ISSN:1983 7402 ITA, 29 SET a 01 OUT de 2015

Two-Level Hazard-Free Logic Minimization of Speed-Independent Extended Burst-Mode Controllers

Duarte L. Oliveira and Lester A. Faria
Divisão de Engenharia Eletrônica do Instituto Tecnológico de Aeronáutica – ITA – IEEA – SJC – SP – Brazil

***Abstract* — Several algorithms of hazard-free logic minimization were proposed for extended burst-mode asynchronous controllers that operate in the generalized fundamental mode (GFM) and accept the so-called extended burst-mode (XBM) Specification. This class of controller's shows to be very interesting because the XBM specification allows describing a wide range of circuits, from complex asynchronous interfaces to Moore-type synchronous controllers that operate in the two edges of the clock signal. Another interesting class of asynchronous controllers is the speed-independent (SI) ones. They are more robusts, more modulate and easy to be verified, when compared with GFM XBM counter-parts. Considering and conjugating the advantages provided by both approaches, in this paper we propose an algorithm for hazard-free logic minimization, exact in the number of literals, for SI asynchronous controllers that starts from the XBM specification, being highly suitable for the implementation on high accuracy, robust and complex systems, like the ones actually embedded in military systems and platforms. The proposed algorithm has been successfully applied to a set of known benchmarks showing good and potential results for practical implementation.**

***Key-words* — finite state machine, asynchronous logic, hazard, XBM specification**

I. Introduction

Currently, a considerable interest has been given to asynchronous design [1] due to the increasing complexity of digital systems and the need for the improvement of their performance. An important component in the asynchronous design is the asynchronous controller. One class of asynchronous controllers is asynchronous finite-state machines, originally proposed by Huffman [1], as well as their extensions, that accept a specification called (extended) burst-mode (XBM/BM) [2]-[5]. These controllers operate according to the bounded gate and wire delay model and interact with the environment in the generalized fundamental mode (GFM), where transition from stable states can occur both for single or multiple input and output changes. The XBM asynchronous controllers are an interesting class of controllers because the XBM specification allows describing a wide range of controllers, from complex asynchronous interfaces to synchronous ones that operates in the two transitions of the clock signal (heterogeneous systems).

Duarte L. Oliveira, duarte@ita.br, Tel +55-12-3947-6813, Fax +55-12-3947-6930; Lester A. Faria, lester@ita.br

In recent years, there have been a number of successful and efficient real-life asynchronous circuits designed as extended burst-mode asynchronous controllers [6]. In contrast, Speed-independent (SI) asynchronous controllers [7,8] are also attractive, because they can be proven correct and tolerate delay variations, resulting from technology migration or temperature variations. These controllers work correctly regardless of the delays of the individual gates, while assuming constant wire delays for multiple fanouts [8]. As a result, achieving speed-independent circuits avoids the need for timing assumptions and delays lines that tend to increase area and reduce the reliability of the circuit (hazard-free essential). Speed-independent modules can thus easily be replaced whenever a faster or more advanced technology is introduced. In addition, speed-independent controllers have a so called "testability advantage", once they are self-checking with respect to output stuck-at-faults [8].

An important step in the synthesis process of (extended) burst-mode and SI controllers is the logic minimization. In this case, it must be considered that:

a) Hazard-free logic minimization, targeting (extended) burst-mode controllers operating in the GFM, has been pursued by different authors [9]-[14]. Initially, Nowick et al. [9] proposed an algorithm based on Quine-McCluskey. Posteriorly, the HFMIN minimization tool [10], although being literal-exact, showed to require thousands of seconds when dealing with large controllers. Furthermore it is not able to complete the largest benchmarks [11]. The IMPYMIN minimization tool [12] is only capable of producing product-exact solutions. The ESPRESSO-HF minimization tool [12] uses heuristic algorithms and thus cannot guarantee either literal or products exact solutions. These logic minimization tools are used in two burst-mode synthesis tools: 3D (HFMIN) [4,5], that synthesize extended burst-mode controllers (target architecture – Extended Huffman Machine - M_HE) from an XBM specification; and MINIMALIST (HFMIN, IMPYMIN and ESPRESSO-HF) [2],[10],[12] that synthesize burst-mode controllers (target architecture – Huffman Machine - M_H) from a burst-mode specification (BM). Jacobson et al [11] proposed a new minimization tool that performs literal exact hazard-free solutions for extended burst-mode controllers (ATACS). This approach can handle large circuits in a reasonable time (less than thousands of seconds). An extension of the ATACS algorithm (inserted in the Miriã synthesis tool) was proposed for multi-burst-mode asynchronous controllers (target architecture – Generalized SR latch - gRS) that also operate in the GFM [13,14].

