ISSN:1983 7402 ITA, 23 a 25 de setembro de 2014

A Novel State Minimization Method for Extended Burst-Mode Machines Based on Genetic Algorithm

Tiago Curtinhas, Duarte L. Oliveira, Lester A. Faria, Osamu Saotome
Divisão de Engenharia Eletrônica do Instituto Tecnológico de Aeronáutica – IEEA – ITA
Marechal Eduardo Gomes, 50 – CEP 12.228-900 – SJC – São Paulo – Brazil

Abstract **— Nowadays, the design of complex synchronous digital systems shows serious difficulties relating to the global clock signal and to Deep-Sub-Micron MOS technology. Asynchronous design shows to be an interesting alternative to solve these difficulties. Asynchronous Finite State Machines (AFSM) are widely used for the design of the control unit of asynchronous digital systems. A very popular machine is the extended burst-mode AFSM (XBM_AFSM), and one of the most important steps in the synthesis of these machines is the state minimization. This paper proposes a novel method for state minimization of XBM_AFSMs. Such machines are implemented in Huffman architecture (XBM_HMs). The proposed method performs a minimum coverage using genetic algorithm. The XBM_HMs present a better interaction with fast environments, reduce the cost of timing analysis and show a high potential for a better cycle time when compared to HM architecture with output feed-back, which is the target architecture of the 3D tool, the state of the art in synthesizing XBM_AFSMs. The proposed tool is tested for an extensive set of benchmarks showing high efficiency, obtaining an average reduction of 25% in the number of states.**

Index Terms**— Asynchronous logic, genetic algorithm, AFSM.**

I. INTRODUCTION

The difficulties in designing complex synchronous digital systems, due to the global clock signal [1,2] and Deep-Sub-Micron MOS technology [3], can be overcome with asynchronous circuits. Asynchronous designs present potential advantages when compared to their synchronous counterparts, such as: no clock skew, no distribution of the clock, lower power consumption, larger modularization and more robustness against temperature variations and electromagnetic interactions [4]. On the other hand, its main disadvantage is the lack of tools for automatic synthesis [5,6].

Asynchronous FSMs are important components of an asynchronous digital system, which may be composed by a network of FSMs + data-paths [6]. Many digital systems are composed by components like data-paths and controllers [5-8]. Two traditional styles of specification have been proposed to describe asynchronous controllers for an **optimized synthesis**: Signal Transition Graph (STG) and Burst-Mode (BM).

Tiago Curtinhas, thiagohd@ita.br; Duarte L. Oliveira, duarte@ita.br, Tel. 55+ (12) 3947-6813; Lester A. Faria, lester@ita.br; Osamu Saotome, osaotome@ita.br.

STG was proposed by Chu [9] and is a Petri-net description. The main strength of STG is the ability to describe concurrence between inputs and outputs (I/O concurrence) that occur in asynchronous systems. It naturally describes timing diagrams that are quite used on the interfaces design. However, there are several descriptions in STG that cannot be implemented, because the description becomes very confusing when manipulating a large number of signals. Furthermore, this type of description may over-grow in size, besides not being familiar to designers of the synchronous world.

BM was proposed by Coates et al. [10], formalized by Nowick [11] and extended by Yun and Dill [12], as extended burst-mode (XBM). It is used to describe asynchronous finite state machines (AFSM) of Mealy type. The XBM specification is based on state diagram, being familiar to designers. She describes from synchronous FSMs of Moore type up to AFSMs of heterogeneous systems. XBM_AFSMs operates as a class of MIC (multiple input change), called burst mode. These machines interact with the environment in generalized fundamental mode (GFM). In this mode, a new set of input signals (input burst) will be activated only if the machine is in a stable state, i.e. there is no activity on the gates and lines. The XBM_AFSMs obey to the bounded gate and wire delay model, and have been applied on important academics and industrials designs [13-17].

