Random Number Generation Based on Oscillatory Metastability in Ring Circuits

By Laszlo Hars (Laszlo at Hars dot US)

Abstract*: Random number generator designs are discussed, which utilize* **oscillatory metastability***, induced by switching between two stable states of ring-connected digital gates. For a short time after the switch-over the circuits behave quite randomly, influenced by the circuit noise. We provide simple programs, which* **simulate** *the fundamental behavior of our circuits. We also present a mathematical* **model** *and* **theoretical explanations** *of the underlying physical phenomena, the random phase* **drift** *and* **pulse decay***. These also illuminate the principles of other recently published random number generators. The feasibility of the designs was confirmed by FPGA prototypes. These random number generators are small, fast and built of standard logic gates. The simplest example contains just* **one XOR gate** *as the source of randomness.*

Keywords*: Electronic random number generators, Ring oscillators, Metastability, Random walk*

Introduction

Many applications need a lot of random numbers, including communication protocols with frequently changing session keys, nonces; secure servers; scientific or engineering simulations; Monte Carlo- or randomized computations; dithering; gambling; electronic games… Often less secure pseudorandom numbers are used; or off-line or off-site generated true random numbers, which may need expensive large secure buffers, or secure communication channels.

In this paper the theory and the most important practical implementation issues of a family of electronic random number generators are discussed. These generators help reduce costs, improving security and performance in many applications. The circuits can be built of few standard digital components, so they are simple, inexpensive, fast, and of low power.

The basic principle of operation of random number generation is amplifying some effects of the always present noise to detectable levels. In the circuits considered below, little, noise-induced phase jitter leads to large differences in settling waveforms.

The idea of using metastability for random number generators in our proposed circuits originates from various designs published or disclosed on or after 2001: [14], [15], [16], [17], [21], [22] and [23]. One family of such random number generators, elaborated below, utilizes the start-up artifacts of rings, which appeared in [23] and in 2008, the first version of this paper, submitted to CHES. The random startup behavior of long latch circuits (instead of oscillators) was exploited later in [25], and a similar idea was explored in [24], without practical considerations, theoretical background or explanations.

Another manifestation of the oscillatory metastability, elaborated in this paper, is in the stopping cycles of ring oscillators, when they are turned to latches. Since these kinds of circuits have not been considered for random number generation before, we put the largest emphasis on them.

The main points of this work are practical random number generator designs, the ***mathematical model*** for the randomness in oscillatory metastability, and explanations of the observed physical phenomenon, the ***pulse decay***. We also present MATLAB ***macros***, which ***simulate*** some aspects of the observed circuit behaviors reasonable well, despite the simplicity of their code.

Oscillatory Metastability at Stopping a Ring Oscillator

Many electronic random number generators use oscillators, designed to change states at uncertain time points. When they are sampled at a low enough frequency, such that the phase of the oscillator can drift more than a full oscillation period in the average, the samples appear to be sufficiently random.

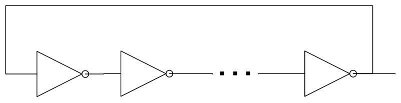

Figure 1. *Ring oscillator of* odd *number of inverters*

However, gates in standard logic libraries are designed to be insensitive to noise and supply voltage variations, so their switching point is quite stable. For sufficient drift millions of switching events have to take place, making the random number generation slow, and too sensitive to interference.

Flipping a Ring

In the quest for improvements we looked at ways to *stop* the ring oscillator, and make it to *maintain* the logic level at the time of the stop. The simplest option is to replace an inverter with an XOR gate, as shown on Figure 2. (Alternatively, a multiplexer could switch between a buffer and an inverter, or an inverter could be bypassed through a digital switch.)

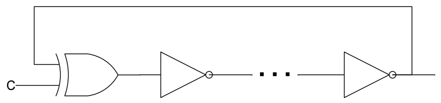

Figure 2. *Flip-able ring: Oscillator or Latch*

If the free input C of the XOR gate is connected to logic level 1, the gate behaves as an inverter of the signal on its other input, maintaining ring oscillator functionality. When applying logic level 0 to input C, the ring oscillator becomes a long (slow) latch circuit (consisting of a buffer and an even number of inverters for positive feedback). Note that a single XOR gate (of large delay) could work like this [23].

Experiments

The ring oscillator output settles to a stable logic level. However, the settling is not instantaneous, as it is seen in the oscilloscope trace below, captured on an FPGA prototype circuit (details will follow). The pulses *decay* gradually:

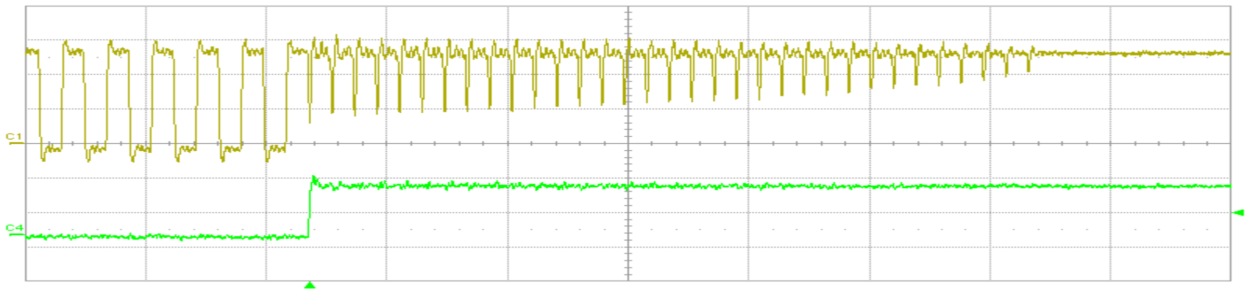

Figure 3*. Pulse decay after turning a ring oscillator to a latch*

The top signal trace on Figure 3 shows the output waveform before and after the level on the C input of the XOR gate changes (inverted lower trace).

If the flipping a negative feedback ring oscillator to a positive feedback latch occurs in the middle of a pulse, the duty cycle of the new pulse train at the output will be close to 50%. There will be no significant change in the shape of the signal for a long time (Figure 4 and Figure 5).

