社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2009-SLDM-138（5）
2009 / 1 / 29

メタスタビリティを利用した真性乱数生成回路のFPGAによる実装

畑　尚志† 市川 周一†

† 豊橋技術科学大学・知識情報工学系，441-8580 豊橋市天伯町雲雀ケ丘 1-1
E-mail: †ichikawa@ieee.org

あらまし 真性乱数生成回路 (TRNG) をデジタル回路で実装する手法として，ラッチのメタスタビリティを利用する回路が提案されている．ラッチ型 TRNG は実装が難しいためカスタム LSI で実現されてきたが，本研究では FPGA で実装する手法を提案する．提案回路は乱数品質を高めるためにハードマクロで実装し，クロックスキュー低減や内部ノードの負荷均衡化に配慮した．さらに乱数品質と生成速度を改善するため，複数ラッチの出力を XOR してエントロピー収穫を行っている．作成した TRNG は Xilinx Virtex4 FPGA XC4VFX20 に実装し，NIST テストに後処理なしで通過することを確認した．ラッチ 128 個からなる TRNG で，回路規模 290 Slice，生成速度 8.33 Mbps を実現した．
キーワード 乱数生成，同期式デジタル回路，FPGA，準安定状態

FPGA Implementation of Metastability-based True Random Number Generator

Hisashi HATA† and Shuichi ICHIKAWA†

† Dept. Knowledge-based Information Engineering, Toyohashi University of Technology
1-1 Hibarigaoka, Tempaku, Toyohashi 441-8580, Japan
E-mail: †ichikawa@ieee.org

Abstract Metastability of RS latch is utilizable as an entropy source for true random number generators (TRNG). This kind of TRNG is comprised of logic gates, which can be integrated into a logic LSI. Though latch-based TRNG has been mostly implemented with full-custom LSI technology, this study presents an implementation with common FPGA technology. The RS latch in our TRNG is implemented as a hard-macro to guarantee the quality of randomness, minimizing the clock skew and load imbalance of internal nodes. The quality and throughput are further improved by XOR'ing the output of 32–128 latches. The derived design was implemented with Xilinx Virtex4 FPGA (XC4VFX20), and passed NIST test without post-processing. A TRNG of 128 latches occupies 290 slices, while achieving 8.33 Mbps throughput.
Key words random number generator, synchronous digital circuit, FPGA, metastability

1. はじめに

多くのセキュリティ技術は安全性の根拠を乱数に依存している．そのためハードウェアによる真性乱数生成器 (True Random Number Generator; TRNG) は，セキュリティの基盤技術として重要である．熱雑音など物理現象を利用した TRNG も実用化されているが，実装にアナログ回路が必要であるため論理 LSI への集積が難しい．デジタル回路で TRNG を構成できれば，論理 LSI に集積可能であるため，応用範囲も広く実用的価値が高いと考えられる．

デジタル回路で構成可能な TRNG として，メタスタビリティを利用する回路が提案されている．しかしメタスタビリティを利用した TRNG は実装が難しいため，カスタム LSI で実装した例がほとんどである．本研究では，RS ラッチのメタスタビリティを利用した TRNG (ラッチ型 TRNG) を FPGA で実装し，その乱数品質と生成速度を報告する．FPGA で実装するための回路設計上の工夫と，最適な設計パラメータを決定するための実験方法についても述べる．提案回路は同期式デジタル回路であるため，FPGA に限らず全ての論理 LSI で実装可能である．

