INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 4.295
(Volume 4, Issue 3)
Available online at: www.ijariit.com

A regular and improved representation for signed digit constant vector multiplication

M. Revathy
revathymuthuvelu20@gmail.com
College of Engineering, Chennai, Tamil Nadu

S. Lavanya
lavanyasanthosham14@gmail.com
College of Engineering, Chennai, Tamil Nadu

ABSTRACT

A novel improved signed digit representation procedure is proposed to overcome the two fundamental disadvantages of the current multiplier-free techniques: 1) circuit inconsistency and 2) computational redundancy. The fundamental difference between the existing multiplier free strategies and the proposed method could be a novel optimization framework based on vector decomposition. The constant vector is decomposed into two terms: a "private" matrix and a "public" vector which consist of the private operations of each individual entry and public operations shared by all of the entries, the overall data flow can be separated into two regular steps: first multiplied by the "public" vector and then by the "private" matrix. The computational complexity lessening task is then accomplished by minimizing the number of operations within the "private" matrix and the length of the "public" vector. Experimental results illustrates that the proposed strategy outflanks the existing multiplier free strategies in less operations and more regular circuit structure.

Keywords— *Canonical-Signed-Digit (CSD), Multiplier-free, Vector multiplication, Improved Signed Digit (ISD)*

1. INTRODUCTION

Multiplication is a very important operation for digital computing systems, process controllers, and signal processors. The CSD[11] implementation of each entry by enumerating all possible circuit implementations to find the one with minimum adders, and propose a greedy strategy to reduce computational redundancy among different entries uses the group method and bounds several adjacent bits together in the representation step. However, this method is proposed for the reconfigurable FIR filters, thus the redundancy reduction performance is not satisfying. These techniques above suffer from a low dimensional search space problem. Since the representation of each entry is fixed before the redundancy reduction, the search space for the optimum implementation has to be restricted on the corresponding representation results.

By shifting the data bus the left, the multiplication of power-of-two can be accomplished proficiently and with the assistance of the binary complement code, the subtraction can be accomplished by adders. By including and subtracting arrangement of power-of-two increase comes about, the steady increase in equipment are finished. There is a noteworthy computational excess in the CSD based methodologies. Since each section is changed over into movements and increases freely, a similar transitional operators must be propelled commonly. Numerous rich strategies have been proposed to diminish the computational excess of CSD [1]– [6].

Reference [1] and [2] propose a graph- synthesis – based procedure to reuse the repetitive operation of each section. Dempster and Macleod make strides the CSD usage of each passage by identifying all of the conceivable circuit executions to discover the one with least adders [3] and propose a covetous procedure to diminish the computational repetition among distinctive sections [4].

Reference [5] utilizes the group method and limits a few nearby bits together in the portrayal step. Nonetheless, this technique is proposed for the reconfigurable FIR filters, therefore, the redundancy reduction performance isn't fuling.

These previously mentioned strategies endure a low dimensional search space issues. Since the representation of each entry is settled before the

redundancy reduction, the search space for the ideal usage has got to be limited on the comparing representation comes about. Reference [6] proposes the Minimal Signed Digit (MSD).rather than a settled representation of each section, MSD records a few limited representation arrangements of each section and choose the ideal one by a joint optimization. In spite of a common and flexible framework, the increment of search space includes the combinatory blast issue in MSD.

Reference [7] employments the greedy strategy to illuminate the optimization issues in MSD. Reference [8] extends the graph-synthesis-based method to the MSD framework. In spite of the fact that all of the existing procedures can diminish the computational redundancy essentially, the normality of the circuit may be yielded. For case, within the greedy-based strategies, any vertices of the data flow can be utilized as an intermediate result for the computation of other vertices. The variant add-path for distinctive entries influence the routability of the circuit and make the organized format fashion recalcitrant.

In this brief, we propose a Improved Signed Digit (ISD) representation approach for the constant vector multiplication. The basic difference between the proposed strategy and the existing multiplier-free procedure is that the circuit consistency is considered jointly with the redundancy reduction process. We propose a novel optimization framework based on the vector decay. To be particular, within the framework, the constant vector is deteriorated into two terms: a "public" vector and a "private" matrix. In this way, the multiplication of the input signal and the consistent vector is accomplished by a successive multiplication of the "public" vector and "private" matrix. The "public" vector creates a set of normal intermediate vertices of the data flow which can be reused for each section. The in general computational excess can be diminished by minimizing the length of the "public" vector and the number of operations of the "private" matrix. At that point, we propose a greedy strategy to fathom solve the vector decomposition problem.

