SAT Encoding-based Verification of Sneak Path Problem in Via-switch FPGA

Ryutaro Doi [†, ‡] Masanori Hashimoto [†]
[†] Department of Information Systems Engineering,
Graduate School of Information Science and Technology,
Osaka University
Email: {doi.ryutaro, hasimoto}@ist.osaka-u.ac.jp
[‡] Research Fellow of Japan Society for the Promotion of Science

***Abstract*—FPGA that introduces via-switches, a kind of nonvolatile resistive RAMs, for crossbar implementation is drawing attention due to higher integration density and performance. However, programming those switches arbitrarily in a crossbar is not trivial since a programming voltage must be given through signal wires that are shared by multiple via-switches. Consequently, depending on the previous programming status, unintentional switch programming may occur due to signal detour, which is called sneak path problem. This paper encodes programming operations in via-switch based crossbar into a satisfiability problem and rigidly verifies the sneak path problem. Verification results show that sneak path problems can be solved by imposing a specific programming constraint.**

I. Introduction

Field programmable gate arrays (FPGAs) become more popular since the development cost of application specific integrated circuits (ASICs) is elevating due to the device miniaturization and larger scale integration. However, conventional FPGAs are still inferior to ASICs regarding operating speed, power consumption, and implementation area [1]. These drawbacks arise from a huge number of programmable switches that are included in FPGAs to acquire reconfigurability. In static random access memory (SRAM)-based FPGAs, which are the most widely used FPGAs, a programmable switch is composed of a transmission gate for switching and an SRAM cell to hold the on/off-state of the switch. Since these components consist of transistors, the transmission gate has high resistance and large capacitance and the SRAM cell that requires six transistors consumes large area. Therefore, SRAM-based programmable switches lead to the degradation of interconnect performance and area efficiency [2].

To overcome drawbacks of conventional FPGAs, FPGAs that exploit resistive random access memories (RRAMs) as programmable switches instead of SRAM-based ones are widely studied [3]–[9]. In these RRAM-based FPGAs, however, one or two access transistors per a programmable switch are required for switch programming. The access transistor is relatively large despite the small footprint of an RRAM-based switch, and hence it interferes with further area reduction. For eliminating access transistors, nonvolatile via-switches are actively developed [10], [11]. The via-switch consists of atom switches, which are a kind of nonvolatile RRAMs developed for application to FPGAs, and varistors in place of access transistors.

In the via-switch FPGA, the crossbar, which has a via-switch at each intersection of horizontal signal wire and vertical signal wire, is responsible for the routing of interconnects. However, programming those via-switches arbitrarily in a crossbar is not trivial since a programming voltage must be given through signal wires that are shared by multiple via-switches. In this case, unintentional switch programming may occur depending on the previous programming status due to signal detour, which is called sneak path problem. This problem interferes the reconfiguration of FPGA, and hence the verification of occurrence conditions and countermeasures is crucially important.

This paper encodes programming operations and occurrence conditions of sneak path problem in a via-switch crossbar into a Boolean satisfiability problem (SAT), and rigidly verifies the sneak path problem by using an SAT solver. Verification results show that the programming status of crossbar where the sneak path problem occurs can be detected by the proposed SAT encoding-based verification. Furthermore, we confirm that sneak path problems can be solved by imposing a specific programming constraint.

The remainder of this paper is organized as follows. Section II explains the structure of via-switch FPGA and sneak path problem. Section III presents the SAT encoding-based verification methodology for sneak path problem in via-switch FPGA followed by the verification results in Section IV. Concluding remarks are given in Section V.

II. Via-switch FPGA

A. Via-switch

The via-switch is a nonvolatile, rewritable, and compact switch that is developed to implement a crossbar switch by Banno et al. [10], and it is composed of atom switches and varistors. Here, we explain the device structure, functionality, and characteristics in the following. The programming of via-switch crossbar will be explained later in Section II-B.

