SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 13, No. 1, February 2016, 59-70

UDC: 004.421.6:519.7 **DOI: 10.2298/SJEE1601059P**

Realization of Universal Periodic Sequence Generator on FPGA

Miljan Petrović[1], Milica Jovanović[1]

Abstract: This paper presents mathematical modeling and performance evaluation of different realizations of universal periodic digital signal generator based on infinite impulse response filter. The development kit used was Spartan 3E-Starter Board. Using Xilinx software environment and VHDL, the generator has been described and then synthesized and implemented on FPGA chip on the board. Included realizations are direct form II (canonical form) of the filter, as well as hardware optimized single register structure with different control mechanism. Comparative analysis of these two digital systems points to their differences, advantages and weaknesses.

Keywords: FPGA, IIR filter, Signal generator, VHDL.

1 Introduction

Discrete signal generators are certainly of great significance in many electronic devices, e.g. their usage is crucial in testing digital systems. Therefore, much attention is being given to design techniques, implementation methods, and verification of generators. As a special case, periodic signal generator emerges with its wide range of application. In this paper, mathematical modeling of discrete periodic signal generator [1] has been discussed and different realizations of the generator are presented and evaluated.

The paper is structured as follows. First, mathematical background of generator model is presented. It is based on IIR filter design. Second, the generator is described using VHDL, and implemented on FPGA development board. The structure is then optimized (minimum number of used hardware components) and once more implemented. There is further elaboration on the performance of two realizations of the generator as well as the comparison of the provided essential characteristics. Finally, guidelines for further research are emphasized in the concluding section.

[1]Faculty of Electronic Engineering, University of Nis, Aleksandra Medvedeva 14, 18000 Nis;
E-mails: miljan.petrovic@elfak.ni.ac.rs; milica.jovanovic@elfak.ni.ac.rs

2 IIR filter as Signal Generator

The basic concept applied in modeling the generator is acquiring filter structure with impulse response corresponding to desired generated output sequence. Since the sequence is periodic and consists of infinitely many samples, the logical choice is to use IIR (infinite impulse response) filter. Its transfer function *H*(*z*) can be written as:

$$\begin{aligned} H(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_k z^{-k} + \\ + a_0 z^{-k-1} + a_1 z^{-k-2} + a_2 z^{-k-3} + \cdots + a_k z^{-2k-1} + \\ + a_0 z^{-2k-2} + a_1 z^{-2k-3} + \cdots \end{aligned} \tag{1}$$

where $a_i, i = 0,...,k$ represent values of the response in each period consisting of $k+1$ samples, and *z* denote the complex frequency. Performing several operations with mathematical expression in (1) leads to:

$$H(z) = \frac{1}{1-z^{-1}} \cdot \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_k z^{-k}}{1 + z^{-1} + z^{-2} + \cdots + z^{-k}}, \tag{2}$$

where notation is the same as in (1).

It can be observed that the first fraction in (2) is Heaviside function transformed into *z* domain. Thus, it is inferred that the output sequence can be generated both by providing a Dirac impulse as input signal of the filter *H*(*z*), and providing Heaviside signal as input signal of the filter with transfer function *H'*(*z*) equal to the second fraction in (2). The second method is chosen because of the way of canceling the generation process. Namely, *H*(*z*) would require negative Dirac impulse, and in the case of *H'* (*z*) it would be enough just to drop the input sequence to zero, thus, avoiding negative values. Finally, designed generator is an IIR filter with transfer function *H'*(*z*), where coefficients in polynomial in the numerator determine values of output sequence in each period.

$$H'(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_k z^{-k}}{1 + z^{-1} + z^{-2} + \cdots + z^{-k}}. \tag{3}$$

It is an interesting fact that all the coefficients in denominator of the transfer function of generator are equal to 1. Lagrange bounding the roots of polynomials tells that in this case maximum amplitude of a root equals 1 [2]. In addition, it can be derived that all the roots are located on the unit circle in a complex plane. This is exemplified on a polynomial of seventh order, with roots presented in the complex plane in Fig. 1. Thus, the stability condition is not fulfilled. Yet, the purpose of the filter is to generate periodic sequences, i.e. to behave as an oscillator structure, which is in compliance with the root locus.

