C-012

Xillybus に付加した真性乱数生成器の予備的評価

Preliminary evaluation of true random number generator via Xillybus

岸部 仁美† 藤枝 直輝† 市川 周一†
Hitomi Kishibe Naoki Fujieda Shuichi Ichikawa

1. はじめに

暗号などのセキュリティ分野において真性乱数は重要である．真性乱数は物理的なランダム要因から生成され，予測が困難という特性を持つ．

藤枝ら[1]はソフトマクロで実装可能な Latch-latch 構造を持つ真性乱数生成器（TRNG; True Random Number Generator）を提案した（図 1）．この TRNG では RS ラッチのメタステーブル状態を利用して乱数を生成する．

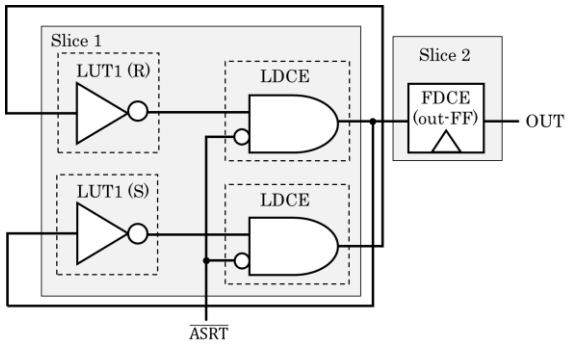

図 1 Latch-latch 構造の TRNG [1]

先行研究[1]では TRNG 単体の設計・評価をおこない，224 ラッチ以上あれば diehard test [2]に合格することが確認されている．しかし，先行研究[1]ではシステムとしての評価がおこなわれていない．

本研究では，藤枝らの TRNG を組込みシステムに統合し，実システムに近い条件下で乱数品質の評価をおこなった．FPGA（Field Programmable Gate Array）で設計した TRNG は CPU コアの周辺回路として接続される．本研究では，Xillybus [3]に TRNG を周辺回路として接続した．乱数品質の評価には先行研究[1]と同じ diehard test [2]を用いた．

あわせて本研究では，乱数品質を保ちながら TRNG の論理規模を減らす手法を検討する．本研究では 1 サイクル前の乱数出力と TRNG から生成された乱数出力で XOR をとり，この結果を次の出力とした．

2. 評価システムについて

2.1 Xillinux

Xillinux [3]を導入することで，Xillybus を通じてソフトウェア側から設計したハードウェアにアクセスできる（図 2）．Xillinux は Xillybus 社がリリースした Ubuntu 12.04 LTS（Long Term Support）に基づく Linux ディストリビューションである．Xillinux のブートパーティーションキットは xillinux-eval-zedboard-2.0a，SD カードイメージは xillinux-1.3.img.gz を使用した．

Xillybus では，ソフトウェアとハードウェア間のデータのやりとりにデバイスファイルを用いる．ハード側からソフト側へのデータの取得には/dev/xillybus_read_32 を使用し，ソフト側からハード側へのデータの書き込みには /dev/xillybus_write_32 を使用した．

†豊橋技術科学大学 電気・電子情報工学系, Department of Electrical and Electronic Information Engineering, Toyohashi University of Technology

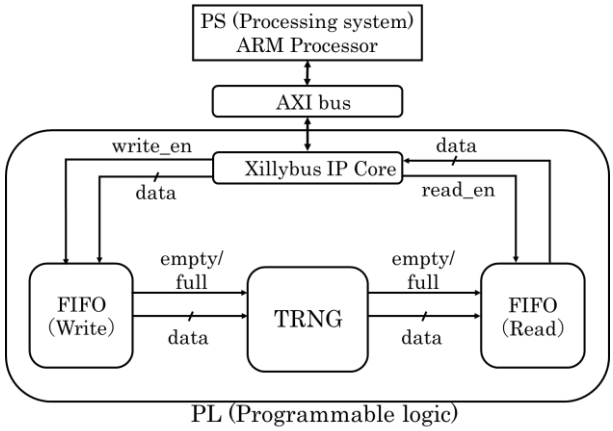

図 2 システムの全体図

3. 設計

3.1 ハードウェア設計

乱数を生成するハードウェアの概念図を図 3 に示す．1 サイクル前に出力された乱数と TRNG から生成された乱数を XOR した結果を Accumulated，生成された乱数をそのまま出力するものを Raw とする．ラッチのクロック周期は 320 ns とした．Xillybus 側のデータ幅（32 bit）に合わせるため，TRNG から生成される乱数 1 bit をシリアル—パラレル変換（S/P）した．

TRNG のラッチ数を変えてデータがとれるよう，16，32，…，256 個のラッチを図 4 に示すスイッチ（SW）で切り替える．SW は Xillybus の書き込みデバイスファイルに値を書き込むことで変更可能である．

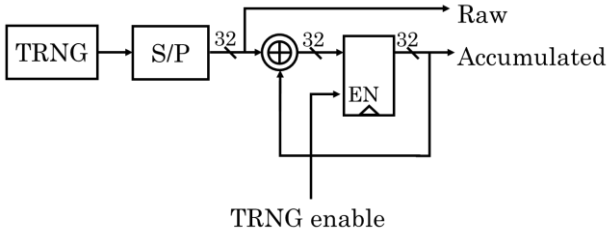

図 3 ハードウェア概念図

Copyright © 2018 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.

