# Multiplication by two integers using the minimum number of adders

**A G Dempster[1] and M D Macleod[2]**

[1]University of Westminster, 115 New Cavendish St, London W1W 6UW, UK

Tel: +44 20 7911 5891, Fax: +44 20 7580 4319, dempsta@wmin.ac.uk

[2] QinetiQ Ltd, St Andrews Road, Malvern, Worcs, WR14 3PL, mdmacleod@iee.org

***Abstract*** – The two-coefficient minimum-adder graph (MAG2) algorithm is described and shown to be optimal. A partial implementation, optimal for pairs of integers up to 8 bits, outperforms the best existing algorithms.

## 1. INTRODUCTION

### 1.1. Efficient Multiplier Design

Circuits for shift-and-add multiplication by one or more fixed-point constant coefficients can be built using both adders and subtractors. We use graph-based methods to illustrate these circuits, where each vertex represents an adder and has two inputs. An edge represents a shift and as it can be hard-wired is considered "free". After Bull and Horrocks [1], we developed algorithms that use fewest adders for multiplication by a single integer [2, 3, 4] and by several integers [5].

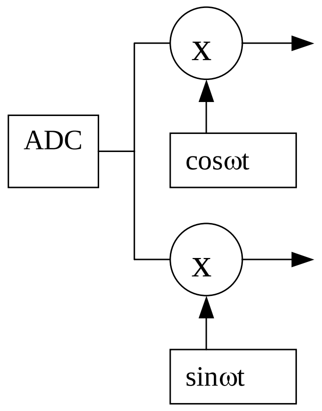

**Figure 1. Structure used for digital downconversion and FFT butterflies: data multiplied by two coefficients**

Subexpression elimination (SE), introduced by Hartley [6], also aims to reduce adders in fixed-point multiplication. It combines pairs [7] or groups [8, 9] of digit patterns that recur in signed-digit (SD) representations of coefficients. A single adder can produce each pattern (subexpression), reducing overall adder count. Canonic signed-digit (CSD) [6, 7, 9] and other representations [8] have been used. Representations using fewest bits [10] and more bits [11, 12, 13] improve results.

We examine a special "multiple" coefficient case: a coefficient pair. This is not dealt with well by the existing techniques. We describe an algorithm, introduced in [14], briefly mentioned in [15], but which has not been widely available. We did not publish the algorithm earlier due to its slowness. Now faster computers are available, results for sensible wordlengths can be achieved.

## 2. THE TWO COEFFICIENT MINIMUM-ADDER GRAPH (MAG2) ALGORITHM

### 2.1. Why use two-coefficient blocks?

The several-coefficient multiplier block algorithms (e.g. [5]) were designed for FIR filters, i.e. for large numbers of coefficients, not for pairs of coefficients. Applications using pairs of coefficients are second-order IIR sections [16], digital downconversion, and twice in the complex rotator in FFT butterflies. The latter two are illustrated in Figure 1. Two-coefficient digital downconversion can also use reconfigurable design [17], with a new multiplier block circuit for each evaluation of the product pair.

### 2.2. Single Coefficient Multipliers

In [2], we introduced the Minimum Adder Graph (MAG) algorithm, which designs shift-and-add circuits that multiply by an integer using fewest possible adders. It considers the circuit as a graph, made up of two-input adders. It performs an exhaustive search of all possible graph topologies (i.e. interconnections of the two input adders) and produces two tables. One contains the number of adders required to produce the integer multiplier. The other contains the partial products that are the outputs of the adders in the circuit (other than the circuit's output) – known as "fundamentals".

## 2.3. The New Algorithm

The new algorithm aims to produce shift-and-add multiplication using fewest adders for any given pair of integers. Below is a more complete description of that in [14] in which we make bald statements about optimality, or minimum and maximum costs. These are discussed more fully in the next section.

Terminology:

- $c_1$, $c_2$ are the two coefficient values,
- $C_1$, $C_2$ are the two optimal adder costs of $c_1$, $c_2$ (i.e. numbers of adders used for them as single integer multipliers), taken from the MAG cost table. $c_1$, $c_2$ are selected without loss of generality such that $C_1 \geq C_2$.
- $C$ is the cost of the multiplier block that produces the two products.

