# High-level synthesis algorithm for the design of reconfigurable constant multiplier

Chen, Jiajia; Chang, Chip Hong

2009

Chen, J., & Chang, C. H. (2009). High-level synthesis algorithm for the design of reconfigurable constant multiplier. IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, 28(12), 1844-1856.

https://hdl.handle.net/10356/80022

https://doi.org/10.1109/TCAD.2009.2030446

# High-Level Synthesis Algorithm for the Design of Reconfigurable Constant Multiplier

Jiajia Chen and Chip-Hong Chang, *Senior Member, IEEE*

***Abstract*—Multiplying a signal by a known constant is an essential operation in digital signal processing algorithms. In many application scenarios, an input or output signal is repeatedly multiplied by several predefined constants at different instances. These temporal redundancies can be exploited for the design of an efficient reconfigurable constant multiplier (RCM). An RCM achieves greater hardware savings than the conventional multiple constant multiplication architecture, limited only by the available latency of the subsystem. Motivated by a number of lucrative examples, this paper presents a new high-level design methodology for RCM. Common subexpressions in the preset constants represented in minimum signed-digit system are first eliminated to obtain a minimum depth multiroot directed acyclic graph (DAG). The DAG is converted into a primitive data flow graph (DFG) where mobile adders are identified. By scheduling each mobile adder into a control step within its legitimate time window with the minimum opportunity cost, mutually exclusive adders can be merged with significantly reduced adder and multiplexing cost. The opportunity cost for each scheduling decision is assessed by the probability displacement and disparity measures of the scheduled node as well as its predecessors and successors in the DFG. The algorithm is runtime efficient as exhaustive search for the best fusion of independently optimized constant multipliers has been avoided. Simulation results on randomly generated 12-b constant sets show that the solutions generated by the proposed algorithm are on average 19% to 25% more area–time efficient than the best reported solutions.**

***Index Terms*—High-level synthesis, multirate digital signal processing (DSP), reconfigurable constant multiplier (RCM), scheduling.**

## I. Introduction

MULTIPLICATION of a variable with constant is essential in many digital signal processing (DSP) applications. It can become a throughput bottleneck when many different constant multiplications are iteratively executed in the data path. Such computationally intensive kernels are commonly found in convolutions, correlations, inner products, fast Fourier transform, recursive discrete cosine transform (DCT), finite impulse response (FIR), and infinite impulse response filters. An analysis from over two hundred industry examples mainly on DSP, communication, graphics, and control applications showed that more than 60% of them have more than 20% of operations that are multiplications with constants [1]. As constant multiplication operations can be reused in many application domains, dedicated hardware resources and programmatic architecture generator capable of delivering a high-peak computational density (in bit operations per unit of silicon area per second) are desired to accelerate a well-defined set of repetitive operations. To maximize the computational density provided by a specialized resource, its use should be generalized, but the more it is generalized, the less suited it is for solving a particular problem [2], [3].

Reconfiguration provides advantages in irregular architectures, and previous work has shown that converting multifunction cores to several reconfigurable field-programmable gate array (FPGA) cores resulted in core area reduction of around 21% and a performance increase of 14% [4]. A FIR filter implementation using a single multiply accumulate (MAC) stage and employing real time reconfiguration to change tap values showed a 37% improvement in clock speed as compared to a static design [5]. However, each time an FPGA fabric is configured, there is a time penalty. Moreover, the time it takes to convert a system description to an FPGA configuration is significant. As the proportion of silicon area devoted to reconfigurable space scales up, the amount of configuration data that is required to set up large parallel structures and the interconnect congestion become a limiting factor.

An alternative approach to optimize computational density and reconfiguration time overhead is to focus on the design methodologies of parametric and lightweight problem-oriented special purpose architecture. Reconfigurable Multiplier Block (ReMB) mappings to FPGA have been proposed in [6]–[11] to specially reduce the complexity of the multiplier block of digital filter. These algorithms are developed to optimize the designs on FPGAs by efficiently utilizing the four-input lookup tables (LUTs). It is based on fixed graph topologies and relies on high-complexity exhaustive search to obtain the minimal solution. The latest known algorithms with similar objective are proposed in [12], [13], and the solution is technology mapped to the standard cell library in an application-specified integrated circuit (ASIC) design flow. Reference [12] reduces the number of partial products of the coefficient multiplier by modified Booth encoding but the sharing of common subexpressions is also limited by the encoding method. In [13], the method uses an existing algorithm [14], [15] to create the directed acyclic graph (DAG) representation for the constant coefficients and performs DAG fusion [13]. The best solution is returned after all admissible assignments have been attempted

Manuscript received January 7, 2009; revised May 4, 2009. Current version published November 18, 2009. This work was supported by the Singapore Ministry of Education's Academic Research Fund Tier 2 under Grant T208B1216. This paper was recommended by Associate Editor R. Camposano.

J. Chen is with 3M Singapore Pte. Ltd., Singapore 738205 (e-mail: herbert.chen@mmm.com).

C. H. Chang is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: echchang@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2009.2030446

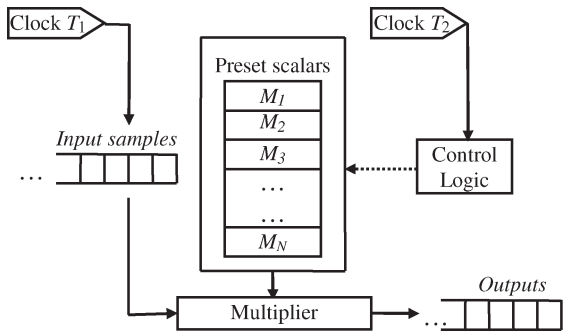



Fig. 1. RCM.

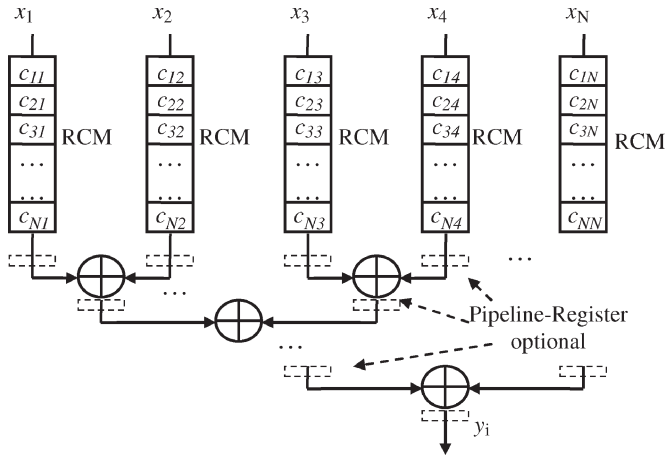



Fig. 2. RCM blocks in matrix multiplication.

for DAG fusion. However, the use of decimal numbers in DAG fusion assumes no specific binary format. Due to the enormous number of alternatives, it is difficult to explore the totality of adder redundancies over all constant multiplications. Common subexpression sharing may be undermined as the DAG and partial sums of each coefficient multiplier are independently optimized before they are fused. In addition, the algorithm requires exhaustive or quasi-exhaustive searches among existing DAG pairs, which results in high computational complexity.

Fusing independently optimized single constant multipliers is not always expedient for the general parallel multiple constant multiplication problems. Section II of this paper exemplifies several classes of system architectures whereby a reconfigurable constant multiplier (RCM) can be most cost effectively integrated without impediment to the system's throughput. Fig. 1 shows the general structure of an RCM where a serial constant multiplier is shared in several multiplications involving $N$ scalars specified *a priori*. The delay of the RCM is limited by the system's throughput $T_1$. The control logic operates at an internal rate $T_2$ which is $d$ time faster than $T_1$. The constant multiplier can be implemented multiplication free by an adder and shifter network. To amortize the processing bandwidth and resource utilization over all multiplications, the logic depth and adder width of the combinatorial adders are to be minimized while eliminating the redundancy among different scalar multiplications. The logic depth is constrained by $d$ and $T_2$, which, in turn, are dependent on the cost of realizing the adders (for the same operand length, faster adder is more costly) in the critical path. This relation has led us to the new formulation of RCM design as a programmatic resource scheduling and allocation problem. A rudimentary method adopting an as soon as possible (ASAP) scheduling strategy to solve this problem was proposed in [16]. In this paper, the adder resources are prepensely generated and minimized by common subexpression elimination (CSE). The data dependences of these adder resources are modeled by a data flow graph (DFG) in Section III. A new heuristic algorithm is proposed to schedule the adders in the DFG to maximize the resource utilization over several control steps to generate a more area $\times$ time (AT) product efficient RCM architecture than previously reported designs. The experimental results are compared with existing methods and discussed in Section IV.

## II. Hardware Reduction by Reconfigurable Constant Multiplications

Making a multiplier reconfigurable for different predefined scalars can provide a significant overall cost savings in many DSP applications. This section suggests several interesting application scenarios that motivate the design of efficient RCM. These scenarios show that ingenious deployment of RCMs can lower the hardware cost significantly at little or no penalty to the overall system performance.

A linear time invariant system without feedback can be described by a linear transform

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & \cdots & c_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{1}$$

where $N$ input data $\boldsymbol{X} = \{x_i\}$ are mapped by an $N \times N$ matrix $\boldsymbol{C} = \{c_{ij}\}$ to generate the outputs $\boldsymbol{Y} = \{y_i\}$. This matrix-vector multiplication is an expensive operation, which is also found in the state-space equations of systems involving feedback. In many DSP applications such as digital filtering [17]–[21], error detection and correction [22], fast Fourier and Consine transforms [23], autoregressive model [24], the matrix $\boldsymbol{C}$ consists of constants that are determined *a priori* from the system specifications. The parallel processing of $\boldsymbol{Y} = \boldsymbol{C} \times \boldsymbol{X}$ requires the replication of $N$ independent sum-of-products (SOP) processing units. Each SOP performs $N$ discrete constant multiplications with different input variables and $N - 1$ additions. In many situations, this quadratic increase in hardware cost is prohibitive due to the limitation in design areas. However, if the $N$ outputs are decimated in time, the $N \times N$ array of SOP units can be collapsed into $N$ parallel scalar multiplications such that each input data are multiplied by only one out of $N$ constant factors at any cycle, as shown in Fig. 2. The scalar products in each cycle are summed by an adder tree to produce an output element of $\boldsymbol{Y}$. The adder tree can be pipelined, if necessary, to improve the throughput. In this example, each column of $N$ processing units is reduced to a single RCM.

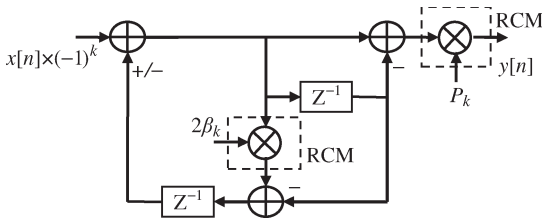



Fig. 3. Recursive Goertzel filter.

