Conference Presentation

Event-driven simulation of digital circuits using modified Petri nets algorithm

Lapin, A., Bulakh, D. and Vagapov, Y

This is a paper presented at the 7th IEEE Int. Conference on Internet Technologies and Applications ITA-17, Wrexham, UK, 12-15 September 2017

Event-driven Simulation of Digital Circuits using Modified Petri Nets Algorithm

Alexander Lapin, Dmitry Bulakh
Department of Integrated Circuits Design
National Research University of Electronic Technology
Zelenograd, Moscow, 124498, Russia

Yuriy Vagapov
School of Applied Science, Computing and Engineering
Glyndwr University
Plas Coch, Mold Road, Wrexham, LL11 2AW, UK

***Abstract*—This paper presents a modified Petri nets simulation algorithm applied as an engine for a logic simulator in digital integrated circuit design. The simulator uses an event-driven algorithm and eliminates the delta delay which occurs in the majority of modern simulation algorithms. The algorithm has been tested for the logic simulation of combinational digital circuits and demonstrated more accurate simulation results. This has been achieved due to solving the issue of the priority choice problem when two or more events are occurring simultaneously.**

Keywords—design automation; integrated circuit design; event-driven simulation; delta delay; gate level simulation; behavioural simulation; Petri net

I. INTRODUCTION

Digital circuit design is the most important stage of integrated circuit (IC) development where circuit elements are represented as logic gates to perform Boolean algebra operations. Logic gate operations are usually described as a combination of logic functions (i.e. AND, NOT, OR and others) performing under algorithms based on the syntax of high level programming languages. The syntax of these languages includes assignment statements, multiple-choice operators, logical conditions, loops and, for some languages, even the function or procedure calls and object oriented programming.

There are several types of algorithms applied for digital circuit simulation. The first type of algorithms is called "cycle simulation" where the time axis is divided into a sequence of small fixed time steps and the simulation is performed at each time step during the simulation time. All of the logic gates of the circuit are evaluated at each time step. However, this algorithm has several important disadvantages:

1. The algorithm can only be implemented at the gate level for the described circuits;
2. The algorithm must arrange all of the elements in the simulated circuit according to the signals passing through the logic gates;
3. The algorithm is inefficient when used to simulate digital circuits containing large numbers of gates;
4. The algorithm is extremely inefficient for the simulation of circuits with delays.

It is obvious that this type of algorithm cannot be applied for the simulation of circuits having a large scale integration and a significant number of feedbacks. Therefore, the simulation of modern digital circuits, comprising of millions of triggers and flip-flops, becomes impossible under the "cycle simulation" approach.

The second type of algorithm is called "event-driven simulation". Nowadays this simulation algorithm is extremely popular and widely used for IC development and design.

During event-driven simulation the time axis is not divided onto a fixed sequence of time steps but dynamically updated with events indicating which gate has to be evaluated. If a circuit signal value is changed (for example, as a result of input switching or gate switching) a new event is added to the event list. This event contains information about operation time and the gate to be evaluated.

Thus, this algorithm solves the majority problems that occur during the execution of the "cycle simulation" algorithm:

1. The algorithm can be applied at any level of abstraction to the described circuits;
2. The algorithm does not need to define all of the elements in the simulated circuit;
3. The algorithm is efficient for simulation of the digital circuits containing a large amount of gates. The simulation under this algorithm is not conducted for all gates; it is only applicable for the gates added to the list of events;
4. The algorithm is efficient when used to simulate circuits with delays. This is achieved due to the fact that the time axis is not divided into fixed time steps and the events appear dynamically.

Although the event-driven simulation algorithm is widely used in industry it has a disadvantage related to accuracy.

II. PROBLEM DEFINITION

At the simulation stage of the digital circuits with multi-input gates, where the inputs of the gates are switched at the same time, the logic simulator must select which gate has to be processed first because the algorithm engine does not support "true parallel" operation.

978-1-5090-4815-1/17/$31.00 ©2017 IEEE

In order to resolve this ambiguity, modern digital circuit simulators implement the concept of "delta delay". This concept means that all of the signal switches that occur at the same moment in time are shifted along the time axis with a very small delay called "delta delay". The time delay is so small that for the user, observation of all of the signal switches occurs in parallel at the same moment in time. Whereas the algorithm actually evaluates them consequently. However, this approach can generate different simulation results for the same signal switching sequences.

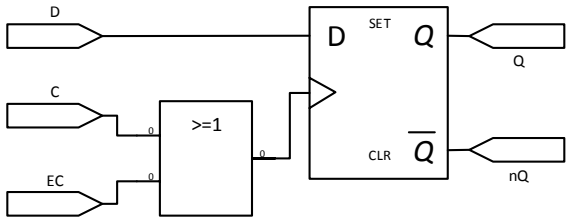

Fig. 1. Example circuit.

Depending on the method of description the algorithm can perform in different ways choosing a different order of switching operations. This situation is illustrated in an example of the clock-gated D Flip-flop (DFF) shown in Fig. 1 where the simulation results depend on the order of the signal switching in the netlist. Fig. 2 shows the result of the simulation of a given circuit using an Icarus Verilog simulator, the most efficient open source digital circuit simulation program.

