Годишник на секция "Информатика"
Съюз на учените в България
Том 4, 2011, 120-126

Annual of "Informatics" Section
Union of Scientists in Bulgaria
Volume 4, 2011, 120-126

GENERALIZED NETS AS A UNIVERSAL METHOD FOR TIMING CLOSURE OF DIGITAL INTEGRATED CIRCUITS

Marin Marinov

European Polytechnical University
23 "St. St. Kiril and Metodiy" Str., Pernik 2300, Bulgaria
email: marin.marinov@epu.bg

Abstract: A unified approach to timing closure of digital integrated circuits, based on the generalized nets modelling, is proposed. The advantage is that by the help of a universal model of the digital structure on every level of design hierarchy both static and dynamic timing analysis can be done. A backward modification of design under control of the net model is able to satisfy the time constrains.

Keywords: Timing closure, Static timing, Dynamic timing, Turing machine, Generalized nets.

1. Introduction

Timing closure is the process by which a digital integrated circuits design is modified to meet its timing requirements. Timing closure includes both static as well as dynamic timing analysis. Dynamic timing analysis (DTA) verifies functionality of the design by applying input vectors and checking for correct output vectors whereas static timing analysis (STA) checks static delay requirements of the circuit without any input or output vectors [1, 2].

In STA, static delays such as gate delay and net delays are considered in each path and these delays are compared against their required maximum and minimum values. Circuit to be analyzed is broken into different timing paths constituting of gates, flip-flops and their interconnections. Each timing path has to process the data within a clock period, which is determined by the maximum frequency of operation.

On the other hand, DTA is a dynamic simulation, which determines the full functionality of the circuit for a given set of input stimulus vectors. STA and DTA do not mutually exclude, moreover, STA is not applicable to asynchronous digital structures. Each of these analyses needs specific methods for implementation. On the other hand, both STA and DTA should be applied to each level of the design abstraction, i.e. behaviour, RTL, circuit, gate, and device (Fig. 1).

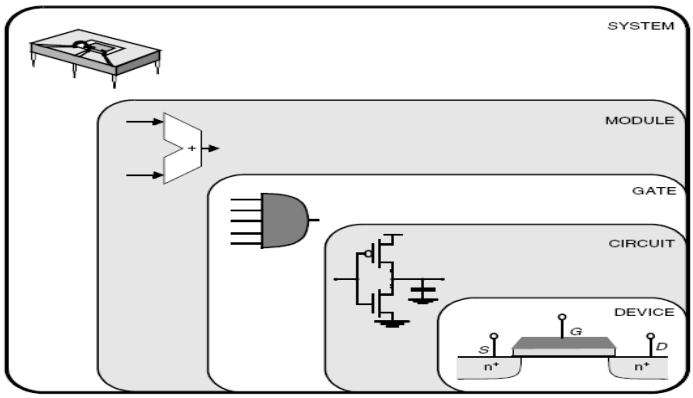


Figure 1. Design hierarchy

The survey of EDA timing tool shows that there are a variety of them, covering certain part of design timing closure process [3, 4]. We propose generalized nets (GNs) as a universal formalism for implementation of complete timing closure, because GNs are aimed at describing discrete parallel processes. Also GN description allows to combine all aspects of timing closure analysis, using one and the same formal approach.

2. Short notes on generalized nets

GNs [5] are extensions of the Petri nets and other modifications of theirs. They are tools intended for detailed modelling of parallel processes.

A GN is a collection of *transitions*, defined in turn as a set of *places* (Fig. 2). For each transition there is an index matrix with elements – predicates. Some GN-places contain *tokens* – dynamic elements entering the net with initial characteristics and getting next ones during their movement in the net. Tokens proceed from the input to the output places of the transitions if the predicate corresponding to these places is evaluated as *"true"*. Every token has its own identifier and collects its own history that may influence the development of the whole process modelled by the GNs.

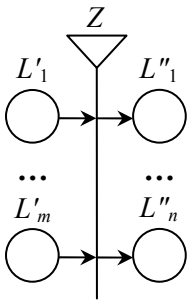


Figure 2. GN-transition

Two time-moments are specified for the generalized nets: startup and termination of functioning, respectively.

Formally, every transition in the reduced GN is described by a triplet:

$$Z = \langle L', L'', r \rangle,$$

where

(a) L' and L'' are finite, non-empty sets of places (the transition's input and output places, respectively);

(b) r is the transition's condition determining which tokens will pass (or transfer) from the transition's inputs to its outputs; it has the form of an index matrix (IM):

$$r = \begin{array}{c|c} & l''_1 \quad \ldots \quad l''_j \quad \ldots \quad l''_n \\ \hline l'_1 & \\ \ldots & r_{i,j} \\ l'_i & (r_{i,j} \ - \ predicate) \\ \ldots & (1 \le i \le m, \ 1 \le j \le n) \\ l'_m & \end{array}$$

(c) $r_{i,j}$ is the predicate that corresponds to the i-th input and j-th output place. When its truth-value is *"true"*, a token from the i-th input place transfers to the j-th output place; otherwise, this is not possible;

The ordered four-tuple

$$E = \langle Z, K, X, \Phi \rangle$$

is called a *generalized net* if:

(a) Z is a set of transitions;

(b) K is the set of the GN's tokens;

(c) X is the set of all initial characteristics, which the tokens could obtain on entering the net;

(d) Φ is the characteristic function that assigns new characteristics to every token when it makes the transfer from an input to an output place of a given transition.

