# Minimization of Parity-Checked Fault-Secure AND/EXOR Networks

M. Eggerstedt N. Hendrich K. von der Heide

Universität Hamburg, Fachbereich Informatik,
Troplowitzstr. 7, 22529 Hamburg

## Abstract

We present a new algorithm for the efficient realization of fault-secure and totally self-checking circuits for multi-output boolean functions using parity-checked AND/EXOR networks.

First results on the minimization of a large set of typical arithmetic and random functions and several MCNC benchmark circuits show that AND/EXOR networks often allow much better solutions than traditional AND/OR networks. The overhead for the realization of the parity output may be as low as a few percent.

## 1 Introduction

The realization of boolean switching functions using AND/EXOR networks is very attractive from a theoretical point of view. In fact, it has been conjectured that the AND/EXOR networks on the average require fewer product terms than AND/OR PLA's [10]. Another well known advantage of the AND/EXOR logic is the excellent testability.

However, up to now they were used in special application areas only, due to the lack of circuit technology and easily available synthesis tools. Recent progress in circuit technology makes the use of AND/EXOR logic more feasible. For example, FPGA families like the Xilinx 3000 allow the realization of any logic function with up to five inputs in each logic block, using a fast table-lookup approach. A five input EXOR gate therefore is as fast and compact as any other logic function with up to five inputs.

In this paper, we concentrate on the realization of fault-secure — parity checked — boolean networks using AND/EXOR networks in Reed-Muller-form. A new algorithm is presented that allows the minimization of multiple-output AND/EXOR networks under the constraint that the network is fault-secure. The overhead for the realization of the parity logic is typically less than 40 percent. This a great advantage over AND/OR networks, where, due to the effects of the OR gates, the only acceptable method to build a fault-secure network is to realize two seperate copies of the whole circuit.

The outline of the paper is as follows: First we give an overview of the relevant theory of AND/EXOR networks and we summarize some of the testability aspects of AND/EXOR circuits.

Then the concept of fault-secure and parity-checked networks is introduced and we show how parity logic can be realized efficiently in AND/EXOR networks.

Finally, we present REMIN, a new algorithm for the minimization of multiple-output boolean functions. For several classes of functions, the results of the REMIN minimization (with parity logic) are better than the corresponding AND/OR realizations without parity.

## 2 Background

We consider the realization of logic functions with AND/EXOR PLA's, where a function is presented by mod-2 sum-of-products. Unlike the standard AND/OR sum-of-product PLA's, where only two canonical forms exist, the possible number of representations of a function using AND/EXOR PLA's is very large [1].

The most important class of AND/EXOR networks is the *Reed-Muller canonical (RMC)* form [6],[4], and its superset, the *Generalized Reed-Muller* form. The *RMC* representation consists of positive product terms only:

$$f(x_1, x_2, \ldots, x_n) = \bigoplus_{i=0}^{2^n-1} a_i R_i$$

where $\oplus$ denotes the mod-2 summation over GF(2), the Galois field of two elements. The $a_i \in \{0,1\}$ are the coefficients of the expansion of the function $f$ using the *RMC* basis functions $R_i$, with $R_i = x_n^{e_n} x_{n-1}^{e_{n-1}} \ldots x_2^{e_2} x_1^{e_1}$ and $e_j \in \{0,1\}$, so that $e_n e_{n-1} \ldots e_2 e_1$ is the binary number that equals $i$. Following [9] we call the *RMC* basis functions $R_i$ *monoterms* and the usual Boolean product terms *minterms*.

The *Generalized Reed-Muller (GRM)* [1] form results, if any variable $\dot{x}_i$ may be used in positive $x_i$ or negative $\bar{x}_i$ polarity. If all variables are restricted to retain the same polarity (either positive or negative) in all monoterms, the resulting realization is called a *Consistent Generalized Reed-Muller (CGRM)* form.

Obviously, for $n$ variables there are $2^n$ possible polarities of the variables; therefore there exist $2^n$ CGRM forms for any function $f$.

