An Algorithm for the Multi-level Minimization of Reed-Muller Representations

Jonathan Saul
Department of Electrical and Electronic Engineering, University of Bristol, Bristol BS8 1TR, England.

Abstract
There is interest currently in using Reed-Muller equations as a way of representing and manipulating switching functions, and as a means of designing circuits based on exclusive-OR gates. There are only two-level Reed-Muller minimizers in use, although the need for a multi-level minimizer has been identified. A procedure for multi-level Reed-Muller minimization has been developed, which introduces a Reed-Muller factored form, and uses algebraic algorithms for factorization, decomposition, resubstitution, collapsing, and extraction of common cubes and sub-expressions. The procedure has been implemented in C as a series of packages which have been added to MISII, and benchmark comparisons with minimal two-level representations are favourable.

1 Introduction
There are two reasons for the current interest in using the Reed-Muller representation in digital circuit design: firstly as a means of designing classes of circuits based on exclusive-OR gates; and secondly as an efficient way of representing and manipulating functions.

Implementations based on exclusive-OR gates are often used in arithmetic and communications circuits, coding schemes for error control, encrypting schemes, system testing, and other applications. Exclusive-OR circuits can be more economical in these applications [1]. Their testability is also very good; they can be used to produce self-testing circuits, or circuits with function independent testing [2]. These circuits are often poorly designed using logic synthesis systems based on the sum of products representation; the Reed-Muller representation can easily lead directly to an efficient exclusive-OR implementation. There is the prospect of exclusive-OR circuits becoming faster and smaller in the future with the availability of optical technologies and current-switching techniques, making these implementations more attractive.

Recently there has been interest in using mixed polarity Reed-Muller equations as a way of representing and manipulating functions, rather than as a direct basis for implementation [3]. For this application mixed polarity Reed-Muller equations have a number of advantages over the more common sum of products representation: they can be considerably more compact — the most dramatic example is for the n-bit parity function, where a minimal sum of products representation requires \(2^{n-1}\) terms, whilst the Reed-Muller representation requires only \(n\) terms. Also many logic operations, such as complementation and Boolean difference, can be performed more easily.

The Reed-Muller representation is not yet popular in these applications because of a lack of efficient algorithms for circuit design and function manipulation. Recently, improved algorithms have been proposed for two-level mixed polarity Reed-Muller minimization [4, 5]. There has been interest in the use of AND/XOR PLAs, which can implement some functions using fewer product terms than AND/OR PLAs [1]. Because of the large size of exclusive-OR gates with more than a few inputs a more attractive way of implementing Reed-Muller representations is as multi-level circuits. The need for algorithms to restructure Reed-Muller representations as multi-level circuits has already been identified for both exclusive-OR circuit design [4], and function manipulation [3]. There are no multi-level Reed-Muller minimizers known at the present time.

This paper describes a new procedure for the multi-level minimization of Reed-Muller representations. Section 2 describes the definitions used in the rest of the paper, and sections 3 details the algorithms used. Section 4 describes the implementation, and the results of benchmark comparisons with two-level minimizers are contained in section 5.

2 Background and Definitions
There are three forms of Reed-Muller expansion: the positive polarity form, which is an exclusive-OR sum of products where each variable is uncomplemented; the fixed polarity form — or generalized Reed-Muller expansion (GRM) — where each variable may appear complemented or uncomplemented, but not both; and the mixed polarity form, which allows both polarities. The mixed polarity form is the most compact, and leads directly to efficient implementations, and so is the form used by the minimization procedure described in this paper.

A number of the terms commonly used in multi-level sum of products minimization will be used in this paper; definitions and a comprehensive list are given in [6]. In particular, the primary divisors of an expression \(f\) are the expressions \(f/c\) where \(c\) is a cube; an expression is cube-free if no cube divides the expression evenly (ie without a remainder); the kernels of an expression \(f\) are the cube-free primary divisors
of the expression; and the cube \( c \) used to obtain the kernel \( k = f/c \) is called the co-kernel of \( f \).

A mixed polarity Reed-Muller form is an exclusive-OR sum of AND product terms. These product terms can be represented using the cubic notation and on a Karnaugh map, in the same way as sum of products representations. However, there are important differences because the relationship between overlapping product terms is now exclusive-OR rather than inclusive-OR: minterms which are covered by an even number of product terms are 0 minterms; product terms are not implicants of the functions; there are no primes; and there are no essential cubes.

Linking rules are operations which join two cubes. Helliwell and Perkowski proposed two linking rules called primary and secondary xlinking, which join cubes by a chain of overlapping terms covering intermediate 0s twice; these are shown in Figure 1. Primary xlinking joins cubes which are of the same dimension; secondary xlinking joins cubes that differ in dimension by one. Making all possible primary xlinks guarantees the input irredundancy of the representation [7].