The atom switch consists of a solid electrolyte sandwiched between copper (Cu) and ruthenium (Ru) electrodes as shown in Figure 1-a. By applying a positive voltage to the Cu electrode, a Cu bridge is formed in the solid electrolyte, and

2159-3477/18/$31.00 ©2018 IEEE
DOI 10.1109/ISVLSI.2018.00084

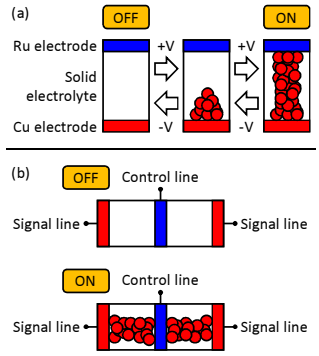

Fig. 1. Structure and operation of (a) atom switch and (b) CAS.

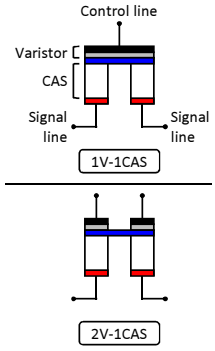

Fig. 2. Structures of via-switch. The top is 1V-1CAS structure, and the bottom is 2V-1CAS structure.

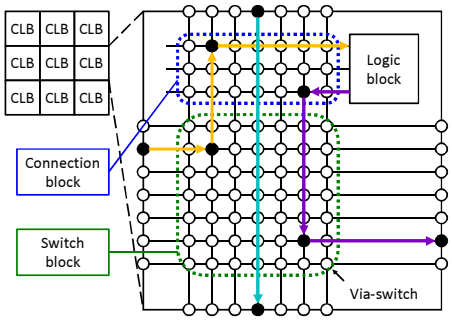

Fig. 3. Structure of via-switch FPGA.

the switch turns on. On the other hand, when a negative voltage is applied, Cu atoms in the bridge are reverted to the Cu electrode, and then the switch turns off. The switching between on-state and off-state is repeatable, and each state is nonvolatile. For improving the device reliability, the complementary atom switch (CAS) is devised, where it consists of two atom switches connected in series with opposite direction as shown in Figure 1-b. In the programming of CAS, a pair of signal line and control line supply a programming voltage to each atom switch, and two atom switches are programmed sequentially. During normal operation, on the other hand, only signal lines are used for routing [9].

To accurately provide the programming voltage only to the target CAS in a switch array, the varistor is introduced into the via-switch. Figure 2 shows the structure of via-switch, and the varistor is connected to the control terminal of CAS. When a voltage higher than the threshold value (programming voltage) is applied between the signal and control lines, the varistor supplies programming current to an atom switch. On the other hand, the varistor isolates the control lines from the signal lines during normal operation [10]. As shown in Figure 2, two structures of via-switch, one-varistor-one-CAS (1V-1CAS) and two-varistor-one-CAS (2V-1CAS) structure, are proposed.

Here, we summarize contribution of via-switch to FPGAs. The footprint, on-resistance, and capacitance are 18 F^2, 200 Ω, and 0.14 fF respectively [10], [12]. Thanks to these characteristics, the area efficiency and performance of via-switch FPGA are dramatically improved compared to SRAM-based one. Hotate et al. report that the crossbar density is improved by 26x, and the delay and energy in the interconnection are reduced by 90% or more [12].

B. Sneak Path Problem in Via-switch FPGA

The structure of via-switch FPGA is an array of configurable logic blocks (CLBs), and each CLB is composed of logic block and crossbar where a via-switch is placed at each intersection of signal lines as shown in Figure 3 [12]. The via-switch in the crossbar is responsible for connection and disconnection between the horizontal and vertical signal lines. Besides, the top half of the crossbar serves a function of input and output multiplexers to the logic block and corresponds to the connection block in conventional FPGAs. On the other hand, the bottom half of the crossbar, which corresponds to the switch block, routes global interconnections. The logic block organizes combinational and sequential circuits.

