International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 10 (2018) pp. 8482-8485

A Software Tool to Detect Race and Deadlock Conditions Caused by Leakage in Speed-Independent Circuits

Abdullah Baz

Senior, IEEE member and assistant Professor,
Computer Engineering Department,
Umm Al-Qura University, Makkah, Saudi Arabia.

Abstract

Asynchronous circuit designers mainly rely upon the timing relationship between gates to judge the correctness of their designs. They rarely consider the operating conditions, technology node features, and variations. Leakage current is one of the device properties that can cause indeterministic behavior in any circuit, especially asynchronous ones. For instance, leakage can change the state of flip-flips and registers or can discharge precharged bit-lines. These conditions can lead to deadlock or race conditions in asynchronous circuits, particularly speed-independent (SI) circuits. Unfortunately, existing asynchronous circuit design tools do not consider this issue during the design and testing phase. Therefore, it is the aim of this research to eliminate this gap. This paper proposes a new software tool that is capable of detecting race and deadlock conditions caused by leakage current in speed-independent circuits. The tool was developed in OCEAN, so it can be easily integrated with Spectre Circuit Simulator. The tool was tested on a number of SI circuits from literature, and results demonstrating the need for such a tool were obtained.

Keywords: Speed-independent circuits, asynchronous design, EDA tools, race, deadlocks.

INTRODUCTION

Microelectronic systems are designed and implemented in silicon integrated circuits (ICs) to process data. Inside the ICs, the computation and processing are achieved via the movement of electrons (in the form of signals) in the silicon layers. In order for the IC to meet its functional requirements, all signals have to move in the silicon layer in a synchronized manner. Otherwise, if any signal arrives earlier/later than its proper time, it will deliver/collect incorrect data [1]. For this reason, all microelectronic systems require a timing controller to regulate the data flow inside them. Depending upon the type of this controller, systems can be categorized as synchronous or asynchronous systems as follows [1].

A. Synchronous systems

Synchronous systems have only one clock generator, which produces a square wave signal with fixed duty cycle and period. This clock signal is responsible for triggering all hardware components to deliver and collect the data from the components at the correct time [1]. Figure 1 shows the block diagram of such a system.

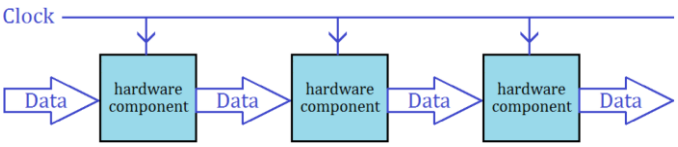

Figure 1. Synchronous system.

B. Asynchronous systems

In contrast to synchronous systems, asynchronous systems either have more than one clock or do not have a clock at all. Every two or more components inside the asynchronous system exchange data via a predefined handshaking protocol to guarantee that the data arrives/leaves the relevant components at the correct time. Bundled data and dual rail are the main types of protocols in asynchronous systems [1, 2]. Figure 2 shows an example of an asynchronous system that utilizes bundled data protocol to regulate its data flow.

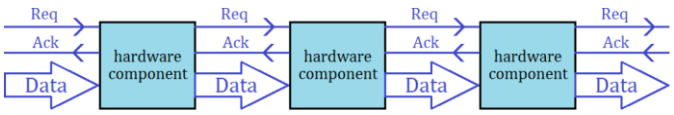

Figure 2. Asynchronous system with bundled data handshaking protocol.

Asynchronous design is crucial for many applications including globally asynchronous locally synchronous (GLAS), pipelines, multiple clock systems, and multi-core processors [2].

Moving one level from the system level down to the circuit level requires determining the method used to generate the handshaking signals and their temporal relations to the circuit components. This is known as the delay model of asynchronous circuits [1-3]. Delay model is the maximum delay the circuit components can tolerate before they malfunction. Based on the delay model, asynchronous circuits can be categorized as delay-insensitive (DI) circuits or speed-independent (SI) circuits [2]. The former currently is the most robust type of circuit, and can tolerate any amount of delay in both gates and wires. The latter is sub-optimal, as it can tolerate any amount of delay in gates only as long as the wire delays are marginal. Most of the existing asynchronous circuits are SI because DI circuits are power hungry and difficult to implement [1, 2]. For DI and SI circuits to work correctly, they must be free from any types of faults or protocol violations under all operational

conditions [1-3]. Deadlock and critical race are two important types of faults that can cause the asynchronous circuit to malfunction. The text below describes each one in detail and gives an example of its mechanism.

1) Deadlock

Deadlock can be defined as the inability of the circuit to proceed to the next state. This issue generally occurs when two or more gates await an output from each other before producing the next output. Consequently, these gates mutually block each other from progressing. Deadlock in arbiters is one of the famous examples of deadlock in asynchronous circuits [1-3].