b) Hazard-free logic minimization, targeting SI controllers, has been proposed by Beerel [8,15] and Kondratyev [16]. These logic minimization tools are used in two SI synthesis tools: **SYN** (use an initial version of ATACS) [15], that synthesizes in an architecture denominated standard C, that uses the C latch and basic gates; and PETRIFY (ESPRESSO-MC) [16,17], that synthesize either non-standard C or standard RS architectures. These methods start from the state graph specification, introduced by [7], and are based on similar theories for the extraction of F_{set} and F_{reset} functions (hazard-free logic minimization step). There is one important drawback in the logic minimization: SG specification grows exponentially with the number of signals, what limits its applicability to small or medium size controllers. To avoid the explosion of states of the SG specification, the PETRIFY tool presents some alternatives, for example, the method of structural logic minimization [18], that extracts the Boolean functions directly from the Petri-net specification, so called signal transition graph (STG), also introduced by [7].

Other hazard-free logic minimization techniques were proposed, focusing on other goals. For example, Nowick et al. [19] proposed a multi-level minimization algorithm for burst-mode controllers. In [20], it is used a synchronous technology mapping technique for the design of generalized fundamental mode asynchronous circuit in the multi-level style. In [21]-[23] some techniques are proposed for logic minimization in the dual-rail style.

In this paper, the hazard-free logic minimization theory previously proposed by Nowick and Yun [2]-[5] is extended, and applied in a new tool of hazard-free logic minimization, suitable for speed-independent extended-burst-mode asynchronous (SI_XBM) controllers. These controllers are implemented in the Feedback RS Standard (FRS) architecture (see Fig. 1). This architecture (in the case, the feedbacks) allows relaxing the SI logic minimization conditions. This tool uses an intermediate data structure based on cubes, therefore allowing extracting Booleans functions of the SI_XBM controllers with a large number of signals. The proposed logic minimizer is based on the ATACS algorithm [11]. The resulting circuits, once conjugating and taking advantages of both approaches ("speed-independent" and "extended-burst-mode asynchronous" controllers) shows to be highly suitable for implementation on high accuracy, robusteness and complex systems, like the ones actually embedded in military systems and platforms.

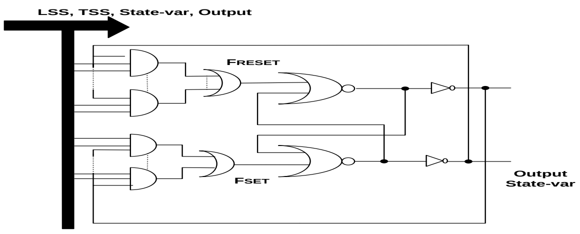

Fig. 1. Feedback RS standard architecture.

II. Extended Burst-Mode Specification

The Burst-Mode specification was proposed by a group from HP, formalized by Nowick [2] and extended by Yun [3]-[5]. Figure 2 shows an extended burst mode specification with 4 inputs (*a*,*b*,*c*,*d*), 3 outputs (*x*,*y*,*z*) and a initial state *0*. The signals b, c and d are signals that are sensitive to transition (TSS). The description $b- d- / y-$ in state transition 8 means that the output (y: $1 \rightarrow 0$) will follow the input burst (*b*: $1 \rightarrow 0$ *AND* *d*: $1 \rightarrow 0$). The level sensitive signal (LSS) *a* is used to describe the mutual exclusion between state transitions 2 and 3. The directed don't care signal b^*, in state transitions 6 and 7, means that *b* may either change its value or remain in its old value. In all state transition, it should have, at least, a signal denominated "compulsory". A compulsory signal is an input signal that, in the previous state transition, is not directed don't-care.

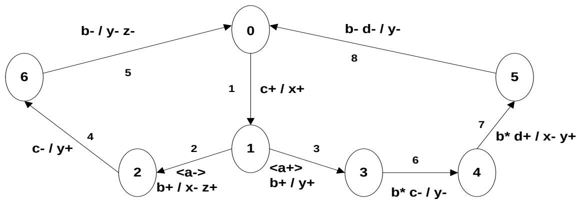

Fig. 2. Extended Burst-Mode (XBM) Specification.

The SI_XBM controllers interact with the external environment through a handshaking protocol. This protocol allows translating the XBM specification to operations of input burst/output burst, I_b/O_b mode, in which a new input burst is accepted as soon as an output burst transition has finished (any timing supposition of the fundamental mode). This mode is a simplification of the I/O mode [1],[13],[14]. For the I_b/O_b operation mode, two more restrictions are required:

- The output burst must never be empty. If this happens in the original behavior, an acknowledgement output signal must be added.
- LSS signals must become stable during the previous state. The value of a LSS must be constant in the state transitions, where it is mentioned. This condition eliminates the existing restrictions of the setup time and hold time.

