13th IEEE Workshop on RTL and High Level Testing (WRTLT'12), pp.1.2.1-1.2.7, Nov. 2012.

WAGSR: Web Application for Generalized Feed Forward Shift Registers

Katsuya Fujiwara[1] and Hideo Fujiwara[2]

[1] Faculty of Engineering and Resource Science
Akita University
Akita, 010-8502, JAPAN
fujiwara@ie.akita-u.ac.jp

[2] Faculty of Informatics
Osaka Gakuin University
Osaka, 564-8511, JAPAN
fujiwara@ogu.ac.jp

***Abstract*—In this paper, we introduce generalized feed-forward shift registers (GF^2SR, for short) to apply them to secure and testable scan design. Previously, we introduced SR-equivalents and SR-quasi-equivalents which can be used in secure and testable scan design, and showed inversion-inserted linear feed-forward shift registers (I^2LF^2SR, for short) are useful circuits for the secure and testable scan design. GF^2SR is an extension of I^2LF^2SR and the class is much wider than that of I^2LF^2SR. Since the cardinality of the class of GF^2SR is much larger than that of I^2LF^2SR, the security level of scan design with GF^2SR is much higher than that of I^2LF^2SR. We consider how to control/observe GF^2SR to guarantee easy scan-in/out operations, i.e., state-justification and state-identification problems are considered. A program called WAGSR (Web Application for Generalized feed-forward Shift Registers) is presented to solve those problems.**

Keywords – design-for-testability; scan design; shift register equivalents; shift register quasi-equivalents; generalized feed-forward shift registers; security; scan-based side-channel attack.

1. INTRODUCTION

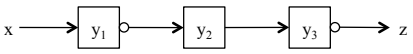
The design of secure chips demands protection of secret information, which may cause conflicts with the requirements for making the chip easily testable. While testing techniques such as scan design entail increased testability (controllability and observability) of the chip, they can also allow access to important data in a secure chip a lot easier. This makes it difficult for scan chains to be used especially in special cryptographic circuits where secret key streams are stored in internal registers, thus a problem in testing these types of circuits is imminent. However, quality of these circuits is highly in demand currently due to increase in the need of secure systems [3]. Fundamentally, the problem lies on the inherent contradiction between testability and security for digital circuits. Hence, there's a need for an efficient solution such that both testability and security are satisfied.

To solve this challenging problem, different approaches have been proposed [4]-[13]. All the approaches except [13] add extra hardware outside of the scan chain. Disadvantages of this are high area overhead, timing overhead or performance degradation, increased complexity of testing, and limited security for the registers part among others. To resolve those disadvantages, we have reported another secure and testable scan design approach by using extended shift registers called "*SR-equivalents*" that are functionally equivalent but not structurally equivalent to shift registers [14]-[17] and "*SR-quasi-equivalents*" [18]. The proposed approach is only to replace part of the original scan chains to SR-equivalents or SR-quasi-equivalents, which satisfy both testability and security of digital circuits. This method requires very little area overhead and no performance overhead. Moreover, no additional keys and controller circuits outside of the scan chain are needed, thus making the scheme low-cost and efficient. We showed *inversion-inserted linear feed-forward shift registers* (I^2LF^2SR, for short) are useful circuits for the secure and testable scan design.

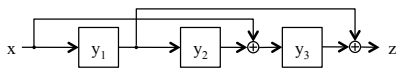
In this paper, we introduce a new class of extended shift registers called *generalized feed-forward shift registers* (GF^2SR, for short) by relaxing the condition of the SR-equivalents and SR-quasi-equivalents. GF^2SR is an extension of I^2LF^2SR and the class is much wider than that of I^2LF^2SR. Since the cardinality of the class of GF^2SR is much larger than that of I^2LF^2SR, the security level of scan design with GF^2SR is much higher than that of I^2LF^2SR. We consider how to control/observe GF^2SR to guarantee easy scan-in/out operations, i.e., state-justification and state-identification problems are considered. A program called WAGSR (Web Application for Generalized feed-forward Shift Registers) is presented to solve those problems.

2. EXTENDED SHIFT REGISTERS

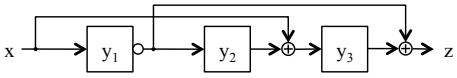
In our previous works [14]-[18], we introduced extended shift registers to organize secure and testable scan design. Figure 1 shows those circuits realized by a linear feed-forward shift register and/or by inserting inverters; inversion-inserted SR (I^2SR), linear feed-forward SR (LF^2SR) and inversion-inserted linear feed-forward SR (I^2LF^2SR).