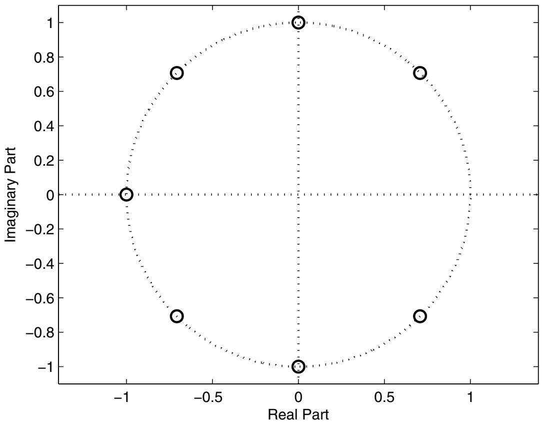


Fig. 1 – *Roots of polynomial of seventh order with all coefficients equal to ones.*

3 Realization of Direct Form II

The block diagram of a widely accepted filter realization also known as direct form II is displayed in Fig. 2. Marks $x[n]$ and $y[n]$ denote input and output signals, respectively, whereas $w[n]$ represents an internal signal whose values in previous time instants are implemented as outputs of delay lines. Multiplier operands are marked with $a_i, i = 0,...,k$.

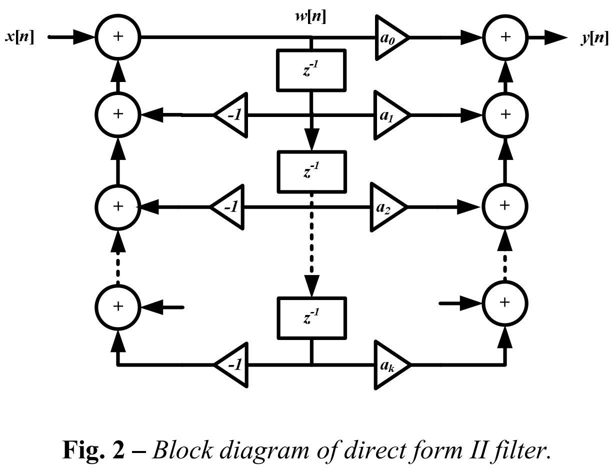


Fig. 2 – *Block diagram of direct form II filter.*

Word length of the system implemented for the purpose of this paper is 8 bits. However, the choice of their interpretation (as integers or other number formats) is not of considerable relevance. This is due to the fact that multiplication and addition operations generally affected by number formats are ostensibly implemented, and only special cases of these operations are actually performed. This shall be further elaborated in the section on the process of optimization to single register structure.

The filter implemented is of order 7. Hence, its realization should consist of 7 delay lines (registers), 15 multipliers and 14 adders. However, it shall be exhibited after synthesis that the list of used hardware components is somewhat different.

3.1 VHDL description

The generator is described in Xilinx software environment using VHDL [3 – 5]. Entity of VHDL code contains input and output ports of the system. Signal *clk* represents clock signal, *rst* signal stands for resetting the output sequence, and *input* and *output* are input (control) and output signals, respectively. There are also 8 more input signals – *koef0*, *koef1*,…, *koef7* – which determine multiplication operands of embedded multipliers in FPGA (which correspond to constants $a_i, i = 0,...,7$ in Fig. 2). They are set to constant value in the test bench, thus avoiding automatic hardware optimization performed in case of declaring these signals as constants.

VHDL description architecture encompasses a single process. Since the generator has a clock signal and an asynchronous reset signal (defined using nested *if* statements), the process sensitivity list contains *clk* and *rst*.