For the automatic synthesis of XBM AFSMs, the 3D method is well-known, as proposed by Yun and Dill [12]. This method implements the circuits as Huffman machines with output feed-back (HMFO – see Fig. 1). The HMFO architecture leads to smaller areas when compared to traditional Huffman machines (MH – see Fig. 2) because it uses the output signals as states signals, thus reducing the need for states signals. As the output signals are feed-back in HMFO architecture, there is a high potential to the output equations have a higher latency time when compared to traditional HMs, once these machines states are represented only by states signals [18].

The BM_HMs allows processing state transitions in one of the two machine cycles, which are: input burst→output burst concurrently with state signals; and input burst→state signals→output burst. Consequently, the HM architecture is the most suitable for interacting with fast environments, because HMFO architecture uses machine cycles that may violate the GFM. By violating the GFM, delay elements should be inserted, degrading the performance and reliability of AFSMs.

As the output signals tend to be fed back in the HMFO architecture, the 3D method uses the machine cycle, input burst→state variable→output burst, when interacting with fast environments. This machine cycle, besides causing a penalty in cycle time [18], does not guarantee a definitive solution for the violation of the GFM, because the output signals are fed back.

Despite showing a potential for larger area, the HM architecture presents some interesting features, especially when interacting with fast environments. The XBM specification supports level sensitive signals (LSS) with non-monotonic behavior, which lets you to interact with conventional functional units, synchronous processors and even heterogeneous systems. These different environments may have a common characteristic: they have a faster response time.

This paper proposes a novel method for state minimization of Huffman machines that support the XBM specification (XBM_HMs). The proposed method uses modified classical algorithms to obtain the maximum compatibility classes and minimum coverage, which is performed by using genetic algorithm. Authors are unaware of the existence of any algorithm for minimizing states for XBM_HMs machines such this one. As the XBM specification incorporates the BM specification and Minimalist tool [19] synthesizes BM_HMs machines, it was possible to compare results of an extensive set of BM benchmarks, considering our method of states minimization and the method of states minimization proposed by Führer et al. [20], which is incorporated in the Minimalist tool.

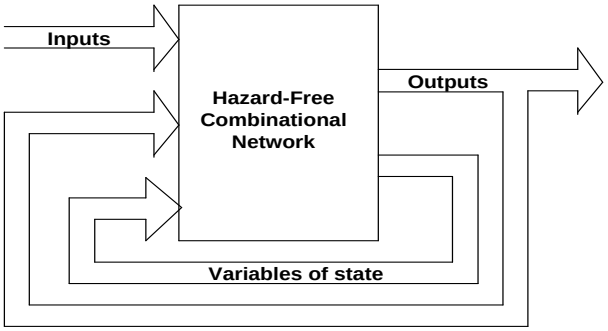

Fig. 1. Huffman machine with feed-back output – HMFO.

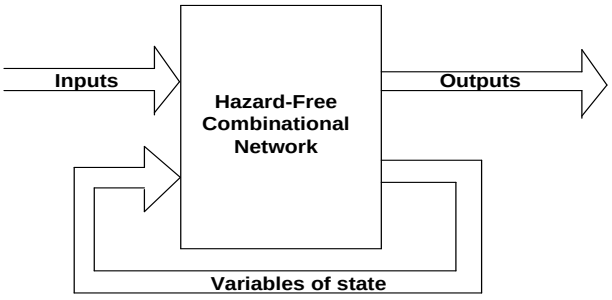

Fig. 2. Huffman machine – HM.

II. EXTENDED BURST-MODE SPECIFICATION (XBM)

The BM specification is represented by a diagram of state transitions, where transitions can occur for single, or multiple input changes. It is necessary to define the initial state. Each state transition is represented by an arc in the diagram, and is labeled by a set of input signals and/or output signals. These sets are called input (output) bursts. The signals are always transition sensitive (0→1, or 1→0) and are called transition sensitive signals (TSS). Each input burst has to be no empty. If any change occurs in the inputs, the machine stays in the same state. The input burst are monotonic, changing only once during each state transition. The BM specification has to respect the polarity property, unique entry point, and the maximal set property.

