This paper presents a multiplierless based FPGA implementation for six different chaotic Pseudo Random Number Generators (PRNGs) that are based on: Chua, modified Lorenz, modified Rössler, Frequency Dependent Negative Resistor (FDNR) oscillator, and other two systems that are modelled using the simple jerk equation. These chosen systems can be employed in high speed applications because they don’t utilize any hardware multiplier. The proposed PRNGs have been implemented using VHDL, synthesized on Xilinx, using the FPGA: XC5VLX50T, and tested using the NIST statistical suite. Furthermore, a comparison has been established between the performance of all the PRNGs, regarding the implementation area, speed, and the statistical randomness quality. It has been found that the modified Lorenz PRNG has the best randomness quality and the best hardware performance as it can pass all the NIST tests while operating at 298.597 MHz and utilizing only 0.23% and 0.62% from the FPGA’s slice registers and Look-Up-Tables (LUTs) respectively.

© 2019 Elsevier GmbH. All rights reserved.

1. Introduction

Random number generators have been used in a wide variety of applications, such as cryptography [1,17,18], Monte Carlo simulators [2], and Artificial Neural Networks [3]. Random number generators are classified into two main families: True Random Number Generators (TRNGs) and Pseudo Random Number Generators (PRNGs) [4].

The TRNGs are used to generate random numbers by exploiting physical processes, such as the thermal noise or jitter. However, the TRNGs are not employed in encryption applications because they cannot generate two identical sequences of random numbers for the ciphering and deciphering operations respectively. The only possible solution for this problem is to store the generated random numbers in a memory. Furthermore, the TRNGs suffer from a low bit generation rate, in the range of hundreds of kilobits per second; hence, they are not used in high speed applications [4].

On the other hand, the PRNGs are based on deterministic functions that continuously update the PRNG's current state leading to a sequence of random states. Moreover, the PRNGs are implemented in the digital domain using computers or FPGAs; therefore, they can be employed in high speed applications. However, in the digital world there is only a finite number of states that can be produced. For example, in case the PRNG’s states are represented using k-bits, the PRNG will produce up to $2^k$ different random states and then will repeat the same sequence of states again. To overcome this problem, the PRNG designers must increase the periodic length as much as possible. Accordingly, a lot of research has been conducted to find the suitable state transition matrices that can deal with many state bits, $k$, and maintain the maximum periodic length $2^k$ as well [4,5].

The Chaos based PRNGs are characterized by having an infinite periodic length; however, since they are solved using numerical methods on computers, the finite precision of the digital domain will lead to a finite periodic length. This problem can be solved in FPGAs by simply increasing the number of fractional bits at the expense of degrading the implementation area and speed [6]. Furthermore, the chaotic PRNGs are very sensitive to the initial condition; thus, they are used in many encryption applications where the initial condition is set according to the encryption key.

The Lorenz [7] and Rössler [8] are considered the most famous chaotic systems and they have been used in a lot of cryptographic applications [9,10]. However, these systems are based on non-linear multiplications that degrade the system's performance, regarding the number of utilized hardware resources and speed. Therefore, newer versions, called the modified Rössler and modified Lorenz, were presented in [11,12]. These modified versions...
do not require the hardware expensive multipliers that limit the system’s performance. Moreover, there are a lot of other chaotic systems [13,14] that do not require hardware multipliers; and therefore, are considered attractive for small area and high-speed applications.

This paper presents a design methodology for realizing multiplierless chaotic PRNGs that are based on Chua [13], modified Rössler [11], modified Lorenz [12], Frequency Dependent Negative Resistor (FDNR) oscillator [11], in addition to other two systems that capture the double scroll and Rössler dynamics using the simple jerk equation [11,12]. A performance comparison for all the mentioned PRNGs have been presented in this paper, regarding the NIST [15] statistical randomness quality, implementation area, and speed. The paper is organized as follows. Section 1 presents the introduction. Section 2 presents the FPGA implementation of the chaotic PRNGs. Section 3 presents the synthesis and NIST results. Section 4 presents the conclusion.