2. MATHEMATICAL MODEL AND SOLVE METHOD FOR THE PROPOSED APPROACH

The proposed vector decomposition procedure benefits from a modern representation methodology which employments $2^i \pm 1$, i $\in Z$,rather than power of 2, as the basic components to represent the entries of the constant vector.

$$n = \sum_{m=0}^{\infty} a_m \times 2^{p_m} \times (2^{q_m} \pm 1), \forall n \in Z, \quad \exists p_m \in Z, q_m \in Z, a_m \in \{0,1\} \qquad (1)$$

Equation (1) can be proved as follows.

Proof: As any integer can be represented in the binary form, we can get

$$n = \sum_{m=0}^{\infty} a_m \times 2^m, \forall n \in Z, \exists a_m \in \{0,1\} \qquad (2)$$

Based on (2), we can get

$$n = \sum_{m=0}^{\infty} a_m \times 2^m \times (2^1 - 1), \forall n \in Z, \quad \exists a_m \in \{0,1\} \qquad (3)$$

$$n = \sum_{m=0}^{\infty} a_m \times 2^{m-1} \times (2^0 + 1), \forall n \in Z, \quad \exists a_m \in \{0,1\} \qquad (4)$$

If $p_m = m$ and $q_m = 1$, (3)can be changed into

$$n = \sum_{m=0}^{\infty} a_m \times 2^{p_m} \times (2^{q_m} - 1), \forall n \in Z, \quad \exists a_m \in \{0,1\}, p_m = m, q_m = 1. \qquad (5)$$

In this manner, any number can be represented by the summation of $2^i - 1$,i ∈Z and their shifts.

$2^{p_m} \times (2^{q_m} - 1), p_m \in Z, q_m \in Z$ can be respected as asset of "basics", which can represent all of the integers.

If p_m = m-1 and q_m = 0, (4) can be changed into

$$n = \sum_{m=0}^{\infty} a_m \times 2^{p_m} \times (2^{q_m} + 1), \forall n \in Z, \quad \exists a_m \in \{0,1\}, p_m = m - 1, q_m = 0. \qquad (6)$$

With (5) and (6), Lemma (1) is demonstrated, which illustrates that 2^i+1, i∈ Zcan represent any integer and it can be regarded as a more adaptable and over total "basis" to represent the integers.

Considering that the shifters can be supplanted by filling the least noteworthy bits with 0, each $2^i \pm 1$, i∈ Z can be accomplished by one adder. We utilize $2^i \pm 1$, i∈ Z as a set of customary and open middle operations for all of the sections, and the shifts of $2^i \pm 1$, i∈ Z change as the private operations for each individual section.

Scientifically, the assignment to speak to all of the passages by $2^i \pm 1$, i∈ Z and their shifts is to represent the constant vector by a multiplication of a "private" matrix (PM) and a "public" vector (PV) which comprise of the private and public sub operators, separately. $2^{p_m} \times (2^{q_m} \pm 1), p_m \in Z, q_m \in Z$ can be respected as an over complete basis to represent the sections, which increment the solution space of the decomposition, and gives us adaptability to break down the vector in a more compelling way.

a is defined as the constant vector, which is a column vector.

x is the input. Therefore, we can get

$$a \times [x] = PM \times PV \times [x] \qquad (7)$$

The number of additions in the whole operation is decided by the length of PV and the number of

nonzero coefficients in PM. If the length(PV) = M , size(PM) = M x N, and the number of nonzero coefficients in every line of PM is mn, n ∈ Z, 1 ≤ n ≤ N . The number of additions within the whole operation can be gotten in

$$num_{additions} = M + \sum_{n=1}^{N}(m_n - 1) \quad (8)$$

We need to find the minimum of (8), in order to decrease the computational complexity in the implementation. The mathematical model of ISD representation approach is,

{PM,PV} = arg min $|num_{additions}|$ = arg min M $\sum_{n=1}^{N}(m_n - 1)$ (9)

Input: The constant multiplication vector a.
Output: The selected PV and PM, which can decompose a.

