Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

Assertion Based Verification of Multiple-Clock GALS Systems

Rostislav (Reuven) Dobkin, Tsachy Kapshitz, Shaked Flur and Ran Ginosar VLSI Systems Research Center, Technion-Israel Institute of Technology, Haifa 32000, Israel

(2013)

ABSTRACT

Standard EDA ABV tools fall short of verifying muhiple clock domain systems on chip (MCD SOC), asynchronous systems and Globally Asynchronous Locally Synchronous (GALS) systems. This paper describes a method for verifying asynchronous lind multi-clock behavior in such systems using PSL and standard ABV tools. We convert STG (signal transition graphs), a common form for specifying asynchronous behavior, into PSL statements, employ standard ABV tools, and formally prove complete verification. The proposed ASE (automatic sequence extraction) algorithm was applied to a MCD SoC model that employed a network-on-chip (NoC) .. for asynchronous inter-modular communications.

Keywords
ABV, MCD, SoC, GALS, Asynchronous logic

1. INTRODUCTION

Large systems on chip (SoC) may incorporate multiple modules operating at different frequencies. Moreover, in dynamic voltage and frequency scaling (DVFS) systems, frequency and voltage m.ay dynamically change during operation [I]-[3]. The resulting multiple clock domains (MCD) SoCs are treated as Globally Asynchronous Locally Synchronous (GALS) systems [4][5]. Inter-modular communications in MCD GALS systems are best implemented by asynchronous logic, eliminating multiple synchronization latencies and complex distribution of multiple clocks. Indeed, the ITRS predicts that by 2020 40% of SOC global signaling will be performed asynchronously [6]. However, to reliably employ asynchronous signaling, suitable verification techniques are required.

In a typical design and verification flow, the specification is fConverted into a design and also into verification statements (e.g. in PSL [7]). The design is typically verified with an 'assertion based verification' (ABV) tool [8}. ABV may be based on either simulation [9][I2} or formal verification (13][14] . In addition. advanced ABV supports temporal expression and/or data validity verification (PSL, e-Ianguage, System-Veri log, etc.) [7]-[12]. However. this scheme is often limited to clocked designs that employ a single clock, due to language limitations and tool constraints. Thus, verification by ABV is usually inapplicable to MCD systems and to any asynchronous circuits that may be included in the design.

Verification techniques for pure asynchronous logic [15]-[20] mostly employ custom tools, complicating their integration into typical design and verification flows. GALS system verification and test method was discussed in [21 J, where a special test extension was added to each GALS wrapper. The test extensions disconnect locally synchronous islands during test data transfer between different GALS wrappers, allowing stand-alone massive testing of the wrappers and their interconnections. This technique appears .. t.o_ be more test-oriented. In [22J a GALS wrapper was modeled by Petri nets and verified for reachability and deadlock using model checking [23J. Clock domain crossing (CDC) verification was discussed in [24], where structural and functional synchronizer verification was performed using PSL. These references do not provide a complete verification method for GALS systems.

One common form of specifying asynchronous behavior is based on signal transition graphs (STG) [25] which define untimed ordering of transitions. However, typical ABV tools cannot employ STG for verification of the design. In this work we combine STG specifications and temporal PSL expressions to enable CDC and asynchronous logic verification in MCD GALS systems. First, clock dori'l'liin crossings and other asynchronous components of the specification are presented formally using STG. Second, an algorithm is presented that converts STG specifications into PSL statements. Third, ABV is performed, using either an artificially generated clock or transition-sensitive verification. We prove that such verification is complete.

The paper is organized as follows. In Sect. 2 we describe the applicable STG and PSL properties. Tlie algorithm that converts STG to PSL is presented and analyzed in Sect. 3, and an example of a complex SoC verification is shown in Sect. 4.

2. STG AND PSL PROPERTIES

In this section we survey the applicable features of signal transition graph (STG) and property specification language (PSL), providing for the description of the algorithm in Sect. 3.