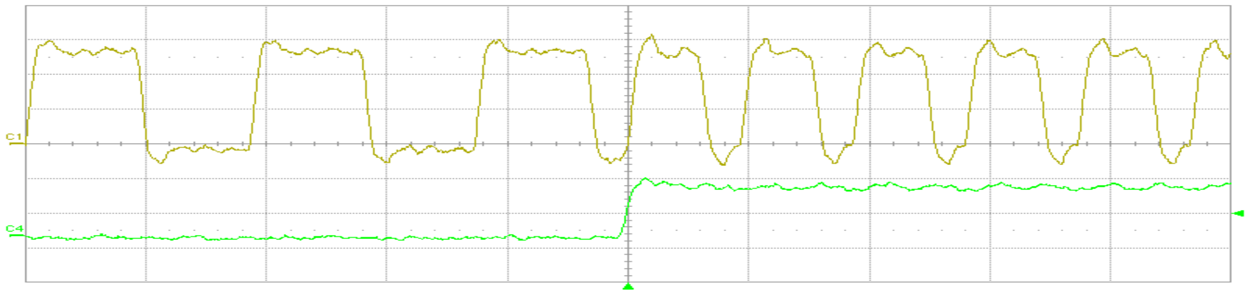

Figure 4. *Flip in the middle of a negative pulse*

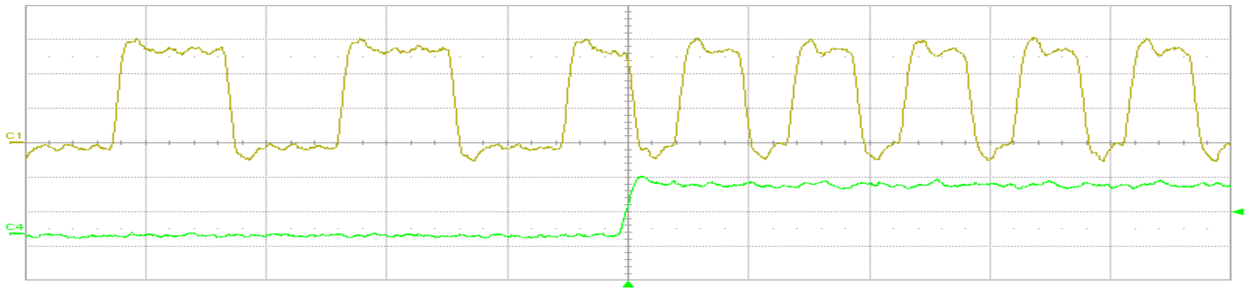

Figure 5. *Flip in the middle of a positive pulse*

The two oscilloscope traces show practically the same behavior after the feedback is switched over from negative to positive, regardless of the high or low output voltage level at the time of the flip.

When the new pulses are narrow, we see changes in their shape. They get gradually narrower and their amplitude drops (Figure 6). This phenomenon is called ***pulse decay***.

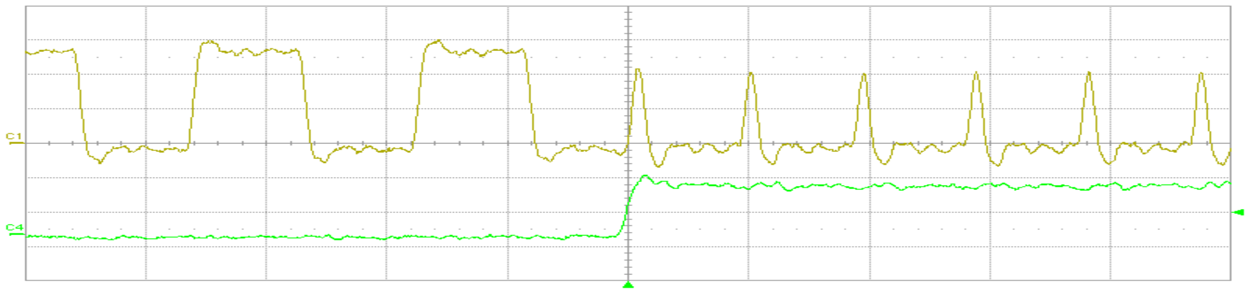

Figure 6. *Flip close to the edge of a pulse: pulses shorten and their amplitude drops*

This pulse decay is best visible, when the pulses are very short. The train of narrow (positive or negative) pulses stops, when they become too short for properly switching the next gate. Very narrow pulses disappear almost instantly, and the circuit output settles to a constant level (Figure 7).

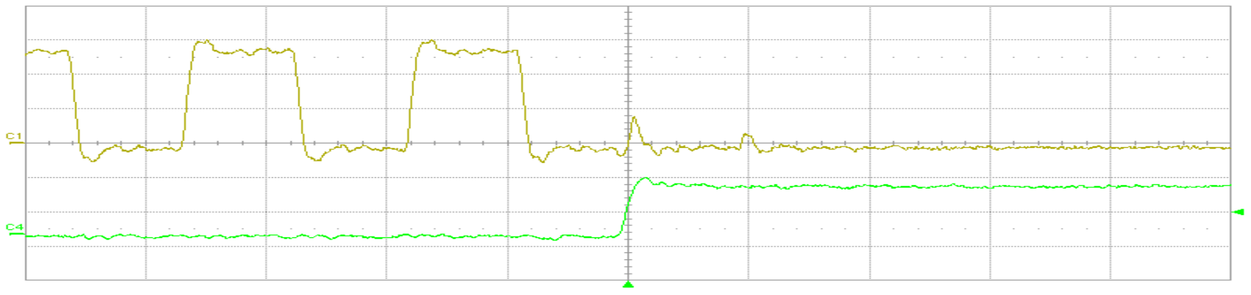

Figure 7. *Oscillation stops when the pulses get too short*

Pulse Decay

Note that when we see a sequence of *wide* pulses at the output of our circuit (e.g. on Figure 3), the pulses of the preceding and subsequent inverter in the ring are *narrow*. Therefore the observed decays of narrow and wide pulses (settling to different stable levels) are just seemingly different manifestations of the same phenomenon. Wide pulses become even wider in time, and eventually disappear, when the output settles to the logic level 1. Narrow pulses almost always settle to logic 0. One can also formulate this pulse decay behavior as the edges circulating on a ring of inverters "attract" each other.

The physical explanation of this phenomenon is simple: The inverters remain immediately after an edge has passed slightly charged in the direction of the state they were in before the edge passed, therefore a closely following second edge will need less charge to restore the charge level before the previous edge and can, therefore, propagate slightly faster. Over a few inverters the second edge gets too close to the first one, and the short pulse between them disappears.

