Design of a Self-Testing Checker for Borden Code

Stanislaw J. Piestrak
Technical University of Wroclaw
Institute of Engineering Cybernetics
50-370 Wroclaw, Poland
e-mail: sjp@ict.pwr.pl

Abstract
A Borden code is the optimal code capable of detecting t-unidirectional errors. In this paper, a new self-testing checker (STC) for Borden code is proposed. It is built of two blocks: a self-testing code-disjoint translator of the Borden code onto the 1-out-of-r code (r ≥ 4) and a well-known STC for the 1-out-of-r code. The translator is built of two multi-output threshold circuits with \([n/2]\) and \([n/2]\) inputs and some NOT-AND-OR circuit. The new checker has significantly better performance than the only STC for Borden code proposed by Jha.

1 Introduction
A unidirectional error (u-error) is a multiple error such that erroneous bits are of either 0 \(\rightarrow\) 1 or 1 \(\rightarrow\) 0 type, but not both at the same time. Such errors have been observed in modern digital devices such as PLAs, ROMs, and compact laser disks [1], [2], [3], [4]. Unidirectional error detecting codes (UEDCs) have been proposed e.g. in [5], [6], [7], and [2]. The unencoded codes such as m-out-of-n codes [6] and Berger codes [5] were proved optimal non-systematic and systematic codes, respectively, that detect u-errors of any multiplicity. However, the data used in a digital system are in many cases organized in bytes and stored, transmitted or transformed by separate units operating on bytes. Therefore, provided that only single hardware failures can occur and they are confined to independent units, instead of u-errors of any multiplicity only up to t u-errors are the most likely to occur (t is the byte length). As a result instead of an unordered code a less redundant (and hence cheaper) t-unidirectional EDC (t-UEDC) can be used. The other potential area of using t-UEDCs is in write-once memory systems such as optical disks [3], where they can be used to implement a low-cost control key scheme which guarantees detection of subversion to the memory system with a certain probability. The optimal t-UEDC is the Borden code [7] which is the only known non-systematic t-UEDC.

The use of EDCs allows for concurrent error detection in a digital system. However, the monitoring circuitry should guarantee its own reliable operation. This is achieved by using self-testing checkers (STCs) which are tested concurrently with their own operation by incoming codewords. The only STC for Borden code was proposed recently by Jha [12]. His design is built of the STCs for m-out-of-n codes. Since an STC does not exist for a 0-out-of-n code or n-out-of-n code the all-0's and the all-1's vectors are excluded from a particular Borden code.

The aim of this paper is to present a new STC for Borden code which not only offers a significantly better performance than the only known design from [12] but also does not have the above limitation.

2 Preliminaries
In this section we briefly explain the basic notions used throughout the paper.

Definition 1: An m-out-of-n code (m/n code) \(C_{m/n}\) is a code consisting of all binary n-tuples whose weight is m. Its capacity is \(|C_{m/n}| = \binom{n}{m}\).

Let \(n\) denote the codeword length and \(t\) the multiplicity of u-errors detected by a code.

Definition 2: A Borden code \(C(n, t)\) is a code which is the union of all \(m/n\) codes whose weight is congruent to \(\frac{n}{2}\) mod \((t+1)\), i.e.

\[
C(n, t) = \bigcup_{m=\lfloor n/2\rfloor \text{mod}(t+1)} C_{m/n}.
\]

Let \(F\) be the set of likely faults in a circuit \(H\) and \(f\) be a fault in \(H\).

Definition 3 [8], [9]: A circuit \(H\) is called self-testing checker (STC) if it is both:
- code-disjoint (CD), i.e. it maps the input code space to the output code space and the input non-code space to the output non-code space; and
- self-testing (ST) for a set of faults \(F\), i.e. for every fault \(f\) from \(F\) it produces a non-code space output for at least one code space input.

The 1/2 code forms the output code space of an STC. We assume that the set \(F\) consists of all stuck-at-\(z\) (s/z) single faults, \(z = \{0, 1\}\).

Let \(I = \{x_1, x_2, \ldots, x_n\}\) denote a set of n input bits and \(m\) denote a threshold.
Definition 4: A threshold function $T^m_n$ is a switching function of $I$, which has the value 1 if and only if at least $m$ out of $n$ input variables from $I$ are 1's, $1 \leq m \leq n$.

Definition 5: A multi-output threshold circuit $T^m_n$ is a circuit that implements all $n$ $T^m_n$ threshold functions of $n$ variables, $1 \leq m \leq n$.

