International Research Journal of Education and Technology
ISSN 2581-7795

OPTIMIZATION OF AREA IN DIGIT SERIAL MULTIPLE CONSTANT MULTIPLICATION AT GATE LEVEL

Logeshwari. N [1], Thilagam.S [2]

[1,2] *Department of Electronics and Communication Engineering*
Kumaraguru college of technology, Coimbatore

Abstract - *In the last two decades, many efficient algorithms and architectures have been introduced for the design of low complexity bit-parallel multiple constant multiplications (MCM) operation which increases the complexity of many digital signal processing systems. Multiple constant multiplication (MCM) is an efficient way of implementing several constant multiplications with the same input data. The coefficients are expressed using shifts, adders, and subtracters. On the other hand, little attention has been given to the digit-serial MCM design that offers alternative low complexity MCM operations. In this paper, we address the problem of optimizing the gate-level area in digit-serial MCM designs.*

***Key Words*: 0–1 integer linear programming (ILP), digit-serial arithmetic, finite impulse response (FIR) filters, gate-level area optimization, multiple constant multiplications.**

1. INTRODUCTION

Finite impulse response (FIR) filters are of great importance in digital signal processing (DSP) systems since their characteristics in linear-phase and feed-forward implementations make them very useful for building stable high-performance filters. The direct and transposed-form FIR filter implementations are there, respectively. Although both architectures have similar complexity in hardware, the transposed form is generally preferred because of its higher performance and power efficiency .The multiplier block of the digital FIR filter in its transposed form, where the multiplication of filter coefficients with the filter input is realized, has significant impact on the complexity and performance of the design because a large number of constant multiplications are required. This is generally known as the multiple constant multiplications (MCM) operation and is also a central operation and performance bottle neck in many other DSP systems such as fast Fourier transforms, discrete cosine

2. LITERATURE REVIEW

2.1 "Low-Cost FIR Filter Designs Based on Faithfully Rounded Truncated Multiple Constant Multiplication/Accumulation"

A generic flow of FIR filter design and implementation can be divided into three stages: finding filter order and coefficients, coefficient quantization, and hardware optimization, as shown in Fig. 1

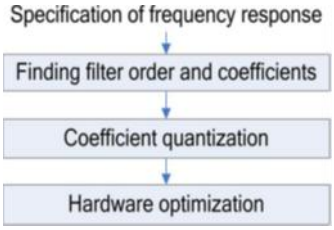

Fig -1: Three stages in FIR design

In the first stage, the filter order and the corresponding coefficients of infinite precision are determined to satisfy the specification of the frequency response. Then, the coefficients are quantized to finite bit accuracy. Finally, various optimization approaches such as CSE are used to minimize the area cost of hardwareplementations. Most prior FIR filter implementations focus on the hardware optimization stage After FIR filter perations, the output signals have larger bit width due to bit width expansion after multiplications. In many practical situations, only partial bits of the full-precision outputs are needed. For example, assuming that the input signals of the FIR filter have 12 bits and the filter coefficients are quantized to 10 bits, the bit width of the resultant FIR filter output signals is at least 22 bits, but we might need only the 12 most significant bits for subsequent processing. In this brief, we adopt the direct FIR structure with MCMA because the area cost of the flip-flops in the delay elements is smaller compared with that of the transposed form. Furthermore, we jointly consider the three design stages in Fig. 1 in order to achieve more efficient hardware design with faithfully rounded output signals. Unlike conventional uniform quantization of filter coefficients with equal bit width, the nonuniform

quantization technique with possibly different bit widths is adopted in this brief.