2.1 Signal Transition Graph (STG)

A module behavior can be described formally with a signal transition graph (STG). An example of STG for a simple latch controller is shown in Figure I. The STG is a special type of a Petri Net (15]. Tokens are marked by solid circles and their position (marking) determine the circuit state; the token marking

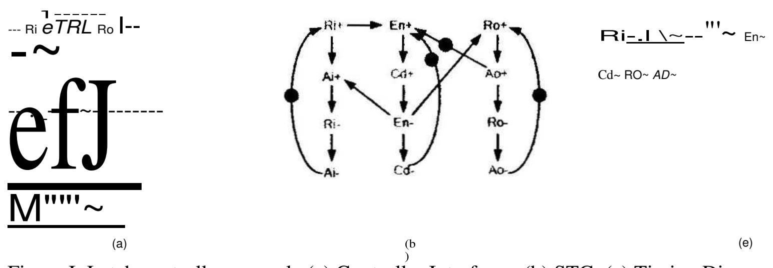

Figure I: Latch controller example (a) Controller Interfaces, (b) STG, (c) Timing Diagram

in Figure I, denotes the initial state. Change of state is denoted by moving tokens along directed edges. A transition of node *n* is enabled when every incoming arc holds a token. When the transition takes place (node *n* "fires"), all incoming tokens are consumed and new tokens are produced on each outgoing arc. STG may also specify choice and merge conditions, which are not shown in this paper, but can be also treated by the ASE approach. The STG can be used for logic synthesis, for example using Petrify tool, which also performs formal verification of the synthesized logic [IS]. Unfortunately, Petrify cannot be used for large system synthesis and verification. In addition, when gatelevel asynchronous design is obtained manually or by tools without formal verifier inside, the verification of internal structure is an essential condition for the design sign-off.

Note that STG tokens circulate in the STG for each new word in a repeating manner. While the cross-relation between the tokens may change for different cycles, the path that a single token goes through is never changed (this is partially true for STGs with choice, where current path is chosen from a certain number of predefined paths according to choice input value).

2.2 Property specification language (PSL)

The PSL language provides operators for defining and verifying timed sequences. For example, the following expression employs the '->' and 'eventually!' PSL operators to verify that acknowledge signal AI is asserted each time request signal RI is asserted.

```
property reqack.iny is always (RI->eventually! At);
```

More complex relations can be defined by Sequential Extcnded Regular Expressions (SERE). A SERE makes it easier to define long sequences, allows re-use of shared sequences and can be used in conditional statements. For example, a simple four-phase handshake protocol (RI+ -7 AI+ -7 RI- -7 AI-) can be defined as follows:

```
sequence hs jnit is {not RI; RI};
sequence hsbody is {RI; AI; not RI; not All;
property sereexarnp is always { hS,inil : 1-> { hsbody l;
assert sereexamp;
```

The brackets define sequences. hs _init expresses the initial transition of the sequence (RI+) and hsbody contains the remaining transitions. The *sere.examp* property uses the "always" operator to specify that it must be valid at all times. The assert statement actually initiates verification of the property.

These sequence examples do not employ any clock. This is important when verifying multiple clock domains: PSL is defined only for a single clock. This verification code may be used in two ways. First, the ABV may be event-based, and asynchronous transitions are handled at arbitrary times rather than on any external clock ticks. Alternatively, a default verification clock may be defined. In any case, the verification is independent of any clock event ordering of external multiple clocks.

Verification effectiveness is measured by coverage. The next example collects coverage for the sequence *hs_body:*

```
cover hsbody;
```

3. AUTOMATIC SEQUENCE EXTRACTION (ASE)

Our goal is to generate assertion expressions .. for ARV from system level specification of GALS system. We use STGs for specification and then apply the Automatic Sequence Extraction (ASE) algorithm. ASE decomposes the STG into STC-I, a set of cyclic non-splitting circles, which are then transformed into PSL assertions. In this section we present all definitions, provide a formal analysis to prove the correctness of the STG decomposition into STC-I (Sect. 3.1), describe the ASE algorithm in Sect. 3.2 and provide an example in Sect. 3.3.