The XBM specification inherits the same characteristics of the BM specification and adds up the l**evel conditional signals** and **direct don't-care signals**. The level conditional signals allow describing in different ways two or more state transitions, depending on the level (0 or 1) of one or more input level conditional signals. The transition sensitive signals could be direct don't-care, which describes concurrency between inputs and outputs. A level conditional signal (level sensitive signal – LSS) can switch freely if it is not labeled by a state transition (non-monotonic behavior). If it is, it has to meet the setup and hold time requirements. The direct don't-care signals should have only one monotonic transition. Every state transition in XBM has a so-called compulsory signal, which in the previous transition was not a direct don't-care signal (terminating signal).

Fig. 3 illustrates a XBM specification that describes the SCSI_INIT-SEND (small computer systems interface) bus controller, as defined by the ANSI standard X3.131-1986, which is a physical and logical communication protocol between computers and peripheral devices (see Fig. 3). The SCSI benchmark contains 4 entries (Cntgt1, Fain, Ok, Rin), 2 outputs (Aout, Frout) and an initial state 0. The description *Rin+ Fain- / Aout+* in the transition *5→3* means that the output (Aout: 0 to 1) will be activated when the burst input is enabled (Rin: 0 to 1 *AND* Fain: 1 to 0). The LSS signal cntgt1 is used to describe the mutual exclusion between transitions *3→6* and *3→4*. One example is the arc labeled with [*<cntgtl-> rin- / aout-*] that means if *cntgtl*=0 and the *rin* input changes from 1 to 0, the *aout* output will change from 1 to 0. The "directed don't-care" signal Rin*, in transition 4→5, means that Rin can change its value or remain in its previous value. In the state transition 2→3, the signal *Fain* is compulsory. Yet other restrictions have to be met on the XBM specification [12].

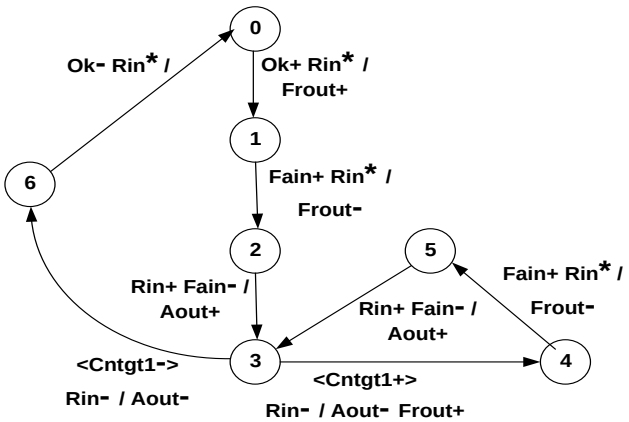

Fig. 3. XBM Specification: SCSI_INIT-SEND.

III. XBM STATE CODING PROPERTY

The XBM specification describes FSMs of Mealy type. The FSMs of Mealy type will be implemented as Huffman machines (HM), where the states are solely identified by the state variables (see Fig. 2).

Definition 1: A finite state machine (FSM) is described by a 4-tuple <S,I,O,T>, where S is a finite set of states; I is a finite set of inputs; O is a finite set of outputs; T is a finite set of state transitions. A state transition is represented by two cubes (A, B), which denotes all legal path [A, B], where A is the cube of input and B is the cube of output [11,21].

A XBM specification is well-formed if it satisfies the properties of polarity, distinguishability constraint and unique entry condition, as proposed in [12]. The XBM specification, when implemented in the HM architecture, must satisfy the condition of the bridge state.

Definition 2: *Bridge state XBM (XBM_BS)* says that a well-formed XBM specification satisfies the XBM_BS condition if in all state transition coming out from a state decision with LSS signals, there is at least one bridge state inserted in each state transition generating a multicycling.

