Scheduling of Transactions Based on Extended Scheduling Timed Petri Nets for SoC System-Level Test-Case Generation

JinShan Yu， Tun Li， Yang Guo，QingPing Tan

School of Computer Science
National University of Defense Technology
Changsha 410073，P.R. China
yujinshan@yeah.net

Abstract. The effective scheduling of transactions has a great potential for SoC functional verification. Petri nets have proven to be a promising technique for solving scheduling problem. This paper aims at presenting a Petri-net based approach to the scheduling of transactions generated by a test-case generator. Firstly, an extended scheduling timed Petri nets (ESTPN) model is given to support transaction scheduling. Secondly, the short term of 'scheduling of transactions problem' is formulated by means of an ESTPN which can accommodate various scheduling policies. Finally, transactions scheduling schemes and scheduling algorithm based on ESTPN are given and cases are studied.

1 Introduction

It is well known that verification today constitutes about 70% to 80% of the total design effort for SoC. Due to SoC's increasing complexities, SoC and multi-chip system developers can no longer rely on traditional simulation tools, testbenches, and methodologies, but must augment new verification methodology. The industry has raise the verification bar to the next level of abstraction -- transactions -- to ensure that designers and verification engineers have the tools and methodologies that give them a higher degree of confidence in their designs. Transaction-Based Verification (TBV) [1] enables the use of transactions at each phase of the verification cycle. A transaction is a single conceptual transfer of high-level data or control. It is defined by its begin time, end time, and all the relevant information associated with the transaction. For example, the relevant information (or attributes) of a Read transaction includes address and data. Transactions can be as simple as a memory Read/Write or as complex as the transfer of an entire structured data packet through a communication channel. The transaction level is the level at which the intended functionality of the design is specified and, therefore, the level at which the design can be verified in a most effective way. Raising the level of abstraction from signals to transactions facilitates the creation of tests, the debugging process, and the measurement of functional coverage. Therefore, there

is a growing interest for the TBV methodology, techniques, tools and coverage analysis method.

In this paper, we're interested in how to schedule transactions based on extended scheduling timed Petri net (ESTPN) to generate certain SoC system-level test cases. The advantage of ESTPN-based method is: Firstly, Petri-net based approach can graphically and concisely represent timing constraints of transactions in a single coherent formulation, which is very helpful for the designers to better understand and formulate the transactions scheduling problems. Secondly, it can support sequent, concurrent, weight-based, random, while scheduling schemes and these basic scheduling schemes can be composed to generate the more complex schemes. Thirdly, changes in the markings in an ESTPN model completely describe the evolution of different transactions scheduling policies, and use timed Petri nets analysis tools to check their schedulability.

The rest of the paper is organized as follows. Section 2 outlines the related works. Section 3 presents the formal definition for our extended scheduling timed Petri net. Section 4 presents how to map transactions scheduling problem into ESTPN. Section 5 lists the transaction scheduling schemes and scheduling algorithm. Cases studies are given in section 6. Finally section 7 provides some concluding remarks.

2 Related Works

For SoC, characterized by the integration of several interacting components ('cores'), cores act in parallel - as a result, logic bugs in systems and SoCs are often related to scenarios which are timing dependent. Some bugs in the design may only show up when test occur in a specific ordering, are separated by a specific time interval, or are executed concurrently. Flexible transaction scheduling is necessary to show up the design bugs and reduce test cost. In [8], R. Emek proposed a transaction scheduling language which has been implemented in the random test-case generator: X-Gen [9]. Harrod demonstrated the use of the AMBA-bus dedicated for test purpose [2].

