Definition and Design of Strongly Language Disjoint Checkers

INGRID JANCSCH AND BERNARD COURTOIS

Abstract—This paper defines strongly language disjoint (SLD) checkers. SLD checkers are to sequential systems what strongly code disjoint checkers are to combinational systems. SLD checkers are the largest class of checkers such that, when associated with an SFS circuit, the totally self-checking goal may be achieved. The approach considered here is of a sequentially self-checking system where the inputs are sequences defined by a given input language and the outputs are sequences defined by a given output language. The checker similarly receives a given input language (the output language of the functional sequential system) and produces outputs belonging to an output language. These outputs are an indication of an error when they do not belong to the output language. The basic scheme used is shown in Fig. 1. The sequential machine is sequentially self-checking as defined by Vauiu and David in [4].

As in [4], \( M = (Q, X, Z, \delta, w, q_0) \) is assumed to be a sequential machine. \( Q \) is the set of internal states. \( X \) and \( Z \) are the input and output alphabets, respectively. \( \delta \) and \( w \) are the state transition function and the output function, respectively, and \( q_0 \) is the initial state. We use the Mealy machine definition for our considerations, but those may be easily applied to the equivalent Moore machine.

Under the fault \( f \), the machine \( M = (Q, X, Z, \delta, w, q_0) \) becomes \( M' = (Q', X, Z', \delta', w', q_0) \). Under normal operation, \( \delta(i, q) \) is the state reached from \( q \) when input sequence \( i \) is applied, and \( w(i, q) \) is the obtained output sequence. If a fault \( f \) occurs in the state \( \delta(i, q) \), it is supposed that \( M' \) reaches immediately the state \( q' \). Hence, \( \delta'(i, q') = q', q' \in Q', \) \( w(i, q) = w(i, q') \) and \( w(i, q') \) is the obtained output sequence. Let \( i = i_1 i_2 \), where ‘'’ denotes the concatenation. The input sequences \( i_1 \) and \( i_2 \) are said to be a prefix and a suffix of \( i \), respectively. \( P(i) \) and \( S(i) \) will denote the set of prefixes and the set of suffixes of \( i \), respectively. \( I_q \) is the set of input sequences which may be applied from that state \( q \) in normal operation: \( I_q = \{ i | \delta(i, q_0) = q, \delta(i, q_0) \in Q_0 \} \) defines the input sequences of nonbounded length. The shortest prefix of \( i_2 \), such that an output sequence not included in the output language of \( M \) is obtained in the presence of fault \( f \), is denoted by \( i_m \). Then, we have \( w(1, q_0) \cdot w(i_m, q) \not\in O_M \).

The concepts of input and output languages and of sequentially self-checking (SeSC) circuits are those defined in [4]. Now we give some basic concepts concerning sequential machines, which had not been given in [4], before defining sequential checkers.

Definition D1: A circuit \( M \) is redundant with respect to a fault \( f \), a state \( q \), and an input sequence \( i_2 \in I_q \), if \( w(1, q_0) \cdot w(i_2, q) = w(1, q_0) \cdot w(i_2, q) \).

Remark R1: In a circuit redundant with respect to a fault \( f \), a state \( q \), and an input sequence \( i_2 \in I_q \), \( i_m \) does not exist. Remind that \( i_m \) is the shortest prefix such that \( w(1, q_0) \cdot w(i_m, q) \in O_M \).

For sequential circuits, the TSC goal is the property which ensures
that the first erroneous output from the machine $M$ results in an output sequence not included in the output language $O_M$. The hypothesis that may be used in order to achieve the TSC goal is expressed as follows:

Hypothesis H1: Between the occurrence of any two faults belonging to $F$, a sufficient time elapses such that, for any $q \in Q$, for any $i1$ such that $b(i1, q) = q$ and such that $i1 \cdot i2 \in I_M$, all $e_{Q,q}$ with respect to each fault $f$ or sequence of faults are applied, whenever they exist.

It may be shown [6] that any sequentially self-checking (SeSC) circuit, under Hypothesis H1, achieves the TSC goal.

Remark R2.—(Remark on Hypothesis H1). The aim of Hypothesis H1 is to ensure the detection of the first fault or of a new one with respect to which the sequential circuit is not redundant before the occurrence of another fault. Another way to formulate Hypothesis H1 is to enforce the appearance of transition sequence (DTS), as defined by [7].