Figure 4 shows the XBM specification of the SCSI_INIT-ISEND benchmark that presents the state decision 3 involving the LSS Cntgt1 signal, where the states bridges 6' and 4' were introduced to satisfy definition 2.

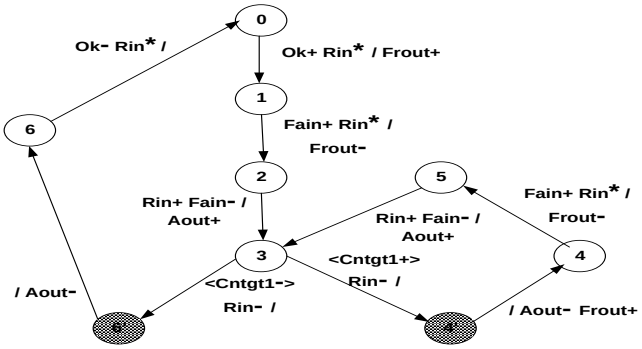

Fig. 4. XBM specification: SCSI-INIT-ISEND with bridge states.

IV. STATE MINIMIZATION: XBM_HM

To perform the merging of states in the XBM_HMs we introduce a new definition for compatible states. The method of merging of states starts from the XBM specification, which satisfies the condition of bridge state (XBM_BS).

Definition 3: *Merging of states MS_XBM_BS.* Be any two states *Si* and *Sj* of a XBM specification. It is said that *Si* and *Sj* can fuse, if Si and Sj are compatible states in XBM_BS.

Definition 4: *Compatible states XBM (CS_XBM).* Be any two states *Si* and *Sj* of a XBM_BS specification. It is said that *Si* and *Sj* are compatible states if the value of the outputs they specifies are equal, if the next states, if specified, are compatible, *Si* and *Sj* are not bridge states and do not violate the requirements of logic cover [22].

The proposed method of state minimization of XBM Huffman machines consists of four steps:

1. Insert bridge states in the XBM specification and obtain XBM_BS.
2. Generate the table of pairs of compatible states XBM (XBM_Table), using definition 4 [23].
3. From the XBM_Table, generate the classes of states of maximal compatibility (CSMC_XBM) [23].
4. From XBM_Table and CSMC_XBM perform the minimum coverage, using genetic algorithm [24]. The minimum coverage obtains the lowest number of CSMC_XBM that covers all states of XBM_BS and satisfies the property closing.

The algorithm in Fig. 5, of XBM-SPEC, creates the merge table and generates a set of maximal compatibility class. The minimum covering is performed by a genetic algorithm

Algorithm 1: Procedure state minimization for HM architecture

```
mergerGraph ← creatMergerGraph(spec);
CG ← creatCompatibleGraph(mergerGraph);
generation ← 0;
nSMC ← k; P(t) ← genChromosome(C);
while true do
    if generation = maxGeneration & covering > 0 then
        nMCS ← nMCS + 1; C ← genChromosome(C);
        generation ← 0;
    else
        classify(S); done;
    end if
    covering ← verifCover(spec,C);
    if covering = 0 then
        S ← insertSol(C);
    end if
    selection(C); crossover(C); mutation(C);
    P(t+1) ← elitism+C;
end while
```

Fig. 5. State minimization for XBM_HMs: procedure.

A. XBM_Table

Illustrating the step of states minimization, we use the XBM specification of Fig. 4 as table of flow in Fig. 6. The first step generates the table of pairs for compatible states, where "V" and "X" shows the compatible and non-compatible states respectively (see Fig. 7).

