A Hardware Random Number Generator

Thomas E. Tkacik

Motorola
2900 South Diablo Way
Tempe, AZ 85282
Tom.Tkacik@motorola.com

Abstract. Some of the desirable properties a cryptographic random number generator should have are lack of bias, bit independence, unpredictiability and nonrepeatability. In this paper, we discuss how a hardware random number generator formed from simple components can provide these properties. The components include two state machines with different structures, and free-running oscillators. The generated numbers pass the DIEHARD battery of tests.

The main uses of random numbers are in simulation and for cryptography. For simulation, the main requirement on the quality of the numbers is on their statistical properties; that they appear random.

Random number generators are used in many cryptographic algorithms and protocols. Their uses include generation of session keys and private keys, as a challenge against a replay attack, and as padding material for short messages. Weak random number generators can be targets for breaking into a cryptographic system [1]. When used for cryptographic purposes, random numbers must be unpredictable as well as have good statistical properties. We describe a hardware random number generator, used at Motorola, which passes Marsaglia's DIEHARD battery of tests [2], as well as FIPS-140 [3] and Crypt-X [4].

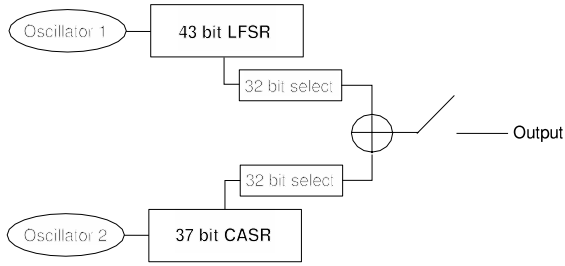

Fig. 1. Hardware random number generator block diagram.

The 32-bit hardware random number generator is based on a linear feedback shift register (LFSR), and a cellular automata shift register (CASR). Figure 1

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 450–453, 2003.

shows a simplified block diagram of the generator. Each shift register is clocked by an independent ring oscillator, and the output is sampled only when a new number is requested. The LFSR has 43 bits, and a characteristic polynomial of $X^{43}+X^{41}+X^{20}+X+1$. This is a primitive polynomial and gives a cycle length of $2^{43}-1$, (the all zero pattern is missing).

Wolfram [5] describes using 1-dimensional cellular automata (CA) with a neighborhood of three for generating random sequences. He defines CA rules as one of the 256 functions of three variables to define the next state of each cell site. The number of the rule is given by the decimal value of its eight row truth table. Pries [6] describes a hybrid CA using rules CA90 and CA150. The important property of this hybrid CA is that, with the appropriate selection of the CA90 and CA150 rules for each cell site, it gives a maximal length sequence. The CA90 rule is defined by the equation

$$a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$$

while the CA150 rule is defined by the equation

$$a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$$

The hardware random number generator uses a 37-bit CASR with a CA150 at cell site 28, and CA90s at all other cell sites. Hortensius [7] states "the hybrid CA's of maximal length that we have found all require null boundary conditions", and our CASR is no exception. The CASR has a maximal length of $2^{37}-1$, (again the all zero pattern is missing).

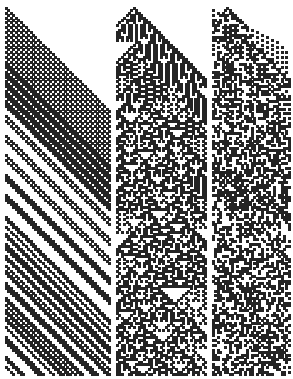

Fig. 2. State-time diagram of the LFSR, CASR and combined generator.

To generate a random number, 32 bits of the LFSR and CASR are selected and permuted, and then XORed together. Because the cycle lengths of the two state machines are relatively prime, the cycle length of the combined generator

is close to 2^{80} (actually $2^{80} - 2^{43} - 2^{37} + 1$). Because of the missing all zero pattern from both the LFSR and the CASR, there is a slight bias in the output bits, on the order of 2^{-43} and 2^{-37} respectively. In the combined generator, this bias drops to close to 2^{-80}. We have never generated enough random output to actually measure this bias.