The best performance offers the circuit $T^m_n$ realized as a special implementation of an $n$-input sorting network [10]. It is built of $Cell(n) = 4/4ln^2 n - log n + 4 - 2$ identical cells, each cell consisting of a pair of 2-input AND and OR gates with fan-out of two. It has $L(n) = 1/2log n(log n + 1)$ gate levels, and is tested for all single faults by Test$(n) = n + [n/2]$ tests.

3 STC for Borden Code

The new STC for Borden code $C(n,t)$, shown on Fig. 1, is built using two basic blocks: H1, a ST and CD translator of the code $C(n,t)$ into the 1/r code ($r \geq 4$), and H2, an STC for the 1/r code. If the input to the H1 is not a Borden codeword, then the output generated by H1 is not a 1/r codeword, and H2 signals an error condition by generating non-1/2 code output. H2 can be designed by many methods, e.g. proposed in [9] and [11]. To reduce the amount of hardware, $r$ should be kept as small as possible. We conjecture that our STC for any Borden code can be realized with $r = 4$ and therefore the optimal STC for the 1/4 code from [11] can be used. Now the design of the circuit H1 is the main problem. This is done by using the following general procedure which will be clarified by an example given in the next section.

1. Partition the set $I$ of $n$ input bits onto subsets $A$ and $B$ with $n_A = [n/2]$ and $n_B = [n/2]$ bits, respectively.

2. Partition the $C(n,t)$ code onto subsets $(j,k)$.

3. Generate the basic products $p(j,k)$ of the translator H1.

4. Identify all untestable input s/1 faults of the AND gates $p(j,k)$, e.g. by using a diagram such as one shown on Fig. 2.

5. Modify all AND gates $p(j,k)$ with untestable input s/1 faults to meet the ST and preserve the CD requirements.

6. Assign each modified product $p'(j,k)$ from $C(n,t)$ to exactly one of $r$ functions $y_i$ ($r$ is as small as possible but greater than 3) in such a way that the circuit meets the conditions for CD and ST.

4 Example

Here we will design an STC for the $C(8,2)$ code which has the structure from Fig. 1. The set of inputs $I = \{x_1, x_2, \ldots, x_8\}$ is first partitioned into two disjoint subsets $A = \{x_1, x_2, x_3, x_4\}$ and $B = \{x_5, x_6, x_7, x_8\}$.

With this partition of $I$, the input space $X$ of the checker (i.e. the set of all 2$^8$ input 8-tuples) can be represented as, $X = U(j,k)$, the union of 25 disjoint subsets $(j,k)$, where $(j,k)$ is the subset of input 8-tuples with exactly $j$ 1's on the bits from $A$ and exactly $k$ 1's on the bits from $B$, $0 \leq j \leq 4$, $0 \leq k \leq 4$.

Figure 2. Partitioning of the Borden code $C(8,2)$ and identification of untested s/1 faults.

Now the $C(8,2)$ codeword, which is $C(8,2) = C_{1/8} \cup C_{4/8} \cup C_{7/8}$, can be represented as the union of $q = 9$ disjoint subsets $(j,k)$

$C(8,2) = \{(0,1) \cup (1,0)\} \cup \{(0,4) \cup (1,3) \cup (2,2)\} \cup \{(3,1) \cup (4,0)\} \cup \{(3,4) \cup (4,3)\}$.

A graphical illustration of this partitioning is the diagram shown on Fig. 2. The row and column numbers correspond to $j$ and $k$, respectively. The entries representing subsets $(j,k)$ are marked with 1's and the empty entries represent the subsets $(j,k)$ of non-codewords.

The above partitioning of $C(8,2)$ translates easily to the structure of H1 built of two threshold circuits $T^{4a}$ and $T^{4b}$ (with the inputs from $A$ and $B$, respectively) which feed some NOT-AND-OR circuit. This is because the logic function which is 1 for any input from $(j,k)$ and 0 for any other input can be expressed as $p(j,k) = T^{4a}_{I}T^{4b}_{I+1}T^{4b}_{I+2}T^{4b}_{k+1}$. Table 1 lists the $q = 9$ basic products $p(j,k)$ which correspond to the legal subsets $(j,k)$.

The translator H1 is CD when each product $p(j,k)$ is a primary output of H1, i.e. when $r = q$. To reduce $r$ and hence the size of H2, the STC for the 1/r code, some products $p(j,k)$ can be OR-ed together. It was found that $r = 4$ was feasible for each of the many Borden codes that we have analyzed.