STEP 1: Initialization:
Set V = [], sign_flag = 0, loop_flag = 1, n = 1, S = , temp_numadd= inf, temp_PV=[], temp_PM=[];
STEP 2: Get the least public suboperators:
while (loop_flag == 1) do if (sign_flag == 0) do
add the entry of 2n 1 into V−
sign_flag = 1
else do
add the entry of 2n +1 into V
sign_flag = 0
n = n +1
end if
for 0 < llength(á) do
if al can be represented by the entries in V and their shifts do
continue else do
break end if
end for
if l == length(a) do
loop_flag = 0
end if end while
STEP 3: Select the combination of PV and PM which uses the least additions:
for 0 < m ≤ length(V) do
choose m entries from V and add all of the possible situations into S
end for
for s in S do
for 0 < llength(á) do
if al can be represented by the entries in s and their shifts do
continue else do
break end if
end for
if l == length(a) do
represent a by a multiplication of a matrix M and a vector V, and get the number of additions in the whole operation numadd according to (8),
if temp_numadd > numadd do
assign temp_PM as M, and temp_PV as V.
end if end if end for
PM = temp_PM, PV = temp_PV
return PV and PM

With the selected PV and PM, the implementation can be divided into two regular steps. The first step is to acquire the results of PV[x], and it can be achieved by an adder array easily, which is a regular architecture as shown in Fig. 1.

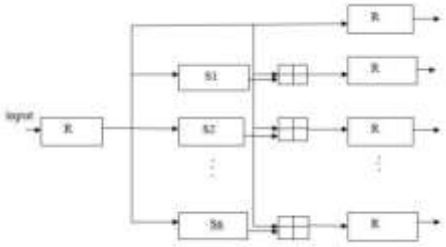

Fig. 1: Architecture of adder array

The second step is to achieve the final results by PM x (PV x [X]), and it can be finished by a few binary adder trees as appeared in Fig 2. In Figs.1 and 2, able to discover that the greatest delay between each two registers is restricted to as it were one adder. The distribution of adders within the usage is balanced, which comes about in a customary circuit and moves forward the routability. Another advantage of the ISD representation approach is that, in the scenarios with low framework recurrence necessity, several stages can be consolidated into one organize, which decreases the hardware assets of registers.

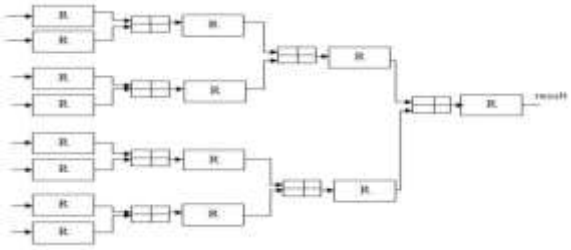

Fig. 2: Architecture of a binary adder tree3.

3. PERFORMANCE EVALUATION OF THE PROPOSED APPROACH

We use three experiments to evaluate the performance of the proposed ISD technique: a Gaussian filter, two reported filters, and random filters with variant orders.
Experiment I: We choose a Gaussian filter with 11 orders as an example to show the circuit design procedure with the proposed ISD technique.

The filter coefficients are shown in:

[14 19 57 108 134 108 57 19 41] (10)