This is the idea behind most of the algorithms to minimize AND/EXOR realizations, e.g. [2],[9],[8]. By correct choice of the polarities of the variables $\dot{x}_i$ it is often possible to drastically reduce the number of terms in the expansion of $f$. The choice of the polarities of the variables, so that the resulting expansion of $f$ has the minimal number of terms, is called the *minimal polarity.*

**Example 1:** Consider the RMC expansion of the NOR function $f = \overline{x_1 \vee x_2 \vee x_3}$:

$$\begin{aligned} f(x_1, x_2, x_3) &= 1 \oplus x_1 \oplus x_2 \oplus x_1 x_2 \oplus x_3 \\ &\quad \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 \\ &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \end{aligned}$$

Using positive polarities, the expansion of $f$ uses all monoterms, while with negative polarity only one term is required.

**Example 2:** A 1-bit full-adder with inputs $a, b$, carry input $c_{in}$, and outputs $s, c_{out}$ has the RMC realization:

$$\begin{aligned} s &= a \oplus b \oplus c_{in} \\ c_{out} &= a\, b \oplus a\, c_{in} \oplus b\, c_{in} \end{aligned}$$

In fact, several important functions, most notably adders and multipliers, are minimal in positive RMC form.

## 3 Testability of AND/EXOR Networks

A well known advantage of the AND/EXOR networks is their high testability. It has been shown [7] that all single-stuck-at faults of an RMC network can be detected (assuming the primary inputs are fault free) with a set of $(n + 4)$ test patterns, and independent of the function. Note that the test complexity of functions in the usual AND/OR realization is $O(2^n)$.

With faulty primary inputs, a set of $(n+4)$ test patterns exist if an extra observable AND gate is added to the network. Even multiple-stuck-at faults can be detected using universal test patterns, if additional observable outputs are added to the network [5].

The testability of networks with parity depends on the realization of the parity function. Obviously, the test complexity of the whole network with parity is still $(n + 4)$, if the parity function is realized as a separate RMC function.

## 4 Fault-Secure Networks and Parity Checking

For several application areas the correct function of the electronical components is an essential requirement. While it may be prohibitive in terms of costs to build a fully fault-tolerant system, one way to increase the confidence into the system is to build it *fault-secure.* For a discussion and related definitions see e.g. [3]. The idea is to use a network that checks itself, and that signals an error if it fails.

**Definition 1:** A circuit with input code $\mathcal{C}_i$ and output code $\mathcal{C}_o$, $f : \mathcal{C}_i \mapsto \mathcal{C}_o$, $x \rightarrow y = f(x)$ is said to be *fault-secure* with respect to a set of faults $\mathcal{F}$, iff

$$\begin{aligned} &\forall f_{error} \in \mathcal{F}, x \in \mathcal{C}_i, f_{error}(x) \in C_o \\ &\Longrightarrow f_{error}(x) = f(x) \end{aligned}$$

That is, a function is called fault-secure, if under a fault its output values either remain correct or result in illegal output values $y \notin C_o$ which can be detected.

**Definition 2:** A boolean function is *self-testing,* if for every fault from a given set of faults, the circuit produces a non-code word for at least one input code word.

**Definition 3:** A circuit that is both *fault-secure* and *self-testing* is called *totally self-checking.*

Therefore a parity-checked fault-secure circuit with $n$ inputs is totally self-checking, if all $2^n$ input combinations are allowed.

In the following we consider only single (stuck-at) faults. While the single stuck-at fault model may be inacceptable during fabrication chip test, it is generally agreed to be adequate during the life time of the circuit, due to the small probability of errors.

The simple method to realize a single-fault fault-secure circuit is to use a parity-code and to realize all outputs $y_1, y_2, \ldots, y_m$ and the parity output $p$ independently from each other. Any fault will then result in only one faulty output signal and therefore be detectable by the (self-checking) checker circuit, which simply compares the (even) parity

$$\mathcal{P}(y_1, \ldots, y_m) = \bigoplus_{i=1}^{m} y_i$$

with the parity output $p$. A sketch of this arrangement is shown in figure 1. The first checker circuit guarantees that the inputs to $f$ are correct.

