Optimization of the PLL Based TRNG Design Using the Genetic Algorithm.
Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer

To cite this version:

HAL Id: ujm-01575708
https://hal-ujm.archives-ouvertes.fr/ujm-01575708
Submitted on 5 Oct 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Optimization of the PLL Based TRNG Design Using the Genetic Algorithm

Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer
Univ Lyon, UJM-Saint-Etienne
CNRS, Laboratoire Hubert Curien UMR 5516
F-42023, SAINT-ETIENNE, France
(oto.petura, ugo.mureddu, nathalie.bochard, fischer)@univ-st-etienne.fr

Abstract—Phase-locked loop (PLL) based true random number generator (TRNG) is very well suited for security applications using field programmable gate arrays (FPGAs) because most of FPGAs feature hardwired PLL blocks. PLL based TRNGs (PLL-TRNGs) are easy to implement and do not require manual placement or routing. The design of such TRNGs is also highly portable within the same device family. This is not the case in many other TRNG designs. However, the design of a PLL-TRNGs is not a trivial task. Due to many PLL parameters, which need to be fine tuned to achieve required security and speed requirements, an exhaustive design space exploration is practically not feasible. Thus, the designers are required to go through many trial and error cycles of manual parameter tweaking and the results are still not guaranteed to be optimal. In this paper, we use a genetic algorithm (GA) based optimization to generate a suitable configuration of the PLL-TRNG, such that it is secure and reaches high output bit rate. GA optimization allows to take into account physical limits of the PLL, such as input/output frequency, and maximum voltage controlled oscillator (VCO) frequency, which avoids invalid configurations and reduces the development time. The method has proven to be very efficient and it significantly reduces the design time without compromising the security. All the presented configurations were tested on recent FPGA families and the statistical quality of the resulting TRNG configurations was verified using the AIS 31 test suite.

I. INTRODUCTION

True random number generators (TRNGs) constitute an essential part of cryptographic systems. The statistical quality and unpredictability of the random output, but also robustness of the generator and ability to detect attacks guarantee the security. Our goal was to design a TRNG, which can be easily implemented in field programmable gate arrays (FPGAs) and fulfill above mentioned security requirements.

We selected the phase locked loop (PLL) based TRNG (PLL-TRNG) because of its simple and comprehensive design and because of the fact that PLLs in FPGAs are physically isolated from the rest of the device, which contributes to data security. In this context, our primary goal was to select the best parameters of the generator, which will ensure the highest entropy rate at its output, and to secure the generator using an AIS 31 compliant approach based on the use of the stochastic model.

The PLL-TRNG was first proposed by Fischer and Drutarovsky in [1] and then enhanced by Fischer et al. in [2]. A simple stochastic model of this generator was first proposed by Simka et al. in [3]. More rigorous and more precise model was proposed later by Bernard et al. in [4].

In the recent survey published in [5], the PLL-TRNG achieved insufficient entropy rate as well as quite low output bit rate. We aim to optimize the design of the generator using a genetic algorithm (GA) in order to increase both of the above mentioned parameters as well as to reduce the design time. We then test our optimized implementation on Altera Cyclone V, Xilinx Spartan-6, and Microsemi SmartFusion®2 FPGAs.

The paper is organized as follows. In Section II, we describe basic architecture of the PLL-TRNG, its characteristics and difficulties related to its design. In Section III, we show the application of the GA on optimization of parameters of this kind of generators. In Section IV, we present the results obtained using selected GA and we discuss the obtained results in Section V. Finally, Section VI concludes the paper.

II. THEORETICAL BACKGROUND OF THE PLL-TRNG

The core of the PLL-TRNG is depicted in Fig. 1. The source of randomness exploited by the TRNG is the jitter of the clock signal entering the PLL (clk_ref) and the tracking jitter introduced by the PLL itself. The clock signal generated by the PLL is sampled by a D flip-flop (DFF) and then K_D subsequent samples are XOR-ed together in the decimator to make one random bit (K_M and K_D are multiplication and division factors of the PLL, respectively).

Fig. 1. General architecture of a PLL-TRNG

If the value of K_D is odd, the output bit rate of the generator is defined as [1]:

$$R = \frac{f_{ref}}{K_D}$$

(1)

and the sensitivity to the jitter is equal to

$$S = \frac{K_D}{f_{jitter}}.$$  

(2)
TABLE I

PLL SPECIFICATIONS FOR ALTERA CYCLONE V, XILINX SPARTAN-6, AND MICROSEMI SMARTFUSION® 2 FPGA FAMILIES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Cyclone V</th>
<th>Spartan-6</th>
<th>SmartFusion® 2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{in}$ [MHz]</td>
<td>Min 5</td>
<td>Max 500</td>
<td>Min 19</td>
<td>Max 540</td>
</tr>
<tr>
<td>$M$</td>
<td>1</td>
<td>512</td>
<td>1</td>
<td>552</td>
</tr>
<tr>
<td>$D$</td>
<td>1</td>
<td>512</td>
<td>1</td>
<td>21</td>
</tr>
<tr>
<td>Post-VCO div.</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$C$</td>
<td>1</td>
<td>512</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>$f_{pfdin}$ [MHz]</td>
<td>5</td>
<td>325</td>
<td>19</td>
<td>500</td>
</tr>
<tr>
<td>$f_{VCO}$ [MHz]</td>
<td>600</td>
<td>1300</td>
<td>400</td>
<td>1000</td>
</tr>
<tr>
<td>$f_{out}$ [MHz]</td>
<td>0</td>
<td>460</td>
<td>3.125</td>
<td>400</td>
</tr>
</tbody>
</table>

