digital filter. In digital signal processing **(**DSP) systems finite impulse response (FIR) filters are of great importance. The FIR filters are of non-recursive type. The present output sample depends on the present input samples and previous input samples. In the common case, the impulse response is finite because there is no feedback in the FIR. As there is no feedback, it will guarantee that the impulse response will be finite. Therefore, the term "finite impulse response" is nearly synonymous with "no feedback".

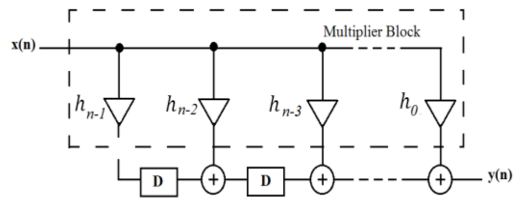
The characteristics of FIR filter in linear-phase and feed-forward implementations make them very useful for building stable high-performance filters. Two types of filter implementations are using, they are (1) direct-form [Fig. 1(a)] (2) transposed-form [Fig. 1(b)]. Both architectures have similar complexity in hardware. But the transposed form is generally preferred because of its higher performance and power efficiency [1].

The multiplier block has significant impact on the complexity and performance of the filter design; because a large number of constant multiplications are required. Multiplications are often implemented with shift-and-add operations for hardware efficiency and are generally known as the multiple constant multiplications (MCM) [Fig. 1(c)] operation. The decomposition of multiplications into shifts and adds is such that as much intermediate computation results (partial products) can be reused as possible.

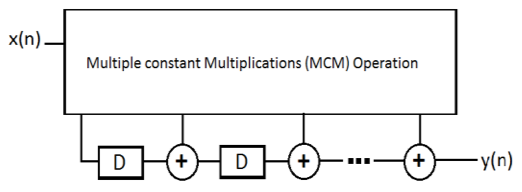
Full flexibility of a multiplier is not needed for the constant multiplication, since filter coefficients are fixed and determined beforehand by the DSP algorithms. So area, delay, and power-efficient multiplier architectures, such as Wallace [2] and modified Booth multipliers [3] are not needed. Hence, the multiplication of filter coefficients with the input data is generally implemented under a shift-adds architecture. In this each constant multiplication is realized using addition/subtraction and shift operations in an MCM operation [Fig. 1(c)].



(a)



(b)



(c)

Fig.1: FIR filters implementations. (a) Direct form. (b) Transposed form with generic multipliers. (c) Transposed form with an MCM block.

For the shift-adds implementation of constant multiplications, generally a straightforward method known as digit-based recoding is used. Initially it defines the constants in binary. Then, in the binary representation of the constant, for each “1” according to its bit position, it shifts the variable and adds the shifted variables to obtain the result. As an simple example, consider the constant multiplications of 29x and 43x. Their decompositions in binary are shown below:

29x= (11101) bin x=x < < 4 + x < < 3+ x < < 2 + x

43x= (101011) bin x=x < < 5 + x < <3 + x < < 1 + x

In the fig.2. Shift-add implementation of 29x and 43x is shown. Here sharing of partial product is not done. Digit-based decoding [5] technique does not allow sharing of common partial product, which in turn will reduce the area and power dissipation of the MCM design at the gate level. The MCM problem can be defined by finding minimum number of addition and subtraction operations, which will implement the constant multiplications.

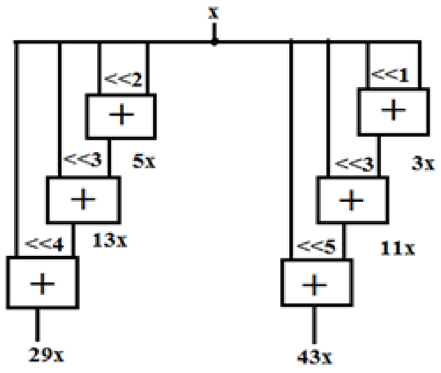


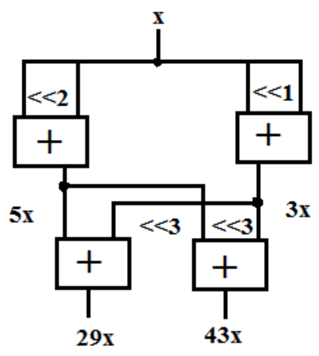
Fig. 2. Shift-adds implementations of 29x and 43x without partial product sharing.