Research has been going on in developing techniques for test scheduling, test access mechanism (TAM) and testability analysis. Recently, there are three major methods to solve the test scheduling problem: 1) graph-based techniques [11-14], 2) bin-packing methods [15-19], and 3) ILP/MILP approaches [20-25]. A survey of test scheduling methods is presented in [10]. Moreover, much research has been done simultaneously on Petri nets and scheduling problems during the last two decade. The general discuss for schedule problem based on Petri net is shown in [26]. Using Petri nets for the modeling of scheduling problems is not a new idea. However, according to our knowledge, there is no report in literatures on transaction scheduling based on Petri net. In this paper, we present a transaction scheduling method based on extended scheduling timed Petri nets (ESTPN's) for SoC system-level test case generation. ESTPN's are more like the graphical representation of the scheduling language described in [8]. However, ESTPN's are more systematic and visualizable because they utilize existing concepts and techniques of Petri nets along with additional representation of timing. In our model, time constraint is associated with place, transition

and places denote the scheduled transactions. Weight is associated with arc. Its formal definition will be given in section 3.

3 Extended Scheduling Timed Petri Net

The scheduling timed Petri net (Scheduling-TPNs) was introduced in [27]. For the works of schedulability analysis by Scheduling-TPNs, we refer the reader to Jeffreay J.P. Tsai [27, Li Huifang [28] and Roux [29]. We extend Scheduling-TPNs by adding weight to arc, and achieve extended timed constraint Petri net (ESTPN). Its formal definition is:

Definition1. A (Scheduling) *Extended Timed constraint Petri net* is a tuple , $ESTPN = (P, T, F, C, D, W, P, M, M_0)$, where:

1. $P = \{p_1, p_2, ..., p_m\}$ is a finite set of places, denoting the scheduled transactions.

2 $T = \{t_1, t_2, ..., t_n\}$ is a finite set of *transitions* such that $T \cap P = \varnothing$. For transaction scheduling, we include three kinds of transitions: basic transition, random transition, weight-based transition.

3. $F \subset (P \times T) \bigcup (T \times P)$ is a set of *arcs*(*flow relation*).

4. C is a set of integer pairs, $(T_{\min}(pt_j), T_{\max}(pt_j))$ where pt_j is either a place or a transition.

5. D is a set of transaction processing time $[F_{dur}(p_j)]$, where p_j is a place.

6. $W : F \to \{1,2,....\}$ is a weight function.

7. $P : p_j \to \{1,2,...\}$ is a priority function, assigning a priority to place p_j.

9. M is a set of marking with $m-vector$, $\{M_{(p_1)}, \cdots, M(p_j), \cdots, M(p_m)\}$, where $M_{(p_j)}$ denotes the numbers of token in place p_j.

10. $M_0 : P \to \{1,2,...\}$ is the initial marking.

Figure 1 shows an ESTPN sample. The state of an ESTPN is given by the distribution of tokens over the places and the corresponding timestamps. Firing a transition results in a new state. This way we can generate a sequence of states $s_0, s_1, \cdots, s_n$, corresponding to a transaction scheduling sequence, such that s_0 is the initial state and s_{i+1} is the state reachable from s_i by firing a transition.

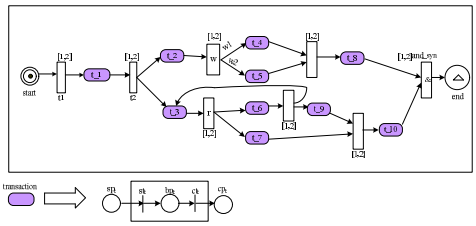

Fig. 1. An ESTPN Sample

Several basic definitions for ESTPN's schuedulability analysis are:

Definition2. A transition t_i with a time pair, it is said to be enabled if each of its input places has at least one token.

Definition3. A transition t_i, which is enabled at time $T0$, is said to be *firable* during the time period form $T0 + TC_{\max}(t_j)$ in which $TC_{\max}(t_j) \geq TC_{\min}(t_j)$.

A friable transition can fire but there is no guarantee that the firing will complete successfully because the firing of a transition takes a period of time $F_{dur}(t_j)$. All the tokens (denotes as $TK's$) used for enabling a transition t_j will be preserved during the $t_j's$ firing, and $TK's$ can be used to enable other transitions if t_j fails to complete their firing. If all the transitions enabled by $TK's$ fail to complete their firing, $TK's$ will be trapped in their corresponding places.