2. 関連研究

TRNG の先行研究は多いが，本章では本研究と直接関係する研究だけを紹介する．より広範で詳細な参考文献リストについては，著者らの別稿 [1] を参照されたい．

デジタル回路で TRNG を構成する手法の代表例として，発振器のジッタを利用する方法と，メタスタビリティを利用する方

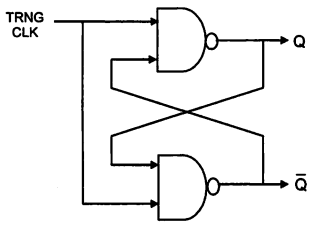


図1 RS ラッチによる乱数生成

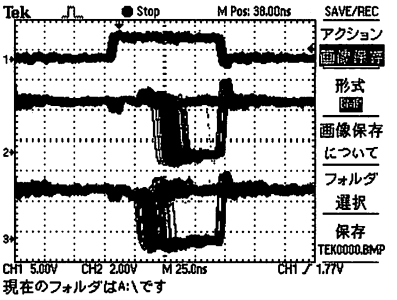


図2 RS ラッチのメタスタビリティ (Virtex4 FPGA)

法があげられる.

発振器のジッタからエントロピーを生成する方法の一例に, リングオシレータ (Ring Oscillator; RO) を利用する手法がある. Sunar ら [2] は, 多数の RO の出力を xor で集約する手法を提案した (以下 Sunar 型). Schellekens ら [3] は Sunar 型の TRNG を Xilinx Virtex II Pro FPGA (XC2VP20) で実装し, 2 Mbps 以上の乱数生成速度を報告している. しかし Shellkekens らの実装では, 330MHz で動作する free-run の RO が 110 個並列で動作しており, 回路規模と消費電力の面で問題がある. それに対し本研究では, 小型で省電力の TRNG 設計を提案する.

メタスタビリティを利用した TRNG の一例として, RS ラッチのメタスタビリティを利用する回路が提案されている [4]. 一般に回路のメタスタビリティから高品質な乱数を生成することは容易でないため, 先行研究ではカスタム LSI 技術を用いて回路設計上の工夫を施している. Bellido ら [4] は, 初期化専用のクロックでノードのプリチャージやディスチャージを行っている. Kinniment と Chester [5] は, 差動増幅器とラッチを組合わせた回路 (R-flop) をエントロピー源とし, ネガティブフィードバックで R-flop のバイアス電圧を調整することにより, 乱数品質を保持している. Tokunaga ら [6] は, 動作中にメタステーブル状態の解消時間 t_d を計測し, t_d の平均値が大きくなるようにラッチの内部電位を初期化することにより, 乱数品質の向上と安定化を果たしている.

一方, 本研究では, システムクロックに同期した論理回路だけでラッチ型 TRNG を実現し, FPGA を用いた実装で NIST テストに合格することを報告する.

ラッチ型 TRNG を FPGA に実装した報告は, 本研究以外には 1 件しか報告されていない. Danger ら [7] は, D flipflop (DFF) のセットアップ/ホールドタイム違反を利用するラッチ型 TRNG を Altera Stratix FPGA で実現した. FPGA に組み込まれた FF はメタスタビリティを発生しにくいため, Danger は LUT を使用して D latch を構成している. メタスタビリティを発生させるためには, D 入力とクロックのスキューを小さくする必要があるので, Danger らは手動で配置配線を行い, 配線遅延でスキューを調整している. Danger らの手法は, アナログ遅延を使用するため電圧や温度の変化に影響されやすく, 移植性の面でも問題があると考えられる.

一方, 本研究ではアナログ的遅延に頼らない, 同期式デジタル回路による TRNG を提案する.

3. 動作原理

3.1 RS ラッチのメタスタビリティ

一般に, ラッチのセットアップ時間やホールドタイム違反が起こると, メタスタビリティが発生する [8]. 例として, RS ラッチの R 入力と S 入力を同時にアサートする回路を図 1 に示す. 図 1 の回路は, TRNG CLK=0 では出力 $(Q, \bar{Q}) = (1, 1)$ で安定状態となるが, TRNG CLK=1 では $(Q, \bar{Q}) = (0, 1)$ あるいは $(1, 0)$ で安定となる. 実際には, ラッチは TRNG CLK の立ち上がりエッジでメタステーブル状態 (準安定状態) になり, その後, いずれかの安定状態へ遷移する. これは (理想的には) コイントスにより 1 bit の乱数を得ることに相当する. 安定状態への遷移時間は一定でなく, 確率分布をもつ [8].