This operation requires 6 additions as shown above. If the common partial products are shared, that can reduce the number of operations and thus reducing the power dissipation and area of the MCM design at the gate level. Finding the minimum number of addition and subtraction operations that implement the constant multiplications is the optimization problem, called MCM problem.. In bit-parallel design of constant multiplications, without representing any area cost, shifts can be realized only using wires in hardware.

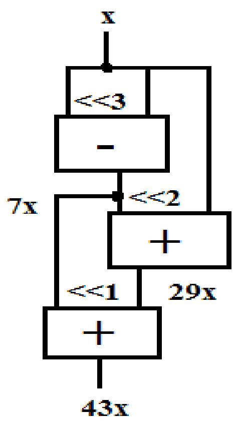
The MCM problem can be solved using two algorithms: common subexpression elimination (CSE) algorithms [6]-[7] and graph-based (GB) [8]-[9] techniques. In the CSE algorithms, initially extracting all possible subexpressions from the binary, canonical signed digit (CSD) [6], or minimal signed digit (MSD) [10] representations of the constants. Then finds the “best” subexpression generally the most common to be shared among the constant multiplications. The GB methods usually yields better solution and are not limited to any particular number representation.

It will consider a larger number of different implementation of constants. In Fig. 3.a the implementation of 29x and 43x using the exact CSE algorithm [7] gives a solution with four operations by finding the most common partial products 3x = $(11)_{bin}$x and 5x = $(101)_{bin}$x .

Here constants are defined using binary. In fig.3.b. The exact GB algorithm is implemented by sharing the common partial product 7x in both multiplication and finds a solution with minimum number of operations. In the exact CSE algorithm the partial product 7x = $(111)_{bin}$x is not possible to extract from the binary representation of 43x.



(a)



(b)

Fig 3.a. Exact CSE algorithm b. GB technique

Above mentioned algorithms are implemented by parallel processing of the input data. Bit-parallel processing is the existing method. In order to improve the efficiency by reducing the area and power, digit-serial method can be adopted.

The rest of this paper proceeds as follows. Section II describes the background concept Section III proceeds with related work. Section IV describes the proposed work. In section v experimental results are shown and in section VI gives the conclusion.

2. BACKGROUND

2.1. Number Representation

For the binary representation of a number, initially decomposing the number in to a set of additions of power of 2.In the case of representing a number using a signed digit system , it makes use of positive and negative digits,{1,0,-1} . For example the CSD representation [11] is a signed digit system that has a unique representation for each number and it will verify the following main properties: 1) two nonzero digits are not adjacent; and 2) the number of nonzero digits is minimum. Any n digit number in CSD has at most [(n+1)/2] nonzero digits and, on average, the number of nonzero digits is reduced by 33% when compared to binary [1]. For obtaining the MSD [10] representation, the first property of CSD has to be dropped [2]. Thus, using MSD and CSD, a constant can be represented in several ways with a minimum number of nonzero digits.

Consider the constant 23 defined in six bits. Its binary representation 010111 includes four nonzero digits. It is represented as 101001 in CSD, and both 101001 and 011001 denote 23 in MSD using three nonzero digits (where 1 stands for−1) [1].