3.1 STG Decomposition and Complete Verification

In this section we prove that the proposed STG decomposition is correct, namely that the ·41enerated PSL assertions preserve the original STG specification.

Definitions

I. Signal transition graph (STG) is a connected directed graph G=(*V,E,* 7), where *V* is a set of nodes representing signal transitions ("+" and ","), *E* are directed edges showing precedence relations, and *T* is the initial marking ('marking' is a set of edges having tokens on them). The STG follows three sets of rules:

a. When all edges leading into a transition have tokens; the transition may "fire", the said tokens are consumed and new tokens are placed on all edges emanating from the fired transition.

b. In this work, STGs are free from deadlocks, I-bounded (no more than one token per edge), and have only input free choices [15][26]. This also means that there are neither source nodes nor sink nodes in the STG, and every node may be revisited infinitely many times.

c. The STG specifies a speed-independent system, namely it has consistent state assignment (transitions strictly alternate between ~+" and "-") and is persistent (enabled transitions must eventually fire) [15][26].

References

[1] J. Campbell, “Speaker recognition: a tutorial,” *Proc. IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.

[2] D. A. Reynolds, T. Quatieri, and R. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.

[3] D. A. Reynolds, “Comparison of background normalization methods for text-independent speaker verification,” in *Proc. Eurospeech*, 1997.

[4] D. A. Reynolds and R. C. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models,” *IEEE Trans. Speech Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[5] J. L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, Apr. 1994.

[6] E. Bocchieri, “Vector quantization for the efficient computation of continuous density likelihoods,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. 692–695.

[7] K. M. Knill, M. J. F. Gales, and S. J. Young, “Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs,” in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[8] D. B. Paul, “An investigation of Gaussian shortlists,” in *Proc. Automatic Speech Recognition and Understanding Workshop*, 1999.

[9] T. Watanabe, K. Shinoda, K. Takagi, and K.-I. Iso, “High speed speech recognition using tree-structured probability density function,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1995.

[10] J. Simonin, L. Delphin-Poulat, and G. Damnati, “Gaussian density tree structure in a multi-Gaussian HMM-based speech recognition system,”

[11] T. J. Hanzen and A. K. Halberstadt, “Using aggregation to improve the performance of mixture Gaussian acoustic models,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1998.

[12] M. Padmanabhan, L. R. ahl, and D. Nahamoo, “Partitioning the feature space of a classifier with linear hyperplanes,” *IEEE Trans. Speech Audio Processing*, vol. 7, no. 3, pp. 282–288, 1999.
in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[13] R. Auckenthaler and J. Mason, “Gaussian selection applied to text-independent speaker verification,” in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[14] J. McLaughlin, D. Reynolds, and T. Gleason, “A study of computation speed-ups of the GMM-UBM speaker recognition system,” in *Proc. Eurospeech*, 1999.

[15] S. van Vuuren and H. Hermansky, “On the importance of components of the modulation spectrum of speaker verification,” in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[16] B. L. Pellom and J. H. L. Hansen, “An efficient scoring algorithm for Gaussian mixture model based speaker identification,” *IEEE Signal Processing Lett.*, vol. 5, no. 11, pp. 281–284, 1998.

[17] J. Oglesby and J. S. Mason, “Optimization of neural models for speaker identification,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1990, pp. 261–264.

[18] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, “Global optimization of a neural network—hidden Markov model hybrid,” *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[19] H. Bourlard and C. J. Wellekins, “Links between Markov models and multilayer perceptrons,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1167–1178, Dec. 1990.

