

# Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters

Andrew G. Dempster, *Student Member, IEEE*, and Malcolm D. Macleod, *Member, IEEE*

**Abstract**—The computational complexity of VLSI digital filters using fixed point binary multiplier coefficients is normally dominated by the number of adders used in the implementation of the multipliers. It has been shown that using multiplier blocks to exploit redundancy across the coefficients results in significant reductions in complexity over methods using canonic signed-digit (CSD) representation, which in turn are less complex than standard binary representation. Three new algorithms for the design of multiplier blocks are described: an efficient modification to an existing algorithm, a new algorithm giving better results, and a hybrid of these two which trades off performance against computation time. Significant savings in filter implementation cost over existing techniques result in all three cases. For a given wordlength, it was found that a threshold set size exists above which the multiplier block is extremely likely to be optimal. In this region, design computation time is substantially reduced.

## I. INTRODUCTION

THE PROBLEM of designing nonrecursive digital filters with constant fixed point binary coefficient values has attracted a great deal of attention. Early work was dedicated to examination of coefficients with fixed wordlength. Optimal techniques [1]–[4] tend to incorporate a branch-and-bound algorithm. Suboptimal techniques [2], [5]–[9] are generally quicker than the optimal techniques.

The complexity of these filters, when implemented in custom or semi-custom integrated circuits or low cost microcontrollers or microprocessors without an in-built multiplier, was dominated by the number of additions used to implement the coefficient multipliers. To reduce this complexity, canonic signed-digit (CSD) representation [10] can be used for the multiplier coefficients. In signed-digit representation, a ternary representation with digits representing 1, 0, and  $-1$  is used, where  $-1$  represents subtraction. Signed-digit representation is not unique. Methods of finding the canonic representation (the one with fewest adders and subtractors) are described in [11], [12]. On average, CSD uses 33% fewer nonzero digits than binary [13]. Typically, the coefficients allowed are restricted to integers that use no more than one adder or subtractor (known as SOPOT-1 coefficients). Techniques that optimally search this coefficient space [1], [14] do not guarantee that the solution produced has the best performance for the resulting total number of adders and subtractors. Suboptimal search techniques have also been described [14]–[19].

Manuscript received September 9, 1993; revised March 3, 1994. This paper was recommended by Associate Editor E. G. Friedman.

The authors are with the Department of Engineering, University of Cambridge, Cambridge, U.K. CB2 1PZ.

IEEE Log Number 9412393.

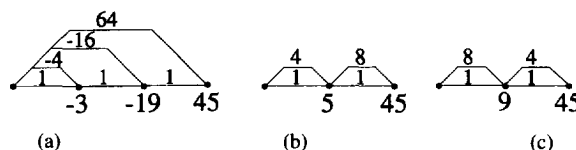


Fig. 1. (a) CSD representation of 45. (b) and (c) 45: Represented with fewer adders than for CSD.

All of the above techniques use the filter performance error (e.g., the sum of the squared error or, alternatively, the minimax error, between the ideal and achieved response at several points in the frequency domain) as the optimization objective function. Some authors [20]–[22] have tried to optimize the design against a cost function such as the number of nonzero CSD digits. This cost function is badly behaved, making these optimizations heuristic, difficult, and possibly highly suboptimal due to local minima. Nongradient methods are required, and have been applied to dual-objective (performance error and implementation cost) problems [23].

In this paper, a different approach is used to the problem of reducing the number of adders and subtractors. Assume that a finite wordlength design has been carried out, for example using one of the existing techniques [1]–[8]. Then, following Bull and Horrocks [24], the filter can be arranged in transposed direct form so that all the multiplications of a given input data sample by filter coefficients are carried out in a single block. The aim of the methods described in this paper is to minimize the number of adders in this multiplier block.

Section II briefly describes some terminology and the use of graphs for the description of multipliers. It also introduces the Bull and Horrocks algorithm [24], and three improvements to it. The new algorithms are presented in Section III. Results are then presented in Section IV followed by discussions of problem complexity and implementation issues.

## II. THE USE OF GRAPHS TO REPRESENT MULTIPLIERS

### A. Graph Representation

The concept of the graph representation of multiplier blocks was introduced by Bull and Horrocks in [24]. We have used a graph-based optimization technique [25], [26] for single coefficients, and shown that it is superior to CSD. An example of a multiplier graph is shown in Fig. 1, where the number 45 is synthesized using CSD and the new technique. Each vertex of the graph represents an adder, and each edge represents a multiplication by a power of 2, which can be executed with minimal complexity; in hardware it is wired as a simple shift.



Hereafter, the original Bull and Horrocks algorithm will be referred to as "BH" and the Bull and Horrocks algorithm incorporating the above modifications will be referred to as "BHM."

### III. THE $n$ -DIMENSIONAL REDUCED ADDER GRAPH (RAG- $n$ ) ALGORITHM

#### A. Description of the New Algorithm

The RAG- $n$  algorithm is in two parts; the first is optimal, i.e., if the set of coefficients is completely synthesized by this part of the algorithm, minimum adder cost is assured, and the second part is heuristic. It uses two lookup tables generated by the MAG algorithm of [25], which, at present, cover the range 1 to 4096. For each coefficient value in this range, the cost lookup table contains the optimum single-coefficient costs of this multiplication and the fundamentals lookup table contains the different sets of fundamentals that can be used to implement the multiplication at optimal cost.