There are eight 8-bit variables of the type *unsigned* that correspond to signals at outputs of delay lines (*w1*,…, *w7*) and at input of the first one (*w0*). The delayed signal assignment in processes in VHDL explains the usage of variables instead of signals. Furthermore, if these signals had been defined as one vector of vectors variable (1D×1D type), the synthesis software would not have performed as required. Namely, it would have ignored certain parts of the variables considered unnecessary. That is why separate vector variables are used.

3.2 Synthesis and implementation

Spartan-3E Starter Board with Xilinx XC3S500E FPGA chip was used for synthesis and implementation of the generator. Some of the parameters in Xilinx 14.7 software environment were set according to the requirement to develop hardware architecture as defined in Fig. 2. Structures embedded in the board were used for the synthesis of multipliers by setting the option *Multiplier Style* to *Block*. In addition, input and output buffers were not included in synthesis (Option *Add I/O Buffers* not checked).

Table 1
List of components used in synthesis.

Type of component	Number of components used
8-bit multipliers	8
16-bit adders	7
8-bit subtractors	7
flip-flops	64

The list of components used in synthesis of the generator can be seen in **Table 1**. When data in **Table 1** and Fig. 1 are compared certain discrepancies emerge. First, pairs of multiplier with constant -1 and corresponding adder seen in Fig. 2 are replaced with 8-bit subtractors. Further, it can be observed that adders (on the right side in Fig. 2) are synthesized as 16-bit since their operands are outputs of multiplication. Flip-flops used are parts of eight 8-bit registers of which 56 (7 times 8) are represented as delay lines z^{-1}. The remaining 8 flip-flops constitute a register at the generator output. This register can not be avoided due to the fact that VHDL description includes summation of 16-bit values and assigning the sum (8 bits at lower bit positions) to the output signal. However, ignoring 8 bits at higher positions in the output signal does not result in information loss since the input is 8-bit signal and multiplication is performed on data from which one operand is either 1 or 0. This fact is again used in optimizing the generator to a single register structure. The number of LUTs (Look-up Tables) used in synthesis is 124.

Timing characteristics of the generator are calculated during synthesis and presented in **Table 2**. However, these are only estimation results. More accurate values are provided after implementation in Post−Place and Route Static Timing Report. There it can be read that actual values of minimum clock signal period and maximum frequency are 32.263 ns and 30.995MHz, respectively, which are slightly different from values in **Table 2**.

Table 2
Timing estimation after synthesis.

Timing constraint	Value
Minimum period (Maximum frequency)	30.107 ns (33.215 MHz)
Minimum input arrival time before clock	29.081 ns
Maximum output required time after clock	0.591 ns

3.3 Behavioral and post-route simulation

Behavioral simulation was performed on direct form II realization of the generator. The goal is to get the system to generate the sawtooth signal shown in Fig. 3. Consecutive values of the signal period are 1, 9, 17, 24, 33, 41, 49 and 57 when transformed into 8-bit unsigned integer format. Multiplier operand signals *koef0*, *koef1*,…, *koef7* are set to corresponding values at the beginning of the test bench, before initial resetting.

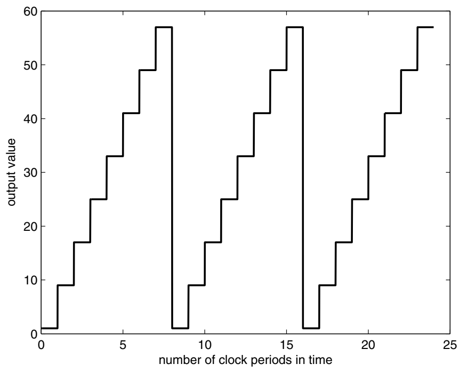


Fig. 3 – *Sawtooth waveform expected at generator output.*

Results of behavioral simulation are presented in Fig. 4. They show regular values generation at output until *rst* signal is activated. While *rst* equals 1, *output* is equal to 0. Once *rst* drops to 0, the generation starts again.