<u>*Step 1.*</u> Divide $c_1$, $c_2$ by 2 until odd. Any even number can be generated by shifting an odd number. This step simplifies many graphical algorithms [2 - 5].

<u>*Step 2.*</u> Eliminate trivial case 1. If $c_1 = c_2$, $C = C_1$, i.e. if the coefficients are the same then the "two" products come from the same output in the circuit.

<u>*Step 3.*</u> Eliminate trivial case 2. If $C_2 = 0$, $C = C_1$, i.e. if one of the coefficients has cost 0, i.e. it is a power of 2 and hence can be achieved through shifts alone and requires no adders, then it contributes nothing to the overall cost of the circuit.

<u>*Step 4.*</u> Look for cases where $c_2$ is a fundamental of $c_1$. In this case $C = C_1$. This search loads all of the possible fundamentals for $c_1$ from the MAG table and simply identifies if $c_2$ appears.

After step 4, we have examined all possible cases where $C = C_1$ so if we find an example of $C = C_1 +1$ then it is optimal and the algorithm can terminate.

<u>*Step 5.*</u> Look at all the sets of fundamentals for both coefficients. Find the maximum number of common fundamentals $C_c$ . Assign $C = C_1 + C_2 - C_c$. If $C = C_1 +1$ then terminate the algorithm.

In this paper, we implemented all steps up to step 5, but no further. Further steps can be implemented as follows:

<u>*Step 6.*</u> Create a new set of tables, similar to those used by MAG, where all possible graphs (i.e. fundamentals) are recorded that describe the graphs that can synthesise $c_1$, $c_2$ at costs $C_1$+1, $C_2$+1. Repeat step 4 using these tables and terminate if a case is found where $C = C_1 +1$.

After step 6, we have examined all possible cases where $C = C_1 +1$ so if we find an example of $C = C_1 +2$ then it is optimal and the algorithm can terminate.

<u>*Step 7 and beyond.*</u> The higher-cost equivalents of steps 5 and 6 can be repeated indefinitely.

If we were looking for a succinct algorithm description, we could ignore steps 1-3 as they are picked up in step 4, and step 5 is similarly picked up in step 6. However, in the interests of speed the steps are written in the order they are because each step is slower than its predecessor. This allows the algorithm to terminate quickly where possible.

In [14], steps 3 and 4 were called "search 1", step 5 was called "search 2", step 6 was called "search 3" and step 7 was called "search 4". The ability to use fundamentals in step 5 is illustrated in Figure 2.

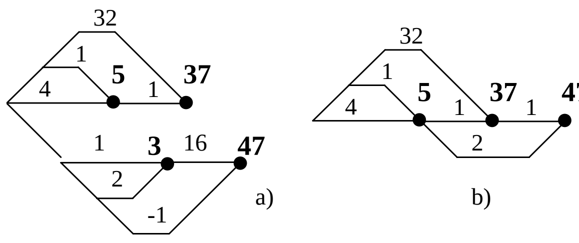

**Figure 2. Cost saving using MAG2. An example lowest cost graph to produce 37 and 47 a) after step 4, and b) after step 5**

## 2.4. Proof of optimality

Optimality proofs were provided in [2] with regard to the MAG algorithm, so the data we receive from the MAG tables can be said to be optimal in the sense that costs are minimum and the lists of fundamentals are exhaustive.

We know that the MAG algorithm is optimal so there is no way we can produce a multiplication by $c_1$ using fewer adders than $C_1$. We also know $C_1 \geq C_2$ so the minimum cost the whole circuit can have is $C = C_1$. Steps 1 to 4 terminate the algorithm at this cost. Because later steps terminate the algorithm at higher costs, we must be convinced that steps 1 to 4 cover all possible cases where $C = C_1$. This is easy. If $C =$

$C_1$, then the graph must be one that is stored in the original MAG tables for $c_1$. We examine all of these in step 4.

The same argument can be made for the remainder of the algorithm. As long as the tables produced for step 6 are exhaustive, then step 6 will produce all results for $C = C_1 + 1$ and so on.