The following explains the programming of a via-switch crossbar and sneak path problem. The crossbar structure with the 1V-1CAS via-switch is illustrated in Figure 4. Signal lines are placed horizontally and vertically, whereas control lines are aligned diagonally. Figure 4 also shows an example of programming steps in 2x2 crossbar where an atom switch is turned on at each step. Two programming drivers are activated at each step, and a positive voltage is given to one of the signal lines, and a ground voltage is given to one of the control lines. Other lines are floated. Steps 1 and 2 successfully turn on the via-switch at the bottom left. However, the next programming of the top left via-switch at steps 3 and 4 cannot be performed correctly. The atom switch that composes the bottom right via-switch is under programming unintentionally at step 4 since the positive voltage is provided through the on-state via-switch at the bottom left. Such an unintentional switch programming due to signal detour that is caused by on-state via-switches is the sneak path problem.

Next, Figure 5 shows the crossbar structure using 2V-1CAS via-switches and an example of programming steps, where this example is the same as Figure 4. In this crossbar structure, both signal and control lines are aligned horizontally and vertically. A pair of the signal and control lines crossing at the via-switch of interest are used for switch programming at each step, whereas other lines are floated. We can see that two via-switches at the bottom left and top left are successfully turned on without sneak path problem at steps 1-4. If turning ON the bottom right or top right via-switch after step 4, on the other hand, sneak path problem occurs.

The sneak path problem interferes the reconfiguration of FPGA, and hence it is essential to verify the occurrence conditions and countermeasures of this problem. The following sections verify the sneak path problem by using SAT encoding.

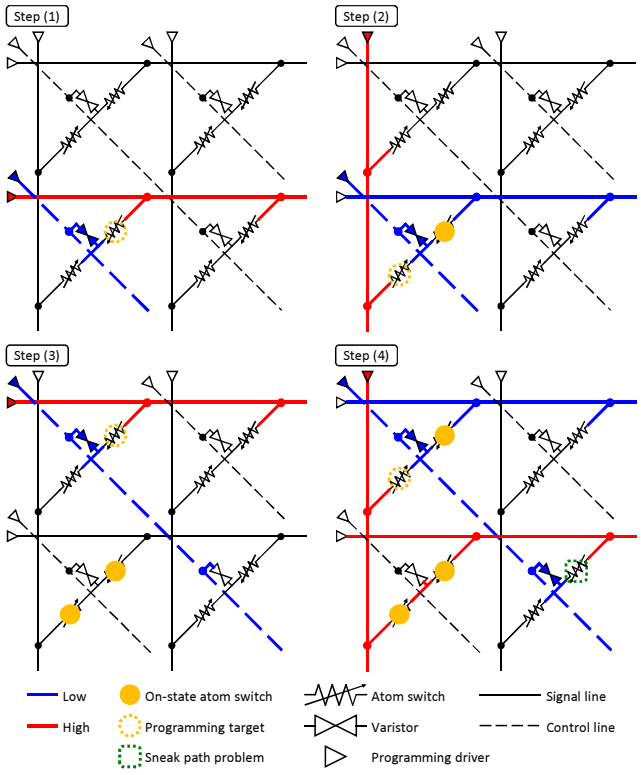

Fig. 4. Crossbar structure with 1V-1CAS via-switch and switch programming steps.