(a) Inversion-inserted SR (I^2SR)

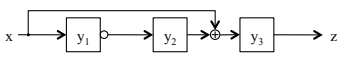


(b) Linear feed-forward SR (LF^2SR)



(c) Inversion-inserted linear feed-forward SR (I^2LF^2SR)

Figure1. Three types of extended shift registers



(a) I^2LF^2SR, R_1

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$	$1\oplus x(t+1)$	$x(t+2)\oplus 1\oplus x(t)$	$z(t+3)=x(t+2)\oplus 1\oplus x(t)$

(b) Symbolic simulation

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$ $=y_1(t+3)$	$1\oplus x(t+1)$ $=y_2(t+3)$	$x(t+2)\oplus 1\oplus x(t)$ $=y_3(t+3)$	$z(t+3)=x(t+2)\oplus 1\oplus x(t)$

$x(t) = 1\oplus y_1(t+3)\oplus y_3(t+3)$
$x(t+1) = 1\oplus y_2(t+3)$
$x(t+2) = y_1(t+3)$

(c) Equations for state-justification

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t)=y_3(t)$
$x(t+1)$	$x(t)$	$1\oplus y_1(t)$	$x(t)\oplus y_2(t)$	$z(t+1)=x(t)\oplus y_2(t)$
$x(t+2)$	$x(t+1)$	$1\oplus x(t)$	$x(t+1)\oplus 1\oplus y_1(t)$	$z(t+2)=x(t+1)\oplus 1\oplus y_1(t)$
$x(t+3)$	$x(t+2)$	$1\oplus x(t+1)$	$x(t+2)\oplus 1\oplus x(t)$	$z(t+3)=x(t+2)\oplus 1\oplus x(t)$

$y_1(t) = z(t+2)\oplus x(t+1)\oplus 1$
$y_2(t) = z(t+1)\oplus x(t)$
$y_3(t) = z(t)$

(d) Equations for state-identification

Figure 2. Example of I^2LF^2SR, R_1

Consider a 3-stage I^2LF^2SR, R_1, given in Figure 2(a). By using symbolic simulation, we can obtain an output sequence (z(t), z(t+1), z(t+2), z(t+3)) and the output $z(t+3)=x(t)\oplus 1\oplus x(t+2)$ as shown in Figure 2(b). So, we can see the input value applied to x at any time t appears at output z after 3 clock cycles with exclusive-OR of some inputs and/or constant 1. By using symbolic simulation, we can derive equations to obtain an input sequence (x(t), x(t+1), x(t+2)) that transfers R_1 from any state to the desired final state ($y_1(t+3)$, $y_2(t+3)$, $y_3(t+3)$) as illustrated in Figure 2(c). Similarly, as illustrated in Figure 2(d), we can derive equations to determine uniquely the initial state ($y_1(t)$, $y_2(t)$, $y_3(t)$) from the input/output sequence.

More generally, for any circuit C of I^2SR, LF^2SR, and I^2LF^2SR with k flip-flops, the input value applied to input x at any time t appears at output z after k clock cycles with exclusive-OR of some inputs and/or constant 1, i.e.,

$$z(t+k) = x(t) \oplus c_0 \oplus c_1x(t+1) \oplus c_2x(t+2) \oplus \ldots \oplus c_kx(t+k)$$

where c_0 , c_1, c_2, … , c_k are 0 or 1. The ordered set of coefficients (c_0 , c_1, c_2, … , c_k) is called the *characteristic coefficient* of the circuit C.

Further, generally as for any circuit C of I^2SR, LF^2SR, and I^2LF^2SR with k flip-flops, (1) for any internal state of C a transfer sequence (of length k) to the state (final state) can be generated only from the connection information of C, independently of the initial state; (2) any present state (initial state) of C can be identified from the input-output sequence (of length k) and the connection information of C, where k is the number of flip-flops.

Here, let us extend the class of I^2LF^2SR by relaxing linear functions in the above equation to arbitrary logic functions, i.e., the input value applied to x at any time t appears at z after k clock cycles with exclusive-OR of some logic function f of x(t+1), x(t+2), …, x(t+k), as follows.

$$z(t+k) = x(t) \oplus f(x(t+1), x(t+2), \ldots, x(t+k)).$$

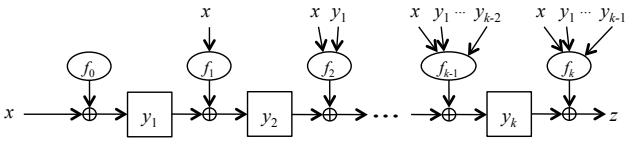
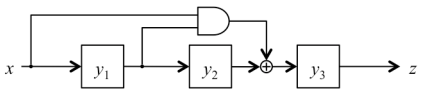


Figure 3. Generalized feed-forward shift register (GF^2SR)

A circuit of the structure shown in Figure 3 is called a *generalized feed-forward shift register (GF^2SR)*. In this figure, f_0, f_1, …, f_k are arbitrary logic functions of input x and state variables y_i of preceding stages. f_0 is a constant function, f_1 is a function of x, f_2 is a function of x and y_1, and f_i is a function of x, y_1, y_2, ..., y_{i-1}. It can be shown that, for any GF^2SR with k flip-flops, the output z at time t+k behaves in accordance with the above equation.