III. Speed-Independent Conditions

In this section, we formally present the conditions when a non-input signal of the XBM specification can be implemented in the target architecture. For the synthesis of SI controllers, using only basic gates with arbitrary fan-in and not needing complex gates, the Lemma 1, presented below, should be satisfied.

Lemma 1: (without proof). Two sufficient conditions for the implementation of the SI circuits with basic gates and Hazard-free logic are:

1. The circuit does not have an achieved state, which is covered for more than one cube.
2. The achieved states of the circuit that form the

sequence of events (0→1...→0) or (1→0...→1) of a non-input signal should enable only a cube.

Lemma 1 explicits the conditions for any architecture related to SI circuits, where the cubes are implemented by basic gates. The Huffman machines and their extensions (in a general application) do not satisfy any of the two conditions of the lemma 1. The theory of hazard-free logic minimization for XBM functions ($f_{GFM\text{-}XBM}$), proposed by Nowick and Yun, is extended to satisfy the lemma 1, where the function $f_{SI\text{-}XBM}$ is implemented in the feedback RS architecture ($F_{SET\text{-}SI\text{-}XBM}$ and $F_{RESET\text{-}SI\text{-}XBM}$). The logic set ($F_{SET\text{-}SI\text{-}XBM}$) and logic reset ($F_{RESET\text{-}SI\text{-}XBM}$) are both implemented as sum-of-products circuits. The feedbacks of this architecture relax the condition 2 of the lemma 1 for the transitions 1→1 and 1→0, when the function is $F_{SET\text{-}XBM\text{-}SI}$; and for the transitions 0→0 and 0→1, when the function is $F_{RESET\text{-}XBM\text{-}SI}$.

A. Required and privileged cubes

Nowick [2,9] created the concept of transition cube, which was generalized by Yun [3]-[5]. This cube ($C_T[A,B]$) contains all the possible minterms that can be reached during the activation of the input burst/ output burst of a state transition, where A is an initial cube and B is a final cube. Figure 3b shows the generalized transition cube of the state transition 7 (see Fig. 3a) of the Fig. 1. In Fig. 3b, the cube $C_T[A,B]$ is formed by the initial cube A that is *bcdaxyz=2022100* and for the final cube B that is *bcdaxyz=2022220*, where 2 means don't-care. The end-subcube B' (cover the final state) is *bcdaxyz=2012010*.

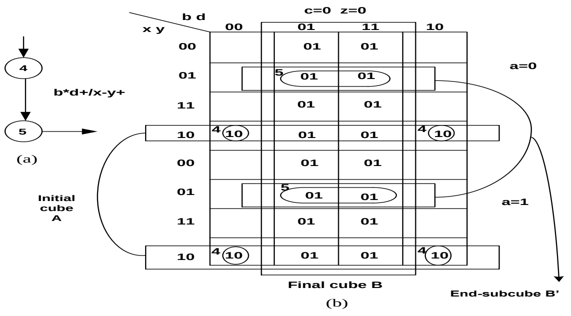

Fig. 3. Generalized transition cube

Nowick [2,9] created the concepts of “required cube” and “privileged cube”, necessary for the development of an algorithm of hazard-free logic minimization.

Definition 1: Required Cube. Be a function $f_{SI\text{-}XBM}$ and a set T of functional hazard-free transitions that are specified in $f_{SI\text{-}XBM}$. The cube $C_{Tj}[A,B] \in T$ that corresponds to the transitions 0→1 or 1→0, the final cube B is called a “required cube” of the type $C_{R\text{-}SET}$ and $C_{R\text{-}RESET}$, respectively.

Definition 2: Privileged Cube. Be a function $f_{SI\text{-}XBM}$ and a set T of functional hazard-free transitions that are specified in $f_{SI\text{-}XBM}$. The cube $C_{Tj}[A,B] \in T$ that corresponds to the transitions 0→1 or 1→0, it is called of privileged cube of the type $C_{P\text{-}SET}$ *and* $C_{P\text{-}RESET}$, respectively.

B. Requirements for logic hazard-free XBM_SI transition

If a Boolean function $f_{SI\text{-}XBM}$ contains a functional hazard for some transition of the inputs, then no implementation of the function guarantees to avoid spurious signals (glitches) during this transition. Therefore, to minimize the function $f_{SI\text{-}XBM}$, a set of transitions T should be functional hazard-free. The requirements (lemmas) to assure that the function $f_{SI\text{-}XBM}$, implemented in the feedback RS standard latch architecture (functions $F_{SET\text{-}SI\text{-}XBM\text{-}}$ and $F_{RESET\text{-}SI\text{-}XBM}$), is logic hazard-free is presented here.