By definition, the period relationship between the reference ($clk_{ref}$) clock and the jittery clock ($clk_{jit}$) is inversely proportional to the frequency relationship. Therefore, from Eq. (2) and from the PLL definition, it follows that

$$S = f_{jit} \cdot K_D = f_{ref} \cdot K_M.$$

From Equations (1) and (3) it follows that to simultaneously increase sensitivity to the jitter, and thus the entropy rate, and the bit rate at the output, it is necessary to

- increase the reference frequency,
- decrease the division factor,
- increase the multiplication factor.

Unfortunately, these changes increase the internal frequencies of the PLL to the point where they might go beyond the limit of the PLL component itself. Hence, we must take into account the physical parameters of the PLL when designing the given TRNG.

As explained in [2], if the PLL-TRNG parameters are outside of intervals feasible in the given technology (i.e. given ranges of PLL parameters), two PLLs can be used. However, to reduce the total cost of the generator, our objective was to find PLL-TRNG configurations using just one PLL.

A. Setup of the PLL internal structure

Figure 2 depicts simplified general structure of PLLs available in selected FPGA families. Indeed, the PLL structure is very similar in the three FPGA device families. Concerning the architecture (but not parameters of its individual blocks), the main difference is in the presence and configuration of the post-VCO divider (VCO means the voltage controlled oscillator): while this divider is not available in the Spartan-6 family, the PLLs in the Cyclone V family can divide the VCO output frequency by one or two, and those in the SmartFusion®2 devices can divide it by 1, 2, 4, 8, 16, or 32. The number of outputs of the PLL block ($i + 1$) is also different in these devices, i.e. 4, 6, and 10 in SmartFusion®2, Spartan-6, and Cyclone V FPGA, respectively.

The physical parameters of every internal component depicted in Fig. 2 have to be taken into account in the PLL-TRNG design. Table I contains the list of the parameters and their limit values documented by Altera [6], Xilinx [7], and Microsemi [8].

The internal frequencies of the PLL are directly dependent on the chosen counter values ($M$, $D$, and $C_i$, respectively) as well as on the input frequency of the PLL. Equations (4), (5), and (6) show the relationship between the internal frequencies and other PLL parameters:

$$f_{pfdin} = \frac{f_{in}}{D}, \quad f_{VCO} = \frac{f_{in} \cdot M \cdot PVCOd}{D}, \quad f_{out} = \frac{f_{in} \cdot M}{D \cdot C_i}.$$
In our experiments, we used an open-source GA optimization utility provided by [10]. This utility is suitable for the GA optimization with up to 5 design variables and 5 constraints, which was sufficient in our case.

The design variables represent the parameters, which should be found (by the GA). In case of the PLL-TRNG, it is the

- input frequency \( f_{\text{in}} \),
- multiplication factor \( M \),
- division factor \( D \),
- post-VCO divider \( PVCOd \),
- output clock divider \( C \).

The design parameters, which are not controlled directly, but which depend on design variables and need to have values within certain limits are called the constraints. In our case, the constraints were as follows:

- phase detector input frequency,
- VCO frequency,
- output frequency.

An important part of the GA optimization is the fitness function, because it determines which of the generated individuals are the best to propagate to next generations. In our case, the fitness function is a TRNG parameter, which we want to optimize for. And since security is the most critical property of a TRNG, we decided to optimize for the highest possible entropy rate, thus maximize the sensitivity to the jitter \( S \).

### IV. Implementation Results

In our implementations, we used the PLL-TRNG architecture depicted in Figure 3.

We used a ring oscillator as a reference clock source for the PLL-TRNG in order to have a configurable source of the input clock frequency.

First, we created three different GA optimization models based on Table I, one for each FPGA family. Next, we used the GA to find optimal PLL-TRNG parameters and we verified that the found parameters are indeed implementable in the corresponding family. Finally, we selected three best candidates for each of three families.

Table II presents the parameters of our optimized implementations of the PLL-TRNG ordered by the achieved bit rate. Although all the generated solutions were implementable in given families, we selected the best candidate in each family, knowing that according to our experience, to obtain high entropy rate, the sensitivity to the jitter \( S \) should be at least 0.04.

We selected Configuration 2 for the Altera Cyclone V family, because it had sufficient jitter sensitivity, much bigger than Configuration 1. We eliminated Configuration 3 in this family, because the division factor \( K_D \) had an even value.