By using symbolic simulation, we can obtain an output sequence (z(t), z(t+1), z(t+2), z(t+3)) and the output $z(t+3)=x(t)\oplus x(t+2)x(t+1)$ as shown in Figure 4(b). From the result of symbolic simulation, we can derive equations to obtain an input sequence (x(t), x(t+1), x(t+2)) that transfers R_2 from any state to the desired final state ($y_1(t+3)$, $y_2(t+3)$, $y_3(t+3)$) as illustrated in Figure 4(b). Similarly, as illustrated in Figure 4(b), we can derive equations to determine uniquely



(a) GF^{2}SR, R_2

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t) = y_3(t)$
$x(t+1)$	$x(t)$	$y_1(t)$	$y_2(t) \oplus x(t) \cdot y_1(t)$	$z(t+1) = y_2(t) \oplus x(t) \cdot y_1(t)$
$x(t+2)$	$x(t+1)$	$x(t)$	$y_1(t) \oplus x(t+1) \cdot x(t)$	$z(t+2) = y_1(t) \oplus x(t+1) \cdot x(t)$
$x(t+3)$	$x(t+2)$ $= y_1(t+3)$	$x(t+1)$ $= y_2(t+3)$	$x(t) \oplus x(t+2) \cdot x(t+1)$ $= y_3(t+3)$	$z(t+3) = x(t) \oplus x(t+2) \cdot x(t+1)$

$$x(t) = y_3(t+3) \oplus y_2(t+3) \cdot y_1(t+3)$$
$$x(t+1) = y_2(t+3)$$
$$x(t+2) = y_1(t+3)$$

$$y_1(t) = z(t+2) \oplus x(t+1) \cdot x(t)$$
$$y_2(t) = z(t+1) \oplus x(t) \cdot y_1(t)$$
$$= z(t+1) \oplus x(t) \cdot z(t+2) \oplus x(t+1) \cdot x(t)$$
$$y_3(t) = z(t)$$

(b) Symbolic simulation

Figure 4. Example of GF^{2}SR, R_2

the initial state ($y_1(t)$, $y_2(t)$, $y_3(t)$) from the input/output sequence.

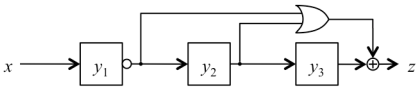
3. How to Control/Observe GF^{2}SR

For an extended shift register, the following two problems are important in order to utilize the extended shift register as a scan shift register in testing. One problem is to generate an input sequence to transfer the circuit into a given desired state. This is called *state-justification problem*. The other problem is to determine the initial state by observing the output sequence from the state. This is called *state-identification problem.*

We have shown in the previous section that, for I^{2}LF^{2}SR, R_1, and GF^{2}SR, R_2, we can derive equations to obtain an input sequence that transfers R_1 and R_2 from any state to the desired final state as illustrated in Figure 2(c) and Figure 4(b), respectively. Similarly, as illustrated in Figure 2(d) and Figure 4(b), we can derive equations to determine uniquely the initial state from the input/output sequence.

This holds for any circuit C in the class of I^{2}LF^{2}SR and GF^{2}SR, i.e., (1) for any internal state of C a transfer sequence (of length k) to the state (final state) can be generated only from the connection information of C, independently of the initial state; (2) any present state (initial state) of C can be identified from the input-output sequence (of length k) and the connection information of C, where k is the number of flip-flops.

Consider three different structured 3-stage GF^{2}SRs, R_2, R_3 and R_4, shown in Figures 4, 5 and 6 From the results of symbolic simulation, we can see their outputs z(t+3) are the same, i.e.,



(a) GF^{2}SR, R_3

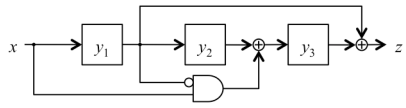
x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t) = y_3(t) \oplus (\overline{y_1(t)} + y_2(t))$
$x(t+1)$	$x(t)$	$\overline{y_1(t)}$	$y_2(t)$	$z(t+1) = y_2(t) \oplus (\overline{x(t)} + \overline{y_1(t)})$
$x(t+2)$	$x(t+1)$	$\overline{x(t)}$	$\overline{y_1(t)}$	$z(t+2) = y_1(t) \oplus (\overline{x(t+1)} + \overline{x(t)})$
$x(t+3)$	$x(t+2)$ $= y_1(t+3)$	$\overline{x(t+1)}$ $= y_2(t+3)$	$\overline{x(t)}$ $= y_3(t+3)$	$z(t+3) = \overline{x(t)} \oplus (\overline{x(t+2)} + \overline{x(t+1)})$ $= x(t) \oplus (x(t+2) \cdot x(t+1))$

$$x(t) = \overline{y_3(t+3)}$$
$$x(t+1) = \overline{y_2(t+3)}$$
$$x(t+2) = y_1(t+3)$$

$$y_1(t) = z(t+2) \oplus x(t+1) \cdot x(t)$$
$$y_2(t) = \overline{z(t+1)} \oplus x(t) \cdot y_1(t)$$
$$= \overline{z(t+1)} \oplus x(t) \cdot z(t+2) \oplus x(t+1) \cdot x(t)$$
$$y_3(t) = z(t) \oplus \left(\overline{y_1(t)} + y_2(t)\right)$$
$$= \overline{z(t)} \oplus y_1(t) \cdot \overline{y_2(t)}$$
$$= \overline{z(t)} \oplus z(t+2) \cdot z(t+1) \oplus x(t) \cdot z(t+2) \oplus x(t+1) \cdot x(t) \cdot \overline{z(t+1)}$$