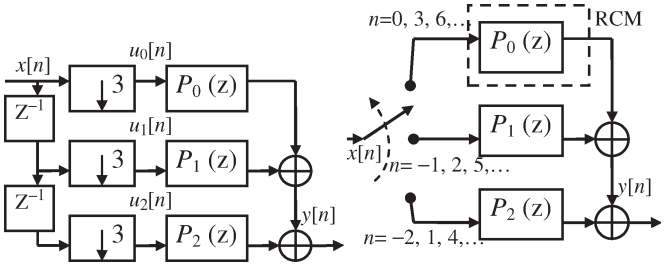



Fig. 4. Polyphase decimation filters ($M = 3$).

Goertzel algorithm has been used to reduce the complex multiplications to real multiplications in the recursive computation of discrete Fourier transform. Recently, it has been shown that Goertzel kernels also offer significant improvement in quantization noise performance for the DCT used in many multimedia and image compression applications [25]. The transfer function of an $N$-point DCT is given by

$$H(z) = \frac{P_k(1 - z^{-1})}{1 - 2\beta_k z^{-1} + z^{-2}} \tag{2}$$

where $k \in [0, \mathrm{N})$ is the frequency bin index. $\beta_k = \cos(k\pi/N)$ and $P_k = 4\beta_k/N$ for all $k$ except $k = 1$. For $k = 1$, $P_1 = (2\sqrt{2}/N)\beta_1$.

This transfer function can be realized with several resonator configurations of recursive Goertzel filter [25]. All Goertzel kernel structures involve a premultiplication of the difference signal by $2\beta_k$in the feedback path and a postmultiplication of the output signal by $P_k$. The two sets of scalars can be predetermined by the discrete cosine function according to the frequency bin index $k$ and the transform length $N$. The same recursive structure can be employed for all frequency bins by multiplexing it in time. One such architecture using two RCM units is shown in Fig. 3. Prestored LUTs have been avoided as the RCM can be implemented multiplication free by combinational adders. To cut down the circuit cost, the RCM can be designed to share the adder resources for different scalar multiplications of different frequency bins by exploiting the time-multiplexed system.

Multirate DSP uses decimators and interpolators to change the sampling rates of a system internally to maximize the performance of DSP system while keeping the cost down. Using the Noble relation [26], a complex DFG can be decomposed into several simpler DSP blocks, which can be processed parallelly at a faster rate. An example of a polyphase decomposition is shown in Fig. 4. The input signal $x$ is down sampled into $M$ subsequences by an $M$-fold decimator and each of the $M$ decimated input sequences is fed to a subfilter. Each subfilter is reduced by a factor of $M$ from the original filter length. The decimator generates one output for every $M$ input samples, and each subfilter can thus be processed at $M$ times slower rate than that of the original input sequence $x[n]$. In each decimation filter, each MAC implementation is realized by an ReMB. One input sample will be multiplied with $N$ constants generated from the ReMB sequentially and the products are stored into the partial sum store. As the input sample frequency has been reduced by $M$ times in each subfilter, the time interval between two samples are much longer. Therefore, the multiplier block in each subfilter can afford to be processed by a serial-parallel architecture by merging $L$ different coefficient multipliers into an RCM. The adder resources of the $L$ coefficient multipliers can then be shared by processing them serially, provided that the total time taken to produce the $L$ scalar multiples is able to meet the slower output rate of the subfilter. If the subfilter sampling period is not sufficient for one RCM to generate all distinct coefficient multiplier outputs, the filter coefficients can be partitioned into several subsets of distinct constants for an input sample to be concurrently multiplied by a few RCM units.

## III. Proposed High-Level Algorithm for RCM Design

The scenarios depicted in the previous section show that RCM units are amenable for integration into multirate digital systems or other subsystems when not all constant multiplications are processed in parallel. A practical consequence of these scenarios is the number of different multiplication operations is bound by the ratio of the system's sampling clock period and the critical path delay of the RCM. Our proposed design methodology allows the elementary operators in an RCM to assume different input operands to fulfill a prespecified set of scalar multiplications by suitably rearranging the input bits through a multiplexer network. The latency of the RCM is thus dependent on the control step granularity and the number of control steps used to execute the most time critical operation among a group of prespecified multiplications. The former is defined by the delay of the most critical adder, and the latter is constrained by the logic depth of the RCM. The resolution of any control step can be minimized by reducing the length of the adders in that control step and the worst case delay of each adder can be independently optimized, if necessary, by different parallel structures at the cost of hardware complexity. To meet the latency constraint with the best utilization of silicon resources, our design methodology emphasizes on a global reduction of adder depth while optimizing the number of adders, and the average sizes of adders and multiplexers. If the latency constraint cannot be met by a single RCM, this process itself also identifies underutilized hardware resources and provides an insight into the partitioning of operations into multiple RCM units in a partially parallel implementation.

### A. Design Problem Formulation

To allow for programmability of different scalar product generations, temporally independent adders are shared across different constant multiplications. This is illustrated with an example of four different constant multiplications using the values

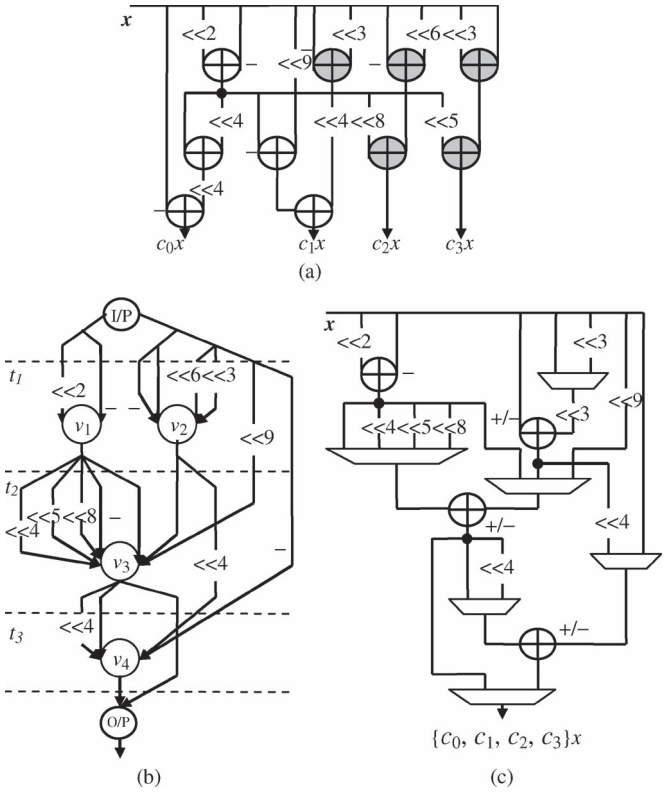



Fig. 5. Example of RCM design. (a) DAGs of constant multipliers. (b) Reduced DFG. (c) Mapping to RCM.

from, [1, Table II]. Fig. 5(a) shows the DAG representation of the four constants after eliminating the common subexpressions. Since an RCM makes only one multiplication at a time, the span of the input variable can be divided into as many number of control steps as, if not more than, the number of adders in the critical path of the DAG. The adders are then scheduled into the control steps without violating the precedence of any connected pair of adders in the DAG. The gray color adders in Fig. 5(a) are called the mobile adders. Mobile adders are critical variants to reduce the implementation complexity as they have some flexibility to be scheduled into more than one control steps without increasing the logic depth of the RCM.

To schedule the adders, the data dependence of DAG is compacted into a primitive DFG. A DFG $G$ has a node set $V$ and an edge set $E$. Each node, $v \in V$ represents an adder, and each edge $e \in E$ is an interconnection between two adders or between an adder and a primary input or output. Each edge is annotated with the output shift and a directed edge implies an intraprecedence constraint of two adders. If the shift amount is negative, it indicates that the node to which this edge is connected is a subtractor. For convenience, adder and subtractor are both called adder. The source and sink of the DFG are the input and output of the RCM, respectively.

In a properly scheduled DFG, each node can only fire when all its predecessor nodes have fired. Upon firing, a node computes the required value and places it on all its output edges. Fig. 5(b) shows a primitive DFG by merging some adders that are not executed concurrently. The number of adders has been reduced significantly from ten to four, but some adders have more inputs than needed to fire. This happens because every time an adder is eliminated, it is accompanied by an increase in the fan-out of some other adder. These contentions are resolved by introducing multiplexers as the point-to-point interconnecting elements between two nodes so that every node $v$ in a properly scheduled DFG has exactly two predecessor nodes firing at time $t$ and satisfies the following dependence relation:

$$v(t) = 2^{e_{l(v(t))}} l\left(v(t)\right) + 2^{e_{r(v(t))}} r\left(v(t)\right) \tag{3}$$

where $v(t)$ is the output of $v$ at time $t$. $l(v(t))$ and $r(v(t))$ are the left and right predecessor operators of $v(t)$. $e_u$ denote the amount of shifts applied to $u \in \{l(v(t), r(v(t)\}$.

To generate a scheduled DFG, cut lines are drawn through the primitive DFG of Fig. 5(b) so that only one adder depth is allowed between two cut lines. For every operator that has a fan-in greater than two, multiplexers are placed at the cut edge. Control steps $t_1$, $t_2$, and $t_3$ define the boundaries between the outputs of one adder or the source, and the inputs of another adder or the sink, by hard wirings or through multiplexers. The resolution of each control step needs not be equal. It is determined by the maximum delay of the adders in the critical paths that are allocated to the control step and their maximum input connection delay through the multiplexer, if the inputs are not directly wired to the adders. Fig. 5(c) shows the architectural mapping of the RCM from the scheduled DFG. Therefore, the design of an area–time efficient RCM can be viewed as the transformation of a given set of constants into a reduced DFG by scheduling and merging of the mobile adders that are not used simultaneously in the same control step. The quality of the solution is measured in terms of the adder and interconnection cost. The adder cost is contributed by the total number of nodes in the set $V$ of the final DFG while the cost of the interconnects is measured by the number and the size of multiplexers used.

### B. CSE

Without affecting the adder depth, the spatial redundancy in the original DAG is converted into temporal redundancy in the primitive DFG to increase the mobility of as many nodes as possible. The nodes in our DFG are atomic in that they encapsulate information about the elementary operations at bit level granularity. To preserve the atomic information of the nodes after removing the redundant adders, the detection and elimination of common subexpressions are better carried out in a positional representation.