In this case, Hypothesis H1 may be formulated as:

"Between the occurrence of any two faults belonging to $F$, a sufficient time elapses such that for any $q \in Q$, for any $i2 \in I_q$, for any $i1$ such that $i1 \cdot i2 \in I_M$, and $b(i1, q) = q$, at least one DTS with respect to each fault or sequence of faults is applied. Note that, when $e_{Q,q}$ exists, $i2 \in I_q$, a transition sequence $q \cdot i2$ may be a DTS.

III. STRONGLY LANGUAGE DISJOINT CHECKERS

Strongly language disjoint (SLD) checkers defined in this paper will be different from the checkers used in the literature for the design of self-checking sequential machines. Strongly language disjoint checkers will check a sequential property, and consequently they will be defined as sequential checkers when combinational checkers have been used in the past for the design of self-checking sequential systems.

Several authors have studied the design of fail-safe or self-checking sequential systems (e.g., [8]-[12]) but in all these studies, the checker was a combinational circuit, checking in general the state's code. Others (like [13], [14], etc.) considered algebraic approaches or the use of reverse or parallel machines. In all these studies, the checker is combinational. On the contrary, SLD checkers are sequential checkers, to be used associated with a sequential machine, according to the general structure already represented in Fig. 1.

The first sequential checkers have been defined by Vidu and David [4] based on the following situation. Consider a functional part having a 2-bit output such that, in normal operation, this output is constituted by an alternation of even and odd number of zeros. This output language may be formally described as $O_M = \{00, 01, 00, 01, \ldots \}$. A fault-free sequence may be $00, 01, 00, 01, \ldots$. For this defined output of a functional circuit, a checker is a sequential circuit.

The language disjoint property had been introduced by [4], as a generalization of the code disjoint property.

Definition D2: A machine $M$ is language disjoint if $\forall i \in I_M$, $w(i, q) \in O_M$ and $\forall i \in I_M$, $w(i, q) \in O_M$.

Definition D3: A circuit is a sequentially self-checking checker if it is both sequentially self-checking and language disjoint.

The problem with these definitions is that nothing is said concerning the language disjoint property when faults are present inside the checker. A similar problem was presented by combinational checkers, when they were defined to be code disjoint, but not strongly code disjoint.

In the next definitions, we name, respectively, $Ic$ and $Oc$ the input and output languages of a checker, and $w(i', q')$ the output sequence obtained from $q' \in Q'$ on the checker. (The symbols are distinguished from those used for the functional block by a mark (') or by an index c.) Notice that, although defining an output language to be obtained in the outputs of the checker, this output language may be degenerated in a string of output codewords (for example, a double-rail code). But in the following, a sequential property at the outputs is considered in order to keep general definitions.

Definition D4: A sequential checker $C$ is redundant with respect to a fault, and with respect to an input language $Ic$ and with respect to an output language $Oc$, if $\forall i' \in Ic$, $\forall i'' \in P(i')$, $w(i', q) \cdot w(i'', q') \cdot w(i', q')' \in Oc$, where $i' = i'' \cdot i'''$ and $q'' = q'(i'' \cdot i''')$, and $\forall i \in Oc$, $\forall i'' \in P(i')$, $w(i', q) \cdot w(i'' \cdot q')' \in Oc$.

Definition D5: A sequential checker is strongly redundant with respect to the fault sequence $(f_1, f_2, \ldots , f_n)$, and with respect to an input language $Ic$ and with respect to an output language $Oc$, if it is redundant with respect to the $n$ fault subsequences $(f_1, f_2, \ldots , f_n)$, $(f_1, f_2, f_3, \ldots , f_n)$, $(f_1, f_2, \ldots , f_n)$, and with respect to the input language $Ic$ and with respect to the output language $Oc$.

Now it is possible to give the definition of strongly language disjoint checkers (SLD checkers).

Definition D6: Before the occurrence of any fault, the checker $C$ is language disjoint. For a fault sequence $(f_1, f_2, \ldots , f_n)$, a state $q'$, and an input sequence $i' \in Ic$, let $k$ be the smallest integer for which $b(i'^{\text{th}}, q) = q'$ and such that $i' \cdot i'' \in Ic$, $w(i', q) \cdot w(i'' \cdot q')' \in Oc$.