(b) Symbolic simulation

Figure 5. Example of another GF^{2}SR, R_3



(a) GF^{2}SR, R_4

x	y_1	y_2	y_3	z
$x(t)$	$y_1(t)$	$y_2(t)$	$y_3(t)$	$z(t) = y_3(t) \oplus y_1(t)$
$x(t+1)$	$x(t)$	$y_1(t)$	$y_2(t) \oplus x(t) \cdot \overline{y_1(t)}$	$z(t+1) = y_2(t) \oplus x(t) \cdot y_1(t)$
$x(t+2)$	$x(t+1)$	$x(t)$	$y_1(t) \oplus x(t+1) \cdot \overline{x(t)}$	$z(t+2) = y_1(t) \oplus x(t+1) \cdot x(t$
$x(t+3)$	$x(t+2)$ $= y_1(t+3)$	$x(t+1)$ $= y_2(t+3)$	$\overline{x(t)} \oplus x(t+2) \cdot \overline{x(t+1)}$ $= y_3(t+3)$	$z(t+3) = x(t) \oplus x(t+2) \cdot x(t$

$$x(t) = y_3(t+3) \oplus x(t+2) \cdot \overline{x(t+1)}$$
$$x(t+1) = y_2(t+3)$$
$$x(t+2) = y_1(t+3)$$

$$y_1(t) = z(t+2) \oplus x(t+1) \cdot x(t)$$
$$y_2(t) = z(t+1) \oplus x(t) \cdot y_1(t)$$
$$= z(t+1) \oplus x(t) \cdot z(t+2) \oplus x(t+1) \cdot$$
$$y_3(t) = z(t) \oplus y_1(t)$$
$$= z(t) \oplus z(t+2) \oplus x(t+1) \cdot x(t)$$

(b) Symbolic simulation

Figure 6. Example of another GF^{2}SR, R_4

$z(t+3)=x(t) \oplus x(t+2)x(t+1)$. Therefore, their input/output behaviors after time t+3 are the same. However, their internal state behaviors are not the same. Therefore, one cannot control/observe internal states unless one knows the structure of the circuit, and hence they are secure. In the following section 4, we shall consider the security level by clarifying the cardinality of the class of GF^2SR's.

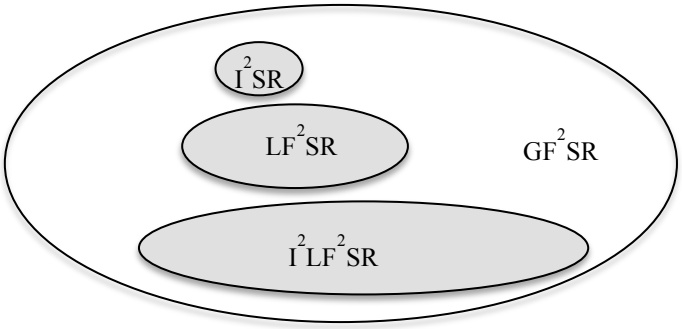


Figure 7. Cover relation among classes

4. Cardinality of Each Class of Extended SRs

When we consider a secure scan design, we need to assume what the attacker knows and how he can potentially make the attack. Here, we assume that the attacker does not know the detailed information in the gate-level design, and that the attacker knows the presence of test pins (scan in/out, scan, and reset) and modified scan chains. However, he does not know the structure of modified scan chains.

Based on the above assumption, we consider the security to prevent scan-based attacks.

The security level of the secure scan architecture based on GF^2SR is determined by the probability that an attacker can identify the structure of the GF^2SR used in the scan design, and hence the attack probability approximates to the reciprocal of the cardinality of the class of GF^2SR.

In [15, 18] we showed the cardinality of each class of linear structured circuits (I^2SR, LF^2SR, and I^2LF^2SR) which is summarized in Table I. Obviously, the class of GF^2SR covers I^2SR, LF^2SR, and I^2LF^2SR. So, we have the covering relation as shown in Figure 7.

Let us calculate the number of circuits in the class of GF^2SR. Let $f_0, f_1, \ldots, f_k$ be the functions shown in Figure 3. The number of functions for each $f_0, f_1, \ldots$, and f_k are $2^{2^0} = 2$, $2^{2^1} = 4$, …, and 2^{2^k}, respectively. Hence the total number of k-stage GF^2SR is $2 \times 4 \times \cdots \times 2^{2^k} = 2^{(2^{(k+1)}-1)}$. The summary of the cardinality of each class is shown in Table I. From this table, we can see the cardinality of GF^2SR is much larger than that of I^2LF^2SR, and hence very secure. For any GF^2SR, the state-justification and state-identification problems can be easily solved, and hence we can use any of them to organize the secure and testable scan circuits.

Table I. Cardinality of each class

CLASS	# OF CIRCUITS IN THE CLASS
I^2SR	$2^{k+1} - 1$
LF^2SR	$2^{k(k+1)/2} - 1$
I^2LF^2SR	$(2^{k(k+1)/2} - 1)(2^{k+1} - 1)$
GF^2SR	$2^{(2^{k+1}-1)}$

5. Program WAGSR

WAGSR (Web Application for Generalized feed forward Shift Registers) is a web application program to compute/solve various problems on GF^2SR by symbolic and logic simulation as follows.