Signed-digit (SD) representations are widely used for coefficient quantization and online arithmetic due to its attractive property for digit-serial and distributive operations. Minimum SD (MSD) [27] and its canonical SD (CSD) subset are two popular symmetric binary SD representations. They allow a fixed point number to be represented by a minimal number of SDs, such that its multiplication with a variable can be realized with reduced depth adder tree. Although representation with more SD terms can help to further reduce the adder cost [28], the improvement is made at the expense of higher logic depth and computational complexity. To reduce the search space, the uniqueness of CSD has been exploited by many CSE algorithms [29], [30] for common subexpressions but the

canonicity also limits the number of common subexpressions. For a given magnitude response specification, a signed powers-of-two coefficient set with more sharable subexpressions of 101 and 10 $\bar{1}$ can be synthesized from CSD or MSD by [31]. On the other hand, the advantage of having more subexpression sharing with binary representation [20], [18] is often offset by the higher number of nonzero power-of-two terms over MSD and CSD. Since solutions for MSD and CSD based on weight-two common subexpressions have the same minimum logic depth of $\lceil \log_2 n \rceil$, where $n$ is the maximum number of nonzero digits in a coefficient, the logic depths (LDs) of contention resolution algorithm (CRA) [32] and non-recursive signed common subexpression elimination (NR-SCSE) [30] reported in [20] are incorrect. Experiments run on random coefficient sets for CSE using binary, CSD, and MSD representations [18] concluded that MSD is preferred when seeking minimum delay solution. As minimum delay solution is critical for the scenarios discussed in Section II, MSD representation of integers will be considered.

We minimize the latency of the RCM by minimizing the adder depth. If the latency is relaxed, more control steps than the adder depth can be allocated which makes every adder mobile. Finding an optimal CSE solution for a set of integers in the MSD space is an NP-complete problem. The search space is reduced by detecting only common subexpressions of hamming weight two instead of all possible common subexpressions in the set of integers. These common subexpressions can be categorized into two different types. An even parity subexpression takes the form, IOI, and an odd parity subexpression takes the form IO$\bar{\text{I}}$, where I $\in \{1, \bar{1}\}$ and O is either void or a string of zeros. The number of zeros in O is the distance of the subexpression. This distinction allows the frequencies of different weight-two subexpressions to be tracked by a simple 2-D $PT$ array defined in [30].

*Definition 1:* A $PT$ array of a set of MSD numbers $C = \{c_i\}$ is a $2 \times (B-1)$ dimensional array. The entry in the upper (lower) row and the $j$th column represents the frequency of occurrences of even (odd) parity subexpressions of distance $j$, where $j = 0, 1, \ldots, B-1$ and $B$ is the maximum wordlength of $c_i$.

As an example, an MSD representation for the constants of Fig. 5 and its corresponding $PT$ array is given by

$$H = \begin{Bmatrix} 815 \\ 831 \\ 621 \\ 105 \end{Bmatrix} = \begin{Bmatrix} 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & \bar{1} \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \bar{1} \\ 0 & 1 & 0 & 1 & 0 & 0 & \bar{1} & 0 & 0 & \bar{1} & \bar{1} \\ 0 & 0 & 0 & 1 & 0 & 0 & \bar{1} & \bar{1} & 0 & 0 & 1 \end{Bmatrix}$$

$$PT(H) = \begin{bmatrix} 5 & 2 & 3 & 3 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 2 & 2 & 1 & 3 & 3 & 0 \end{bmatrix}.$$

An entry in the $PT(C)$ array with value greater than one indicates the presence of a common subexpression in $C$. The number of adders saved by every common subexpression is one less than its frequency in the $PT$ array. We adopt the MSD generation method of [33] to generate all possible sets of MSD numbers for a given set of constants. With the help of the $PT$ arrays, the MSD set with the highest total saving of adders is selected. The common subexpression with the highest value in the $PT$ array is first identified, and an adder node $v$ is created in a DAG to add the two operands corresponding to the input variable shifted by the positions of the two nonzero digits. If there are more than one entry with the same highest value, the one with the least distance $d$ is selected to minimize the wordlength of the adder. The distance $d$ and coefficients from which the common subexpression is detected are also recorded. These information are used to determine the amount of shifts applied to the left and right operands of $v$ and the firing time $t$ of $v$. The two nonzero digits of the common subexpression are replaced by zeros in all their occurrences in the set of MSD coefficients. The $PT$ array is updated to reflect the removal of this subexpression. This process of searching for the highest frequency subexpression and the creation of adder node in the DAG to eliminate the common subexpression from the MSD coefficient set is repeated until no entry in the $PT$ array is greater than one. Finally, new nodes are created to add the operands corresponding to the remnant nonzero digits (which are not parts of the common subexpressions) to the outputs of the existing nodes in the DAG accordingly.

### C. Proposed Mobile Operator Scheduling Scheme

After the aforementioned global CSE process, the total number of adders in the DAG $G$ can be further reduced by merging the adders allocated for different constant multiplications. There are many different ways of merging the adders without violating the intraprecedence relationship of $G$ by routing the inputs to an adder from the output of another adder or the primary inputs through multiplexers. The objective of our proposed adder scheduling scheme is to minimize the total cost of the adders and multiplexers within the given latency constraint. Let $D(G)$ be the adder depth of $G$. The latency constraint can be sliced into a number of control steps $t_1, t_2, \ldots, t_{\max}$ so that $t_{\max} \geq D(G)$ and each control step $t_i$ has a time resolution sufficient to complete the longest latency addition scheduled to that control step. The time resolution of a control step can be reduced by minimizing the wordlengths of the merged adders allocated to the control step without resorting to faster but more complex adder structure. This is possible if there exist mobile adders.

An adder is said to be mobile if it can be assigned to more than one control step in a DFG. The mobility of a node $v$ in $G$ is defined as $\mu(v) = t_{\max} - \delta(v)$, where $\delta(v)$ is the number of nodes in the longest path through $v$ from a root to a leaf. The significance of the mobility of a node is that before scheduling, the probability of a node $v$ appearing in a control step $t_i$, denoted by $p_v(t_i)$, is equal to $\{\mu(v)+1\}^{-1}$ for $t_i \in [t_e, t_l]$ and zero outside this time window, where $t_e$ and $t_l$ are the earliest and the latest control steps the node can fire due to the intraprecedence relation of the DFG. Scheduling a mobile adder to a particular control step fixes its probability to one in that control step and to zero in all other control steps. The probability displacement before and after scheduling a mobile node $v$ to a specific control step $t_j \in [t_e, t_l]$ is given by

$$\Delta p_v(t_i) = \begin{cases} \frac{\mu(v)}{\mu(v)+1}, & \text{if } t_i = t_j \\ -\frac{1}{\mu(v)+1}, & \text{if } t_i \neq t_j \end{cases} \tag{4}$$

where $t_i, t_j \in [t_e, t_l]$.

This change in adder probability is associated with the opportunity cost of merging a mobile adder with the mutually exclusive adders in any control step within the time window defined by $[t_e, t_l]$. Two adders in the same control steps are said to be mutually exclusive if they are not used to generate the same scalar multiplier output simultaneously. Merging two mutually exclusive adders results in a saving of one adder but the merged adder assumes the length of the longer adder. In other words, the output of the shorter length adder will be expanded by an amount equal to the difference between the two adder lengths. This will increase the bitwidths of its successive adders and multiplexers. To minimize the cost of RCM, it is important to schedule the mobile node to a control step that will minimize the disparities in the bitwidths among mutually exclusive adders.

The output of each node $v(t) = a(v) \cdot x(t)$, where $x(t)$ is the input variable $x$ to the RCM, and the constant $a(v)$ can be expressed in a SD representation as follows:

$$a(v) = \sum_{i=1}^{h(v)} s_i 2^{w_i} \tag{5}$$

where $h(v)$ is the hamming weight of $a(v)$, $s_i \in \{-1, 1\}$ is the $i$th SD and $w_i \in \{0, 1, 2, \ldots, w_{\max}\}$ is the weighted bit position of $s_i$.

From [34], if the length of the variable $x$ is fixed, the bitwidth of an adder is largely dependent on the difference between the maximum most significant and the maximum least significant SD positions of its input operands. Since the SDs of $a(v)$ are inherited from those of its input operands and the input operands to an adder are commutative, a meaningful metric to assess the disparity of mutually exclusive adders is through the aggregate distance of all SDs to the leading SD position $w_l$. Let

$$W(v) = \sum_{i=1}^{h(v)-1} (w_l - w_i). \tag{6}$$

The disparity of a node $v$ in control step $t_i$ is defined as

$$\Delta W_v(t_i) = \left| W(v) - \frac{1}{M_i} \times \left( \sum_{j=1}^{M_i} W(u_j) \right) \right| \tag{7}$$

where $M_i$ is the total number of existing nodes in control step $t_i$ that are mutually exclusive to $v$.

The aggregate distance $W(v)$ measures how compact the sign digits in $a(v)$ are distributed and the disparity of $v$ is an indicator of the bitwidth variation by merging the existing adders in the same control step with $v$. Fig. 6 shows the disparities of mobile nodes $v_4$, $v_5$, and $v_6$ in control steps $t_1$ to $t_3$ after nodes $v_1$, $v_2$, and $v_3$ have been scheduled. Scheduling $v_4$ to $t_1$, $v_5$ to $t_1$ or $t_2$, and $v_6$ to $t_2$ will minimize the disparity and reduced the costs of the merged adders and multiplexers.

To minimize the bitwidth of the merged adders and multiplexers, mobile nodes should be scheduled to control steps with minimal disparity. However, scheduling a mobile node to a particular control step will restrict the mobility of its predecessor and successor nodes and hence the probabilities and disparities of their scheduling in other control steps. The overall effect can be accounted by the probability displacement and the time window reduction of the predecessor and successor nodes when evaluating the opportunity cost of scheduling a mobile node $v$ in control step $t_i \in [t_e, t_l]$. This opportunity cost is given by

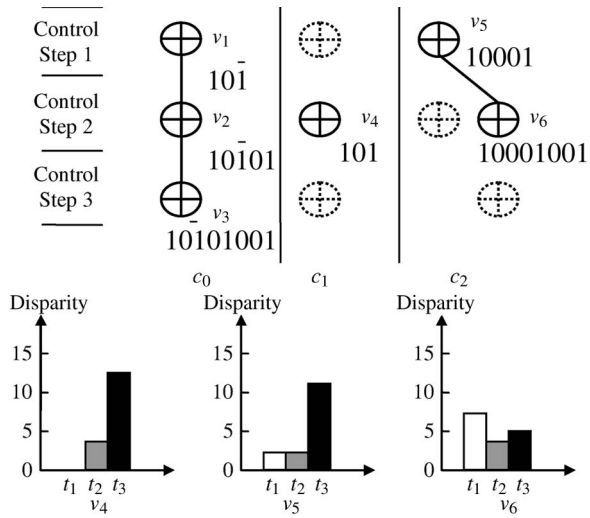



Fig. 6. Disparities of operators scheduled in different control steps.

$$\begin{aligned} \text{Cost}(v, t_i) = & \sum_{t=t_e}^{t_l} \Delta p_v(t) \cdot \Delta W_v(t) \\ & + \sum_{u \in \text{parent}(v)} \sum_{t=\tilde{t}_e}^{\tilde{t}_l} \Delta p_u(t) \cdot \Delta W_u(t) \\ & + \sum_{u \in \text{children}(v)} \sum_{t=\tilde{t}_e}^{\tilde{t}_l} \Delta p_u(t) \cdot \Delta W_u(t) \end{aligned} \tag{8}$$

where $[\tilde{t}_e, \tilde{t}_l]$ denotes the reduced time windows of the predecessor or successor nodes, in general, due to the scheduling of $v$ to control step $t_i$. parent$(v)$ and children$(v)$ refer to the immediate predecessors and successors of $v$, respectively.