Lemma 2: (without proof) If $f_{SI\text{-}XBM}$ presents a transition T_i *0→0* in the cube $C_{Ti}[A,B]$, then $F_{SET\text{-}SI\text{-}XBM}$ is logic hazard-free because there is no covering.

Lemma 3: (without proof) If $f_{SI\text{-}XBM}$ presents a transition T_i *0→1* in the cube $C_{Ti}[A,B]$, then $F_{SET\text{-}SI\text{-}XBM}$ is logic hazard-free if and only if: a) there is a unique product $p_i \in F_{SET\text{-}SI\text{-}XBM}$ that covers completely the final cube B; b) it does not exist a product $p_j \in F_{SET\text{-}SI\text{-}XBM}$, where i≠j such that $p_i \cap p_j \neq \emptyset$.

C. Logic hazard-free SI_XBM covering

In this section it is presented the sufficient conditions for a hazard-free covering of a function $f_{SI\text{-}XBM}$ with a set of specified transitions T. The function $f_{SI\text{-}XBM}$ is indefinite for all the other state inputs that are not specified. For the function $F_{SET\text{-}SI\text{-}XBM}$, the transitions 1→1 and 1→0 of the function $f_{SI\text{-}XBM}$, the states with value 1 have a don't-care behavior.

Definition 3: Illegal Intersection. Be the function $F_{SET\text{-}SI\text{-}XBM}$ implemented in the feedback Set-dominant latch architecture; a set T of functional hazard-free transitions specified in $f_{SI\text{-}XBM}$; a privileged cube $C_{P\text{-}SET}$ of $f_{SI\text{-}XBM}$ of transitions $T_j \in T$; and a product p of $F_{SET\text{-}SI\text{-}XBM}$. An illegal intersection in $f_{SI\text{-}XBM}$ ($p \cap C_{P\text{-}SET} \neq \emptyset$) happens if, and only if, the transition T_j is 0→1 and p doesn't cover completely the required cube of T_j.

Theorem 1: (without proof) SI_XBM hazard-free covering. The function $F_{SET\text{-}SI\text{-}XBM}$ is a hazard-free covering for the function $f_{SI\text{-}XBM}$ with a set of transitions T in the Feedback RS Standard architecture if, and only if:

1. No product of $F_{SET\text{-}SI\text{-}XBM}$ intersects the set OFF of $f_{SI\text{-}XBM}$.
2. Each required cube of $f_{SI\text{-}XBM}$ should be contained in a unique product of $F_{SET\text{-}SI\text{-}XBM}$.
3. No product of $F_{SET\text{-}SI\text{-}XBM}$ intersects any privileged cube illegally.

IV. SI_XBM LOGIC MINIMIZATION ALGORITHM

Logic minimization follows the state minimization and the state assignment steps [1],[2]-[5]. The algorithm proposed for SI_XBM logic minimization is based on the ATACS algorithm [11] (exact two-level - state of the art) and denominated in this article as

ATACS_FSD-SI. Firstly it transforms the XBM specification into a cubes table [11]. The ATACS_FSD-SI algorithm that follows performs mainly six steps:

1. Trigger cubes extraction.
2. Context signals extraction.
3. Intersection cubes extraction.
4. Violation cubes extraction.
5. Minimum cover: binate table.
6. Minimum cover: unate table.

These steps are performed twice for each non-input signal (output signal and state signal): one for its $F_{SET\text{-}SI\text{-}XBM}$ (F_S) and one for the $F_{RESET\text{-}SI\text{-}XBM}$ (F_R) (the target architecture adopts feedback RS standard cell for each non-input signal – see Fig. 1).

A. Extraction of the trigger cubes and context signals

Definition 4: Trigger signal. Be a state transition T_J and a input signal X that belong to XBM specification. The signal X is denominated a trigger signal in the transition T_J if, and only if, it is active in T_J and his value is non-directed don't-care.

Definition 5: Trigger cube. Be a state transition $T_J \in$ XBM specification and a non-input signal Y $\in$ XBM, where Y is active in T_J. It is called trigger cube [$Cg_{FSET(Y)J}$ or $Cg_{FRESET(Y)J}$] in T_J, the cube that is composed by the values of all of the trigger signals in T_J, adding *don't-care* values for the other signals.

Definition 6: Context signal. Be a state transition $T_J \in$XBM specification, the cube C_{TJ}[A,B] and a signal X$\in$XBM. The signal X is denominated a context signal in the transition T_J if, and only if, his value stays constant in C_{TJ} [A,B].