Noise Effects

With ideal inverters and in the absence of noise the shape of the pulses in the flipped ring oscillator would remain constant and the oscillation would never stop. The feedback keeps everything repeated forever. However, external disturbances and internal noise vary the position of the pulse edges. Unsteady switch on/off times of the gates and the pulse decay also alter the pulse shapes.

Switching events take time because of the finite slope of the incoming signal and the limited slew rate and gain of the gates. During the transitional period the circuit noise randomly shifts the switching threshold, causing jitter: wobbling time positions of the edge of the pulses, at every inverter.

When stopping a ring oscillator at time points independent from the oscillation signal, randomness comes twofold into play: (a) the phase drift moves the stopping point randomly within the output pulses and (b) the settling waveform varies dependent on the noise.

Random Numbers from Oscillatory Metastability

Oscillatory metastability can be exploited for a random number generation in different ways: e.g. capturing the transitional waveforms or just measuring the time till settling (by counting pulses). The *stopping* waveform is quite regular, so counting the number of pulses either for a certain time period or till the circuit settles to a stationary state is adequate. The final logic level could be included, too.

Counting pulses in a *fixed time period* is easier and more robust, because the circuit occasionally keeps oscillating for a long time: if we flip an oscillator too often, the output signal has not always enough time to settle to a stable logic level (marked with an arrow on the waveform of Figure 8).

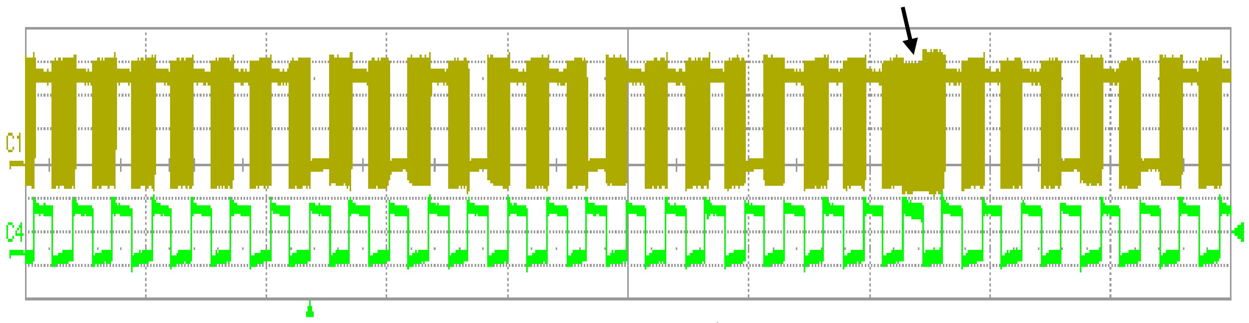

Figure 8. *Output logic levels at periodic start/stop on response to input C*

Design Tradeoffs

By choosing the ring length and the timer values properly, the random number generator will be stable and of high throughput. The exact values depend on the implementation platform.

- Longer ring oscillators accumulate more jitter, and so they provide more randomness in their oscillation period (proportional to the square root of the length of the ring, because of averaging effects). However, the oscillation period increases proportionally to the length of the ring, thus the resulting throughput decreases. The circuits of the largest throughput are the ones having the *shortest* still stable rings, and high frequencies of switching them on/off.
- **Stop-to-Restart** period
  - If it is too long, the throughput (the number of random bits a second) will be small, because the output often settles early to a logic level.
  - If it is too short, the output has rarely time to settle to a logic level. The circuit will provide little randomness, behaving similarly to a random number generator which periodically samples a faster ring oscillator.
- **Start-to-Stop** period
  - If it is long, the oscillation pulses drift too much, and after the stop we will too often see slow settling. The throughput fluctuates. In this case the circuit could benefit from some commonly used feedback based stabilization to keep the settling times reasonable, e.g. [18].
  - If it is short, the oscillation pulses do not drift much. The circuit is stable, but the randomness comes mostly from the settling process, the contribution of the oscillator drift is smaller.
  - Without the need for external stabilizer circuits, an optimally chosen, short start-to-stop period length represents the best tradeoff between complexity and throughput!
- We have to maintain the optimum stopping **position** within the ring pulses, roughly, for reasonable settling times. A typical value could be a start-to-stop period 9.1 times the ring period. The start-stop *clock* is best laid out as another ring oscillator, so the stopping point within the ring period stays constant: temperature and supply voltage affect the two rings similarly. Moreover, the jitter of this clock ring further improves the randomness.

Physical Background

The variability of the observed stopping (and starting) behavior of feedback rings may also contribute to the apparent good randomness of some recently published random number generators ([3], [4]), so understanding it is of theoretical, and of practical importance, as well.

The ring oscillator exhibits random phase drift (due to the noise), so an asynchronous stop occurs at a random position between the edges of the output signal. When the XOR gate is turned to a buffer (by a 0 in the input C), its output inverts keeping the remaining portion of the oscillator pulse inverted, and introducing an extra edge in the circulating waveform. We saw that the oscillation continues for a while, at virtually double frequency, because the output pulse has been cut into two. The newly established positive feedback keeps the pulse structure of the ring shifting circularly, under ideal conditions unchanged, forever. Because the delay of the flipped gate is usually slightly different in inverting and in non inverting state, the frequency doubling is only ***approximate***. Also, when these types of rings are built from noise sensitive components, there can be significant jitter of the edges, making frequency claims only valid in the average.

We observed that the settling time depends on the position of the flip relative to the edges of the output pulses, but with large random variations. If the flip occurs near to the edge of the output signal, we saw narrow output pulses, vanishing quickly (Figure 6), if the flip occurs in the middle of the output pulses, the circuit keeps on oscillating for a long time (Figure 4 and Figure 5).

Real life circuits exhibit phase jitter, making the pulse widths varying, when they pass through a gate. This edge jitter accumulates as the pulses propagate.

If a pulse gets too narrow, the earlier explained pulse decay phenomenon makes it progressively narrower as it propagates through the ring, and eventually it cannot make the output of a gate to reach a logic level. In a few transitions the pulse disappears, settling the output to a fixed level (Figure 3).