Figure 2 shows a state-time diagram for the LFSR, CASR and combined generator. The initial states are shown at the top, and time progresses downward for 150 steps. The LFSR shows significant patterns, as the data in the LFSR is simply shifted right each step, and the only new bit is inserted at the left. The CASR is better, but triangular artifacts can readily be seen. The combination of the LFSR and CASR is much better (at least visually), but further testing will show that used this way, the combination still is not of acceptable quality.

Hortensius [7] also describes "site spacing" and "time spacing" as means to reduce the correlations between bits of the CA. The hardware random number generator uses site spacing in the selection and permutation of the LFSR and CASR bits it combines. Time spacing is also used in that there are two free running oscillators used as clocks for the two state machines, and a variable sampling period to capture the output data. The oscillators' frequencies vary with temperature, voltage and processing parameters. The state machines cycle through their states at different rates from each other and from the system clock. Even when not used, the hardware random number generator is active, keeping the internal state unpredictable. When multiple words are requested, there is a minimum sampling time which allows both state machines to clock at least twice their length (i.e., the LFSR is allowed to clock through at least 86 clock cycles). This minimum number of system clock cycles is determined by the lowest expected frequency of the free running oscillators.

Because this hardware random number generator has internal state, it is critical that the sequence of numbers it generates is not repeatable. The frequencies of the two oscillators are not controlled, and they drift with variations in temperature and voltage. Also, the state registers are not reset at power up, so that the intial state may take different values. These features allow the random number generator to cycle through a different sequence each time it is restarted.

The different components of the hardware random number generator were tested using DIEHARD. Figure 3 presents the results of running DIEHARD on individual pieces of the hardware random number generator, as well as actual silicon. DIEHARD is a collection of 15 tests, most of which give several results. In total, there are 234 p-values generated by DIEHARD. P-values are uniform over the range [0,1), for true random numbers. If uniform p-values are sorted and plotted, the result is a straight line, shown in Figure 2 as Ideal. The results for truely random data should approximate this line. The next three plots assume a single clock and sample every clock cycle. Both the LFSR and CASR fail the tests miserably, their p-values are not uniform. For the combination of the LFSR and CASR, there are 15 p-values equal to 1.0000, showing that there are still flaws in this generator, but it is a significant improvement over either individually. The final graph is from output of the actual hardware random number generator.

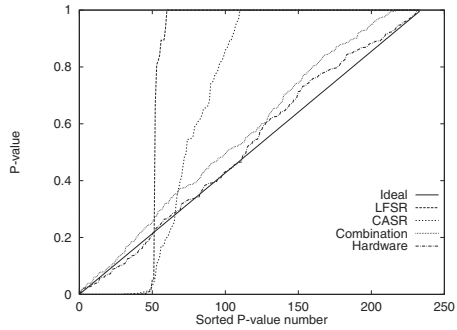

Fig. 3. DIEHARD results of the LFSR, CASR and combined generator

With variable time sampling, the hardware random number generator passes these tests.

The entire design of the hardware random number generator is written in Verilog RTL, with the exception of the ring oscillators, which are netlists of a number of inverters. The selected lengths of the inverter chains depend on the process technology and system clock frequency. If the random number generator will not be used for some time, the oscillators can safely be turned off to reduce power, thus allowing its use in lower power applications.

Acknowledgements. This random number generator has been used within Motorola for a number of years, and has gone through several minor variations in that time. Ezzy Dabbish and Steve Tugenberg developed the original hardware design.

References

1. I. Goldberg and D. Wagner. Randomness and the Netscape Browser. *Dr. Dobb's Journal*, January 1996.
2. G. Marsaglia. DIEHARD Statistical Tests. `http://stat.fsu.edu/~geo/diehard.html`
3. FIPS PUB 140. Security requirements for cryptographic modules. May 2001.
4. Crypt-X. `http://www.isrc.qut.edu.au/resource/cryptx`.
5. S. Wolfram. Random sequence generation by cellular automata. *Advances in Applied Mathematics*, 7:123–169, June 1986.
6. W. Pries, A. Thanailakis, and H. C. Card. Group properties of cellular automata and VLSI applications. *IEEE Transactions on Computers*, 35(12):1013–1024, December 1986.
7. P. D. Hortensius, R. D. McLeod, and H. C. Card. Parallel random number generation for VLSI systems using cellular automata. *IEEE Transactions on Computers*, 38(10):1466–1473, October 1989.