A Novel Strategy for Formal Verification of Asynchronous Circuit Design in PAiD tool

Tin Thien Nguyen[1], **Khoi-Nguyen LE-HUU**[2], **Thang H. Bui**[3], and **Anh-Vu Dinh-Duc**[4], Non-members

ABSTRACT

EDA has been proposed for a long time as a category of reliable software tools for designing electronic systems. Although some of them have been considered as powerful tools for asynchronous circuits, a prominent approach to cope with the biggest defect of synchronous circuits: clock distribution issue, researches in verifying the correctness of those circuits are still limited. Therefore, an enhanced version of PAiD, an EDA tool that has been developed at Ho Chi Minh City University of Technology (HCMUT), will be proposed in this work along with case studies. It will enable engineers not only design, synthesize asynchronous circuits but also verify them. Furthermore, a good strategy to improve the verifying performance is also discussed.

Keywords: Asynchronous Circuit Design, EDA tool, Formal Verification, Model Checking

1. INTRODUCTION

In the state of the art of digital circuit design, synchronous circuit has been become the dominant approach since 1960's [4]. This circuit utilizes the facilities of a periodic timing signal, called clock, to synchronize the operations of its components. However, the clock signal has to be distributed globally to entire circuit. As the circuit's size becomes larger and more complex, this results in some main problems such as clock skew, jitter and high power consumption. To overcome these drawbacks, a self-time circuit, called asynchronous circuit [4], has been taken much attention from researchers. Instead of using clock signal, the asynchronous circuit does synchronization locally by mean of handshaking protocols ([1][4][16][21]). This circuit is considered as a promising approach for concurrent systems such as SoC (system on chip) and NoC (network on chip). Asynchronous circuit has rapidly attracted most of concerns from not only academia but also industry. This researching field covers many interesting aspects related to circuit design process such as circuit description languages, simulation, synthesis and verification.

Obviously, describing circuits in high-level abstract languages plays an important role in digital circuit design. It allows designers to focus on the circuit functioning and leave the lower level implementation beside. In asynchronous circuit design, some description languages have been proposed such as CSP [10], CHP [6] and ADL [8]. Their main principle is to concentrate on the description of concurrent process that is the nature of asynchronous circuit. The later - ADL - is the extension of CHP language with additional structures that are optimized to describe asynchronous circuit more efficiently.

In addition to circuit design, it is necessary to simulate the circuit functions before proceeding further to lower-level design steps. This can be done with the help of Petri net [17]. Petri net (PN) is claimed to be a good representation for concurrent systems. However, it lacks of some conditional structures that might be commonly existed in asynchronous circuit description. To scope with this obstacle, authors in [15] has already combined it with the DFG - Data Flow Graph - to generate a more suitable intermediate representing model, called PN-DFG. This model has been proved its efficiency in representing [15], simulating [20], placing and routing [25] and technology mapping [29] of asynchronous circuit at high level of abstraction.

Although asynchronous circuit has been studied for decades, the research aspects are mostly separated. Putting it all together in order to build an entire EDA tool is very important for both academia and industry. Two notable EDA tools are TAST [7] and PAiD [9] that have been successfully synthesized many asynchronous systems. The former is developed at TIMA Lab, France while the latter is the generated at HCMC University of Technology, Viet Nam. The PAiD tool covers many level of circuit design from high-level description using ADL to the implementation at logic-gate level. This tool is very efficient for either researching or teaching at university.

In order to help asynchronous circuit be used widely, it must be significantly verified for important properties such as its correctness and reliabil-

Manuscript received on July 15, 2014 ; revised on January 27, 2015.

Final manuscript received March 16, 2015.

[1,3] The authors are with Ho Chi Minh City University of Technology - VNUHCM, Vietnam., E-mail: thientin@cse.hcmut.edu.vn and thang@cse.hcmut.edu.vn

[2,4] The authors are with University of Information Technology - VNUHCM, Vietnam., E-mail: khoinguyen@uit.edu.vn and anhvu@uit.edu.vn

ity. Moreover, in industrial circuit design, the sooner the bugs in circuit design are detected, the more the cost saving people will have. This requirement has motivated many researches in asynchronous circuit verification. One of the promising approaches is to apply formal verification [24]. This methodology consists of two main methods that are theorem proving and model checking. While theorem proving requires the expert background in mathematic [11], model checking take the advantages of modern computer system power [18]. Given the circuit description and its specification, model checking tool will handle the rest of verifying process. Researches of applying model checking in hardware design have been proposed for years ([12][18][23]). One of their notable achievements is the NuSMV model checker when it can increase the number of system's states that can be verified [2].

In this paper, a novel symbolic model checking - based approach for asynchronous circuit is proposed in details. Based on NuSMV tool, this method utilizes not only PN-DFG intermediate model to represent circuits but also symbolic technique to reduce the computational complexity. Moreover, two possible verifying strategies are also discussed in this paper.

The rest of this paper is constructed as follows: Section 2 is for technical background such as asynchronous circuit description and representation, PN-DFG intermediate representing model, model checking approach and the transformation from PN-DFG to model checking model. The architecture of PAiD tool with verification module is in Section 3. Section 4 discusses possible verifying strategies. Some case studies are provided in Section 5. The last section is for conclusion and the future work.