Random Walk Model

We make a number of observations:

Noise originates from many independent sources. Their cumulative effects (at every gate) are nearly of *normal distribution* (as the central limit theorem states: the mean of a large number of independent random variables is approximately normally distributed, under very general conditions [26]).

The noise shifts the pulse edges. The resulting *edge* jitter is, therefore, very close to *normally distributed* in time, too, when the noise is *small* enough, such that its affects on the transition function of the gates around the switching point is close to linear. (Note that at very strong interferences this linearity assumption does not hold, they could fully synchronize the ring, masking the small noise.)

The feedback of the ring keeps the shifted pulse edges in their new position (relative to an ideal oscillator of the same fundamental frequency as the ring), until the noise moves them again.

The width of pulses is the time between its edges. The ring feedback keeps the relative position of the edges, and so it keeps the altered pulse widths at their new values, until noise adds to or subtracts from them. Consequently, in a noisy ring the *pulse widths vary like at Gaussian random walks*, with a standard deviation of the steps (called pulse *width jitter*) $\sqrt{2}$ times the standard deviation of the edge jitter.

These observations hold until a (positive or negative) pulse gets too narrow. At this point the oscillation stops (within a few pulse periods).

Random Walk of Discrete Steps

The translation distance theorem of random walks (see e.g. [1] or [2]) states that for steps distributed according to any distribution with 0 mean and a finite variance σ^2 (not just at normal distribution), the root mean squared translation distance after n steps (S_n) satisfies the equation

$$\sqrt{E\left(S_n^2\right)} = \sigma\sqrt{n}$$

Accordingly, after several oscillation periods the drift of the pulse widths in a ring is approximately the same at continuous noise of standard deviation σ, as at noise pulses of the discrete values $\pm\sigma$. Thus, one can regard the pulse width jitter as drifting by the discrete values $\pm\sigma$, and model the pulse width drift of the ring as a random 1-dimensional walk, starting at the position corresponding to the flipping event within a pulse. At every other signal transition at each gate a random step is made left or right in the random walk model, representing the drift of the pulse width. The signal settles to a logic level if the random walk reaches its boundaries $[t_0, t_D - t_0]$, where t_D is the total ring delay (the period length), and t_0 is the minimum pulse width for properly switching the gates.

Settling Time

Note that the random walk model gives the same estimates as the accumulation of edge jitter via the Central Limit Theorem (summation formula), which also applies to stable oscillations in ring oscillators.

Considering discrete phase steps allows the application of the Boundary Theorem (see [2]). The Boundary Theorem of *unit step* random walks, states that if a and b are positive integers, then the expected number of steps until a one dimensional simple random walk starting at 0 first hits b or $-a$ is ab. The probability that this walk will hit b before $-a$ is $a/(a+b)$.

In our case a and b correspond to the distance of the stopping point from the two surrounding signal edges, in the units of the standard deviation of the jitter. The above formulae give the expected time until the flipped ring oscillator settles to 0 output level as $a^2b/(a+b)$, and the expected time until the stopped ring settles to 1 is $b^2a/(a+b)$. Their ratio is a/b.

If a or b is small, the signal of the stopped ring oscillator settles quickly to the logic level corresponding to the closer edge, although there is always a slight chance that the settling takes long, or even reaches the logic level corresponding to the opposite edge. If a and b are nearly equal, the expected settling time is $a^2/2$, which can be very large (if the jitter is small relative to the total delay).

Using time values in usual notation: let t_D denote the total ring delay, the period length; t_0 the minimum pulse width for properly switching the gates ($t_0 \ll t_D$), and t_j the standard deviation of the edge jitter. We saw that the standard deviation of the pulse width jitter $\sigma = \sqrt{2}\,t_j$. The position of the event flipping the ring, from the leading edge of the pulse it dissects, is t_P.

$$a+b=\frac{t_D-2t_0}{\sqrt{2}\cdot t_j}\approx\frac{t_D}{\sqrt{2}\cdot t_j},\quad a=\frac{t_P-t_0}{\sqrt{2}\cdot t_j}\approx\frac{t_P}{\sqrt{2}\cdot t_j},\quad b=\frac{t_D-t_0-t_P}{\sqrt{2}\cdot t_j}\approx\frac{t_D-t_P}{\sqrt{2}\cdot t_j}.$$

The expected time till a positive or negative pulse becomes too narrow, or too wide, and disappears is approximately

$$T_{narrow}=\frac{a^2b}{a+b}\approx\frac{t_P^2\left(t_D-t_P\right)}{2\cdot t_j t_D},\quad T_{wide}=\frac{ab^2}{a+b}\approx\frac{t_P\left(t_D-t_P\right)^2}{2\cdot t_j t_D}$$

If the flip occurs in the middle of a pulse, the expected time till the oscillation stops (at either logic level) is approximately

$$\frac{t_D^2}{16\cdot t_j}$$

Note that in the approximate equations we disregarded t_0, the time limit of the pulse decay effects, which affect very narrow pulses. This would make only insignificant differences in the analysis, when the oscillator is stopped at relatively far from the edges. A stopping point close to an edge, on the other hand, shows fast settling and practically no randomness.

Simulations

The equations derived from our random walk model above, agree principally to our measurements, depicted in Figure 3…Figure 8. Below we present very simple software simulations, based on this model.

The MATLAB macro below draws an animated figure, simulating the waveform of the positive feedback ring, primed with a pulse of a specified width (a snapshot of the oscillating state, before turning the oscillator to a latch). With this program we did not intended to build an accurate physical model of the circuits, but to build the possible *simplest* discrete model, which still demonstrated the pulse decay observed in the prototype FPGA circuits. The simulation waveforms are reasonably close to real life measurements, showing that the random walk model captures the most important characteristics of the stopped-ring circuit. (Commercial circuit modeling SW can be used for more accurate simulations.)

The model contains R (even) identical inverters, with the same switching delay D+0.5 for inputs with positive or negative slope. The noise effects are modeled with randomly changing the delay of every switching event to D or D+1, at equal probability. The initial condition is the stable (latch) state, artificially flipped over for a certain time period P. This corresponds to a free running negative feedback ring oscillator, when it is switched to a positive feedback ring, in the position in the last output pulse corresponding to P.

In real life circuits, when a pulse gets narrower than a certain width, the pulse decay phenomenon makes the resulting output pulse narrower than the input. This does not cause an immediate stop of the oscillation, but the pulses get progressively narrower, so within a few periods the oscillation likely stops. This condition is included in the simulation model in a very simple form: when a pulsewidth becomes less than the limit W (4), the pulsewidth gets reduced by 1.

