PROCEEDINGS OF SCIENCE

Security Path Checking of A Circuit with Behavior Description

Chao Ma[1]
School of Information Science and Engineering, Lanzhou University
Lanzhou, 730000, China
E-mail: mach2015@lzu.edu.cn

Anping He[2a]**; Tingting Jia**[b]**; Lian Li**[c]
School of Information Science and Engineering, Lanzhou University
Lanzhou, 730000, China
E-mail: [a]heap@lzu.edu.cn;[b]jiatt15@lzu.edu.cn;[c]lil@lzu.edu.cn

Zhihua Feng
Institute 706, Second Academy of China Aerospace Science and Industry Corporation
Beijing, 100854, China
E-mail: zhihua_feng@126.com

Model Checking is one of the formal tools that uses state space searching to automatically verify whether a finite state system can meet the design specifications or not. In the course, IP core is widely used in circuit design. However, it still remains unknown whether the critical security data would be modified or leaked through IP cores. In this paper, an innovative tool chain based on black box tint technique is proposed. We used the tint based method to tint every input and output, which could verify the security properties of every path in a circuit. NuSMV was selected as the model checker. Finally, the experiments suggested that the security path checking of a circuit could be easily verified by using this pattern.

ISCC2017
16-17 December 2017
Guangzhou, China

[1] Speaker

[2] Corresponding author, This work is support by the Chinese NSF NO. 61402121 and 61650207, the Fundamental Research Funds for the Central Universities of Lanzhou University, No. 861914, and Guangxi Science and Technology Project (AB17129012).

PoS(ISCC 2017)046

1. Introduction

1.1 The Necessity of Security Path Checking

In the recent 20 years, Model Checking, as one of the most successful automatic formal verification techniques, can exhaust all states of a system and automatically verify whether a model can satisfy the requirements. Model Checking is mainly used for complex hardware and software systems with high security requirements [1].

Model Checking uses state space searching to verify automatically whether a finite state system meets for design specifications [2]. It features advantages of full automation, fast and efficient. If a security property is not satisfied, Model Checking will indicate reasons. Researchers can improve the system to meet such requirements.

In electronic design, Intellectual Property core, IP core for short, is a reusable logic unit, which can be used as the building blocks within system on chip (SOC), application-specific integrated circuit (ASIC) or field-programmable gate array (FPGA). It could shorten the development cycle, increase the productivity and improve the reliability systems based on FPGA and SOC. For these reasons, IP core will become the mainstream direction of integrated circuit in the future. It has been wildly applied to speech recognition, image processing and software engineering; therefore, there's urgently increasingly demand for integrated circuits with strong abilities of security, reliability and confidentiality [3-5].

Nowadays, there have been reports on security threats of data interaction among IP cores in the information systems. It has remained unknown whether the IP core should be responsible for the modification or leakage of the critical data. Thus, experiments are implemented to evaluate the security reliability of systems based on IP cores. The security property of systems can be divided into two categories: systems that could keep key data from leakage and systems that could keep key data from interference. In these experiments, key data, i.e. tainted data, were marked in the IP core code. As the data between IP cores can be observed, traceable and verifiable, it is easy to monitor the propagation of critical data and evaluating the security property. Model Checking can verify the IP core code in the traceable information flow by using this pattern.

1.2 Related Work

Håkansson and Rosencrantz solved a problem that the operating system automatically trusted any externally connected peripheral. They used Model Checking to confirm in which cases the hardware peripherals can be trusted. In their experiments, a model of the universal asynchronous transmitter/receiver (UART), a model of the main memory (RAM) and a model of a DMA controller were constructed. These models were used to analyze interactions between user processes, hardware peripherals and the memories. The results suggested that hardware peripherals were secure when these devices were properly configured after connection [2].

In a paper written by Schwarz and his colleagues, software Model Checking for security properties had been used on an extremely large scale. MOPS, a static analyzer, had been used to evaluate the security properties in a Linux distribution system. As a result, 108 exploitable bugs had been discovered, indicating that Model Checking could be a feasible and integral part of software development process. It is possible to develop models of incorrect and insecure program behavior that were precise enough to prevent false positives from dwarfing the real bugs [6].