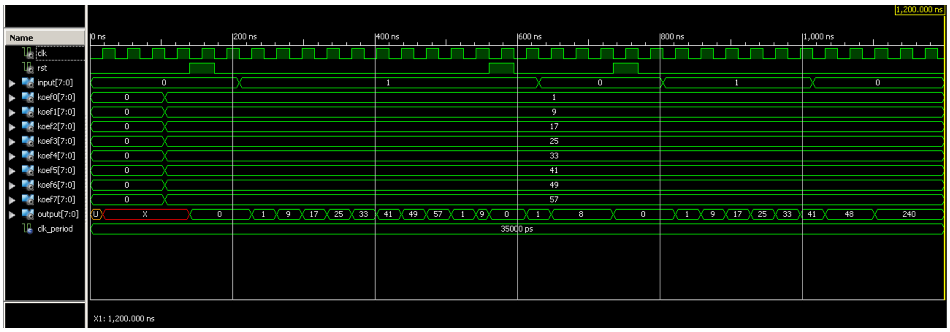
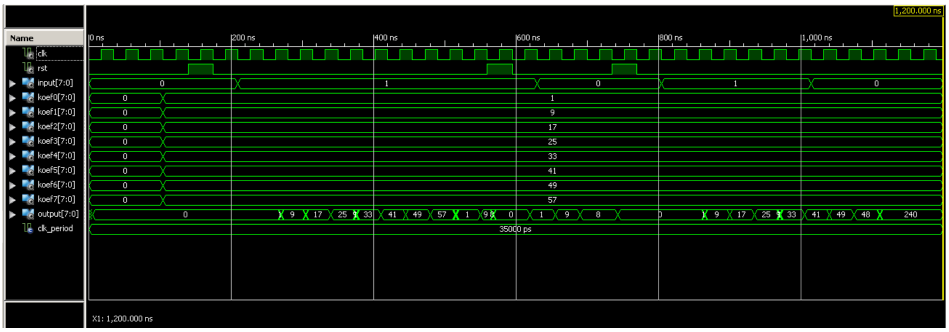


Fig. 4 – *Waveforms in behavioral simulation of direct form II generator.*

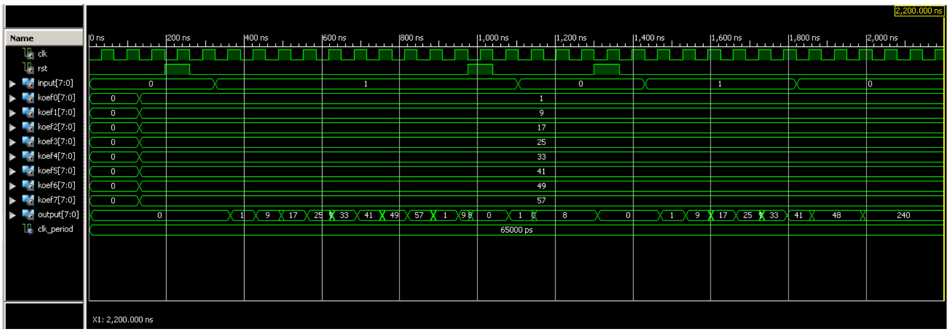
However, a problem can be observed, previously described in [1]. Setting *input* to 0 does not always stop the output generation. This happens only if the change is made after an integer number of output signal periods have finished generating (in the case of the written test bench, when *output* equals 57). Due to this fact, the generator can be stopped in two ways. One way of stopping the generator is by deactivating *input* when *rst* was previously set to 1. The other is by resetting after setting *input* to 0. In the latter case irregular output signal values shall emerge (values 8, 48 and 240 in Fig. 4).