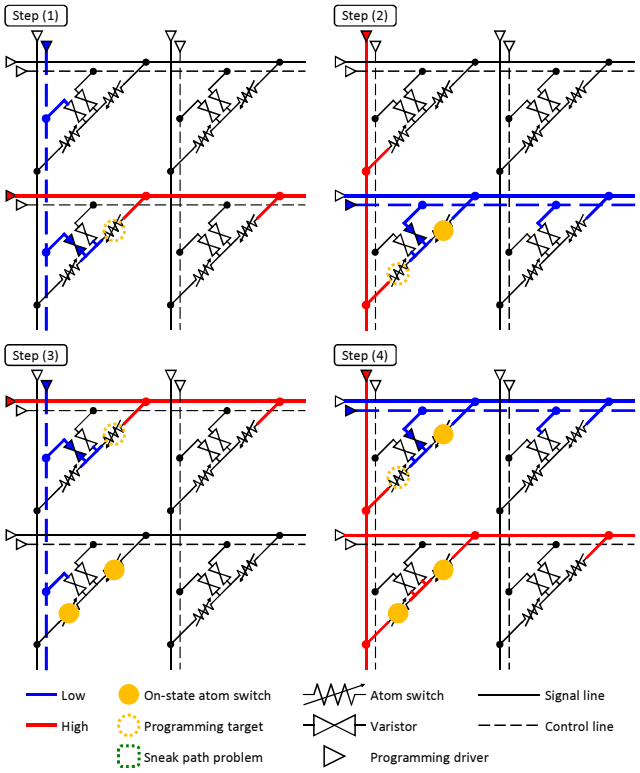

Fig. 5. Crossbar structure with 2V-1CAS via-switch and switch programming steps.

III. SAT Encoding of Sneak Path Problem

A. Basics of SAT

A Boolean satisfiability problem (SAT) is a problem of searching an interpretation that satisfies a given logical expression. When a variable assignment that makes the given expression True exists, this expression is called satisfiable (SAT). On the other hand, if the given expression is False for all the possible variable assignments, this expression is unsatisfiable (UNSAT). In recent years, SAT is widely used in various fields thanks to dramatic performance improvement of SAT solvers. The typical flow of problem-solving using SAT is as follows. First, the original problem is translated into a logical expression, and this step is called SAT encoding. Next, we search the solution of the encoded problem by using an SAT solver. Finally, the solution from the solver is decoded, and then we can obtain the solution of the original problem [13].

B. Verification Methodology of Sneak Path Problem

The sneak path problem in via-switch FPGA changes the on/off-state of multiple via-switches due to the detour of the programming signal. This paper encodes this problem into SAT and mathematically verifies the sneak path problem. The verification flow is as follows. First, we translate the programming operations of via-switch crossbar and occurrence conditions of sneak path problem into Boolean logical expressions. We define logical variables that express the on/off-state of via-switches, the electrical potential of wires, and so on. Then, we encode each circuit operation such as an operation to turn on/off the atom switch with the defined variables. The occurrence conditions of sneak path problem can be encoded into a proposition "multiple conditions for turning on/off an atom switch are satisfied at the same time". The details of SAT encoding are presented in Section III-C. After the SAT encoding, the logical expressions are input to the SAT solver. Finally, by analyzing the output of solver, namely, SAT or UNSAT and Boolean values assigned to each variable, we can evaluate whether the sneak path problem exists and the programming status of the crossbar when the sneak path problem occurs.

C. SAT Encoding of Sneak Path Problem

Here, let us explain how to SAT-encode the programming operations and sneak path problem in via-switch FPGA. Although the following supposes the 2x2 crossbar with 2V-1CAS via-switches, the SAT encoding for crossbars of any size and crossbars with 1V-1CAS via-switches can be derived in the same manner.

First, we define logical variables that represent the status of each element of the crossbar as shown in Figure 6. Atom switch and varistor have two states, on- and off-states, and

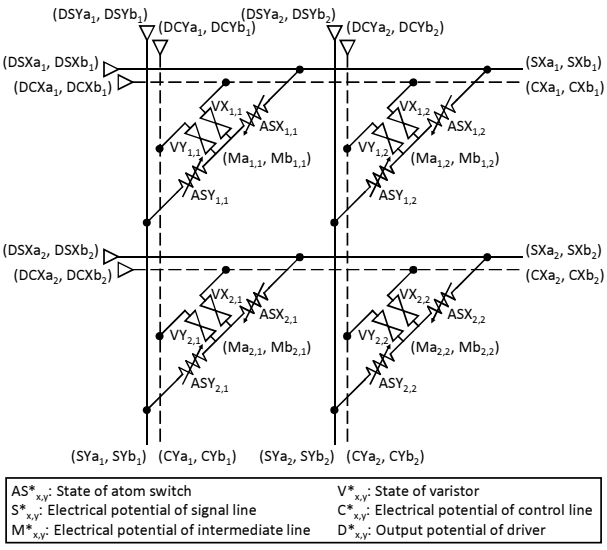

Fig. 6. Definition of Boolean variables in crossbar with 2V-1CAS via-switches.