上に述べた TRNG の動作原理は単純であるが, 実際に高品質の乱数を生成することは容易でない. 例えば TRNG CLK 信号にスキューがある場合, ラッチの出力が 0 もしくは 1 に偏る可能性がある. 2 つの NAND ゲートのドライブ能力に差がある場合も同様である. スキューやドライブ能力差がない場合でも, $Q, \bar{Q}$ 間に電位差がある状態で TRNG CLK=1 になれば, 出力に偏りが生じうる. $Q, \bar{Q}$ の電位は TRNG CLK=0 の期間に初期化されるため, TRNG CLK はアサート期間だけでなくネゲート期間も十分にとる必要がある.

3.2 FPGA を用いた乱数生成

このようにラッチ型 TRNG の実現には注意深い回路設計が必要であるため, 多くの先行研究ではカスタム LSI で実装評価している. 従って, FPGA 上のラッチを利用して良い乱数が生成できるかどうかは, 必ずしも自明でない. そこで予備実験として, FPGA 上に図 1 のような回路を実装し, 出力を観察した.

観測には Xilinx Virtex4 FX20 FPGA を使用し, NAND ゲート 1 個に対して 4 入力 LUT 1 個を割り当てた. 遷移時間の観測を行うため, 出力は FF を通さず直接汎用ピンに出力している. IO バッファの出力電圧設定は LVCMOS 3.3V とし, Tektronix の TDS2024B で観察した. ラッチの個体毎にメタスタビリティの発生状況が違うため, 多くのラッチを実装した中から, メタスタビリティを頻繁に発生する RS ラッチを選択して観測した.

観測波形を図 2 に示す. 波形は上から TRNG CLK, $Q, \bar{Q}$ を表し, 5 秒間の波形を重ねて表示している. オシロスコープの掃引時間は 25 ns/div とした. IO バッファを通しているため, 遷

移中の中間電位の観測はできなかったが，周期毎に Q の値がランダムに変化することや，準安定状態からの遷移時間が一定でないことが観測できた．観測された遷移時間は数 10 ns に達する場合があり，乱数生成速度の制約要因になりうることがわかった．

このように FPGA の RS ラッチでもメタスタビリティを観測できるが，ラッチの個体差は大きく，出力が 0 または 1 に固定されているものも多い．ラッチの生成エントロピーの個体差については，5.3 章で評価結果を示す．実用的には，複数ラッチの出力を xor で集約し，TRNG の出力を生成すればよい．以下，n 個の RS ラッチを使用する TRNG を **LUT latch** n と表記する．

4. 実装と評価の方法

4.1 実装評価環境

実装評価には，Xilinx Virtex4 FX20 (XC4VFX20) を搭載する Xilinx ML405 ボードを使用した．CAD は Xilinx ISE 10.1.03 を使用し，パラメータはデフォルトで使用した．乱数の取得は，FPGA に搭載されている PowerPC 405 に TRNG を周辺回路として接続して行う．測定システムは PowerPC 405 に UART, System ACE(注1), Ethernet MAC, DDR SDRAM, TRNG を接続した構成とした．乱数検定に大量 (10^9 bit) のデータが必要になるため，OS に組込み Linux を使用している．Linux のファイルシステムはコンパクトフラッシュに置かれており，取得した乱数データはコンパクトフラッシュに蓄積される．クロックは PowerPC が 300MHz，バスクロックが 100MHz である．

TRNG はバッファ(1 MB SRAM) を介して PowerPC 405 に接続されている．動作開始後，TRNG はバッファを満たすまで出力を行い，バッファが満たされると停止する．PowerPC 405 は，バッファが満たされると結果をファイルに保存する．1 MB 以上の出力を生成する場合は，1 MB の計測を複数回繰り返す．