Now we will set conditions under which 1 is ST, i.e. when all single s/1 faults of H1 are tested by applying Borden codewords only. Consider the 4-input AND gate $p(j,k) = T^{4a}_{I}T^{4b}_{I+1}T^{4b}_{I+2}T^{4b}_{k+1}$. Any s/0 fault in $p(j,k)$ is
Table I. Basic and modified products used in the translator H1.

<table>
<thead>
<tr>
<th>SUBSET</th>
<th>BASIC PRODUCT</th>
<th>MODIFIED PRODUCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,1)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(0,4)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(1,9)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(1,3)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(2,2)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(3,4)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
<tr>
<td>(4,2)</td>
<td>$e_1 e_2 e_3$</td>
<td>$e_1 e_2 e_3$</td>
</tr>
</tbody>
</table>

An asterisk * denotes an input with an untested s/l fault.

Table II. Partitioning of the code and non-code subsets.

<table>
<thead>
<tr>
<th>$r^1$</th>
<th>(j,k)</th>
<th>$r^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>(0,1) (0,4) (1,9) (1,3) (2,2) (3,4) (4,2)</td>
<td>(0,1) (0,4) (3,4) (4,2)</td>
</tr>
<tr>
<td>$r_2$</td>
<td>(0,1) (0,4) (1,9) (1,3) (2,2) (3,4) (4,2)</td>
<td>(0,1) (0,4) (3,4) (4,2)</td>
</tr>
<tr>
<td>$r_3$</td>
<td>(0,1) (0,4) (1,9) (1,3) (2,2) (3,4) (4,2)</td>
<td>(0,1) (0,4) (3,4) (4,2)</td>
</tr>
<tr>
<td>$r_4$</td>
<td>(0,1) (0,4) (1,9) (1,3) (2,2) (3,4) (4,2)</td>
<td>(0,1) (0,4) (3,4) (4,2)</td>
</tr>
</tbody>
</table>

Tested by any codeword from (j, k). Since the s/l fault on the output of p(j, k) is tested by any test for any input line s/l faults, only the input line s/l faults are of our concern. Consider the s/l fault on the input line $T_{j,k}^a$, for brevity denoted by $T_{j,k}^a$. All tests for this fault are from any subset $(j',k')$ such that for any $X \in (j',k')$ $T_{j,k}^a(X) = 0$ and $T_{j',k'}^a(X) = T_{j,k}^a(X) = T_{j,k}^{a+1}(X) = 1$. This occurs if and only if $k' = k$ and $j' < j$. Similar reasoning and notation applies to s/l faults on three other input lines as well as for any other AND gate (also with less than four inputs).

The testability analysis of s/l input faults of the products $p(j,k)$ is easy by using a diagram from Fig. 2. All s/l faults untested by $C(8,2)$ codewords are identified with arrows. Any arrow leaving the (j, k) entry (marked with 1) is directed towards missing tests (codewords) for a particular input s/l fault of the product $p(j,k)$.

The only way to eliminate s/l faults untested by Borden codewords is to modify all products with such faults in the following way. Let $p'(j,k)$ denote the product derived from $p(j,k)$ in such a way that $p'(j,k) = 1$ not only for codewords from (j, k) but also for non-codewords $X$ from some subsets $(j',k')$. Figure 3 shows how such a modification of the product $p(2,2)$ influences the testability of its input s/l faults. Suggested modifications of the products $p(j,k)$ impose a new partition of the set $X = \cup(j,k)$ onto three following subsets: (1) $C(n,t)$—the set of subsets $(j,k)$ for which all modified products are 0's; and (3) $X_{nc1}$—the set of subsets $(j,k)$ for which at least two modified products are 1's. The translator H1 with modified AND gates preserves CD under the following conditions:

C1) for any $X \in C(n,t)$ H1 generates a 1/r codeword;
C2) for any $X \in X_{nc0}$ H1 generates the all-0's output;
C3) for any $X \in X_{nc1}$ H1 generates some i/r codeword, $i > 1$.

Figure 3. Test sets for the s/l faults of the AND gate:
(A) original—$p(2,2) = T_2^{a+1} T_4^{a+1} T_6^{a+1} T_8^{a+1}$; and
(B) modified—$p'(2,2) = T_2^{a+1} T_4^{a+1} T_6^{a+1} T_8^{a+1}$.