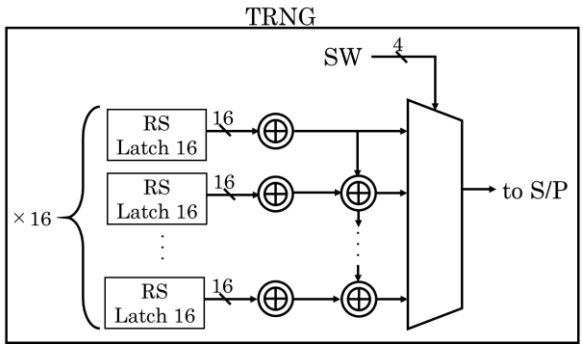

図 4 TRNG 概念図

3.2 ハードウェア実装環境

表 1 に示す環境でハードウェアを実装した．

表 1 ハードウェア実装環境[4]

CAD software	Xilinx Vivado 2017.1
FPGA Board	ZedBoard
Series	Zynq-7000
Device Package	XC7Z020-CLG484-1

3.3 ソフトウェア設計

ソフトウェアで約 100 Mbit の乱数データを取得した．ソフトウェアの記述には C 言語を用いた．

デバイスファイルへの値の書きこみには write()を用いる．ラッチ数を切り替える SW に値を与えるため，キーボード入力からの値を scanf(“%2d”, &SW)で取り込み，SW を write()でデバイスファイルに書き込む．

デバイスファイルからの乱数の取得には read()を用いた．一度に取得する乱数は 32 bit×128 個=4096 bit とした．TRNG から取得した乱数は 16 進数 8 桁のデータとして配列の要素に保存される．128 個の乱数を読み込むと，それらをすべてテキストファイルに出力する．

乱数が 100 Mbit になるまで， read()で乱数取得，テキストファイルに出力を繰り返す．

乱数生成速度は Accumulated，Raw ともに約 3 Mbps となった．これはラッチのクロック周期と一致する．

4. 評価

乱数品質の評価に diehard test [2]を用いた．18 種類のテストから構成され，約 100 Mbit のビット列が必要となる．テストごとに 1～100 個の p 値が出力される．p 値が$10^{-6} < p < 1 - 10^{-6}$の範囲内のとき合格とした．

1 サイクル前に出力された乱数と TRNG から生成された乱数を XOR する場合（Accumulated）と，生成された乱数をそのまま出力する場合（Raw）について評価をおこなった．図 5 より，横軸がラッチ数，縦軸が diehard test に不合格となった数を表している．Accumulated の場合では，96 ラッチ以上になると，18 種類すべてのテストに合格した．Raw の場合では， 144 ラッチ以上あると質の良い乱数が取り出せることがわかった．Accumulated は Raw の 2/3 のラッチ数で diehard test に合格することがわかった．

生成された乱数を XOR で累積することにより，TRNG 実装コストを大幅に削減できる可能性がある．

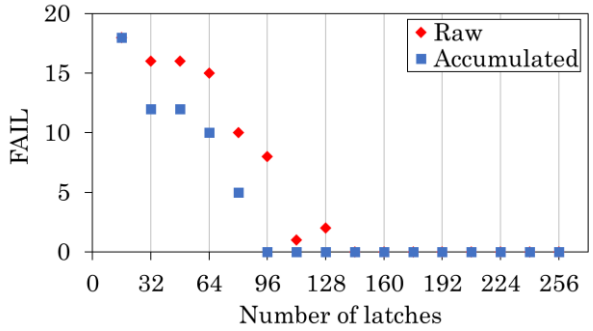

図 5 diehard test の結果

5. おわりに

藤枝ら[1]が提案するソフトマクロで実装可能な Latch-latch 構造を持つ TRNG を利用し，TRNG のラッチ数が 16, 32, …, 256 個の場合の乱数品質の評価をおこなった．

1 サイクル前に出力された乱数と TRNG から生成された乱数を XOR する場合（Accumulated）と，生成された乱数をそのまま出力する場合（Raw）で評価をおこなった．Accumulated では 96 ラッチ以上，Raw では 144 ラッチ以上あれば，diehard test に合格することがわかった．

6. 今後の課題

今後は，乱数の取り出し時間と乱数品質のトレードオフについて調査する予定である．乱数を取り出す時間が短いと乱数品質に差が出るのか検討したい．

ラッチ数と乱数生成速度のトレードオフについても検討し，コストパフォーマンスの良い TRNG の設計方法について調査する．

さらに乱数品質を厳密に評価するために NIST テスト[5]に通すことを今後の課題とする．

謝辞

本研究の一部は JSPS 科研費 16K00072 の支援による．

参考文献

[1] Naoki Fujieda, Shuichi Ichikawa, “A latch-latch composition of metastability-based true random number generator for Xilinx FPGAs,” IEICE Electronics Express, vol.15, No.10, pp.1-12(2018).
[2] George Masaglia, “Diehard battery of tests of randomness (mirror) ,” https://github.com/reubenhwk/diehard.
[3] Xillybus Ltd., “XILLYBUS. IP cores and design services,” http://www.xillybus.com/
[4] AVNET, “ZedBoard,” http://www.zedboard.org/product/zedboard.
[5] NIST, “NIST SP 800-22: Download Documentation and Software,” https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software.

Copyright © 2018 by
The Institute of Electronics, Information and Communication Engineers and
Information Processing Society of Japan All rights reserved.