2.2. Digit-Serial Arithmetic

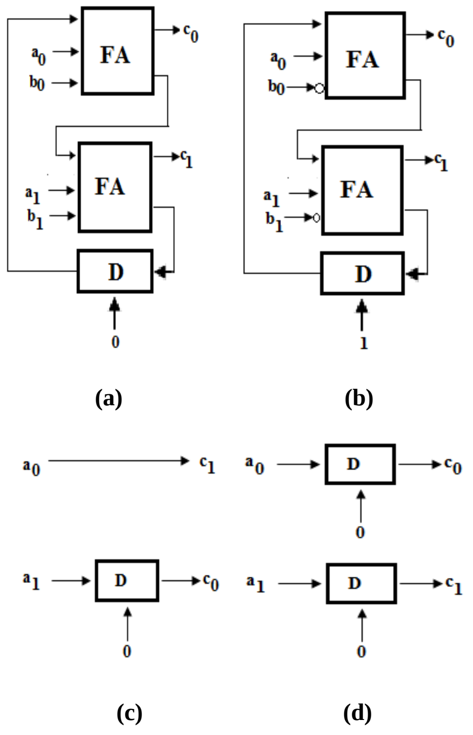


Fig. 4: Digit-serial operations when d = 2. (a) Addition operation. (b) Subtraction operation. (c) Left shift by one time. (d) Left shift by two times

For digit-serial design, the input data is processed serially by dividing it into d bits and then applying each d-bit data in parallel. When the digit size d is equal to 1 and equal to input data wordlength, then a special case called bit-serial and bit-parallel will occur respectively. The basic digit-serial arithmetic operations are, the digit-serial addition, subtraction, and left shift operations are depicted in Fig. 4 . Here digit size d equal to 2 is used., where the bits with index 0 and 1 denote the least significant bit (LSB) and the most significant bit (MSB), respectively.

Notice from Fig. 4(a) that a digit-serial addition operation, in general, requires d full adders (FAs) and one D flip-flop. The subtraction operation [Fig. 4(b)] is implemented using 2's complement. Here the initialization of the D flip-flop with 1 and additional d inverter gates with respect to the digit-serial addition operation is used. In a left shift operation [Fig. 4(c) and (d)], the number of required D flip-flops is equal to the shift amount and is realized in d layers (one for each bit).

As an example of digit-serial realization of constant multiplications under the shift-adds architecture, Fig. 5 presents the digit-serial implementation of 29x and 43x illustrated in Fig. 3(b) when d is 2. For the sake of clarity, the initializations of D flip-flops are omitted in this figure. As can be easily observed, the network includes two digit serial additions, one digit-serial subtraction, and five D flip-flops for all the left shift operations.

In this network, at each clock cycle, two bits of the input data x(x1x0) are applied to the network input, and at the outputs of each digit-serial addition/subtraction operation two bits of a constant multiplication are computed. In general, dbits are processed at each clock cycle. The digit-serial design of the MCM operation occupies significantly less area when compared to its bit-parallel design since the area of the digit-serial design is not dependent on the bit width of the input data. However, the latency is determined in terms of clock cycles.

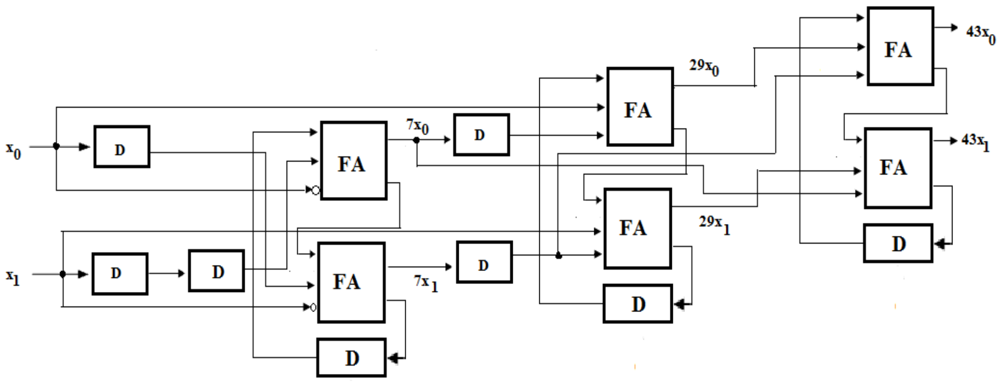


Fig.5: Digit-serial design of shift-adds implementation of 29x and 43x given in Fig. 3(b) when d is 2.