2. The FPGA implementation of the chaotic PRNGs

Chaotic systems are modelled by differential equations that can be solved using numerical methods. The Euler’s method is usually employed in high speed PRNGs where the approximation error is not a matter of high concern [16]. Hence, in this paper, the chaotic systems are solved using Euler’s method as shown in (1) where $X_i$ is the system’s variable, $X_i$ is the system’s differential equation, the subscript $i$ is the iteration index, and $h$ is the step size.

$$X_{i+1} = X_i + h \cdot X_i$$ \hspace{1cm} (1)

Eq. (1) is realized using a loadable accumulator as shown in Fig. 1. The hardwired shifter (gates), which doesn’t utilize any hardware resource, is used to perform the multiplication by the step size $h$ where $h$ is equal to $2^{-10}$. The Reset signal and the multiplexer are used to set the initial point, $X_0$, of the Register. The 2’s complement fixed point scheme is used in this paper where the fractional size is adjusted according to the ranges of the variables. On the other hand, the amount of integer bits is adjusted based on the comparison of the ranges of the variables. In this paper, the parameters and the step size ($\alpha, \beta, m_0, m_1, h$) are set to $(2^4, 2^4, -2^{-3}, 2^{-2}, 2^{-4})$ respectively; hence, all the multiplications are performed using hardwired shifters. Furthermore, the truncation rounding scheme is performed after any multiplication in order to maintain the same N fractional bits. Moreover, based on MATLAB simulations, the ranges of the system’s variables are $-2.36 \leq X \leq 2.36$, $-0.54 \leq Y \leq 0.54$, and $-3.73 \leq Z \leq 3.73$. Therefore, the sizes of the loadable accumulators of the $X, Y,$ and $Z$ variables are $(3+N)$-bits, $(1+N)$-bits, and $(3+N)$-bits respectively. The hardware realization of the system is presented in Fig. 2.

The most complex part of the circuit is the realization of (2-a) because of the comparisons that are available in the function $h(X)$. From (2-d), in case the absolute value of $X$ is less than $1 \times 10^{-6}$, $X$ will be multiplied by $m_0$ otherwise $X$ will be multiplied by $m_1$. However, $m_0$ is negative; therefore, the multiplication by $m_0$ cannot be performed using a hardwired shifter only because an adder is needed to compute the 2’s complement of the shifted value. To optimize the design, this adder is merged with the last adder/subtractor that is available in the path that leads to $X$. Accordingly, (2-a) and (2-d) are realized as follows. First a comparator is used to compare $|X|$ with $1 \times 10^{-6}$ using the 3 Most Significant Integer Bits (MSBs) of $X$, and then the comparator’s output, $S$, is used to control a multiplexer that selects between $-m_0X$ and $m_1X$. The output of this multiplexer will have at most 1 integer bit. Moreover, according to the comparator’s output and the sign of $X$, the multiplexer’s output will be added to $m_0$ or $m_1X$. These two options can be represented as 0000 using 1 integer bit and 3 fractional bits only. Therefore, only a 4-bit adder/subtractor, which is controlled by the sign bit of $X$, is needed because the remaining fractional bits will remain unchanged. Finally, the second $(1+N)$-bit adder/subtractor, which is controlled by $S$, is used to finalize the computation of (2-a). Thus, the critical path that generates $X$ includes two logic gates, one multiplexer, one 4-bit adder/subtractor, one $(1+N)$-bit adder/subtractor, in addition to the $(3+N)$-bit loadable accumulator.

The hardware realization of (2-b) and (2-c) is simple. (2-c) is realized using the loadable accumulator only. On the other hand, (2-b) is more complicated and requires another $(3+N)$-bit adder and $(3+N)$-bit subtractor. Thus, the critical path that generates $Y$ includes one $(3+N)$-bit adder, one $(3+N)$-bit subtractor, and the $(1+N)$ loadable accumulator.