If there is no such $k$, let $k = n + 1$. Then $C$ is strongly language disjoint (SLD) with respect to the fault sequence if

$w(i'^{\text{th}}, q') \cdot w(i'' \cdot q')' \in Oc$.

Definition D7: The checker $C$ is strongly language disjoint (SLD) with respect to the fault set $F$ if $C$ is SLD with respect to all fault sequences whose members belong to $F$.

Checkers that are SLD with respect to class of faults are given by the following property.

Property PY1: If a checker is strongly language disjoint with respect to all sequences of faults $fi$ belonging to a class $C_f$ of fault hypotheses, it is strongly language disjoint with respect to the class $C_f$ of fault hypotheses.

Hypothesis H2: Between the occurrence of any two faults affecting the functional block, a sufficient time elapses such that for any $q \in Q$, for any $i1$ such that $b(i1, q) = q$ and such that $i1 \cdot i2 \in I_M$, and such that $i1 \cdot i2 \in I_M$, all $e_{Q,q}$ with respect to each fault $f$ or sequence of faults are applied to this block whenever they exist and no fault occurs in the checker. Between the occurrence of any two faults affecting the checker, a sufficient time elapses such that for any $q \in Q$, for any $i1$ such that $b(i1, q) = q$ and such that $i1 \cdot i2 \in I_M$, all $e_{Q,q}$ with respect to each fault $f$ or sequence of faults are applied to the checker whenever they exist, and no fault can occur in the functional block.

Proposition P1: A system composed of an SeSC circuit and an SLD checker achieves the TSC goal under Hypothesis H2.

This proposition is demonstrated in [6].

Comments on SLD checkers may be derived from their definition.

1) Strongly language disjointness ensures the mapping of input sequences included (and, respectively, not included) in the input language to output sequences included (and, respectively, not included) in the output language, even if faults are present inside. The SLD property ensures also the detection of the first fault with respect to which the checker is not redundant with the application of a sequence defined by the input language.

2) The sequentially self-checking property is not verified, but the TSC quality of the system is achieved.
3) The sequentially fault secureness is not necessary for checkers, in general, as the output values will not be used subsequently.

Stronger definitions may be given for SLD checkers.

**Definition D8:** Before the occurrence of a fault, the checker C is language disjoint. For a fault sequence \((f_1, f_2, \ldots, f_n)\), a state \(q'\), and an input sequence \(i' \in I_q\), let \(k\) be the smallest integer for which \(b(i_1, q_0) = q' \) and such that

\[ i' \cdot i_2' \in I_c : w_c(i_1', q_0') \cdot \hat{w}_c(j_1^{(j_2 \cdots j_m)}(i_2', q') \in O_c. \]

If there is no such \(k\), let \(k = n\). Then C is very strongly language disjoint (VSLD) with respect to the fault sequence if

\[ w_i' \in I_c, w_m' \in \{1, 2, \ldots, k\}, \]

\[ w_c(i_1', q_0') \cdot \hat{w}_c(j_1^{(j_2 \cdots j_m)}(i_2', q') \in O_c. \]

**Definition D9:** The checker C is very strongly language disjoint (VSLD) with respect to the fault set \(F\) if \(C\) is VSLD (by Definition D8) with respect to all fault sequences whose members belong to \(F\).

**Property P2:** If (but not only if) a checker is VSLD, then it is SLD.

Definition D9 is more restrictive than D7. It (D9) ensures that, even if a fault with respect to which the circuit is not redundant occurs, the language disjoint property is kept (i.e., an output sequence \(i\in O_c\) will be necessarily generated by an input sequence \(I_c\), while an input sequence \(\not\in I_c\) will always produce an output sequence \(\not\in O_c\)). This property allows us to consider a less strong hypothesis than H2 with a sequential system in order to achieve the TSC goal which means that it is admitted the occurrence of faults in the functional block (resp., in the checker) preceding the detection of another one, occurred before, in the checker (resp., in the functional block). Hence, “simultaneous” faults in the checker and in the functional block are admitted.