2. TECHNICAL BACKGROUND

2.1 Asynchronous Description Language (ADL)

ADL is a high-level abstraction language designed to describe systems that contain many concurrent processes. In ADL, the communication between those processes is abstracted as communicating channel with necessary operations such as read or write. It, therefore, enables designers to concentrate on behaviours of the circuit without worrying about the low level implementation, for example, communicating protocols or structures. ADL was first designed as basic asynchronous circuit description language of PAiD tool.

Fig. 1 represents block diagram of a 1-bit input asynchronous decoder.

The input is read and stored by Buffer module. Then, Buffer module will communicate with Decoder module via channel C. According to the value received, Decoder module will assert one of its two outputs. The ADL description of the circuit is shown in Fig. 2.

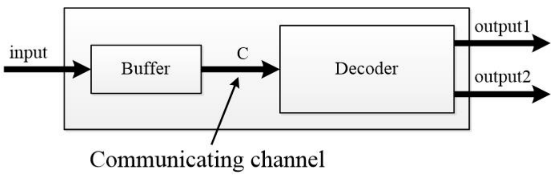


Fig.1: *An asynchronous decoder.*

2.2 PN-DFG model

PN-DFG model is the combination of Petri nets (PN) and Data Flow Graph (DFG). In this model, the DFG can be attached to transition or place of original PN. This modification makes PN-DFG model become a very efficient representing model of asynchronous circuits.

```
Module Buffer
    (in inChannel: bit,
     out outChannel: bit)

    Variable C_value: bit;
    inChannel  >> C_value;
    outChannel << C_value;
End Module

Module Decoder
    (in inChannel:bit,
    out outChan1: bit,
    out outChan2: bit)

    Variable inValue: bit;
    inChannel >> inValue;
    if (inValue = 0) then
       outChan2 << 0; outChan1 << 1
    else
       outChan1 << 0; outChan2 << 1
End Module

Main
    (in input: bit,
    out output1: bit,
    out output2: bit)

    Channel C: bit;
    Buffer(input, C)
    Decoder(C, output1, output2)
End Main
```

Fig.2: *ADL description of decoder in Fig. 1.*

The most notable aspect of PN-DFG model is the different roles of DFG when attaching to transition or place. When a DFG comes with a transition, it is responsible for guarding the fire action of the transition. In contrast, when it comes with a place, the DFG is responsible for representing operation that will be carried out whenever the place holds token.

PN-DFG model is claimed that it can model any asynchronous circuits. Each process is represented by one PN-DFG model. Fig. 3 illustrates the PN-DFG model of the Buffer module in above asynchronous decoder example.

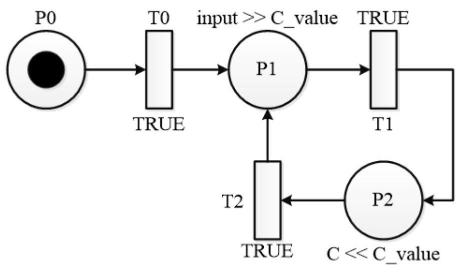


Fig.3: *PN-DFG model of module Buffer in Fig. 1.*

2.3 Model checking

Model checking [13] is one of well-known approaches in system formal verification. It models system as finite-state transition model, and then the desired property is checked for every approached state systematically.

In model checking, the property is represented formally by temporal logic expression, which is proposition qualified in term of time. The two typical temporal logic systems are Linear Temporal Logic (LTL) and Computation Tree Logic (CTL).

To attack the state space explosion problem, four typical solutions have been derived including (1) symbolic representation, (2) partial order reduction, (3) abstraction and (4) composition [14][26]. The first solution uses symbolic representation in order to increase the number of state that a model checker can represent. The latter three aims to decrease the size of system, so that the model checker can handle it.

The symbolic model checking [22] is proved its capacity for representing a model with (beyond) 1020 states. In our approach, the symbolic model checking is chosen as the main formal method for verifying asynchronous circuit. In addition, the well-known NuSMV model checking tool is used [3].

2.4 NuSMV representation of PN-DFG model

2.4.1 System representation

In ADL, a concurrent system is usually constructed by many modules. The Decoder in Fig. 1, for example, consists of two modules that are Buffer and Decoder. It leads to two type of representation in NuSMV. The first one - system as a whole - merges all PN-DFG modules into only one NuSMV module. The second one - system as components - uses one NuSMV module for each PN-DFG module. These two approaches are examined and turned out that the system-as-components are more suitable in representing asynchronous circuits [28].

2.4.2 PN-DFG model in NuSMV

By considering the correspondence between PN-DFG model and NuSMV module, the transformation rules are built to represent PN-DFG model by NuSMV input language - SMV language. These rules are described clearly in [27]. Following summarizes some of its important principles.

i. Place is transformed into Boolean variable whose true value implies that corresponding place holds token.
ii. Transition and its attached DFG are transformed into the firing condition represented by a variable under DEFINE keyword. This variable has true value if the transition is fired in original PN and the attached DFG is evaluated to be true.
iii. All actions related to transition firing are described under TRANS keyword. They include toggling corresponding place variables and executing operations of the attached DFG in the outgoing places.
iv. Finally, all partial transformed modules of a system will be combined in the main module in NuSMV.

3. PAID ARCHITECTURE

In this section, the general design structure of PAiD tool will be described. In addition, its verification model is also taken into account.