1. Design of GF^2SR by means of logic expression
2. Illustration of GF^2SR
3. Computation for GF^2SR to solve state-justification and state-identification problems
   - Symbolic simulation
   - Logic simulation by partially specifying values 0,1, and/or X to input/output sequence, initial state, and/or final state.

WAGSR adopts GUI (graphical user interface) for expressing outcome by circuit diagram and table. SR-ID code is introduced to represent the structure of each type of extended shift register uniquely. In Appendix, some examples of the outcome by WAGSR are presented. For example, from Figure 11, we obtain an input sequence to transfer the circuit to all 1's state independently of the initial state. In Figure 12, we can identify the initial state from the input/output sequence.

6. Conclusion

In our previous work, we reported a secure and testable scan design approach by using extended shift registers called *SR-equivalents* [14-17] and *SR-quasi-equivalents* [18], where the class of I^2LF^2SR is one of the most useful class. In this paper, we introduced a further extended class of *generalized feed-forward shift registers* (GF^2SR). GF^2SR is an extension of I^2LF^2SR and the class is much wider than that of I^2LF^2SR. Since the cardinality of the class of GF^2SR is much larger than that of I^2LF^2SR, the security level of scan design with GF^2SR is much higher than that of I^2LF^2SR. We considered how to control/observe GF^2SR to guarantee easy scan-in/out operations, i.e., state-justification and state-identification problems were considered. A program called WAGSR (Web Application for Generalized feed-forward Shift Registers) that solves those problems was introduced.

References

[1] H. Fujiwara, Y. Nagao, T. Sasao, and K. Kinoshita, "Easily testable sequential machines with extra inputs," *IEEE Trans. on Computers,* Vol. -24, No, 8, pp. 821-826, Aug. 1973,

[2] H. Fujiwara, *Logic Testing and Design for Testability*, *The MIT Press* 1985.

[3] K. Hafner, H. Ritter, T. Schwair, S. Wallstab, M. Deppermann, J. Gessner, S. Koesters, W. Moeller, and G. Sandweg, "Design and test of an integrated cryptochip," *IEEE Design and Test of Computers*, pp. 6-17, Dec. 1999.

[4] D. Hely, M.-L. Flottes, F. Bancel, B. Rouzeyre, and N. Berard. "Scan design and secure chip." *10th IEEE International On-Line Testing Symposium,* pp. 219–224, 2004.

[5] D. Hely, F. Bancel, M.L. Flottes, and B. Rouzeyre. "Securing scan control in crypto chips." *Journal of Electronic Testing - Theory and Applications*, Vol. 23, No. 5, pp. 457–464, Oct. 2007.

[6] B. Yang, K. Wu, and R. Karri. "Scan based side channel attack on dedicated hardware implementations of data encryption standard." *International Test Conference 2004*, pp. 339–344, 2004.

[7] B. Yang, K. Wu, and R. Karri. "Secure scan: A design-for-test architecture for crypto chips." *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No.10, pp. 2287–2293, Oct. 2006.

[8] J. Lee, M. Tehranipoor, and J. Plusquellic, "A low-cost solution for protecting IPs against scan-based side-channel attacks," *24th IEEE VLSI Test Symposium,* pp. 94 - 99, 2006.

[9] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic. "Securing designs against scan-based side-channel attacks." *IEEE Trans. on Dependable and Secure Computing*, Vol. 4, No. 4, pp. 325–336, Oct.-Dec. 2007.

[10] S. Paul, R. S. Chakraborty, and S. Bhunia. "VIm-Scan: A low overhead scan design approach for protection of secret key inscan-based secure chips." *25th IEEE VLSI Test Symposium*, pp. 455–460, 2007.

[11] M. Inoue, T. Yoneda, M. Hasegawa, and H. Fujiwara, "Partial scan approach for secret information protection," *14th IEEE European Test Symposium*, pp. 143-148, May 2009.

[12] U. Chandran and D. Zhao, "SS-KTC: A high-testability low-overhead scan architecture with multi-level security integration," *27th IEEE VLSI Test Symposium,* pp. 321-326, May 2009.

[13] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury. "Secured flipped scan-chain model for crypto-architecture." *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vo. 26, No.11, pp. 2080-2084, November 2007.

[14] H. Fujiwara and M.E.J. Obien, "Secure and testable scan design using extended de Brujin graph," *15th Asia and South Pacific Design Automation Conference*, pp. 413-418, Jan. 2010.

[15] K. Fujiwara, H. Fujiwara, M.E.J. Obien, and H. Tamamoto, "SREEP: Shift register equivalents enumeration and synthesis program for secure scan design," *13th IEEE International Symposium on Design and Diagnosis of Electronic Circuits and Systems,* pp. 193-196, April 2010.

[16] K. Fujiwara, H. Fujiwara, and H. Tamamoto, "SREEP-2: SR-equivalent Generator for Secure and Testable Scan Design," *11th IEEE Workshop on RTL and High Level Testing,* pp. 7-12, Dec. 2010.

[17] H. Fujiwara, K. Fujiwara, and H. Tamamoto, "Secure Scan Design Using Shift Register Equivalents against Differential Behavior Attack," *16th Asia and South Pacific Design Automation Conference*, pp.818-823, Jan. 2011.