3. RELATED WORK

The exact CSE algorithms that formalize the MCM problem as a 0–1 ILP problem were introduced in [12] and [13]. The target constants are defined under a number representation in these algorithms and all possible implementations of constant multiplications are extracted from the representations of constants. The model simplification and problem reduction techniques for the exact CSE algorithms were presented in [7] and [14]. Prominent CSE heuristics were proposed in [7], [8], and [15]. In [7] author proposed an algorithm that finds all the minimum singed digit (MSD) representation of a constant and it also presents an algorithm to synthesize digital filters based on the MSD representation. Based on the number system used for the implementation, the hardware complexity varies.

The exact GB algorithms define a solution with the minimum number of operations. This will be in breadth-first and depth-first manners. Optimal and heuristic are the two parts in which efficient GB algorithms can be explained. These are introduced in [8], [9], and [16]. The graph representation concept was introduced by Bull and Horrocks. In their optimal parts, each target constant that can be implemented with a single operation is synthesized.

In the target set, if there exist unimplemented elements, then they switched to their heuristic parts. Here the required intermediate constants are found.. In their optimal parts, each target constant that can be implemented with a single operation is synthesized. The RAG-n algorithm is having two parts and the algorithm [8] initially chooses a single unimplemented target constant with the smallest single coefficient cost. This is then evaluated by the algorithm of [17], and after that it is synthesized with a single operation including one (two) intermediate constant(s) that has (have) the smallest value in its heuristic part. The H cub algorithm [11] selects a single intermediate constant which yields the best cumulative benefit over all unimplemented target constants for the implementation of each target constant. The approximate algorithm [12] computes all possible intermediate constants. This can be synthesized with the current set of implemented constants by using a single operation and chooses the one that leads to the largest number of synthesized target constants.

4. PROPOSED WORK

In this project digit-serial FIR filter is implemented. When the bit-parallel implementation cannot meet the delay requirements, digit-serial computation is used. Thus, the trade-off between area and delay can be explored by changing the digit-size. Also bit-parallel design requires excessive hardware. Here, the data words are divided in to digit sets, consisting of d bits which are processed one by one. Comparing to bit-parallel design, digit-serial architectures offers lower complexity. This is possible because of the less area of digit-serial operators and is independent of data wordlength. Bit-parallel designs are free of hardware but in digit-serial, the shifts require the use D flip- flops. Hence, while choosing the algorithms, the high-level algorithms should take into account for the sharing of shift operations as well as the sharing of addition/subtraction operations in digit-serial MCM design. Furthermore, finding the minimum number of operations realizing an MCM operation does not always yield an MCM design with optimal area at the gate level. Hence, the high-level algorithms should consider the implementation cost of each digit-serial operation at the gate level [18].

In this paper, initially the shift-add implementation is proposed, then the exact CSE algorithm. Since there are still instances which the exact CSE algorithm cannot handle, an approximate GB algorithm [14] is proposed that iteratively finds the "best" partial product which leads to the optimal area in digit-serial MCM design at the gate level. From these the better one, that is the GB based technique is selected and is implemented in the digit serial design. This digit-serial design is then implemented in the MCM block of the FIR filter. The power and area of this design is then compared with the existing bit-parallel design.

5. EXPERIMENTAL RESULTS

The simulation result of digit-serial and bit-parallel FIR filters is obtained using XILINX ISE 8.1i with SPARTAN 2E XC2S150E as target device. Simulation is also carried out in ModelSim SE 6.3f using VHDL language. The simulation result shows that the digit-serial design of FIR filter is having less area compared to the bit-parallel design. The power analysis shows that the new design has a power consumption of 119mw which is less than the other approaches. The total equivalent gate count is 320 wherein the other approach is having gate count of 644. Thus it is clear that the digit-serial design is having 41% less power compared to the existing method.