The algorithm is as follows:

- 1) Reduce all coefficients in the set to odd fundamentals (i.e., divide even coefficients by 2 until odd). Store the results in the 'incomplete set'—the set of fundamentals not yet synthesized.
- 2) Evaluate all single-coefficient costs by using the cost lookup table.
- 3) Remove from the incomplete set all cost-0 fundamentals (i.e., powers of two that have been reduced to "1") or repeated fundamentals, as these incur no cost.
- 4) Create the 'graph set' for the storage of selected fundamentals. Into this set, enter all cost 1 (e.g.,  $5 = 4 + 1$ ) fundamentals and remove these from the incomplete set.
- 5) Examine pairwise sums of fundamentals in the graph set with power-of-2 multiples of these same fundamentals. If any of the coefficients in the incomplete set is produced, remove them from the incomplete set and place them in the graph set.
- 6) Repeat step 5 until no more fundamentals are added to the graph set.

(If at any time, the complete set of coefficients is synthesized, the algorithm terminates.)

If the multiplier block is fully synthesized by the end of step 6, it will be optimal, that is it will have the minimum cost possible for that set of coefficients. This can be explained with reference to the following theorems.

**Theorem 1:** A set of  $n$  nonrepeated odd coefficients which each have single-coefficient cost 1 or more cannot be synthesized using fewer than  $n$  adders.

**Proof:** Since none of the coefficients is a power-of-2 multiple of any other, none can be derived from the others without the use of at least one adder.  $\square$

**Theorem 2:** For a set as described in Theorem 1 to incur an adder cost of  $n$ , at least one cost-1 coefficient must appear in the set.

**Proof:** If the graph representing the coefficient set is synthesized one vertex at a time, the first (odd) vertex synthesized must be cost-1. If this vertex is not in the coefficient set, then

at least  $n$  more adders are required in order to synthesize the multiplier block. Therefore this cost-1 vertex must be in the coefficient set if it is to be an optimal cost block.  $\square$

If the multiplier block is fully synthesized by the end of step 6, it will be optimal, because, after step 3, there are only ( $n$ , say) odd fundamentals in the incomplete set, which are non-repeated and have cost greater than zero. In steps 4 to 6, the algorithm only creates a new graph vertex, i.e., adds an adder to the block, if a required coefficient is synthesized by the addition of that adder. Therefore, vertices only exist if they are labelled with one of the required coefficients. Therefore, a maximum of  $n$  vertices can be assigned by this section of the algorithm. If all the coefficients are synthesized, then by Theorem 1, this is an optimal design.

It is worth noting that although incomplete, the algorithm in this form would fully process all of the examples in the Bull and Horrocks paper [24]. One of these examples, in Fig. 3(a) (Fig. 8 of [24]), forms the coefficient set (1, 7, 16, 21, 33) using 4 adders. The RAG- $n$  algorithm produces the true minimum cost of 3 adders (fundamentals 7, 21, and 33) in Fig. 3(b).

The remaining heuristic part of the algorithm uses a new concept—'adder distance.' The adder distance of a new vertex from an existing graph is the number of extra adders (i.e., adders that are not in the existing graph) that are needed to reach the new vertex. The aim of the heuristic part of the algorithm is to find the minimum adder distance between the graph created by the optimal part of the algorithm and the remaining terminating vertices, which correspond to the coefficients not yet synthesized. Step 5 of the optimal section above can be considered to be an exhaustive exploration of the vertices at adder distance 1 from the existing graph.

The heuristic part of the algorithm continues as follows:

- 7) Check each of the incomplete set for two different instances of distance 2 vertices:
  - a) The case where the single-coefficient cost of the difference between the coefficient of interest and any existing fundamental is 1.
  - b) The case where the single-coefficient cost of the difference between the coefficient of interest and the sum of any two existing fundamentals is 0.

In both these cases, two new fundamentals will be created, the new coefficient and one other. In the first case, this latter fundamental is the cost 1 difference which was calculated. In the second case it is the sum of the two existing fundamentals. If such a case is found, add the two fundamentals to the graph set, and remove the coefficient from the incomplete set.

- 8) Repeat steps 6 and 7 until no new distance 1 or 2 fundamentals are found.
- 9) If this point is reached, there are coefficients that are at a greater distance than 2 from the existing graph, or at a distance 2 with a topology which is not covered by the two examples in step 7. Therefore an arbitrary choice of fundamentals to add to the graph set must be made. Of the incomplete set, select the minimum fundamental of lowest single-coefficient cost. Synthesize

it by selecting the set of fundamentals which has the lowest numerical sum. Add these fundamentals to the graph set and remove the coefficient from the incomplete set.

10) Repeat steps 6 to 9 until all coefficients are synthesized.

Suboptimality of the algorithm is due to the following:

- 1) The check of the distance 2 vertices in step 7 is not exhaustive.
- 2) The order in which distance 2 or greater coefficients are added, and the fundamental paths to those coefficients, is arbitrary.
- 3) In the implementation used to produce the results presented in this paper, some limits were set due to computer speed and memory allocation. For instance, the MAG algorithm used in [25] created lists of fundamentals for cost 4 numbers that were truncated to length 200. Since this restriction was placed on the fundamentals table, not all graphs were searched. This restriction did not affect the optimality of the cost lookup table created by MAG.

#### B. A Hybrid Algorithm

Initial results (presented in Section IV) showed that for small set sizes, the RAG- $n$  algorithm was substantially slower than the BH and BHM algorithms. This was because the although the optimal part of the algorithm is relatively fast, the subsequent search was slow and was dominating computation time. A hybrid solution was tested, which uses the optimal part of RAG- $n$ , and if further processing is required, it is performed using the BHM algorithm.