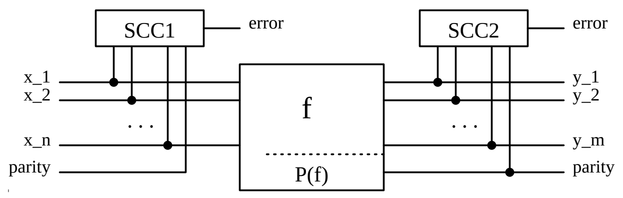


Figure 1: A fault-secure circuit with parity and two self-checking checkers. If all input combinations to $f$ are allowed, the circuit is totally self-checking.

Obviously it is very inefficient to realize every single output function independent from all others. The alternative is to double the whole circuit and to compare the corresponding output signals — with AND/OR realizations this is usually the most compact method to design a fault-secure circuit.

## 5 Efficient generation of the parity in AND/EXOR networks

The AND/EXOR networks offer better solutions. First note that changing any one input of an $m$ input EXOR gate will change its output, due to the mod-2 summation operation. Therefore a single fault in the AND plane of an AND/EXOR network with parity checking is still detectable if it affects an odd number of output signals — because in this case the fault will change the parity of the outputs.

Therefore it is possible to realize any monoterm in the RMC expansion of a fault-secure multiple-output function only once, if it is used in an odd number of outputs.

To generate the parity output one has to add additional logic to the network. It is easy to see that the even parity of an $m$ output function in RMC form can be written

$$\begin{aligned}\mathcal{P} &= \bigoplus_{j=1}^{m} y_j = \bigoplus_{j=1}^{m} \bigoplus_{i=0}^{2^n-1} a_{ji} R_i \\ &= \bigoplus_{i=0}^{2^n-1} (\bigoplus_{j=1}^{m} a_{ji}) R_i\end{aligned}$$

where the term in parentheses counts how often (mod-2) the coefficient $a_{ji}$ is used in the RMC expansion.

Unfortunately, it is not possible to realize the parity output $p$ of a fault-secure network using this formula directly, $p = \mathcal{P}$, because each term $R_i$ would be used in an even number of outputs — in $f$ and in $p$. An error would change the values of both $f$ and $p$ and could not be detected.

We therefore introduce the function $q$,

$$q = \bigoplus_{i=0}^{2^n-1} (1 \oplus \bigoplus_{j=1}^{m} a_{ji}) R_i$$

and use the identity (see example 1)

$$q = \bigoplus_{i=0}^{2^n-1} R_i \oplus p = NOR(x_1, \ldots, x_n) \oplus p$$

That is, all monoterms that are used in an even number of outputs of $f$ are included in $q$. There are three possible fault-secure realizations of $p$ now (similar results hold for the realization of odd parity):

1. All monoterms that are used in an even number (including zero) of outputs are included in $q$ and the parity output is realized as $p = q \oplus NOR(x_1, \ldots, x_n)$.
2. Only monoterms that are used in an even number $\geq 2$ are included in $q$, and $q$ is realized using a seperate AND/OR or AND/EXOR network. The parity output again is $p = q \oplus NOR(x_1, \ldots, x_n)$.
3. $p$ is realized as $f_{m+1}$. Any monoterm that is used in an even number of outputs is realized twice and each copy is used in an odd number of outputs.

There remains the question which alternative will be the best for a given function $f$. If $f$ uses all or nearly all monoterms, the first realization will probably be the most compact, as only one extra NOR gate is required. If $f$ needs only a few monoterms, the third realization will be the best — only some of the few terms will be used an even number of times and therefore need to be doubled. The second alternative needs the seperate realization of the sum of several monoterms.

## 6 The REMIN algorithm

The discussion of the previous section shows that even in a fault-secure (totally self-checking) realization of an AND/EXOR network different outputs can share monoterms.

A generalization of this idea leads to our REMIN algorithm: Different outputs can share EXOR gates too. That is, if the same pair of monoterms $R_k \oplus R_l$ is used in an odd number of outputs $\{f_{j_1}, f_{j_2}, \ldots f_{j_{2l+1}}\}$, it is possible to realize the pair once as *macro term* $M_{kl}$ — and the network still remains fault-secure.

The interesting problem now is to select the mono- and macroterms which should be combined into *macro terms*, so that the resulting AND/EXOR network is minimal. Note that the number of possible macro terms is very large, on the order of $O(2^{2n})$ for an $n$ input network.

