An Efficient Constant Multiplier Architecture Based On Vertical-Horizontal Binary Common Sub-Expression Elimination Algorithm

Veerraju Kaki¹, Chinaraju Manda²
¹²Department of Electronics and Communication Engineering, INDIA

ABSTRACT

Constant multiplier is a preliminary unit of FIR filter which serves the purpose of multiplying the input with set of coefficients to get desired filter response. High performance DSP systems are implemented in custom hardware, in which the designer has the ability to choose which logic elements will be used to perform the computation. By exploiting the properties of binary multiplication, it is possible to realize constant multiplication with fewer logic resources than required by a generic multiplier.

In this thesis, a thorough analysis of the existing algorithms, the underlying frameworks, and the associated properties is provided. This project based on vertical-horizontal binary common sub-expression elimination (VHBCSE) algorithm attempt to address the weaknesses of the existing fixed bit algorithms such as 2-bit and 3-bit binary common sub-expression algorithms. We also included new strategies which are fundamentally different from the existing methods. Binary common sub-expression elimination algorithms are known to be very efficient in reducing the number of logical operations that are to be performed in constant multiplication there by accelerating the speed of chip as well as reducing power consumption. It is also efficient in reducing the number of adder cells required for the computations.

Keywords— vertical-horizontal binary common sub-expression elimination (VHBCSE), Digital Signal Processing (DSP), Finite Impulse Response (FIR), Generic Multiplier

I. INTRODUCTION

Constant Multiplier:

The purpose of a constant multiplier is to multiply input values with a set of coefficients, which is a common operation in Digital systems. Multiplying a variable by a set of known constant coefficient is a common operation in many digital signal processing (DSP) algorithms. Compared to other common operations in DSP algorithms, such as addition, subtraction, using delay elements, etc., multiplication is generally the most expensive.

There is a trade-off between the amount of logic resources used (i.e. the amount of silicon in the integrated circuit) and how fast the computation can be done. Compared to most of the other operations, multiplication requires more time given the same amount of logic resources and it requires more logic resources under the constraint that each operation must be completed within the same amount of time. A general multiplier is needed if one per forms multiplication between two arbitrary variables. However, when multiplying by a known constant, we can exploit the properties of binary multiplication in order to obtain a less expensive logic circuit that is functionally equivalent to simply asserting the constant on one input of a general multiplier.

II. COMMON SUB-EXPRESSION ELIMINATION METHOD

The target multiplication process involves many partial product terms but consider only unique terms and sharing these terms for obtaining products for duplicate ones will reduce complexity and power consumption.
A CSE algorithm using binary representation of coefficients for the implementation of constant multiplier used in higher order FIR filter with a fewer number of adders than any other algorithms. CSE method is more efficient in reducing the number of adders needed to realize the multipliers when the filter coefficients are represented in the binary form. The observation is that the number of unpaired bits (bits that do not form Common Sub-expressions (CSs)) is considerably few for binary coefficients compared to other schemes. Particularly for higher order FIR filters. The Binary CSE (BCSE) algorithm deals with elimination of redundant binary common sub-expression that occurs within the coefficients. The BCSE technique focuses on eliminating redundant computations in coefficient multipliers by reusing the most common binary bit patterns (BCSs) present in coefficients. The number of BCSs that can be formed in an n-bit binary number is

No. of BCSE”s = $2^n - (n + 1)$.

Where n=length of BCSE algorithm (for ex,2-bit,3-bit..)

In a reconfigurable constant multiplier, the coefficient values can be dynamically programmable. Therefore, the idea behind the reconfigurable multiplier is to consider the worst case (which involves the largest number of addition steps) whereby all the relatively better cases will also be taken care of. Hence, considering a reconfigurable multiplier having 16-bit input (X) and the 16-bit coefficient (H), the worst case condition will occur for the coefficient of values 16'HFFFF. Shift and add based multiplication operation between the inputs (X) with this coefficient (16'HFFFF) values can be written as

$X*H=-------$

Thus the equation represents a partial product for corresponding bit in coefficient H. The division operation is because the coefficient is fractional value which is less than one. Hence the result of multiplication will not exceed the length of input X. From the equation 2.1 we can observe that the terms are inter related i.e. one can be obtained from any other one by shifting either left or right. This flexibility of binary expressions can be exploited to eliminate the redundant computation by producing only unique partial products and generating the remaining from hardwired shifting will make the design efficient and faster. Algorithms such as 2-bit, 3-bit BCSE are proposed based on this flexibility.