### IV. RESULTS

#### A. Set Size and Wordlength Variation—RAG- $n$

For set sizes (numbers of coefficients) of 3, 5, 7, 10, 15, 25, 40, and 80, one hundred uniformly distributed random sets of coefficients were costed for even wordlengths up to 12 b. The average adder cost, average computation time and the number of sets that were completely costed by the optimal part of the algorithm are shown in Figs. 4, 5, and 6.

It can be seen in Fig. 4 that for a given wordlength, average adder cost increases roughly linearly with set size. For the smaller wordlengths, an asymptote is reached which is the cost of the graph that can fully represent all of the coefficients of that wordlength. The value of this asymptote is the number of odd integers of wordlength  $w$ , i.e.,  $2^{w-1}$ . Once this asymptote is reached, any "new" coefficient is simply a repetition of a coefficient already in the set. For this reason, no further work is done by the algorithm after step 3, so only steps 1 to 3 contribute to increasing computation time.

Fig. 5 illustrates possibly the most interesting characteristic of the RAG- $n$  algorithm. For a given wordlength, a graph of computation time versus set size exhibits a definite peak followed by a definite trough, followed by a region of linear increase. The explanation for this behaviour is aided by Fig. 6, which shows the percentage of sets known to be costed optimally. In the initial region of increasing computation time,

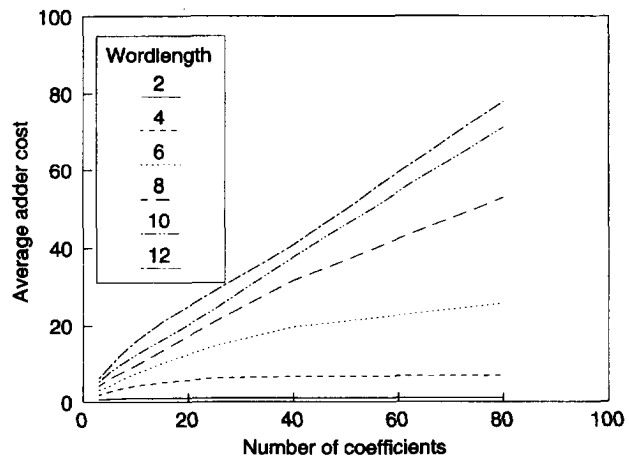


Fig. 4. Average adder costs for the RAG- $n$  algorithm for various wordlengths against uniformly distributed coefficient set size.

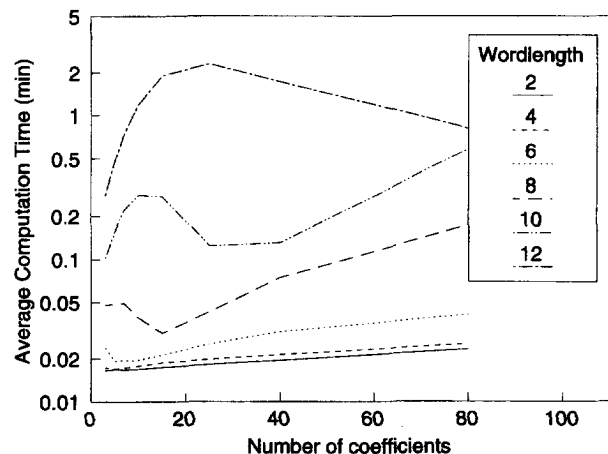


Fig. 5. Average computation times for the RAG- $n$  algorithm for various wordlengths against uniformly distributed coefficient set size. Note the logarithmic scale for computation time.

the suboptimal part of RAG- $n$  is being used on each occasion to design the multiplier block. This can be seen in Fig. 6, where the percentage of optimally costed sets is 0. The optimal part of RAG- $n$  requires the occurrence of cost-1 coefficients in the set before it can produce optimal results, as indicated by Theorem 2. In fact, this part of the algorithm requires a cost-1 coefficient even if it only partially synthesizes the block. Increasing the set size increases the probability of the occurrence of cost-1 coefficients. When some sets use only the optimal part (percentage of optimal sets in Fig. 6 is greater than 0), the computation time drops, approaching a minimum when all sets are costed optimally. After this point, the computation cost increases linearly, because the effort in the optimal part of the algorithm is linear with set size.

An important implication of Fig. 6 is that for a given wordlength, there is an "optimality threshold" set size above which it is highly likely that the design has optimal cost. This is an almost counterintuitive result, since given the potential explosive growth of complexity of the problem with set size, further discussed in Section 4.3, optimality may have been expected to be less likely for large set sizes.

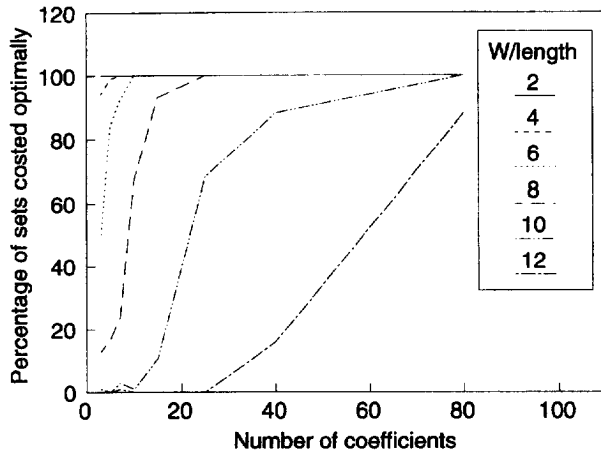


Fig. 6. The percentage of sets known to be costed optimally (i.e., completely designed by the optimal part of the algorithm) by the RAG- $n$  algorithm for various wordlengths against uniformly distributed coefficient set size.