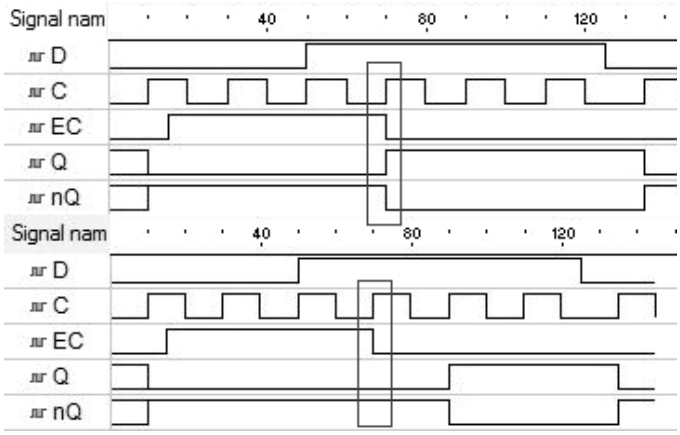

Fig. 2. Simulation results.

In this example, there are two signals switching at the same time – C (clock signal) and EC (enable clock signal). The simultaneous switching disables the clock signal passing to DFF when using the power save mode. This method known as "clock gating" is usually applied to minimise power consumption of a circuit operating in power save mode.

Obviously, the simulation program cannot evaluate the simultaneous switching of several gates so therefore it implements a "delta delay" algorithm to perform the sequential switching of inputs C and EC.

The upper plot in Fig. 2 shows the results when the simulation algorithm chooses the input EC to be switched first. In this case the input C is switched after EC and the simulation program considers that if EC is already equal to 0 then switching C to 1 will lead to the switching of the OR2 gate and the DFF output is changed accordingly.

The lower plot in Fig. 2 demonstrates the opposite condition, where the simulation algorithm takes switching of input C first. Under this condition, the input EC is still equal to 1, so nothing happens at the output of the OR2 gate and the trigger is not switched.

This paper proposes a new digital circuit simulation algorithm based on the modification of the Petri net simulation approach aimed to avoid the "delta delay" and improve the accuracy of the digital circuit simulation.

III. Proposed Approach

Petri nets are a class of the basic model of parallel and distributed systems designed by Carl Adam Petri. An example of a Petri net is shown on Fig. 3.

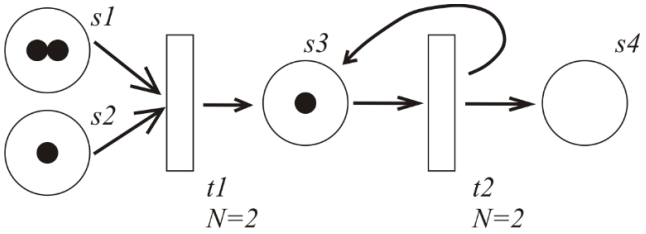

Fig. 3. An elementary Petri net.

Algorithms based on Petri nets can simulate parallel processes with sequential instructions. Petri nets contain the places (s_1, s_2, s_3, s_4) and the transitions (t_1, t_2) which can be connected by the directed arcs. The places in the net may contain tokens in (s_1, s_2, s_3) and not in (s_4) as shown in Fig. 3. These tokens can be moved to other places using "firing" actions. Each step of the simulation in the Petri net consists of three stages: t_- (Fig. 4a), t_0 (Fig. 4b) and t_+ (Fig. 4c).

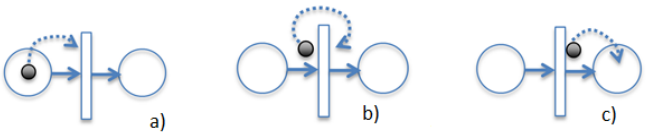

Fig. 4. Functioning of Petri net.

Petri nets are used as an effective tool for parallel process simulation in hardware design. There are two main application approaches of Petri nets in the digital IC design:

1. Description of a circuit behaviour [1], [2] where the simulation of the circuit is conducted using standard algorithms;
2. The basis of gates representation [3], [4] where digital elements are described in terms of safe Petri nets and standard algorithms are used for simulation.

The method suggested in this paper is based on the modified Petri nets simulation algorithm being used as a basis of the logic simulation engine. Instead of the circuit description using a set of traditional Petri net places and transitions it is proposed that the modified transitions and places are implemented in accordance with the logic functions described

in the netlist. This approach reduces the number of Petri net elements (states and transitions) used for the simulation compared to previous approaches [2]-[4].

The proposed modifications to the Petri nets are as follows:

1. Each place of Petri net corresponds to one node of the digital circuit. If the state has a dot it means that this net of a circuit has logic 1 whereas if the state has no dot – the logic value in this net equals to logic 0. Using coloured Petri nets the multi-valued logic (0, 1, U, Z etc.) could be implemented.
2. A net transition implements a logic gate of the digital circuit. At the gate level of simulation algorithm there is a set of predefined transitions corresponding to the logic gates NOT, AND, OR, NAND, NOR and XOR.
3. Unlike the basic Petri net algorithm the transition will fire in any case; this action is not linked to the presence of a dot in its input. In the proposed modification the transition firing depends on type of transition only and could occur when there is no dot in the input. For example, if the transition implements a logic gate NOT it will fire each time when there is no dots in its input and will not fire if there is a dot. This performance is corresponding to the logic gate functioning rules.
4. If the transition fires, this action will not remove the dots from its inputs.