The algorithm firstly finds the **required cube** that covers the F_S (*For the sake of reading simplicity we will always refer to the F_S function. However the reader should keep in mind that the same procedure also applies for the F_R*) of each non-input signal for each state transitions of the XBM, where it is active (*F_S is active whenever there exists a 0→1 transition. F_R is active whenever there exists a 1→0 transition*). In order to obtain a minimum cover for F_S, the algorithm proceeds the determination of the **trigger cubes** and **context signals** (steps 1 and 2). This cube constitutes an initial (non-minimized) cover for F_S. The context signals are later used to remove possible violations of this initial cover. By definition, the trigger cube covers completely the required cube of the respective state transition and therefore satisfies lemma 3.

As illustration, in state transition 3 of Fig. 2, there is a trigger cube: $Cg_{FSET\text{-}T3(Y)} \rightarrow$ *bcdaxyz=1222222,* where b=1 is a trigger signal. The context signals in this state transition are c=1, d=0, a=1, x=1 and z=0 (this may be seen observing the XBM description in Fig. 2)

B. Extraction of the violation and intersection cubes

Each trigger cube can have two types of violations: covering and intersection violations. The steps 3 and 4 of our algorithm are executed if there are one or both of these violations.

Definition 7: Covering Violation. Be a state transition $T_J \in$XBM specification; and $Cg_{TJ\text{-}X}$ a trigger cube of T_J of the non-input signal X of the function $f_{SI\text{-}XBM}$ (*F_S* or *F_R*). A "covering violation" of the $Cg_{TJ\text{-}X}$ happens if, and only if, there is, at least, a minterm of $Cg_{TJ\text{-}X}$ where the signal X has an opposite value (0 or 1).

Definition 8: Intersection Violation. Be a state transition $T_J \in$XBM specification; and $Cg_{TJ\text{-}X}$ the trigger cube of T_J of the non-input signal X of the function $f_{SI\text{-}XBM}$ (*F_S* or *F_R*). An "intersection violation" of the $Cg_{TJ\text{-}X}$ happens if, and only if, there is an illegal intersection among a privileged cube Cp and $Cg_{TJ\text{-}X}$ (see definition 3.3).

Definition 9: Violation cube is a cube formed by the group of adjacent minterms of the "covering violation" of a trigger cube C_g.

Definition 10: Intersection cube is a cube formed by the group of adjacent minterms obtained by the "intersection violation" among a privileged cube C_p and a trigger cube C_g.

There are two types of violations. Cover violations occur when the initial cover (trigger cube) covers states where the F_S is defined to have the opposite value (violation cubes). Intersection violations occur when the initial cover (trigger cube) violates the lemma 3. The intersections between a required cube and a trigger cube are called intersection cubes. Both, violation and intersection cubes, must be removed from the initial cover

C. Covering tables

In this section, it is presented the solution of the binate and unate tables. Step 5 consists of extracting the cubes denominated trigger-FV cubes (free from violations), which generates a problem of binate covering [24]. These cubes should satisfy lemma 3.

Definition 11: Trigger-FV cube. Be a state transition $T_J \in$ XBM; and $Cg_{TJ\text{-}X}$ a trigger cube of T_J of the non-input signal X of the function $f_{SI\text{-}XBM}$ (*F_S* or *F_R*). It is called a $Cg_{TJ\text{-}X}$ of trigger-FV cube ($Cg_{TJ\text{-}X\text{-}FV}$ – free-violations) if, and only if, there are not covering and intersection violations.

If the trigger cube of a transition T_J is not FV, then there is some type of violation, being necessary the insertion of appropriate context signals of T_J in order to have this violation eliminated. The lines of the binate covering table for $Cg_{TJ\text{-}X}$ are the context signals of T_J, *while* the columns are the violation and intersection cubes that are contained in the $Cg_{TJ\text{-}X}$. The solution of the minimum coverage of the binate table leads to finding one or more minimum trigger-FV cubes (smaller number of context signals - $Cg_{TJ\text{-}X\text{-}MIN}$) that cover (required cubes) the largest number of state transitions and satisfies lemma 3. To obtain the minimum coverage of the binate table, ATACS_FSD-SI uses classic reduction techniques [24], the branch and bound algorithm for cyclical binate tables [24].