An n-bit binary sub expression consists of n-bit vector terms and partial products corresponding to these are only $2^n - (n+1)$. The remaining terms can be obtained from these terms or from input just by shifting them to right.

### III. EXISTING ALGORITHMS AND COMPARISON

**Multiple Constant Multiplication:**

Multiple Constant Multiplication- is a process in which one input variable is multiplied with number of constant values known as coefficients. This is a common process in convolution, and transforms techniques and digital signal processing. An input x (n) is multiplied with set of coefficients ($C_1$, $C_2$, $C_3$,……..$C_n$).

In such kind of multiplication process direct multiplication of individual terms i.e. $x(n) \times C_1$, $x(n) \times C_2$, $x(n)\times C_3$,……..$x(n) \times C_n$, will result in lot of power consumption and area consumption on chip. Due to large number of logical operations the speed of the chip will also be poor. The flexibility of digital logics is applied to reduce these difficulties. Several low complexity architectures have been used for the optimization of Multiple Constant Multiplication (MCM), i.e., the multiplication of a data sample by set of constants, for further improvement in area and power consumption. In this chapter the existing schemes for the constant multiplication namely 2-bit, 3-bit binary common sub-expression algorithms are discussed in contrast to the entitled VHBCSE algorithm.

In order to understand VHBCSE algorithm first we inevitably needed have the knowledge of fixed bit BCSE algorithms. For this purpose we discussed them in the sections below.

**Introduction to fixed bit BCSE algorithms:**

Fixed bit algorithms are applied either vertically or horizontally in only one layer of the architecture. There after the partial products will get added in addition layers. Eg:2-bit, 3-bit BCSE algorithms

**2-bit and 3-bit BCSE algorithms:**

Considering the coefficients in binary pattern, the FIX fixed bit BCSE (FBCSE) algorithms mentioned in chapter 2 in attempt to eliminate the redundant computation vertically by considering 3-bit or 2-bit BCS present across the adjacent coefficients. As defined and explained in and, horizontal BCSE algorithm utilizes CSs occurring within each coefficient to get rid of redundant computations, while vertical BCSE uses CSs found across adjacent coefficients to eliminate redundant computations. According to BCSE algorithm a total of $2n - (n+1)$ BCS’s can be formed out of an n-bit binary number and the number of adders required to generate the partial products for n-bit BCS is $2n-1$. There are some measures for the hardware cost considered while designing any multiplier architecture. Those are **adder cost** and **adder step**.

In a reconfigurable constant multiplier, the coefficient values can be dynamically programmable. Therefore, the idea behind the reconfigurable multiplier is to consider the worst case (which involves the largest number of addition steps) whereby all the relatively better cases will also be taken care of. Hence, considering a reconfigurable multiplier having 16-bit input (X) and the 16-bit coefficient (H), the worst case condition will occur for the coefficient of values 16'HFFFF.
IV. VHBCSE ALGORITHM

Concept of VHBCSE:
The algorithm vertical-horizontal binary common sub-expression elimination applies 2-bit BCSE algorithm vertically and then 4-bit and 8-bit BCSE algorithms horizontally so that maximum number of duplicates can be eliminated. Thus this technique utilizes the redundancy present both horizontally and vertically. Constant multiplier architecture based on vertical-horizontal binary common sub-expression elimination (VHBCSE) algorithm for designing are configurable finite impulse response (FIR) filter whose coefficients can dynamically change in real time.

To design an efficient reconfigurable FIR filter, according to the VHBCSE algorithm, 2-bit binary common sub-expression elimination (BCSE) algorithm has been applied vertically across adjacent coefficients on the 2-D space of the coefficient matrix initially, followed by applying variable-bit BCSE algorithm horizontally within each coefficient. This technique is capable of reducing the average probability of use of the switching activity of the multiplier block adders by 6.2% and 19.6% as compared to that of two existing 2-bit and 3-bit BCSE algorithms respectively. ASIC implementation results of FIR filters using this multipliers how that the proposed VHBCSE algorithm is also successful in reducing the average power consumption by 32% and 52% along with an improvement in the area power product (APP) by 25% and 66% compared to those of the 2-bit and 3-bit BCSE algorithms respectively.

