FPGA Implementation Of Chaotic Cellular Automaton with Binary Synchronization Property

Ioana Dogaru, Radu Dogaru
University “Politehnica” of Bucharest
ETTI, Dept. of Applied Electronics and Info. Eng.,
Natural Computing Laboratory
Bucharest, Romania
e-mail: ioana_d70@yahoo.com, radu_d@ieee.org

Cristian Damian
M.A.I.
Inspectoratul General pt. Situatiu de Urgenta
Bucharest, Romania
e-mail: damian_crstn@yahoo.com

Abstract— In this paper a FPGA implementation of a cellular automaton defined by rule 101 and it’s negate is presented. Such HCA (hybrid cellular automata) was previously proved to behave as a pseudo noise sequence generator with the additional binary synchronization property and with demonstrated capabilities in reducing the overall complexity of image transmission systems. The raster scan is replaced by a chaos scan provided by HCA. That allows implementation a progressive image compression and spread spectrum transmission with the advantage in low complexity implementation. A VHDL description of the HCA counter is provided for an arbitrary number of cells n. The resulting hardware complexity is several tens of times lower than needed for implementation of other pseudo noise generators reported in the literature.

Keywords: cellular automata, chaos, chaos synchronization, FPGA, pseudo noise sequence

I. INTRODUCTION

Most of the communication systems require low power consumption and low complexity of their implementation solutions. Applications like remote transmission, surveillance, sensors networks for different kind of systems, are very good candidates for an efficient hardware realization. In video transmissions the classical approach uses an orderly exploration of the scanned information. An obvious example is the raster scan in analog television. In [1] the regular counter associated with a raster scan was replaced with a chaotic counter addressing the same pixels. To recover the information properly a similar chaotic counter running synchronously with the one in the transmitter is needed. In effect, uncorrelated consecutive image pixels are sent over the channel instead of correlated nearby pixels in the case of raster scan. Such methods can be easily extended to other applications, particularly those employing multi-dimensional sensors (e.g. arrays of sonar, infrared and other type of sensors etc.) with a high degree of correlation among geometrically close sensors. The effect of this replacement is impressive, in that low complexity compression, ciphering and distributed spectrum are obtained solely as an effect of this modification. In a recent paper [2] a software simulator is presented demonstrating the capabilities of a compression and transmission system based on binary synchronization of chaos in a hybrid cellular automata (HCA) based on rule 101 and it’s negated (also called HCA101). This HCA addresses pixels in an image such that a certain geometrical distance exists between consecutive pixels. This counter is a binary automaton providing addresses in the range [0, 2^n-1], where n is the number of cells. A detailed description of the theory behind binary synchronization in a certain class of HCA is provided in [3]. There we identified the conservative elementary CA with rule ID=45 (as well as its “twins” ID=75, ID=89 and ID=101) as the best CA among all elementary CA with 3 neighbors exhibiting both chaotic behavior (in fact pseudo-random sequence generator) and binary synchronization (one bit of the suffices to recover the entire state in the receiver). Although the period of the counter is sometimes smaller than the maximal when all CA cells obey the same rule, in [4] we provided a solution to this problem. We demonstrated that by properly choosing some of the cells with a negate ID (in the case of ID=101, its negate is ID=154) the result is a HCA still possessing the binary chaos synchronization property, but with a period of the counter approaching the maximal possible.

This paper presents for the first time a scalable hardware implementation (in FPGA) of the previously mentioned chaotic counter as well as various simulations confirming its functional capability and giving information about implementation complexity. It is the first hardware implementation reported, while our previous cited papers with regards to the new HCA random number generator dealt with software models. The remainder of this paper is organized as follows: The principle of binary chaos synchronization in HCA is reviewed in Section II, while in Section III the FPGA implementation and the simulation results for different width lengths (or representation precision, or number of cells) n are presented. Concluding remarks are summarized in Section IV.

II. HYBRID CELLULAR AUTOMATA AS CHAOTIC COUNTERS WITH SYNCHRONIZATION PROPERTY

The discrete-time dynamics of the HCA chaotic counter is given by the next equation, which applies synchronously to all cells (a cell is identified by an index \( i \in \{1,2,..,n\} \)):

\[
x_i^T(t+1) = m_i \oplus \text{Cell}(x_{i-1}^T(t), x_i^T(t), x_{i+1}^T(t), ID)
\]

(1)
Where the upper index “T” stands for the transmitting CA counter; \(\oplus\) is the logical XOR operator and \(\text{Cell}(u_1, u_2, u_3, ID)\) is a Boolean function with 3 binary inputs \((u_1, u_2, u_3)\). In our case, the Boolean function corresponds to rule ID=101 in Wolfram’s notation [5].