Fain Ok Rin	Cntgt1=0								Cntgt1=1							
	000	001	011	010	110	111	101	100	100	101	111	110	010	011	001	000
0	S0, 00	S0, 00	S1, 01	S1, 01									S1, 01	S1, 00	S0, 00	S0, 00
1			S1, 01	S1, 01	S2, 00	S2, 00					S2, 00	S2, 00	S1, 01	S1, 01		
2			S3, 10	S2, 00	S2, 00	S2, 00					S2, 00	S2, 00	S2, 00	S3, 10		
3			S3, 10	S6', 10									S4', 10	S3, 10		
4			S4, 01	S4, 01	S5, 00	S5, 00					S5, 00	S5, 00	S4, 01	S4, 01		
4			S3, 10	S5, 00	S5, 00	S5, 00					S5, 00	S5, 00	S5, 00	S3, 10		
6	S0, 00	S0, 00	S6, 00	S6, 00									S6, 00	S6, 00	S0, 00	S0, 00
S6'				S6, 00									S6, 00			
S4'				S4, 01									S4, 01			

Scsi-init-send-specification

Fig. 6. Table of flow of the SCSI-INIT-ISEND.

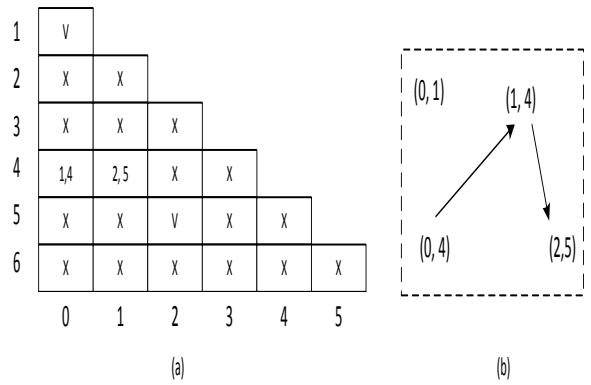

Fig. 7. SCSI-INIT-ISEND: a) XBM_Table; b) graph of dependency.

B. *CSMC_XBM*

The second step generates the classes of maximum compatibility, shown in Fig. 8 and the classes {3},{6} and the classes (bridge states) {4'} and {6'}.

State	Compatible	Máximal compatibility
2	5	{2, 5}
1	4	{2, 5}; {1, 4}
0	1,4	{2, 5}; {0, 1, 4};

set of maximal compatibles

Fig. 8. Set of classes of maximal compatibles of the SCSI-INIT-ISEND.

C. *Minimum coverage based in genetic algorithm*

The algorithm 1 (see Fig. 5) is a technique for optimization and stochastic search, called Genetic Algorithm (GA), being applied in the step of minimum coverage [24]. In the step of state minimization, the step of minimum coverage is a problem of unate coverage [20] that is, to obtain the smaller number of classes of maximal compatibility, covering all states. Due to the possibility of enumerating the classes of maximal compatibility, the encoding used on the chromosome is an entire coding, a vector of integers. From the listed classes, it is one way to represent a possible solution. The chromosome C_0, in Fig. 9 is a sub-set of the classes, using the entire coding.

C1. Evaluation

The states minimization for XBM_HMs, in case the problem of minimum coverage, the equation (1) shows the *Fitness* function, where "Sc" is the number of states covered. The *Fitness* is responsible for classifying a population of individuals aiming to find the smallest set of classes of maximal compatible *Fitness=K*, where K ≤ N, and N is the number of CSMC and K is the smallest set of CSMC that cover all states of XBM_BS.

$$Fitness = \sum Sc \qquad (1)$$

C2. Selection

The selection is performed in groups of 15 individuals where the best ones are compared "two by two" and the algorithm chooses the two "most suitable" chromosomes for the crossover. The same 15 individuals continue to be considered for the next generation processes, once it is desirable to keep the best features of these "most adapted" chromosomes improving the next generations.