TRNG CLK はバスクロックを m 分周して作成する．係数 m は，メモリマップされたレジスタによってソフトウェア的に変更可能である．実験の簡素化のため，本研究では TRNG CLK のデューティ比を 50%に固定している．従って TRNG CLK の周期は $(20 \times m)$ ns となる．TRNG CLK はクロック専用線を使用せずに一般配線によって配線される．

4.2 評価方法

本研究では 2 つの代表的テストを用いて，設計段階の比較検討と最終的な品質評価を行う．Diehard テスト [9] は必要なデータ量が小さく短時間でテストできるため，設計上の試行錯誤やパラメータ決定に最適である．一方，NIST テスト [10] は網羅的で大量のデータを必要とするため，TRNG の最終評価に使用する．

4.2.1 Diehard テスト

Diehard テスト [9] は Marsaglia が提案した乱数テストセットである．本研究では Diehard テスト v0.2 beta [11] を使用した．Diehard v0.2 beta では 17 種類のテストが提供されているが，そのうち GCD，Gorilla, Overlapping permutation テストは 10 MB 以上の乱数を必要とするため，除外して用いた．

乱数の評価は p value によって行う．p value は理想的な乱数ならば [0,1) の区間に一様分布するとされる値で，Diehard テストは 10 MB の乱数を入力すると 229 個の p value を出力する．Diehard テストでは p value の評価基準を設けていないので，本研究では，0.01 <p value< 0.99 の範囲に収まる値がいくつあるかで評価した．理想的な乱数であれば，225 個前後の p value がテストに合格する．入力データのエントロピーが低くすべてのテストが実行できない場合は，そのテストを除外して評価した．

このテストは十分でないが，悪い設計を排除することは可能なので，設計時の簡便な比較評価には有用である．

4.2.2 NIST テスト

NIST テスト [10] [12] は，National Institute of Standards and Technology が提案する乱数検定テストで，15 種のテストから構成されている．本研究ではバージョン 1.8 を使用し，データ量は 10^9 bit，パラメータはデフォルトとした．NIST テストでは 10^6 bit のテストを 1000 回行い，それから得られる p value の分布を評価する．p value の評価は NIST の評価基準に準じて行い，15 種類のテストすべてに合格した場合に NIST テストに合格したと判定する．

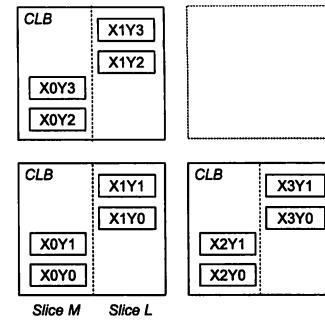


図 3 Virtex4 CLB [13]

5. ラッチの実装

5.1 Virtex4 でのラッチ実装

FPGA の構造は品種により様々であるが，論理資源と配線資源を基本とする点は共通である．論理資源は階層化されており，論理演算は最終的に LUT (look-up table) で実現される．

Xilinx Virtex4 FPGA の場合，4 入力 LUT と FF のペア 2 組で 1 Slice が構成され，4 Slice を 1 組として CLB が構成される (図 3)．CLB 内の Slice のうち，偶数列は Slice L，奇数列は Slice M が配置されている．Slice M は分散メモリ等にも使用可能な構成を持ち，一方 Slice L は論理用途のみに利用できる．

TRNG の RS ラッチは，図 4 に示す通り，2 Slice に分けて実装される．本研究では自動配線の影響を避けるため，各 RS ラッチをハードマクロで実装した．ただしラッチの各インスタンスの配置は，自動配置配線で行う．同一 CLB の Slice L と Slice M を使用してハードマクロ化した場合，自動配置配線が配線不能により失敗するため，本研究では**異なる CLB の同種の Slice**