Lowe wrote that Model checking has been proved as a successful approach to validate the security protocols. If a model checker fails to find an attack, it means that there is no attack on

the particular small system, while there may be an attack on certain larger system running the same protocol. Model Checking has been proven qualified for evaluating the protocol security on small systems [7].

2. Tint Based Method

2.1 How to Tint Data

In general, it is difficult to formalize the security behavior directly because security behavior of IP core is complex. Therefore, the code and security description of IP core are processed respectively, and the high-level attribute description is combined with low-level code behavior for security checking. In order to describe the security risks, IP core data confidentiality security property model and IP core data integrity security property model should be established [8]. Based on evaluation of the IP core security properties, the IP core code, which can effectively detect the violation of security requirements, has been verified. The IP core security requirements of information systems can be divided into two categories: a) systems that could keep key data from being leaked, i.e. confidentiality. b) Systems that could keep key data from being maliciously interfered, i.e. integrality. This paper describes how to evaluate the security behavior property of the IP core key data by using a formal descriptive grammar. Specific description is defined as follows:

- Select a module in a system that is sensitive and a security object in this sensitive module (e.g. the secret key of storage).
- Ensure whether there is an interaction between the security object and IP core with unknown security.
- Confirm the security requirements of the security object.
- Analyze threats that may occur in the system.
- According to the security object, IP core, security requirements of security object and threats describe security properties of the security object.

The above security properties define the security of data interaction among IP cores. However, it is difficult to implement these security properties by using ordinary tools. IP core code lacks the logic for security checking. In order to solve these problems, the tint technique is used to process IP cores code to assist validations of the security properties. The tint technique does not affect normal functions of the system and can work normally within the original system. In this way, the tint technique can provide provable evidences for security property descriptions and a reliable tracking environment for the entire system.

In SOC and FPGA systems integrated with a large number of IP cores, certain IP core suppliers could provide gate-level IP core code in order to protect their own intellectual property rights [9]. Thus, this kind of IP core code could be regarded as a black box. And a fine granularity gate-level information flow tracking strategy should be adopted.

2.2 The Verification Method Based on Tint

AND gate, OR gate and NOT gate were chosen as basic logical unit collections. The basic logical unit collection is complete in describing any digital circuits system. In the process of constructing the information flow tracking model, Boolean logic was used to encode the tint data. Besides, the security property labels were added to input of the logical unit. In this paper, if the tint label of data was *1* in Boolean logic, it meant that the information in data was tinted. On the contrary, if the tint label of data was *0* in the Boolean logic, it meant that the information in data was clean, in other words, not tinted. Considering the integrity of information,

untrustworthy data was considered as the tint data. And secret data were regarded as tint data according to the information confidentiality.

According to the information science, if *A* influences *B*, we can conclude that *A* flows to *B*. The actual effects of these logic gates on the output were analyzed by using the fine granularity information flow analysis. Each bit stream was accurately measured. Only when the input of the logic gate had an actual impact on the output, the information flow could be contained in the flow from the input to the output [14]. A few examples are as follows.

As shown in Fig 1, two inputs AND gate tinted truth table, the input a_t , b_t and the output o_t were tint labels of a , b and o , When a_t , b_t or o_t was *1*, it meant this port was tinted. Obviously, when both a and b were tinted, the output o was tinted. On the contrary, when neither of a nor b was tinted, the output o was not tinted. Moreover, in the 1st line of the truth table, $a=0 \wedge b=0 \wedge a_t=0 \wedge b_t=1$, b_t changed the value of b , but output o was still *0*. Because the input was a not tint, $b_t=1$ can not influence o . So $o_t=0$.

NO	a	b	a_t	b_t	o	o_t
1	0	0	0	1	0	0
2	0	0	1	0	0	0
3	0	1	0	1	0	0
4	0	1	1	0	0	1
5	1	0	0	1	0	1
6	1	0	1	0	0	0
7	1	1	0	1	1	1
8	1	1	1	0	1	1

Figure 1: AND Gate Tinted Truth Table

The two inputs OR gate tinted truth table was shown in Fig. 2. The OR gate was more sensitive than the AND gate. For example, in the 1st line of the truth table, $a=0 \wedge b=0 \wedge a_t=0 \wedge b_t=1$, although $a_t=0$, $b_t=1$ influenced the output o . Another situation was shown in the 5th line of the truth table, $a=0 \wedge b=1 \wedge a_t=1 \wedge b_t=0$, $b_t=1$ changed the value of , b but the output o was still *1* because of a . The 6th line was the same.

