Chaotic generation of PN sequences : a VLSI implementation

Citation for published version (APA):

Dornbusch, A., & Pineda de Gyvez, J. (1999). Chaotic generation of PN sequences : a VLSI implementation. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999, ISCAS '99, 30 May - 2 June 1999, Orlando, Florida* (Vol. 5, pp. 454-457). Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/ISCAS.1999.777607>

DOI:

[10.1109/ISCAS.1999.777607](https://doi.org/10.1109/ISCAS.1999.777607)

Document status and date:

Published: 01/01/1999

Document Version:

Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

CHAOTIC GENERATION OF PN SEQUENCES: A VLSI IMPLEMENTATION

Andrew Dornbusch and José Pineda de Gyvez

Department of Electrical Engineering

Texas A&M University, College Station, TX, 77843

Email: dornbush@ee.tamu.edu, gyvez@pineda.tamu.edu

Abstract

Generation of repeatable pseudo-random sequences with chaotic analog electronics is not feasible using standard circuit topologies. Component variation caused by imperfect fabrication causes the same divergence of output sequences as does varying initial conditions. By quantizing the output of a simple chaotic difference equation and using delayed feedback, we are able to construct a circuit which has properties both of randomness and repeatability. Simulation results show sequence lengths in excess of 10,000 bits with cross correlations less than 0.02 for our architecture. The system has been implemented using AMI 1.2 μ m CMOS technology, and digitally using AMD MACH complex programmable logic devices (CPLDs). Circuit measurements are in agreement with our theoretical findings.

1. Introduction

Binary pseudo-noise (PN) sequences have several uses in communications, a notable one being use as frequency-spreading codes for direct-sequence spread-spectrum (DSSS) [1]. Standard PN sequences have limited properties in both low cross-correlation between sequences, and large sequence[1]. Conventional sequence generation is done using maximal length (m-) sequences, which are generated using linear feedback shift registers. These shift registers use feedback to implement Boolean algebraic polynomial operations[2] and generate unique maximum-length sequences. Sequence length is determined by shift register length n , but cannot exceed $2^n - 1$ bits in length. Discussions on sequence generation can be found in [1,2,3]. By decimating and combining pairs of m-sequences it is possible to generate Gold sequences[1]. Gold sequences improve upon the cross-correlation properties of m-sequences at a cost of increased auto-correlation. Ultimately, the number of PN sequences that can be generated with a linear feedback shift register is limited.

In this paper we present a system that generates chaotic PN sequences derived from a 1-D discrete map, specifically the tent map. Several theoretical studies have been carried out in the past exploring the inherent properties of chaos to generate very large sequences[4,5]. While numerical simulations are straightforward, an actual VLSI implementation of a circuit is a challenge due to the unavoidable process variations which result in sequence deviation due to the sensitivity to initial conditions inherent in the chaotic system[6].

2. Iterated Chaotic Functions

A *piecewise-linear* function is a function which consists of a finite number of contiguous linear segments. A simple example is the tent map, shown in (1).

$$f(x) = \begin{cases} \frac{x}{p}, & x \leq p \\ \frac{1-x}{1-p}, & x > p \end{cases} \quad (1)$$

The mapping has a constant coefficient p , referred to as the peak value. This is the point at which the mapping reaches its maximum output value, shown at $x = p = 0.7$ in Figure 1. Above and below this value the function decreases linearly to reach zero at $x = 0$ and $x = 1$.

Figure 1. Tent Function Mapping

An *iterated function* is a discrete-time function generated by recursive evaluation of some function mapping (2).

$$x_{n+1} = f(x_n) \quad (2)$$

A continuous-time system requires a minimum of three dimensions in order to exhibit chaotic behavior[7]. A discrete-time system is, by definition, not continuous, and it is possible to achieve chaos with a single-dimension iterated function system.

It has been shown[8,9] that it is reasonably straightforward to construct a discrete-time tent-map circuit using either voltage-mode or current-mode circuitry. When iterated, the tent map function generates sequences which are qualitatively stochastic and exhibit *sensitive dependence upon initial conditions* (SDIC)[6]. An example is shown in Figure 2. Given two identical sequence generators with close but distinct initial conditions, the sequences generated will diverge exponentially. Similarly, given two identical initial conditions with close but distinct parameters, the sequences will again diverge exponentially. For an iterated tent mapping, given any two signals with initial conditions differing by as little as 0.01%, their corresponding sequences diverge completely in less than 20 cycles. To generate pseudo-noise sequences useful for DSSS, a chaotic iterated function must generate a consistent and repetitive sequence of some arbitrary length. This consistency must be preserved over time within one device, and across different devices from different production lots. SDIC becomes a hindrance under these conditions. Any noise in a circuit or

variations from fabrication will cause the chaotic iterated sequences to diverge.