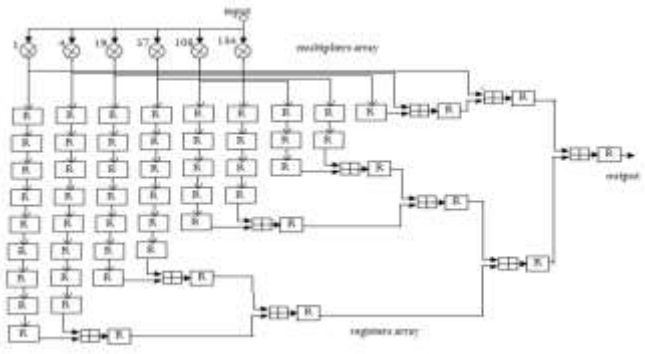

Fig. 3: Gaussian filter implementation

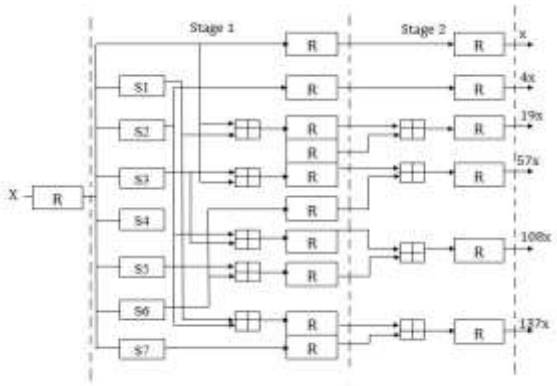

Fig. 4: Multiplier block implementation in CSD form.

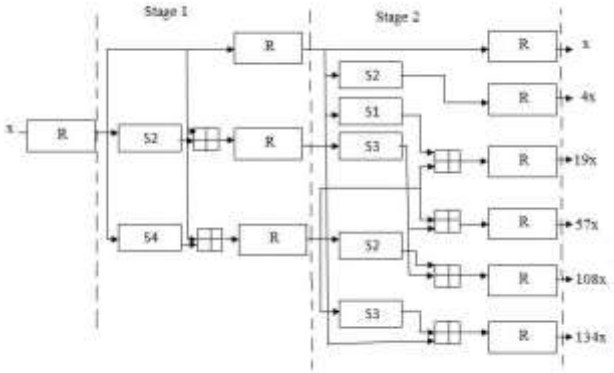

Fig. 5: Multiplier block implementation with the ISD technique.

In Fig. 3, this execution comprises of two parts: the multiplier piece and the register array. The multiplier block finishes the multiplication of each input and the filter coefficients. The register array gets the filtering results by including the multiplication comes about after different delays. It is appreciated that the most complex operation in this execution is to realize the multiplications of the inputs and six filter coefficients. In arrange to include the CSD form multiplier-free execution, the filter coefficients are represented as follows:

$$\begin{bmatrix}1\\4\\19\\57\\108\\134\end{bmatrix} = \begin{bmatrix} 2^0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2^0 & 2^0 & 0 & 0 & 2^0 & 0 & 0 & 0 \\ 2^0 & 0 & 0 & -2^0 & 0 & 0 & 2^0 & 0 \\ 0 & 0 & 2^0 & 2^0 & 0 & 2^0 & 2^0 & 0 \\ 0 & 2^0 & 2^0 & 0 & 0 & 0 & 0 & 2^0 \end{bmatrix} x \begin{bmatrix}2^0\\2^1\\2^2\\2^3\\2^4\\2^5\\2^6\\2^7\end{bmatrix} \quad (11)$$

In fig 4, the implementation of multiplier block requires nine adders, and maximum delay between every two registers is one adder.With the proposed ISD approach discussed in Section II, we can get (12) and the implementation in Fig. 5

$$\begin{bmatrix}1\\4\\19\\57\\108\\134\end{bmatrix} = \begin{bmatrix} 2^0 & 0 & 0 \\ 2^2 & 0 & 0 \\ 2^1 & 0 & 2^0 \\ 0 & 2^3 & 2^0 \\ 0 & 2^3 & 2^2 \\ -2^1 & 0 & 2^3 \end{bmatrix} \times \begin{bmatrix}2^0\\2^2+2^0\\2^4+2^0\end{bmatrix} \quad (12)$$

In Fig. 5, there are two stages in the multiplier block implementation. The first stage accomplishes the same sub operators, 2^0x, (2^0+2^2)x, and (2^0+2^4)x. Within the second stage, the intermediate of stage 1 are shifted and added with several binary adder trees. The shift bits within the second stage are chosen by the coefficient of PM in (12). It is obvious that the maximum delay between two registers in this implementation is additionally one adder. However, the number of adders is diminished to 6. The detailed numbers of operations within the two sorts of implementations are recorded in table I.

Table 1: Comparison of operations

Implementation	CSD form	Proposed ISD
Number of adders	9	6
Number of Registers	17	10
Maximum delay between every two registers	One adder	One adder

From Table I, we are able to discover that the proposed implementation uses less adders and registers than the usage in CSD form. It employment six adders to realize the multiplication of inputs and six filter coefficients, which is approximately 66% of the CSD form.

