EFFICIENT ALGORITHMS FOR COMMON SUBEXPRESSION ELIMINATION IN DIGITAL FILTER DESIGN

Fei Xu, Chip-Hong Chang and Ching-Chuen Jong

Centre for Integrated Circuits and Systems, Nanyang Technological University
School of EEE, Nanyang Avenue, Singapore 639798

ABSTRACT

In this paper, a contention resolution algorithm (CRA) is proposed for the common subexpression elimination of the multiplier block of digital filter structure. CRA synthesizes common subexpressions of any hamming weight to achieve an overall minimization with the emphasis that every logic depth increment must be accompanied by a reduction in logic complexity. A new data structure, called the admissibility graph is introduced to succinctly represent a set of coefficients and the admissible subexpressions are progressively labeled on the graph as either precedence or contention edges (or paths). The performance of CRA is evaluated based on benchmarked circuits and randomly generated coefficients. It is demonstrated that our algorithm outperforms several distinguished algorithms in both the logic depth and logic complexity.

1. INTRODUCTION

The decomposition of the multiplication of a variable by a set of constants into a multiplierless shift-and-add block has been a core operation and often performance bottleneck in many digital signal processing applications including FIR filters, IIR filters, Farrow structure interpolators and filter banks [3]. High throughput rate, low power dissipation and increased programmability are among the benchmark performance criteria. Dedicated hardware optimized for a fixed coefficient set meets the stringent goals of sampling rate and power budget at reduced flexibility and scalability. For applications that demand expandable, single-platform, multiservices flexibility, the emergence of fast reconfigurable Field Programmable Gate Arrays (FPGAs) has made the run time reconfigurable Application Specific Integrated Circuit (ASIC) a reality and choice technology for high-speed digital filters. Both design approaches require the automation techniques to perform combinatorial optimization of formidable complexity with drastically different emphasis than software compilers for the general purpose DSPs. It is therefore imperative to derive optimization methods that facilitate efficient automation in the compilation of silicon solutions and form the basis for high-level synthesis tools targeted at the next-generation ASICs.

By replacing individual coefficient multipliers by a single multiplier block, the core operation of the transpose direct form FIR filter can be modeled as a general Multiple Constant Multiplications (MCM) problem [6, 8]. The efficiency in reducing the cost metric (power, hardware, and logic depth) of the MCM operation directly influences the performance of the ASIC implementation of FIR filters. Due to the complexity of the MCM problem defined in [8], there is no exact algorithm that guarantees optimality of the solutions for all kinds of coefficients. Existing common subexpression elimination (CSE) algorithms are based on heuristic search techniques or combined exhaustive search with steepest descent approach [1, 2, 6-8] to select the common subexpressions for elimination. These algorithms suffer from a common problem that once a subexpression is identified as the most profitable common subexpression, the decision cannot be reverted. Each candidate subexpression is independently selected based on some cost function to form the final solution progressively. The gross effect of interdependency of coefficients to the cost of implementation is not considered.

In this paper, we proposed a drastically different graph based technique to the formulation of CSE algorithms for the MCM problems. A generalized contention resolution algorithm featuring the ability to change the precedence of subexpressions to resolve the local minima problem is proposed. It extends the basic CRA of [9] which eliminates only weight two common subexpressions. The generalized CRA not only reduces the logic complexity but also maximizes the performance by reducing the logic depth. Reducing the logic depth has the added advantage that shallow taps shorten the propagation paths of spurious switching activities through long chain of adders, thus lowering the power dissipation.