The slightly more complicated proof is for what we have actually done in this paper. Because we did not implement step 6, we must examine the cases where we know step 5 provides optimality. First, any $C = C_1 + 1$ result is optimal because step 4 would have found any lower cost result if one existed. Also, if $C_1 = C_2$ (except for the trivial case $c_1 = c_2$) then if a $C = C_1 + 1$ result existed, it would use the fundamentals of one coefficient to produce the other – part of the graph must be a complete graph from the MAG tables. This would be picked up by step 5 (i.e. step 6 does not have to check the $C_1 = C_2$ case). So, after step 5 any pair of coefficients with $C_1 = C_2$ and a graph with $C = C_1 + 2$ is also optimal.

A short example to show that in fact step 6 is necessary and there may be $C = C_1 + 1$ cases that are not picked up by step 5: Say there is one coefficient ($c_1$) with optimal cost ($C_1$) of 3 and one ($c_2$) with optimal cost ($C_2$) of 2. It is possible that the only optimal $C = C_1 + 1 = 4$ graph for these two coefficients has a subgraph using 3 adders to produce $c_2$ and a further adder using $c_2$ and one of the fundamentals to produce $c_1$. Step 6 would find this case but step 5 would not.

## 3. ALGORITHM BEHAVIOUR AND RESULTS

The following discussion is based on an exhaustive set of results produced by applying the MAG2 algorithm (up to step 5) to all pairs of integers up to 12 bits (i.e. up to 4096). The 8 bit results took 10.5 hours when first examined in [14]. A more efficient design and use of a high-spec Pentium now mean that these 8 bit results take 3.6 minutes. The 12 bit results took 4.0 days. As was reported in [14], the algorithm produces results that are known to be optimal for the reasons discussed in the previous section for pairs up to 8 bits (i.e. 256). The optimality results for longer wordlengths are shown in Table 1. For 12 bit integers, we know that 85.3% of pairs are optimally designed. Some of the remaining pairs may also be (and probably are) optimal, but that cannot be stated categorically.

| Wordlength | Optimal pairs (%) |
|---|---|
| Up to 8 | 100 |
| 9 | 99.7 |
| 10 | 98.1 |
| 11 | 95.3 |
| 12 | 85.3 |

**Table 1 The percentage of MAG2 designs for pairs of a given wordlength that are known to be optimal. Others may be optimal but cannot be confirmed as being so.**

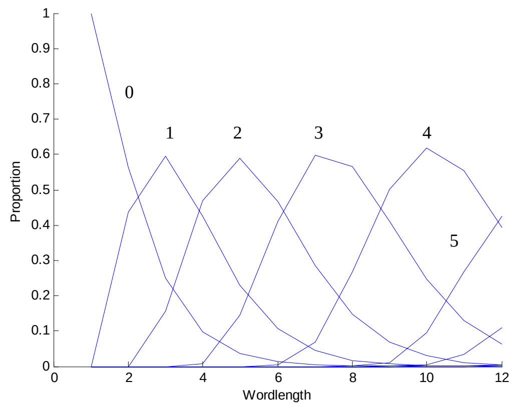

**Figure 3. Frequency of occurrence of adder costs for MAG2 designs for all pairs having wordlengths 1-12 bits.**

The frequency of occurrence of particular costs is shown in Figure 3. For 12-bit pairs, costs of 4 and 5 are the most common. The average costs of the algorithm are shown in Figure 4. On average, two 12-bit integers are produced using 4.6 adders, a big improvement on binary which would require 10 adders and CSD which would require 6.7 adders. The best algorithms with which to compare MAG2 are RAG-n [5] which for 12 bit pairs uses 8% more adders and H(1) [13] which for 11 bit pairs uses 1.5% more adders. The H(1) algorithm is also much more slow (hence the absence of 12-bit results - the 11 bit results took 10 days).

## 4. CONCLUSION

A new optimal algorithm, MAG2 has been presented for designing circuits to produce products of two integers. A partial implementation of the algorithm, known to be optimal up to 8 bits, outperforms the best existing algorithms, RAG-n and H(1).