3 Minimization Algorithm

3.1 Introduction

This section is organised as follows: the Reed-Muller factored form — which is used to estimate the area of the final circuit — is introduced; then the algorithms used for restructuring the logic network into a more efficient form are outlined; and finally the algorithms that minimize the function at each node are described.

3.2 Factored forms

Reed-Muller equations represent a two-level network; a more useful way of representing multi-level logic would be a factored form. In this paper a Reed-Muller factored form is defined, which is an extension of the definition of the sum of products factored form. This form has the usual advantages of factored forms: it is more compact than the straightforward Reed-Muller equation; and corresponds more directly to the way a function will be implemented, allowing a more accurate measure of circuit complexity. It is an exclusive-OR sum of products of exclusive-OR sums of products, of arbitrary depth. For example,

\[
F = \overline{ac} \oplus \overline{ad} \oplus \overline{bc} \oplus \overline{bd} \oplus e
\]

can be factored to

\[
F = (a \oplus k) \cdot (c \oplus d) \oplus e.
\]

In multi-level logic synthesis systems based on the sum of products representation the number of literals in the factored form is often used as an estimate of the area that would be required by an implementation of the circuit. This is not a good estimate for the Reed-Muller form. For example:

\[
F = a \cdot \overline{c} \cdot \overline{d}
\]

and

\[
G = a \cdot (\overline{b} \oplus c \oplus d)
\]

have the same number of literals, but \( G \) has a far more expensive implementation because exclusive-OR gates are so much larger than AND gates. The Reed-Muller factored form can be used to estimate the area of an implementation using the weighted sum of the number of AND and exclusive-OR operations.

3.3 Logic Restructuring

The objective of multi-level optimization is to find a good multi-level representation of a logic function that minimizes a cost function that is dependent on area and delay. The optimization procedure presented in this paper provides algorithms to manipulate Reed-Muller representations of the logic function. These algorithms can factorize the logic function, decompose it, extract common cubes and sub-expressions, substitute one function into another, and collapse one function into another. Previously there were no algorithms to perform these operations on Reed-Muller representations.

Algebraic algorithms do not make use of Boolean identities (such as \( x \cdot x = x, x + x = x, x, x' = 0 \) etc). The key idea behind the algorithms presented in this paper is that techniques developed for sum of products expressions (based on the set of operators \( \{+, \cdot\} \) can be applied to Reed-Muller expressions (based on the set of operators \( \{\oplus, \cdot\} \) provided that they do not use the \( + \) operator. Algebraic algorithms fall into this category.

Using this observation the minimization procedure uses the weak division algorithm described in [6] to
divide a given function \( f \) by a given function \( p \) to give \( q \) and \( r \) so that \( f = q.p \oplus r \).

The factorization and decomposition algorithms are based on kernels. There are two algorithms provided for each operation: quick.fact and quick.decomp select one level 0 kernel as the divisor; and good.fact and good.decomp generate all of the kernels and select the one which maximally reduces the cost function.

The extraction algorithms are based on rectangle covering. Common sub-expressions are obtained by extracting rectangles from the co-kernel cube matrix. Overlapping rectangles can be used — which corresponds to non-algebraic factoring of the equations — but in a different way from sum of products kernel extraction: the relationship between overlapping rectangles is exclusive-OR, and so 1 entries in the matrix can be covered by an odd number of rectangles (using the identity \( a \oplus a \oplus a = a \)); 0 entries in the matrix can be covered by an even number of rectangles (using \( a \oplus a = 0 \)).

Common cubes are obtained by extracting rectangles from the cube literal matrix. Again overlapping rectangles can be used; this is valid because the relationship between overlapping rectangles is AND, and so the Boolean identity \( a.a = a \) has been used.

An expression \( y \) is substituted into an expression \( x \) by computing \( x/y \) and \( x/y' \) and making the substitution if the cost of either is less than the cost of \( x \).

### 3.4 Node Simplification

The goal of simplifying a node in a Boolean network is to replace it with an equivalent function with a smaller cost. Each node is minimized using a two-level minimizer, which reduces the estimated area of the circuit. The aim of node minimization of Reed-Muller representations is to minimize the number of literals and exclusive-OR operators after two-level minimization. The two-level equations were then minimized using the algorithms described above. It allows the option of maintaining both representations, and the two-level representations are the same.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Two-level</th>
<th>Multi-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 5 \times 1 )</td>
<td>223</td>
<td>105</td>
</tr>
<tr>
<td>( 9 \times 1 )</td>
<td>507</td>
<td>94</td>
</tr>
<tr>
<td>( b w )</td>
<td>404</td>
<td>92</td>
</tr>
<tr>
<td>( d u k e 2 )</td>
<td>1762</td>
<td>172</td>
</tr>
<tr>
<td>( c l i p )</td>
<td>812</td>
<td>123</td>
</tr>
<tr>
<td>( c o n 1 )</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td>( f 2 )</td>
<td>36</td>
<td>8</td>
</tr>
<tr>
<td>( f 5 1 m )</td>
<td>132</td>
<td>36</td>
</tr>
<tr>
<td>( m i s e x 1 )</td>
<td>90</td>
<td>35</td>
</tr>
<tr>
<td>( m i s e x 2 )</td>
<td>191</td>
<td>11</td>
</tr>
<tr>
<td>( r d 5 3 )</td>
<td>66</td>
<td>15</td>
</tr>
<tr>
<td>( r d 7 3 )</td>
<td>237</td>
<td>68</td>
</tr>
<tr>
<td>( r d 8 4 )</td>
<td>411</td>
<td>101</td>
</tr>
<tr>
<td>( s a o 2 )</td>
<td>752</td>
<td>103</td>
</tr>
<tr>
<td>( v g 2 )</td>
<td>2352</td>
<td>230</td>
</tr>
<tr>
<td>( z 4 m l )</td>
<td>89</td>
<td>28</td>
</tr>
</tbody>
</table>