(注1)：コンパクトフラッシュインターフェース．

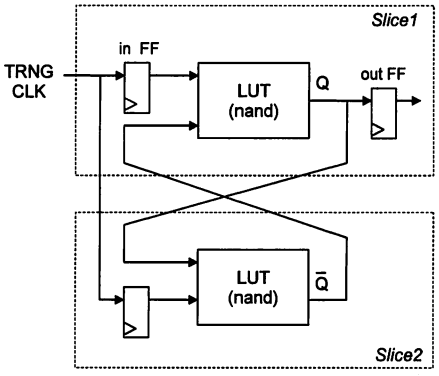


図4　FF を使用した LUT latch の実装

2 個を使用して RS ラッチを実装する．

2 つの Slice 間の距離を大きくすると，スキューが増大して生成エントロピーが小さくなる可能性がある．配線遅延も大きくなるため，遷移時間が増大して生成速度が低下する可能性もある．一方で，配線が長いとノイズレベルが大きくなり，エントロピーが増大する可能性も否定できない．この点は 5.2 章で定量的実験結果を示す．

当面，スキューと配線コストの低減を重視して，X 軸方向に隣り合った CLB の Slice L で RS ラッチを実装するものとする．図 4 の Slice 1 を“基点”と表記すると，図 3 の X1Y0 に基点がおかれた場合，Slice 2 は X3Y0 に置かれることになる．ハードマクロは Slice L と M を区別するため，自動配置配線は基点を Slice L のみに割り当てる．図 3 の例では，基点は X1Y0, X1Y1, X1Y2, X1Y3, X3Y0, X3Y1 のいずれかに割り当てられる可能性がある．

図 4 の FF の有無についても，3 種類の設計を比較した．in FF も out FF も，LUT と同一 Slice 内の FF を使用して，回路規模に影響を与えず実装することができる．

（1） **no FF**　図 4 の in FF, out FF を取り除いた RS ラッチ．

（2） **in FF**　図 4 の in FF のみを実装する．図 5 に示すように，in FF は NAND LUT と同一の Slice に実装して，LUT 入力のスキューを最小化する．

（3） **in out FF**　図 4 をそのまま実装する．出力に out FF を追加することで，Slice 1 の後段回路の影響を小さくし，Q と $\bar{Q}$ の配線負荷を均衡化する．

上記 3 種のラッチを用いて LUT latch 32 を作成し，Diehard テストで評価した (図 6)．サンプリング周期はソフトウェアで変更しているため，FPGA の書き換えは行っていない．出力に FF をつけることにより，乱数品質が顕著に改善することがわかる．in FF は no FF に対して優位性が見られるため，スキューを小さくする効果は認められる．サンプリング周期が 320 ns 程度必要であることもわかる．以下，RS ラッチとして in out FF の実装を採用し，**LUT latch** と表記する．

5.2　Slice 間の距離の影響

LUT latch を構成する Slice 間の距離を変えた場合の影響を調べた．Slice L を基点とし，X 軸方向に離した場合と Y 軸方向

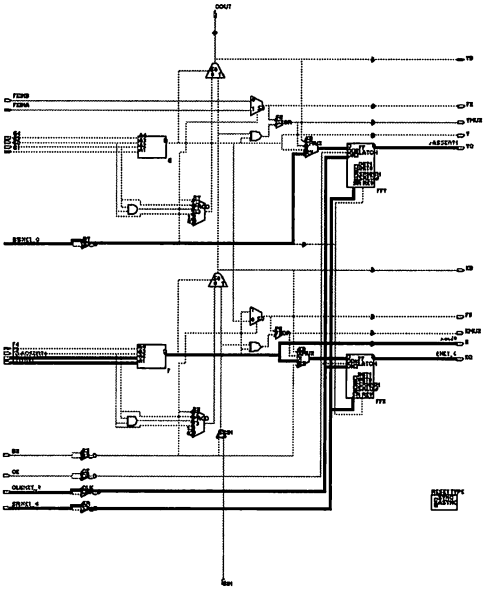


図5　Slice 1 の資源割り当て (Slice L)