Table.1. Area and power comparison of existing and proposed system

	Existing System	**Proposed System**
Area(gate count)	644	320
Power (mW)	160	119

6. CONCLUSION

In this paper, CSE algorithm and GB technique for designing digit-serial MCM operation are introduced. Since there are still instances with which the exact CSE algorithm cannot cope, an approximate GB algorithm is proposed that finds the best partial products in each iteration which yield the optimal gate-level area and power reduction in digit-serial MCM design compared to parallel design. This paper also introduced the design architectures for the digit-serial MCM operation and FIR filters. It was shown that the realization of digit-serial FIR filters under the shift-adds architecture yields 40.5% area reduction and 41% of power reduction, when compared to the filter designs whose multiplier blocks are implemented using bit-parallel constant multipliers. It is observed that a designer can find the circuit that fits best in an application by changing the digit size. In order to reduce noise interruption, adaptive filters are used. Adaptive filter can also be implemented in both bit-parallel and digit-serial design. As in FIR filter, digit-serial design is the efficient one. As an application adaptive filters can be used to reduce noise from ECG signals.

7. REFERENCES

[1] L. Wanhammar, DSP Integrated Circuits. New York: Academic, 1999.

[2] C. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., vol. 13, no. 1, pp. 14–17, Feb. 1964.

[3] W. Gallagher and E. Swartzlander, "High radix booth multipliers using reduced area adder trees," in Proc. Asilomar Conf. Signals, Syst. Comput., vol. 1. Pacific Grove, CA, Oct.–Nov. 1994, pp. 545–549.

[4] J. McClellan, T. Parks, and L. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," IEEE Trans. Audio Electroacoust., vol. 21, no. 6, pp. 506–526, Dec. 1973.

[5] M. Ercegovac and T. Lang, Digital Arithmetic. San Mateo, CA: Morgan Kaufmann, 2003.

[6] R. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[7] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplications," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 27, no. 6, pp. 1013–1026, Jun. 2008.

[8] Dempster and M. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[9] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," ACM Trans. Algor., vol. 3, no. 2, pp. 1–39, May 2007.

[10] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on minimal signed digit representation," in Proc. DAC, 2001, pp. 468–473.

[11] Avizienis, "Signed-digit number representations for fast parallel arithmetic," IRE Trans. Electron. Comput., vol. 10, no. 3, pp. 389–400, Sep. 1961.

[12] P. Flores, J. Monteiro, and E. Costa, "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications," in Proc.Int. Conf. Comput.-Aided Design, Nov. 2005, pp. 13–16.

[13] O. Gustafsson and L. Wanhammar, "ILP modelling of the common subexpression sharing problem," in Proc. ICECS, 2002, pp. 1171–1174.

[14] Y.-H. Ho, C.-U. Lei, H.-K. Kwan, and N. Wong, "Global optimization of common subexpressions for multiplierless synthesis of multiple constant multiplications," in Proc. ASPDAC, 2008, pp. 119–124.

[15] M. Potkonjak, M. Srivastava, and A. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," IEEE Trans. Comput-Aided Design Integr. Circuits Syst., vol. 15, no. 2, pp. 151–165, Feb.1996.

[16] L. Aksoy, E. Gunes, and P. Flores, "Search algorithms for the multiple constant multiplications problem: Exact and approximate," J. Micro-process. Microsyst., vol. 34, no. 5, pp. 151–162, Aug. 2010.

[17] `Dempster and M. Macleod, "Constant integer multiplication using minimum adders," IEE Proc.-Circuits, Devices, Syst., vol. 141, no. 5, pp. 407–413, Oct. 1994.

[18] L. Aksoy, C. Lazzari, E. Costa, P. Flores, J. Monteiro, "Design of Digit-Serial FIR Filters: Algorithms, Architectures, and a CAD Tool," IEEE Trans. VLSI Syst, vol. 21, no. 3, Mar 2013.