2. ADMISSIBILITY GRAPH

A simple admissibility graph, $G(V, E)$ is proposed to describe a filter coefficient where the vertex set V is formed by the nonzero digits of the coefficient and the edge set E is a collection of all subexpressions of weight two generated from the coefficient. An edge $e = (u, v)$ is a connection of two vertices u and v. A pair of edges incident with a common vertex are said to be adjacent. A FIR filter can be modeled as a union of admissibility graphs, i.e., $G(V,E)=\bigcup_{i=1}^{N} G_i(V_i,E_i)$ where N is the total number of coefficients, and $G_i(V_i, E_i)$ is an admissibility graph for the i-th coefficient. Canonical Signed Digit (CSD) form is used to represent the filter coefficients. Each vertex, v is associated with a unique positional index, $index(v)$, determined by its position in the CSD form. Two types of vertices are defined, the signed and unsigned vertices correspond to the digits $\bar{1}$ (–1) and 1, respectively. All vertices in the admissibility graph are set free initially. A precedence edge, e_p is used to link two free vertices. Once a vertex is connected by a precedent edge, it becomes fixed. Two vertices are linked via a contention edge e_c if at least one vertex is fixed. A precedence edge is used to commit a subexpression into the optimal common subexpression list and a contention edge is used to indicate subexpression that is denied admission as an optimal common subexpression by another subexpression that has already been committed. In other words, each contention edge implies a conflict (contention) between two

common subexpressions, as there exist adjacent edges of different precedences. Selection of one of which into the final common subexpression list will automatically nullify the admissibility of the other. Each edge, e is given an order of precedence, $order(e)$ according to the number of occurrences (henceforth refer to as the frequency) of its subexpression, with the lowest value of $order(e)$ signifying the highest frequency. Two edges are said to be equivalent if they have the same order. Equivalent edges are common subexpressions. Two subexpressions are common if one can be obtained from another without incurring adder costs. A path is a series of connected edges. Two paths $\eta_1 = e_1, e_2 \ldots e_{w-1}$ and $\eta_2 = e_1', e_2', \ldots e_{w-1}'$ are said to be equivalent, if they have the same weight and all their corresponding edges are equivalent, i.e., $order(e_i) = order(e_i')\ \forall i = 1, 2, .., w-1$.

Fig. 1 shows a 5-vertex admissibility graph for the coefficient $C_1 = \bar{1}00\bar{1}010\bar{1}001$. The unsigned and signed vertices are drawn as solid and hollow circles, respectively. The precedence and contention edges are drawn with solid and dash lines, respectively. Each vertex is annotated with its positions, and each edge is annotated with its order of precedence. This admissibility graph features one precedence edge, cd of order 1 and three contention edges, bc, ce and de of order 1, 2 and 4 respectively. Vertices a, b and e are free vertices.

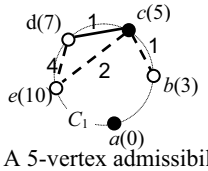


Figure 1. A 5-vertex admissibility graph.

3. CONTENTION RESOLUTION ALGORITHM (CRA)

The proposed CRA has the ability to rectify poor decisions made in earlier iterations or postpone ambiguous decisions to later stages. The rigidity of the search for common subexpressions has been relaxed by allowing the earlier chosen candidate subexpressions marked by the precedence paths to be substituted by the adjacent contention paths determined by a specific cost metric. The final solution is generated at the last stage when all the major contentions have been resolved. The algorithm considers subexpressions of different weights. Unlike other existing algorithms, CRA increases the logic depth only when it leads to a saving in the adder cost.

Let L be the set of unique CSD coefficients that have more than one nonzero digits. The essential steps involved in the CRA are given as follows:

(1) The coefficients in L are decomposed into subexpressions of hamming weight two. These subexpressions are sorted in descending order of frequency to form an order list S. An order number is assigned to each subexpression in S such that the subexpression with higher frequency has lower order number and common subexpressions have the same order. Set $j = 1$ and generate the free vertices of the primitive admissibility graphs, G_i from the nonzero digits of the coefficients C_i of L.

(2) Label the subexpressions with order = j on G with precedence or contention edges accordingly. Update the list of precedence and contention edges, E_P and E_C.