IV. Example Circuit

An example shown in Fig. 5 and Fig. 6 demonstrates how a one bit full adder (Fig. 5) is implemented in the proposed modification of Petri nets (Fig. 6).

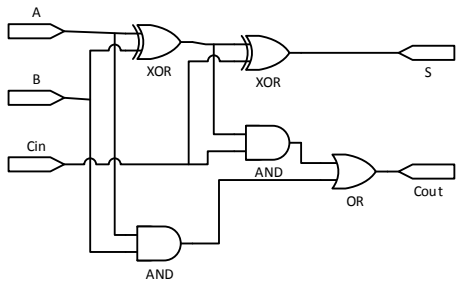

Fig.5. Full adder schematic

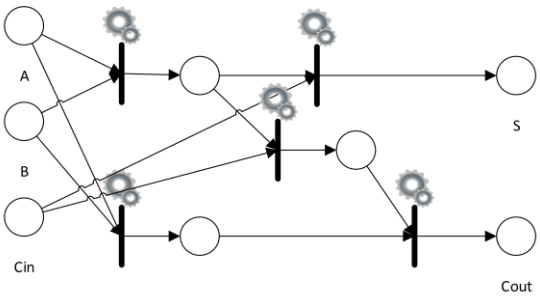

Fig.6. Full adder schematic translated to Petri nets

Simulation under the proposed algorithm takes 1.5–2 times longer than the similar process without Petri nets. For example, if the simulation queue is {G1, G2, G3} (where G_ is a logic gate) then using Petri nets, it is changed to the following: {G1(t_-), G2(t_-), G3(t_-), G1(t_0) G2(t_0) G3(t_0), G1(t_+) G2(t_+), G3(t_+)}. Despite this, the performance of the proposed simulation algorithm equals that of the Icarus Verilog. However, the proposed algorithm increases the simulation queue by three times when compared to a standard simulation algorithm.

V. Experimental Results

This algorithm was implemented using C++ [5] and was compared to the Icarus Verilog simulation program.

For the circuit in Fig. 1, different input vectors were generated. When these were compared to the results obtained from the Icarus Verilog simulation program (Fig. 2), the simulation under the proposed algorithm demonstrated that any sequence of the input vectors results in the unique and correct waveforms as shown in Fig. 7.

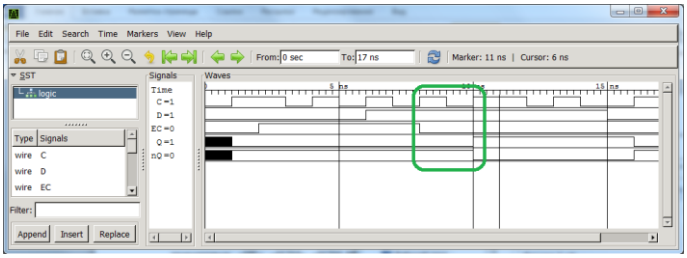

Fig.7. Simulation results.

The developed simulation program was tested using ISCAS'85 benchmarks. The results of the simulation are fully consistent with the results obtained from Icarus Verilog simulation.

VI. Conclusion

It has been shown that the simulation result for the proposed algorithm is correct for any type of netlist description. In further work, several approaches will be implemented for the simulation of behavioural descriptions of digital circuits using a proposed algorithm. Behavioural descriptions can also be represented as the set of Petri net instances (places and transitions) with specific functioning.

References

[1] A. Yakovlev, L. Gomes, and L. Lavagno, Eds., *Hardware Design and Petri Nets*, New York: Springer, 2011.

[2] M. Uzam, M. Avci, and M.K. Yalcin, "Digital hardware implementation of Petri net based specifications: Direct translation from safe automation petri nets to circuit elements," in *Proc. Int. Workshop on Discrete-Event System Design*, Przytok, Poland, 27-29 June 2001, pp. 25-34.

[3] A.A. Veselov, "D-extended Petri nets for simulating of digital devices," in *Proc. Annual Conf. Processor-oriented Methods and Tools for the Development of Information Systems*, Potsdam, Germany, 9-11 Oct. 2002, vol. 65, pp. 116-127.

[4] H. Kubatova. "Direct implementation of Petri net based model in FPGA", in *Proc. 2nd Int. Workshop on Discrete-Event System Design,* Dychow, Poland, 15-17 Sept. 2004, pp. 31-36.

[5] A.V. Lapin, D.A. Bulakh, A.V. Korshunov, and G.G. Kazennov, "The use of Petri nets as the basis of algorithm for gate level digital circuits simulation," in *Proc. IEEE East-West Design and Test Symp*., Yerevan, Armenia, 14-17 Oct. 2016, pp. 1-4.