Note that if the pulse decay function is removed from the model, we experience larger uncertainty of the settling time, but the principal circuit behavior remains the same: the output settles to a fix level in randomly varying time, dependent on the arming pulsewidth.

```
R = 4;   % ring length (even)
N = 499; % display time window
L = 1e6; % max simulation length
D = 11;  % gate delay (±0.5 edge shift = jitter)
P = 5;   % arming pulse width in [2,D-1]
W = 4;   % width of long enough pulses, which don't shorten

S = repmat(~mod(1:R,2), L,1);        % S(time,gate#) = STATE of gate at time
S(N-P:N-1,:) = ~S(N-P:N-1,:);        % inject initial pulse of width P
stop = 0; c = 0;

for t = N:L
    if t == stop, break, end
    for i = 1:R
        j = mod(i-2,R)+1;            % previous inverter in ring
        if S(t-D-1,j) == S(t-D,j)    % new output from input before delay
            S(t,i) = ~S(t-D,j);      % no state change at input -> no output jitter
        else
            S(t,i) = round(rand(1)); % edge in input: ±0.5 random delay
            c = c + 1;               % count edges
            stop = t + R*(D+1);      % no edge for a long time -> oscillation stopped
        end

        f = [find(S(t-(1:W),i)==S(t,i)); W+1];
        if f(1) > 1 && f(1) <= W     % f(1)-1 = width of last pulse
            S(t-1,i) = S(t,i);       % shorten pulse by 1
        end

        if i==2, hold on, end        % output traces shown in one figure
        plot(S(t-N+1:t,i) + 2*i-2)   % last N states of invert#i, shifted up
    end

    axis([0 N+1 -0.05 2*R-0.95])
    pause(1e-6);                     % small delay to display animation plot
    hold off
end
title(['Number of pulses ~= ' num2str(ceil(c/R/2))])
```

Figure 9 below shows a snapshot of the simulation window (containing the output waveforms of the 4 inverters in a ring), taken just after the ring output settled to a fixed logical level (the oscillations stopped).

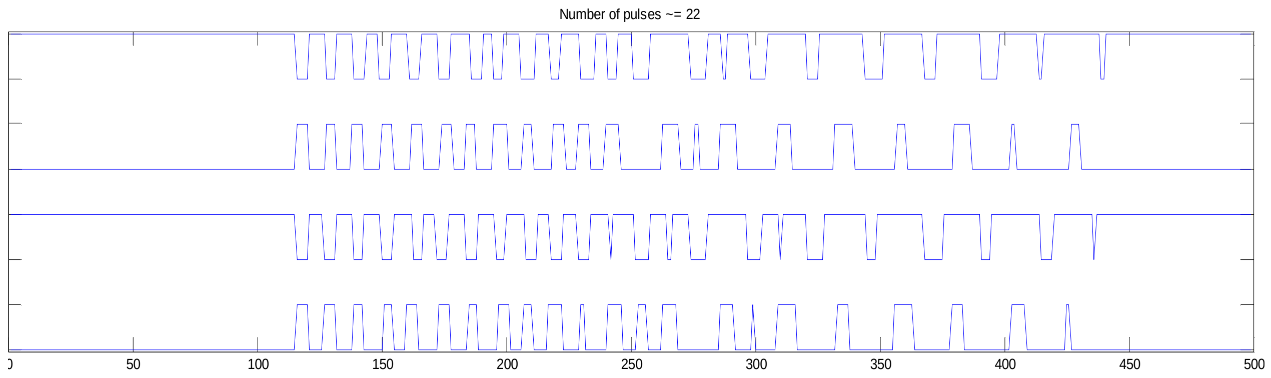

Figure 9. *Simulated waveforms of 4 inverters in a ring, when the oscillation stops*

Reference FPGA Implementation

We have seen that the settling time depends on the noise in the system, making it highly random. It can be captured by a circuit counting the edges of the output signal. This counter value and the somewhat random final output level together are the generated random numbers.

There are many implementation possibilities in terms of the size of the ring oscillator; and the lengths of the on- and off periods. In case of our test bed, an **Altera Cyclone III** FPGA board, we implemented dozens of variants. The highest throughput was achieved with a ring of 15 inverter-buffer pairs. The on-off clock was 390KHz, with 50% duty cycle. The counter values provided 6 bits of entropy, close to normally distributed. It corresponds to 390K*6 = **2.34 Mbit/s throughput** with using 40 logic elements.

For higher yield several such random number generator circuits can run in parallel. Our experiments did not show synchronization between individual rings on the same FPGA, but for safety they could include a varying number or types of buffer- or inverter gates, ensuring sufficiently different oscillation frequencies, which do not interfere with each other.

29-Inverter Ring with 98 KHz Start/Stop

This circuit was typical among the many we implemented. It was neither the fastest nor the most stable, although stable enough without any extra measures, and provided reasonably high throughput. The counter values (with the final logic state) were very close to normally distributed (Figure 10).

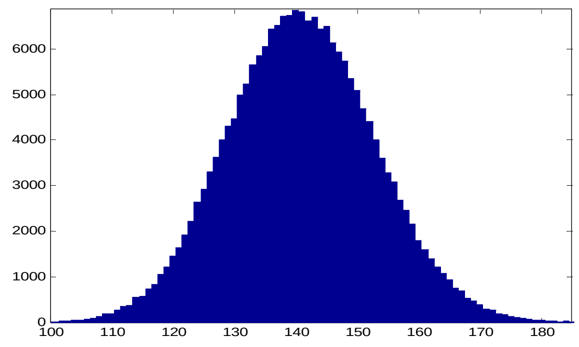

Figure 10. *Histogram of the counter values*

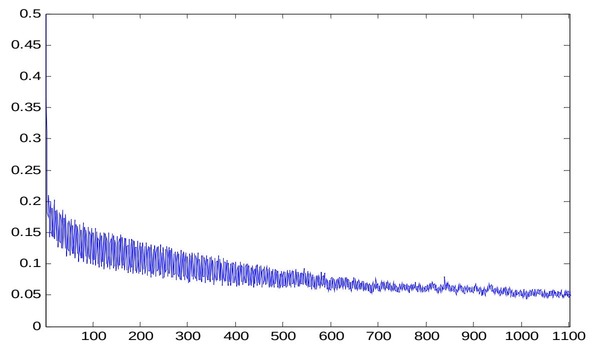

Figure 11. *Autocorrelation*

The raw entropy is 5.63 bits per sample, but there is 35% autocorrelation between neighboring counter values (Figure 11). The high autocorrelation is mostly the result of the non-uniform distribution. It can be radically reduced by keeping only the 5 LS bits of the edge counter (Figure 12).