Figure 2. Sensitive Dependence Upon Initial Conditions

3. Implementation of Chaotic PN Generator

A. Quantized Tent Circuit

In order to produce repetitive sequences which retain the character of the original chaotic sequences, we modify the tent mapping by quantizing the input and output values. The input to the circuit is now generated with a digital-to-analog converter (DAC), and the output is sampled with an analog-to-digital converter (ADC). This creates a circuit that is insensitive to process variations below a certain calculable threshold. By adjusting the number of bits on the input and output, it becomes possible to tailor sensitivity of the circuit to arbitrary conditions. This quantized tent circuit is shown in Figure 3.

Figure 3. Quantized Tent Circuit

The tent map function used (1) is mathematically a two-to-one mapping. This results in error multiplication with an inherent precision loss. In order for there to be no possibility of an errant mapping, the precision of the output ADC must be less than the precision of the input DACs divided by the errors' gain factor. The output ADC must then have at least one bit less resolution (bitwise) than the input DAC.

Transistor matching in standard CMOS processes rarely exceeds 2%, even with good layouts[10]. Doubling this error to account for two inputs yields potential 4% mismatch. We must select our input precision so that a 4% variation in either direction will not cause an input signal to pass a bit boundary. Assuming a factor of two increase in the DAC output error due to the tent mapping, the output precision must be such that one half bit variation does not exceed 8%. This indicates three bit precision on the input DAC. The ADC output can then have at most two bits of resolution. To simplify the circuitry, it will be reduced further to a single bit (a threshold comparator).

If we return to the mapping of Figure 1 and place a threshold at an output value of 0.5 then we see two intersect points at

$$t_{low} = \frac{peak}{2}, \quad t_{high} = \frac{peak}{2} + 0.5$$

From this we see that, rather than generating an actual piecewise-linear transfer function in our circuit, it is possible to directly compare the input value with two comparators; one set at half the peak value, the other offset from the first by $0.5 V_{full}$. The circuit can now be implemented as in Figure 4.

Figure 4. Mixed-signal Implementation of Tent Circuit

If the transfer function of the single-bit output system is generated (Table 1) and examined, it can be seen that the system can be implemented entirely using digital logic with the transfer function

$$output = \begin{cases} (S0 + S1) \oplus S2, & threshold = 00 \\ \overline{S1} \oplus S2, & threshold = 01 \\ (S0 \times S1) \oplus S2, & threshold = 10 \\ \overline{S2}, & threshold = 11 \end{cases} \quad (3)$$

where S2, S1, and S0 indicate the state input bits

Table 1. Transfer Function of Reduced Tent Map

State Input	Threshold Input			
	00	01	10	11
000	0	0	0	0
001	1	0	0	0
010	1	1	0	0
011	1	1	1	0
100	1	1	1	1
101	0	1	1	1
110	0	0	1	1
111	0	0	0	1

This can be implemented logically as shown in Figure 5. The circuit has now been reduced to the point where it requires only 24 transistors to implement (4 ea. For NAND & NOR, 2 for invert, 10 for 4-input MUX).

Figure 5. Digital Implementation of Tent Circuit

At this point there is a mismatch in data widths between the input and output. This mandates some source of additional data input.

Latching and delaying the output and using the delays as additional inputs provides two bits. The third input is provided from an external source. This circuit is shown in Figure 6.

Figure 6. Tent Circuit with Delayed Feedback

B. Ring Structure

With three internal register bits, it is not possible for the circuit generate sequences longer than 2^3 , or 8 bits in length. However, by using the output of one tent map circuit (cell) as the input bit to another, we can cause each cell to perturb the next. Assembling a number of cells into a ring, we can construct a circuit with any desired level of complexity, and which will be able to generate arbitrarily long sequences. The generated sequences, though finite in length, have some characteristics of chaos – they are qualitatively stochastic, and they are sensitive to the initial conditions. They are not, however, sensitive to process variations. A ring consisting of N cells yields a 3N-bit system, shown in Figure 7.

4. System Level Simulation Results

Table 2 summarizes results obtained from software simulations of our system along with conventional m- and Gold-sequence systems. The top section shows maximum auto- and cross-correlation values for standard m-Sequences with lengths of 127, 2047, and 32767 bits. In the center are the values for Gold-sequences of the same length. The bottom row shows results for the chaotic sequences generated by the new architecture. The quantity of sequences generated exceeds that of the comparable linear sequences, and the cross-correlation properties are excellent as well. Auto-correlation properties do not equal those of the other sequences; the auto-correlation behavior of the m-sequence is theoretically optimal and cannot be beaten. Gold-sequences are constructed from combinations of decimated m-sequences; the combination does not always yield the optimal auto-correlation behavior of m-sequences but the lessened behavior has proven sufficient over many years of field use.

Figure 7. Ring Structure.