Obviously, it will be best to combine those terms which are used most often (and in an odd number). On the other hand, a decision for one macro term may exclude further combinations. The simple heuristic used by REMIN therefore tries to combine terms $R_k$, $R_l$ that are used in several outputs, but allow only few combinations with other terms:

```
The REMIN algorithm:
{
  read the function table y = f(x);
  generate the functions q and/or p;
  calculate the number k of required monoterms R_i
    and initial costs Φ_2;
  generate an array T[k,k];
  foreach( pair of monoterms R_i < R_j) do {
   T[i,j] =number of uses of (R_i, R_j);
  }
  do {
    Φ_1 = Φ_2;
    select the pair R_i, R_j:
      – with highest number of uses and
      – least coupling to other pairs from T
    generate the macro term M_{i,j}
    replace all uses of R_i, R_j with M_{i,j}
      and update T;
    calculate actual costs Φ_2;
  } while( Φ_2 < Φ_1 );
  realize the network using the M_{i,j} macro terms;
}
```

## 7 Results

The algorithm presented above was implemented using C. The REMIN program accepts standard PLA files or RMC expansions of switching functions as input and generates a minimized positive RMC AND/EXOR realization.

The program was run on a variety of test circuits, including several MCNC benchmark circuits. Some results of the REMIN minimization are shown in table 1 and compared with two-level realizations found by ESPRESSO. For each function ($n$ inputs, $m$ outputs) the number of minterms required by ESPRESSO and the number of positive RMC monoterms required by REMIN are listed.

The cost estimations are gate-inputs costs. To allow a comparison with other algorithms, we quote both the costs of a CMOS realization (where an $n$-input EXOR gate is counted as $4(n-1)$ gate-inputs, an $n$-input AND or OR gate as $n$ gate-inputs) and the FPGA realization (where typically the costs of $n$-input EXOR gates are the same as for $n$-input AND/OR gates).

Some comments are in order. We show only one single-output function (xor5) — good algorithms for GRM or CGRM minimization of single-output functions already exists, e.g. [9], [8]. Rather, we concentrate on typical arithmetic functions, large random functions and several multi-output functions from the MCNC benchmark set.

The positive RMC AND/EXOR networks are especially efficient for adder circuits. Even with parity outputs and CMOS realization, the adders are more compact than the two-level solutions found by ESPRESSO. However, the realization of the parity output actually requires to double the whole adder circuit, because all needed RM coefficients are used once. Note that ESPRESSO exceeded the available memory on circuit add169.

As was to be expected, the REMIN results are best for circuits with many outputs. Note that the overhead for the realization of the parity output in the large random circuits is very small, e.g. just 14% for ran824. The AND/EXOR realizations using CMOS technology of the random circuits are somewhat larger than the AND/OR PLA's, while the AND/EXOR realizations using FPGA technology are clearly more compact.

For some of the MCNC benchmark circuits (see e.g. clip), the realization with REMIN is not satisfactory. In fact, most of these circuits allow efficient reduction of the number of CGRM terms (a minimization currently not tried by REMIN), as recent results from [9] show.

## 8 Conclusion

We have presented how single-fault fault-secure and totally self-checking circuits can be realized efficiently using parity checked AND/EXOR PLA's.

A new algorithm to minimize totally self-checking multi-output AND/EXOR networks was introduced. First results on the realization of several test and benchmark circuits are promising. While doubling the whole circuit usually is the best way to implement a fault-secure AND/OR network, the realization of the parity function in AND/EXOR circuits often requires a few percent overhead only.