After extracting all the minimum trigger-FV cubes for all transition $T_J \in$XBM specification, where the non-input signal X is activated for the function $f_{SI\text{-}XBM}$ (F_S or F_R), a unate covering table is built, where the lines are the minimum trigger-FV cubes and the columns are the state transitions (required cubes) of the non-input signal X of the function $f_{SI\text{-}XBM}$ (F_S or F_R). The solution for minimum covering of the unate table (step 6) leads to finding the smallest number of minimum trigger-FV cubes that cover all state transitions (required cubes), where X is $0 \rightarrow 1$ (F_S) or $1 \rightarrow 0$ (F_R), and that satisfy the theorem 1 To obtain the minimum coverage of the unate table, ATACS_FSD-SI uses classic reduction techniques, the branch and bound algorithm.

V. Example

We will illustrate the ATACS_FSD-SI algorithm through the example shown in Fig. 2. Figure 4 shows the context signals, trigger cubes, and required cubes for Y_SET (Y+). Figure 5 shows the binate tables of the trigger cubes (state transitions 3 and 4, in the Fig. 2). The trigger cube of the state transition 7 is FV, because there were not violations or intersections cubes. The exact minimum solutions for the binate tables are respectively the cubes 1121122 and 1022221. For example, the trigger_FV cube 1121122 was generated inserting the values of the context signals (*c*, *a* and *x* in the binate table) into the trigger cube. Figure 6 shows the corresponding unate table. Observe that it only contains state transitions 3, 4 and 7 (transitions where the output signal transition is Y $0 \rightarrow 1$). The exact minimum solutions are the three cubes in the unate table. The cube bcdaxyz=1121122 describe the term product *bcax*.

Figures 7, 8 and 9 show the same steps, but now for the function Y_RESET (Y−). This example needs a feedback in the product terms that contain the signal *a* (LSS), as shown in Fig. 10.

	Context	Trigger	Required
Y+ /3	2101120	1222222	1101120
Y+ /4	1202021	2022222	1002021
Y+ /7	2022220	2212222	2012220

Fig. 4. Context, trigger and required cubes for Y_SET.

Y_SET/3	1100202	1102001	1002120	1012220
c	0	0	1	1
$\overline{d}$	0	0	0	1
a	1	0	0	0
x	0	1	0	0
$\overline{z}$	0	1	0	0

Y_SET/4	0002022	2002120
b	1	0
x	1	0
z	0	1

Fig. 5 Binate cover tables from Y_SET.

Y_SET	1101120	1002021	2012220
1121122	1	0	0
1022221	0	1	0
2212222	0	0	1

Fig. 6. Unate cover table from Y_SET.

Out/Tran	Context	Trigger	Required
Y- / 5	2002022	0222222	0002022
Y- / 6	2202120	2022222	2002120
Y- / 8	2022020	0202222	0002020

Fig. 7. Context, trigger and required cubes for Y_RESET.

Y_RESET/5	0102110	0012220	0002120
$\overline{c}$	1	0	0
$\overline{d}$	0	1	0
$\overline{x}$	1	0	1

(a)

Y_RESET/8	0102110	0002120
$\overline{c}$	1	0
$\overline{x}$	1	1

(b)

Y_RESET/6	1002021	2012220	1022010
$\overline{d}$	0	1	0
$\overline{x}$	1	0	1
z	1	0	0

(c)

Fig. 8. Binate cover tables from Y_RESET.

Y_RESET	0002022	2002120	0002020
0202022	1	0	1
2002122	0	1	0

Fig. 9. Unate cover table from Y_RESET.

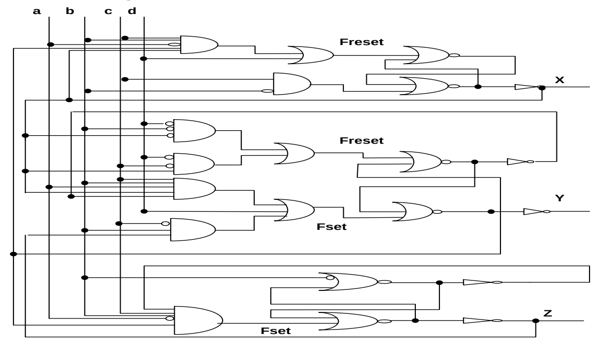

Fig. 10. Logic circuits.

VI. Results

We applied our logic minimization tool in 14 different benchmarks, specified in XBM specification. The proposed logic minimization tool is named Miriã-SI. Table I shows the XBM specification of the 14 benchmarks. The specification is given in terms of the number of states, number of state transitions (St/Tran), number of inputs and number of outputs (I/O). Table I also shows the area (number of literals – N-Lit and number of gates – N-gate) obtained by the tools: Miriã-SI, Miriã-GFM-gRS [13,14] and 3D-HMOF (Huffman Machine with output feedback) [4,5]. The Miriã-SI results were 20% smaller in literals and 20% smaller in gates (*The latch (complex gate – to see Fig. 1) was esteemed as a simple gate*) when compared with 3D. There was an increase of literals (20%) and gates (10%), compared to Miriã-GFM.