Initially, subroutine Parks McClellan () is used to find the filter order M for the given frequency response. Then, we quantize the coefficients with enough bits and generate the set of uniformly quantized coefficients ai with equal bit width B. The subroutine freq_resp_satisfied () checks if the frequency response is still satisfied after quantization. After coefficient quantization, we perform recoding to minimize the number of nonzero digits. In this brief, we consider CSD recoding with digit set of {0, 1,−1} and radix-4 modified Booth recoding with digit set of {0, 1,−1, 2,−2} and select the one that results in smaller area cost. While most FIR filter designs use minimum filter order, we observe that it is possible to minimize the total area by slightly increasing the filter order. Therefore, the total area of the FIR filter is estimated using the subroutine area_cost_estimate() using the approach in [20]. Indeed, the total number of PPBs in the MCMA is directly proportional to the number of FA cells required in the PPB compression because a FA reduces one PPB. After Step 1 of uniform quantization and filter order optimization, the non-uniform quantization in Step 2 gradually reduces the bit width of each coefficient until the frequency response is no longer satisfied. Finally, we fine-tune the non-uniformly quantized coefficients by adding or subtracting the weighting of LSB of each coefficient and check if further bit width reduction is possible. Using the algorithm, we can find the filter order M and the non-uniformly quantized coefficients that lead to minimized area cost in the FIR filter implementation.

2.2"A Computer Program for Designing Optimum FIR Linear Phase Digital Filters"

2.3 "Multiple Constant Multiplication for Digit-Serial Implementation of Low Power FIR Filters"

In the n-dimensional Reduced Adder Graph (RAG-n) algorithm was introduced. This algorithm is known to be one of the best MCM algorithms in terms of number of adders. Based on this algorithm an n-dimensional Reduced Shift and Add Graph (RSAG-n) algorithm has been developed [10] that not only tries to minimize the adder cost, but also the number of shifts. However, this algorithm has an increased adder cost, which will be dominating for larger digit-sizes Here, an n-dimensional Reduced Add and Shift Graph (RASG-n) algorithm is proposed. The new algorithm is a hybrid of the RAG-n and RSAG-n algorithms. RASG-n work with odd coefficients, like RAG-n and only realizes one coefficient in each iteration, like RSAG-n. When it is possible to realize more than one coefficient RASG-n selects the one that require the lowest number of additional shifts. This makes it possible for RASG-n to minimize both the number of adders and shifts in an effective way. These algorithms are graph based. Node values are referred to as fundamentals. Realized coefficients are removed from the coefficient set and added to an interconnection table that specifies how the value is obtained.

The termination condition of the algorithm is that the coefficient set is empty. The steps in the RSAG-n algorithm are;

1. Divide even coefficients by two until odd, and save the number of times each coefficient is divided. These shifts at the outputs can be considered to be free when other coefficients are synthesized.
2. Remove zeros, ones, i.e., coefficients which corresponds to a power-of-two, and repeated coefficients from the coefficient set.
3. Compute the single-coefficient adder cost for each coefficient, which is done by using a look-up table.
4. Compute a sum matrix based on power-of-two multiples of the fundamental values included in the interconnection table. At start this matrix is and is then extended when new fundamentals are added. If any required coefficients are found in the matrix, compute the required number of shifts. Find the coefficients which require the lowest number of additional shifts, and select the smallest of those. Add this coefficient to the interconnection table and remove it from the coefficient set.
5. Repeat step 4 until no required coefficient is found in the sum matrix.
6. For each remaining coefficient, check if it can be obtained by the strategies. For both cases two new adders are required. If any coefficients are found, select the smallest coefficient of those which require the lowest number of additional shifts. Add this coefficient and the extra fundamental to the interconnection table. Remove the coefficient from the coefficient set.
7. Repeat step 5 and 6 until no required coefficient is found.
8. Choose the smallest coefficient with lowest single coefficient adder cost. Different sets of fundamentals that can be used to realize the coefficient are obtained from a look-up-table. For each set, remove fundamentals that are already included in the interconnection table and compute the required number of shifts. Find the sets which require the lowest number of additional shifts, and of those, select the set with smallest sum. Add this set and the coefficient to the interconnection table. Remove the coefficient from the coefficient set.