In its binary representation, the most significant bit of ID corresponds to the cell output when \([u_3, u_2, u_1] = [1, 1, 1]\). A periodic boundary condition is assumed i.e. the leftmost cell \((i=1)\) is connected to the rightmost one \((i=n)\). The binary mask vector \(m = [m_1, m_2, ..., m_n]\) is optimized [6] for any odd counter size up to \(n \leq 29\) an optimal value of such that \(r = N / 2^n \rightarrow 1\) (maximal cycle length). In the case \(n=21\) the optimal mask vector is \([00000000000000000010100000010]\) or 1282 in decimal.

When this mask is used \(N = 2097151 = 2^{21} – 1\) states, i.e. only one pixel of the array is not addressed. A table with optimal masks for different numbers of cells is given in [2][6].

As shown in [3][6] and confirmed by simulations of our FPGA implementation, synchronization between Rx (receiver) and Tx (transmitter) counters occurs in a finite number of cycles and is independent on the two different starting states of the two counters, provided that a clock recovery scheme is used in the receiver. Various schemes for digital clock recovery may be used, i.e. the one in [8].

Another direction of our research detailed in [3] was to provide some measure of chaos (randomness) in finite automata. In [3] such a measure was proposed, observing that in a “chaotic counter”, unlike in a “normal counter” the jumps (in terms of Hamming distance) between consecutive binary vector states becomes \(n/2\) on average instead of 1. Therefore for any arbitrary counting cycle \(L_j\) of length \(L_j\), a scattering coefficient \(S_j\) is defined by averaging the Hamming distances between all consecutive binary vector states in that cycle:

\[
S_j = \frac{1}{nL_j} \sum_{k=1}^{L_j} \sum_{i=1}^{n} |x_i(k) - x_i(k-1)| \tag{2}
\]

where \(k\) is the time index of consecutive states in the cycle “j”. A degree of chaos \(\lambda_j\) is defined such that it becomes maximum if \(S_j = 0.5\) and zero for the extreme, non-chaotic cases of both fixed points and period 2 cycles (with \(S_j = 0\) and \(S_j = 1\) respectively):

\[
\lambda_j = 1 - 2|S_j - \frac{1}{2}| \tag{3}
\]

The degree of chaos \(\lambda_j\) may be regarded as qualitatively similar to the Lyapunov exponents usually used in continuous-state systems to characterize chaotic behaviors. It ranges between \(\lambda_j = 0\) for low complexity dynamics (fixed points, periodic limit cycles where consecutive states are close to each other), to \(\lambda_j = 1\) in good random number generators. Numerical simulations provided that \(\lambda_j = 1\) for the proposed HCA chaotic counter. Note that in [3] we analyzed the HCA against the famous “rule 30” cellular automaton that was previously proposed as a good random number generator and found that HCA based on rule 101 has better randomness properties. In a companion paper submitted to this conference [10] it was concluded that HCA chaotic counter is better in any respect than logistic functions while having a much compact FPGA implementation complexity.

III. FPGA IMPLEMENTATION OF THE CHAOTIC COUNTER

A modern solution for developing efficient hardware systems is FPGA implementation. FPGAs offer many advantages, and some of those are referring in reducing design time, power consumption, and flexibility.

For a practical implementation we used a development board from XESS using the FPGA Spartan II circuit (XC2S50-5-TQ144) from Xilinx [9]. The software used is Webpack from Xilinx with the advantage that it is a free software, appropriate for the low complexity of our implementation.

In a first step, a VHDL description of the HCA chaotic was generated, with the advantage to obtain a generic code that can be particularized for any desired \(n\) (width length or register size). The associated symbol of the implementation model entity for the chaotic counter is presented in “Fig. 1”. For the next examples we use a width length of \(n=7\) data input bits. The code is provided with the possibility to change the width length \(n\) and the different optimal masks given in [2][6]. A slightly changed version of the chaotic counter provides a receiver chaotic counter (in addition it has an input receiving the synchronization signal that replaces the feedback loop from cell \(i=1\)).

![Figure 1. Chaotic counter entity](image)

We simulated the basic chaotic counter module using ModelSimXE simulation program. In “Fig. 2” and in “Fig. 3” snapshots from the “Simulate Post- Place and Route VHDL Model” are presented.

![Figure 2. Snapshot for the simulation of the chaotic counter (n=7 bits)](image)
Using a random initial state with \( n=7 \) data bits we obtain at “DOUT” (the output sequence, corresponding to all CA cells) random sequences, as seen in Figs.2 and 3.