(3) Generate precedence paths (weight > 2) by expanding existing edges or paths. Two types of simple candidate path are considered, one is the hybrid path comprising a precedence edge or path adjacent to a contention edge, and the other is the contention path comprising two adjacent contention edges. The pattern graph being considered consists of the candidate path and its adjacent edges. Calculate the difference in adder costs, R in changing the candidate paths, η_c to precedence paths and the status of their affected adjacent edges. If $R > 0$, η_c will be promoted to precedence paths, with the status of their affected adjacent edges in the pattern graph changed to resolve any conflict. Insert the resulting precedence paths η in P_P and update E_P and E_C. Sort the list P_P by the path weights and frequencies, assigned order numbers as for S but with the order numbers lower than the smallest order number of S.

(4) Search for precedence edges which are adjacent to contention edge(s) that have a free endpoint. Calculate the cost difference, R of swapping the status of the precedence edge with the contention edge(s) based on a simple pattern graph comprising a precedence edge and at most two adjacent contention edges (one from each endpoint) and their equivalent edges in E_P. If $R > 0$, transfer the precedence edge, e_p from E_P to E_C and the contention edge(s), e_c from E_C to E_P.

(5) Increment j and repeat Step 2 to 4 until the expressions in S are exhausted.

(6) Empty E_C. Traverse all precedence edges and paths (represent partial sums) in E_P and P_P by a Hamiltonian path. The inserted links (represent adders to sum the partial sums) are stored in E_C.

In Steps 3 and 4, R is equal to the difference in the adder costs between G_1 and G_2, where G_1 and G_2 are the subgraphs comprising the candidate path (for Step 3) or precedence edge (for Step 4) and its emanating edges before and after the conversion, respectively. In evaluating R, each additional contention edge generated by the conversion induces an adder. It should be noted that we will not break an already formed precedence path, η to create a hybrid path for further optimization in future iterations as this will introduce an adder if its frequency, $f(\eta) > 2$. Also, if $f(\eta) = 2$, changing it to a hybrid path will cause the remaining equivalent path to become hybrid as well, which incurs one more adder. From rigorous experimental simulation, it has been found that the probability of such decomposition of precedence paths that leads to further cost savings is low and often the savings, if any, are insignificant. Therefore, we consider only those edges emanating from the precedence edge or candidate path that have their attributes changed after the conversion. Let Δe_p be the number of precedence edges in G_1 that have been changed to contention edges in G_2. If $n(e_c)$ is the number of contention edges of the candidate path η_c, then the differential adder cost is

$$R(\eta_c) = n(e_c) - \Delta e_p \quad (1)$$

Equivalent candidate paths with positive R will be converted to precedence paths by promoting their contention edges to precedence edges if the aggregate saving is positive, i.e.,

$$\sum_{i=1}^{|\eta_c|} R_i(\eta_i) > 0 \quad (2)$$

where η_c is the set of equivalent candidate paths that have positive R.

The total number of adders required can be estimated from the final Hamiltonian path. Let P_w be the set of distinct order precedence paths of weight w. Further, let $C(\eta_w)$ denote the number of adders required to construct the precedence path, $\eta_w \in P_w$. A partial sum represented by a weight w path, η may require from zero to $w-1$ adders to construct depending on (i)

whether there exists an equivalent path of weight w or (ii) whether there exist other lower weight paths equivalent to some segments of η. Using the above notations, the formula for estimating the adder cost can be expressed as:

$$A = \sum_{w=2}^{\max(w)} C(\eta_w) + |E_C| \tag{3}$$

Due to the hidden vertex problem caused by the order of summing the vertices of a precedence path, the first term in (3) is implementation dependent. Since it is not likely to provide a closed form estimation of the exact minimal logic depth until the implementation of the precedence paths is fixed, the upper bound of the logic depth, D is given by.

$$D = \max\left(D_1, \max_{i=0}^{|\ell|} D_i + 1\right) \tag{4}$$

$$D_i = \left\lceil \log_2\left(|V_{free}| + \sum_{\eta} 1\right)\right\rceil + \max(w(\eta)) - 1$$

where $|V_{free}|$ is the number of free vertices and $w(\eta)$ is the weight of path η in the solution graph.