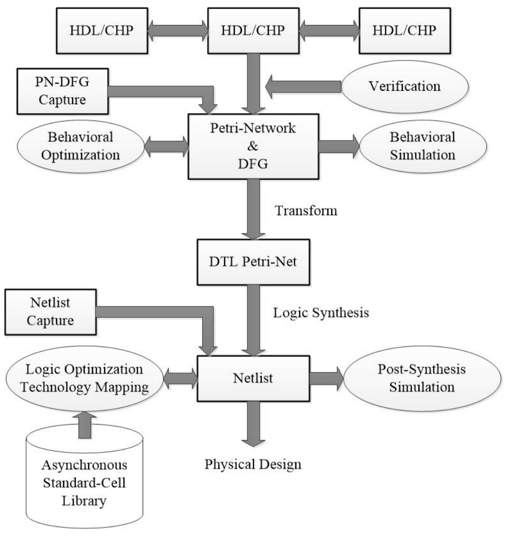


Fig.4: *The architecture of PAiD tool.*

3.1 PAiD tool architecture

The architecture of PAiD tool is shown in Fig. 4. In general, the design flow of an asynchronous circuit using PAiD tool will consists of four main phases. Firstly, the circuit specification is described by high-level abstraction languages such as ADL or CHP. Secondly, an intermediate representing model - PN-DFG model is generated based on the circuit's description. This model is very useful for either simulating or optimizing the circuit's behaviours. Thirdly, the synthesis is carried out. And finally, the circuit at gate net-list level is generated. Further information about the details of PAiD architecture can be found in [30].

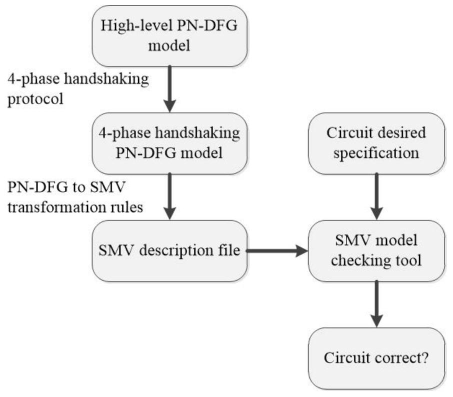

High-level PN-DFG model
4-phase handshaking protocol
4-phase handshaking PN-DFG model
Circuit desired specification
PN-DFG to SMV transformation rules
SMV description file
SMV model checking tool
Circuit correct?

Fig.5: *PAiD verification module design flow.*

3.2 PAiD verification module design flow

The original design flow of PAiD tool is just a one-way flow. Hence, the additional verification module will make it more reliable. Moreover, this module aims at verifying the desired circuit specification at the highest level - description level. The design flow of verification module is illustrated in Fig. 5.

First stage of the verifying design flow relates to expansion of high-level PN-DFG. As mentioned so far, the ADL provides communicating channel for exchanging information between processes. This channel is abstracted at high level; hence, the verification module has to expand the channel so that it can be represented and verified in NuSMV. This is illustrated in Fig. 6. The 4-phase handshaking protocol is applied to construct 4-phase handshaking PN-DFG model from high-level PN-DFG model. Fig. 6 illustrates an example of the expanded PN-DFG model. Sender module writes Data to the channel C and Receiver module reads that value into variable X.

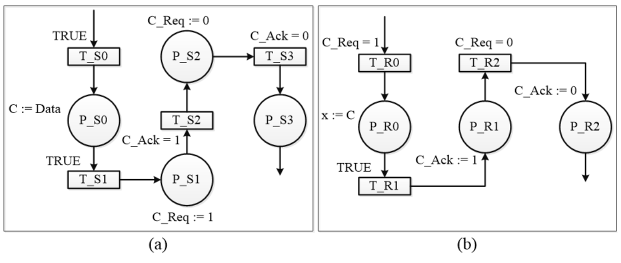


Fig.6: *4 phase handshaking PN-DFG model: (a) Sender, (b) Receiver.*

The next stage is to apply transformation rules to represent the expanded PN-DFG module in NuSMV. In addition, the circuit property is expressed in term of LTL or CTL formula. Then, NuSMV is run to verify if the ADL description of circuit conforms the desired property.

4. VERIFYING STRATEGY

This section describes two strategies that can be applied to verify asynchronous circuits using symbolic model checking.

4.1 General verification method

This method follows exactly the design flow of verification module described previously. In this approach, all channels are expanded by 4-phase handshaking protocol.

For example, the 4-phase handshaking protocol described in Fig. 6 requires up to six places, six transitions and two more variables (C_Req and C_Ack) to synchronize the communication process of the two modules over one channel. Consequently, it enlarges the number of state that NuSMV has to verify. Moreover, those DFGs attached to additional transitions also make the NuSMV computation more complex. When the system is more complicated, it may consist of many communicating channels. Hence, the situation may lead to state space explosion problem easily. That's why we need a more efficient verification method.

4.2 Novel verification method

In this novel verification method, a strategy is developed in order to reduce the size of the expanded PN-DFG model. This goal can be met by eliminating communicating channel expansion that is not necessary.

This approach derives a verifying strategy as following steps.

1. Verify each partial PN-DFG model independently to make sure it functions properly. This work can be done easily since these PN-DFG models may have small size.

2. Try to combine two or more small components that are verified in step 1 into larger components if possible to save handshaking stages.