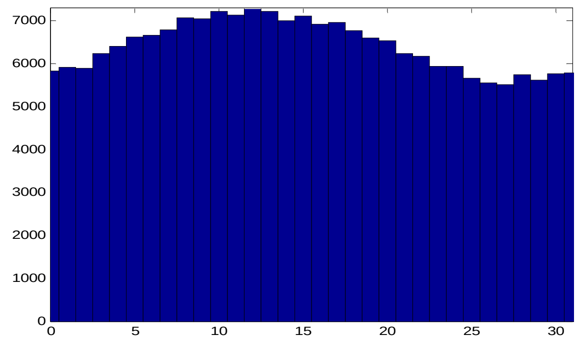

Figure 12. *Histogram of the 5 LS bits*

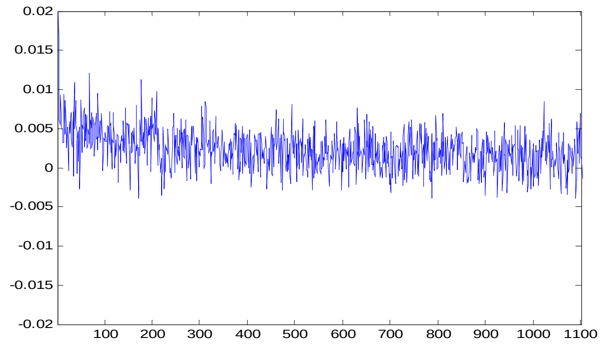

Figure 13. *Autocorrelation of the 5 LS counter bits*

The empirical entropy is 4.99 bit per counter, as expected, proving that the autocorrelation was due mostly to the (discarded) high order bits. The autocorrelation is reduced to less than 2% at all offsets (Figure 13), confirming 500 KHz throughput, by 5 bits entropy at every 10 µs.

Note that the generated random numbers still show somewhat non-uniform distribution, which, for most applications, has to be corrected by whitening, slightly more reducing the throughput.

Statistical Randomness Tests

We applied statistical tests to the generated random numbers. Many such randomness tests are published, for example [5], [6], [7], [8] or [10]. In [9] there is a survey. The most recent test suite, the NIST 800-22 Randomness tests, is provided as C-99 source code, to be compiled by the user. It confirmed the expected good quality of the generated random numbers: after whitening by general file compression software (about 1% reduction the size of the captured data files), ***all*** tests passed.

Practical Considerations

More implementation details are deferred to a follow-up paper. Here we only informally list the main ideas and our experiences.

Stability

The stability of the reference implementations of the discussed circuits was found satisfactory on FPGAs, when the free running oscillation period was sufficiently short. If, for whatever reason, longer free oscillation periods are chosen, well known and commonly used feedback techniques can be used to keep the circuits in their optimal operational conditions, (at varying environmental conditions or unstable supply voltages). See e.g. [18].

Experimental stabilizers were implemented by inserting a multiplexer in the ring to select one of several delay elements (different types of buffers or inverters, one or more in series) to be included in the ring. A simple circuit maintained a running count of the number of times the output signal settled too slowly or too quickly after flipping. If either of these occurred more often than a preset limit, the multiplexer advanced to another delay element. This way the optimum flipping point was maintained with the help of just two dozen extra logic elements.

Interferences

Strong internal or external signals might synchronize ring oscillators, and cause deterministic phase drift. Therefore, random number generator circuits practically always need some protection. Off the shelf, general solutions work well here, too. For example, duplicating the circuits and correlating the output sequences reveal the presence of strong interference, and the affected bits could be discarded.

Whitening

Near perfect randomness can be achieved with entropy amplification, also known as whitening. Here again well known, general techniques can be used: estimating the entropy and applying simple lossy compression to the corresponding degree, or compressing the output stream with a good lossless compression algorithm. (See, for example [11], [12] or [13].) The whitening step is mandatory for cryptographic applications, where it is achieved by seeding a secure pseudorandom number generator with imperfect physical random numbers. It is practically hashing based whitening.

Metastability in Ring Startup

Instead of investigating the (noise induced) uncertainties at *stopping* an oscillator, one can consider the complementary option: the metastable start up of a ring oscillator or latch. The gates are brought to a fixed state (e.g. all off), and let the circuit go by switching the gates to inverters, all in the same time.

A possible implementation is a ring of 2-input NAND (or analogously of NOR) gates, with ***all*** free inputs connected to a switch S. If S is at the logic level 0, all gate outputs are high (logic level 1). Turning switch S to logic level 1, makes all the NAND gates behave like inverters of their ring-connected inputs. After an initial chaotic (metastable) period the circuit behaves like a regular ring oscillator (with odd number of inverters) or settles to a stable state of alternating on/off outputs (at even number of inverters).

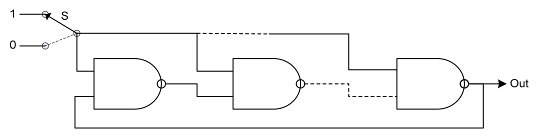

Figure 14. *Starting a ring oscillator with metastable initial state*

Physical Model

Immediately after arming the ring is metastable. All gates flip at roughly the same time, to another metastable state. Little parameter differences and the noise make the gates flip at slightly different times. These phase differences accumulate, and eventually separate the flipping events, resolving to a stable oscillatory state − or to a static state, dependent on the parity of the number of inverters in the ring.

As we saw at the analysis of the stopped ring oscillators, the pulses start up at the period length of the total delay in the ring, that is half of the period at which the ring would oscillate (in case of odd number of inverters). Because of the jitter, some of the pulses get narrower, others get wider. Again, as we discussed earlier, if a pulse gets too narrow (or too wide), it cannot properly switch the next inverter, making the pulse even narrower (inverted or straight) as it propagates through the ring. It eventually disappears. At that time the ring settles either to a stable oscillation mode or to fixed logic levels.