Figure 4. Modification map of the AND gates p(j,k).

Only the condition C3 needs special consideration. It is satisfied if:
1. Each subset $(j_e,k_e) \in X_{nc1}$ appears in at least two extended sets (or, graphically, the entry $(j_e,k_e)$ appears in at least two clusters);
2. For each subset $(j_e,k_e) \in X_{nc1}$ there are at least two functions $y_i$ and $y_j$ which are 1's for all non-codewords from $(j_e,k_e)$.

Examples of modifications that allow to meet the above requirements are shown on Fig. 4, where: (i) the entries with 1's correspond to the sets $(j,k) \in C(n,t)$; (ii) the entries with an asterisk * denote the subsets of non-codewords $(j_e,k_e) \in X_{nc1}$; and (iii) the empty entries
denote the subsets of non-codewords \((j, k) \in X_{\text{nc0}}\). All inputs with \(s/1\) faults not tested by the codewords of \(C(8, 2)\), identified earlier with the help of the diagram from Fig. 2, are marked in the second column of Table I. The third column of Table I lists the modified products obtained on the basis of Fig. 4. Table II shows the final assignment of the subsets \((j, k)\) from \(C(8, 2)\) to \(r = 4\) sets \(Y_i\) which guarantees ST and preserves CD of \(H1\). The functions of the translator \(H1\) are:

\[
\begin{align*}
 y_1 &= T_1^{4*} + T_2^{4*} \\
 y_2 &= T_2^{4*} T_3^{4*} + T_2^{-3^*} T_3^{4*} + T_2^{4*} T_4^{4*} \\
 y_3 &= T_2^{4*} T_3^{4*} T_4^{4*} \\
 y_4 &= T_3^{4*} T_4^{4*} T_5^{4*}
\end{align*}
\]

5 Parameter Estimation

The complexity measures of the new STC for the \(C(n, t)\) code can be estimated by using the following formulas:

- Gate count \(Ga(n, t)\):
  \[
  < Ga(n/2) + Ga(n/2) + (n + q + r) + Ga(1/r);
  \]

- Total input count \(In(n, t)\):
  \[
  < In(n/2) + In(n/2) + (n + 5q) + In(1/r);
  \]

- Gate level \(L(n, t) = L(n/2) + 3 + L(1/r);

- Number of tests (codewords):
  \[
  \text{Test}(n, t) = \max \{\text{Test}(n/2), q\};
  \]

where \(Ga(n/2), Ga(n/2), \ldots\), are the parameters of the threshold circuits \(T[n/2], T[n/2], \ldots\), and \(Ga(1/r)\), etc., are the parameters of the STC for the \(1/r\) code.

A circuit \(T^n\) implemented as a sorting network has:

\[
Ga(n) = 2C\text{ell}(n) \quad \text{and} \quad In(n) = 4C\text{ell}(n) \quad (\text{for more details, see Section 2}), \text{while the optimal STC for the 1/4 code from [11] has:} \quad Ga(1/4) = 8, In(1/4) = 16, \text{and} \quad L(1/4) = 3.
\]

The upper-bounds on the complexity of the NOT-AND-OR network are: \(n\) inverters, \(q\) four-input AND gates, and \(r\) OR gates with a total of \(q\) inputs. However, as an example of the circuit \(H1\) for the \(C(8, 2)\) code shows, the actual gate/input count can be essentially lower: \(12/22\) as estimated \(21/53\).

The figures given in Table III prove that the new checker for the \(C(8, 2)\) code significantly outperforms the checker from [12]. (We assumed that Jha's checker is built using optimal checkers: for the 4/8 code from [10], and the STCs for the 1/8 and 7/8 codes designed with the multi-level 3-pair 2-rail STCs.)

The superiority of the new STC can be proved for other Borden codes by the following argument: the same threshold circuits \(T[n/2]\) and \(T[n/2]\), which in our design contribute most to the checker complexity, are also used in every Jha's checker to implement the STC for the \([n/2]/n\) code. However, Jha's checker also uses the STCs for other \(m/n\) codes and some output circuitry which, as a whole, are significantly more complex than our NOT-AND-OR circuit and the STC for the 1/4 code.

6 Conclusions

A new efficient STC for Borden codes is presented. It is significantly less complex and faster than the only known STC by Jha and does not have the limitations of the latter. Its highly regular modular structure and easy testability make it particularly attractive for a VLSI implementation.

References