then their status can be expressed by one variable, where True and False are allocated to on-state and off-state respectively. On the other hand, the electrical potential of each wire and the output voltage of each programming driver have three states, namely, low, high, and floating. To express these three states, we introduce a pair of two variables (a, b), where (False, False), (False, True), and (True, False) correspond to low, high, and floating respectively.

Next, we encode circuit operations of crossbar into logical expressions using the variables defined above. The conditions for turning on/off an atom switch, for example, are represented as Equations 1 and 2. To make Equation 1 True, both sides of the equality operator ($\Leftrightarrow$) need to be False or True. The right side becomes True only when (SXa_1, SXb_1) is (False, True) and ($Ma_{1,1}$, $Mb_{1,1}$) is (False, False), in other words, a high voltage is provided to signal line SX_1 and a low voltage is provided to intermediate line $M_{1,1}$. Also, the left side $ON_{ASX1,1}$ becomes True only when the right side is True. Therefore, $ON_{ASX1,1}$ is a variable that becomes True when a programming voltage for turning on is applied to atom switch $ASX_{1,1}$. Otherwise, it becomes False. Similarly, $OFF_{ASX1,1}$ in Equation 2 represents whether a programming voltage to turn off $ASX_{1,1}$ is applied.

$$ON_{ASX1,1} \Leftrightarrow (\neg SXa_1 \wedge SXb_1) \wedge (\neg Ma_{1,1} \wedge \neg Mb_{1,1}). \quad (1)$$

$$OFF_{ASX1,1} \Leftrightarrow (\neg SXa_1 \wedge \neg SXb_1) \wedge (\neg Ma_{1,1} \wedge Mb_{1,1}). \quad (2)$$

Due to the sneak path problem, the programming voltage is given to multiple switches by the signal detour, and their on/off-states are changed at once. Given the conditions for turning on/off an atom switch expressed by Equations 1 and 2, the sneak path problem can be translated into a proposition "two or more conditions for turning on/off an atom switch are True". When logical expressions presenting this proposition is input to SAT solver and the output is satisfiable, we notice that the sneak path problem occurs. Then, the status of atom switches and electrical potentials can be obtained by checking the assigned Boolean value to each variable. On the other hand, when the solver output is unsatisfiable, there is no sneak path problem.

For investigating countermeasures for the sneak path problem, this paper verifies the crossbar operations under some programming constraints. Banno et al. claim that the sneak path problem in the 2V-1CAS crossbar occurs when multiple on-state via-switches exist in both the same horizontal line and the same vertical line [10]. In the crossbar with 1V-1CAS via-switches, on the other hand, the authors say that the sneak path problem arises when multiple via-switches in either the same horizontal line or the same vertical line are on-state. For rigidly verifying their claims, we introduce the following two programming constraints, namely, row constraint and row & column constraint. The row constraint prohibits configurations where multiple on-state via-switches exist in the same horizontal line, whereas multiple via-switches in the same vertical line are acceptable. On the other hand, under row & column constraint, multiple on-state via-switches in both the same horizontal line and the same vertical line are prohibited. By adding logical expressions of these constraints to solver input, we can verify whether the sneak path problem occurs under the programming constraints.

In addition to the above, we encode the behaviors below of all components in the crossbar, namely, atom switches, varistors, wires, and drivers

- The varistor is turned on when the applied voltage is higher than the threshold.
- The electrical potential of wire is determined by active programming drivers and connection status of signal lines.
- The number of active programming drivers (two in total) and their positions (horizontal, vertical, diagonal directions) are restricted.
- Assigning variables (True, True) to wires and drivers is prohibited.