For flip-flops, this method can be extended. Because the current state of flip-flops was related to the previous state. JK flip-flop was taken for example (Fig. 3), Q_n means the previous state of flip-flops, and Q_{n+1} means the next state of flip-flops. When $J_t=0 \wedge K_t=1 \wedge Q_n=0$ or $J_t=1 \wedge K_t=0 \wedge Q_n=1$, s the $Q_t=0$. In addition, Q_t was always *1* in other situations.

NO	a	b	a_t	b_t	o	o_t
1	0	0	0	1	0	1
2	0	0	1	0	0	1
3	0	0	1	1	0	1
4	0	1	0	1	1	1
5	0	1	1	0	1	0
6	0	1	1	1	1	0
7	1	0	0	1	1	0
8	1	0	1	0	1	1
9	1	0	1	1	1	0
10	1	1	0	1	1	0
11	1	1	1	0	1	0
12	1	1	1	1	1	1

Figure 2: OR Gate Tinted Truth Table

NO	J	K	J_t	K_t	Q_n	Q_{n+1}	Q_t
1	0	0	0	1	0	0	0
2	0	0	0	1	1	1	1
3	0	0	1	0	0	0	1
4	0	0	1	0	1	1	0

5	0	1	0	1	0	0	0
6	0	1	0	1	1	0	1
7	0	1	1	0	0	0	1
8	0	1	1	0	1	0	0
9	1	0	0	1	0	1	0
10	1	0	0	1	1	1	1
11	1	0	1	0	0	1	1
12	1	0	1	0	1	1	0
13	1	1	0	1	0	1	0
14	1	1	0	1	1	0	1
15	1	1	1	0	0	1	1
16	1	1	1	0	1	0	0

Figure 3: JK Flip-flop Tinted Truth Table

3. An Innovative Tool Chain

3.1 Basic Framework

SMV is a language for verifying the finite state systems (FSM) [2]. The NuSMV can be regarded as the representation of synchronous and asynchronous finite state systems, and for analyzing the specifications as expressed in Computation Tree Logic (CTL) and Linear Temporal Logic (LTL), BDD-based and SAT-based model checking techniques could be used. The NuSMV project aims at the development of a state-of-the-art symbolic model checker. It is applicable in technology transfer projects. The NuSMV project is a well-structured, open, flexible and documented platform for model checking, it is robust and can satisfy the industrial system standards. The software is available and distributed under the LGPL v2.1 license. NuSMV, the model checker, is an Open Source license that allows free academic and commercial usage of NuSMV [10-13].

The basic framework of an innovative tool chain is shown in Fig. 4. In the beginning, the behavior level description and RTL level description have to be synthesized or translated manually to obtain gate-level Verilog description. Then a Java program is used to transform Verilog into a kind of data structure (describe in Section 3.3). This data structure can record and describe the internal circuit structure and logical structure of the input Verilog file in detail. After recombination and coding, the information flow tracking model is added, NuSMV code can be generated by the Java code with this kind of logical data structure. In summary, if the Verilog design to be verified is not a gate-level description, it has to be integrated into the gate-level description and then put into the conversion program. The output is NuSMV code for model checking.

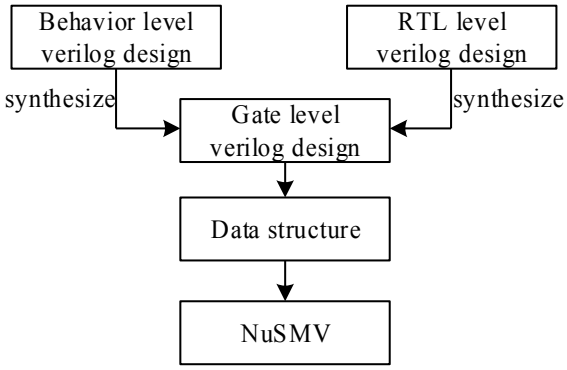

Figure 4: Basic Framework

3.2 A Tinted SMV Design

We chose SMV and NuSMV since they were already available, well established, free of charge and easy to use. Algorithm 1 is tint two input AND gate NuSMV design. A tint input should be added to each original input and every output as well. As we can see, a two input AND gate is transformed to four input and two output. The original output o and the tint output o_t are assigned separately. So the additional tint output o_t cannot influence the original output. Algorithm 2 shows the tint two input OR gate.