Example: Consider the filter F1 of $L = \{C_1, C_2, C_3\} = \{\bar{1}00\bar{1}010\bar{1}001, 10\bar{1}00100\bar{1}0\bar{1}, 1010\bar{1}00\bar{1}000\}$. The traces of the execution of CRA algorithm are shown in Fig. 2-4. There is no contention resolution required until the fifth iteration, $j = 5$. The equivalent candidate paths, $abc \equiv hij$ in Step 3 of the algorithm are marked with dotted boxes in Fig. 2. If both candidate paths are converted to precedence paths, only 10 adders are required to generate and sum all partial products of Fig. 2. Otherwise, 11 adders are required if they remain unchanged.

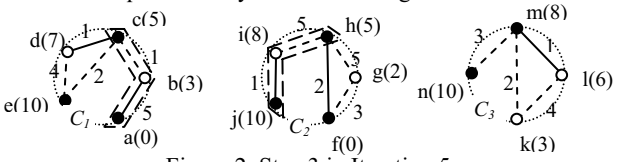


Figure 2. Step 3 in Iteration 5.

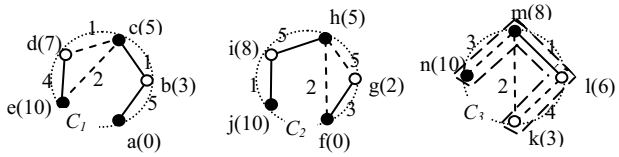


Figure 3. Step 4 in Iteration 5.

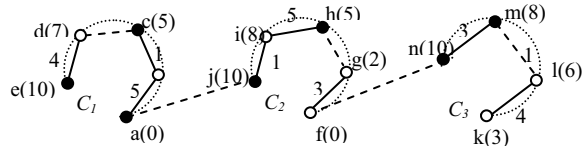
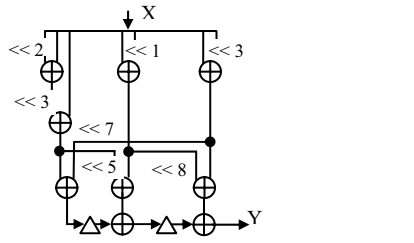


Figure 4. Final admissibility graph.



Δ Delay ⊕ Logic operator << n left shift by n bits

Figure 5. Circuit implementation.

After promoting abc and hij to P_P, the graph of Fig. 3 is obtained. Step 4 of the algorithm is evoked to change the precedence edge ml to contention edge and the contention edges mn and kl to precedence edges. This contention resolution results in another saving of an adder.

The final solution graph is shown in Fig. 4.The total number of adders used can be calculated from (3). Since $C(\eta_2) + C(\eta_3) = 4$ and $|EC| = 5$, $A = 9$ adders are required to implement F1. The circuit is shown in Fig. 5. It is noticed that Tap 2 is the critical path and the logic depth is 3.

4. EXPERIMENTAL RESULTS

The simulation results of the benchmark filters for the comparison of various algorithms are shown in Table 1. LO and LD are the acronyms of the number of logic operators and the logic depth, respectively. The benchmark filters and the algorithms being compared are annotated with the referenced sources. For those algorithms that are not implemented, their results are taken from [7] directly or filled with single dash lines if they are not available. For consistency, LO includes the accumulator associated with each filter tap. This accounts for the slight deviation in some of the values reported in the literature.

Table 1. Simulation results on FIR filters.

