A Novel DA-based Architecture for Efficient Computation of Inner-Product of Variable Vectors

Pramod Kumar Meher  
School of Computer Engineering  
Nanyang Technological University  
50 Nanyang Avenue, Singapore - 639798  
Email: aspkmeher@ntu.edu.sg

Sang Yoon Park  
Engineering Department  
Institute for Infocomm Research  
1 Fusionopolis Way, Singapore - 138632  
Email: sypark@i2r.a-star.edu.sg

Abstract—Distributed arithmetic (DA) has been widely used for area-time efficient implementation of inner-products, where one of the vectors is fixed and known a priori. Computation of inner-product of a pair of variable vectors, however, is required very often for matrix-multiplication of different forms, and implementation of digital filters of unknown coefficients and variable lengths. But the possibility of using DA for the computation of inner-product of variable vectors is yet to be explored. In this paper, we analyse the design issues relating to DA-based implementation of inner-product of variable vectors, and derive a novel area-time efficient flexible solution for the bit-parallel DA-based implementation of inner-product of variable vectors and variable inner-product lengths. It is found that the proposed structures are nearly 34% faster than the conventional multiplier-based implementation in average for different inner-product lengths \((N = 8, 16, 32 \text{ and } 64)\) and for input word-lengths, \(L = 8\) and \(L = 16\). Moreover, proposed designs offer saving of nearly 22% and 36% area-delay product (ADP) and saving of nearly 16% and 24% power delay product (PDP) over the multiplier-based designs for \(L = 8\) and \(L = 16\), respectively, in average, for various inner-product lengths.

I. INTRODUCTION

Inner-product can be defined for any two vectors \(A\) and \(B\) of equal-length, and given by

\[
C = \sum_{k=0}^{N-1} A_k \cdot B_k
\]

where \(N\) is the length of the vectors. Computation of inner-products is frequently encountered in several engineering applications. It is a basic task for computing matrix-vector product and matrix-matrix product. It is also the core computation to realize several important functions such as circular and linear convolutions, discrete transforms, finite impulse response (FIR) and infinite impulse response (IIR) digital filters for digital signal processing (DSP) applications [1].

Distributed arithmetic (DA) has been widely used for area-time efficient implementation of inner-products, where one of the vectors is fixed and known a priori [2]. For DA-based implementation of such inner-products, the sum of all possible combinations of the elements of the known vector are stored in a ROM-based look-up table (LUT) and bit-slices of other vector are used as address to read the corresponding partial inner-product from the LUT. The partial inner-products corresponding to all the bit-slices of the unknown vector are read from the LUT and shift-added for the computation of inner-product. In many DSP applications the impulse response vector (or the coefficient vector) is fixed. Therefore, DA has been popularly used for efficient implementation of those filters [2]. The memory requirement of DA-based implementation of inner-product increases exponentially with the inner-product length. To get rid of such large memory requirement, systolic decomposition techniques have been suggested by Meher and used for DA-based implementation of long-length convolutions and FIR filter of large orders in [3] and [4]. Conventionally, DA was used to be bit-serial, and accordingly it was slow. To overcome this issue, bit-parallel systolic DA has been proposed, and further optimizations for high-throughput DA-based implementation of FIR filter have been proposed in [3]–[5]. We also find quite a few works on DA-based implementation of adaptive filters [6]–[8].

Till now, the DA-based approach is used for the computation of inner-products where one of the vectors is fixed and known a priori. Inner-product (of variable vectors in general) is a core computation for matrix multiplication problems including the implementation of digital filters of unknown coefficients and variable lengths. However, the scope, overhead and design perspectives relating to the use of DA principle for computing the inner-product of variable vectors are yet to be explored. By variable vector, we mean here the vector whose components as well as the length are not known a priori, and not restricted to remain constant. In this paper, we have analysed the design issues of DA-based computation of inner-product of variable vectors, and proposed a novel architecture for high-performance power-delay efficient solution.

In the next section, we discuss the key principle of DA-based computation, followed by the design issues for implementing inner-product of variable vectors using DA. The proposed design is described in Section-III. We compare the complexities of proposed design with that of conventional multiplier-based design in Section IV. It is shown from synthesis results pertaining to various inner-product lengths that the proposed scheme provides significant saving of computation-time, area-delay product and power-delay product in the implementation of inner-product of a pair of variable vectors over the conventional multiplier-based structure.
II. DESIGN STRATEGY FOR DA-BASED COMPUTATION OF INNER-PRODUCT OF VARIABLE VECTORS

We briefly outline here the conventional distributed arithmetic approach for inner-product computation and necessary DA-decomposition for memory-efficient low-complexity implementation of inner-product of variable vectors.

A. Conventional DA Approach for Computation of Inner-Product of a Unknown Vector with a Known Constant Vector

Assuming \( L \) to be the word-length, each component of vector \( \mathbf{B} \) may be expressed in 2’s complement representation:

\[
B_k = -b_{k0} + \sum_{l=1}^{L-1} b_{kl} \cdot 2^{-l}
\]

where \( b_{kl} \) denotes the \( l \)-th bit of \( B_k \), the \( k \)-th component of vector \( \mathbf{B} \).