2.2. PRNG2: modified Rössler

The modified Rössler system, which doesn’t include any non-linear multiplication as in the case of the original Rössler system, is modelled as shown in (3) where $A, C, \alpha_1,$ and $\alpha_2$ are the system’s parameters [11].

$$\dot{X} = -Y + Z$$ \hspace{1cm} (3-a)

$$\dot{Y} = X + AY$$ \hspace{1cm} (3-b)

$$\dot{Z} = BZ - CZ$$ \hspace{1cm} (3-c)

$$B = \begin{cases} \alpha_1, & X \geq 1 \\ \alpha_2, & X < 1 \end{cases}$$ \hspace{1cm} (3-d)

The system’s variables and the step size ($A$, $C$, $\alpha_1$, $\alpha_2$, $h$) are adjusted to $(2^{-1}, 1, 2^3, 0, 2^{-5})$ respectively. By simulating the system on Matlab it has been revealed that the ranges of the variables are $1.64 \leq X \leq 1.58$, $-2.14 \leq Y \leq 1.31$, and $0 \leq Z \leq 3.38$. Therefore, the sizes of the loadable accumulators of the $X, Y,$ and $Z$ variables are $(2+N)$-bits, $(3+N)$-bits, and $(3+N)$-bits respectively. The block diagram of the system is presented in Fig. 3.
The hardware realization of (3-a) and (3-b) is straightforward. The critical path that generates $Y_i$ includes one (2 + N)-bit adder in addition to the (3 + N)-bit loadable accumulator. Similarly, the critical path that leads to $X_i$ includes a (3 + N)-bit adder and the (2 + N)-bit loadable accumulator.

On the other hand, the hardware realization of (3-c) and (3-d) requires a comparator that adds extra overhead to the system. Fortunately, the small number of integer bits in $X$ has been exploited to speed up the comparison between $X$ and 110 by using only two logic gates that deal with the 2 MSBs of $X$. The comparator’s output will be multiplied by $a_i$. $X$ using AND gates only. In this way, if $X \geq 1$, $X$ will be multiplied by $a_i = 2^2$; otherwise, $X$ will be multiplied by 0. The rest of the design is straightforward. The critical path that leads to $Z_i$ includes three logic gates, one (4 + N)-bit subtractor and the (3 + N)-bit loadable accumulator.

2.3. PRNG$_{3}$: modified Lorenz

The modified Lorenz system, which doesn’t utilize non-linear multipliers as the original Lorenz system, is modelled as shown in (4) where $a$, $b$, and $c$ are the system’s parameters. [12]

$$
\begin{align*}
\dot{X} &= a(Y - X) & (4-a) \\
\dot{Y} &= -sgn(X)(Z - b) & (4-b) \\
\dot{Z} &= sgn(X)X - cZ & (4-c) \\
sgn(X) &= \begin{cases} 
+1, & X \geq 0 \\
-1, & X < 0 
\end{cases} \\
\end{align*}
$$
The system's variables and the step size \((a, b, c, h)\) are adjusted to \((2^{-1}, 2^{-1}, 2^{-2}, 2^{-4})\) respectively. The ranges of the system's variables are \(-0.45 \leq X \leq 0.71, -1.25 \leq Y \leq 1.01, \) and \(0.012 \leq Z \leq 1.01\). Therefore, the sizes of the loadable accumulators of the \(X, Y,\) and \(Z\) variables are \((1 + N)\)-bits, \((2 + N)\)-bits, and \((2 + N)\)-bits respectively. The block diagram of the system is presented in Fig. 4.

The critical path that leads to \(X_i\) includes one \((2 + N)\)-bit subtractor in addition to the \((1 + N)\)-bit loadable accumulator. Similarly, the critical path that leads to \(Y_i\) includes a \((1 + N)\)-bit adder/subtractor, which is controlled by the sign bit of \(X\), in addition to the \((2 + N)\)-bit loadable accumulator. On the other hand, the critical path that leads to \(Y_i\) can be optimized by taking into consideration that \(b\), which is equal to 0.510, can be represented using 1 fractional bit and 1 integer bit only. Hence, \((Z - b)\) is computed using a 3-bits subtractor that deals with the 2 integer bits of \(Z\) in addition to its most significant fractional bit. The rest of the fractional bits of \((Z - b)\) will be connected directly to the corresponding fractional bits of \(Z\). In this way the critical path that leads to \(Y_i\) will include a 3-bit subtractor in addition to the \((2 + N)\)-bit loadable accumulator that performs addition or subtraction based on the MSB of \(X\).