The relationships among the checkers, obtained from the proposed definitions, are depicted in Fig. 2. In this scheme, it may be observed that sequentially self-checking (SeSC) checkers that are sequentially self-testing (SeST), are the most restrictive group. The next one is the group of SeSC checkers which are not SeST. The checkers defined by D9 may be SeST and/or SeFS or just very strongly language disjoint, even in the presence of faults with respect to which the circuit is not redundant. That is why they are a subgroup of SLD checkers described by D7, which is the largest class of checkers.

### A. General Considerations

Strongly language disjoint checkers may be designed from traditional logical devices (i.e., using flip-flops and combinational logic or other approaches) or from regular structures, as PLA’s, registers, etc.

But, in a general manner, flip-flops seem not to be a good approach for the design of SLD checkers. Some types of flip-flops are inadequate, as input arrangements with “DON’T CARE” values are allowed; this behavior may hide some stuck-at faults or faults in the input language, for example. There are some difficulties also to ensure the propagation of certain faults to the outputs of flip-flops.

Another traditional approach used for the design of sequential circuits is that of asynchronous circuits, executed from simple logical gates and designed with basis in the fundamental mode (eliminating races). Such a design supposes that the primitive flow table is constructed with the corresponding flow graph and that the states are stable. The design of the complete circuit, from the transition maps, may lead to a good solution but it would be probably too fastidious, considering the designer’s point-of-view because 1) a new design will correspond to each new input language (which depends on the SeSC circuit); 2) each design may require a great number of transistors and the analysis of the SLD properties becomes hard and time consuming.

It is preferable to choose a solution where predefined blocks and/or regular structures may be used.

### B. Design of SLD Checkers Based on Regular Structures

SLD checkers cannot be assembled directly from basic cells, as it is the case of strongly code disjoint checkers [15]. But as they are sequential checkers, they may be decomposed internally in combinational and memorization blocks. Then, each one of these blocks may be designed using regular structures such as PLA's, registers, etc. According to the definitions, the outputs of the checker are included in an output language. This language may be particularized into a combinational double-rail code for the error indication. In the following, we will refer to such a particular system, having two double-rail coded outputs.

The inputs of the checker, which are defined by a language, come from the functional part. They are received by the internal combinational block, which executes the logic of the next state function—giving coded states, and the logic of the output function—giving the error indication. The memorization logic feeds back the combinational block and together with the general inputs constitutes the data for the succeeding next states function. Therefore, the complexity of the checker will depend on the sequential property of the input language to be checked, and not on the complexity of the functional system.

We consider a system based on the Mealy model, with a master-slave register and stable inputs (which come from the functional block). When the Moore model is considered, it will be explicitly noted.

The checker, being a finite state accepter, is language disjoint with respect to input language, and Hypothesis H2 may be reformulated. The LD accepter is such that is recognizes the language according to the fact that an input, belonging to an input sequence \(i\) appears when the present state belongs to some set of states.

Let \(B\) be the set of codewords, where the codewords are each constituted by the concatenation of an input \(x \in X\) (\(X\) is the set of input vectors) and a state \(q' \in Q'\), such that a word is a codeword iff there is an input sequence \(i' \in I_c\) corresponding to this word. The set of states being encoded, examples of noncodewords \(y \in B\) are:

- one input \(x_1 \in X\) that cannot appear in any sequence \(i' \in I_c\) when the state is \(s_j\),
- any input \(x_4 \in X\) when the state \(q' \in Q'\).

Now Hypothesis H2 may be reformulated as follows.

**Hypothesis H2:** Between the occurrence of any two faults affecting the functional block, a sufficient time elapses such that, for any \(q' \in Q\), for any \(i\) such that \(b(i, q_0) = q\) and such that \(i \cdot i_2 \in I_m\), all \(i_2\) with respect to each fault \(f\) or sequence of faults that may occur in the functional block are applied, whenever they exist, and no fault occurs in the deterministic finite state accepter (checker).

Between the occurrence of any two faults affecting the deterministic finite state accepter, a sufficient time elapses such that all codewords \(B\) are applied to the checker (i.e., sequences \(i' \in I_c\) applied to the checker are such that all codewords \(B\) are applied), and no fault occurs in the functional block.