Post-route simulation was performed after applying timing constraints for specific implementation of the systems. Results are shown in Fig. 5, where (a) and (b) parts display simulation waveforms for values of clock period of 35ns and 65ns, respectively. Both values are acceptable considering timing estimation in **Table 2**.

Exploring the waveform in Fig. 5a in detail displays certain glitches, i.e. irregular values of *output*. These glitches are inevitable in practical system realizations but their duration is negligible when compared to clock period and regular values duration. For example, between values 41 and 49, output transiently sets to 57 and lasts for 0.52ns. However, the duration of glitches does not exceed 1.66 ns (e.g. consecutive values 9 and 1 set between regular values 25 and 33).



(a)



(b)

Fig. 5 – (a) *Waveforms in post-route simulation of direct form II generator with clock period of* 35ns; (b) *Waveforms in post-route simulation of direct form II generator with clock period of* 65ns.

In addition, an interesting observation can be made on the order of *output* values. Namely, the first value (which should be equal to 1) does not appear at the beginning of the first generated period, although it is present in latter sequences. This phenomenon can be explained by the input arrival time before clock equal to 17.5ns, which is considerably less than the constraint in **Table 2**. Significantly lower value is a consequence of the fact that changes of *input* and *rst* are synchronized with inactive edge of the clock in test bench. If this was not considered as a problem, a clock with period of 35ns would be acceptable for implementation. Otherwise, minimum clock period shall be 65ns (with maximum frequency 15.3 MHz). Waveforms for this case show the same glitch durations (but shorter relative duration due to longer clock period), however, this certainly fulfills all the constraints. Another solution which does not change the maximum clock frequency includes substituting the first and last multiplier structure of the generator.

4 Realization of Single Register Structure

4.1 Hardware optimization of direct form II generator

In order to perform hardware optimization of direct form II, it is crucial to be acquainted with values of signals at certain points in the system. Signals at inputs of the first adder and at inputs of first three multipliers are provided in Fig. 6. For each signal, values at the first three time instants can be seen.

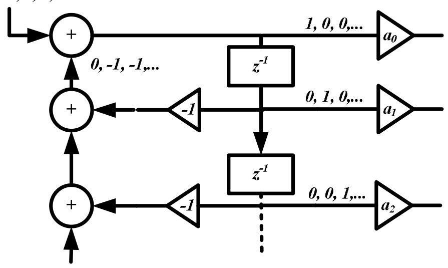


Fig. 6 – *Block diagram of part of direct form II filter with values of signals in certain points at first 3 time instants.*

The input signal is equal to 1 for as long as the generation of output is needed. In the first time instant, under the assumption the system was reset to zero-state, a_0 is multiplied with 1 and the result appears at the output. Next, the output signal of the first delay line equals to 1, and the signal at its input is now 0 because of the addition of 1 and –1 in the first adder. The addend –1 changes to 0 only at first time sample of each period. It can be concluded that value 1 follows a circular path through the delay line sequence, i.e. inputs of multipliers encompass one value 1 and all the other values 0 at each time instant.

Furthermore, outputs of multipliers encompass one value a_i and all the other values 0 at each time instant. Hence, the interpretation of number formats is not significant in this system.

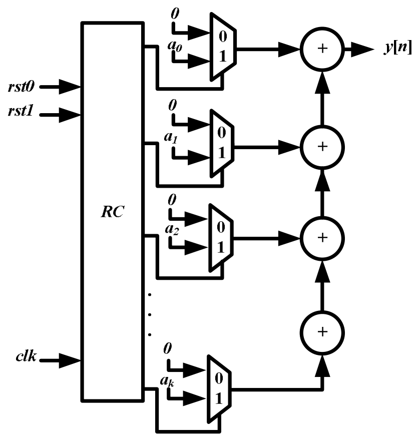


Fig. 7 – *Block diagram of single register structure filter.*

Finally, a block diagram of optimized structure is derived and presented in Fig. 7. All delay lines, multipliers with –1, and corresponding adders from direct form II are replaced with one register – straight ring counter *RC*, whereas multiplexers stand in lieu of multipliers. The number of register output pins is equal to the number of samples in one period of desired output sequence. Input signal $x[n]$ is removed, and the control is implemented through signals *rst1* and *rst0*. Signal *rst1* sets register output to “00…01”, and thus, the system output resets to a_0 and generation continues. Signal *rst0* sets all register outputs to zero, making the output sequence $y[n]$ equal to 0, which means the generator is stopped.