The locations of the peaks and troughs of Fig. 5 are illustrated in Fig. 7. Also shown are the set sizes at which the percentage of known optimal sets reaches 100 in Fig. 6 (the optimality threshold). Due to the quantization in set size of Figs. 5 and 6, these estimates of peak, trough and threshold location are not highly accurate. From the discussion above, it could be expected that there is a relationship between the set sizes at which peaks, troughs and thresholds occur, and the probabilities of a cost 1 coefficient appearing in the set. The probability of a cost-1 coefficient of given wordlength  $w$ ,  $P_w^1$  can be evaluated by using the cost lookup table generated by the MAG algorithm of [25]. The probability of at least one cost-1 coefficient occurring in a set of size  $n$  and wordlength  $w$  is

$$P_{nw}^1 = 1 - (1 - P_w^1)^n. \quad (1)$$

Therefore the set size  $n = n_p$  required to achieve a probability  $P_{nw}^1 = p$  of a cost-1 occurring in the set is

$$n_p = \frac{\log(1 - p)}{\log(1 - P_w^1)}. \quad (2)$$

Overlaid on Fig. 7 are the graphs of the set sizes required to ensure that the probability of a cost 1 coefficient is 0.60, 0.92, and 0.992, given a uniform distribution of coefficient values in the coefficient space. These were the curves which were found to give the best fits to locations of peaks, troughs, and optimality thresholds, respectively. These seem to confirm the argument that it is the occurrence of cost 1 coefficients that critically affects the computation time of the RAG- $n$  algorithm. The two overlays provide a method of predicting the location of peaks and troughs for other wordlengths. Also, if the distribution of coefficients is not uniform, but is known, it is a simple matter to calculate the values for  $P_w^1$  by summing the probabilities of the individual cost-1 coefficients in the given wordlength range, to produce curves to replace those in Fig. 7. For instance, if cost-1 coefficients are more likely to occur than in a uniform distribution,  $P_w^1$  is higher, and  $n_p$  in (2) is lower, i.e., the optimality threshold is lower.

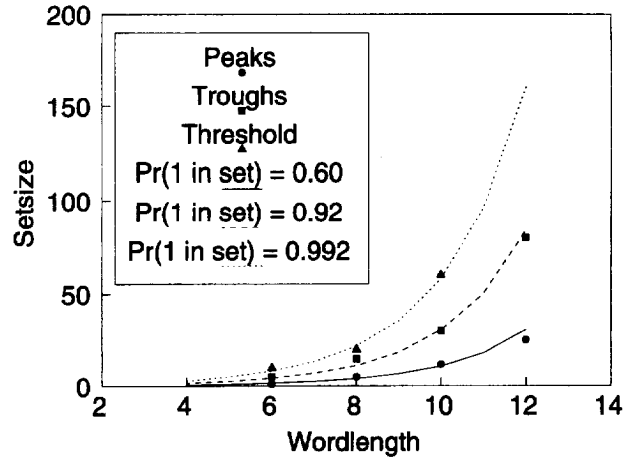


Fig. 7. The set sizes at the peaks and troughs of Fig. 5 and the optimality thresholds of Fig. 6 for various wordlengths. The set sizes required for probabilities of 0.60, 0.92, and 0.992 of a cost 1 coefficient occurring in the set are overlaid.

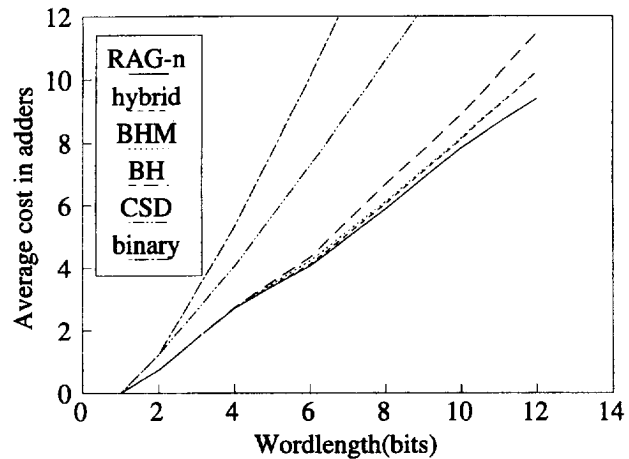


Fig. 8. Average cost in adders evaluated for various algorithms against wordlength. Each point represents the average over one hundred uniformly distributed 5-coefficient sets.

### B. Wordlength Variation—All Algorithms

For set sizes of 5 and 25 (the number of coefficients that would be required for the implementation of linear phase FIR filters of order 9 or 10, and 49 or 50, respectively), the average adder cost of a multiplier block, and average computation times for the BH, BHM, hybrid and RAG- $n$  algorithms, were compared over a range of wordlengths. One hundred uniformly distributed random sets were used to calculate the averages. These results are shown in Figs. 8–10 and 11. In Fig. 8, comparisons with CSD and binary are also made.