Table II shows, for the same set of 14 benchmarks, the cycle time and latency time of the circuits obtained by the three compared tools. The latency and cycle times were obtained assuming a gate delay given by the equations (assuming fan-out=1): $t_{min\text{-}gate}$=(0.1*fan-in+0.9) time units and $t_{max\text{-}gate}$=(0.25*fan-in+2.1) time units. The average latch delay was "2 time units". For Miriã-SI the times of cyle and latency are equals.

Table II shows an average cycle time reduction of 35% when compared to 3D and 7% when compared to Miriã_GFM. The average latency time penalty was 36% when compared to both 3D and Miriã_GFM.

These results were obtained in less than 20 seconds (each) when running ATACS_FSD-SI on a PC (Windows XP, Pentium 4, 480MB RAM and 2.8 GHz). In spite of the penalty in area and latency time, these circuits present the advantages of being SI, as: high modularity, higher robustness, easiness in formal verification and testability (covering without redundancy and self-checking property).

Table I Results: Literals and Gates

	SPEC		MIRIÃ-SI	MIRIÃ-GFM-gRS	3D-HMOF
	I / O	St /Tran	N-lit /N-gate	N-lit /N-gate	N-lit /N-gate
Biu-fifo	4/2	6/7	25/15	23/19	27/20
Biu-fdma	4/2	7/9	50/30	41/23	51/31
Diffeq-alu2	5/7	14/16	142/56	95/47	139/68
D-SET-flip-flop	2/1	4/6	19/13	16/11	23/28
D-DET-flip-flop	2/1	4/8	28/17	24/13	33/18
JK-SET-flip-flop	3/1	6/12	19/14	45/21	107/47
JK-DET-flip-flop	3/1	8/32	28/18	24/14	469/155 **
Sbuf-send-pkt2	3/2	4/5	18/12	15/14	22/16
Scsi-inic-send	4/2	6/8	33/18	23/18	23/17
Scsi-targ-send	6/4	11/14	81/36	70/30	72/35
Select2ph	2/2	4/8	28/19	24/14	30/17
Selmerge-2ph	3/2	8/12	60/29	48/25	68/31
T-SET-flip-flop	2/1	4/6	19/17	16/12	****
T-DET-flip-flop	2/1	4/8	28/13	24/14	29/15

Table II Results: Latency and Cycle times

	MIRIÃ-SI	MIRIÃ-CFM-gRS		3D-HMOF	
	Cycle Time	Latency Time	Cycle Time	Latency Time	Cycle Time
Biu-fifo	12.3	8.5	12.9	8.5	16.6
Biu-fdma	13.4	8.0	13.8	9.9	19.0
Diffeq-alu2	11.6	8.7	14.3	9.1	19.5
D-SET-flip-flop	13.7	7.4	10.8	6.4	15.2
D-DET-flip-flop	13.7	7.8	13.2	8.0	15.4
JK-SET-flip-flop	14.4	7.6	11.3	7.6	17.5
JK-DET-flip-flop	10.7	8.1	13.2	9.0	27.8
Sbuf-send-pkt2	6.0	7.4	10.4	6.8	13.3
Scsi-init-send	14.0	8.0	12.9	7.9	19.8
Scsi-targ-send	12.3	8.9	13.8	8.1	23.2
Select2ph	8.2	8.1	13.2	8.9	15.4
Selmerge-2ph	9.4	8.1	14.0	8.1	19.8
T-SET-flip-flop	13.6	7.5	11.1	****	****
T-DSET-flip-flop	11.9	7.7	13.2	8.0	13.3

VII. CONCLUSION

In this paper the theory from Nowick of hazard-free logic minimization of an XBM function was extended to accomplish the logic minimization of a SI_XBM function that interacts with the external environment in I_b/O_b mode (SI_XBM controllers). A variant of the architecture from [20] was proposed, called feedback RS standard that relaxes the covering conditions of an SI_XBM function. The ATACS Algorithm, that is exact in number of literals (state-of-the-art), was modified to satisfy the theory of the logic minimization of the SI_XBM function and to be suitable to the proposed architecture. In order to validate the algorithm, it was applied to a set of experiments/benchmarks, showing relevant results, which can be easily extended to high complex military applications, as the embedded systems actually existing in aircrafts, battle ships and Army's vehicles.

REFERENCES

[1] C. J. Myers, "*Asynchronous Circuit Design*," Wiley & Sons, Inc., 2004, 2^{a} edition

[2] S. M. Nowick, "*Automatic Synthesis of Burst-Mode Asynchronous Controllers*," Ph.D. thesis, Stanford University, 1993.