Filter	N	l	CSD		BHM[3]		Hartley[2]		ITM**[8]		Pasko[6]		NRSCSE[7]		CRA	
			LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	LD	LO	L
F1	3	1	13	5	11	6	10	4	12	4	9	4	10	3	**9**	**4**
F2[7]	4	8	15	4	9	7	10	3	11	4	10	4	9	3	**9**	**3**
F3[8]	4	1	18	4	11	7	13*	4	12	4	11	5	13	4	**11**	**5**
F4[6]	25	9	35	3	31	6	33*	3	34	3	30	3	30	3	**30**	**3**
F5[6]	59	1	11	3	-	-	99	4	-	-	90	3	90	3	**90**	**3**
F6[4]	12	1	26	4	-	-	203	4	-	-	178	5	188	4	**178**	**5**

The results show that CRA is capable of generating better solutions with lower number of logic operators than BHM at only half the logic depth or shorter. CRA does not guarantee its solutions to have the lowest logic depth, but in cases where it yields minimal logic depth solutions, the number of logic operators is also the least as observed from Table 1. In short, CRA has the best overall performance in view of its achievable logic depth for solutions generated with commensurate optimality of logic complexity.

Due to the unique evolution of the algorithm, a constraint on the logic depth of the solution can be imposed on CRA to suit the application if necessary. CRA(w_{max}) will generate only common subexpressions of hamming weight $w \le w_{max}$ where w_{max} is a preset maximum weight limit. Unlike the logic depth constrained CSE algorithm of [5], our logic depth driven CRA(w_{max}) incurs no overhead. Thus, it is of interest to scrutinize how the adder cost changes as w_{max} increases. 50 randomly generated coefficient sets for each combination of the number of taps (N) and coefficients' word lengh (l) are tested. The number of coefficient sets that have the minimum number of logic operators from each w_{max} is recorded and the percentage of the least adder cost solutions obtained from each w_{max} is listed in Table 2 for every combination of N and l. For example, for $N = 40$ and $l = 11$, only 2% of the coefficient sets have the least adder cost when $w_{max} = 2$. This means that for this particular coefficient set, no better result can be achieved by expanding the subexpressions. There exist filter sets that exhibit a critical turning point at some w_{max} beyond which the average adder cost will increase by relaxing the logic depth. This is an interesting phenomenon contrary to conventional belief and the experimental results in [5]. The number of coefficient sets that display this critical descend phenomenon for each combination

of N and l are accounted and its percentage is recorded in the last row of Table 2 labeled *%CDP*.

Table 2. Simulation results of CRA3 constraints.

l	10				20				30			
N	20	40	60	80	20	40	60	80	20	40	60	80
$w_{max} = 2$	14%	2%										
$w_{max} = 3$	64%	58%	42%	36%	28%	18%	10%	24%	28%	14%	6%	18%
$w_{max} = 4$	22%	40%	58%	64%	70%	58%	56%	46%	52%	56%	40%	38%
$w_{max} = 5$					2%	24%	32%	30%	16%	30%	52%	44%
$w_{max} = 6$							2%					
$w_{max} = 7$									4%			
%CDP		**4%**	**4%**	**2%**	**2%**	**6%**	**18%**	**18%**	**14%**	**10%**	**10%**	**24%**

From Table 2, everything being equal, the larger the coefficient set, the higher the weight limit to obtain the least adder cost solution and the longer the word length, the higher the weight limit is needed to achieve the best cost solution. The *%CDP* also increases as l and N increases. Like CRA, CRA(w_{max}) generates higher hamming weight subexpressions only when they lead to reduction in adder cost. From the *%CDP* of Table 2, we conjecture that longer (higher weight) subexpressions may not always foster adder cost reduction, as one would expect. The problem may occur when the elimination of longer common subexpressions prevent the more beneficial elimination of some shorter common subexpressions whose number of occurrences may be outweighing. Although the ways to obtain longer subexpressions differ from algorithm to algorithm, this local minimum exists, especially for CSE algorithms that start the search from the longest subexpressions. One possible suggestion to mitigate this critical descend problem is to apply edge resolutions first and then apply path resolutions to the resultant solution. We call this revised CRA algorithm as progressive CRA (P-CRA). P-CRA performs slightly better than CRA with an improved computational efficiency.

The MATLAB programs of CRA, CRA (w_{max} = 2), P-CRA, NRSCSE and Pasko's algorithms are run on a Pentium IV 1.9 GHz personal computer with 256MB of system memory. The average adder costs and the average computation time in seconds for filters with different number of taps at a fixed word length of 11 are simulated and the results are charted in Fig. 6. It is obvious that CRA and P-CRA outperform other algorithms being compared.