Fig. 8 shows that, for 5-coefficient sets, the RAG- $n$  algorithm provides a significant improvement in cost over BHM (8.4% for 12 b words), which in turn provides a significant improvement over the original BH (10.6%). However, it does so at a far greater cost in computation time. All the algorithms that utilize graph synthesis techniques are far superior in terms of adder cost to CSD and binary. For 25-coefficient sets, however, there is no great improvement in performance due to RAG- $n$  until the wordlength exceeds 8 b. Up to

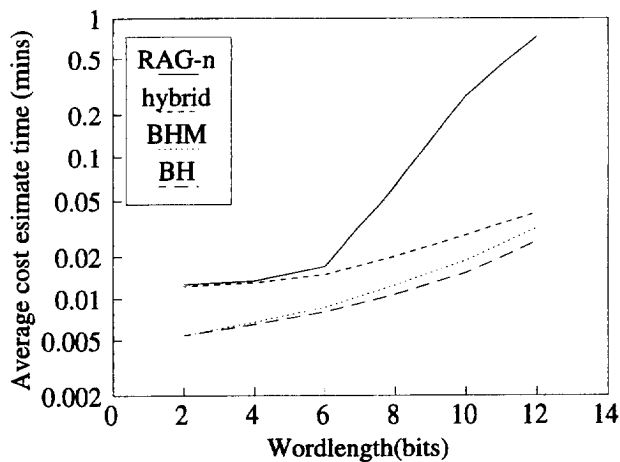


Fig. 9. Average computation time for various cost evaluation algorithms against wordlength. Each point represents the average over one hundred uniformly distributed 5-coefficient sets.

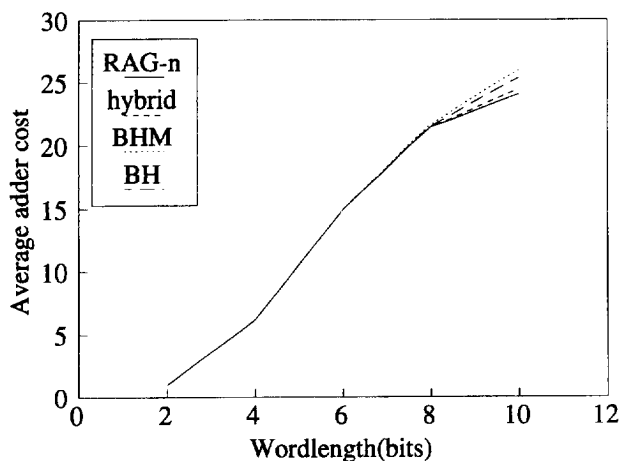


Fig. 10. Average cost in adders evaluated for various algorithms against wordlength. Each point represents the average over one hundred uniformly distributed 25-coefficient sets.

8 b wordlengths (see Fig. 6), RAG- $n$  is almost certainly optimal, i.e., for 8 b wordlengths, 25 is above the optimality threshold set size. So the other algorithms must also be operating optimally in this region. For the BH algorithm, this is due to "extraneous" vertices becoming less likely—i.e., a vertex produced in the synthesis of another vertex becomes increasingly likely to be in the set itself. For the BHM algorithm, the effect is stronger, because cost-1 coefficients are synthesized first, so the advantages inherent in RAG- $n$ , of firstly creating a graph using cost-1 coefficients, also serve to make BHM near-optimal below the optimality threshold.

There is a rapid increase in the computation time (Fig. 9) used by the RAG- $n$  algorithm for 5 coefficients as wordlength is increased. This is related to the rapid growth in the complexity of the problem with wordlength for small set sizes. By contrast, the computation time for the BH and BHM algorithms rises more slowly with wordlength, and thus does not suffer the same explosion in computation time. The heuristic part of the RAG- $n$  algorithm is slow, whereas the optimal part is relatively fast (this is confirmed by the analysis of Figs. 5 and

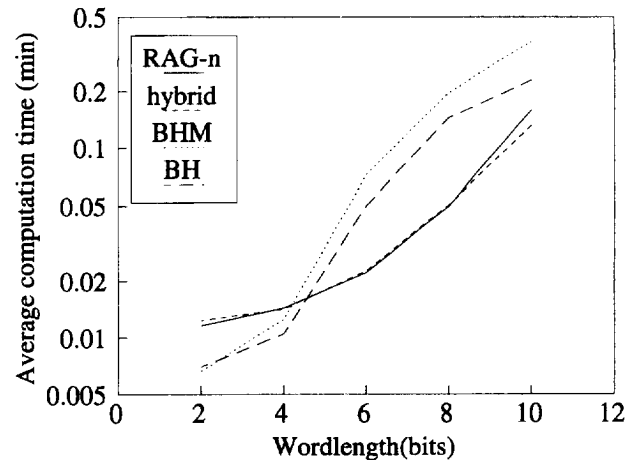


Fig. 11. Average cost in adders evaluated for various algorithms against wordlength. Each point represents the average over one hundred uniformly distributed 25-coefficient sets.

6). This is the reasoning behind the introduction of the hybrid algorithm mentioned in Section 3.2. For the 5-coefficient examples of Figs. 8 and 9, the computation cost is indeed reduced, but there is little gain in adder cost over BHM. This is because the optimal part of RAG- $n$  requires there to be cost 1 coefficients in the set, and for longer wordlengths, this becomes a rarer occurrence (see Fig. 7). The advantage of the hybrid algorithm over BHM peaks at 3% for 6 b words. For 25-coefficient sets (Fig. 11), the RAG- $n$  algorithm is faster than BH and BHM for useful wordlengths. All algorithms show rapid growth in computation time and RAG- $n$  appears to have the fastest growth. While the set size is below the optimality threshold (for wordlengths up to 8), the hybrid algorithm and the RAG- $n$  algorithm have very similar computation times, as expected since they are performing identical tasks. For 10-b wordlengths, however, the hybrid is superior, confirming that the BHM algorithm is more computationally efficient than the suboptimal part of RAG- $n$ . As can be seen in Fig. 10, there is a minor penalty in adder cost associated with using the hybrid algorithm.