Currently we study the use of don't care information for the minimization algorithm, and we will integrate REMIN with a recently proposed very fast algorithm [8] intended to minimize the number of monoterms. This will allow efficient fault-secure realizations for most classes of boolean functions using parity checked AND/EXOR networks.

| Name | $n$ | $m$ | ESPRESSO | | REMIN | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | with parity | | | without parity | |
| | | | terms | costs | terms | costs (FPGA) | (CMOS) | (FPGA) | (CMOS) |
| 5xp1 | 7 | 10 | 65 | 362 | 61 | 496 | 853 | 287 | 473 |
| apex4 | 9 | 19 | 437 | 5483 | 445 | 4501 | 10488 | 3417 | 8839 |
| clip | 9 | 5 | 119 | 784 | 217 | 2229 | 3609 | 1183 | 1785 |
| ex10 | 10 | 10 | 720 | 7919 | 1023 | 9587 | 18686 | 5972 | 12290 |
| ex5 | 8 | 63 | 75 | 1956 | 256 | 2292 | 4924 | 1463 | 3252 |
| rd73 | 7 | 3 | 127 | 903 | 63 | 490 | 852 | 245 | 422 |
| rd84 | 8 | 4 | 255 | 2070 | 107 | 900 | 1524 | 450 | 756 |
| squar5 | 5 | 8 | 26 | 137 | 23 | 139 | 248 | 81 | 140 |
| xor5 | 5 | 1 | 16 | 96 | 16 | 10 | 32 | 5 | 16 |
| add43 | 4 | 3 | 11 | 43 | 8 | 36 | 68 | 18 | 30 |
| add64 | 6 | 4 | 31 | 147 | 17 | 102 | 184 | 51 | 86 |
| add85 | 8 | 5 | 75 | 415 | 34 | 264 | 444 | 132 | 214 |
| add169 | 16 | 9 | — | — | 518 | 8208 | 11276 | 4104 | 5622 |
| mul44 | 4 | 4 | 7 | 30 | 5 | 28 | 40 | 16 | 20 |
| mul88 | 8 | 8 | 127 | 907 | 97 | 961 | 1522 | 541 | 746 |
| mul1212 | 12 | 12 | 1990 | 20655 | 2047 | 26508 | 39154 | 15343 | 19474 |
| ran88 | 8 | 8 | 233 | 2320 | 255 | 2054 | 4156 | 1267 | 2692 |
| ran816 | 8 | 16 | 256 | 3957 | 256 | 2390 | 5939 | 1813 | 5008 |
| ran824 | 8 | 24 | 256 | 5018 | 256 | 2610 | 7290 | 1986 | 6248 |

Table 1: REMIN minimization results, compared with ESPRESSO. The first 9 functions are from the MCNC logic-synthesis benchmark set. Costs are gate-inputs, see the text. The REMIN circuits with parity output are totally self-checking. Double the ESPRESSO costs for totally self-checking AND/OR realizations.

## References


[1] M. Davio, J. P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill, 1978

[2] M. Helliwell and M. Perkowski, A Fast Algorithm to Minimize Multi-Output Mixed-Polarity Generalized Reed-Muller-Forms, *Proc. 25th DAC*, 427–432, 1988

[3] P. K. Lala, *Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall, Englewood Cliffs, New Yersey, 1985

[4] D. E. Muller, Application of Boolean Algebra to Switching Circuit Design and to Error Detection, *IRE Trans. on Elec. Computers*, Vol. EC-3, 6–12, 1954

[5] D. K. Pradhan, Universal Test Sets for Multiple Fault Detection in AND-EXOR Arrays, *IEEE Trans. on Computers*, C-27, No. 2, 181–187, 1987

[6] I. S. Reed, A Class of Multiple-Error-Correcting Codes and Their Decoding Scheme, *IRE Trans. on Inf. Theory*, Vol. PGIT-4, 38–49, 1954

[7] S. M. Reddy, Easily Testable Realization for Logic Functions, *IEEE Trans. on Computers*, C-21, No. 11, 1183–1188, 1982

[8] M. W. Riege and Ph. W. Besslich, Low-complexity synthesis of incompletely specified multiple-output mod-2 sums, *IEE Proceedings-E*, Vol. 139, 355–362, 1992

[9] A. Sarabi and M. A. Perkowski, Fast Exact and Quasi-Minimal Minimization of Highly Testable Fixed-Polarity AND/XOR Canonical Networks, *Proc. 29th DAC*, 30–35, 1992

[10] T. Sasao and P. Besslich, On the Complexity of Mod-2 Sum PLA's, *IEEE Trans. on Computers*, Vol. 39, 262–266, 1990