C3. Crossover

This step is based on the "crossover operator", which is related to a relative occurrence rate and shows a rate above 60%. Its main goal is to provide a greater diversity of codes for dichotomy equal to zero. It is responsible for generating new offsprings from this population. Individuals are classified as they better satisfy condition 1. Fig. 9 illustrates the effects of crossover and mutation operators in a chromosome of four classes. As shown in Fig. 9, the pairs are selected among the best chromosomes: C1 and C2 generate a new individual that is C3.

C4. Mutation

"Mutation operator" is related to a relative occurrence rate. Due to the application of crossover operator, after some interactions, the population tends to become very homogeneous. To prevent that and further explore the search space, the mutation operator changes the value of some chromosomes as shown in Fig. 9, where one bit of chromosome "C_3" has its value changed from 1 to 0.

Chromossome	**Crossover**	**Mutation**
[3 2 0 1] : C_0	**[3 2 0 1] :** C_1 **[4 5 0 6] :** C_2 **[4 5 0 1] :** C_3 **[3 2 0 6] :** C_4	**[3 2 0 6] :** C_3 **[3 2 1 6] :** $C_{3'}$

Fig. 9. Representation of chromossome.

The evolutionary process, starts from a randomly generated population P_0 of chromosomes and a value of k classes equal to 1. For 500 generations, the population is subjected to the evolutionary process: objective function, selection, crossover and mutation (Fig. 4). If it finds a solution, the algorithm returns; otherwise k is incremented and the chromosome size increases in a position and the process restarts.

After the coverage is performed, the states of the specification SCSI-INIT-ISEND were reduced from 7 to 6 (see Fig. 10), two classes were used from Fig. 8, in addition to the class {S3},{S6}, {S4'} and {S6'} that not are compatible with any other class, and were inserted in the final solution.

S0' = {0, 1, 4}
S1' = {2, 5}
S2' = {3}
S3' = {6}
S4' = {S4'}
S5' = {S6'}

Fig. 10. Coverage unate: SCSI-INIT-ISEND.

V. EXPERIMENTAL RESULTS

The proposed method of states minimization was implemented in C language and incorporated in the SAGAAs (State Assignment Genetic Algorithm Asynchronous) tool [25]. It was tested in 44 benchmarks of asynchronous paradigm, where 14 are XBM and other 30 are BM. The compared tool is the Minimalist [19] which is considered the state of the art, but only accepts BM. For the XBM specification, implemented as Huffman machine (XBM_HM), does not exist in the literature a tool that synthesizes the XBM_HM machines. Also, is not possible any kind of comparisons, due to the need of inserting bridge states when there are LSS signals, we evaluated the ability of our method of minimization. All benchmarks were synthesized in HM architecture.

Table I shows XBM specifications, where *in/out/st/tran* mean respectively the numbers of input signals, output signals, states and state transitions. Table 1 also shows, the states minimization of the XBM specification objectifying to HM architecture. For SAGAAs tool, the results *St/T* which mean respectively: final number of states and time of processing. The SAGAAs tool obtained an average reduction of states of 25%.

TABLE I RESULTS: NUMBER OF STATES IN XBM

Benchmark	Specification In Out St Tr	#Sagaas St tempo(s)
ack-xbm1	4/5/8/10	6 1.11
ates-xbm	5/3/15/21	9 1.67
biu-dma2fifo	4/2/7/9	8 0.01
chu-133-2	3/3/4/4	3 0.01
ex02	4/2/10/15	9 1.40
ex-des	5/7/7/8	5 0.86
ex-sad	5/11/11/15	11 1.02
ex-speak	4/3/8/10	6 0.91
ex-tese_yun	3/2/4/4	2 0.23
fifo-cell-ctrl	2/2/3/3	3 0.02
hp-ex01	5/5/7/7	2 0.35
sbuf-sct	3/3/8/9	4 0.52
yun-diffeq-mul1	3/3/4/4	3 0.04
yun-diffeq-mul2	3/3/3/3	3 0.02

Table II shows BM specification where *in/ Out /St/ Tr* mean respectively the numbers of input signals, output signals, states and state transitions.