We selected Configuration 1 for the Xilinx Spartan-6 FPGA, because it had the highest bit rate, while the sensitivity to the jitter was very close to that of Configuration 2 and 3 (although smaller than required, i.e. smaller than 0.04).

Finally, we chose Configuration 2 for SmartFusion® 2, because it was faster than Configuration 3 (while having sufficient sensitivity) and we eliminated Configuration 1 (although it was faster), because its sensitivity was not sufficient.

### V. Discussion

The use of the GA is very useful in optimization of parameters of the PLL-TRNG. This is true for two reasons:

- In order to manage the obtained result, the multiplication and division factors \( K_M, K_D \) of the PLL-TRNG (and thus multiplication and division factors \( D, M, C \) of the PLL) must be setup very precisely.
- Because of the fact that the technical documentation of PLL blocks, which is publicly available, is not sufficiently detailed and the user is forced to use the PLL intellectual
property (IP) function generators, which indicate only success or failure of the PLL setup procedure and often not the direct reason of the failure.

As an example, we can mention the Microsemi FAB CCC Configurator (the Microsemi PLL IP function generator), in which the user has to enter the input and output frequencies of the PLL with the precision from two to three decimal places, in order to get required multiplication and division coefficients. This is clearly not possible using the ‘failure and success’ strategy. On the contrary, the application of the genetic algorithm together with the associated Excel file specified the needed coefficients and input/output frequencies and simplified largely our task.

Table II specifies sensitivities to the jitter of different PLL-TRNG configurations, but the final entropy rate depends also on the clock jitter, which can vary from family to family. Therefore, we verified the quality of the generated raw random numbers using Procedure B of the AIS 31 [11] (tests T6 to T8) and we estimated the entropy rate at the generator output using the results of test T8, as it was made in [5].

Table III compares results of our implementations with those presented in [5], which used the same sources of randomness and the same FPGA families.

### Table III

<table>
<thead>
<tr>
<th>PLLs used</th>
<th>Bit rate [Mbits/s]</th>
<th>AIS 31 (T6 - T8) Entropy estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altera Cyclone V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td>2.604</td>
<td>PASS</td>
</tr>
<tr>
<td>[5]</td>
<td>2</td>
<td>FAIL</td>
</tr>
<tr>
<td></td>
<td>0.986</td>
<td></td>
</tr>
<tr>
<td>Xilinx Spartan-6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td>3.975</td>
<td>FAIL</td>
</tr>
<tr>
<td>[5]</td>
<td>2</td>
<td>FAIL</td>
</tr>
<tr>
<td></td>
<td>0.981</td>
<td></td>
</tr>
<tr>
<td>Microsemi SmartFusion® 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This work</td>
<td>1.2</td>
<td>PASS</td>
</tr>
<tr>
<td>[5]</td>
<td>0.37</td>
<td>FAIL</td>
</tr>
<tr>
<td></td>
<td>0.921</td>
<td></td>
</tr>
</tbody>
</table>

We can observe that the output bit streams of generators implemented in the Cyclone V and SmartFusion2 families passed the AIS 31 tests without any problems. It was not the case of the Spartan-6 family, because of the limitations of the PLL block. The small range of the PLL division factor made it extremely difficult to implement a TRNG using only one PLL. Since the TRNG bit rate is high enough, a possible solution would be to use an XOR based post-processing. This would reduce the bit rate in favor of increasing the entropy.

The entropy estimation gave us a better knowledge about the proximity to the required statistical quality. Indeed, according to AIS 31, the Shannon entropy rate at the generator output should be higher than 0.997. We can see that even in the case of the Spartan-6 family, our solution is much closer to the required case than that published in [5].

However, differences in bit rates are even more remarkable: comparing to [5] we increased the bit rate 3, 4, and 8 times in SmartFusion® 2, Cyclone V, and Spartan-6, respectively. Last but not least, using the GA optimization, we were able to obtain satisfying results using only one PLL. A suitable solution with only one PLL was extremely hard to find manually.

### VI. Conclusion

In this paper, we proposed a method of optimization of the PLL-TRNG design using a genetic algorithm. The optimized configurations reached considerably higher bit rate and their area was reduced by two by using only one PLL.

Besides increasing the bit rate and reducing the cost, the genetic algorithm was useful in increasing the entropy rate in all selected families. In two of them, Altera Cyclone V and Microsemi SmartFusion® 2, the entropy rate was increased to the value, which is required by the AIS-31 standard and in the third one, Xilinx Spartan-6, the estimated entropy rate was much closer to the required value and therefore easy to increase by some low cost post-processing method.

In the near future, we plan to extend the number of input parameters and of design constrains, to manage the design of a generator using two PLLs. This could be particularly useful in the case, when a low jitter clock source must be used.

### Acknowledgment

This work has received funding from the European Union’s Horizon 2020 research and innovation programme in the framework of the project HECTOR (Hardware Enabled Crypto and Randomness) under grant agreement No 644052.

### References