Substituting (2) on (1), we can express the inner-product of \( (1) \) in an expanded form:

\[
C = -\sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{k=0}^{N-1} A_k \cdot \left[ \sum_{l=1}^{L-1} b_{kl} \cdot 2^{-l} \right]
\]

To convert the conventional sum-of-products form of inner-product of \( (1) \) into a distributed form, the order of summations over the indices \( k \) and \( l \) in the second term of (3) can be interchanged to have

\[
C = -\sum_{k=0}^{N-1} A_k \cdot b_{k0} + \sum_{l=1}^{L-1} 2^{-l} \cdot \left[ \sum_{k=0}^{N-1} A_k \cdot b_{kl} \right]
\]

(4)

We can also merge the pair of terms in \( (4) \) to have a unified form:

\[
C = \sum_{l=0}^{L-1} 2^{-l} \cdot r(l) \cdot C_l
\]

(5a)

where the partial inner-product

\[
C_l = \sum_{k=0}^{N-1} A_k \cdot b_{kl}
\]

and \( r(l) = -1 \) for \( l = 0 \) and \( r(l) = 1 \) for \( l > 0 \).

Since each bit of the \( N \)-point bit-slice \( \{b_{kl} \} \) for \( 0 \leq k \leq N-1 \} \) can either be zero or one, any of the partial inner-products \( C_l \) for \( l = 0, 1, \cdots, L-1 \) can have \( 2^N \) possible values. If vector \( \mathbf{A} \) is assumed to be constant, all the \( 2^N \) possible values of \( C_l \) can be pre-computed and stored in a ROM, such that while computing the inner-product, the partial inner-products \( C_l \) for \( l = 0, 1, \cdots, L-1 \) can be read out from the ROM using the bit-slice \( \{b_{kl} \} \) as address. The inner-product can, therefore, be calculated according to (5), by \( L \) cycles of ROM-read operations followed by shift-accumulation corresponding to \( L \) bit-slices \( \{b_{kl} \} \) for \( 0 \leq l \leq L-1 \).

B. Design Issues for the DA-based Computation of Inner-Product of Variable Vectors

The conventional DA approach is not suitable for the DA-based computation of inner-product of variable vectors due to the following two main design issues.

- When both the vectors are variable, we need to use RAM-based LUT instead of ROM-based LUT for DA-based computation. For computing \( N \)-point inner-product a RAM of \( 2^N \) words need to be used. Since RAM is costlier in terms of area and power consumption compared to the ROM, inner-products of very short lengths could only be supported in case of variable vector case.

- For every pair of new input vectors, the RAM content is to be re-evaluated which will require \( (2^N - N - 1) \) additions. To perform all these additions in parallel the adder-complexity for LUT generation could be prohibitively high for large values of \( N \).

To prevent very high area cost of DA-unit we need to limit the length of inner-product to small values. The systolic DA-based designs suggested in [3] and [4] for large order digital convolution and FIR filters, respectively, require large number of RAM LUTs and large number of adders to populate the RAM LUTs. In the following subsection, therefore, we discuss the DA-decomposition scheme for the computation of inner-product of variable vectors of larger lengths.

C. DA-Formulation for Computation of Decomposed Inner-Product of Variable Vectors

Let the inner-product length \( N \) be a composite number given by \( N = PM \), (\( P \) and \( M \) may be any two positive integers.) one can map the index \( k \) into \((m + pM)\) for \( m = 0, 1, \cdots, M-1 \) and \( p = 0, 1, \cdots, P-1 \) so as to express (4) in the form:

\[
C = \sum_{l=0}^{L-1} 2^{-l} \cdot r(l) \cdot \sum_{p=0}^{P-1} S_{l,p}
\]

(6a)

where

\[
S_{l,p} = \sum_{m=0}^{M-1} A_{(m + pM)} \cdot b_{(m + pM)l}
\]

(6b)

for \( l = 0, 1, \cdots, L-1 \) and \( p = 0, 1, \cdots, P-1 \).

For any given pair of vectors the \( 2^M \) possible values of \( S_{l,p} \) corresponding to the \( 2^M \) permutations of \( M \)-point bit-sequence \( b_{(m + pM)l} \), for \( m = 0, 1, \cdots, M-1 \) for \( l = 0, 1, \cdots, L-1 \) may be stored in an LUT of \( 2^M \) words. These values of \( S_{l,p} \) can be read out when the bit-sequence is fed to the RAM LUT as address. \((2^M - M - 1) \) additions need to be performed to populate the RAM LUT. To minimize the number of adders to generate the LUT content we have taken \( M = 2 \), such that only one addition is required to generate the 4 words to be stored in the RAM LUT.