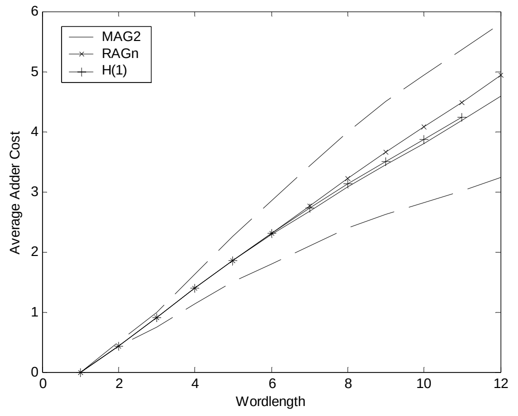

**Figure 4. Average adder costs versus wordlength for MAG2, RAG-n and H(1). The dotted lines are the two-integer cost bounds derived purely from the single-integer costs: upper bound is the MAG cost $C_1+C_2$ , lower bound is $\max(C_1, C_2)$, (note this is not the optimal cost – up to 8 bits the MAG2 results are optimal).**

## 5. REFERENCES

[1] D R Bull and D H Horrocks, "Primitive operator digital filters", IEE Proceedings G, vol 138, no 3, pp401-412, Jun 1991

[2] A G Dempster and M D Macleod, "Constant integer multiplication using minimum adders", IEE Proceedings - Circuits, Devices and Systems, vol 141, no 5, pp407-413, Oct 1994

[3] A G Dempster and M D Macleod, "General algorithms for reduced-adder integer multiplier design", Electronics Letters, vol 31, no 21, pp1800-1802, Oct 1995

[4] O Gustafsson, A G Dempster and L Wanhammer, "Extended Results for Minimum-Adder Constant Integer Multipliers", Proc ISCAS 2002, ppI73-76

[5] A G Dempster and M D Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters" IEEE Trans Circuits and Systems II, vol 42, no 9, pp569-577, September 1995

[6] R I Hartley, "Optimization of canonic signed digit multipliers for filter design", ISCAS91, pp1992-1995

[7] R I Hartley, "Subexpression Sharing in Filters Using Canonic Signed-Digit Multipliers", IEEE Trans Circuits And Systems II, vol.43, no.10, pp.677-688, 1996

[8] M Potkonjak, M B Srivastava and A Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination", IEEE Trans Computer-Aided Design of IC Systems, vol. 15, no. 2. pp151-165, 1996

[9] R Pasko et al, "A new algorithm for elimination of common subexpressions", IEEE Trans Computer-Aided Design of IC Systems, vol. 18, no. 1, pp58 – 68, 1999

[10] In-Cheol Park and Hyeong-Ju Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations", IEEE Trans Computer-Aided Design of IC Systems, vol. 12, no. 12. pp1525-1529, 2002

[11] A G Dempster and M D Macleod, "Generation of Signed-Digit Representations for Integer Multiplication, to be published in Signal Processing Letters

[12] A G Dempster and M D Macleod, "Using All Signed-Digit Representations To Design Single Integer Multipliers Using Subexpression Elimination", Proc ISCAS 2004, 24-26 May, 2004, Vancouver

[13] A G Dempster and M D Macleod, "Digital Filter Design Using Subexpression Elimination And All Signed-Digit Representations", Proc ISCAS 2004, 24-26 May, 2004, Vancouver

[14] Andrew G Dempster, "Digital Filter Design for Low-Complexity Implementation", PhD Thesis, Cambridge University, June 1995

[15] Dempster A G, "Graphical Design Techniques for Fixed-Point Multiplication", *VLSI Design*, vol 11, no4, pp 363-379, Oct 2000

[16] A G Dempster and M D Macleod, "IIR Digital Filter Design Using Minimum-adder Multiplier Blocks", IEEE Trans Circuits & Systems II - Digital & Analog Signal Processing, vol. 45, no. 6, pp. 761-763, June 1998.

[17] Demirsoy S. S., A. Dempster and I. Kale "Design Guidelines for Reconfigurable Multiplier Blocks", *Proc ISCAS 2003*, Bangkok, May 2003, pp IV293-296