Algorithm 1 Tint two input AND gate NuSMV design
Input: The original input a, b and the tint input a_t, b_t;
Output: The original output o and the tint output o_t;
1: $o=a$ & b
2: $o_t=(b$ & $a_t)$ | $(a$ & $b_t)$ | $(a_t$ & $b_t)$;
3: **return** o,o_t;

Algorithm 2 Tint two input OR gate NuSMV design
Input: The original input a, b and the tint input a_t, b_t;
Output: The original output o and the tint output o_t;
1: $o=a$ | b
2: $o_t=(a$ & !b & a_t & $b_t)$ | (!a & b & !a_t & $b_t)$ | (!a & !b & $(a_t$ | $b_t))$ | $(a$ & b & a_t & $b_t)$;
3: **return** o,o_t;

Algorithm 3 Tint JK flip-flop NuSMV design
Input: The original input J, K and the tint input J_t, K_t;
Output: The original output Q and the tint output Q_t;
1: **if** $clk == 1$ **then**
2: **if** $(J_t == 0$ & $K_t == 1$ & $Q == 0)$ | $(J_t == 1$ & $K_t == 0$ & $Q == 1)$ **then**
3: $Q_t = 0$;
4: **else**
5: $Q_t = 1$
6: **end if**
7: **if** $J == 0$ & $K == 0$ **then**
8: Q hold state;
9: **else if** $J == 0$ & $K == 1$ **then**
10: $Q = 0$;
11: **else if** $J == 1$ & $K == 0$ **then**
12: $Q = 1$;
13: **else if** $J == 1$ & $K == 1$ **then**
14: Q toggle;
15: **end if**
16: **else**
17: Q; Q_t hold state;
18: **end if**
19: **return** Q; Q_t;

Since SMV is a state-based verification language, the problem could be simplified. The *clk* is the clock signal. The clock signal only has two states *0* and *1,* and the clock signal will change state every time when the system state changes. Under this circumstance, it could be seen that when $clk=1$, it must pass through the rising edge of the clock. Algorithm 3 is the tint JK flip-flop NuSMV design. Line 2~Line 6 are the main tint output logical algorithm. As we described in Section 2.2, $Q_t=0$ when $J_t=0\wedge K_t=1\wedge Q_n=0$ or $J_t=1\wedge K_t=0\wedge Q_n=1$, and in other situations, $Q_t=1$.

3.3 From Gate-Level Verilog to Tinted SMV

Considering the efficiency and extendibility of the conversion program, all two input gates and flip-flop of the file library are built. The conversion program generates the corresponding NuSMV code according to the library files. There are two parts of files in library. One part of the files are tint gates and tint flip-flop NuSMV design. They are used to generate tint NuSMV code. Others are normal gates and flip-flop NuSMV design, the program can also generate normal NuSMV code for verification.

The data structure is mainly shown in Fig 5. As we described in Section 3.1, input the gate-level Verilog file into the Java conversion program. In general, a Verilog file contains many

"module". For the data structure of the program, one "module" is one "VComponent" in Fig. 5. The "VComponent" is the basic unit of the data structure. It contains "VConnection", "VDeclaration" and "VInstantiation". This "VConnection" the interconnection between two "VComponents". In addition, the "VDeclaration" describes declaration variables in the "VComponent", including "input", "output", "wire" and "reg", etc.. A "VInstantiation" is an instantiation of a "module" in Verilog file, which contains "VConnection" and "VExpression". This "VConnection" describes input and output of the "VInstantiation". Finally, the "VExpression" represents an expression in a code such as "assign" statement.