Experiment II: The proposed ISD method is utilized in the implementations of the two filters which are received in [26]. The comparison of operations within the proposed ISD technique and the existing procedures is recorded in Table II.
It can be watched that the computational excess among different channel coefficients is decreased by the proposed ISD procedure. Besides, among these existing procedures, the proposed ISD procedure needs the least adders.

Table 2: Comparison of operations for two filters

Filter	ALGORITHM	Adders
9 bits quantization 25 orders	[17]	23
	[27]	19
	[19]	21
	[26]	18
	Proposed ISD	12
14 bits quantization 60 order	[11]	114
	[27]	61
	[19]	85
	[26]	75
	Proposed ISD	60

Experiment III: The exhibition of CSD and ISD are assessed on random filters with 116 bit quantization and variation order (from 116 to 255). The number of adders in CSD and ISD are checked, and their proportion is plotted in Fig.6. It can be observed that the ISD method saves around 35% adders of CSD, and the gain increases with the increment of filter orders.

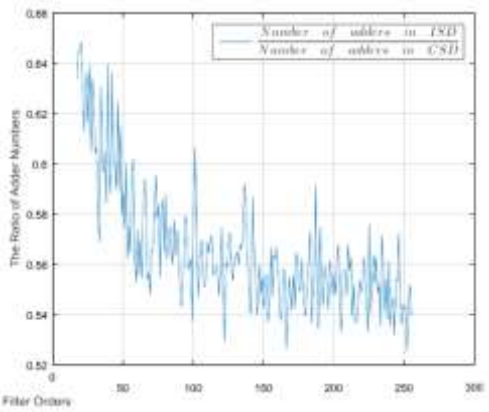

Fig. 6: Ratio of operations in ISD and CSD

4. CONCLUSION

In this brief, we have proposed an ISD representation approach for constant vector multiplication. Within the proposed ISD technique, the constant vector is deteriorated into two terms: a "private" matrix and a "public" vector. The "private" matrix contains the private sub operators for each section, the "public" vector contains the same sub operators for all of the entries. Based on the deterioration comes about, the "public" vector generates a set of regular intermediate vertices of the data flow which can be reused for each section. Hence, the circuit regularity can be considered jointly with the redundancy reduction handle within the proposed ISD strategy, which can improve the routability within the execution and make it more suitable of the circuit plan than the existing strategies. In addition, we have also proposed a greedy strategy to get the decomposition results in a least complexity way.

5. REFERENCES

[1] R. I. Hartley, "Optimization of canonic signed digit multipliers for filter design," in Proc. IEEE Int. Symp. Circuits Syst., 1991, pp. 1992–1995.

[2] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 43, no. 10, pp. 677–688, Oct. 1996.

[3] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," Proc. Inst. Elect. Eng.—Circuits Devices Syst., vol. 141, no. 5, pp. 407–413, Oct. 1994.

[4] G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 42, no. 9, pp. 569–577, Sep. 1995.

[5] Hatai, I. Chakrabarti, and S. Banerjee, "An efficient constant mul- tiplier architecture based on vertical–horizontal binary common sub- expression elimination algorithm for reconfigurable FIR filter synthesis," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 62, no. 4, pp. 1071–1080, Apr. 2015.

[6] C. Park and H. J. Kang, "Digital filter synthesis based on an algo- rithm to generate all minimal signed digit representations," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 21, no. 12, pp. 1525–1529, Dec. 2002.

[7] G. Dempster and M. D. Macleod, "Using all signed-digit represen- tations to design single integer multipliers using subexpression elimina- tion," in Proc. Int. Symp. ISCAS, 2004, vol. 3, pp. III-165–III-168.

[8] Chen and C. H. Chang, "High-level synthesis algorithm for the design of reconfigurable constant multiplier," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 28, no. 12, pp. 1844–1856, Dec. 2009.

[9] D. R. Bull and D. H. Horrocks, "Primitive operator digital filter synthesis using a shift biased algorithm," in Proc. IEEE Int. Symp. Circuits Syst., 1988, vol. 2, pp. 1529-1532.