In the next paragraphs, some propositions referring to internal properties of SLD checkers are given. A complete set of schemes including the respective demonstrations of the necessary internal properties are given in [6].

**Proposition P2:** For a partitioned design of an SLD checker according to Fig. 3 (Mealy model), the strongly language disjoint
property of the checker (as stated by Definition D6) will be ensured if it is composed of an SFS combinational logic for the next state function realization, an SCD combinational logic for the error indication output function (double-rail code), and an SFS/SCD* memorization logic, assuming Hypothesis H2* and that any fault can affect only one block.

The SCD property of the output function refers to the input code B (concatenation of the external input with the state code) and to the output code Oc. The SCD and SFS properties of the memorization logic refer to the state code C. Both refer to the large (weak) definition of the combinational SCD checkers. The next state function block is SFS with respect to the output code C.

Proposition P2 is demonstrated in [6].

The star is associated with the denomination SFS/SCD* to mean the use of combinational properties to circuits where time variables are also involved. While load enable and clock signals are not valid, the circuit outputs do not change. When the circuit is enabled by load/clock, the outputs have to be the copy of the input values, or they have to be a noncodeword (then the SFS property is ensured). The noncode inputs have to produce the same noncode outputs, even if faults with respect to which the circuit is redundant have occurred; or if a different noncode output is produced, the concatenation of this one with the external input word has to result in a noncode input word (B) to the SCD output function circuit. Therefore, we will keep the star to assign the delay that will be needed between the input application and the output production (depending on clock period and signal propagation). It is assumed that the clock and load circuitry lines do not fail—or that they are checked separately.

If the Moore model is used similarly to the architecture depicted by Fig. 3, the next-state function block needs to be SCD (in addition to the SFS property).

Other architectures for SLD checkers may be used, if required properties are ensured. The combinational functions may be combined in one single block with SFS/SCD properties, which with an SFS* memorization logic and an internal SCD checker (that checks states code) will compound another proposition. This one and others (including the design of VSLD checkers) are all demonstrated in [6].

In practice, the difficulties for the design of these checkers, as well as the area increasing (if compared to the original circuits) depend on the properties of the internal blocks and on the considered fault hypotheses. The rules for the design of blocks with SFS and SCD properties have been studied by [16] and [17], respectively. In these references, Class I of fault hypotheses is considered, and a complete coverage for the faults included in this class may be achieved if the rules are respected. The complexity of the checker and its additional cost will depend basically on the output language of the functional block; therefore, it is impossible to estimate these parameters as a function of the system complexity.

IV. CONCLUSION

In this paper, the largest class of checkers is defined—the strongly language disjoint checkers. They check inputs that follow sequential patterns and ensure the language disjoint property, even if faults with respect to which the circuit is redundant have already occurred. The SLD checkers, when associated with SeSC (sequentially self-checking) circuits, compose sequential systems that achieve the TSC property independently from the state during which the fault has occurred. Before this detection, the circuit outputs are the same as they would be during normal operation.

Concerning practical applications, the most important conclusion is the possibility of using regular structures for the implementation of these SLD checkers, such as PLA's and register cells. This is interesting as CAD resources may be employed for the design of these checkers. Propositions suggest the use of internal combinational blocks with SFS, SCD, and SFS/SCD properties for the design of checkers defined by the large (weak) or the strong definitions, without loss of generality for the global sequential properties.

The last point to be noted is that to be really efficient, the design of SLD checkers, and then the design of SFS, SCD, etc., blocks should be based on realistic fault hypotheses. Such realistic fault hypotheses should include MOS s-on/s-open, shorts, etc.; if an MOS technology is the target technology, such as NMOS or CMOS. Rules for the design of such NMOS [16] and CMOS circuits have been studied, and they are for the design of several projects, including large industrial ones.

REFERENCES

A New Bit-Serial Systolic Multiplier Over $GF(2^m)$

B. B. ZHOU

Abstract—A new bit-serial systolic array is developed to compute multiplications over $GF(2^m)$. In contrast to another systolic multiplier designed in [5], this new systolic algorithm allows the input elements to enter a linear systolic array in the same order and the system only requires one control signal.

Index Terms—Finite field, finite field multiplication, Massey-Omura multiplier, systolic array, VLSI.