To minimize the computational complexity, the opportunity costs are evaluated in batches of identical mobility nodes, commencing from the least mobility nodes. When a node with the least opportunity cost is scheduled, the mobilities of all its affected predecessor and successor nodes are recalculated. After all nodes with the least mobility have been allocated into the DFG, the opportunity costs of all mobile nodes with the next higher mobility are evaluated and the least cost mobile node will be scheduled. The process repeats until all nodes in the DAG have been scheduled into the DFG.

### *D. Adder Merging and Multiplexer Splitting*

After all nodes in the DAG have been scheduled into a DFG, the DFG are reduced by merging mutually exclusive nodes in

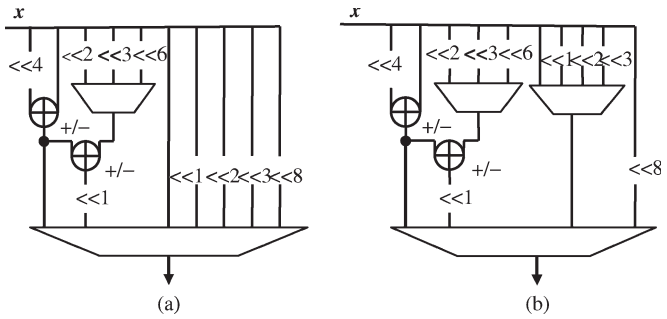



Fig. 7. RCM design (a) before and (b) after multiplexer splitting.

the same control step. For the calculation of disparity, each node $v$ in the DAG is annotated with a set of labels $\phi(v)$ to indicate its phases of computation in the RCM. An RCM is programmed to output only one product of an input variable $x$ and a constant $c_i$ in any one phase but each node in a reduced DAG may involve in more than one phase of computation due to the presence of common subexpressions. Using the index of the coefficient as phase number, the criterion for merging two scheduled nodes $v$ and $u$ is given by $\phi(v) \cap \phi(u) = \varnothing$. Merging two mutually exclusive nodes $v$ and $u$ introduces at least a multiplexer at either or both inputs of the merged node $\hat{v}$, where

$$\text{child}(\hat{v}, t) = \overline{c(t)} \cdot \text{child}(v, t) + c(t) \cdot \text{child}(u, t) \qquad (9)$$

where the control input to the multiplexer $c(t) = 0$ if $t = \phi(v)$ and 1 if $t = \phi(u)$. $\text{child}(v, t) \in \{l(v(t)), r(v(t))\}$ and $\text{child}(u, t) \in \{l(u(t)), r(u(t))\}$ with the constraint that $l(\hat{v}, t) \neq r(\hat{v}, t)$.

In general, more than one mutually exclusive node can be merged after scheduling. Every legitimate merging increases the in degree of the merged node in the DFG. Large multiplexers for high in degree nodes can be decomposed into smaller multiplexers. The data inputs to the large multiplexer can be split and rescheduled into one or more smaller multiplexers. The split inputs can be merged into existing smaller multiplexers in the earlier control steps instead of forming new multiplexers provided that these inputs are not required by the adders in two consecutive control steps. Multiplexer splitting attempts to speed up the connection time by reducing the fan-ins of large multiplexers at the expense of increasing the fan-ins of smaller multiplexers. Fig. 7 shows the result of multiplexer load balancing by extracting four data inputs from an eight-to-one multiplexer into a new four-to-one multiplexer so that the interconnection time is reduced in the critical path.

An $m$-to-one multiplexer has $m$ data inputs and the number $m$ is also called the fan-in of the multiplexer. It should be noted that to reduce the adder width, appropriate amount of shifts have been applied to the inputs and output of each adder during the scheduling and merging processes so that the most significant digit of its $a(v)$ is either 1 or $-1$. Thus, the data inputs to the multiplexer have nonuniform bitwidth. Unfortunately, the cost of a multiplexer is a function of its bitwidth and the bitwidth of a multiplexer must be catered to the largest data input [13]. This cost may be reduced by splitting

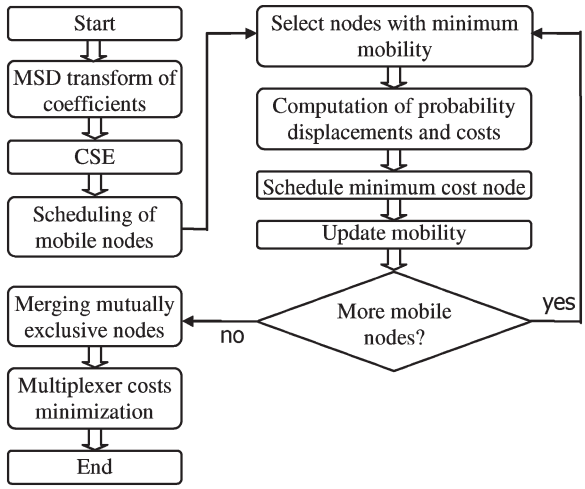



Fig. 8. Flow chart of the proposed algorithm.

a large multiplexer into two smaller multiplexers. Which data inputs to be split can be judicially decided by examining the disparity of each fan-in, which takes a similar form as (7). The disparity of a data input $f_i$ to an $m$-to-one multiplexer is defined as

$$\Delta b(f_i) = \left| b(f_i) - \frac{1}{m} \times \left( \sum_{j=1}^{m} b(f_j) \right) \right| \qquad (10)$$

where $b(f_i)$ is the bitwidth of the $i$th data input of the multiplexer.

$\Delta b(f_i)$ measures the deviation of the bitwidth of a specific data input $f_i$ from the average bitwidth of a multiplexer. Decoding of wide multiplexers requires large logical effort. When the stray capacitance is considered, over a broad range of assumption, the best multiplexer has 4 inputs [35]. For each multiplexer, MUX with $m \geq 4$, we first assume that it is cost effective to decompose it into two multiplexers $\text{MUX}_1$ and $\text{MUX}_2$. Let $F$ be the set of data inputs to MUX sorted by their bitwidths, i.e., $F = \{f_i\}_{i=1}^{m}$ with $b(f_i) \leq b(f_j)$ if $i < j$. Initially, the data input sets of $\text{MUX}_1$ and $\text{MUX}_2$ are set to $F_1 = \{f_i\}_{i=2}^{m}$ and $F_2 = \{f_1\}$, respectively. Then, the disparity of the smallest data input of $F_1$, i.e., $\Delta b_1(f_j)$ for $j = \arg\{\min(F_1)\}$ is computed and compared with the average bitwidth of $\text{MUX}_2$, $ave(F_2)$. If $\Delta b_1(f_j) < ave(F_2)$, then $F_1 = F_1 - \{f_j\}$ and $F_2 = F_2 + \{f_j\}$. The process is repeated with the next smallest data inputs of $F_1$ until all data inputs of $F_1$ have been evaluated. If the final fan-in of $\text{MUX}_2$, $|F_2| \geq 2$, MUX will be decomposed into $\text{MUX}_1$ and $\text{MUX}_2$. Otherwise, MUX will be preserved.

The flow chart in Fig. 8 summarizes the proposed optimization strategy for the design of RCM.

### *E. Design Example*

A design example of an RCM for a eight-point Goertzel recursive DCT from [6, Fig. 7] is used to illustrate the proposed adder scheduling algorithm. The three constants of the RCM are $C = \{362, 392, 473\}$. The best MSD set with the highest

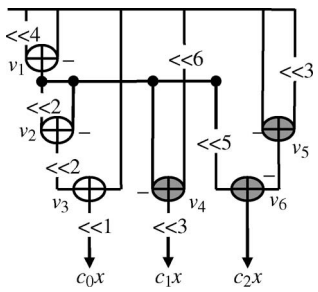



Fig. 9. DAG of constant multipliers with $C = \{362, 392, 473\}$.

common subexpression frequency happens to be its CSD subset and its $PT$ array are given as follows:

$$C = \begin{Bmatrix} 362 \\ 392 \\ 473 \end{Bmatrix} = \begin{Bmatrix} 1 & 0 & \bar{1} & 0 & \bar{1} & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & \bar{1} & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \bar{1} & 0 & \bar{1} & 0 & 0 & 1 \end{Bmatrix}$$

$$PT(C) = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 3 & 1 & 4 & 1 & 2 & 0 & 0 & 0 \end{bmatrix}.$$

Following the CSE procedure described in Section III-B, the reduced DAG is shown in Fig. 9.

We assume the number of control steps to be the minimal adder depth of the reduced DAG for the most stringent timing requirement. Therefore, $t_{\max} = 3$. From the DAG, the number of nodes in the longest path through each node is given by: $\delta(v_1) = \delta(v_2) = \delta(v_3) = 3$ and $\delta(v_4) = \delta(v_5) = \delta(v_6) = 2$. The computation phases of each node in the DFG are $\phi(v_1) = \{0, 1, 2\}$, $\phi(v_2) = \phi(v_3) = \{0\}$, $\phi(v_4) = \{1\}$, $\phi(v_5) = \phi(v_6) = \{2\}$. The three fixed nodes are identified by $\mu(v_1) = \mu(v_2) = \mu(v_3) = 0$. These three adders will be directly scheduled into control steps $t_1$, $t_2$, and $t_3$, respectively. Since $\mu(v_4) = \mu(v_5) = \mu(v_6) = 1$, we need to compute the opportunity costs of scheduling these mobile nodes into specific control steps within their legitimate time windows.

Consider $v_4$. From (7), since $a(v_4) = 10\bar{1}0001$, the aggregate distance of $v_4$ is $W(v_4) = (w_3 - w_1) + (w_3 - w_2) = (6-0) + (6-4) = 8$. $v_4$ can be scheduled into either control step $t_2$ or $t_3$. According to (8), its disparity in $t_2$ and $t_3$ can be calculated as follows. If $v_4$ is scheduled into $t_2$, $\Delta p_{v_4}(t_2) = 0.5$ and $\Delta p_{v_4}(t_3) = -0.5$. Since $v_2$ is the only scheduled (fixed) node in $t_2$ and $\phi(v_2) \cap \phi(v_4) = \varnothing$, $v_2$ and $v_4$ are mutually exclusive. From $a(v_2) = 10\bar{1}0\bar{1}01$, $W(v_2) = (w_4 - w_1) + (w_4 - w_2) + (w_4 - w_3) = (6-0) + (6-2) + (6-4) = 12$ and $\Delta W_{v_4}(t_2) = |W(v_4) - W(v_2)| = |8 - 12| = 4$. Similarly, $v_3$ is the only fixed node in $t_3$ and $\phi(v_3) \cap \phi(v_4) = \varnothing$. From $a(v_3) = 10\bar{1}0\bar{1}0101$, $W(v_3) = (w_5 - w_1) + (w_5 - w_2) + (w_5 - w_3) + (w_5 - w_4) = (8-0) + (8-2) + (8-4) + (8-6) = 20$ and $\Delta W_{v_4}(t_3) = |W(v_4) - W(v_3)| = |8 - 20| = 12$. As $v_4$ has no unscheduled predecessor or successor node, according to (9), the opportunity cost of scheduling $v_4$ into step $t_2$ is $\text{Cost}(v_4, t_2) = 0.5 \times 4 + (-0.5) \times 12 = -4$. If $v_4$ is scheduled into $t_3$, $\Delta p_{v_4}(t_2) = -0.5$ and $\Delta p_{v_4}(t_3) = 0.5$. Hence, $\text{Cost}(v_4, t_3) = (-0.5) \times 4 + 0.5 \times 12 = 4$.