The drift always behaves as a one dimensional random walk, and so our physical model for the stopping of a ring oscillator applies to the ring startup, as well. The starting point is halfway between the stopping points ($a = b$), which makes the settling slow (thousands of pulses in our prototype FPGA circuits).

Practical Considerations

In practice the parameters of the gates are always slightly different, making the steps in the random walk different in positive and negative direction, at each gate. This makes the circuit to settle earlier, and mostly to the same state. With intentionally imbalanced implementations the settling can be made faster, while its exact waveforms remain strongly dependent on the noise.

Another way to speed up the settling process is making the initial pulses intentionally asymmetric by introducing synchronous switching events in the beginning. (It has similar effects to stopping a ring oscillator from an artificially set up snapshot of the oscillatory state.) This makes the circuit more complex, but faster, more predictable and stable under varying environmental conditions. See [24], [25].

Extracting Random Numbers

There are many ways the startup uncertainty could be turned to random numbers. The simplest (lowest throughput) solution is to count the number of signal edges at a gate, in a certain time period, because in the metastable state (of randomly varying length) pulses are generated at higher frequency. A higher throughput but more complex possibility is taking frequent high resolution snapshots of the gate outputs, and concatenating the resulting bit sequences to form long numbers, encoding the whole chaotic startup process. The hash values of these numbers provide very good quality random numbers.

Simulations

The MATLAB macro below draws an animated figure of the simulated startup process of rings of inverters. Again, we strived for the simplest discrete model, which still exhibits the observed randomness in startup (this time by using a circular sliding window, so arbitrary slow settling can still be handled). The simulation is based on the random walk model described earlier, with a pulse decay condition, simplified to its extreme: if a pulse width becomes 1, the pulse is removed.

Despite of the excessive simplifications, the simulation results exhibit similar pulse width variations and settling waveforms to real life circuits, showing that our model captured the most important aspects of the circuit behaviors. (Note again that commercial SW can model our circuits more accurately.)

```
R = 3;        % ring length
D = 5;        % gate delay (D+0.5 ± 0.5 edge shift = jitter)
N = 500;      % simulation time window (circular)
e = 0; f = 0; % last edge positions of gate1
sm = 0; cnt = 1;
% S(time,inverter#) = STATE of inverter at time
% initially fast erratic oscillation, slowly get to normal period ~2R*(D+0.5)
S = zeros(N,R);                       % all gates start from reset
for idx = D+1:1e9
    t = O(idx,N);                     % O(x,m) = mod(x-1,m)+1, user function
    u = O(t-D-1,N); v = O(t-D,N);     % new output from input before delay
    for i = 1:R
        j = O(i-1,R);                 % preceding inverter
        if  S(u,j) == S(v,j)
            S(t,i) = ~S(v,j);         % no state change at input -> no output jitter
        else
            S(t,i) = round(rand(1)); % edge in input: ±0.5 random delay
        end
        w = O(t-1,N);
        if S(w,i) ~= S(t,i) && S(w,i) ~= S(O(t-2,N),i)
            S(w,i) = S(t,i);          % remove too narrow pulse (±)
        end
    end
    if S(O(t-1,N),1)~= S(t,1)         % output edge: compute period
        if S(t,1), period = O(t-e,N); e = t;
        else       period = O(t-f,N); f = t; end
        if idx > 2*R*(D+1), sm = sm + period; cnt = cnt+1; % moving average of periods
        else sm = period; end
    end
    for i = 1:R                      % display all output traces
        if i==2, hold on, end
        plot(S(O(t+(1:N),N),i) + 2*i-2)
    end
    axis([0 N+1 -0.05 2*R-0.95])
    title(['Last Period: ' num2str(period) ',  Avg: ' num2str(sm/cnt)])
    pause(1e-6);                     % small delay to display animation plot
    hold off
end
```

On Figure 15 the sartup behavior of rings with odd number of inverters (3) are simulated. In this case, after an intial chaotic state, the circuit functions as a stable oscillator. (At even number of inverters constant output levels would be reached.) Three different simulated startup traces are shown, with 12, 6 and 10 narrow pulses before the oscillation stabilizes. For demonstration we chose large jitter (10%), which also visibly affects the stable oscillation mode (jitter).

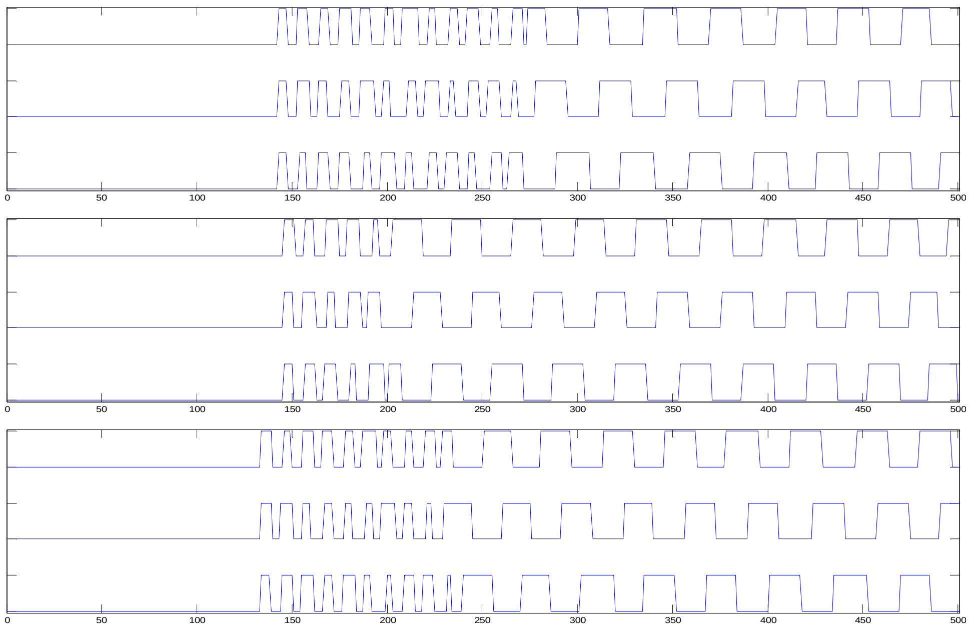

Figure 15. *Three sets of startup waveforms on the 3 inverters of a ring*

Conclusions

We investigated two fundamentally different ways, how oscillatory metastability can be used for stable, fast random number generation. The theoretical background of their working lies in a random walk model, driven by the circuit noise; and in the pulse decay phenomenon, which explains the final settling behavior of the proposed circuits.