All times are in cpu seconds on a Sparcrstation IPC.

Table 1: MCNC Benchmarks.

Each remaining node is minimized separately.

Another node operation required in a minimization system based on the Reed-Muller representation is translation to and from the sum of products representation, to interface with existing systems and circuit descriptions. This program uses a fast recursive algorithm to make a representation disjoint; disjoint sum of products and Reed-Muller representations are the same.

### 4 Implementation

The algorithms described above were implemented in C as a series of packages linked to the Berkeley multi-level synthesis system MISII. This allowed Reed-Muller and factored Reed-Muller representations to be attached to each node of a Boolean network. The program can work normally as a sum of products based synthesis system; or it can generate Reed-Muller representations, delete the sum of products representations, and then perform multi-level Reed-Muller optimization using the algorithms described above. It allows the option of maintaining both representations at each node, using whichever can be implemented the most efficiently.

### 5 Results

There are no previous multi-level Reed-Muller minimization algorithms known. This program has been used to minimize some of the MCNC benchmarks, and the results compared with the best Reed-Muller representation that could be obtained using two-level minimizers [4, 5, 8].

The results are shown in Table 1. First the representation was minimized using a two-level minimizer; the left hand columns in the table show the number of literals and exclusive-OR operators after two-level minimization. The two-level equations were then minimized using the algorithm described in this paper.
Nodes of low value were eliminated, and each node was simplified using Hermes. Then kernels and cubes of high value, and successively lower values were extracted; and finally the equations were factored using good factor. The right hand columns in the table show the number of literals and exclusive-ORs in the factored form after this multi-level minimization, and the time it took in CPU seconds on a Sparcstation IPC. The algorithm produced considerably smaller representations than the two-level minimizers, in times similar to currently available multi-level sum of products minimizers.

6 Conclusions
A multi-level Reed-Muller minimization system has been produced; there were previously no such systems or algorithms available. It has been implemented, and benchmark comparisons with the best previous two-level minimizers show that the new method produces more compact representations. If these representations are directly implemented using exclusive-OR gates this will lead to smaller circuits.

A large amount of work still needs to be done to develop algorithms for logic synthesis systems based on the Reed-Muller representation. The following ideas are being considered:

- There is considerable scope for the improvement of two-level Reed-Muller minimizers, as was shown in [6]. This would allow the multi-level minimizer to start from a more economical equation, and to produce better results after node minimization.

- Node minimization would also be improved by using the don’t care conditions at the node. In sum of products systems the full don’t care set is prohibitively large for many practical circuits, because the observability of the node at the output is calculated using the Boolean difference, which is computationally expensive and produces very large representations. In a system based on the Reed-Muller representation the Boolean difference can be calculated easily and produces a compact representation, since it is just an exclusive-OR sum.

- Technology mapping algorithms must be developed to map from a set of Reed-Muller equations to a circuit by selecting gates from a library, and then this system can be compared to a sum of products system. It is important that this algorithm can map to a wide range of gates — if it relies solely on exclusive-OR gates then it is unlikely to produce more economical circuits than a sum of products system.

- Now that there are algorithms available for restructuring Reed-Muller as well as sum of products equations, it is possible to develop a system that uses both representations and restructures nodes using whichever is the least costly. Such a system would use a mixed Reed-Muller/sum of products factored form:

$$F = (a + b + cd).(c \oplus d) + e.$$  

This would improve existing sum of products based systems by allowing them to restructure the logic using exclusive-OR decompositions. The complexity of the circuit could be estimated more accurately, by taking account of factors in the factored form which could be economically implemented using exclusive-OR gates.

Acknowledgements
This work was jointly funded by Plessey Research and Technology, and the Science and Engineering Research Council. It benefited from the ideas and suggestions of Ed Pitty and Martin Abrahams at Plessey, Martin Bolton at SGS-Thomson, and Erik Dagless at Bristol University.

References