[18] K. Fujiwara, H. Fujiwara, and H. Tamamoto,"SR-Quasi-Equivalents: Yet Another Approach to Secure and Testable Scan Design," *12th IEEE Workshop on RTL and High Level Testing*, pp. 77-82, Dec. 2011.

[19] SREEP: http://sreep.fujiwaralab.net/

[20] WAGSR: http://wagsr.fujiwaralab.net/

APPENDIX

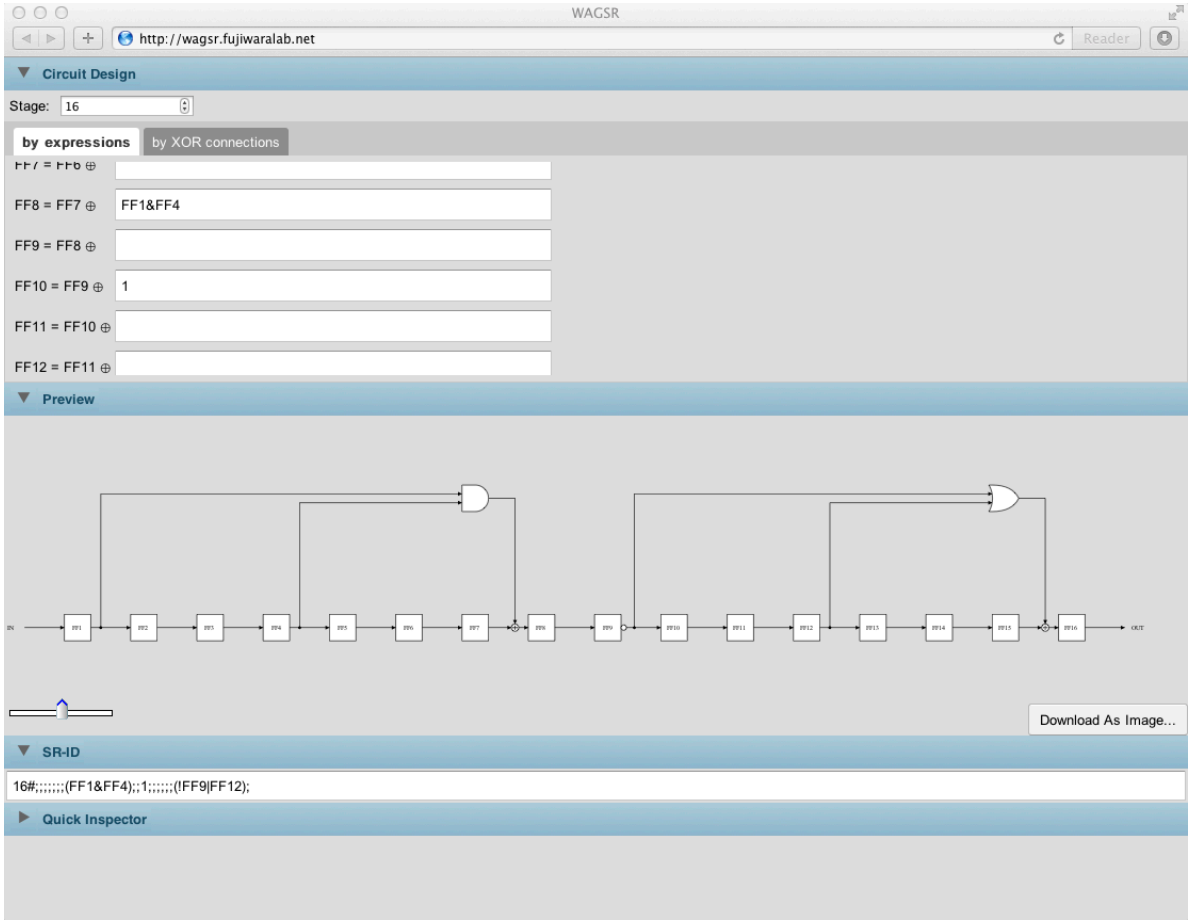
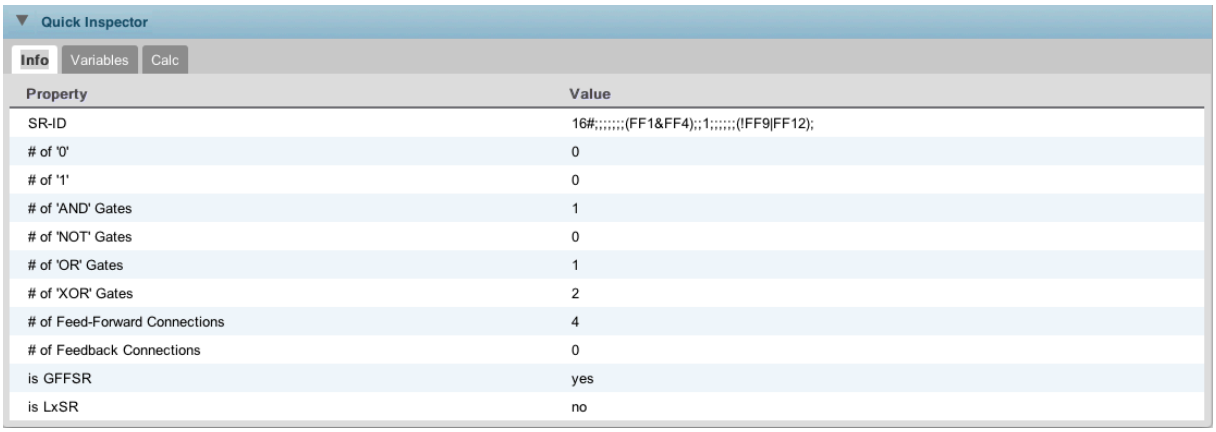