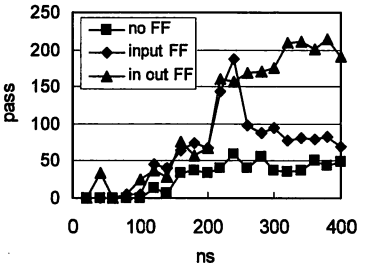


図6　FF の実装と乱数品質

に離した場合について評価を行った．X 軸方向に離す場合は偶数 Slice(注2)，Y 軸方向は 1 Slice ずつ離して評価する．LUT latch 64 でサンプリング周期を 320 ns としたとき，Diehard テストの結果は表 1 の通りであった．

表 1 のラベルは，基点からの相対距離を表している．例えば図 3 において，基点が X1Y0 で相対距離が dX2Y0 の場合，X1Y0 と X3Y0 のスライスが使用される．dX0Y1 の場合は，同じ CLB 内の 2 つの Slice を使用することになり，配置配線でエラーがでるため除外した．また，dX4Y0 の自動配置配線でもエラーが出たため，配置配線の重みを指定する cost table を変えて対応した．自動配置配線のエフォートレベルは変更していない．

表 1 で dX2Y0 の結果が最良であり，また Slice 間の距離は小さい方が配線資源を節約できることから，LUT latch のハードマクロとしては dX2Y0 を採用する．

5.3　LUT latch 1 個あたりのエントロピー

個々の LUT latch が出力するエントロピーについて評価を

(注2)：奇数列は Slice M である．

表1 Slice 間の距離による影響

	dX2Y0	dX4Y0	dX6Y0	dX8Y0	dX0Y2	dX0Y3	dX0Y4
pass	226	0	224	0	215	224	131

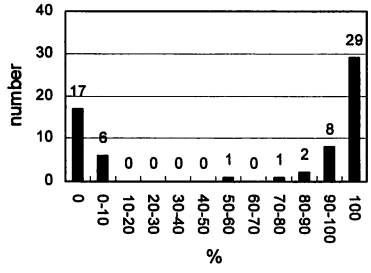


図7 ラッチの個体差とエントロピー

行った．64 個の LUT latch を実装し，各ラッチの出力を MUX で選択して 1 MB ずつ取得し，乱数列中の 1 の割合を調査する．LUT latch の選択はソフトウェアで行い，サンプリング周期は 320 ns に固定した．評価結果をヒストグラム (図 7) として示す．横軸は 1 の割合，縦軸は個数を表す．72%の LUT latch は出力が 0 または 1 に固定されており，エントロピーを生成していない．残る 28%のラッチも，多くは出力に 90%以上の偏りを示している．この原因としては，ラッチを構成する LUT のドライブ能力の不均衡などが考えられる．FPGA ではフィードバック回路等で回路を制御することは難しいため，複数の LUT latch からエントロピーを収穫することにより，乱数品質の向上を図る必要がある．

5.4 ラッチの配置方法の影響

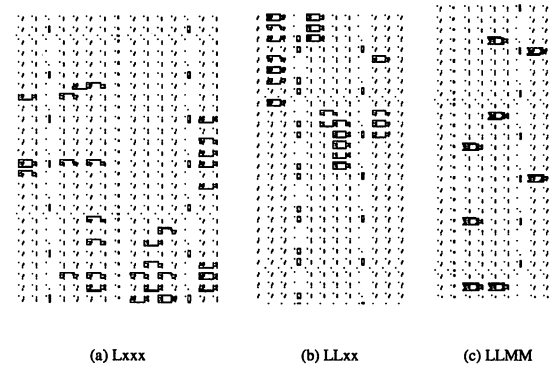
5.1 章のハードマクロでは，2 CLB (8 Slice) のうち 2 つの Slice L を用いて LUT latch を実装した．未使用の Slice は自動配置で利用されるため無駄にはならないが，ハードマクロで LUT latch を並べて密に実装することも可能である．そこで，LUT latch の基点が密集した場合の影響を，以下の 3 種類について評価した．