### C. Set Size Variation—All Algorithms

For 10-b wordlengths, average adder cost and average computation time were evaluated for the RAG- $n$ , BHM, BH and hybrid algorithms. Again, 100 sets were used for the averages. These are shown in Figs. 12 and 13.

The computation time of BH and BHM in Fig. 13 increases quadratically with set size. The operation which dominates this timing is the checking of the difference between the current "error" (Bull and Horrocks' term for the difference between the current coefficient of interest and the closest partial sum) and the pairwise sums of all partial sums with all other partial sums. The complexity of this operation grows as a square law with the number of partial sums. Initially, computation time for RAG- $n$  is higher than that for these algorithms, but at a point after RAG- $n$  reaches its peak, it is then faster. The hybrid algorithm, however, uses the BHM algorithm for small set sizes, as the optimal part of RAG- $n$  does not produce a result, and RAG- $n$  for large set sizes where optimal performance

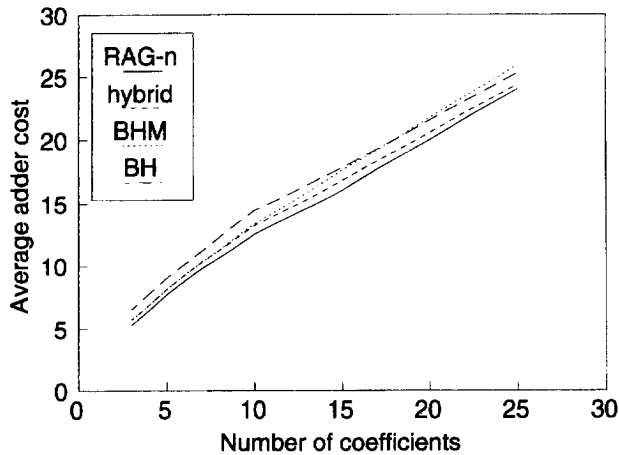


Fig. 12. Average adder cost for various algorithms against coefficient set size for a 10-b wordlength.

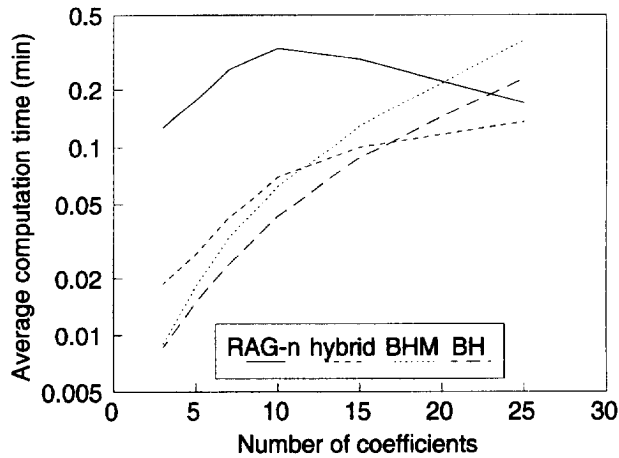


Fig. 13. Average computation time for various algorithms against coefficient set size for a 10-b wordlength.

dominates. This means that it uses the quickest algorithm in each of these two regions, and its computation time is thus the best compromise characteristic. Above the optimality threshold, the hybrid algorithm uses the optimal part of RAG- $n$  exclusively, so the timing for these two algorithms coincides. Fig. 12 shows that the penalty in adder cost for using the hybrid algorithm as opposed to RAG- $n$  is minor. In fact, the advantage of RAG- $n$  over the other algorithms is expected to diminish with increasing set size, due to the ability of all the algorithms to cost large set sizes near-optimally due to the high rate of occurrence of cost-1 coefficients, which was discussed in the previous section.

Bull and Horrocks [24] state that the problem of synthesising a graph for a set of coefficients of size  $n$  is  $NP$ -complete. The definition of  $NP$ -completeness uses the maximum time for any problem instance as the time complexity function [27]. The results in Figs. 5 and 13 seem to show, however, that the *average* processing time does not grow exponentially. On the contrary, above a certain set size, optimality can be assured to a very high probability, and the computation time of the algorithm grows linearly with set size. This is partly because the RAG- $n$  algorithm introduces techniques to exploit

TABLE I  
ADDER COSTS FOR EXAMPLES 2 AND 3 FROM [1]. COSTS ARE EVALUATED FOR BINARY, CSD, AND RAG- $n$  MULTIPLIER BLOCK IMPLEMENTATION OF THE COEFFICIENT MULTIPLIERS. BOTH ARE COSTED OPTIMALLY BY RAG- $n$

L&P Example (from [1])	Order	Wordlength	Coeff. Type	Costs		
				Binary	CSD	RAG- $n$
2	63	10	Integer	76	49	22
3	36	8	SOPOT-1	22	16	5

redundancy. First, only odd fundamentals (vertex values) are allowed. This restriction reduces significantly the valid values that edges can attract. Second, knowledge of the nature of the single-coefficient costs is used, both to dismiss coefficients of zero cost, and to identify cost-1 coefficients which are the key to the optimal part of the algorithm. The third, and most important saving is made because there are likely to be many optimal solutions and some of these are easily located by simply building a graph by adding coefficients at adder distance one from the existing graph. Acknowledgement of the fact that a non-repeated coefficient will always have at least one adder assigned is the key. Even when below the optimality threshold, for the complexity of the problem to increase exponentially, each coefficient in the set must be at least adder distance 2 from the graph built up from the other coefficients. As set size increases, this becomes increasingly unlikely. In fact the multiplicity of graph synthesis options makes distance-1 coefficients relatively easy to find. In other words, the very complexity of the problem works to decrease the computation time.