### 2.4. PRNG4: FDNR based chaotic oscillator

The FDNR based chaotic system is modelled using (5) where \(a, b, c, h\), and \(k\) are the system's parameters[11].

\[
\dot{X} = \frac{1}{\epsilon_1} (Y - X - Z) \tag{5-a}
\]

\[
\dot{Y} = \frac{1}{k} \left( X - Y - \epsilon_2 Y - f(Y) \right) \tag{5-b}
\]

\[
\dot{Z} = \beta X \tag{5-c}
\]

\[
f(Y) = a \begin{cases} Y - 1, & Y \geq 1 \\ 0, & Y < 1 \end{cases} \tag{5-d}
\]

In this paper, the parameters and the step size \((a, b, c, h, k)\) are adjusted to \((2^6, 2^5, 2^5, 1^2, 2^3)\) respectively. Accordingly, the system is solved using Euler's method as shown in (6). The choice of equalizing \(h\) and \(\epsilon_1\) has simplified the calculation of the variable \(X_{i+1}\) as shown in (6-a).

\[
X_{i+1} = Y_i - Z_i \tag{6-a}
\]

\[
\dot{Y}_{i+1} = \dot{Y} \tag{6-b}
\]

\[
Y_{i+1} = Y_i + 2^{-5} \left( X_i - Y_i - \dot{Y}_i - f(Y_i) \right) \tag{6-c}
\]

\[
Z_{i+1} = Z_i + 2^{-10} X_{i+1} \tag{6-d}
\]

\[
f(Y) = 2^6 \begin{cases} Y - 1, & Y \geq 1 \\ 0, & Y < 1 \end{cases} \tag{6-e}
\]

The ranges of the system's variables are \(-1.89 \leq X \leq 1.67, -2.15 \leq Y \leq 1.17, -0.62 \leq Z \leq 0.29,\) and \(-0.467 \leq \dot{Y} \leq 0.44. Therefore, the sizes of the loadable accumulators of the \(X, Y,\) and \(Z\) are \((2 + N)\)-bits, \((3 + N)\)-bits, \((1 + N)\)-bits, and \((1 + N)\)-bits respectively. The block diagram of the PRNG is presented in Fig. 5.

In this PRNG, (6-a) cannot be realized using the loadable accumulator because \(X_{i+1}\) doesn't depend directly on \(X_i\). Hence, the critical path that leads to \(X_i\) includes a \((3 + N)\)-bits subtractor and a multiplexer. On the other hand, the hardware realization of (6-c) and (6-d) is straightforward and is realized using the loadable accumulators only.

The most complex part of the circuit is the comparison in (6-e). Like the previous circuits, the small number of integer bits in \(Y\) has been exploited to optimize the comparator by examining only the 3 MSBs of \(Y\). Furthermore, the small number of integer bits in \(Y\) has reduced the size of the subtractor that computes \((Y_i - I_{10})\) to a 3-bit subtractor only because the fractional bits of \(Y_i\) and \((Y_i - I_{10})\) are the same. The hardware realization of (6-e) is completed by multiplying \((Y_i - I_{10})\) by the comparator's output using AND gates only. The rest of the hardware realization of Eq. (6-b) is straightforward. The critical path that leads to \(Y_i\) includes a 3-bit subtractor, 1 logic gate, a \((3 + N)\)-bit adder, \((9 + N)\)-bits subtractor, and the \((1 + N)\)-bits loadable accumulator.

### 2.5. PRNG5: simple equation modelling for the double scroll-like dynamics

A simple double scroll chaotic system is modelled using (7) where \(a\) is the system's parameter[12]. In this paper the values of \(a\) and the step size \(h\) are set to \(2^{-1}\) and \(2^{-6}\) respectively.

\[
X = -a \left( \dot{X} + X + X - \text{sgn}(X) \right) \tag{7}
\]

Fig. 4. The block diagram of the modified Lorenz system.
The ranges of the system’s variables are $-2.62 \leq X \leq 2.62$, $-1.74 \leq X \leq 1.74$, and $-1.2 \leq X \leq 1.2$; therefore, the sizes of the loadable accumulators are $(3 + N)$, $(2 + N)$, and $(2 + N)$-bits respectively. The block diagram of this PRNG is shown in Fig. 6. The part of the circuit that computes $(X - \text{sgn}(X))$ is optimized by taking into consideration that the integer part of $X$ has three bits only. Therefore, only a 3-bit subtractor is used to calculate $(X - \text{sgn}(X))$. The output of $\text{sgn}(X)$ has three integer bits where the LSB is always equal to logic ‘1’ while the two most significant bits are equal to the sign bit of $X$. The critical path of the circuit includes two $(3 + N)$-bits adders in addition to the $(2 + N)$-bits loadable accumulator.