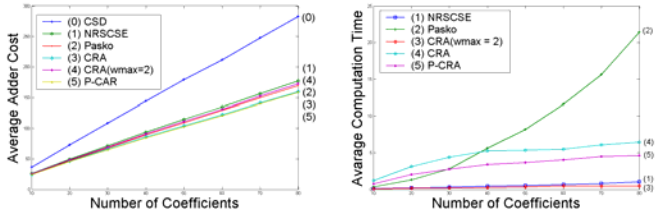


Figure 6. Comparison of average adder cost and computation time of CSD(0), NRSCSE(1), Pasko(2), CRA(3), CRA(w_{max}=2)(4) and P-CRA(5).

CRA and P-CRA require more computation time than Pasko's algorithm on small coefficient sets. This is because for smaller coefficient sets, more time is spent on the contention path resolution than the search for the most common subexpressions at each iteration. As the size of the sets increases, the computation time of Pasko exacerbates while that of other algorithms remain relatively constant. For CRA and P-CRA, the number of iterations is dependent on the maximum order number of the subexpression list, which is limited by the word length. As the word length is fixed, the iteration times are relatively stable. Therefore, the computational time increases only marginally as the coefficient set grows larger. CRAs generate common subexpression list only once at the very beginning, which save a lot of computational effort. P-CRA has effectively avoided some of the repetitive effort in comparison with CRA.

5. CONCLUSION

In this paper, a new CRA based on graph theoretic approach has been proposed to circumvent the local minimum problem encountered by the existing CSE algorithms. A novel admissibility graph is employed as an efficient data structure for the proposed algorithm, and this synthesis approach is unique in its fundamental concept of operations. The benefit of the algorithm is derived from its ability to appraise and substitute the chosen subexpressions when better alternatives emerge. Comparison with related work shows that our logic depth driven algorithm produces low cost circuits with shorter critical path delay. Moreover, our algorithm is also runtime efficient.

6. ACKNOWLEDGMENT

The authors would like to thank Panasonic Singapore Laboratory for their support and collaboration in this research.

7. REFERENCES

[1] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. on Circuits and Syst.- II*, vol. 42, no. 9, pp. 569-577, Sep. 1995.

[2] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. on Circuits and Syst.- II*, vol. 43, no. 10, pp. 677-688, Oct. 1996.

[3] R. A. Hawley, B. C. Wong, T. J. Lin, J. Laskowski and H. Samueli, "Design techniques for silicon compiler implementations of high-speed FIR digital filters," *IEEE J. of Solid-State Circuits*, vol. 31, no. 5, pp. 656-666, May 1996.

[4] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. on Circuits and Syst.*, vol. CAS-30, pp. 723–739, Oct. 1983.

[5] I. C. Park and H. J. Kang, "FIR filter synthesis algorithms for minimizing the delay and the number of adders," *IEEE Trans. on Circuits and Syst.- II*, vol. 48, no. 8, pp. 770-777, Aug. 2001.

[6] R. Paško, P. Schaumont, V. Derudder, S. Vernalde and D. Ďuračková, "A new algorithm for elimination of common subexpressions," *IEEE Trans. on Computer-Aided Design*, vol. 18, no. 1, pp. 58-68, Jan. 1999.

[7] M. M. Peiró, E. I. Boemo, and L. Wanhammar, "Design of high-Speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. on Circuit and Syst.*, vol. 49, no. 3, pp. 196-203, Mar. 2002.

[8] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplicaitons: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 2, pp. 151-165, Feb. 1996.

[9] F. Xu, C. H. Chang and C. C. Jong, "A new contention resolution algorithm for the design of minimal logic depth multiplierless filters," in *Proc. IEEE Int. Sypm. Circuits and Systems*, Vancouver, Canada, May, 2004 (Accepted).