Another fact that greatly reduces the average computation time as set size grows is that repetition of coefficients gradually becomes more common. Ultimately, a limit is reached, as was discussed with reference to Fig 4, where all coefficients of a given wordlength are synthesised. In other words, for a given wordlength, there is a limit to the problem complexity.

#### D. Example Filters

Two example filters are taken from Lim and Parker [1] for examination and discussion. The first is order 63 with 10 b integer coefficients. The second is order 36 with 8 b coefficients which are allowed only one adder or subtractor each (SOPOT-1 coefficients); both are linear phase (symmetric). The results of costing these filter coefficient sets as binary and CSD numbers, and as multiplier blocks designed by RAG- $n$ , are shown in Table I.

It can be seen that for both of these examples, the reduction in adders due to using multiplier blocks is substantial. The costs are 45% and 31% of the CSD costs, respectively, for Examples 2 and 3. Example 3, because of its coefficient adder constraints, only has cost-1 coefficients, and is hence optimally costed. However, Example 2 also had a large number of cost-1 coefficients, which led to it being optimally costed. This is partly due to the large set size (31), which is in the transition region for optimality (i.e., just below the optimality threshold) for uniformly distributed 10 b words (see Fig. 6), but more predominantly, it is due to the nature of the impulse response

of the filters, which, like most FIR filters, have many small coefficients. The smaller the coefficient, the more likely it is to be cost-1. Of the 32 coefficients, which could attain a maximum magnitude of 1023, only 6 had magnitude greater than 128 and 11 had magnitude greater than 64. Because the distribution of coefficient values is not uniform, and favours the occurrence of cost-1 coefficients and the repetition of coefficients (reducing the effective set size), the optimality threshold for a lowpass filter of this type can be expected to be much lower than that shown in Fig. 6, as discussed in Section 4.1.

Another interesting implication of the results in Table I is that the number of adders in the multiplier block is significantly less than the number of adders intrinsic to the filter structures in Fig. 2. This effect is more pronounced in the second example, because the coefficients are 8 b words which only require one adder anyway. However, the first example, which has 10 b unconstrained integer coefficients, requires only 22 adders to perform the multiplication function, whereas there are 63 adders and 63 delays required outside the multiplier block to complete the filter. This means that to save one adder in the multiplier block does not decrease the complexity by 1 in 22, but by 1 in 85 in adders alone (or 1 in 148 if delay elements are assigned the same area as adders). In other words, the effort required to search in coefficient space for the optimum RAG- $n$  cost is not rewarded by a significant cost saving.

### E. Implementation Issues

The technique described in this paper can be applied to both programmed processors with shift and add but no multiplier, or to dedicated VLSI hardware. It has been suggested in [28] that the irregularity of multiplier blocks as described by Bull and Horrocks [24] may be a disadvantage in VLSI implementations. However, the penalty is small, especially since the adders and delays intrinsic to the filter often greatly outnumber those in the multiplier block. For fixed-coefficient filters, the technique described in [28] requires not only more adders than the BH algorithm, but also additional RAM, ROM and control logic. All three algorithms presented here require fewer adders than the BH algorithm, no extra circuitry and no extra interconnection complexity.

### V. CONCLUSION

Results can be summarized as follows:

- 1) The heuristic RAG- $n$  multicoefficient cost multiplier block design algorithm produces an average improvement of about 20% over the BH algorithm for five coefficients of 12 b wordlength.
- 2) Two alternative algorithms which produce less good results than RAG- $n$  (i.e., produce a higher cost graph) are the modified Bull and Horrocks algorithm, BHM, which produces improvements of about 10% over the original BH algorithm for 12 b wordlengths, and a hybrid algorithm.
- 3) The RAG- $n$  algorithm is slower than BHM for small coefficient sets but is quicker for large sets. For large

set sizes, the computation time of the BH and BHM algorithms has a square law growth rate with set size, whereas for RAG- $n$  it grows linearly. The hybrid algorithm tends to use the quicker of the two algorithms in both regions. This behaviour contrasts strongly with the theoretical NP-completeness of the problem.

- 4) Optimality of the design becomes highly likely for set size above a threshold for a given wordlength. A method for estimating this optimality threshold, based on probability of occurrence of a cost-1 coefficient, for uniformly-distributed coefficients, has been presented. For most FIR filters, this threshold estimate is likely to be quite conservative because the distribution of coefficients is likely to be very nonuniform and biased toward lower costs. The RAG- $n$  and hybrid algorithms can indicate to the user if the multiplier block has been designed optimally. It may be still be optimal even it is not guaranteed to be.
- 5) By using the multiplier block technique, the contribution of the multipliers to overall implementation complexity can be reduced to such a degree that it is far less significant than the contribution from the structural adders and delay elements.

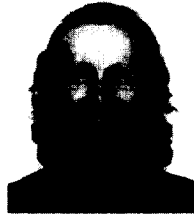
For set sizes above the optimality threshold, the RAG- $n$  algorithm produces optimal results, while BH and BHM are near-optimal. RAG- $n$  is quicker, however and is therefore unconditionally recommended for FIR multiplier block design for large coefficient sets. Below the optimality threshold, RAG- $n$  again produces the best results, although they are not necessarily optimal, but it is slow. The hybrid algorithm uses the most computationally efficient of RAG- $n$  and BHM in a given circumstance. Therefore, for small sets, if the main emphasis is on speed, the hybrid algorithm is recommended; if cost minimization is most important, the RAG- $n$  algorithm is recommended.