2) Race

Race condition occurs in asynchronous circuits when different delays cause the circuit to change its states in an unpredictable manner. An example is the case in which a circuit is supposed to change its state from a to b and some operational conditions cause this circuit to pass an intermediate state c. If these operational conditions affect the final state of the circuit in a way that results in no final stable state or no unique stable state, the circuit is said to have a critical race condition. Otherwise, there is no race or the race is not critical [1-5].

RESEARCH GAP AND PROBLEM STATEMENT

As demonstrated in the previous section, asynchronous systems and SI circuits are crucial for many applications. Importantly, before production, SI circuits have to be free from any faults or protocol violations [1]. Deadlock and critical race are two types of faults that might occur in SI circuits. If any of these faults occurs, the circuit and the whole system will malfunction [2]. Since the introduction of SI circuits, designers have made all possible efforts to guarantee that the circuit is free of deadlocks and races. However, in the past, designers were only interested in deadlock and race conditions that are primarily caused by incorrect circuit design or metastability [1-5]. Leading universities and researchers in this area have proposed many techniques and software tools to address this issue. Examples include Balsa by University of Manchester, CAST by California Institution of Technology, Clp by Newcastle University, MINIMALIST by Columbia University, Petrify by Universitat Politenica de Catalunya, and VerySAT by Newcastle University [1-5].

As the technology scales down in an aggressive manner, the leakage current increases significantly [4]. The effect of this dramatic increase reaches every part of the circuit and causes many operational issues [1]. These issues include discharging precharged lines [5] and flipping the state of flip-flops and registers [4]. Consequently, even if the SI circuit is correctly designed, the leakage can cause deadlock or race conditions [4-5] if it exceeds a certain limit. Until the time of writing this paper and to the best of the author's knowledge, there is no technique nor software tool that can detect the deadlock and race conditions caused by leakage in SI circuits. Therefore, the main aim of this research paper is to investigate this research gap.

CONTRIBUTION

We proposed to address the highlighted issue via developing a software tool that can be integrated with the existing simulation tool. We chose to integrate our proposed tool with Spectre Circuit Simulator, by Cadence Design Systems, due to its high accuracy and popularity in the academic and research community. Open Command Environment for Analysis (OCEAN) is a scripting language that can be used in Cadence Design Systems to simulate analog and digital circuits at the differential equation level. We chose to develop our tool in OCEAN so it can be easily used to test SI circuits whenever Spectre simulation is available. It is worth highlighting that this tool is not an asynchronous design circuit tool. However, it is a tool that can be used after the circuit is designed to detect if the leakage current in the technology node can cause deadlock or critical race conditions.

Based on definition, SI circuit is a circuit that can tolerate any gate delay so long as wire delay is negligible. This definition guided us to the theory behind developing our proposed tool. Basically, the delay increases the effect of leakage because the more time the circuit stays ideal, the more current it leaks. This leakage current can discharge precharged lines or flip the state of flip-flops and registers, which might put the circuit in critical race or deadlock conditions. Unfortunately, asynchronous circuit design tools, including those mentioned earlier in this text, do not consider this effect [1-5]. Moreover, circuit designers overlook this behavior and only consider the general timing relationships between gates. The mechanism behind our proposed tool is inserting a delay element after each gate in the circuit under test and waiting until the leakage takes place inside the circuit components. It is worth highlighting that the amount of delay in each gate is calibrated so that it does not exceed the corners of the technology node. Doing so makes the test more reliable and does not overestimate the faults. Following that, the tool compares the final outputs of the circuit with delay elements with its counterpart without delay elements. If all signals are generated in the same order, then the circuit is said to be not affected by leakage. If, however, any signal is generated in the wrong position, then the circuit is said to have faults due to leakage.

The algorithm of the OCEAN code is shown below:

```
NetlistWithDelay(netlist) //a function to generate a netlist after inserting delays
{
fnp //an array to hold the maximum delay of the devices per technology node
for each gate G_i in the SI circuit loop
    for each device D_i in G_i loop
        add the corresponding fnp element to the device in each gate
    end loop
end loop
return netlistwithdelay
```

```
}
Signals = Simulate(netlist)
SignalsWithDelay = Simulate(netlistwithdelay)

for each signal S_i in Signals loop
   for each signal 〖SWD〗_i in SignalsWithDelay loop
      if S_i <> 〖SWD〗_i then
         return false
      end if
   end loop
end loop
```

EXPERIMENT

We used the above listed algorithm to test a number of SI circuits from literature. The testing ranged from circuits of a few gates to circuits of tens of gates. Generally, deadlock and race conditions occurred in high replication circuits (HRC) [18], where a circuit of a few transistors is repeated hundreds or thousands of times to form a bigger circuit. Examples of HRC include banks of SRAM, FF, and registers. As a general rule, the failure occurs because each replicated circuit leaks an amount of current that is accumulated with others due to the high number of replications to cause the deadlock or race condition. The problem becomes worse if the devices are low threshold devices and/or the temperature increases.