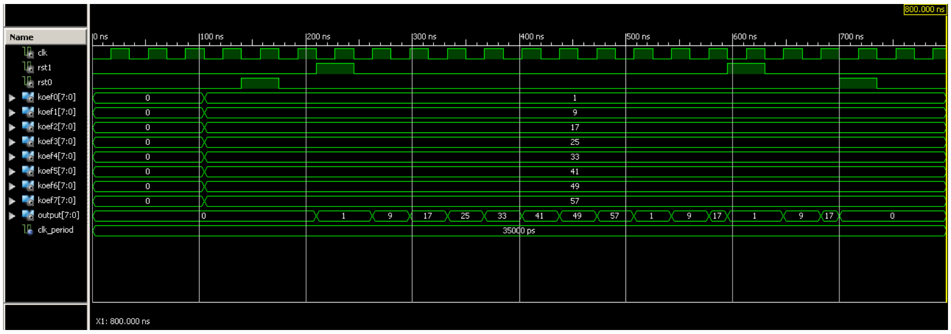
4.2 VHDL description

The entity of VHDL description of the single register generator encompasses one 8-bit output signal (marked $y[n]$ in Fig. 7) and input signals *koef0*, *koef1*,…, *koef7* representing input signals of multiplexers (which correspond to constants a_i in Fig. 7), *clk* clock signal, *rst0* signal for asynchronous reset (setting the output to zero value) and *rst1* asynchronous signal for restarting the output sequence (corresponds to *rst* signal in direct form II realization). Signal *rst0* is of higher priority than *rst1*.

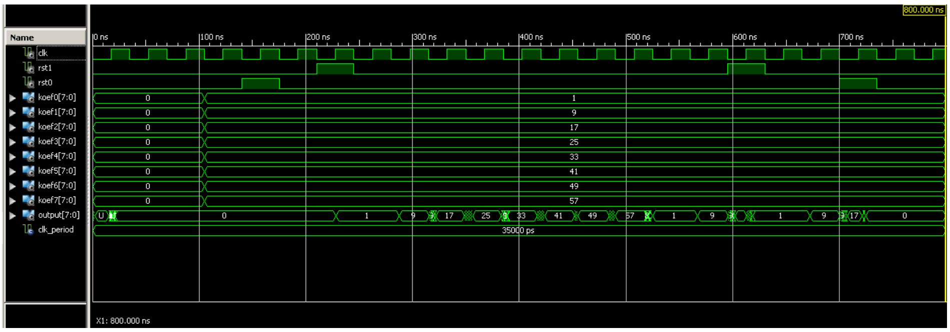
The architecture of the description includes one process and several concurrent statements. The process describes the register *RC* in which bit ‘1’ moves through register outputs every time the active edge of the clock signal appears. Controlled assignments of ‘0’ or a_i (*koefi*) to corresponding addition operands are described using concurrent *when* statements.

4.3 Synthesis and implementation

Synthesis and implementation of single register structure are performed in the same environment as the realization of direct form II. Automatic replacement of adders with combination of multiplexers and adders is avoided by deactivating the option *Resource Sharing* in Xilinx. For a generator with fixed values output sequence it is possible to minimize Boolean functions of multiplexers. However, the aim of this paper is to examine a generator with variable output sequence. Hence, multiplexers shall be implemented as blocks embedded in FPGA board. To achieve this, *Mux Extraction* option was set to *Force*, which yields in inferring multiplexers from *when* statements in VHDL code. Input and output buffers were not included in the synthesis.