Definition4. A transition is said to be schedulable if it is friable and can complete its firing successfully, i.e., $(TC_{\max}(t_j) - TC_{\min}(t_j)) \geq FIRE_{dur}(t_j)$.

Definition5. A marking M_n is said to be reachable in $PN's$ modeling if there is a firing sequence, $\sigma = (M_0 t_0 M_1 \cdots M_j t_j \cdots t_n M_n)$.

The set of all possible markings reachable from M_0 is denoted by $R(M_0)$, and the set of all possible firing sequences from $L(M_0)$. With the considering of timing constraints, in ESTPN's modeling, a marking M_n is said to be reachable if and only if all transitions in σ are proved to be schedulable with respect to the timing constraints, i.e., $(TC_{\max}(t_j) - TC_{\min}(t_j)) \geq F_{dur}(t_j)$.

4 Mapping transaction scheduling problems into ESTPN

To show that ESTPN can be used to model and analyse transaction scheduling problems, we provide a translation from transaction scheduling problem to a 'suitable' extended timed constraint Petri net. A transaction t is defined by its begin time, end time, and all the relevant information. So we can identify three stages for transaction t: (1) t is waiting to be processed, (2) t is being processed and (3) t has been processed. Basically, Figure 2 shows how we model a transaction t in terms of an ESTPN. Transitions st_t and ct_t represent the beginning and termination of transaction t respectively. The place sp_t, bp_t and cp_t correspond to the stages just mentioned.

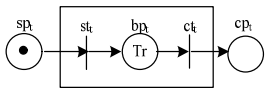

Fig. 2. Mapping Transaction Scheduling into ESTPN

Now transaction scheduling problem can be easily formulated by the following procedures:

Step1: For transaction $trans_i$, each scheduling activity is represented by two transitions st_t, ct_t and one place bp_t.

Step2: In terms of transaction partial order, all the transactions involved in a test case are linked, and modeled as an EPTPN model.

Step3: All the scheduled transactions are interconnected, and a complete EPTPN model for the scheduling of transactions is created.

5 Transaction Scheduling Schemes and Scheduling Algorithm

5.1 Transaction Scheduling Schemes

Base transaction scheduling schemes in ESTPN include sequent, concurrent, while, weight-based, random. These scheduling schemes can be composed to generate more complex scheduling sequence, named composed scheduling scheme.

(1) Sequent scheduling scheme

In sequent scheduling scheme, transaction t' is scheduled after transaction t has been accomplished, i.e. bus write transaction preceding bus read transaction. It is modeled by adding extra places. Figure 3 shows the situation where transaction t precedes transaction t', i.e. the execution of transaction t has to be completed before the execution of transaction t'. Place $pre<t,t'>$ prevents $st_{t'}$ form firing until ct_t fires. Notes that places are used to model the stages of a transaction.

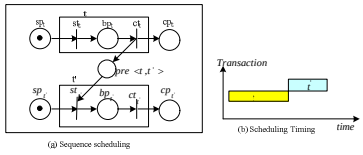

Fig. 3. Sequent Scheduling Scheme and Scheduling Timing

(2) Concurrent scheduling scheme

Concurrent scheduling scheme is modeled as shown in Figure 4. Transaction t and t' are scheduled concurrently.

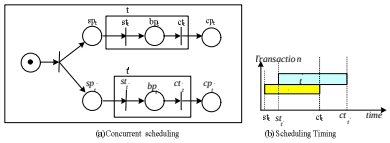

Fig. 4. Concurrent Scheduling Scheme and Scheduling Timing

(3) While scheduling scheme

In while scheduling scheme, transaction can be scheduled many times. While scheduling scheme can be used to generate burst mode transactions. Figure 5 shows this case.