[20] J. Navrátil, U. V. Chaudhari, and G. N. Ramaswamy, “Speaker verification using target and background dependent linear transforms and multi-system fusion,” in *Proc. Eurospeech*, 2001.

[21] L. P. Heck, Y. Konig, M. K. Sonmez, and M. Weintraub, “Robustness to telephone handset distortion in speaker recognition by discriminative feature design,” *Speech Commun.*, vol. 31, pp. 181–192, 2000.

[22] A. Dempster, N. Laird, and D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. R. Statist. Soc.*, vol. 39, pp. 1–38, 1977.

[23] K. Shinoda and C. H. Lee, “A structural Bayes approach to speaker adaptation,” *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.

[24] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.

[25] J. C. Junqua, *Robust Speech Recogntion in Embedded Systems and PC*

[26] U. V. Chaudhari, J. Navrátil, S. H. Maes, and R. A. Gopinath, “Transformation enhanced multi-grained modeling for text-independent speaker recognition,” in *Proc. Int. Conf. Spoken Language Processing*, 2000.

[27] Q. Lin, E.-E. Jan, C. W. Che, D.-S. Yuk, and J. Flanagan, “Selective use of the speech spectrum and a VQGMM method for speaker identification,” in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[28] S. Raudys, *Statistical and Neural Classifiers: An Integrated Approach to Design*. New York: Springer, 2001.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986, pp. 318–364.

[30] [Online] Available: http://www.nist.gov/speech/tests/spk/index.htm.

[31] J. Pelecanos and S. Sridharan, “Feature warping for robust speaker verification,” in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[32] B. Xiang, U. V. Chaudhari, J. Navrátil, N. Ramaswamy, and R. A. Gopinath, “Short-time Gaussianization for robust speaker verification,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2002.

[33] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, “The NIST speaker recognition evaluation—overview, methodology, systems, results, perspective,” *Speech Communication*, vol. 31, pp. 225–254, 2000.

Bing Xiang (M’03) was born in 1973 in China. He received the B.S. degree in radio and electronics and M.E. degree in signal and information processing from Peking University in 1995 and 1998, respectively. In January, 2003, he received the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY.

From 1995 to 1998, he worked on speaker recognition and auditory modeling in National Laboratory on Machine Perception, Peking University. Then he entered Cornell University and worked on speaker recognition and speech recognition in DISCOVER Lab as a Research Assistant. He also worked in the Human Language Technology Department of IBM Thomas J. Watson Research Center as a summer intern in both 2000 and 2001. He was a selected remote member of the SuperSID Group in the 2002 Johns Hopkins CLSP summer workshop in which he worked on speaker verification with high-lelvel information. In January, 2003, he joined the Speech and Language Processing Department of BBN Technologies where he is presently a Senior Staff Consultant-Technology. His research interests include large vocabulary speech recognition, speaker recognition, speech synthesis, keyword spotting, neural networks and statistical pattern recognition.

Toby Berger (S’60–M’66–SM’74–F’78) was born in New York, NY, on September 4, 1940. He received the B.E. degree in electrical engineering from Yale University, New Haven, CT in 1962, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA in 1964 and 1966, respectively.

From 1962 to 1968 he was a Senior Scientist at Raytheon Company, Wayland, MA, specializing in communication theory, information theory, and coherent signal processing. In 1968 he joined the faculty of Cornell University, Ithaca, NY where he is presently the Irwin and Joan Jacobs Professor of Engineering. His research interests include information theory, random fields, communication networks, wireless communications, video compression, voice and signature compression and verification, neuroinformation theory, quantum information theory, and coherent signal processing. He is the author/co-author of Rate Distortion Theory: A Mathematical Basis for Data Compression, Digital Compression for Multimedia: Principles and Standards, and Information Measures for Discrete Random Fields.

Dr. Berger has served as editor-in-chief of the IEEE Transactions on Information Theory and as president of the IEEE Information Theory