A lot of operations (e.g., union, intersection and others), relations (e.g., inclusion, coincidence and others) and operators are defined over generalized nets. Operators change the GN-forms, the strategies of token transfer and other parameters. Operators are of six types: global, local, hierarchical, reducing, extending and dynamic operators.

The formal reason of using GNs for the purpose of digital circuits analysis is the theorem of equivalency of GN and Turing machine [6]. On Fig. 3, a GN description of such a machine is shown.

Briefly the operation of the model is as follows:

The token with characteristic *"Write on the tape"* enters place L_1. If what is then written corresponds to the final characteristic, the token goes to L_2, else – to place L_3, where it obtains the characteristic *"The machine head should be shifted to left or right"*. The token goes to place L_4, where it obtains a new characteristic *"The symbol, which should be written in the cell under the head"*. In place L_5, the token obtains as a characteristic the newly written symbol and returns to transition Z_1.

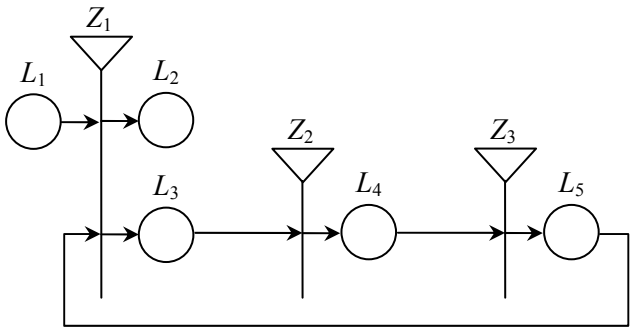


Figure 3. GN model of the Turing machine

3. Features of GNs to describe the process dynamics

As already said, the formalism of GNs is suitable for modelling of concurrent, parallel processes, where many temporal characteristics of the processes could be precisely studied. For that purpose, the following GN parameters should be fixed: initial characteristics of the tokens; the priorities between tokens; the enter time of each token; all possible characteristics of tokens. Using three global temporal components, the GN models the processes in the frame of an absolute time scale with arbitrary steps.

4. GN for timing analysis

On the top level of design hierarchy, GNs can successfully describe computing architectures, as shown in [7]. The graphic presentation of the GN model is given on Fig. 4.

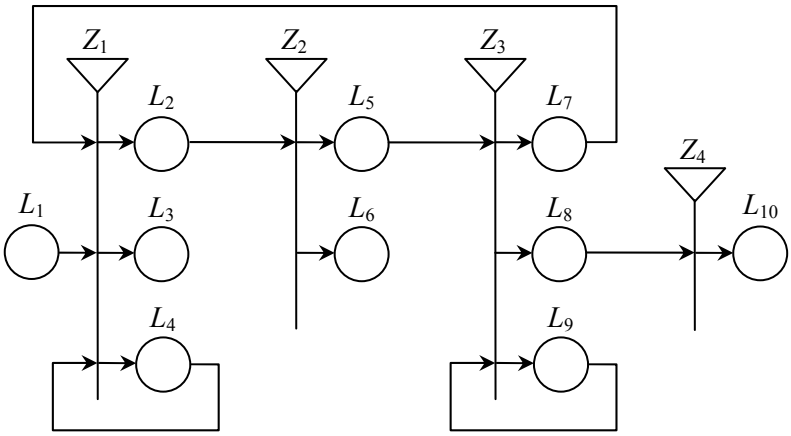
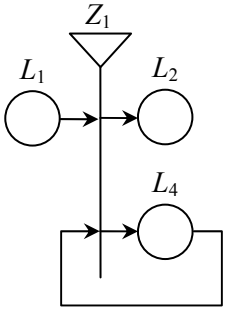


Figure 4. The GN model of 4-phase pipeline processor architecture

The four pipeline's phases (IF, ID, EX and WR) are considered as transitions in the GN model, respectively Z_1 to Z_4. The tokens in the GN model represent instructions, executed in the pipeline processor. In this model, by choosing proper GN's transition functions, such as $\Theta_1(t)$, being the time moment of transition activation, and $\Theta_2(t)$, being the duration of the transition's active state, the time constraints of the instructions execution could be satisfied.

On the circuit level, we illustrate the GN modeling by describing basic Moore and Mealy finite state automata (FSA) (see Fig. 5) with their transition matrices, respectively, where L_1 is an input place, L_2 is an output place and L_4 is a state place.