[3] K. Y. Yun, "*Synthesis of Asynchronous Controllers for Heterogeneous Systems*", Ph.D. thesis, Stanford University, 1994.

[4] K. Y. Yun and D. L. Dill, "Automatic Synthesis of Extended Burst-Mode Circuits: Part I (Specification and Hazard-.Free Implementation) and Part II (Automatic Synthesis)," IEEE Trans. on CAD of Integrated Circuit and Systems, Vol. 18:2, pp. 101-132, Feb. 1999.

[5] K. Y. Yun, et al., "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver," I*EEE Transactions on VLSI Systems*, vol. 6, no 4, pp.643-655, Dec.1998.

[6] S. Rotem, et al., "RAPPID: An asynchronous instruction length decoder,"in Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, April, 1999, pp.60-70.

[7] Tam-Anh Chu, "*Synthesis of Self-Timed VLSI Circuits from Graph-Theory Specifications*," Ph.D. thesis, June, 1987, Dept. of EECS, MIT.

[8] P. A. Beerel, "*CAD tools for the synthesis, verification, and testability of robust asynchronous circuits*," Ph.D. dissertation, Stanford University, August, 1994.

[9] S. M. Nowick e D. L. Dill, "Exact Two-Level Minimization of Hazard-Free Logic with Multiple-Input Changes," *IEEE Trans. on CAD of Integrated Circuits and Systems,* Vol. 14, no 8, August 1995.

[10] R. M. Fuhrer, "*Sequential Optimization of Asynchronous and Synchronous Finite-State Machine*,' Ph.D. thesis, Department of Computer Science, Columbia University, 1999.

[11] H. M. Jacobson and C. J. Myer, "Efficient algorithms for exact two-level hazard-free logic minimization," *IEEE on Trans. CAD of integrated Circuits and Systems*, vol.21, no.11, pp 1269-1283, November, 2002.

[12] M. Theobald and S. M. Nowick, "Fast heuristic and exact algorithms for two-level hazard-free logic minimization", *IEEE Trans. on Computer-Aided Design*, vol. 17, no 11, pp. 1130-1147, Nov. 1998.

[13] D. L. Oliveira, M. Strum et al., "Modified ATACS algorithm for the logic minimization of multi-burst-mode asynchronous controllers," Proc. VIII Workshop Iberchip, Cartagena, Colômbia, (mídia eletrônica), March, 2004.

[14] D. L. Oliveira, et al., 'Miriã: a CAD tool synthesize multi-burst controllers for heterogeneous systems," *Microelectronics Reliability*, 43 (2003) 209-213.

[15] P. A. Beerel, C. J. Myers and T. H. Meng, "Covering Conditions and Algorithms for the Synthesis of Speed-Independent Circuits," *IEEE Trans on CAD of Int. Circuits and Systems,* vol.17, no.3, March, 1998.

[16] A. Kondratyev, M. Kishinevsky and A. Yakovlev, "Hazard-Free Implementation of Speed-Independent Circuits," *IEEE Trans. CAD of Int. Circuits and Systems,* vol.17, no. 9, September, pp. 749-771, 1998.

[17] J. Cortadella, et al., 'Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Trans. on Information and Systems*, E80-D(3), pp.315-325.

[18] E. Pastor, J. Cortadella, A. Kondratyev and O. Roig, "Structural methods for the synthesis of speed-independent circuits," *IEEE Trans. CAD*, vol 17, pp. 1108-1129, November, 1998.

[19] S. M. Nowick, C. W. O'Donnell, "On the Existence of Hazard-free Multi-Level Logic," Proc. IEEE Ninth Int. Symposium on Asynchronous Circuits and Systems, 2003.

[20] F. Shi, Removing hazards in multi-level logic optimization for generalized fundamental-mode asynchronous circuits", IEEE International Conference on Computer Design, pp.640-645, 2008.

[21] J.Cortadella, A. Kondratyev, L. Lavagno, and C. Sotiriou, "Coping with the Variability of Combinational Logic Delays," IEEE Int. Conf. On Computer Design, pp.505-508, 2004.

[22] I.Lemberski, "Method of Asynchronous Two-Level Logic Implementation," IAENG International Journal of Computer Science, 35:1, IJCS_35_1_09 19, February 2008.

[23] I.Lemberski, and P. Fišer, "Asynchronous Two-Level Logic of Reduced Cost," IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 68-73. April 15-17, Liberec, Czech Republic, 2009.

[24] O. Coubert and J. C. Madre, "New ideas for solving covering problems," Proc. 32th ACM/IEEE DAC, pp.641-646, 1995.