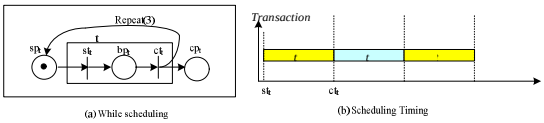

Fig. 5. While Scheduling Scheme and Scheduling Timing

(4) Weight-based scheduling scheme

In weight-based scheduling scheme, scheduler first evaluates the weights on output arcs from weight transition, and schedules the transaction with greatest weight. Figure 6 shows this case.

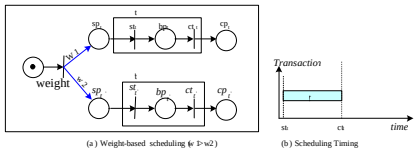

Fig. 6. Weight-based Scheduling Scheme and Scheduling Timing

(5) Random scheduling scheme

In random scheduling scheme, random transition is used to randomly schedule one transaction. Figure 7 shows this case.

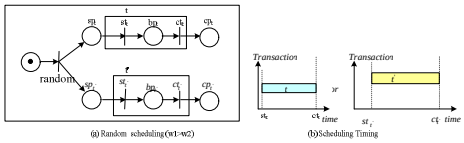

Fig. 7. Random Scheduling Scheme and Scheduling timing

(6) Composed scheduling scheme

Above five basic scheduling schemes can be composed to generate more complex scheduling scheme, named composed scheduling scheme.

5.2 Scheduling Algorithm

Based on extended scheduling timed Petri net model, we present the following transactions scheduling algorithm:

Step1 Model transaction scheduling problem and the imposed time constraints using ESTPN:

Step1.1 Model the transaction scheduling problem as a Petri net based model.

Step1.2 Mapping the time constraints for the transactions and conditions to the time pair $[T_{\min}(pt_j), T_{\max}(pt_j)]$ and the time duration $[F_{dur}(p_j)]$ which are associated with the places or transitions in the Petri nets based model. Then prepare the ESTPN system model.

Step1.3 Determine the initial marking M_0 and its arrival time $T_{Arr}(M_0)$.

Step2 Determine all friable transitions in marking M_k (if k=0, then $M_k = M_0$) and find the number J of all friable transitions. If the friable transition is the start transition of one transaction and timing constraints is met, scheduling transaction. If there are several such transitions, concurrently schedule these transactions.

Step3 Determine the next mark $\{M_{next(p_1)}, \cdots, M_{next}(p_j), \cdots, M_{next}(p_m)\}$ according to the Petri net transition rule, and go to step2.

Step 4 Stop.

6 Case studies

We have implemented this scheduling methodology in our SoC system-level functional verification prototype system: SL-Gen. SL-Gen provides interfaces to time Petri nets analysis tools (Remeo [30], TINA [31]) to simulate and analysis transaction scheduling Petri nets model.

We illustrate our approach on a SoC design based on AMBA 2.0 Specification. We create the following transactions for AMBA in SL-Gen: initialize_amba, HCLK_generator, abort_HCLK_generator, reset, abort_reset, AHB.initialize_ahb_master, AHB.write, AHB.abort_write, AHB.read, AHB.idle, and AHB.busy. Figure 8 shows one AMBA transaction scheduling scenario modeled by ESTPN.