	L_2	L_4
L_1	*false*	*true*
L_4	*true*	*true*

a)

	L_2	L_4
L_1	*true*	*true*
L_4	*true*	*true*

b)

Figure 5. GN of Moore/Mealy FSA and their transition matrices a) and b), respectively

And on the gate level, GN models of logic gates will be quite simple and regular. On Fig. 6, a GN of a two-variables logic gate is presented. The tokens from places L_1 and L_3 (input variables) are passing through transition Z_1 and in output place L_2 the resulting token will appear with a characteristic equal to the logic function of the gate. The GN could be easily extended to describe multiple variables logic gate, implementing an arbitrarily complex logic function.

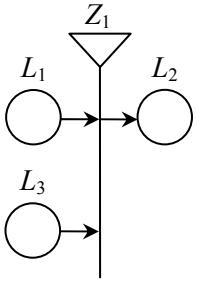


Figure 6. GN of two-inputs logic gate

In the same way as in the case of GN modelling of top level design by changing the temporal transition functions $\Theta(t)$ of automata and logic gates, all GN's time requirements and restrictions could be met.

As it was noted, STA is based on the delay calculation of logic paths, which are extracted using following rules:

- A path starts at a source flip-flop (or at a primary input) and terminates at the destination flip-flops.
- A path does not go through flip-flops unless it is a transparent latch.
- The path terminates when it encounters a clocked device.

Thus, for STA only GN models of combinational logic will be considered.

5. GN for dynamic timing analysis

DTA, instead, requires that the full functionality is tested with the aid of a stimulus. For this purpose, we propose the following approach, based also on the GN models of the digital structure (Fig. 7).

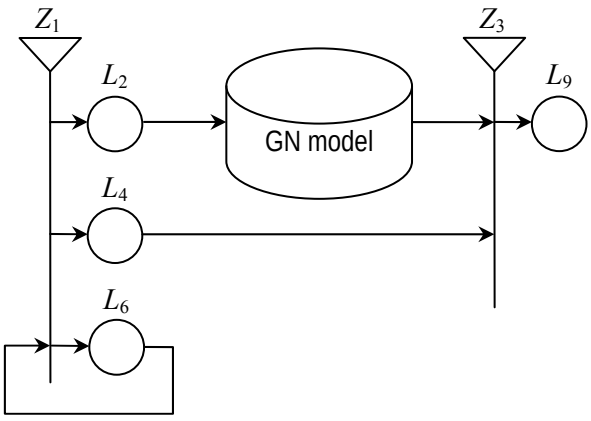


Figure 7. GN for the dynamic timing analysis

To the GN model of the structure we add two transitions:

- Z_1 – an input transition to split the input stimuli, generated by the place L_6 (the base of stimulus) in two streams: L_2, dedicated to activate the model of the structure, and L_4, where the characteristics of the tokens correspond to the hypothesis for the reaction of the structure (GN model) to the stimulus, applied via L_2; and
- Z_2 – an output transition, which collects all responses of the structure and hypotheses.

Finally, in place L_9 of GN model, the tokens with characteristics, showing the degree of consistency between the structure response parameters and corresponding hypotheses, will appear.

Thus, the result of the dynamic timing analysis will be directly interpreted. Moreover, thanks to the inherent feature of the GN to control, the backward modification of certain parameters could be also done in case of improper dynamic behavior of the structure (design).

Conclusion

Generalized nets as a flexible and powerful means for modelling of discrete processes are able to provide a unified environment for extensive and precise timing and functional analysis of the digital designs for integrated circuits. This approach could be very useful also because in the presence of widely applied reuse of different building blocks (soft and hard IP) the process of the integrated circuit timing closure will be shortened due to the ability of GN models to be decomposed and extended by using hierarchical operators. Finally, the control function of GN models is an extra advantage for design modification in an adaptive way.

References

[1] Mano, M., M. Ciletti. *Digital design*. Fourth edition. Prentice Hall, 2007.

[2] Rabaey, J. M., A. P. Chandrakasan, B. Nikolic. *Digital Integrated Circuits: A Design Perspective*, Second edition, Prentice Hall Electronics and VLSI Series, Upper Saddle River, NJ: Pearson Education, 2003.

[3] ConMan-Automatic Constraints Compiler, `http://www.excellicon.com/product`

[4] PrimeTime – Timing Sign-Off Solution and Environment, `http://www.synopsys.com/Tools/Implementation/SignOff/Pages/PrimeTime.aspx`

[5] Atanassov, K. *Generalized Nets*. World Scientific, Singapore, 1991.

[6] Atanassov, K. *On Generalized Nets Theory*. "Prof. M. Drinov" Academic Publ. House, Sofia, 2007.

[7] Marinov, M., K. Atanassov. Generalized nets model of pipeline processor architecture. *Proc. of the Int. Conf. on Information Technologies (InfoTech-2010)*, Sept. 17-19, 2010, Bulgaria, Vol. 1, 3–4.