Figure 8. Design of GF^{2}SR by means of logic expression

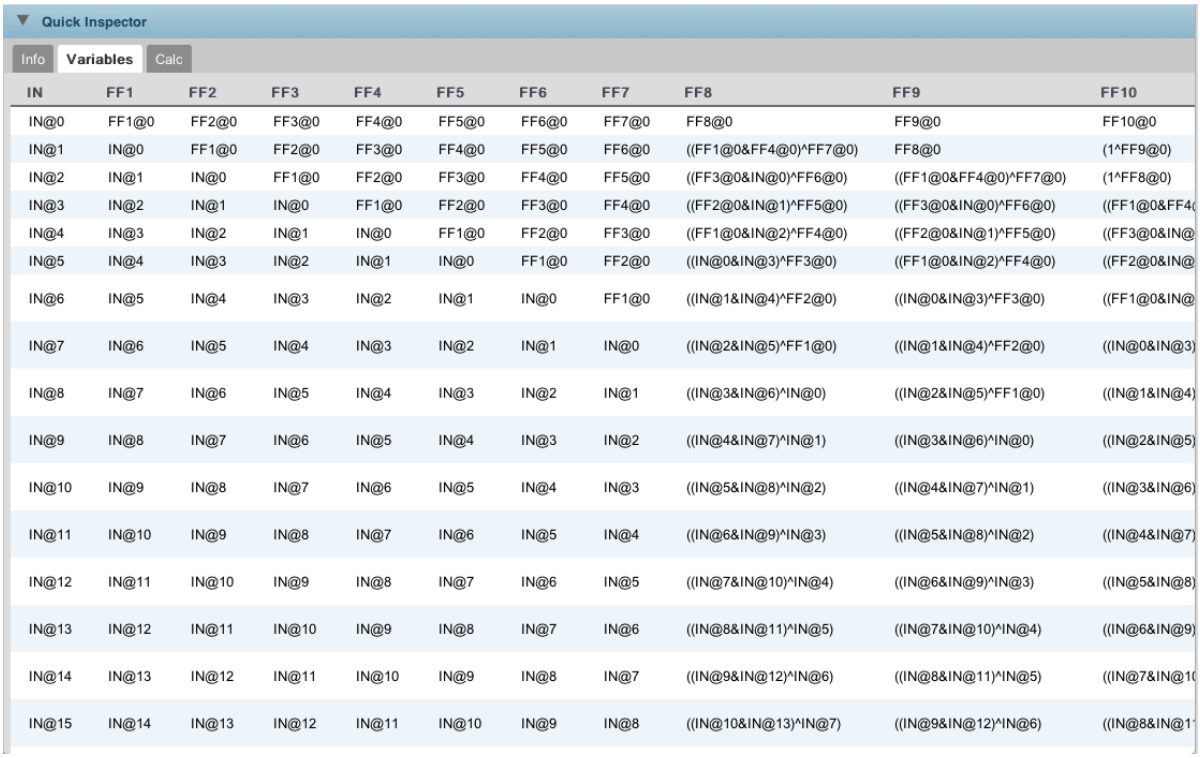


Quick Inspector

Info | Variables | Calc

Property	Value
SR-ID	16#;;;;;;;(FF1&FF4);;1;;;;;;(!FF9|FF12);
# of '0'	0
# of '1'	0
# of 'AND' Gates	1
# of 'NOT' Gates	0
# of 'OR' Gates	1
# of 'XOR' Gates	2
# of Feed-Forward Connections	4
# of Feedback Connections	0
is GFFSR	yes
is LxSR	no