By doing this approach, two processes that communicate over the eliminated channel will be merged together, and so an appropriate method needs to be developed to retain synchronization. A novel idea would be to create two places called P_write and P_read. The P_write place informs receiving process about the validation of data over the eliminated channel. In contrast, the P_read place informs transmitting process that the data is already absorbed by the receiving process. These two places are only responsible for synchronizing communication between processes, and so there is no further action for them. Therefore, their DFGs would be "skip" DFGs.

Detailed information on how to expand channel operations is discussed shortly. Those operators include transmitting operator ($\ll$), receiving operator ($\gg$) and probe operator (#).

4.2.1 Transmitting operator

The transmitting operator is expanded as shown in Fig. 7.

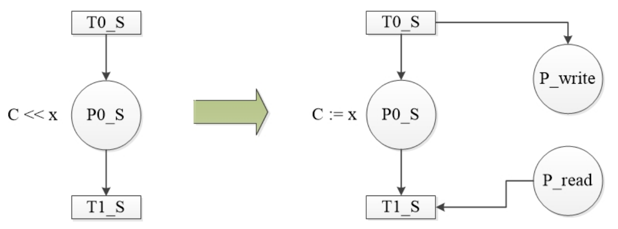


Fig.7: *Expanded PN-DFG represents transmitting operator in Novel verification method.*

According to the expansion, the transmitting process can be divided into three phases: (1) checking the channel if it is free (2) writing data and (3) waiting for receiver accepts the data. In the first phase, the transmitting process will be hold until the P_write place has no token. Then, the second phase will perform two operators: sending data X is written out to channel C and assigning a token to P_write place. Finally, in third state, the Sender will wait until P_read hold token to fire T1_S transition.

4.2.2 Receiving operator

The receiving operator is expanded as shown in Fig. 8. In contrast to the expansion of transmitting operator, the receiving operator is only divided into two phases: (1) waiting for valid data and (2) reading the data.

In the first phase, the receiver waits until P_write place hold token which means the data is written to channel already. Then T0_R transition fires in the second phase, and so the data is read into variable y. In addition, P_read place is assigned with a token in order to inform the sender of data communicating accomplishment.

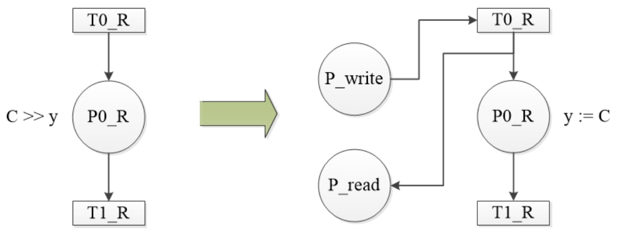


Fig.8: *Expanded PN-DFG represents receiving operator in Novel verification method.*

4.2.3 Probe operator

Communicating channel is used not only for transferring data but also for synchronizing between two processes. The probe operator plays an important role in the latter mode. The probe operation is an action of querying the status of communicating channel. Its expansion is depicted in Fig. 9.

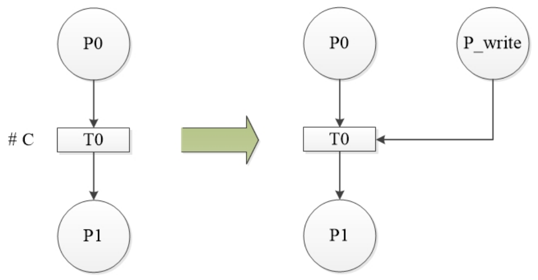


Fig.9: *Expanded PN-DFG represents probe operator in Novel verification method.*

It is understandable that the current state of communicating channel is determined by the P_write place. If this place hold token, the channel is busy. Otherwise, it is free.

4.2.4 Combining operators

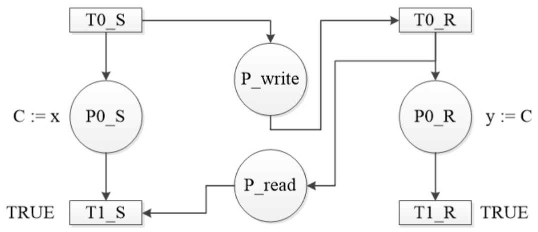


Fig.10: *Expanded PN-DFG represents channel communicating in Novel verification method.*

Based on the expansion of individual operators, the PN-DFG that represents communicating between two processes is formed as shown in Fig. 10.

The expansion technique in Novel verification method only requires two places for each eliminated channel despite how many transmitting operator or receiving operator there are in each process. Fig. 11 represents the expansion of PN-DFG for two transmitting operators and one receiving operator. The two transmitting operator are sharing two place P_write and P_read.

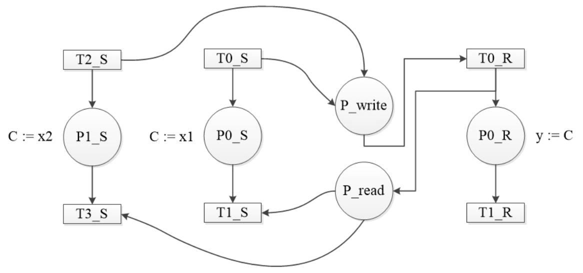


Fig.11: *Expanded PN-DFG represents channel communicating with two transmitting operator in Novel verification method.*

4.2.5 Novel PN-DFG expansion flow

The general flow of PN-DFG expansion in Novel verification flow is shown in Fig. 12. As indicated in the flow, there may be some channels that are not eliminated. Therefore, those channels will be expanded by 4-phase handshaking protocol.