Among them, determining the potentials of signal lines is complicated since it depends on the connection status of signal lines. The connection status of signal lines is different for each FPGA configuration, and we found that we needed to prepare logical expressions to determine the potentials separately for each configuration. For example, when all the via-switches in a crossbar are off-state, the potential of each signal line can be determined independently since each signal line is not connected to any other signal lines. In other words, each line is driven only if the programming driver that directly connects to the line is active. Otherwise, the signal line is floating. On the other hand, once via-switches are turned on, the potentials of all the signal lines that are connected to each other through on-state switches cannot be determined independently. We

need to check the status of all the drivers that connect to the signal lines of interest and determine the potentials according to the combinations of their status. When all the drivers of the connected signal lines are inactive, the connected signal lines are floating. Otherwise, the potentials of the connected lines are fixed by the active drivers. From the above, when determining the potentials, the position of drivers that we have to check varies depending on the FPGA configuration.

The details of SAT encoding for the varistors and programming drivers are omitted due to space limitations.

IV. Verification Results

A. Effectiveness of SAT Encoding-based Verification

First, we confirm that the proposed SAT encoding-based verification can evaluate the sneak path problem in the via-switch crossbar. For this purpose, we prepare a logical expression that connects all the expressions explained in Section III-C by logical product, and input it to an SAT solver. We use MiniSat [14], which is open source SAT solver. We evaluate both 2V-1CAS and 1V-1CAS crossbars where their size is 2x2. No programming constraint is given in this evaluation.

In both 2V-1CAS and 1V-1CAS crossbars, the output of the solver is satisfiable, which means sneak path problem arises. The analysis of the variable assignment results successfully reveals the circuit status that causes the sneak path problem. Thus, we confirm the detectability of the sneak path problem.

It should be noted that the total number of possible variable assignments that satisfy the input expression is 256 in a 2V-1CAS crossbar and 456 in a 1V-1CAS crossbar, and we analyzed the variable assignments not in all cases but in some of them. Our future works include confirming whether all sneak path problems can be detected and whether the input expression is not satisfied with the circuit status that does not cause the sneak path problem.

B. Countermeasures for Sneak Path Problem

If the proper programming constraint that prohibits all the configurations where sneak path problem occurs is available, we can correctly reconfigure the via-switch FPGA since the programming of any target via-switch without any unexpected programming is ensured. As candidates of such constraints, we verify whether the output of SAT solver varies by imposing the row constraint and row & column constraint described in Section III-C. If the output varies from satisfiable to unsatisfiable, the programming constraint is confirmed to be effective for the sneak path problem.

Verification results are listed in Table I. As previously mentioned, the sneak path problem occurs when the output is satisfiable (SAT), and there is no sneak path problem when the output is unsatisfiable (UNSAT). Although we verify 14 sizes of the crossbar, 2x2-2x7, 3x2-3x4, 4x2-4x3, 5x2, 6x2, and 7x2, Table I summarizes the verification results since the same output is obtained regardless of the crossbar size. We can see that the sneak path problem in a 2V-1CAS crossbar can be avoided by either row constraint or row & column constraint.

TABLE I
Effectiveness of programming constraints.

	No constraint	Row constraint	Row & column constraint
2V-1CAS crossbar	SAT	UNSAT	UNSAT
1V-1CAS crossbar	SAT	SAT	UNSAT

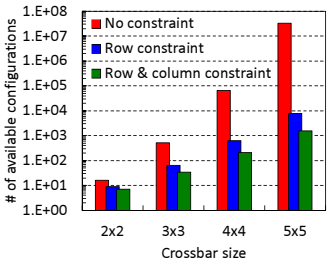

Fig. 7. Number of available configurations under programming constraints.

In a 1V-1CAS crossbar, on the other hand, the row constraint cannot prevent the sneak path problem. To solve the sneak path problem in the 1V-1CAS crossbar, we need to impose the row & column constraint, which is more severe constraint than row constraint.