CONCLUSION

This paper introduced a software tool to detect race and deadlock conditions caused by leakage in speed-independent circuits. The tool was developed in OCEAN, which is a scripting language that can be used in Cadence Design Systems to simulate analog and digital circuits at the differential equation level. The developed tool was tested on a number of SI circuits from literature, and the obtained results confirm the existence of failures in HRC circuits due to leakage.

ACKNOWLEDGMENT

The author would like to acknowledge and thank the Deanship of Scientific Research (DSR) at Umm Al-Qura University (Projects No: 15-COM-3-2-0001, 15-COM-4-1-0001, 15-COM-4-1-0002, & 17-COM-1-01-0008) for their financial support.

REFERENCES

[1] Neil Weste, David Harris, "CMOS VLSI Design: A Circuits and Systems Perspective", Addison Wesley, 2011.

[2] Jens Sparsø, Steve Furber, "Principles of Asynchronous Circuit Design: A Systems Perspective", Springer Science & Business Media, 2013.

[3] Morris Mano, "Digital Design", EBSCO Publishing, Inc., 1998.

[4] Siva Narendra, Anantha Chandrakasan, "Leakage in Nanometer CMOS Technologies", Springer Science & Business Media, 2006.

[5] Abdullah Baz, Delong Shang, Fei Xia, Alex Yakovlev, "Self-Timed SRAM for Energy Harvesting Systems", Journal of Low Power Electronics, Volume 7, Number 2, 2011, pp. 274-284.

[6] Feng Shi, Yiorgos Makris, "Fault simulation and random test generation for speed-independent circuits", Proceedings of the 14th ACM Great Lakes symposium on VLSI, pp. 127-130.

[7] Xi Chen, Abhijit Davare, Harry Hsieh, Alberto Sangiovanni-Vincentelli, Yosinori Watanabe, "Simulation based deadlock analysis for system level designs", Proceedings of the 42nd annual Design Automation Conference, pp. 260-265.

[8] Josef Haid, Bernd Zimek, Thomas Leutgeb, Thomas Künemund, "Impact of leakage current on data retention of RF-powered devices during amplitude demodulation-based communication", Proceedings of the conference on Design, automation and test in Europe, pp. 784-787.

[9] Sudeep Pasricha, Nikil Dutt, "On-Chip Communication Architectures: System on Chip Interconnect", Morgan Kaufmann, 2010.

[10] Kaushik Roy, Saibal Mukhopadhyay, Hamid Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deepsubmicrometer CMOS circuits", Proceedings of the IEEE, pp. 305-327.

[11] Michael Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, TN Vijaykumar, "Gated-V dd: a circuit technique to reduce leakage in deep-submicron cache memories", Proceedings of the 2000 international symposium on Low power electronics and design, pp. 90-95.

[12] Kaushik Roy, Sharat C Prasad, "Low-power CMOS VLSI circuit design", John Wiley & Sons, 2009.

[13] Saibal Mukhopadhyay, Hamid Mahmoodi, Kaushik Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 24, Number 12, 2005, pp. 1859-1880.

[14] S-H Yang, Michael D Powell, Babak Falsafi, Kaushik Roy, TN Vijaykumar, "An integrated circuit/architecture approach to reducing leakage in deep submicron high-performance I-caches", The Seventh International Symposium on High-Performance Computer Architecture, pp. 147-157.

[15] Mark C Johnson, Dinesh Somasekhar, Kaushik Roy, "Leakage control with efficient use of transistor stacks in single threshold CMOS", Proceedings of the 36th annual ACM/IEEE Design Automation Conference, pp. 442-445.

[16] Ali Keshavarzi, Kaushik Roy, Charles F Hawkins, "Intrinsic leakage in low power deep submicron CMOS

ICs", Proceedings of Test Conference, pp. 146-155.

[17] Saibal Mukhopadhyay, Kaushik Roy, "Modeling and estimation of total leakage current in nano-scaled CMOS devices considering the effect of parameter variation", Proceedings of the 2003 international symposium on Low power electronics and design, pp. 172-175.

[18] Amith Singhee, Rob A. Rutenbar, "Novel Algorithms for Fast Statistical Analysis of Scaled Circuits".