（1） **Lxxx** 1 組の Slice L で 1 つの LUT latch を実装する．ただし，1 つの CLB に LUT latch が 1 つだけ配置されるとは限らない．自動配置配線で，同じ CLB 内に 2 つの基点が置かれる可能性はある．

（2） **LLxx** ハードマクロで，2 つの LUT latch の基点 (Slice L) が隣接して配置されるように実装する．ただし 2 組の LUT latch の配置は自動配置配線が行う．図 3 の例では，X1Y0, X1Y1 のように 1 つの CLB に基点が置かれる場合と X1Y1, X1Y2 のように CLB を超えて配置される場合がある．

（3） **LLMM** ハードマクロで，1 つの CLB に基点が必ず 4 つ配置されるように実装する．即ち，一対 (2 個) の CLB に，4 つの LUT latch が実装される．図 3 の例では，X0Y0, X0Y1, X1Y0, X1Y1 の全てに基点が置かれる．4 個 1 組の LUT latch の配置は，自動配置配線が行う．

LUT latch 32 個を自動配置配線した結果を，図 8 に示す．各



(a) Lxxx (b) LLxx (c) LLMM

図8 LUT latch の配置

表2 CLB に割り当てられる基点の数の影響

	Lxxx	LLxx	LLMM
pass	209	1	28

配置図の濃く見える部分が LUT Latch である．Lxxx が最も広い範囲に渡って配置されており，一方 LLxx や LLMM では基点が置かれる CLB 数が減っていることが確認できる．

これらの LUT latch32 について，サンプリング周期 320 ns で Diehard テストを行った．評価結果 (表 2) から明らかなように，同じ CLB に複数の基点が置かれると乱数品質が顕著に低下する．この原因として，(1) 配置が密集すると配線が難しくなって生成エントロピーが低下する，(2) 近接したラッチでは何らかの理由で出力に相関関係が発生する，等が考えられる．これ以上の原因究明は行っていないが，実験結果に従い，本研究では Lxxx の実装を採用する．

5.5 LUT latch の数の影響

TRNG の LUT latch 数を変えた場合の影響を評価した．サンプリング周期を 320 ns とし，LUT latch の数を変えて Diehard テストで評価した結果を，図 9 に示す．この結果では，64 個以上の LUT latch を使うことが望ましい．

次に LUT latch の数とサンプリング周期の関係について評価した．LUT latch 32, 64, 128 について，サンプリング周期を 20 ns 刻みで変えながら Diehard テストで評価した結果を，図 10 に示す．Diehard テストには LUT latch 64 で通過するが，さらに LUT latch の数を増やすことにより，サンプリング周期を短縮できることがわかった．LUT latch が増えることにより，遷移時間の短い LUT latch が増えるためと考えられる．

6. 乱数品質と回路規模

最終評価として，LUT latch 64, 128, 256 を NIST テストで評価し，全て合格することを確認した．各回路の論理規模と乱数生成速度を表 3 に示す．論理規模は乱数生成部のみを示している．乱数生成速度は，サンプリング周期を変えながら NIST テストを繰り返して求めた最短周期に基づく．

回路規模は 145～580 Slice であるが，XC4VFX20 の回路規模は 8544 Slice なので，TRNG はチップの 1.7～6.7%程度となる．

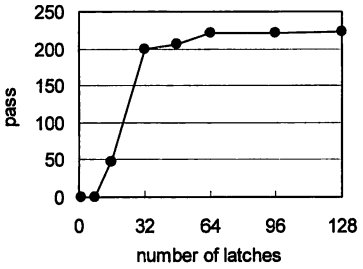


図 9　LUT latch の数と乱数品質

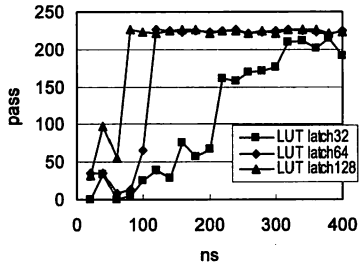


図 10　サンプリング周期と乱数品質

表 3　回路規模と乱数生成速度

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

乱数生成速度では LUT latch 256 が最も高い値になったが，各 TRNG は論理規模が違うため一律に生成速度を比較することは出来ない．LUT latch 128 の生成速度は 8.33 Mbps であるが，2 つ並列に実装すれば LUT latch 256 と同等の回路規模で 16.66 Mbps の生成速度を達成する．ただし TRNG 間に相関が発生する可能性があるので，実装して検証する必要がある．