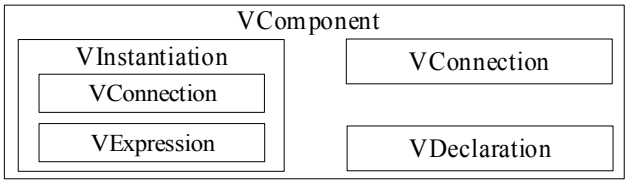

Figure 5: Main Data Structure

This kind of data structure divides the Verilog file into a logical set. After recombination and serialization, SMV code can be expediently generated.

3.4 Configuration of Platform

Operating system: Ubuntu 14.04 LTS;
Programming language: Java version 1.8.0; NuSMV 2.6.0.

4. Conclusion

The tool chain in this paper has verified the security path problem of multi bit full adder (as a black box). Nevertheless, as each input from adder can exert direct effect on the output, many risks have been detected. It is unrepresentative. But it also proved the theories are feasible. This innovative tool chain will be applied to the security path checking of CAN bus, which will be described in the next paper.

In this paper, an automated security path checking method of a circuit was proposed. NuSMV was selected as a model checker. The tint technique was used to design an innovative tool chain. The security path checking problem of the behavior level, e.g., the Verilog description of a circuit was studied. The security property can be detected while the IP cores were used in a circuit with high security requirements. All of these theories were applied in the system and suggested that they were feasible.

References

[1] Edmund M. Clarke, Orna Grumberg, Doron A. Peled, *Model Checking* [M], The MIT Press, London, 1999:1-4, 27-30

[2] J. Yao Håkansson, N. Rosencrantz, *Formal Verification of Hardware Peripheral with Security Property* [D]. 2017, KTH Computer Science and Communication, Stockholm, Sweden.

[3] A. Sengupta and S. Bhadauria and S. P. Mohanty, *Embedding low cost optimal watermark during high level synthesis for reusable IP core protection* [C] 2016 IEEE International Symposium on Circuits and Systems (ISCAS): 974-977.

[4] Liu Y, Wu L, Niu Y, et al. *A High-Speed SHA-1 IP Core for 10 Gbps Ethernet Security Processor*[C]. Eighth International Conference on Computational Intelligence and Security. IEEE, 2012:237-241.

[5] Deshpande A. *Verification of IP-Core Based SoC's*[C] International Symposium on Quality Electronic Design. IEEE Computer Society, 2008:433-436.

[6] Schwarz, Oliver, and M. Dam. *Formal Verification of Secure User Mode Device Execution with DMA* [J]. Hardware and Software: Verification and Testing. Springer International Publishing, 2014:236-251.

[7] Lowe, Gavin. *Towards a Completeness Result for Model Checking of Security Protocols (Extended Abstract)* [J]. Journal of Computer Security 7.2-3(1998):96-105.

[8] Suneeta, Srinivasan R, RamSagar, et al. *SoC implementation of three phase BLDC motor using Microblaze soft IP core*[C]. International Conference on Computer, Communications and Electronics. 2017:360-364.

[9] Chakraborty R S, Bhunia S. *HARPOON: an obfuscation-based SoC design methodology for hardware protection*[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2009, 28(10):1493-1502.

[10] Cimatti A, Clarke E, Giunchiglia F, et al. *NUSMV: a new symbolic model checker*[J]. International Journal on Software Tools for Technology Transfer, 2000, 2(4):410-425.

[11] Cimatti A, Clarke E M, Giunchiglia E, et al. *NuSMV 2: An OpenSource Tool for Symbolic Model Checking*[C] International Conference on Computer Aided Verification. Springer-Verlag, 2002:359-364.

[12] Bozzano M, Villafiorita A. *Improving System Reliability via Model Checking: The FSAP/NuSMV-SA Safety Analysis Platform*[M] Computer Safety, Reliability, and Security. Springer Berlin Heidelberg, 2003:49-62.

[13] R. Cavada, A. Cimatti, C. Arthur Jochim, G. Keighren, E. Olivetti, M. Pistore, *NuSMV 2.6 User Manual* [OL], 2015, http://nusmv.fbk.eu/.

[14] Yuan Yue, *An Equivalence Checking Method for Circuits with Black Boxes Based on Logic Cone and SAT* [D]. 2005, School of Information Science and Engineering, Lanzhou University, Lanzhou, China (in Chinese).