Consider $v_5$. Since $a(v_5) = \bar{1}001$, $W(v_5) = 3$. $v_5$ can be scheduled into either $t_1$ or $t_2$. Since the only scheduled adder in $t_1$, $v_1$, is involved in all phases of computation, $\phi(v_1) \cap \phi(v_5) \neq \varnothing$. There is no fixed node that is mutually exclusive with $v_5$ in $t_1$ and $\Delta W_{v_5}(t_1) = |W(v_5) - 0| = 3$. In addition, $v_2$ is the only fixed node in $t_2$ and $\phi(v_2) \cap \phi(v_5) = \varnothing$. Thus, $v_2$ and $v_4$ are mutually exclusive and $\Delta W_{v_5}(t_2) = |W(v_5) - W(v_2)| = 9$. If $v_5$ is scheduled into $t_1$, $\Delta p_{v_5}(t_1) = 0.5$ and $\Delta p_{v_5}(t_2) = -0.5$ and its successor node, $v_6$, can still be scheduled into $t_2$ and $t_3$. There is no change in the probability of $v_6$ and $\Delta p_{v_6}(t_2) = \Delta p_{v_6}(t_3) = 0$. According to (9), $\text{Cost}(v_5, t_1) = 0.5 \times 3 + (-0.5) \times 9 = -3$. If $v_5$ is scheduled into $t_2$, $\Delta p_{v_5}(t_1) = -0.5$ and $\Delta p_{v_5}(t_2) = 0.5$. Its successor node, $v_6$, has to be scheduled into $t_3$. Therefore, $\Delta p_{v_6}(t_2) = -0.5$ and $\Delta p_{v_6}(t_3) = 0.5$. Since $a(v_6) = 1000\bar{1}0\bar{1}001$, $W(v_6) = 19$. There is only one fixed node, $v_2$, in $t_2$ and $\phi(v_2) \cap \phi(v_6) = \varnothing$. Thus, $\Delta W_{v_6}(t_2) = |W(v_6) - W(v_2)| = 7$. $v_3$ is the only fixed node in $t_3$ and $\phi(v_3) \cap \phi(v_6) = \varnothing$. Thus, $\Delta W_{v_6}(t_3) = |W(v_6) - W(v_3)| = 1$. According to (9), $\text{Cost}(v_5, t_2) = [(-0.5) \times 3 + 0.5 \times 9] + [(-0.5) \times 7 + 0.5 \times 1] = 0$.

Now, consider node $v_6$. Since $v_6$ can be scheduled into either $t_2$ or $t_3$. If $v_6$ is scheduled into $t_2$, $\Delta p_{v_6}(t_2) = 0.5$ and $\Delta p_{v_6}(t_3) = -0.5$. Its predecessor node, $v_5$, has to be scheduled into $t_1$. Thus, $\Delta p_{v_5}(t_1) = 0.5$ and $\Delta p_{v_5}(t_2) = -0.5$. $\text{Cost}(v_6, t_2) = [0.5 \times 7 + (-0.5) \times 1] + [0.5 \times 3 + (-0.5) \times 9] = 0$. If $v_6$ is scheduled into $t_3$, $\Delta p_{v_6}(t_2) = -0.5$ and $\Delta p_{v_6}(t_3) = 0.5$. $v_5$ can still be scheduled into $t_1$ and $t_2$. Since there is no change in the probability of $v_5$, $\Delta p_{v_5}(t_1) = \Delta p_{v_5}(t_2) = 0$. According to (9), $\text{Cost}(v_6, t_3) = (-0.5) \times 7 + 0.5 \times 1 = -3$.

TABLE I
TRACES OF COMPUTATION OF THE PROPOSED SCHEDULING ALGORITHM

| Stage | $v$ | $W(v)$ | $t$ | $\Delta W_v(t)$ | Cost($v$, $t$) |
|---|---|---|---|---|---|
| 1 | $v_4$ | 8 | $t_2$ | 4 | **−4** |
| | | | $t_3$ | 12 | 4 |
| | $v_5$ | 3 | $t_1$ | 3 | −3 |
| | | | $t_2$ | 9 | 0 |
| | $v_6$ | 19 | $t_2$ | 7 | 0 |
| | | | $t_3$ | 1 | −3 |
| 2 | $v_5$ | 3 | $t_1$ | 3 | −2 |
| | | | $t_2$ | 7 | −2 |
| | $v_6$ | 19 | $t_2$ | 9 | 2 |
| | | | $t_3$ | 1 | **−4** |
| 3 | $v_5$ | 3 | $t_1$ | 3 | **−2** |
| | | | $t_2$ | 7 | 2 |

The traces of computation are summarized in Table I. In Stage 1, the minimum opportunity cost is $-4$. Consequently, $v_4$ is scheduled into $t_2$ and become a fixed node. The opportunity costs of scheduling $v_5$ and $v_6$ need to be reevaluated after updating their mobilities. The traces of computation are shown in Stage 2 of Table I, from which $v_6$ is scheduled into $t_3$ with the minimum opportunity cost. After fixing $v_6$ in $t_3$, the only mobile node is $v_5$. It can be scheduled in either $t_1$ or $t_2$ as computed in Stage 3. The scheduling of $v_5$ into $t_1$ has less opportunity cost, hence $v_5$ is scheduled into $t_1$.

After all mobile nodes have been fixed, the mutually exclusive nodes in the same control steps are merged. Multiplexers are inserted between the nodes with fan-in greater than two and their immediate predecessor nodes or primary inputs. Since all

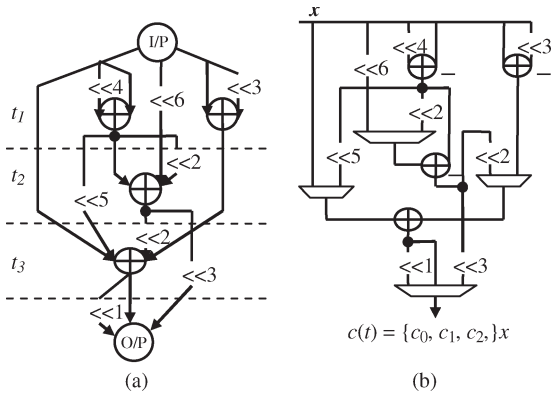



Fig. 10. RCM of the design example. (a) DFG. (b) Circuit architecture.

multiplexers have fan-in less than four, no multiplexer splitting is performed. The optimized DFG and the RCM circuit are shown in Fig. 10.

## IV. Experimental Results and Discussion

In this section, the area and time complexities of the RCM solutions generated by our proposed method are evaluated and compared against other methods in the literature. In the first part of the experiments, four commonly cited sets of constants in the ReMBs are considered. Set A is obtained from [7, Fig. 6], Sets B and C are taken from [6, Figs. 7 and 8], respectively, and Set D is taken from [9, Fig. 6]. Unitless cost function in [13] is adopted for the area estimate. In this estimate, the cost of an operator is equal to $a \cdot k$, where $k$ is the bitwidth of the operator and $a$ is a constant of proportionality. The value of $a$ can be characterized by the implementation technology. Technology mapping of typical operators using a commercial 0.18-$\mu$m standard cell library [13] yields $a_{m-1\text{multiplexer}} = 14 \cdot m$, $a_{\text{adder}} = 67$, $a_{\text{subtractor}} = 75$, and $a_{\text{adder/subtractor}} = 98$ measured in square micrometers. With Artisan TSMC 0.18-$\mu$m standard cell library, $a_{m-1\text{multiplexer}} = 13.3 \cdot m$ and $a_{\text{adder}} = 69.9$, which show that the same cells from different libraries developed for the same process technology have very similar areas. With Avant! Passport 0.35-$\mu$m standard cell library, $a_{m-1\text{multiplexer}} = 1.17 \cdot m$, $a_{\text{adder}} = 5.0$, $a_{\text{subtractor}} = 5.66$, and $a_{\text{adder/subtractor}} = 7.0$ measured in number of equivalent gates.

The numbers of operators and multiplexers used in our designs in RCM and their area costs, "I" and "II" estimated based on 0.18- and 0.35-$\mu$m cell libraries, respectively, are presented in Tables II–V. The results are compared with those reported in [6], [7], and [9] by ReMB methods, [13] by DAG fusion algorithm, and [16] by our preliminary ASAP scheduling approach. The bitwidth of the input signal to the RCM is assumed to be 8 b for all designs. It should be noted that the result of DAG fusion for Set C is different from that of [13] because of an erratum in [13]. The inputs to the top 2-1 multiplexer in, Fig. 16(c)[13] should be multiplied by "16 and 4," instead of "4 and 2." We have verified this using the online tool provided by authors of [13]. The results reported in Table IV are based on the corrected design of DAG fusion.