7. おわりに

本研究では RS ラッチのメタスタビリティを利用した TRNG を FPGA で実装・評価した．提案回路は 290 Slice で 8.33 Mbps を達成し，後処理なしで NIST テストに合格する．ハードマクロで実装することにより FPGA でも高品質の乱数を安定して生成することを可能にした．提案回路は同期式デジタル回路で構成されており，チップ外に追加回路が必要ないため容易に実装できる．クロックを停止することで電力消費を抑えることができるため，組込み用途にも適している．

本研究では TRNG CLK のデューティ比を 50%に固定した．しかし本来，TRNG CLK = 1 の期間は準安定状態から安定状態の遷移時間であり，一方 TRNG CLK = 0 の期間は $Q,\bar{Q}$ の初期化時間である．それぞれが別の物理過程であるため，それぞれを最適値に調整することにより，生成速度を改善できる可能性がある．

本研究でラッチ型 TRNG の基本動作は確認したが，実応用には更なる検討が必要である．まず，電源電圧や温度の変化に対する，乱数品質と生成速度の評価が必要である．しかし提案回路はアナログ的な配線遅延等を利用していないため，電源電圧，温度変化の影響は小さいと思われる．

XC4VFX20 以外のデバイスでも評価する必要がある．ラッチ型 TRNG は FF と論理要素で構成されるシンプルな回路で有ること，ラッチ数の変更で乱数品質と生成速度の調整ができることから，別のデバイスに対する実装も可能だと推測される．また，FPGA でも実装できるので，セミカスタムやフルカスタムの論理 LSI にも適用可能であると考えられる．

謝辞

本研究の一部は，科学研究費補助金・基盤研究 (C)19500042 により行われた．

文　献

[1] 市川周一，畑尚志，“RS ラッチのメタスタビリティを利用した真性乱数生成回路,” 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 Transactions on Computers, vol.56, no.1, pp.109–119, 2007.

[3] D. Schellekens, B. Preneel, and I. Verbauwhede, “FPGA vendor agnostic true random number generator,” Proc. FPL 2006, pp.1–6, 2006.

[4] M. Bellido, A. Acosta, M. Valencia, A. Barriga, and J. Huertas, “Simple binary random number generator,” Electronics Letters, vol.28, no.7, pp.617–618, 1992.

[5] D. Kinniment, and E. Chester, “Design of an on-chip random number generator using metastability,” Proc. ESSCIRC 2002, vol.4, no.6, pp.595–598, 2002.

[6] C. Tokunaga, D. Blaauw, and T. Mudge, “True random number generator with a metastability-based quality control,” IEEE Journal of Solid-State Circuits, vol.43, no.1, pp.78–84, 2008.

[7] J.L. Danger, S. Guilley, and P. Hoogvorst, “Fast true random generator in FPGAs,” Proc. IEEE NEWCAS 2007, pp.506–509, 2007.

[8] J.R. Marino, “General theory of metastable operation,” IEEE Transactions on Computers, vol.30, no.2, pp.107–115, 1981.

[9] G. Marsaglia, “The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness,” `http://www.stat.fsu.edu/pub/diehard/`, 2008.

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

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

[12] A. Rukhin, et al., “A statistical test suite for random and pseudorandom number generators for cryptographic applications,” NIST Special Publication 800-22 (with revisions dated May 15, 2001).

[13] Xilinx, “Virtex-4 FPGA user guide,” 2008.