The “View Synthesis Report” tool reported the following complexity results, summarized in Table I. The implementation of the chaotic counter is very efficient in the terms of using the FPGA resources, even a small and cheap FPGA like the one of our choice suffices to implement a HCA with up to \( n = 1536 \) cells (one CA cell associated to one slice). Our selected Device is 2s50lq144-5 from Spartan Family with 768 slices, 1536 number of slice flip-flops, 1536 4-input LUTs (look-up table).

In the following, the binary synchronization property will be demonstrated in a ModelSim Post-Place-and-Route simulation. In order to emulate a transmitter-receiver system, the schematic module represented in Fig.4, was designed. Modules transmitter and respectively the receiver are chaotic counters previously defined as VHDL code and they are connected via a single information bit (DOUT(0)) as seen in “Fig. 4”. In order to determine the moment of synchronization a circuit comparing the states of both counters and providing the "SYNC" signal is introduced. In a binary synchronization system, the entire state of the automaton (i.e. \( n \) bits) is recovered in the receiver while sending serially over the transmission channel only one information bit. The process require a certain amount of clock cycles, depending on the different initial states of transmitter and receiver automata. In the case of the HCA counting automaton a superior bound for the synchronization time is 1.6\(^n\) cycles.

For the hardware implementation of the chaotic counter it is necessary to use four slices from the total number of 768 slices, requiring 0.52% of the available resources. For comparison, in [7] the authors implement a pseudo-number generator based on the cellular automata with \( n=5 \) cells using rules 90 and 150. Their design is specified in VHDL targeted on a XC3S400 FPGA from Spartan 3 family (with similar basic resources like our 2S50 chip). The resources for their hybrid cellular automaton were reported as 2% of the available slices (82 out of 3,584). Total number 4 input LUTs (look-up table) is 140 out of 7,168, which represent 1% of the available resources. Scaling to the lower number of gates in our chip (50000 instead of 400000) it follows that their implementation requires about 10 times more resources. Therefore, our solution provides a very efficient implementation for a chaotic counter with even more functional properties (the HCA in the above mentioned work has no binary synchronization property) while we used a FPGA circuit having less resources.

When the “SYNC” signal has logical “High” value, both systems (transmitter and receiver) are synchronized. In “Fig. 5” the moment of synchronization is emphasized (vertical line after 22 \( \mu \) seconds). As seen, after that moment at DOUT (a bus with \( n \) wires representing the state of the chaotic counters) exhibits the same sequence of states (synchronization) in both transmitter and receiver.

| FPGA Resources Used for the Implementation of the Chaotic Counter Module |
|-----------------------------|-------------------|------------------|
| Number of Slices           | 4 out of 768      | 0.52%            |
| Number of Slice Flip Flops | 7 out of 1536      | 0.45%            |
| Number of 4 input LUTs     | 7 out of 1536      | 0.45%            |
| Number of bonded IOBs      | 8 out of 96        | 8%               |
| Number of GCLKs            | 1 out of 4         | 25%              |
The tool “View Synthesis Report” reports on the complexity of the Tx-Rx system implementation as seen in Table 2. From those results it is remarkable to observe the low complexity in terms of FPGA resources used for the system implementation. Simulations for a large size \( n = 9 \) are provided in “Fig. 6”. Using the “Post-place and route” tools (where propagation delays of gates and other parasitic effects are included) allowed us to determine empirically (by changing the clock periods of the stimuli) the maximum operating frequency. For the above specified chip that was about 20 Mhz.

**TABLE II. FPGA RESOURCES USES FOR THE IMPLEMENTATION OF THE SYSTEM**

<table>
<thead>
<tr>
<th>Selected Device</th>
<th>2s50tq144-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slices</td>
<td>9 out of 768</td>
</tr>
<tr>
<td>Number of Slice Flip Flops</td>
<td>15 out of 1536</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>14 out of 1536</td>
</tr>
<tr>
<td>Number of bonded IOBs</td>
<td>16 out of 96</td>
</tr>
<tr>
<td>Number of GCLKs</td>
<td>2 out of 4</td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

An FPGA implementation for a novel PN-sequence generator with binary chaos synchronization property was presented. Such systems may be efficiently used in various communications and ciphering applications and its functional effectiveness was already demonstrated in [1]. The regular cellular automata structure makes it an ideal match for FPGA implementations, and as shown in this paper the resulting implementation complexity for the same register size \( n \) is about 10 times smaller compared to another HCA reported in [7] and about 50 times smaller than an implementation of a logistic map PN-sequence generator reported in [11]. Post place and route simulations confirmed the binary synchronization effect, not present in other PN-sequence generators but very important to build low complexity PN-sequence acquisitions schemes. Such performances recommend our HCA implementation as an ideal Intellectual Property module (IP) with further applications in mobile and low power sensing-nodes of sensors networks or other low power communication applications.

**REFERENCES**