2.6. PRNG6: simple equation modelling for the Rössler system

The dynamics of Rössler system can be captured with a simple differential equation as shown in (8) where $a_1$ and $a_2$ are the parameters [11]. In this paper, $(a_1, a_2, h)$ are set to $(2^3, 0, 2^{-4})$ respectively.

\[ X = (X + BX + X) \] (8-a)

\[ B = \begin{cases} a_1, & X \geq 1 \\ a_2, & X < 1 \end{cases} \] (8-b)

The ranges of the variables are $-2.75 \leq X \leq 2.94$, $-3.54 \leq X \leq 3.93$, and $-10.56 \leq X \leq 6.67$; hence, the sizes of the loadable accumulators are $(3 + N)$, $(3 + N)$, and $(5 + N)$-bits respectively. The hardware realization of the system is presented in Fig. 7. The comparison in (8-b) is realized as explained previously in PRNG4 using three logic gates only. Then, (8-b) is computed by multiplying the comparator’s output with $a_1X_i$ using AND gates only. Accordingly, the critical path of the circuit includes three logic gates, two $(7 + N)$-bit adders, and the $(5 + N)$ loadable accumulator.

3. Performance analysis and comparison of the PRNGs

The proposed PRNGs have been implemented using VHDL and synthesized using Xilinx 13.1 ISE where the target FPGA is XC5VLX50T. The number of utilized LUTs and registers, and the maximum operating frequency are summarized in Table 1 where the proposed PRNGs have been realized using 20 fractional bits.
Furthermore, Table 1 includes the performance of other two chaotic PRNGs that were presented in [9] and [10]. As shown in Table 1, the 3rd PRNG, which is based on the modified Lorenz system, has the best performance as it can operate at 298.597 MHz and utilize only 65 out of 28,800 and 178 out of 28,800 slice registers and LUTs respectively. According to XILINX XPOWER analyzer, the power consumption of all the PRNGs is approximately 0.5 W. Fig. 8 shows the post synthesis simulation results of the modified Lorenz system while operating at maximum speed.

Furthermore, Table 1 includes the performance of other two chaotic PRNGs that were presented in [9] and [10]. As shown in Table 1, the 3rd PRNG, which is based on the modified Lorenz system, has the best performance as it can operate at 298.597 MHz and utilize only 65 out of 28,800 and 178 out of 28,800 slice registers and LUTs respectively. According to XILINX XPOWER analyzer, the power consumption of all the PRNGs is approximately 0.5 W. Fig. 8 shows the post synthesis simulation results of the modified Lorenz system while operating at maximum speed.

Furthermore, a VHDL test bench has been designed to simulate the proposed PRNGs and generate 24 random bits per clock cycle, which is suitable for image stream ciphering applications where each image pixel consists of 24-bits. In case the PRNG has three variables, as in PRNG1,2,3,5 and 6, the 24 bits are captured from the

Table 1
FPGA synthesis results of the chaotic PRNGs.

<table>
<thead>
<tr>
<th>System</th>
<th>PRNG1</th>
<th>PRNG2</th>
<th>PRNG3</th>
<th>PRNG4</th>
<th>PRNG5</th>
<th>PRNG6</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPGA</td>
<td>XC5VLX50T</td>
<td>200</td>
<td>224</td>
<td>178</td>
<td>246</td>
<td>175</td>
</tr>
<tr>
<td>Slice LUTs</td>
<td>67</td>
<td>68</td>
<td>65</td>
<td>87</td>
<td>67</td>
<td>71</td>
</tr>
<tr>
<td>Slice Registers</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>DSP blocks</td>
<td>233.672</td>
<td>295.508</td>
<td>298.597</td>
<td>219.214</td>
<td>271.209</td>
<td>215.785</td>
</tr>
</tbody>
</table>

Table 2
NIST statistical results for the PRNGs.