VHBCSE algorithm procedure:
The algorithm presented in this report solves problems concerned with other fixed bit algorithms. It consists of the following steps:
1) At first, the filter coefficient has been multiplexed between its original and complemented values depending on the most significant bit (MSB) of the coefficient to support the signed decimal data representation. This technique helps in reducing the hardware complexity when the coefficients consist of small negative decimal numbers.
2) According to the proposed algorithm in layer-1 of MAT, the 2-bit BCSE has been applied vertically followed by conditional 4-bit and 8-bit BCSEs horizontally in layer-2 and layer-3 respectively to find out the common sub-expressions (CSs) present within the coefficients. This technique helps in solving the additional hardware consumption problem by eliminating more CSs.
3) Extending the BCSE in the lower level or applying the BCSE horizontally will reduce the probability of use of the MB adders present in the lower levels of the MAT. This will reduce the power consumption to a great extent by lowering the switching activities of these MB adders.
4) Application of different lengths of 2-bit, 4-bit, and 8-bit BCSE at different layers of MAT will produce the area optimized constant multiplier during execution of the high level synthesis procedure.
5) This technique can solve the problem of high power and area consumption problem for both the cases, viz. small valued negative coefficient and high valued positive coefficient. The proposed design also supports the data representation in signed decimal format. Apart from that, area and power efficient reconfigurable FIR filter can be obtained during the high level synthesis procedure using constant multiplier based on proposed VHBCSE algorithm. All of these observations make the proposed constant multiplier an excellent candidate for designing any (higher or lower) order efficient reconfigurable FIR filter.

Architectural flow description
The architecture flow diagram of Vertical-horizontal binary common sub-expression based constant multiplier is shown in the figure. The architecture consists of 8 building blocks.

Application VHBCSE algorithm is as follows:
In three layers namely, layer-1, layer-2 and layer-3 BCSE algorithm is applied according to the VHBCSE scheme.

Application of 2-bit VBCSE:
The algorithm applied vertically in the layer-1. The layer-1 consists of partial product generator and multiplexer unit. The partial products generated according to 2-bit BCSE algorithm can be used for other coefficients also, whenever same pattern appear among the coefficients. This is nothing but vertical redundancy elimination. The figure shows 2-D space of coefficients consisting of four coefficients H0, H1, H2 and H3 each of
16-bit wide. The table represented in binary form clearly depicts the scheme we are following.

**Application of 4-bit HBCSE:**

The dashed-line boxes covering each nibble of a coefficient compares for similar frames anywhere within the coefficient. If match is found, in the following controlled addition layer the sum for the match is not generated and instead shifted form of its match is used as the corresponding partial product.

![Coefficient table for VHBCSE algorithm](image)

**Fig:** Coefficient table for VHBCSE algorithm

The bit pattern 1111 is present four times in H3, literally H3 [15:12], H [11:8], H [7:4] and H [3:0]. Since the same pattern is found in the four nibbles of H3 the control signals for controlled addition at layer-2 will be generated such that partial product corresponding to H [15:12] will be shifted and used as partial products for other nibbles. Shifting H [15:12] by 4 bits right will produce a partial product corresponding to H [11:8]. Partial products corresponding to H [7:4] and H [3:0] are also produced in similar manner. This avoids the regeneration of unwanted partial products i.e. duplicates.