Table II also shows for Minimalist and SAGAAs tools the results *St/T*, which mean respectively: final number of states and time of processing. The SAGAAs tool compared to Minimalist obtained an average reduction of states of 1% and average reduction of time of processing of 98% The Minimalist tool failed to minimize the benchmark counter-bin.

TABLE II RESULTS: NUMBER OF STATES IN BM

Benchmark	Specification In Out St Tr	#Minimalist St T(s)	#Sagaas St T(s)
alloc-outbound	4 3 8 9	5 0.26	5 0.21
call-proc	3 3 12 16	2 0.41	2 0.31
concur-mixer	3 3 5 6	3 0.25	3 0.12
counter	2 9 17 17	16 13,471	16 8,729
counter-bin	1 5 32 32	-- --.--	32 93,352
dme-e	3 3 8 10	3 0.40	3 0.19
dme-fast-e	3 3 8 10	5 0.45	4 0.29
hp-ir-sc-ctrl	13 14 33 42	13 51,10	13 65,625
isend	4 3 9 11	6 0.35	5 0.27
isend-bm	5 4 10 12	4 0.38	4 0.23
isend-csm	5 4 8 9	3 0.47	3 0.18
it-control	5 7 10 12	4 0.38	5 0.66
mp-f-p	3 4 4 4	2 0.23	2 0.14
nak-pa	4 5 6 6	2 0.27	2 0.16
nowick	3 2 6 6	2 0.58	2 0.15
opt-token-dist	4 4 12 12	6 0.40	6 0.68
pe-send-ifc	5 3 11 14	5 0.47	5 0.83
pscsi	10 5 45 62	10 --.--	9 58,178
pscsi-tesend-bm	4 4 10 12	6 0.48	6 1,191
ptrcvb1	4 4 7 9	4 0.35	4 0.25
qr42	2 3 4 4	3 0.27	3 0.15
re-setup	3 2 6 7	2 0.42	3 0.31
ring-counter	1 2 8 8	8 0.35	8 6,708
sbuf-sct	3 3 8 9	4 0.39	4 0.43
scsi-tsend-bm	5 4 11 13	5 0.86	5 0.55
sdcont2	8 12 27 32	13 11,554,329	13 17,156
stetson-p2	8 12 25 28	13 95,368	13 8,603
stetson-p3	4 2 8 11	3 0.33	3 0.21
strcv-bm	5 4 10 12	4 0.67	4 0.54
yun-diffeq-alu1	3 5 7 9	5 0.37	5 0.18
	total	161 11,662,668	160 149,671

VI. CONCLUSION

In this paper a novel method for state minimization for the XBM_AFSM was presented, which is based on genetic algorithm. The XBM_AFSMs were implemented in HM architecture, which potentially presents a better latency time and better interacts with fast environments, when compared to the HMFO architecture. This paper also introduced the concept of bridge state, which solves the problem of logical hazard associated with the LSS signals with non-monotonic behavior. As future work it is foreseen to finish the SAGAAs tool for logic synthesis of XBM_HM machines. For this we will develop algorithms for steps of state assignment and logic minimization.

REFERENCES

[1] E. G. Friedman, “Clock Distribution Networks in Synchronous Digital Integrated Circuits,” Proc. of. The IEEE, vo. 89, pp. 665-692, 2001.
[2] A. Jain et al., “A 1.2 GHz alpha microprocessor with 44.8 GB/s chip pin bandwidth,” in IEEE Int. Solid-State Circuits Conf. Tech. Dig., pp. 240–241, February, 2001.
[3] B. H. Calhoun, et al. “Digital Circuit Design Challenges and Opportunities in the Era of Nanoscale CMOS,” *Proceedings of the IEEE*, Volume 96, Issue 2, February 2008.
[4] C. J., Myers, “*Asynchronous Circuit Design*”, Wiley & Sons, Inc., 2004, 2a edition.
[5] T. Chelcea, et al., “A Burst-Mode Oriented Back-end for the Balsa Synthesis System,” Proc. Design, Automation and Test in Europe (DATE), pp.330-337, March 2002.
[6] J. Cortadella, et al., “Desynchronization: Synthesis of Asynchronous Circuits from Synchronous Specifications,” IEEE Trans. on CAD of Intr. Cir. and Sys. vol. 25, Nro. 10, pp.1904-1921, October 2006.