Table 2. Simulation Comparison of PN Generators

	Autocorrelation	Crosscorrelation
m-Sequence		
255-bit	0.008	0.373
4095-bit	0.00049	0.3436
32767-bit	0.000031	0.06638
Gold-Sequence		
255-bit	0.185	0.185
4095-bit	0.0447	0.0447
32767-bit	0.00784	0.00784
Chaotic-Sequence		
127-bit	0.188	0.141
2048-bit	0.0781	0.0357
32768-bit	0.01495	0.00394

Figure 8. Example Measured Auto- and Cross-Correlation

The best known use of DSSS is in code-domain multiple-access (CDMA) communications. CDMA systems operate with multiple users occupying the same frequency band[1]. Signal isolation is achieved by the using spreading sequences that have low cross-correlation properties. When multiple signals are received, only the correct user signal is despread. Other signals with low cross-correlation properties (good signal orthogonality) are rejected. Shown in Figure 8 are examples of auto- and cross-correlations from a 2048-bit sequence. The graphs display the correlation values for all offset values.

5. Monolithic Implementation of Chaotic PN Generator

The PN generator was implemented at the transistor level. A ring of ten cells was arranged in a loop, with the input of each cell by default the output of the previous cell. A switch was included to allow alternate feedback loops which include XOR gates. Addition of the selectable XOR breaks the rotational symmetry of the loop and provides for an even greater range of output sequences. The complete circuit was simulated in Hspice. Transient analysis runs showed complete agreement between the sequences generated by the transistor circuits and by the software simulator. Monte-Carlo analysis was performed and showed stability under process variations for a range of 10%. Figure 9 shows an example output sequence of the circuit in the top window operating at a clock rate of 1 MHz. The lower window shows a closeup of one rising edge with the entire window representing a 10ns interval. The vertical scale for both plots is 5 V.

Figure 9. Circuit Output, Monte-Carlo Simulation

The chip as designed is a full-static circuit. The internal logic was constructed using minimum size N-channel transistors paired with P-channel FETs widened sufficiently to match g_m s. The layout was done using MAGIC, and the ICs were fabricated through MOSIS using the AMI 1.2 μ m double-metal double-poly process. Final active circuit area on the die was 1300 μ m x 1100 μ m. Four packages received from MOSIS were used for testing. Figure 10 displays the chip's microphotograph.

Figure 10. Photomicrograph of IC

Since the logic is full-static, the power dissipation is a linear function of frequency. The average dissipation for the four devices was 1.7 mW/MHz with a power supply of 5V. The maximum operating frequency was measured to be an average of 27.4 MHz for the four test chips.

6. CPLD Measurement

The final version of the tent map cell can also be modeled using Boolean logic, which enables implementation using standard programmable logic. The complete ten-cell circuit required 80 latches and consumed approximately 65% of a MACH221 CPLD manufactured by Advanced Micro Devices (AMD). The circuits were initialized and then monitored to measure auto- and cross-correlation properties experimentally. Measurements corresponded with theory to the limits of the accuracy of the sampling system, and a plot of an example sequence is shown in Figure 11.

7. Conclusions

A circuit for generation of long pseudo-random chaotic sequences is given. This circuit is insensitive to standard process variations for current CMOS technologies, and was implemented in a 1.2 μ m CMOS process on a 2200 μ m x 2200 μ m die.

Figure 11. Fabricated Chip Output, 1 MHz Clock

The architecture was also tested using commercially produced programmable logic devices. The measured results for the auto- and cross-correlation were precisely those predicted.

The sequences generated show low cross correlation properties with a very large signal space, which is desirable for spread spectrum communications.

References

- [1] R.L. Peterson, R.E. Ziemer, D.E. Borth, *Introduction to Spread Spectrum Communications*, Prentice Hall, Upper Saddle River, NJ, 1995
- [2] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, Laguna Hills, CA, 1982
- [3] J.G. Proakis, M. Salehi, *Communications Systems Engineering*, Prentice Hall, Englewood Cliffs, NJ, 1994
- [4] S. Papadimitriou, G. Pavlides, A. Bezerianos, T. Bountis, "Chaotic Systems of Difference Equations for Real-Time Encryption," *1995 IEEE Workshop on Nonlinear Signal and Image Processing*, pp. 5.9.1-5.9.5, 1995
- [5] S. Papadimitriou, A. Bezerianos, T. Bountis, "Chaotic Real-Time Encryption Using Systems of Difference Equations with Large Parameter Spaces," *IEEE Circuits and Systems*, 1996, pp. 566-569
- [6] R.L. Devaney, *An Introduction to Chaotic Dynamical Systems*, 2nd Ed., Addison-Wesley Publishing Company, New York, 1989
- [7] M.P. Kennedy, "Three Steps to Chaos - Part I: Evolution," *IEEE Transactions on Circuits and Systems*, Vol. 40, No. 10, pp. 640-656, 1993
- [8] M. Delgado-Restituto, "Diseño de Circuitos Integrados no Lineales," Ph.D. Dissertation, Universidad de Sevilla, 1996
- [9] A. Rodríguez-Vázquez, J.L. Huertas, A. Rueda, B. Perez-Verdu, L.O. Chua, "Chaos from Switched-Capacitor Circuits: Discrete Maps," *Proceedings of the IEEE*, Vol. 75, No. 8, pp.1090-1106, 1987
- [10] K.R. Laker, W.M.C. Sansen, *Design of Analog Integrated Circuits and Systems*, McGraw-Hill, New York, 1994