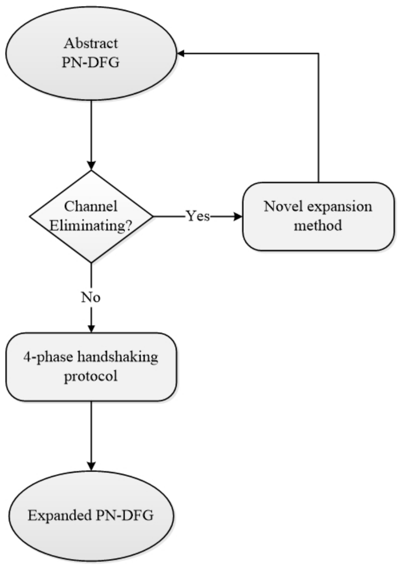


Fig.12: *Novel PN-DFG expansion flow.*

4.3 Channel expansion comparison

This section gives some simple comparisons between two expansion techniques: (1) 4-phase handshaking protocol and (2) channel eliminating. They include the number of place, transition and additional variables that required expanding the transmitting and receiving operators. Details are shown in Table 1.

The 4-phase handshaking technique requires a large number of places and transitions when there are many channel operators. In contrast, the channel eliminating technique only use two place for certain eliminated channel. Therefore, this technique is expected to reduce the number of system state. As a result, it can enhance the efficiency of verification process.

Table 1: *TTC Results and CPU Time from EP, conventional PSO, and hybrid-PSO on IEEE RTS 24-bus system.*

	N transmitting operator & M receiving operator		
	No. Place	No. Transition	Additional Variable
4-phase handshaking	3N + 3M	3N + 3M	4
Channel eliminating	2	0	0

5. EXPERIMENTATION

In this section, the verification methods described so far will be applied to some case studies in order to check their capacity in verifying asynchronous circuits.

5.1 Case studies

The experimentation is performed on four case studies. They are asynchronous arbiter, selector, distributed mutual exclusion and asynchronous pipelined finite impulse response filter.

5.1.1 Asynchronous Arbiter

Arbiter [31] is a well-known device used to control access to a shared resource among different processes. Fig. 13 shows block diagram of an arbiter in a system with two processes that want to access to a common resource.

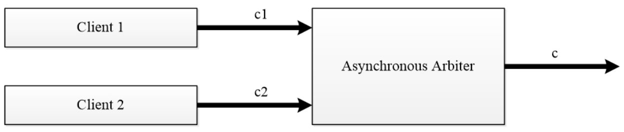


Fig.13: *Architecture of asynchronous arbiter.*

The specification “If the channel c1 issues an access request, there should be an execution path so that c presents 1 eventually” is chosen to verify.

5.1.2 Asynchronous Selector

This asynchronous selector block diagram is shown in Fig. 14. Its function is to determine which output channel the data from channel E will pass through depending on control value from channel C.

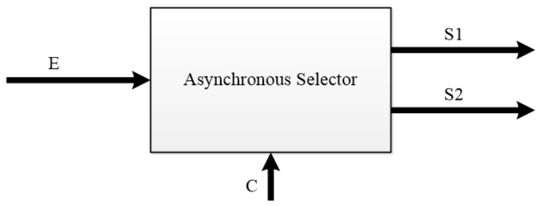


Fig.14: *Architecture of the asynchronous selector.*

The specification ”If channel C sends out value 0, the data from channel E should be passed to S1 eventually” is chosen to verify.

5.1.3 Distributed Mutual Exclusion

The Distributed Mutual Exclusion (DME) is a well-known mutual exclusion problem between N concurrent processes. A distributed algorithm requires at most N message exchanges for one mutual exclusion invocation. Fig. 15 represents the model of DME that has three processes.

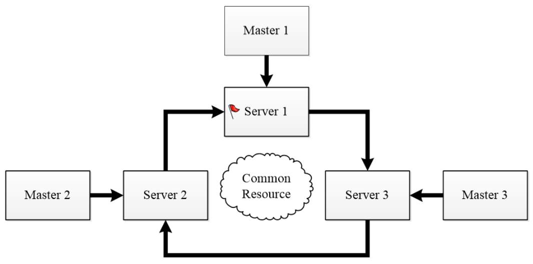


Fig.15: *Model of 3-process DME.*

This case study implements "The Reflecting Privilege" [5] solution for this DME problem. The specification "Only one process can access common resource at a time" is chosen to verify.

5.1.4 Asynchronous Pipelined Finite Impulse Response Filter

This benchmark illustrates the pipelined implementation of a finite impulse response (FIR) filter. Different from the FIR filter described in [19], the filter mentioned in this work is described by the following equation:

$$\text{y(n)=h(n)} * \text{x(n)} = \sum_{\text{k=0}}^{\text{N-1}} \text{h(k)} \times \text{x(n-k)}$$

The filter is so-called N-tap FIR filter. Fig. 16 shows the design of a 3-tap asynchronous pipelined FIR filter.