[7] J. Sparsø, “Current Trend in High-Level Synthesis of Asynchronous Circuits,” Proc.16th IEEE Int. Conf. on Electronics, Circuits and Systems, pp.347-350, 2009.
[8] J. Yang, et al., “HDLs Modeling Technique for Burst-Mode and Extended Burst-Mode Asynchronous Circuits,” IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, vol. 2010, N. 12, pp. 2590-2599, 2010.
[9] T. -A. Chu, “*Synthesis of Self-Timed VLSI Circuits from Graph-Theory Specifications*,” Ph.D. thesis, June, 1987, Dept. of EECS, MIT.
[10] B. Coates, A. Davis, and K. Stevens, “The post office experience: Designing a large asynchronous chip,” *Integration, VLSI J.*, vol. 15, no. 3, pp. 341–366, Oct. 1993.
[11] S. M. Nowick, “*Automatic synthesis of burst-mode asynchronous controllers*,” Ph.D. dissertation, Stanford Univ., Dept. Comput. Sci., Stanford, CA, 1993.
[12] K. Y. Yun e 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.
[13] S. M. Nowick et. al, “The Design of a High Performance Cache Controller: A Case Study in Asynchronous Synthesis” *Integration, the VLSI Journal*, Vol. 15, no 3, pp. 241-262, October 1993.
[14] K. Y. Yun, et al., "The design and verification of a high-performance low-control-overhead asynchronous differential equation solver," IEEE Transactions on VLSI Systems, vol. 6, no 4, pp.643-655, Dec.1998.
[15] K. S. Stevens, et al., "An asynchronous instruction length decoder," IEEE Journal of Solid-State Circuits, vol. 36, nro. 2, pp. 217-228, February, 2001.
[16] J. Muttersbach, “Globally-Asynchronous Locally-Synchronous Architectures for VLSI Systems,” Ph.D. Thesis, ETH, Zurich, 2001.
[17] J. Pontes, et al., “SCAFFI: an Intrachip FPGA asynchronous interface based on hard macros,” 25th Int. Conf. on Computer Design, pp.541-546, 2007.
[18] S. M. Nowick and B. Coates, “UCLOCK: Automated design of high performance asynchronous state machines,” in *Proc. Int. Conf. Computer Design (ICCD)*, pp. 434–441, October 1994.
[19] R. M. Fuhrer, et al., “Minimalist: An environment for the Synthesis, verification and testability of burst-mode machines,” Technical Report, Columbia University, TR-CUCS-020-99, 1999.
[20] R. M. Fuhrer, “*Sequential Optimization of Asynchronous and Synchronous Finite-State Machines: Algorithms and Too*ls,” Ph.D. Thesis, Columbia University, 1999.
[21] R. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued optimization for PLA optimization. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 6(5):727{750, September 1987.
[22] S. M. Nowick and D. L. Dill, “Exact two-level minimization of hazard free logic with multiple-input changes,” *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 986–997, Aug. 1995.
[23] S. H. Unger, “*Asynchronous Sequential Switching Circuits,”*. New York: Wiley-Interscience, 1969.
[24] J. F. Miller (editor), “*Cartesian Genetic Programming*,“ Springer, p.344, 2011.
[25] T. Curtinhas, et al., “A novel state assignment method for Extended Burst-Mode FSM design using Genetic Algorithm,” 27th Symposium on Integrated Circuits and Systems Design,” paper accepted, 2014.