Next, we evaluate the impact of programming constraints on the number of available configurations. The programming constraints prohibit some configurations of via-switch FPGA, and hence imposing the constraint leads to a decrease in the number of usable configurations. Figure 7 shows the number of available configurations under no constraint, row constraint, and row & column constraint. Here, 2x2, 3x3, 4x4, and 5x5 crossbars are evaluated. We can see that the decrease in the number of usable configurations due to the programming constraints is significant in larger crossbars. For example, compared to the 5x5 crossbar with no constraint, the row constraint and row & column constraint diminish the number of available configurations by three orders of magnitude and four orders of magnitude, respectively.

From the above results, we find that the sneak path problem can be solved by employing the proper programming constraint. Verification results also show that the programming constraints to avoid the sneak path problem are different depending on the structure of via-switch. We need to pay attention to the fact that the programming constraints reduce the total number of available configurations and consequently limit routing flexibility.

C. Impact of Crossbar Size on Verification Time

Finally, we discuss the impact of crossbar size on the verification time. Figure 8 shows the verification time of SAT solver when the crossbar size is varied from 2x2 to 2x7. We evaluate two cases; 2V-1CAS crossbar with no programming constraint where the output is satisfiable, and 2V-1CAS crossbar with row constraint where the output is unsatisfiable. The verification time was measured ten times for each condition, and the average value of them is shown in Figure 8. We executed

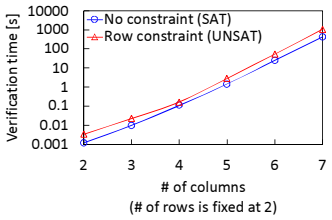

Fig. 8. Verification time of SAT solver when crossbar size is varied.

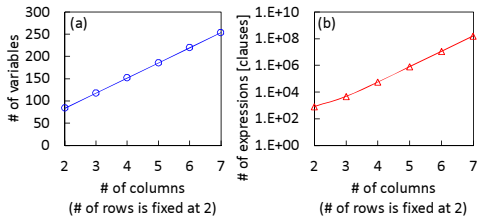

Fig. 9. Number of (a) logical variables and (b) logical expressions when crossbar size is varied.

the SAT solver on Xeon X5680 3330 MHz x2 with 96 GB RAM.

The maximum verification time in this evaluation is 18 minutes, and it is not so long. However, we can see that the verification time exponentially increases as the crossbar size becomes larger. To investigate the cause of this, we evaluate the number of logical variables and expressions when the crossbar size is varied from 2x2 to 2x7. Figure 9 shows the evaluation results. The number of logical variables is proportional to the crossbar size. On the other hand, the number of logical expressions exponentially increases as the crossbar size becomes larger. When the crossbar size is 2x7, the number of logical expressions is over 164 million. This exponential increase in the number of expressions is supposed to cause an exponential increase in the verification time.

The above analysis of the number of logical expressions finds that the exponential increase in the number of expressions originates from the logical expressions to determine the potentials of signal lines explained in Section III-C. Remind that we separately prepare the logical expression to determine the potential for each FPGA configuration. When the number of atom switches in a crossbar is n, the number of possible configurations is 2^n since each atom switch has two states, on- and off-states. Here, n is proportional to the crossbar size. Therefore, the number of logical expressions to determine the potential increases exponentially.

From the above discussion, we conclude that the verification time increases due to the exponential increase in the number of logical expressions and it is not scalable for large crossbars. Future works include the modification of SAT encoding to reduce the increase in the number of expressions.

V. Conclusion

This paper verified the sneak path problem that interfered reconfigurations of via-switch FPGA. We encoded programming operations of via-switch crossbar and occurrence conditions of sneak path problem into SAT and verified them by an SAT solver. We confirmed that the proposed SAT encoding-based verification could detect the sneak path problem. Verification results show that imposing proper programming constraint can eliminate the sneak path problem. Future works include the evaluation of sneak path problem detectability in details and improving the SAT encoding to deal with larger crossbars.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP17J10008 and JST CREST Grant Number JPMJCR1432, Japan.