I. INTRODUCTION

Finite field arithmetic operations play a very important role in certain error-correcting codes. Multiplication in $GF(2^m)$ is one among those most complex and time-consuming operations. Recently, two VLSI architectures, systolic architecture [5] and the Massey and Omura algorithm-based pipeline architecture [4], were developed to compute multiplications in $GF(2^m)$. The Massey and Omura algorithm utilizes a normal basis $\{a, a^*, \alpha^4, \cdots, \alpha^{2^{m-2}}\}$ to represent elements in the field $GF(2^m)$, where $\alpha$ is a root of an irreducible polynomial of degree $m$ over $GF(2)$. In this basis representation, the squaring of an element in $GF(2^m)$ is a simple cyclic shift of its binary digits followed by an Exclusive-OR operation. Based on this important property, multiplication for any one product digit requires the same logic circuitry as it does for any other product digit. The advantage of this design is that it takes less area than that required in systolic designs. However, it is much less expandable because irreducible polynomials for different $m$ are not related, but the design only focuses on one particular $m$ each time. Since systolic arrays have the advantages of regularity, modularity, and expandability [1], systolic designs are still worthy of consideration.

The systolic multiplier of [5], which uses a conventional basis, has two disadvantages. One is that it does not allow the two input elements to enter the system in the same order. If this order of the elements is essential, extra registers and control signals are required. The other is that it requires two control signals, which increases the chip area.

A new systolic algorithm for multiplications over $GF(2^m)$ will be discussed here. This algorithm allows the input elements to enter a linear systolic array in the same order and the system only requires one control signal.

II. ALGORITHM

Using a conventional basis representation, let

$$A = \sum_{k=0}^{m-1} a_k \alpha^k$$

and

$$B = \sum_{k=0}^{m-1} b_k \alpha^k$$

be two elements in $GF(2^m)$,

$$C = \sum_{k=0}^{m-1} c_k \alpha^k$$

be the product of $A$ and $B$, and

$$F = \sum_{k=0}^{m} f_k \alpha^k \quad (f_m = 1)$$

be the irreducible polynomial.

The equation $F = 0$, or $\alpha_m = \sum_{k=0}^{m-1} f_k \alpha^k$ is used to reduce the product $C = AB$ to a polynomial of degree less than $m$ [2]. $C$ can be written as follows.

$$C = AB$$

or

$$C(0) = Ab_{m-1},$$

$$C(1) = Ab_{m-2} + Ab_{m-1} = C(0) + Ab_{m-2},$$

$$\vdots$$

$$C(i) = C(i-1) + Ab_{m-1-i}$$

$$C = C(m-1) = C(0) + Ab_0.$$  \hspace{1cm} (1)

Using the fact that

$$\alpha^m = \sum_{k=0}^{m-1} f_k \alpha^k,$$

$C(0)$ in (1) can be reduced to a polynomial of degree less than $m$. $C(0)$ in (1) can be rewritten as

$$C(0) = Ab_{m-1} = \sum_{k=0}^{m-1} a_k b_{m-1-k} \alpha^k = \sum_{k=0}^{m-1} C_k(0) \alpha^k$$  \hspace{1cm} (2)

where $C_k(0) = a_k b_{m-1-k}$.

Suppose

$$C(1) = \sum_{k=0}^{m-1} C_k(1) \alpha^k,$$

which has been reduced to a polynomial of degree less than $m$. Then

$$C(1) = C(1) + Ab_{m-1-i}$$

$$= \sum_{k=0}^{m-1} C_k(1) \alpha^k + Ab_{m-1-i}$$

$$= C(1) \alpha^m + \sum_{k=0}^{m-2} C_k(1) \alpha^k + Ab_{m-1-i}$$

$$= \sum_{k=0}^{m-1} C_k(1) f_k \alpha^k + \sum_{k=0}^{m-1} a_k b_{m-1-k} \alpha^k$$

$$= (C(1) + Ab_{m-1} + Ab_{m-1-i})$$

$$+ \sum_{k=1}^{m-1} (C_k(1) f_k + a_k b_{m-1-k}) + C_k(1) \alpha^k$$

$$= \sum_{k=0}^{m-1} C_k(1) \alpha^k.$$  \hspace{1cm} (3)