FPGA Implementation of Metastability-based True Random Number Generator

Graduate Adviser：Shuichi Ichikawa　　043732　Hisashi Hata

1 Introduction

The security of many cryptographic systems depends on random numbers. Thus, hardware True Random Number Generator (TRNG) is essential for many applications. Some of the TRNGs extract entropy from physical source (e.g., thermal noise). This kind of TRNG uses analog devices, while it is difficult to implement analog and digital circuit on one chip.

It is possible to implement TRNGs with pure digital circuit. Sunar proposed a TRNG that is composed of ring oscillators (RO) comprised of odd inverters. This type of TRNG dissipates much power, since it requires many free-run oscillators. It is also possible to adopt metastability for entropy source, where full-custom LSI technology has been adopted in most preceding studies, since it is difficult to sustain the quality of random numbers.

This paper describes a latch-type TRNG which is implemented with FPGA (Field Programmable Gate Array) technology. Since FPGA is a hard target for TRNG, our design is expected to work with semi- and full-custom LSI technologies.

2 Random number generation with RS latches

Generally, metastability occurs when set-up/hold-time is violated. Figure 1(a) illustrates a TRNG that utilizes the metastability of an RS latch. In figure 1(a), when TRNG CLK is 0, its output becomes stable at $(Q, \bar{Q}) = (1, 1)$. When TRNG CLK is 1, its output becomes $(Q, \bar{Q}) = (0, 1)$ or $(1, 0)$. Actually, the RS latch becomes metastable at the rising edge of TRNG CLK, and eventually it moves to one of the stable states. This corresponds to obtaining a random bit by a coin toss.

This type of TRNG is simple, while it is difficult to sustain the quality of random numbers. For example, if TRNG CLK has skew, the output might be biased. The unbalance of two NAND gates will lead to the same consequence.

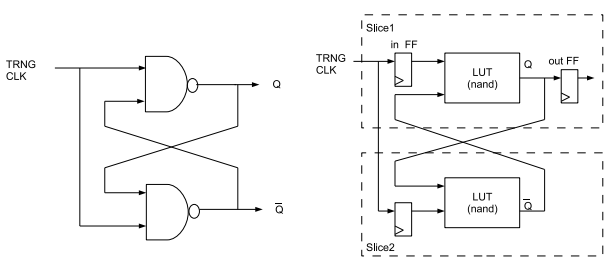
To implement this TRNG with FPGA, NAND gates are converted into LUTs. Figure 1(b) illustrates the RS latch implemented with LUTs (LUT latch). Flip-Flops (FFs) are inserted into input and output sides. Input FFs reduce TRNG CLK skew. Output FF balances the output loads of Q and $\bar{Q}$. Though it is essential to balance two NAND gates, we cannot guarantee it in FPGA. Therefore, we improve the quality of random numbers by XOR'ing the outputs of many latches. This design with *n* latches is designated as *LUT latch n* in the following discussion.

3 TRNG implementation and evaluation

In this study, Xilinx ML405 board is used as the evaluation platform. ML405 includes a Xilinx Virtex4 FPGA (XC4VFX20), which has an embedded PowerPC405 (PPC405) core. Xilinx ISE 10.1.03 software is used with default parameters. Our TRNG is attached to PPC405 as a peripheral device. The respective clock frequencies are 300 MHz for PPC405 and 100 MHz for peripheral devices.

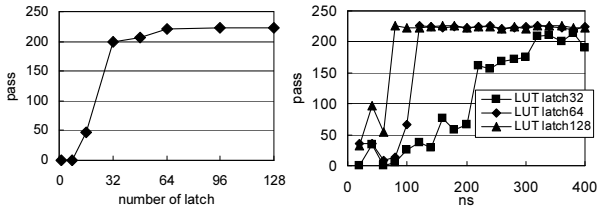
Our TRNG is connected to PPC405 through a buffer (SRAM 1MB), and it generates random numbers until buffer is filled. PPC405 checks buffer status by polling, and reads random numbers out when the buffer status is full. To generate more random numbers than 1 MB, sampling processes are repeated for each 1 MB data.

We use Diehard test and NIST test to evaluate the quality of random numbers. Diehard test is a test set of randomness proposed by



(a) Random number generation with an RS latch.　(b) LUT latch implementation.

Figure 1: RS latch.



(a) The effect of the number of LUT latch.　(b) The effect of sampling cycle.

Figure 2: Diehard test result.

Table 1: Logic scale and throughput.

Design	Slice	Mbps
LUT latch 64	145	3.8
LUT latch 128	290	8.3
LUT latch 256	580	12.5

Marsaglia. In this study, we use Diehard v0.2 beta [4]. Random numbers are evaluated by p values, which are uniformly distributed between [0,1] for ideal random numbers. Diehard test generates 229 p values, while it does not define the explicit guidelines on p values. Thus, we adopt the numbers of p values that are $0.01 <$p< 0.99 as a measure. If a random number sequence is ideal, approximately 225 p values are expected to pass.

NIST test was proposed by National Institute of Standards and Technology. It is composed of 15 tests. In this study, we use the NIST test ver 1.8 with default parameters for 10^9 input bits. The default rules are applied for p values. When a random number sequence passes all tests, we consider that it passed the NIST test.

Figure 2 summarizes the results of Diehard test. Figure 2(a) shows the influence of the number of LUT latches, where the sampling cycle is fixed to 320 ns. According to Diehard test, 64 or more LUT latches are required. Figure 2(b) shows the relation between sampling cycle and the number of LUT latches. When the number of LUT latches increases, sampling cycle can be shortened. Logic scale and throughput are summarized in Table 1.

4 Conclusion

In this study, we confirmed the fundamental functionality of latch-type TRNG. Further examinations are necessary for practical use.

For example, the influence of power-supply voltage and temperature should be exzmined. It is also necessary to evaluate our design with other devices, though our design does not depend on any device-specific features. Since our TRNG can adjust the random number quality be changing the number of LUT latches, it is expected to be portable among any kind of digital devices.

References

[1] S. Ichikawa, H. Hata, "True random number generator based on metastability of RS latch," Proc. SCIS 2009, 2F1-5, 2009. (CDROM)

[2] B. Sunar, W.J. Martin, and D.R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," IEEE Trans. Comput., vol.56, no.1, pp.109–119, 2007.

[3] NIST, "Statistical test suite," `http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html`, 2008.

[4] "Diehard Battery of Tests of Randomness v0.2 beta," `http://i.cs.hku.hk/~diehard/`, 2008.