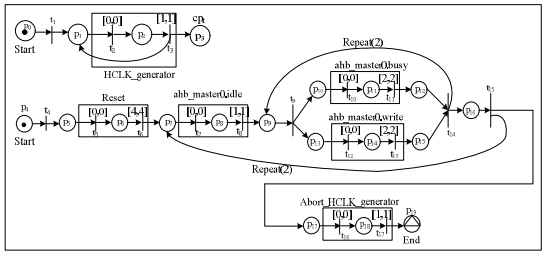

Fig. 8. AMBA Transaction Scheduling Scenario

SL-Gen automatically generates the following system-level testbench:

```
Initial
   integer burstLength;
    $timeformat(-9,3,"ns",7);
// $Initialize
  tb_initialize_amba;
// Apply transaction scheduling
  HCLK_generator_looping;
reset;
   repeat (2) // number of bursts to perform
    begin
        ahb_master0.idle;
```

```
        writedata.randomize;
        repeat (2)
            begin
              ahb_master0.   busy   (writedata.addr,   1,
writedata.size, writedata.burst);
              ahb_master0.    write    (writedata.trans,
writedata.addr,
 writedata.data, writedata.size, writedata.burst);
      writedata.addr  =  writedata.addr  +  (1  <<  write-
data.size);
      writedata.randomize_data;
             end
    end
Abort_HCLK_generator;
 end
```

In SL-Gen, we can get the following simulation timing, shown in Figure 9.

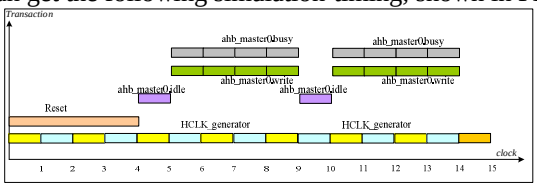

Fig. 9. AMBA Transaction Scheduling Timing

7 Conclusions

An extended scheduling timed Petri net (ESTPN) formulation for the scheduling of transactions has been proposed. Changes in the markings in an ESTPN model completely describe the evolution of different transactions scheduling problem. The great benefit of the Petri-net based approach is graphically and concisely represent timing constraints of transactions in a single coherent formulation, which is very helpful for the designers to better understand and formulate the transactions scheduling problems. Moreover, it is understood from this research that the Petri-net based approach has a great potential for solving a variety of complicated scheduling problems in transaction based test case generation. In this regard, further research is also being undertaken to accommodate complicated constraints such as resource, power, TAM and to implement the integrated design of the supervisory control and scheduling of transactions for SoC functional verification.

8 ACKNOWLEDGMENTS

This work was supported by National Natural Science Foundation of China under Grant No. 60403048 and No. 60573173.

References

1. Cadence Berkeley Labs: The Transaction-Based Verification Methodology. Technical Report # CDNL-TR-2000-0825, August 2000.
2. http://www.synopsys.com/
3. http://www.verisity.com/html/specmanelite.html
4. http://www.chronology.com/
5. http://www.testbuilder.net
6. http://www.systemc.org
7. Rohit Jindal and Kshitiz Jain: Verification of Transaction-Level SystemC models using RTL Testbenches. In Proceedings of the First ACM and IEEE International Conference on Formal Methods and Models for Co-Design (2003)199-204
8. R. Emek and Y. Naveh: Scheduling of Transactions for System-Level Test-Case Generation. In Proceedings of the Eighth IEEE International High-Level Design Validation and Test Workshop (2003) 149-154
9. R.Emek, I.Jaeger, Y.Naveh, G.Bergman, G.Aloni, Y.Katz, M.Farkash, I.Dozoretz and A.Goldin: X-Gen: A Random Test-Case Generator for Systems and SoCs. In Proceedings of the Seventh IEEE International High-Level Design Validation and Test Workshop (2002) 145–150
10. V. Iyengar, K. Chakrabarty, and E.J. Marinissen: Recent Advances in Test Planning for Modular Testing of Core-Based SOCs. In Proceedings of the 11th Asian Test Symp (2002) 320-325
11. R. Chou, K. Saluja, and V. Agrawal: Scheduling Tests for VLSI Systems under Power Constraints. IEEE Trans. VLSI, vol. 5, no. 2 (1997)175-185
12. P. Rosinger, B. Al-Hashimi, and N. Nicolici: Power Profile Manipulation: A New Approach for Reducing Test Application Time under Power Constraints. IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 21, no. 10 (2002)1217-1225
13. D. Zhao and S. Upadhyaya: Adaptive Test Scheduling in SoC's by Dynamic Partitioning. In Proceedings of the 17th IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems (2002) 334-342
14. D. Zhao and S. Upadhyaya: Power Constrained Test Scheduling with Dynamically Varied TAM. In Proceedings of the 21st VLSI Test Symp (2003) 273-278
15. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, and R.E. Tarjan: Performance Bounds for Level-Oriented Two-Dimensional Packing Algorithm. SIAM J. Computing, vol. 9 (1980) 809-826
16. Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y.Zaidan, and S.M. Reddy: Resource Allocation and Test Scheduling for Concurrent Test of Core-Based SoC Design. In Proceedings of IEEE Asian Test Symposium (ATS) (2001) 265-270
17. Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, O. Samman, Y.Zaidan, and S.M. Reddy: On Concurrent Test of Core-Based SoC Design. J. Electronic Testing: Theory and Applications, vol. 18(2002) 401–414
18. V. Iyengar, K. Chakrabarty, and E.J. Marinissen: Wrapper/TAM Co-Optimization, Constraint-Driven Test Scheduling, and Tester Data Volume Reduction for SOCs. Proc. 39th Design Automation Conf (2002) 685-690