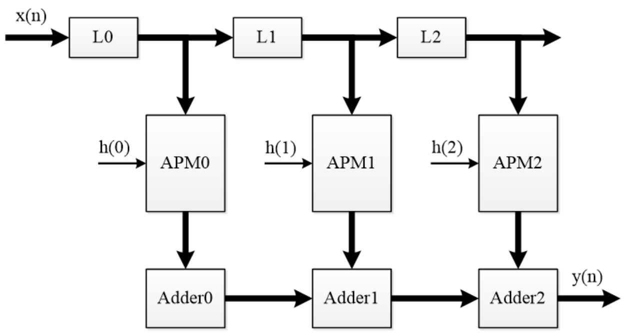


Fig.16: *Architecture of the asynchronous pipelined FIR filter.*

The specification "Each buffer must read its predecessor buffer's value and then send the value to its successor buffer" is chosen to verify.

5.2 Running environment setup

The running environment is set on Intel Core i7 - 4770 CPU 3.40Gh x 8 processor. The memory size is 15.4 GB. The PC runs Ubuntu 13.04.

The NuSMV version is 2.5.4.

5.3 Running results

The experimentation results include (1) experiment on transforming process form PN-DFG model to SMV description, and (2) running results of NuSMV.

5.3.1 PN-DFG model to SMV description transforming results

This experiment collects the information about the number of places and transitions that are generated in SMV description file after being transformed from PN-DFG model. Table 2 shows this experiment result. The first column, N column, describes the complexity of benchmarks.

Table 2: *Experiment on PN-DFG model to SMV description transformation.*

N	No. of places		No. of transitions	
	General strategy	Novel strategy	General strategy	Novel strategy
Asynchronous Arbiter				
	29	-	28	-
Asynchronous Selector				
	35	-	36	-
Distributed Mutual Exclusion				
2 cells	86	76	92	76
3 cells	129	114	138	114
4 cells	172	152	184	152
6 cells	258	228	276	228
8 cells	344	304	368	304
Asynchronous Pipelined FIR Filter				
2-tap	87	71	84	68
3-tap	130	110	126	105

This table aims at comparing results from two verifying strategies described previously. Due to the first two case studies are too simple, applying the novel strategy is totally unnecessary. Thus, it is marked as "-".Table 2 also indicates that the number of places and transitions reduce proportionally to the size of the case studies. This reduction is expected to enhance the computation of NuSMV.

5.3.2 NuSMV running results

NuSMV running results include: (1) Running time spent on NuSMV tool to finish process of verifying desired property and (2) the number of BDD nodes that are generated by NuSMV while it is running. The running time indicates how fast the circuit specification is verified. On the other hand, the number of BDD nodes provides information about how complex the benchmark is. NuSMV experiment results are summarized in Table 3.

It is noticeable that verification module can easily verify small systems such as asynchronous arbiter, asynchronous selector and 2-cell DME in extremely short time (less than one second) using either general strategy or novel strategy.

For verifying 6-cell DME benchmark, the general strategy mode required more than 7 million BDD

Table 3: *Experiment on running NuSMV.*

N	Running time (s)		No. of BDD nodes	
	General strategy	Novel strategy	General strategy	Novel strategy
Asynchronous Arbiter				
	0.02	-	20,234	-
Asynchronous Selector				
	0.07	-	140,211	-
Distributed Mutual Exclusion				
2 cells	0.22	0.10	614,474	392,800
4 cells	2.03	0.62	1,449,572	1,729,723
6 cells	49.82	0.50	7,585,691	1,565,183
8 cells	Time out	0.99	-	1,370,241
Asynchronous Pipelined FIR Filter				
2-tap	141.22	4.28	21,081,300	744,715
3-tap	Time out	154.99	-	3 ,404,546

nodes; however, only about 2 minutes are needed for verifying its property.

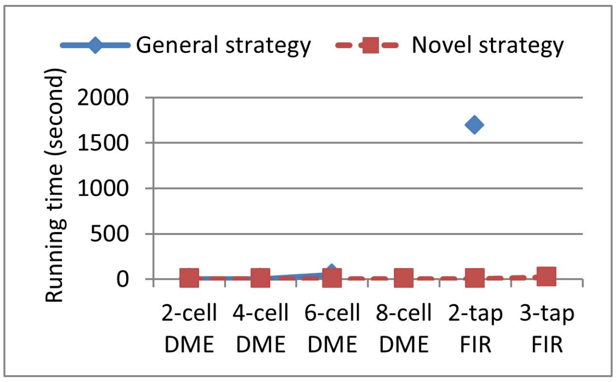


Fig.17: *Comparison of NuSMV running time of two verifying strategies.*

Table 3 also contains two situations that the state space explosion occurs when applying general strategy. These two complicated case studies are 8-cell DME and 3-tap FIR filter. NuSMV running time is marked as "Time out" to indicate that verifying process cannot finish in given expecting time. This time is pre-set by 4 hours.

The number in Table 3 clearly proves the genius of novel verifying strategy. It obviously helps reduce the number of BDD nodes in every case study except for 4-cell DME. Moreover, it makes NuSMV run significantly faster. For example, the 6-cell DME requires less than one minute to be verified and less than 5 seconds is taken to verify the 2-tap FIR filter. In case of 4-cell DME, although it requires more BDD nodes when applying novel strategy mode, the running time is indeed much smaller. This abnormal case will be investigated more in the near future.

It is also absolutely astonishing that the novel strategy can handle the two complicated situations that general mode cannot. In fact, it only took one second to verify the 8-cell DME and 3 minutes for 3-tap FIR filter. The comparison of NuSMV running time of two verifying strategies is illustrated clearly by the graph in Fig. 17.