**Application of 8-bit HBCSE algorithm:**

In the next stage partial products corresponding to 4 nibbles of coefficient will be transferred to layer 3 where controlled addition is performed according to 8-bit HBCSE algorithm. First the four partial products from controlled addition at layer 2 are summed up to produce partial products corresponding to upper and lower bytes. The control signal c7 performs controlled addition at this layer 3. If c7 is „1‟ means there is a match of upper with lower byte of the multiplexed coefficient (Hm). Then the partial product of upper byte is shifted right by 8 bits and used as partial product of lower byte. If c7 is „0‟, there is no match of upper byte with lower byte resulting in separate partial product generation. In the figure, on 2-D coefficient space H1, H2 and H3 have their upper and lower bytes as same. Thus 8-bit horizontal BCSE algorithm, nevertheless, will make use of this redundancy that is instead of using its respective adder output it reuses existing term of upper byte

**Sign conversion for final result:**

The 16-bit product obtained from final addition layer will be 2‟s complemented and multiplexed between its original and complemented versions in this block. The sign information of inputs X and H will be passed to this sign restore block. The final sign of the result is used as the selection line for this 2x1 multiplexer. If both inputs are negative or positive, the final result has to be positive. Otherwise the sign of result will be negative. An ex-or operation between signs of inputs will produces the sign for final result.

Unlike fixed bit algorithms such as 2-bit or 3-bit algorithms, VHBCSE algorithm supports signed decimal format for both the inputs. Whereas the fixed bit algorithms consider signed magnitude format for input which is not common in real time verification of design operation. The simulation result obtained from Modelsim software tool is given in the figure. The results are compared with theoretical computation as described in the following lines simulation result of Aimed constant multiplier.

**Theoretical calculation**

Input Xin = +269
Constant H = -28444/ 216 = -0.43402099

The simulator assumes direct 2”s complement version of H. It cannot assume floating value hence the division operation is to obtain the actual value of the coefficient. Division by 216 shifts the coefficient by 16 bits so that the actual floating value is obtained

Thus,

Xin * H = (269) * (-0.43402099) = -116.75164794921875

Since the constant multiplier involves floating point operation using fixed point algorithms, small truncation error takes place in the results obtained. In the above example the error is just the fractional part.
Performance metrics:

The major concerns in a constant multiplier are power, delay and area. In this thesis we have simulated and synthesized the targeted VHBCSE based constant multiplier as well as the 2-bit BCSE algorithm based constant multiplier. The results are furnished in the table given below. For the purpose of comparison we have designed both the architectures for the same FPGA, Spartan 3E-3s500, FG320 package. The results show that the delay is reduced with the VHBCSE architecture compared to the existing and efficient architecture, the 2-bit BCSE algorithm.

<table>
<thead>
<tr>
<th></th>
<th>VHBCSE based architecture</th>
<th>2-bit based BCSE architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay (maximum)</td>
<td>27.59 ms</td>
<td>28.59 ms</td>
</tr>
<tr>
<td>Power consumption @500MHz</td>
<td>0.126W</td>
<td>0.129W</td>
</tr>
<tr>
<td>Power consumption @1000MHz</td>
<td>0.171W</td>
<td>0.177W</td>
</tr>
<tr>
<td>Number of LUT* used</td>
<td>334</td>
<td>298</td>
</tr>
</tbody>
</table>

Table: Comparison of results of VHBCSE and 2-bit BCSE

Timing reports:

<table>
<thead>
<tr>
<th>Timing report</th>
<th>Delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT1:2 (1-0)</td>
<td>0.704</td>
</tr>
<tr>
<td>LUT2:1 (2-0)</td>
<td>0.706</td>
</tr>
<tr>
<td>LUT3:1 (3-0)</td>
<td>0.446</td>
</tr>
<tr>
<td>LUT4:1 (4-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT5:1 (5-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT6:1 (6-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT7:1 (7-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT8:1 (8-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT9:1 (9-0)</td>
<td>0.439</td>
</tr>
<tr>
<td>LUT10:1 (10-0)</td>
<td>0.439</td>
</tr>
</tbody>
</table>

V. CONCLUSION

The presented VHBCSE algorithm is proved to be efficient in terms of delay and power consumption. The Vertical-Horizontal BCSE algorithm removes the initial common sub-expressions by applying 2-bit BCSE vertically. Further elimination of the BCSE’s has been performed through finding the common sub-expressions present within the coefficients by applying BCSEs of different lengths horizontally to different layers of the shift and add architecture.

The results of delay and power can be improved by extending the number of coefficients. Since only horizontal elimination is possible with the single coefficient multiplier the performance metrics are of minor improvement. When designing an FIR filter 2-bit vertical elimination also comes into picture. Thus reduction of hardware cost, delay and power will be improved considerably.

REFERENCES