19. Y. Huang, S.M. Reddy, W.-T. Cheng, P. Reuter, N. Mukherjee,C.-C. Tsai, O. Samman, and Y. Zaidan: Optimal Core Wrapper Width Selection and SOC Test Scheduling Based on 3D Bin Packing Algorithm. In Proceedings of ITC 2002 (2002)74-82
20. Y. Huang, W.-T. Cheng, C.-C. Tsai, N. Mukherjee, and S.M.Reddy: Static Pin Mapping and SOC Test Scheduling for Cores with Multiple Test Sets. In Proceedings of the Fourth International Symposium on Quality Electronic Design (2003) 99- 104
21. M. Nourani and C. Papachristou: An ILP Formulation to Optimize Test Access Mechanism in SoC Testing. In Proceedings of ITC 2000 (2000)902-910
22. K. Chakrabarty: Test Scheduling for Core-Based Systems Using Mixed Integer Linear Programming. IEEE Trans. Computer-Aided Design, vol. 19(2000)1163-1174
23. V. Iyengar and K. Chakrabarty: Precedence-Based, Preemptive, and Power-Constrained Test Scheduling for System-on-a-Chip. In Proceedings of the 19th IEEE VLSI Test Symposium (2001) 368-374
24. M. Nourani and J. Chin: Power-Time Trade-Off in Test Scheduling for SoCs. In Proceedings of IEEE International Conference on Computer Design(ICCD '03)(2003) 548-553
25. James Chin and Mehrdad. Nourani: FITS: An Integrated ILP-Based Test Scheduling Environment. IEEE Trans. on computer, VOL. 54, NO. 12(2005) 1598- 1613
26. W.M.P. van der Aalst: Petri net based scheduling. Computing Science Reports 95/23, Eindhoven University of Technology, Eindhoven (1995)
27. Jeffrey J.P.Tsai, Steve Jennhwa Yang and Yao-Hsiung Chang: Timing Constraints Petri Net and Their Application to Schedulability Analysis for Real-Time System Specification. IEEE Transaction on Software Engineering, VOL. 21. NO. 1(1995)
28. Li Huifang and FAN Yushun: Schedulability analysis method for Timing Constraint Petri Nets. Tsinghua Science and Technology, Vol.7, No.6(2002) 596-601
29. Roux, O.H., Deplanche, A.M: A t-time Petri net extension for real time-task scheduling modeling. European Journal of Automation (JESA) 36 (2002) 973-987
30. 20. Guillaume Gardey, Didier Lime, Morgan Magnin, and Olivier (H.) Roux. Romeo: A Tool for Analyzing Time Petri Nets. CAV 2005, LNCS 3576 (2005) 418-423
31. B. Berthomieu, P-O. Ribet, and F. Vernadat. The tool TINA: Construction of abstract state. spaces for Petri nets and time Petri nets. International Journal of Production Research, V42, N14(2004)