Acknowledgement

The practical circuit implementations and measurements were performed by Mr. Tarun Gupta at Seagate Research. Without his thorough work this paper could not have been completed.

Literature

[1] Leo Breiman: Probability. Original edition published by Addison-Wesley, 1968; reprinted by Society for Industrial and Applied Mathematics, 1992. (See Sections 3.9, 12.9, and 12.10.)

[2] William Feller: An Introduction to Probability Theory and its Applications (Volume 1). 1968. ISBN 0-471-25708-7. Chapter 3.

[3] M. Dichtl and J. D. Golić: High-speed true random number generation with logic gates only, in Cryptographic Hardware and Embedded Systems - CHES 2007, ser. Lecture Notes in Computer Science (LNCS), vol. 4727. Springer, Sep 2007, pp. 45-62.

[4] J. D. Golić: New methods for digital generation and postprocessing of random data, IEEE Transaction on Computers, vol. 55, no. 10, pp. 1217-1229, Oct 2006.

[5] D. E. Knuth: Seminumerical Algorithms, Volume 1 of The Art of Computer Programming. Addison-Wessley, 1997.

[6] G. Marsaglia: A current view of random number generators. In Computer Science and Statistics: The Interface, pp 3–10. Elsevier Science, 1985.

[7] G. Marsaglia, A. Zaman: Monkey tests for random number generators. Computers and Mathematics, 26(9): pp 1–10, 1993.

[8] U. Maurer: A universal statistical test for random bit generators. Journal of Cryptography, 5(2): pp 89–105, 1992.

[9] T. Ritter: Randomness tests: A literature survey. http://www.ciphersbyritter.com/RES/RANDTEST.HTM, 1996.

[10] NIST Special Publication 800-22: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, August 2008: http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf. Source code: http://csrc.nist.gov/publications/nistpubs/800-22-rev1/sp800-22rev1.zip

[11] M. Blum: Independent unbiased coin flips from a correlated biased source: a finite state Markov chain. In 25th Ann. Symp. on Foundations of Computer Science, pp 425–433, 1984.

[12] M. Blum and S. Micali: How to generate cryptographically strong sequences of pseudo-random bits. SIAM J. Comput., 13(4):850–864, 1984.

[13] B. Chor and O. Goldreich: Unbiased bits from sources of weak randomness and probabilistic communication complexity. In 26th Ann. Symp. on Foundations of Computer Science, pp 429–442, 1985.

[14] L. Hars: Switching electronic circuit for random number generation. United States Patent 6,771,104. Filed July 25, 2002.

[15] L. Hars: Latching electronic circuit for random number generation. United States Patent 7,124,155. Filed July 25, 2002.

[16] L. Hars: Electronic circuit for random number generation. United States Patent 7,315,874. Filed March 15, 2004.

[17] L. Hars, M. Epstein: Method and apparatus for choosing a combination of logic for generating random numbers using a difference signal. United States Patent 7,302,458. Filed March 15, 2004.

[18] L. Hars: Method and apparatus of retaining maximum speed of flip-flop metastability based random number generators. United States Patent 7,356,551. Filed March 15, 2004.

[19] L. Hars: VLSI implementation of metastability-based random number generator using delay ladders, United States Patent Application 20050004959, Filed March 15, 2004.

[20] L. Hars: Feedback random number generation method and system, United States Patent Application 20040049525, Filed September 6, 2002.

[21] L. Hars: VLSI implementation of a random number generator using a plurality of simple flip-flops, United States Patent Application 20040267845, Filed March 15, 2004.

[22] M. Epstein, L. Hars, R. Krasinski, M. Rosner, H. Zheng: Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts. CHES 2003, pp 152–165, LNCS 2779, Springer-Verlag.

[23] L. Hars: SummerCon-2004 invited talk. http://www.hars.us/Papers/RandomTopics-SummerCon.ppt

[24] M. Dichtl, B. Meyer: Apparatus and Method for Generating a Random Bit Sequence, International Patent Application No.: PCT/EP2009/059837, Filing Date 7/30/2009.

[25] M. Varchola, M. Drutarovsky: New High Entropy Element for FPGA Based True Random Number Generators. CHES 2010. LNCS Volume 6225/2010, pp 351-365, Springer-Verlag.

[26] J. Rice: Mathematical Statistics and Data Analysis (Second ed.) 1995, Duxbury Press, ISBN 0-534-20934-3

[27] S. Walker, S. Foo: Evaluating metastability in electronic circuits for random number generation, VLSI, 2001. Proceedings. IEEE Computer Society Workshop on , vol., no., pp.99-101, May 2001

[28] C. Tokunaga, D. Blaauw, T. Mudge: True Random Number Generator With a Metastability-Based Quality Control, Solid-State Circuits, IEEE Journal of , vol.43, no.1, pp.78-85, Jan. 2008

[29] D. J. Kinniment, E. G. Chester: Design of an on-chip random number generator using metastability, Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European , vol., no., pp.595-598, 24-26 Sept. 2002

[30] D. C. Ranasinghe, D. Lim, S. Devadas, D. Abbott, P. H. Cole: Random numbers from metastability and thermal noise, Electronics Letters , vol.41, no.16, pp. 13-14, 04 August 2005

[31] T. Nakura, M. Ikeda, K. Asada: Ring oscillator based random number generator utilizing wake-up time uncertainty, Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian , vol., no., pp.121-124, 16-18 Nov. 2009

[32] F. Pareschi, G. Setti, R. Rovatti: Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems, Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.57, no.12, pp.3124-3137, Dec. 2010

[33] J. Holleman, S. Bridges, B. P. Otis, C. Diorio: A 3 μW CMOS True Random Number Generator With Adaptive Floating-Gate Offset Cancellation, Solid-State Circuits, IEEE Journal of , vol.43, no.5, pp.1324-1336, May 2008

[34] T. Sugiura, Y. Yamanashi, N. Yoshikawa: Demonstration of 30 Gbit/s Generation of Superconductive True Random Number Generator, Applied Superconductivity, IEEE Transactions on , vol.21, no.3, pp.843-846, June 2011