### REFERENCES

- [1] Y. C. Lim and S. R. Parker, "Discrete coefficient FIR digital filter design based upon an LMS criteria," *IEEE Trans. Circuits Syst.*, vol. 30, pp. 723-739, Oct. 1983.
- [2] D. Kodek and K. Steiglitz, "Comparison of optimal and local search methods for designing finite wordlength FIR digital filters," *IEEE Trans. Circuits Syst.*, vol. 28, pp. 28-32, Jan. 1981.
- [3] D. M. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, pp. 304-308, June 1980.
- [4] J. P. Marques de Sa, "A new design method of optimal finite wordlength linear phase FIR digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 31, no. 4, pp. 1032-1034, May 1983.
- [5] C. Young and D. L. Jones, "Improvements in finite wordlength FIR digital filter design by cascading," in *ICASSP*, IEEE, Mar. 1992, pp. V109-112.
- [6] J. W. Adams and A. N. Willson Jr., "A new approach to FIR digital filters with fewer multipliers and reduced sensitivity," *IEEE Trans. Circuits Syst.*, vol. 30, pp. 277-283, May 1983.
- [7] ———, "Some efficient digital prefilter structures," *IEEE Trans. Circuits Syst.*, vol. 31, pp. 260-265, Mar. 1984.
- [8] Y.-L. Tai and T.-P. Lin, "Design of multiplierless FIR filters by multiple use of the same filter," *Electron. Lett.*, vol. 28, no. 2, pp. 122-123, Jan. 1992.
- [9] U. Heute, "A subroutine for finite wordlength FIR filter design," in (ASSP DSP Committee) *Programs for Digital Signal Process.*, IEEE, 1979.



- [10] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. 10, pp. 389-400, Sept. 1961.
- [11] G. W. Reitwiesner, "Binary arithmetic," *Advances in Computers*, vol. 1, pp. 233-308, 1960.
- [12] K. Hwang, *Computer Arithmetic: Principles, Architecture and Design*. New York: Wiley, 1979.
- [13] H. L. Garner, "Number systems and arithmetic," *Advances in Computers*, vol. 6, pp. 131-194, 1965.
- [14] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 31, no. 3, pp. 583-591, June 1983.
- [15] Y. Ching Lim and B. Liu, "Design of cascade form FIR filters with discrete valued coefficients," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 11, pp. 1735-1739, 1988.
- [16] A. Miron and D. Koo, "Design of multiplierless FIR digital filters with two to the  $N$ th power coefficients," *IEEE Trans. Consumer Electron.*, vol. 33, no. 3, pp. 109-114, Aug. 1987.
- [17] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044-1047 July 1989.
- [18] Q. Zhao and Y. Tadokoro, "A simple design of FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 566-570, May 1988.
- [19] N. Benvenuto, M. Marchesi, and A. Uncini, "Applications of simulated annealing for the design of special digital filters," *IEEE Trans. Signal Process.*, vol. 40, pp. 323-332, Feb. 1992.
- [20] R. Jain, J. Vandewalle, and H. de Man, "Efficient CAD tools for the coefficient optimisation of arbitrary integrated digital filters," in *ICASSP*, IEEE, 1984, pp. 30.11.1-4.
- [21] ———, "Efficient and accurate multiparameter analysis of linear digital filters using a multivariable feedback representation," *IEEE Trans. Circuits Syst.*, vol. 32, pp. 225-235, Mar. 1985.
- [22] R. Jain, "Computer-aided discrete coefficient optimisation for custom integrated digital filters," Ph.D. dissertation, ESAT Lab., Katholieke Universiteit Leuven, May 1985.
- [23] P. B. Wilson and M. D. Macleod, "Low implementation cost IIR digital filter design using genetic algorithms," in *Natural Algorithms in Signal Processing Workshop*. Essex, Chelmsford: IEE, Nov. 1993.
- [24] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *IEE Proc. G*, vol. 138, no. 3, pp. 401-412, June 1991.
- [25] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. — Circuits, Devices, Sys.*, vol. 141, no. 5, pp. 407-413, Oct. 1994.
- [26] ———, "Multiplication by an integer using minimum adders," in *Mathematical Aspects of Digital Signal Processing Colloquium*, IEE, Feb. 1994.
- [27] D. S. Johnson and M. R. Garey, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.
- [28] D. L. Jones, "Efficient computation of time-varying and adaptive filters," *IEEE Trans Signal Process.*, vol. 41, pp. 1077-1086, Mar. 1993.



**Andrew G. Dempster** (S'94) received the B.E. and M.Eng.Sc. degrees from the University of New South Wales, Australia, in 1985 and 1992, respectively.

He worked as a transmission equipment design engineer for STC in Sydney for three years, followed by five years with Auspace Limited in Canberra, where he was a system engineer and project manager of a DSP-based satellite navigation receiver development. He is currently reading for the Ph.D. degree in engineering with the Communications and Signal Processing Laboratory, Cambridge University, England.



**Malcolm D. Macleod** (M'86) was born on September 26, 1953 in Cathcart, Glasgow, Scotland. He received the B.A. (with distinction), M.A., and Ph.D. degrees from the University of Cambridge, England, in 1974, 1978, and 1979, respectively.

From 1978 to 1988 he was with Cambridge Consultants Ltd, part of the Arthur D. Little group, on a wide range of signal processing, electronics and software research and development projects. Since 1988 he has been a lecturer with the Signal Processing and Communications group of the Engineering

Department of Cambridge University. His main research interests are in digital filter design, nonlinear filtering, optimal detection, high resolution spectrum estimation and beamforming, and applications in sonar and communication systems.

Dr. Macleod is a member of the IEE (U.K.) professional group on signal processing.