From Tables II–V, the proposed RCM design algorithm generates lower cost solutions than the classical ReMB approaches. On average, the proposed algorithm is 19.3% more area efficient than ReMB algorithms, [6], [7] and [9], which are originally designed to target on FPGAs. This indicates that the ReMB approaches may not be as efficient on ASIC implementation as they are on FPGA. For ASIC implementation, a more insightful conclusion can be drawn from the comparison with the solutions generated by DAG fusion [13], which is not tailored to dedicated configurable logic cell structure. On average, the proposed algorithm is still 7% more area efficient than this latest and most competitive method [13]. This saving is mainly contributed by the holistic consideration of opportunity costs in scheduling mobile adders. Merging adders with minimal disparity has successfully reduced the overall bitwidth of the adders and multiplexers. As shown in Tables III–V,

TABLE II
Area Cost Estimation for Data Set A

| Algorithm | Adder | Subtractor | Adder/ Subtractor | Multiplexers | I | II |
|---|---|---|---|---|---|---|
| DAG Fusion [13] | 0 | 0 | 1× (12-bit)<br>1× (14-bit) | 1×Mux$_{3\text{-to-}1}$(14-bit)<br>1×Mux$_{7\text{-to-}1}$(16-bit) | 4704 | 361.7 |
| ReMB [7] | 1×(10-bit) | 1×(12-bit) | 1×(10-bit)<br>2×(11-bit)<br>1×(12-bit)<br>1×(16-bit) | 1×Mux$_{2\text{-to-}1}$(8-bit)<br>1×Mux$_{2\text{-to-}1}$(9-bit)<br>2×Mux$_{2\text{-to-}1}$(10-bit)<br>1×Mux$_{2\text{-to-}1}$(11-bit)<br>1×Mux$_{2\text{-to-}1}$(12-bit)<br>1×Mux$_{2\text{-to-}1}$(16-bit) | 9578 | 715.3 |
| ASAP [16] | 0 | 0 | 1× (12-bit)<br>1× (14-bit) | 1×Mux$_{3\text{-to-}1}$(14-bit)<br>1×Mux$_{4\text{-to-}1}$(11-bit)<br>1×Mux$_{4\text{-to-}1}$(16-bit) | 4648 | 357.0 |
| Proposed | 0 | 0 | 1× (12-bit)<br>1× (14-bit) | 1×Mux$_{3\text{-to-}1}$(14-bit)<br>1×Mux$_{4\text{-to-}1}$(11-bit)<br>1×Mux$_{4\text{-to-}1}$(16-bit) | 4648 | 357.0 |

TABLE III
Area Cost Estimation for Data Set B

| Algorithm | Adder | Subtractor | Adder/ Subtractor | Multiplexers | I | II |
|---|---|---|---|---|---|---|
| DAG Fusion [13] | 1×(19-bit) | 1× (16-bit) | 1× (12-bit) | 2×Mux$_{2\text{-to-}1}$(12-bit)<br>1×Mux$_{2\text{-to-}1}$(19-bit)<br>1×Mux$_{3\text{-to-}1}$(16-bit)<br>1×Mux$_{3\text{-to-}1}$(20-bit) | 6365 | 496.0 |
| ReMB [6] | 1×(11-bit)<br>1×(12-bit)<br>1×(17-bit) | 0 | 1×(11-bit) | 3×Mux$_{2\text{-to-}1}$(11-bit)<br>1×Mux$_{2\text{-to-}1}$(14-bit) | 5074 | 386.7 |
| ASAP [16] | 2×(17-bit) | 0 | 1×(10-bit)<br>1×(17-bit) | 1×Mux$_{2\text{-to-}1}$(13-bit)<br>2×Mux$_{2\text{-to-}1}$(17-bit) | 6240 | 486.5 |
| Proposed | 1×(17-bit) | 1× (11-bit)<br>1× (12-bit)<br>1× (14-bit) | 0 | 1×Mux$_{2\text{-to-}1}$(14-bit)<br>1×Mux$_{2\text{-to-}1}$(16-bit)<br>1×Mux$_{2\text{-to-}1}$(17-bit)<br>1×Mux$_{3\text{-to-}1}$(18-bit) | 5986 | 467.3 |

TABLE IV
Area Cost Estimation for Data Set C

| Algorithm | Adder | Subtractor | Adder/ Subtractor | Multiplexers | I | II |
|---|---|---|---|---|---|---|
| DAG Fusion [13] | 0 | 1× (16-bit) | 1× (12-bit) | 2×Mux$_{2\text{-to-}1}$(12-bit)<br>1×Mux$_{3\text{-to-}1}$(16-bit)<br>1×Mux$_{3\text{-to-}1}$(18-bit) | 4476 | 349.7 |
| ReMB [6] | 0 | 1×(15-bit) | 1×(12-bit)<br>1×(15-bit) | 2×Mux$_{2\text{-to-}1}$(10-bit)<br>1×Mux$_{2\text{-to-}1}$(15-bit) | 4751 | 355.7 |
| ASAP [16] | 0 | 0 | 1× (10-bit)<br>1× (14-bit) | 1×Mux$_{2\text{-to-}1}$(10-bit)<br>1×Mux$_{2\text{-to-}1}$(11-bit)<br>1×Mux$_{3\text{-to-}1}$(16-bit) | 3612 | 273.0 |
| Proposed | 0 | 0 | 1× (10-bit)<br>1× (14-bit) | 1×Mux$_{2\text{-to-}1}$(10-bit)<br>1×Mux$_{2\text{-to-}1}$(11-bit)<br>1×Mux$_{3\text{-to-}1}$(16-bit) | 3612 | 273.0 |

TABLE V
AREA COST ESTIMATION FOR DATA SET D

| Algorithm | Adder | Subtractor | Adder/ Subtractor | Multiplexers | I | II |
|---|---|---|---|---|---|---|
| DAG Fusion [13] | 1× (10-bit) | 0 | 1× (14-bit) | 2×$Mux_{2\text{-to-}1}$(10-bit) 1×$Mux_{2\text{-to-}1}$(14-bit) | 2994 | 227.3 |
| ReMB [9] | 1×(13-bit) | 1×(11-bit) | 1×(13-bit) | 1×$Mux_{2\text{-to-}1}$(8-bit) 1×$Mux_{2\text{-to-}1}$(12-bit) | 3530 | 265.0 |
| ASAP [16] | 0 | 0 | 1× (10-bit) 1× (14-bit) | 1×$Mux_{2\text{-to-}1}$(10-bit) 1×$Mux_{2\text{-to-}1}$(14-bit) | 2996 | 224.0 |
| Proposed | 0 | 0 | 1× (10-bit) 1× (14-bit) | 1×$Mux_{2\text{-to-}1}$(10-bit) 1×$Mux_{2\text{-to-}1}$(11-bit) | 2940 | 217.0 |

TABLE VI
SYNTHESIZED AREAS OF DATA SETS A–D (IN SQUARE MICROMETERS)

| Data sets | Algorithms | RCA | Improvement | CLA | Improvement |
|---|---|---|---|---|---|
| A | ReMB | 17360.48 | 62.67% | 18108.92 | 61.00% |
| | DAGfusion | 7477.75 | 13.35% | 8212.88 | 14.01% |
| | Proposed | 6479.83 | | 7061.95 | |
| B | ReMB | 11991.68 | –0.72% | 12520.57 | –2.47% |
| | DAGfusion | 11901.86 | –1.48% | 12294.37 | –4.36% |
| | Proposed | 12078.16 | | 12829.92 | |
| C | ReMB | 9749.68 | 38.76% | 9972.55 | 34.82% |
| | DAGfusion | 7221.61 | 17.32% | 7693.96 | 15.52% |
| | Proposed | 5970.89 | | 6499.78 | |
| D | ReMB | 8086.48 | 28.14% | 8226.19 | 26.04% |
| | DAGfusion | 6127.23 | 5.16% | 6300.2 | 3.43% |
| | Proposed | 5811.22 | | 6083.99 | |

TABLE VII
SYNTHESIZED DELAYS OF DATA SETS A–D (IN NANOSECONDS)

| Data sets | Algorithms | RCA | CLA |
|---|---|---|---|
| A | ReMB | 9.25 | 6.06 |
| | DAGfusion | 6.65 | 4.45 |
| | Proposed | 6.72 | 4.6 |
| B | ReMB | 9.81 | 6.06 |
| | DAGfusion | 9.15 | 6.49 |
| | Proposed | 8.36 | 5.89 |
| C | ReMB | 9.17 | 6.1 |
| | DAGfusion | 8.15 | 5.2 |
| | Proposed | 7.27 | 4.99 |
| D | ReMB | 7.61 | 4.69 |
| | DAGfusion | 7.95 | 4.79 |
| | Proposed | 7.85 | 4.78 |

the bitwidths of the arithmetic operators and multiplexers of the proposed designs are always smaller than those of [13]. In addition, the proposed multiplexer splitting also helps. For example, in Table II, instead of having a 16-b seven-to-one multiplexer [13], our proposed algorithm divides the large fan-in multiplexer into multiplexers of lower fan-in and bitwidth. The improved scheduling method also outperforms ASAP in data sets B and D.

To corroborate the merits of our proposed algorithm, the designs are technology mapped to TSMC 0.18-$\mu$m standard cell library using Synopsys Design Compiler. The synthesized areas and delays are tabulated in Tables VI and VII, respectively. Ripple carry adders (RCAs) and carry look-ahead adders (CLAs) are used to demonstrate the two different scenarios of using the faster but more costly adders versus using the slower but simpler adders for the RCM design under the same timing constraint. Due to the multiplexer overheads for reconfigurability

TABLE VIII
AT PRODUCTS OF DATA SETS A–D (IN SQUARE MICROMETERS × NANOSECONDS)

| Data sets | Algorithms | RCA | Improvement | CLA | Improvement |
|---|---|---|---|---|---|
| A | ReMB | 160584.4 | 72.88% | 109740 | 70.40% |
| | DAGfusion | 49727.04 | 12.43% | 36547.32 | 11.12% |
| | Proposed | 43544.46 | | 32484.97 | |
| B | ReMB | 117638.4 | 14.17% | 75874.65 | 0.40% |
| | DAGfusion | 108902 | 7.28% | 79790.46 | 5.29% |
| | Proposed | 100973.4 | | 75568.23 | |
| C | ReMB | 89404.57 | 51.45% | 60832.56 | 46.68% |
| | DAGfusion | 58856.12 | 26.25% | 40008.59 | 18.93% |
| | Proposed | 43408.37 | | 32433.9 | |
| D | ReMB | 61538.11 | 25.87% | 38580.83 | 24.62% |
| | DAGfusion | 47811.48 | 4.59% | 30177.96 | 3.63% |
| | Proposed | 45618.08 | | 29081.47 | |

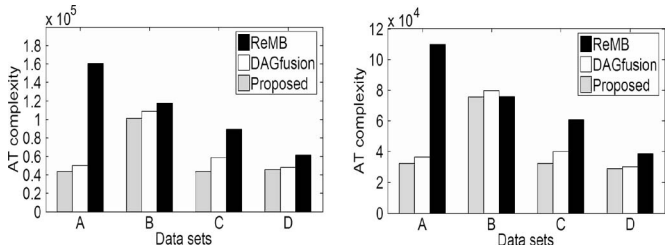



Fig. 11. AT complexity comparison of proposed RCM, ReMB methods, and [13] with (left) RCA and (right) CLA adders for data sets A–D.

and the reduced number of adders, the overall area increment caused by the use of CLA over RCA is not as significant.

The area and delay values are combined into an overall AT complexity evaluation. The AT product of each design is shown in Table VIII. The results demonstrate the effectiveness of our proposed algorithm in reducing both the critical path delay and the logic area for different RCM designs. On average, our proposed algorithm generates RCM designs of 41.09% and 35.53% lower AT complexities than ReMB with RCA and CLA implementations, respectively. Comparing with DAGfusion, our AT products are 12.63% and 9.74% lower with RCA and with CLA implementations, respectively. The AT complexity comparisons are shown in Fig. 11. The reduction of AT product is attributed in part to the CSE using MSD representation and in part to the minimization of average adder disparity in each control step. The balanced adder graph after CSE and the reduction in adder/subtractor and multiplexer bitwidths play a crucial role in shortening the critical path delay.