In addition, by using Novel verification strategy, the BDD nodes are reduced substantially - 1.4 million nodes for 8-cell DME and 3.5 million nodes for 3-tap FIR filter. Detailed comparison of BDD node of two strategies is shown by the graph in Fig. 18.

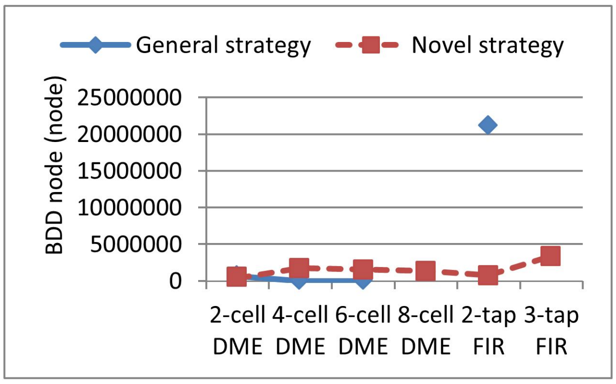


Fig.18: *Comparison of BDD node in two verifying strategies.*

6. CONCLUSION

This paper already described PAiD tool for designing and synthesizing asynchronous circuits. The symbolic model checking based verification module is also presented. Some case studies have been examined and proven the feasibility of this approach in asynchronous circuit verification.

The novel verifying strategy is represented and analyzed. The benchmarks have proved its great capacity to enhance the performance of NuSMV. This also makes the PAiD tool in general and verification module in specific more reliable.

However, refer to Table 3 the progress of running time is faster than that of BDD nodes. Hence, in order to apply this verification method in large-scale circuits, future researches need to focus on controlling the number of BDD nodes. The novel verifying strategy could be a very good approach to deal with this challenge.

References

[1] A. Bink, and R. York, "ARM996HS, the first licensable, clockless 32-bit processor core," *IEEE Micro*, 2007.

[2] A. Cimatti, E. M. Clarke, F. Giunchiglia, M. Roveri, "NuSMV: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer 2.4 (2000)*,pp. 410-425, 2000.

[3] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani and A. Tacchella, "NuSMV 2: An OpenSource

Tool for Symbolic Model Checking," *In Proceeding of International Conference on Computer-Aided Verification (CAV 2002).* Copenhagen, Denmark, July 27-31, 2002.

[4] A. Davis, and S. M. Nowick, *An introduction to asynchronous circuit design.* The Encyclopedia of Computer Science and Technology 38, 1997.

[5] A. J. Martin, "Distributed mutual exclusion on a ring of processes," *Science of Computer Programming*, pp. 256-276, 1985.

[6] Alain J. Martin, "A synthesis method for delay-insensitive VLSI circuits," Invited paper, Formal Methods for VLSI Design, ed. J. Straunstrup, pp. 237-283, Elsevier, 1990.

[7] A.V. Dinh-Duc et al. - TAST CAD Tools, Proc. ACiDWG workshop, Germany, 2002. See also TIMA internal report ISRN:TIMA-RR-02/07/01FR, http://tima.imag.fr/cis.

[8] A. V. Dinh-Duc, L. Fesquet, and M. Renaudin, "A New Language-based Approach for Specification of Asynchronous Systems," *Proc. 3rd Int. Conf. in CS: Research, Innovation and Vision of the Future (RIVF)*, pp.224-229, 2005.

[9] A-V. Dinh-Duc, "PAiD-A Novel Framework for Design and Simulation of Asynchronous Circuits," *Journal of Science and Technology Development*, Vol. 14, No. K2, ISSN 1859-0128, pp. 37-45, 2011.

[10] C. A. R. Hoare, "Communicating sequential processes," *Communications of the ACM 21.8*, pp. 666-677, 1978.

[11] D. Cyrluk, S. Rajan, N. Shankar and M. K. Srivas, "Effective theorem proving for hardware verification," Theorem Provers in Circuit Design, Springer Berlin Heidelberg, 1995.

[12] D. L. Dill, and E. M. Clarke, "Automatic Verification of Asynchronous Circuits Using Temporal Logic," Michael Yoeli (Ed.), Formal Verification of Hardware Designs, IEEE CS, 1991, pp. 176-182.

[13] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, The MIT Press, 1999.

[14] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-Guided Abstraction Refinement," *CAV'00*, LNCS, vol. 1855, pp. 154-169, 2000.

[15] H. H. Tran, T. L. Ho, and A.V. Dinh-Duc, "PETRI-DFG - an intermediate representation of asynchronous circuits," *Proc. 10th Conf. on Science and Technology*, Vietnam, 2007.

[16] H. van Gageldonk, K. van Berkel, A. Peeter, D. Baumann, D. Gloor, and G. Stegmann, "An asynchronous low-power 80C51 microcontroller," *Proc. Int Symp. on Advanced Research in Async. Circuits and Systems*, pp. 96-107, 1998.

[17] J. L. Peterson, "Petri net theory and the modeling of systems." (1981).

[18] J. R. Burch, E. M. Clarke, K. L. McMillan, and D. L. Dill, "Sequential circuit verification using symbolic model checking," *Design Automation Conference, Proceedings., 27th ACM/IEEE. IEEE*, 1990.

[19] Joseph Rodrigues, K R Pai, Lucy J Gudino, "Synthesis of Linear Phase Sharp Transition FIR Digital Filter", *ECTI Transactions on Computer and Information Technology (ECTI-CIT)*, pp.96-99, Vol.1, No. 2, 2005.