3. EXISTING SYSTEM

The direct and transposed-form FIR filter implementations are illustrated in Fig.2(a) and (b), respectively. Although both architectures have similar complexity in hardware, the transposed form is generally preferred because of its higher performance and power efficiency. The multiplier block of the digital FIR filter in its transposed form [Fig.2 (b)], where the multiplication of filter coefficients with the filter input is

International Research Journal of Education and Technology
ISSN 2581-7795

realized, has significant impact on the complexity and performance of the design because a large number of constant multiplications are required. This is generally known as the multiple constant multiplications (MCM) operation and is also a central operation and performance bottleneck in many other DSP systems such as fast Fourier transforms, discrete cosine transforms (DCTs), and error-orrecting codes.

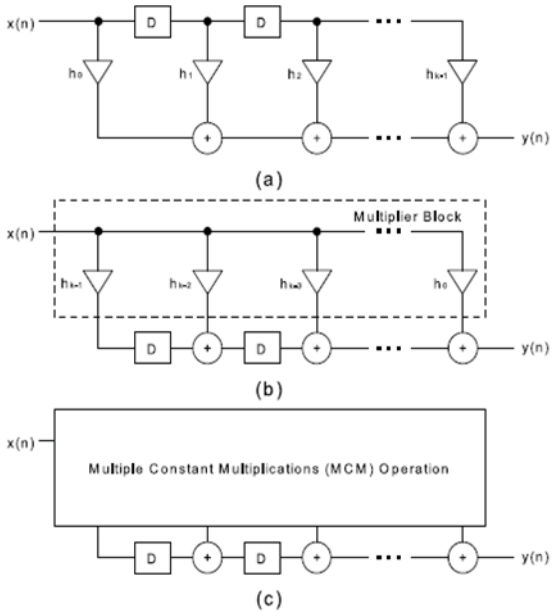

Fig -2: FIR filters implementations. (a) Direct form. (b) Transposed form with generic multipliers. (c) Transposed form with an MCM block.

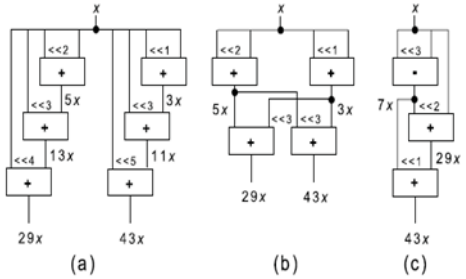

Fig.-3: Shift-adds implementations of 29x and 43x. (a) Without partial product sharing and with partial product sharing. (b) Exact CSE algorithm. (c) Exact GB algorithm

.

However, the digit-based recoding technique does not exploit the sharing of common partial products, which allows great reductions in the number of operations and, consequently, in area and power dissipation of the MCM design at the gate level. Hence, the fundamental optimization problem, called the MCM problem, is defined as finding the minimum number of addition and subtraction operations that implement the constant multiplications. Note that, in bit-parallel design of constant multiplications, shifts can be realized using only wires in hardware without representing any area cost. The algorithms designed for the MCM problem can be categorized in two classes: common sub expression elimination (CSE) algorithms and graph-based (GB) techniques.

The CSE algorithms initially extract all possible sub expressions from the representations of the constants when they are defined under binary, canonical signed digit (CSD), or minimal signed digit (MSD). Then, they find the "best" sub expression, generally the most common, to be shared among the constant multiplications. The GB methods are not limited to any particular number representation and consider a larger number of alternative implementations of a constant, yielding better solutions than the CSE algorithms. Returning to our example in Fig.3, the exact CSE algorithm of gives a solution with four operations by finding the most common partial products 3x = (11) binx and 5x = (101) binx when constants are defined under binary, as illustrated in Fig. 2(b). On the other hand, the exact GB algorithm finds a solution with the minimum number of operations by sharing the common partial product 7x in both multiplications, as shown in Fig.3 (c). Note that the partial product 7x = (111) binx cannot be extracted from the binary representation of 43x in the exact CSE algorithm.

However, all these algorithms assume that the input data x is processed in parallel. On the other hand, in digit-serial arithmetic, the data words are divided into digit sets, consisting of d bits that are processed one at a time. Since digit serial operators occupy less area and are independent of the data word length, digit-serial architectures offer alternative low complexity designs when compared to bit-parallel architectures. However, the shifts require the use of D flip-flops, as opposed to the bit-parallel MCM design where they are free in terms of hardware. Hence, the high-level algorithms should take into account 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.