The RCM of the $2 \times \alpha$ multipliers of type A Goertzel recursive DCT structure in [25] are implemented for two different DCT transform lengths $N = 8$ and $N = 16$. The coefficients are generated in three levels of precision $w = 8, 12,$ and 16 b. Every design is implemented with two types of adders, RCA and CLA. The synthesis results are presented in Table IX. Since [13] is the latest and most relevant method for ASIC implementation, the AT product is compared with that in [13] in Fig. 12. On average, the proposed algorithm reduces the AT product by 8.51% and 10.60% over DAG fusion method [13] based on RCA and CLA, respectively. The AT products are also evaluated with the constant sets from seven FIR filters in Table X. The results show that the proposed method is able to generate more area–time efficient solutions in general.

TABLE IX
SYNTHESIZED AREA (A) (IN SQUARE MICROMETERS), DELAY (T) (IN NANOSECONDS), AND AT PRODUCT (IN SQUARE MICROMETERS × NANOSECONDS) OF $2 \times \alpha$ MULTIPLIERS IN 8-POINT/16-POINT TYPE A GOERTZEL RECURSIVE DCT STRUCTURES

| DCT attributes | Algorithms | RCA | | | CLA | | |
|---|---|---|---|---|---|---|---|
| | | A | T | AT | A | T | AT |
| 8-point, $w$=8 | DAGfusion | 10854.0 | 7.3 | 79668.7 | 11446.1 | 5.8 | 66731.0 |
| | Proposed | 10824.1 | 7.6 | 82696.2 | 11592.5 | 5.1 | 58542.1 |
| 8-point, $w$=12 | DAGfusion | 15863.6 | 10.3 | 163395.1 | 16914.7 | 6.9 | 116880.9 |
| | Proposed | 16349.3 | 9.3 | 151394.1 | 17433.7 | 6.8 | 117677.2 |
| 8-point, $w$=16 | DAGfusion | 25985.5 | 14.8 | 385369.9 | 28044.9 | 8.8 | 246794.9 |
| | Proposed | 24811.1 | 12.6 | 313370.7 | 27163.4 | 7.3 | 198835.9 |
| 16-point, $w$=8 | DAGfusion | 13771.3 | 9.5 | 130827.3 | 14463.2 | 6.8 | 98349.7 |
| | Proposed | 10787.5 | 7.9 | 85221.4 | 11422.9 | 5.5 | 62825.7 |
| 16-point, $w$=12 | DAGfusion | 21298.9 | 11.1 | 235353.3 | 22463.2 | 7.2 | 161285.6 |
| | Proposed | 23830.3 | 11.6 | 276908.4 | 25340.5 | 7.4 | 186252.8 |
| 16-point, $w$=16 | DAGfusion | 35239.9 | 14.6 | 514502.3 | 37405.4 | 9.1 | 339640.7 |
| | Proposed | 34554.6 | 13.2 | 454739.1 | 37804.5 | 7.9 | 299033.9 |

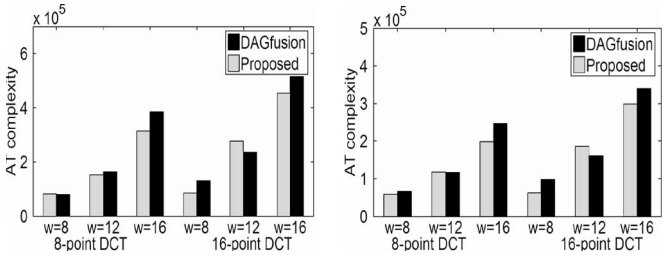



Fig. 12. AT complexity comparison of proposed RCM and [13] for type A Goertzel recursive DCT with (left) RCA and (right) CLA adders.

TABLE X
AT PRODUCTS OF FIR FILTER COEFFICIENT SETS (IN SQUARE MICROMETERS × NANOSECONDS)

| FIR filter | Algorithms | AT product (RCA) | AT product (CLA) |
|---|---|---|---|
| FIR1 [36] | DAGfusion | 26490.7 | 19084.7 |
| | Proposed | 18087.1 | 12793.4 |
| FIR2 [37] | DAGfusion | 17195.4 | 13336.5 |
| | Proposed | 15497.5 | 11809.4 |
| FIR3 [38] | DAGfusion | 19208.2 | 14389.7 |
| | Proposed | 14900.6 | 11357.7 |
| FIR4 [39] | DAGfusion | 103410.2 | 80104.6 |
| | Proposed | 111724.1 | 74093.8 |
| FIR5 [40] | DAGfusion | 106368.3 | 76512.9 |
| | Proposed | 95732.5 | 64922.0 |
| FIR6 [38] | DAGfusion | 97721.1 | 65528.4 |
| | Proposed | 103662.9 | 67406.6 |
| FIR7 [41] | DAGfusion | 31688.2 | 21375.8 |
| | Proposed | 29929.8 | 21205.8 |

The proposed algorithm is further compared against DAGfusion algorithm with the help of the online tool [42] using large sets of randomly generated constants. The number of constants in each set, $N = 10$ and 20, and the wordlength of the constants, $w = 8, 10$ and 12 b. The average AT products with different $N$ and $w$ are shown in Fig. 13.

From Fig. 13, the average reduction of area–time cost of our proposed algorithm over DAGfusion algorithm is lower (about 4%) for $w \leq 10$ and more significant (up to 25%) for $w = 12$. This is because by scheduling mobile operators based on adders disparity features, mutually exclusive mobile adders are more likely to be scheduled into the same control step and merged by the proposed algorithm. Due to the inclination to merge

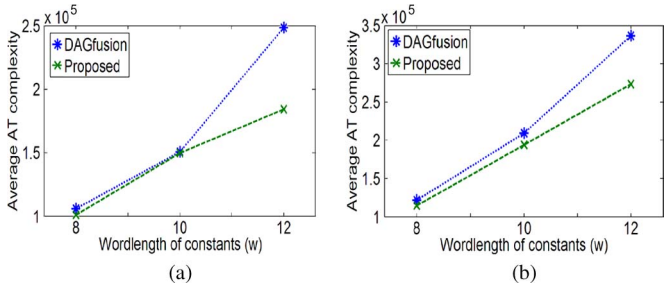



Fig. 13. Comparison of average AT products on randomly generated constant sets. (a) $N = 10$. (b) $N = 20$.

adders of similar disparity values, the merged adders are more likely to have smaller adder width. In addition, the proposed multiplexer splitting based on multiplexer disparities has also helped to reduce the fan-in of large bitwidth multiplexers.

Aside from the solution quality, computational effort is another important criterion for design automation algorithms. The complexity of our proposed algorithm is dominated by the scheduling decisions of the mobile adders. The computational complexity of our algorithm can be evaluated as follows.

Let $L(v)$ be the wordlength of $a(v)$ for an arbitrary node, $v \in V$ of the DAG, $G(V, E)$. It is well known that MSD reduces the number of nonzero digits of an integer by 33.3% on average from its binary representation [43]. Assuming an equal probability of "1" and "0" b in a binary number, the expected number of SDs in $a(v)$ is $\hat{h}(v) = (1 - 1/3) \times 1/2 \times L(v) = 1/3 \times L(v)$. The average number of iterations in the computation of $W(v)$ is equal to $\hat{h}(v) - 1$. Hence, the complexity for the aggregate distance computation is $O(|V| \times L(v))$.

Since only one mobile node is scheduled in each iteration, the total number of iterations required to compute the disparity and opportunity cost is equal to the number of mobile nodes. In each iteration, the number of operations required to compute the disparity of a mobile node can be enumerated by the total number of fixed nodes in all its legitimate control steps. In the worst case, every node in the DFG is mobile initially and can be scheduled into any control step. $|V|$ iterations are required. The number of fixed nodes is zero in the first iteration and incremented by one in each iteration until all the $|V|$ mobile nodes become fixed in the last iteration. The number of operations is given by the sum of an arithmetic progression of $|V|$ terms with a common difference of one starting from zero. This sum is equal to $|V|/2 \times (|V| - 1)$. The complexity for the disparity computation is thus $O(|V|^2)$.

From (9), the opportunity cost is computed by summing the product of the disparity and probability displacement over all legitimate control steps. Each iteration requires at most $t_{\max}$ multiplications and $t_{\max} - 1$ additions, where $t_{\max}$ is the number of control steps. Let $H$ be the maximum hamming weight of all constants. Then, $t_{\max} = \lceil \log_2(H) \rceil$ if we assume that $t_{\max} = D(G)$. The number of operations required for the computation of opportunity cost is given by $|V| \times (2\lceil \log_2(H) \rceil - 1)$. It should be noted that the hamming weights, $H$ of the constants in MSD representation is usually less than ten, The complexity of opportunity cost computation is $O(|V|)$.

A small fraction of nondeterministic computations due to the opportunity costs of predecessor and/or successor nodes and the

update of mobilities can be omitted since the aggregate distance and disparity computations of all nodes required for their evaluation have been conservatively included in the aforementioned analysis. The total computational complexity of proposed algorithm is $O(|V| \times L(v) + |V|^2 + |V|)$. In practice, $L(v)$ is finite and can be safely assumed to be lower than 20. Hence, the computational complexity of our algorithm is $O(|V|^2)$.

This complexity is far less than the brute force approach of DAG fusion which searches exhaustively for all possible pairs of DAGs for the best fusion. The runtime for fusing $N$ DAGs is given by $O(\text{Num\_iterations} \times N \times \text{Runtime(FusePairDAGs)})$, where $\text{Runtime(FusePairDAGs)} = O(n!(mN+n)/\lceil \log_2(m+1) \rceil !(n-m)!)$ and Num_iterations is the number of different orderings of fusing a pair of DAGs, with $n$ and $m$ being the numbers of adders in the first and second DAGs, respectively [13]. The first DAG is obtained iteratively by the previous fusion of $N$ DAGs, thus $n$ is comparable to $m$ and in the worst case, $m = n$. The maximum Num_iterations for $N$ DAGs is $N!$ [13]. Limiting Num_iterations to reduce the runtime will compromise the solution quality. The overall complexity can be approximated to $O(N^2 N! n!)$. This is much higher than our overall runtime complexity of $O(N^2 n^2)$ if the $|V|$ adders are amortized over the $N$ constants so that the average number of adders contributed by each DAG is $n$.