III. DERIVATION OF PROPOSED DA-BASED STRUCTURE FOR INNER-PRODUCT OF VARIABLE VECTORS

The heterogeneous dependence graph (DG) based on (6) for the computation of any two \( N \)-point vectors is shown in Fig.1.
It consists of three types of nodes. The functions of nodes $R$, $T$ and $F$ are shown in Fig. 1 (b), (c) and (d), respectively. Node $R$ performs the LUT read operation, while node $T$ and node $F$, respectively, perform the shift-add operation and final summation of (6). The $R$ and $T$ nodes of the DG can be projected vertically along $l$ direction to derive a systolic-like DA-based bit-serial pipeline structure (shown in Fig. 2). Each DA-unit in Fig. 2 performs an LUT read and a systolic-like DA-based bit-serial pipeline structure (shown in Fig. 3). In the proposed structure we have assumed $M=2$. Therefore, during each clock cycle a pair of inputs from each of the vectors is fed to the partial inner-product (PIP) generator, which produces all the $L$ PIPs corresponding to all the $L$ bits of the input operands and feeds to the shift-add tree. The output of shift-add tree is accumulated for $P$ cycles to produce the desired inner-product. The PIP generator consists of $L$ MUXes, which selects the desired PIPs based on the different bit-slices of 2-point vector derived from vector $B$. Note that proposed PIP generator does not contain any register or conventional RAM unit, but the LUT contents are generated by an adder.

IV. IMPLEMENTATION RESULTS AND DISCUSSIONS

The proposed DA-based structures for the computation of inner-product of variable vectors is coded in VHDL and
TABLE I

<table>
<thead>
<tr>
<th>Design</th>
<th>Word-Length, L</th>
<th>Inner-Product Length, N</th>
<th>DAT (ns)</th>
<th>Area (sq.um)</th>
<th>ADP (mW×ns)</th>
<th>Power (mW)</th>
<th>PDP (mW×ns)</th>
<th>ADP Saving (%)</th>
<th>PDP Saving (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference Design</td>
<td>8</td>
<td>10.00</td>
<td>5294</td>
<td>52940</td>
<td>0.37</td>
<td>3.72</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>20.88</td>
<td>7772</td>
<td>162279</td>
<td>0.66</td>
<td>13.83</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>42.24</td>
<td>12575</td>
<td>531168</td>
<td>1.23</td>
<td>52.20</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>88.64</td>
<td>22241</td>
<td>1971442</td>
<td>2.36</td>
<td>209.74</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Proposed Design</td>
<td>8</td>
<td>16.40</td>
<td>14122</td>
<td>231600</td>
<td>0.88</td>
<td>14.47</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>33.92</td>
<td>18960</td>
<td>643123</td>
<td>1.48</td>
<td>50.40</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>68.00</td>
<td>28619</td>
<td>1946092</td>
<td>2.65</td>
<td>180.42</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>136.00</td>
<td>48088</td>
<td>6539968</td>
<td>4.94</td>
<td>673.18</td>
<td>--</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

DAT: data arrival time, ADP: area-delay product, PDP: power-delay product.

synthesized by Synopsys Design Compiler using the TSMC 90-nm library [9] for different inner-product lengths \( N = 8, 16, 32 \) and 64 and word-lengths \( L = 8 \) and 16. The conventional multiplier based structures for those values of \( N \) and \( L \), using Synopsys DesignWare multiplier block, are also synthesized. The data arrival time (DAT), area, power consumption, area-delay product (ADP), and power-delay product (PDP) are estimated from the synthesis results and listed in Table I. It is found that the proposed structures for \( L = 8 \) and \( L = 16 \), respectively, are nearly 26% and 42% faster than the conventional multiplier-based implementation, in average, for different inner-product lengths. The proposed designs involve higher area complexity compared with the other but offer saving of nearly 22% and 36% ADP over the multiplier-based designs for \( L = 8 \) and \( L = 16 \), respectively, in average, for different inner-product lengths. Similarly, proposed designs offer saving of nearly 16% and 24% PDP over the multiplier-based designs for \( L = 8 \) and \( L = 16 \), respectively, in average, for different inner-product lengths.

V. SUMMARY AND CONCLUSIONS

The computation of inner-product of variable vectors is very often required for matrix-multiplication of different forms, and implementation of digital filters of unknown coefficients. But, the scope of DA for the computation of inner-product of variable vectors was not explored. Therefore, in this paper, we have analysed the design issues relating to DA-based implementation of inner-product of variable vectors, and derived a novel area-time efficient solution for bit-parallel DA-based implementation of inner-product of variable vectors and variable inner-product lengths. It is found that the proposed structures are nearly 34% faster than the conventional multiplier-based implementation, in average, for several different inner-product lengths and for input word-lengths \( L = 8 \) and \( L = 16 \). The proposed designs offer saving of nearly 22% and 36% ADP, and saving of nearly 16% and 24% PDP over the multiplier-based designs for \( L = 8 \) and \( L = 16 \), respectively, in average, for various inner-product lengths.

REFERENCES