References

[1] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 203–215, Feb 2007.

[2] M. Lin, A. E. Gamal, Y. C. Lu, and S. Wong, "Performance Benefits of Monolithically Stacked 3-D FPGA," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 216–229, Feb 2007.

[3] P. E. Gaillardon, D. Sacchetto, G. B. Beneventi, M. H. B. Jamaa, L. Perniola, F. Clermidy, I. O'Connor, and G. D. Micheli, "Design and Architectural Assessment of 3-D Resistive Memory Technologies in FPGAs," *IEEE Transactions on Nanotechnology*, vol. 12, no. 1, pp. 40–50, Jan 2013.

[4] S. Tanachutiwat, M. Liu, and W. Wang, "FPGA Based on Integration of CMOS and RRAM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 11, pp. 2023–2032, Nov 2011.

[5] J. Cong and B. Xiao, "FPGA-RPI: A Novel FPGA Architecture With RRAM-Based Programmable Interconnects," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 4, pp. 864–877, April 2014.

[6] X. Tang, P. E. Gaillardon, and G. D. Micheli, "A high-performance low-power near-Vt RRAM-based FPGA," in *2014 International Conference on Field-Programmable Technology (FPT)*, Dec 2014, pp. 207–214.

[7] Y. Y. Liauw, Z. Zhang, W. Kim, A. E. Gamal, and S. S. Wong, "Nonvolatile 3D-FPGA with monolithically stacked RRAM-based configuration memory," in *2012 IEEE International Solid-State Circuits Conference*, Feb 2012, pp. 406–408.

[8] K. Okamoto, M. Tada, T. Sakamoto, M. Miyamura, N. Banno, N. Iguchi, and H. Hada, "Conducting mechanism of atom switch with polymer solid-electrolyte," in *2011 International Electron Devices Meeting*, Dec 2011, pp. 12.2.1–12.2.4.

[9] M. Miyamura, T. Sakamoto, M. Tada, N. Banno, K. Okamoto, N. Iguchi, and H. Hada, "Low-power programmable-logic cell arrays using nonvolatile complementary atom switch," in *Fifteenth International Symposium on Quality Electronic Design*, March 2014, pp. 330–334.

[10] N. Banno, M. Tada, K. Okamoto, N. Iguchi, T. Sakamoto, M. Miyamura, Y. Tsuji, H. Hada, H. Ochi, H. Onodera, M. Hashimoto, and T. Sugibayashi, "A novel two-varistors (a-Si/SiN/a-Si) selected complementary atom switch (2V-1CAS) for nonvolatile crossbar switch with multiple fan-outs," in *2015 IEEE International Electron Devices Meeting (IEDM)*, Dec 2015, pp. 2.5.1–2.5.4.

[11] N. Banno, M. Tilda, K. Okamoto, N. Iguchi, T. Sakamoto, H. Hada, H. Ochi, H. Onodera, M. Hashimoto, and T. Sugibayashi, "50x20 crossbar switch block (CSB) with two-varistors (a-Si/SiN/a-Si) selected complementary atom switch for a highly-dense reconfigurable logic," in *2016 IEEE International Electron Devices Meeting (IEDM)*, Dec 2016, pp. 16.4.1–16.4.4.

[12] J. Hotate, T. Kishimoto, T. Higashi, H. Ochi, R. Doi, M. Tada, T. Sugibayashi, K. Wakabayashi, H. Onodera, Y. Mitsuyama, and M. Hashimoto, "A highly-dense mixed grained reconfigurable architecture with overlay crossbar interconnect using via-switch," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2016, pp. 1–4.

[13] D. E. Knuth, *The Art of Computer Programming, Volume 4, Fascicle 6: Satisfiability*, 1st ed. Addison-Wesley Professional, 2015.

[14] "MiniSat," http://minisat.se/.