[20] L. Nguyen-Thanh, K. P. Phan, and A.V. Dinh-Duc, "Behavior-Level Simulation of Asynchronous Circuits", *Proc. Int. Workshop on Advanced Computing and Applications (ACOMP)*, pp. 80–85, 2007.

[21] M. Renaudin, P. Vivet, and F. Robin, "ASPRO-216: a standard-cell QDI 16-bit RISC asynchronous microprocessor," Advanced Research in Asynchronous Circuits and Systems, 1998. Proceedings. 1998 Fourth International Symposium on. IEEE, 1998.

[22] McMillan, L. Kenneth, "Symbolic model checking," Springer US, 1993.

[23] O. Roig, J. Cortadella, and E. Pastor, "Verification of Asynchronous Circuits by BDD-based Model Checking of Petri Nets," *Proc. 16th Int. Conf. on App. and Theory of Petri Nets*, Italia, pp. 374-391, 1995.

[24] P. Camurati and P. Paol, "Formal verification of hardware correctness: Introduction and survey of current research," Computer 21.7, pp. 8-19, 1988.

[25] Q. C. Pham, T. N. Nguyen-Vu, A.V. Dinh-Duc, and H. A. Pham, "Placement and Routing Algorithms for Asynchronous Logic Circuits," *Proc. Int. Workshop on Advanced Computing and Applications (ACOMP)*, pp. 178-186, 2007.

[26] T. H. Bui, and A. Nymeyer, "Formal Verification Based on Guided Random Walks," *Proc. IFM'2009*, pp.72-87.

[27] T.H. Bui, A-V. Dinh-Duc, B.D. Ho, T.T. Nguyen, "Towards a verification approach for asynchronous circuits," *J. of Sci. & Tech.*, vol. 49, no. 4A, pp. 178-182, 2011.

[28] T.H. Bui, A-V. Dinh-Duc, T.T. Nguyen, "Encoding PN-DFG in NuSMV for Verifying Asynchronous Circuits," *SEATUC 2012*, Thailand.

[29] T. H. Dam-Thi, V. H. Bui, and A. V. Dinh-Duc, "Automatic Technology Mapping for Quasi Delay-Insensitive (QDI) Asynchronous Circuits," *Proc. Int. Workshop on Advanced Computing and Applications (ACOMP)*, 2007, pp. 23-32.

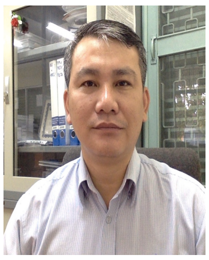
[30] T. T. Nguyen, K.-N. Le-Huu, T. H. Bui and A.-V. Dinh-Duc, "A New Approach and Tool in Verifying Asynchronous Circuits," *The International Conference on Advanced Technologies*

for Communications (ATC), Hanoi - Viêt Nam, 2012.

[31] W. W. Plummer, “Asynchronous arbiters,” Computers, IEEE Transactions on 100.1 (1972): 37-42.R.B. Standler, Protection of Electronic Circuits from Overvoltages, John Wiley and Sons, Inc., New York, 1989, ch. 5.

Tin Thien Nguyen has been a lecturer at Faculty of Computer Science and Engineering (CSE), University of Technology - Vietnam National University at Ho Chi Minh City (HCMUT) since 2012. He received the MSc. degree in Computer Science in 2014. His research interests include Digital Signal Processor, Embedded System, Asynchronous Circuit Design, and Image Processing.

Khoi-Nguyen LE-HUU has been with the Faculty of Computer Engineering, University of Information Technology - Vietnam National University Ho Chi Minh City as Lecturer and Research Coordinator since 2013. He is also a former assistant lecturer at Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT) where he received the BEng. (Honor Program) & MEng. degrees in Computer Engineering in 2012 and 2014, respectively. His research interests include Multi-core/Multi-Issue Digital Signal Processor, Energy Harvesting Wireless Sensor Networks, Asynchronous Circuit Design, and Super-Resolution based Image Processing. Currently, he also takes role as Coordinator of the UIT-VLSI Design group, Computer Engineering Faculty. He has been an IEEE member since 2013.



Thang H. Bui received the B.E. (1997) degree in computer engineering from Vietnam National University-HCMC University of Technology, M.E. (2001) degree in computer engineering from Asian Institute of Technology, Thailand, and PhD (2010) degree in computer science from New South Wales University, Australia. He is a Lecturer, Faculty of Computer Science and Engineering, Vietnam National University-HCMC University of Technology. His current interests include information systems, formal methods, especially model checking.

Anh-Vu Dinh-Duc is an associate professor at the University of Information Technology - Vietnam National University at Ho Chi Minh City where he has served as Vice-Rector, R&D and External Relations since 2012. He also leads the UIT-VLSI Design group at the Faculty of Computer Engineering. His research interests include Design Automation of Embedded Systems, Hardware/Software Verification, VLSI CAD, and Reconfigurable Architectures. Dr. Anh-Vu Dinh-Duc received the Master and Ph.D. degrees in Microelectronics from the Institut National Polytechnique de Grenoble (INPG), France, in 1998 and in 2003, respectively. Previously, he served as Vice Dean, Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Vietnam. Dr. Anh-Vu Dinh-Duc currently serves as a program/organizing committee member of several ACM and IEEE conferences. He is a member of the IEEE.