## V. Conclusion

For many general DSP problems, the need for multiple constant multiplications is mandatory. This paper presents a new insight into the design of RCMs where the adder resources can be time multiplexed to reduce the hardware cost. Comparing with existing design methodologies which are mostly based on the reduced adder graph or matured techniques in the design of the multiplier block of digital filters, the concept conceived in this paper is new and unique. It is the first proposal that delineates the RCM architecture as a DFG and solves this design problem by an efficient heuristic scheduling algorithm. A disparity measure is defined to assess the cost of merging two or more temporally correlated adders by evaluating the compactness of SD distribution among mutually exclusive nodes in the same control step. This is used in conjunction with their probability displacement to determine the opportunity cost of scheduling a mobile adder into a control step, from which an optimized scheduling decision is made. Since the depth of the adder tree is first minimized by a global CSE, the RCM resulting from the scheduled DFG is minimized in the area $\times$ delay sense. Our synthesis results on some practical RCMs and randomly generated sets of constants show that the proposed algorithm produces solutions which are up to 25% more area–time efficient than the latest and best reported solutions known in the literature for 12-b constant sets.

## References


[1] M. Potkonjak, M. B. Srivastava, and A. P. 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.

[2] A. DeHon, "The density advantage of configurable computing," *Computer*, vol. 33, no. 4, pp. 41–49, Apr. 2000.

[3] G. Estrin, "Reconfigurable computer origins: The UCLA fixed-plus-variable computer," *IEEE Ann. Hist. Comput.*, vol. 24, no. 4, pp. 3–9, Oct.–Dec. 2002.

[4] J. Macbeth and P. Lysaght, "Dynamically reconfigurable cores," in *Proc. Int. Conf. Field-Programmable Logic Appl.*, 2001, vol. 2147, pp. 462–472.

[5] J. P. Heron, R. Woods, S. Sezer, and R. H. Turner, "Development of a run-time reconfiguration system with low reconfiguration overhead," *J. VLSI Signal Process. Syst.*, vol. 28, no. 1/2, pp. 97–113, May/Jun. 2001.

[6] S. S. Demirsoy, A. G. Dempster, and I. Kale, "Design guidelines for reconfigurable multiplier blocks," in *Proc. IEEE Int. Symp. Circuits Syst.*, Bangkok, Thailand, May 25–28, 2003, pp. 293–296.

[7] S. S. Demirsoy, I. Kale, and A. G. Dempster, "Efficient implementation of digital filters using novel reconfigurable multiplier blocks," in *Proc. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 7–10, 2004, pp. 461–464.

[8] S. S. Demirsoy, I. Kale, and A. G. Dempster, "Synthesis of reconfigurable multiplier blocks—Part I: Fundamentals," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, May 23–26, 2005, vol. 1, pp. 536–539.

[9] S. S. Demirsoy, I. Kale, and A. G. Dempster, "Synthesis of reconfigurable multiplier blocks—Part II: Algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, May 23–26, 2005, vol. 1, pp. 540–543.

[10] N. Sidahao, G. A. Constantinides, and P. Y. K. Cheung, "Multiple restricted multiplication," in *Proc. Int. Conf. Field-Programmable Logic Appl.*, Aug. 2004, pp. 374–383.

[11] S. S. Demirsoy, I. Kale, and A. Dempster, "Reconfigurable multiplier blocks: Structures, algorithms and applications," *Circuits Syst. Signal Process.*, vol. 26, no. 6, pp. 793–827, Dec. 2007.

[12] Y. E. Kim, K. J. Cho, and J. G. Chung, "Low power small area modified booth multiplier design for predetermined coefficients," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. E90-A, no. 3, pp. 694–697, Mar. 2007.

[13] P. Tummeltshammer, J. Hoe, and M. Püschel, "Time-multiplexed multiple-constant multiplication," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 9, pp. 1551–1563, Sep. 2007.

[14] O. Gustafsson, A. G. Dempster, and L. Wanhammar, "Extended results for minimum-adder constant integer multipliers," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2002, vol. 1, pp. I-73–I-76.

[15] O. Gustafsson, A. G. Dempster, K. Johansson, M. D. Macleod, and L. Wanhammar, "Simplified design of constant coefficient multipliers," *Circuits Syst. Signal Process.*, vol. 25, no. 2, pp. 225–251, Apr. 2006.

[16] J. Chen, C. H. Chang, and C. C. Jong, "Time-multiplexed data flow graph for the design of configurable multiplier block," in *Proc. IEEE Int. Symp. Circuits Syst.*, Taipei, Taiwan, May 2009, pp. 1145–1148.

[17] P. Flores, J. Monteiro, and E. Coista, "An exact algorithm for the maximum sharing of partial terms in multiple constant multiplications," in *Proc. IEEE Int. Conf. Comput.-Aided Des.*, Nov. 6–10, 2005, pp. 13–16.

[18] L. Aksoy, E. da 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.

[19] K. Johansson, O. Gustafsson, and L. Wanhammar, "A detailed complexity model for multiple constant multiplication and an algorithm to minimize the complexity," in *Proc. Eur. Conf. Circuit Theory Des.*, Cork, Ireland, Aug. 28–Sep. 2, 2005, pp. 465–468.

[20] R. Mahesh and A. P. Vinod, "A new common subexpression elimination algorithm for realizing low-complexity higher order digital filters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 2, pp. 217–229, Feb. 2008.

[21] M. D. Macleod and A. G. Dempster, "Common subexpression elimination algorithm for low-cost multiplierless implementation of matrix multipliers," *Electron. Lett.*, vol. 40, no. 11, pp. 651–652, May 2004.

[22] K. Wu and R. Karri, "Algorithm-level computing with shifted operands—A register transfer level concurrent error detection technique," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 3, pp. 413–422, Mar. 2006.

[23] C. Chao and K. K. Parhi, "High-throughput VLSI architecture for FFT computation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 10, pp. 863–867, Oct. 2007.

[24] Y. Shao and C. H. Chang, "A Kalman filter based on wavelet filter-bank and psychoacoustic modeling for speech enhancement," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kos, Greece, May 21–24, 2006, pp. 121–124.

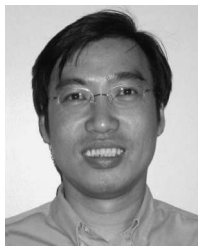
[25] S. S. Demirsoy, R. Beck, A. G. Dempster, and I. Kale, "Reconfigurable implementation of recursive DCT kernels for reduced quantization noise," in *Proc. IEEE Int. Symp. Circuits Syst.*, Bangkok, Thailand, May 25–28, 2003, pp. 289–292.

[26] P. Vaidyanathan, *Multirate Systems and Filter Banks*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[27] S. Arno and F. S. Wheeler, "Signed digit representations of minimal hamming weight," *IEEE Trans. Comput.*, vol. 42, no. 8, pp. 1007–1010, Aug. 1993.
[28] A. G. Dempster and M. D. Macleod, "Generation of signed-digit representations for integer multiplication," *IEEE Signal Process. Lett.*, vol. 11, no. 5, pp. 663–665, Aug. 2004.
[29] R. Paško, P. Schaumont, V. Derudder, S. Vernalde, and D. Ďuračkovâ, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 18, no. 1, pp. 58–68, Jan. 1999.
[30] M. M. Peiro, E. I. Boemo, and L. Wanhammar, "Design of high-speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 49, no. 3, pp. 196–203, Mar. 2002.
[31] F. Xu, C. H. Chang, and C. C. Jong, "Design of low-complexity FIR filters based on signed-powers-of-two coefficients with reusable common subexpressions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 10, pp. 1898–1907, Oct. 2007.
[32] F. Xu, C. H. Chang, and C. C. Jong, "Contention resolution algorithm for common subexpression elimination in digital filter design," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 52, no. 10, pp. 695–700, Oct. 2005.
[33] I. C. Park and H. J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 21, no. 12, pp. 1525–1529, Dec. 2002.
[34] C. H. Chang, J. Chen, and A. P. Vinod, "Information theoretic approach to complexity reduction of FIR filter design," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 8, pp. 2310–2321, Sep. 2008.
[35] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA: Morgan Kaufmann, 1999.
[36] R. Jain, P. T. Yang, and T. Yoshino, "FIRGEN: A computer-aided design system for high performance FIR filter integrated circuits," *IEEE Trans. Signal Process.*, vol. 39, no. 7, pp. 1655–1668, Jul. 1991.
[37] Y. H. Jang and S. J. Yang, "Low-power CSD linear phase FIR filter structure using vertical common sub-expression," *Electron. Lett.*, vol. 38, no. 15, pp. 777–779, Jul. 2002.
[38] A. P. Vinod, E. M. Lai, A. B. Premkuntar, and C. T. Lau, "FIR filter implementation by efficient sharing of horizontal and vertical common subexpressions," *Electron. Lett.*, vol. 39, no. 2, pp. 251–253, Jan. 2003.
[39] Q. Zhao and Y. Tadokoro, "A simple design of FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 566–670, May 1988.
[40] J. Laskowski and H. Samueli, "A 150-MHz 43-tap half-band FIR digital filter in 1.2-um CMOS generated by silicon compiler," in *Proc. IEEE Custom Integr. Circuits Conf.*, Boston, MA, May 1992, pp. 11.4.1–11.4.4.
[41] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. Circuits Syst.*, vol. CAS-30, no. 10, pp. 723–739, Oct. 1983.
[42] Spiral Project: Software/Hardware Generation for DSP Algorithms. [Online]. Available: http://www.spiral.net
[43] R. W. Reitwiesner, "Binary arithmetic," in *Advances in Computers*, vol. 1. New York: Academic, 1960, pp. 231–308.

**Jiajia Chen** received the B.Eng. degree from Nanyang Technological University, Singapore, in 2004.

From 2004 to 2008, he joined the School of Electrical and Electronic Engineering, Nanyang Technological University, as a Teaching Assistant while working toward the Ph.D. degree. He is currently a Research Engineer with 3M Singapore Pte. Ltd., Singapore. His main research interest includes computational transformations of low-complexity digital filters, reconfigurable filters, and filter architectural optimization.



**Chip-Hong Chang** (S'92–M'98–SM'03) received the B.Eng.(Hons.) degree from National University of Singapore, Singapore, in 1989 and the M.Eng. and Ph.D. degrees from Nanyang Technological University (NTU), Singapore, in 1993 and 1998, respectively.

He served as a Technical Consultant in industry prior to joining the School of Electrical and Electronic Engineering (EEE), NTU, in 1999, where he is currently an Associate Professor. He is the holder of joint appointments at the university as Assistant Chair of Alumni, School of EEE since June 2008, Deputy Director of the Centre for High Performance Embedded Systems since 2000, and Program Director of the Centre for Integrated Circuits and Systems since 2003. He has published three book chapters and more than 140 research papers in refereed international journals and conferences. His current research interests include low-power arithmetic circuits, digital filter design, application specific digital signal processing, and digital watermarking for IP protection.

Dr. Chang is a Fellow of the Institution of Engineering and Technology. He serves as an Editorial Advisory Board member of The Open EEE Journal and the Journal of Electrical and Computer Engineering. He is listed in the Marquis Who's Who in the World since 2008 and is appointed the Charter Fellow of Advisory Directorate International by the American Biographical Institute, Inc.