<table>
<thead>
<tr>
<th>System</th>
<th>PRNG1</th>
<th>PRNG2</th>
<th>PRNG3</th>
<th>PRNG4</th>
<th>PRNG5</th>
<th>PRNG6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequency</td>
<td>0.98</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Block Frequency</td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>0.99</td>
<td>0.81</td>
<td>1</td>
</tr>
<tr>
<td>Cumulative Sums</td>
<td>0.98</td>
<td>0.97</td>
<td>0.94</td>
<td>0.91</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Runs</td>
<td>1</td>
<td>0.01</td>
<td>1</td>
<td>0.99</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>Longest Run</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.98</td>
<td>0.98</td>
<td>0.97</td>
</tr>
<tr>
<td>Rank</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>1</td>
</tr>
<tr>
<td>FFT</td>
<td>0.99</td>
<td>1</td>
<td>1</td>
<td>0.99</td>
<td>0.67</td>
<td>0.98</td>
</tr>
<tr>
<td>Nonoverlapping Template</td>
<td>0.95</td>
<td>0</td>
<td>0.96</td>
<td>0.95</td>
<td>0.94</td>
<td>0.95</td>
</tr>
<tr>
<td>Overlapping Template</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.99</td>
<td>0.97</td>
<td>1</td>
</tr>
<tr>
<td>Universal</td>
<td>0.99</td>
<td>0.16</td>
<td>1</td>
<td>1</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Approximate Entropy</td>
<td>1</td>
<td>0.08</td>
<td>1</td>
<td>1</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Random Excursions</td>
<td>0.957</td>
<td>0.875</td>
<td>0.963</td>
<td>0.964</td>
<td>0.983</td>
<td>0.966</td>
</tr>
<tr>
<td>Random Excursions variant</td>
<td>0.957</td>
<td>0.875</td>
<td>0.981</td>
<td>0.947</td>
<td>0.983</td>
<td>0.966</td>
</tr>
<tr>
<td>Serial</td>
<td>1</td>
<td>0.74</td>
<td>0.98</td>
<td>0.99</td>
<td>0.99</td>
<td>0.98</td>
</tr>
<tr>
<td>Linear Complexity</td>
<td>1</td>
<td>0.99</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.98</td>
</tr>
</tbody>
</table>
8 LSBs of each chaotic variable. On the other hand, in case the PRNG has four variables, as in PRNG4, the 24-bits are captured from the 6 LSBs of each chaotic variable. For each PRNG, the NIST tests are applied on a stream of 100 million random bits that is divided into 100 sequences. The NIST suite includes 15 major tests where some of these tests includes several subtests. In case there are more than one subtest, as in the case of the non-overlapping template, only the worst subtest result will be taken into consideration.

The NIST statistical results are summarized in Table 2 for all the PRNGs. These results show the percentage of sequences that have passed the tests. The passing criteria for each test is approximately 96%. As shown in Table 2, only the 3rd PRNG can pass all the NIST tests. Thus, the 3rd PRNG (Modified Lorenz) is the most efficient PRNG as it has the best hardware performance and the best statistical randomness quality.

Finally, another VHDL test bench has been implemented to plot the chaotic attractors of the proposed PRNGs as shown in Fig. 9.

4. Conclusion

A design methodology for implementing multiplierless chaotic PRNGs on FPGAs has been presented in this paper. The proposed methodology has been used to implement six different chaotic PRNGs. It has been found that the modified Lorenz PRNG has the best hardware performance, because it requires a small number of additions and subtractions that can be performed in parallel. Furthermore, the modified Lorenz has the smallest number of integer bits; therefore, the implementation area and delay of the adders and subtractors are lower than the other PRNGs. The six PRNGs have been realized using VHDL, with the same number of fractional bits, and implemented on XC5VLX50T. The synthesis and NIST statistical results have showed that only the modified Lorenz can pass all the NIST tests and achieve the best hardware performance compared to the other PRNGs.

Declaration of Competing Interest

The authors of this paper declare that there is no conflict of interest.

References