Figure 9. Structural information of GF^{2}SR



Quick Inspector

Info | Variables | Calc

IN	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10
IN@0	FF1@0	FF2@0	FF3@0	FF4@0	FF5@0	FF6@0	FF7@0	FF8@0	FF9@0	FF10@0
IN@1	IN@0	FF1@0	FF2@0	FF3@0	FF4@0	FF5@0	FF6@0	((FF1@0&FF4@0)^FF7@0)	FF8@0	(1^FF9@0)
IN@2	IN@1	IN@0	FF1@0	FF2@0	FF3@0	FF4@0	FF5@0	((FF3@0&IN@0)^FF6@0)	((FF1@0&FF4@0)^FF7@0)	(1^FF8@0)
IN@3	IN@2	IN@1	IN@0	FF1@0	FF2@0	FF3@0	FF4@0	((FF2@0&IN@1)^FF5@0)	((FF3@0&IN@0)^FF6@0)	((FF1@0&FF4(
IN@4	IN@3	IN@2	IN@1	IN@0	FF1@0	FF2@0	FF3@0	((FF1@0&IN@2)^FF4@0)	((FF2@0&IN@1)^FF5@0)	((FF3@0&IN@
IN@5	IN@4	IN@3	IN@2	IN@1	IN@0	FF1@0	FF2@0	((IN@0&IN@3)^FF3@0)	((FF1@0&IN@2)^FF4@0)	((FF2@0&IN@
IN@6	IN@5	IN@4	IN@3	IN@2	IN@1	IN@0	FF1@0	((IN@1&IN@4)^FF2@0)	((IN@0&IN@3)^FF3@0)	((FF1@0&IN@
IN@7	IN@6	IN@5	IN@4	IN@3	IN@2	IN@1	IN@0	((IN@2&IN@5)^FF1@0)	((IN@1&IN@4)^FF2@0)	((IN@0&IN@3)
IN@8	IN@7	IN@6	IN@5	IN@4	IN@3	IN@2	IN@1	((IN@3&IN@6)^IN@0)	((IN@2&IN@5)^FF1@0)	((IN@1&IN@4)
IN@9	IN@8	IN@7	IN@6	IN@5	IN@4	IN@3	IN@2	((IN@4&IN@7)^IN@1)	((IN@3&IN@6)^IN@0)	((IN@2&IN@5)
IN@10	IN@9	IN@8	IN@7	IN@6	IN@5	IN@4	IN@3	((IN@5&IN@8)^IN@2)	((IN@4&IN@7)^IN@1)	((IN@3&IN@6)
IN@11	IN@10	IN@9	IN@8	IN@7	IN@6	IN@5	IN@4	((IN@6&IN@9)^IN@3)	((IN@5&IN@8)^IN@2)	((IN@4&IN@7)
IN@12	IN@11	IN@10	IN@9	IN@8	IN@7	IN@6	IN@5	((IN@7&IN@10)^IN@4)	((IN@6&IN@9)^IN@3)	((IN@5&IN@8)
IN@13	IN@12	IN@11	IN@10	IN@9	IN@8	IN@7	IN@6	((IN@8&IN@11)^IN@5)	((IN@7&IN@10)^IN@4)	((IN@6&IN@9)
IN@14	IN@13	IN@12	IN@11	IN@10	IN@9	IN@8	IN@7	((IN@9&IN@12)^IN@6)	((IN@8&IN@11)^IN@5)	((IN@7&IN@1(
IN@15	IN@14	IN@13	IN@12	IN@11	IN@10	IN@9	IN@8	((IN@10&IN@13)^IN@7)	((IN@9&IN@12)^IN@6)	((IN@8&IN@1'

Figure 10. Symbolic simulation

Quick Inspector

Info | Variables | Calc

Input Sequence:

Output Sequence:

FF Initial States:

FF Final States: 1111111111111111

IN	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13	FF14	FF15	FF16	OUT
0																	OUT@0=
0	0																OUT@1=
0	0	0															OUT@2=
1	0	0	0														OUT@3=
0	1	0	0	0													OUT@4=
0	0	1	0	0	0												OUT@5=
1	0	0	1	0	0	0											OUT@6=
0	1	0	0	1	0	0	0										OUT@7=
0	0	1	0	0	1	0	0	1									OUT@8=
1	0	0	1	0	0	1	0	0	1								OUT@9=
1	1	0	0	1	0	0	1	0	0	0							OUT@10=
1	1	1	0	0	1	0	0	0	0	1	0						OUT@11=
1	1	1	1	0	0	1	0	0	0	1	1	0					OUT@12=
1	1	1	1	1	0	0	1	0	0	1	1	1	0				OUT@13=
1	1	1	1	1	1	0	0	0	0	1	1	1	1	0			OUT@14=
1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0		OUT@15=
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	OUT@16=1

Figure 11. State-justification

Info | Variables | Calc

Input Sequence: 0001001001111111

Output Sequence: 0111111000000000

FF Initial States:

FF Final States:

IN	FF1	FF2	FF3	FF4	FF5	FF6	FF7	FF8	FF9	FF10	FF11	FF12	FF13	FF14	FF15	FF16	OUT
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OUT@0=0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	OUT@1=1
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	OUT@2=1
1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	1	OUT@3=1
0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	OUT@4=1
0	0	1	0	0	0	0	0	0	0	1	1	1	1	1	0	1	OUT@5=1
1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	OUT@6=1
0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	OUT@7=0
0	0	1	0	0	1	0	0	1	0	1	1	1	1	1	1	0	OUT@8=0
1	0	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	OUT@9=0
1	1	0	0	1	0	0	1	0	0	0	1	1	1	1	1	0	OUT@10=0
1	1	1	0	0	1	0	0	0	0	1	0	1	1	1	1	0	OUT@11=0
1	1	1	1	0	0	1	0	0	0	1	1	0	1	1	1	0	OUT@12=0
1	1	1	1	1	0	0	1	0	0	1	1	1	0	1	1	0	OUT@13=0
1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	1	0	OUT@14=0
1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	0	0	OUT@15=0
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	OUT@16=1

Figure 12. State-identification