4. EXPERIMENTAL RESULTS

The experimental results indicate that the complexity of digit-serial mcm designs can be further reduced using the high-level optimization algorithms proposed in this paper.

International Research Journal of Education and Technology
ISSN 2581-7795

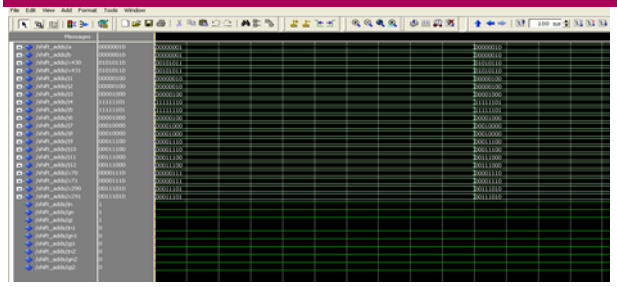

Fig -4: Simulated result of shift adds implementation

It was shown that the realization of digit-serial fir filters under the shift-adds architecture yields significant area reduction when compared to the filter designs whose multiplier blocks are implemented using digit-serial constant multipliers. It is observed that a designer can find the circuit that fits best in an application by changing the digit size.

The shift adds operation for the targets (29x and 43x) is performed. This implementation produces the constants of 29x and 43x and their multiples (i.e.) if the given input is 3(00000011), the output of shift adds operation will be 87(01010111) and 129(10000001). The same constants are implemented using exact CSE algorithm (fig 7.2).

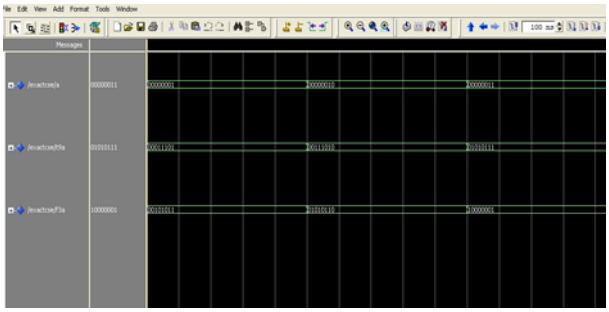

Fig -5: Simulated result of exact CSE algorithm and GB algorithm

In approximate graph base algorithm (GB technique). In which, the optimization of gate-level area problem in digit-serial MCM design is an NP-complete problem due to the NP-completeness of the MCM problem. Thus, naturally, there will be always 0–1 ILP problems generated by the exact CSE algorithm that current 0–1 ILP solvers find difficult to handle. Hence, the GB heuristic algorithms, which obtain a good solution using less computational resources, are indispensable. In GB algorithm, the operation is same as the CSE algorithm but GB uses less resources than the others. Hence, the area and cost size are reduced using this algorithm.

REFERENCES

[1] L. Aksoy, E. Costa, P. Flores, and J. Monteiro, "Optimization of area in digital FIR filters using gate-level metrics," in Proc. DAC, 2007

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

[3] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic,"

[4] R. Hartley and P. Corbett, "Digit-serial processing techniques," Jun. 1990.

[5] 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.

[6] J. McClellan, T. Parks, and L. Rabiner, "A computer program for designing optimum FIR linear phase digital filters," Dec. 1973.

[7] H. Nguyen and A. Chatterjee, "Number-splitting with shift-and-add decomposition for power and hardware optimization in linear DSP synthesis," Aug. 2000

[8] K. Johansson, O. Gustafsson, A. Dempster, and L. Wanhammar, "Algorithm to reduce the number of shifts and additions in multiplier blocks using serial arithmetic," May 2004.

[9] Y.-N. Chang, J. Satyanarayana, and K. Parhi, "Low-power digit-serial multipliers,"1997.

[10] F. Dittmann, B. Kleinjohann, and A. Rettberg, "Efficient bit-serial constant multiplication for FPGAs," 2003.