(a)



(b)

Fig. 8 – (a) *Waveforms in behavioral simulation of single register structure*; (b) *Waveforms in post-route simulation of single register structure with clock period of* 35ns.

The realization of the generator includes seven 8-bit adders, eight multiplexers and one 8-bit register. The number of used LUTs is 98, which is considerably less than direct form II hardware requisites.

Although Post Place and Route Static Timing Report states that the minimum clock period is 2.065ns, certain logic gates exhibit delay times ranging up to 30ns. This results in a requirement that the working frequency is lower than 28.5MHz. Hence, post-route simulation was performed for the case of clock period being equal to 35ns.

4.4 Behavioral and post-route simulation

Waveforms representing results of behavioral and post-route simulation of single register generator are shown in Fig. 8a and 8b, respectively. Fixed multiplexers inputs *koef1*,…, *koef7* are set to values according to Fig. 3 at the beginning of test bench. Behavioral simulation displays generator control with signals *rst0* and *rst1*. Post-route simulation was performed for the case of clock period being equal to 35ns, according to timing constraints derived in implementation process.

Table 3
Durations of glitches in first period of output sequence.

Regular value preceding glitch	Regular value following glitch	Duration [ns]
1	9	0
9	17	6.337
17	25	8.083
25	33	4.869
33	41	6.62
41	49	3.073
49	57	5.801
57	1	5.965

Glitches (irregular output signal values) are observed in waveforms in Fig. 8b. Their durations are provided in **Table 3** for each pair of consecutive regular values in the first period of generated output sequence. It is evident they are much greater than durations of glitches in the case of direct form II, both in absolute value and relatively to the clock period. However, glitches are negligible if maximum working frequency (or hardware requisites) is the criterion with higher priority. In the case of single register structure it is equal to 28.5MHz. Further, there is no omitting the first value in the output sequence.

5 Conclusion

Modeling universal periodic sequence generator as infinite impulse response filter has been presented. Furthermore, performance evaluation is provided for two implemented hardware structures, direct form II and single register structure. It has been concluded that significant characteristics such as maximum working frequency and glitches durations suggest an optimal choice between these realizations. However, more efficient hardware implementations can be explored for different applications [6].

The following steps needed for the completion of elaboration on this topic are verification and evaluation of the generator when used as a subsystem. A potential device encompassing this kind of generator is an inverter in power management electronics. Direct form II realization should be considered for the purpose of processing PWM input signal. In addition, the generator can be examined when used for test sequence generation [7].

6 References

[1] M. Petrović: Design of IIR Filter based Periodic Sequence Generator, 58th ETRAN Conference, Vrnjačka Banja, Serbia, 02-15 June 2014, pp. EL2.2. 1 – 5. (In Serbian).

[2] H.P. Hirst, W.T. Macey: Bounding the Roots of Polynomials, The College Mathematics Journal, Vol. 28, No. 4, Sep. 1997, pp. 292 – 295.

[3] C. Maxfield: The Design Warrior's Guide to FPGAs, Newnes, Burlington, MA, USA, 2004.

[4] A. Moore: FPGAs for Dummies, John Wiley and Sons, Hoboken, NJ, USA, 2014.

[5] S. Brown, Z. Vranešić: Fundamentals of Digital Logic with VHDL Design, McGraw-Hill, New York, NY, USA, 2009.

[6] M. Hasnain, S. Hammad, M.A. Iqbal: Efficient Hardware Implementation of Digital Filters using Distributed Arithmetic (DA), Journal of Basic and Applied Scientific Research, Vol. 3, No. 7, July 2013, pp. 562 – 573.

[7] N.K. Jha, S. Gupta: Testing of Digital Systems, Cambridge University Press, Cambridge, UK, 2003.