VHDL Generation of Optimized IIR Filters

Anatoliy Sergiyenko
Computer Engineering Department
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
aser@comsys.kpi.ua

Anastasia Serhienko
Computer Engineering Department
Igor Sikorsky Kyiv Polytechnic Institute
Kyiv, Ukraine
a.serhienko@comsys.kpi.ua

Abstract — In this paper a method is proposed, which consists in integer coefficient searching, forming the filter structure and modeling it. The use of the VHDL language in all steps of the filter design helps to speed-up the design process and to improve the filter optimization. Examples of the multiplier-less IIR filter design show the method effectiveness.

Keywords — VHDL, FPGA, IIR filter, allpass filter

I. INTRODUCTION

A traditional approach of the digital filter design for the field programmable gate array (FPGA) consists in performing the next steps. A set of filter coefficients is searched, which satisfy the filter specification using the proper CAD tool like Matlab. Then, the coefficients are quantized accompanying by the filter frequency response proving. The rounded coefficients are built in the filter IP core, which is provided by the FPGA supplier or in the filter model, described by VHDL or Verilog languages. Finally, the filter model is tested using the proper testbench before and after the netlist synthesis [1,2].

The special and not free program like Matlab is of demand for this approach implementation. Another disadvantage is checking the frequency response after each coefficient quantization. Besides, both the filter structure and the result rounding scheme infer this frequency response, which has to be proven. Most of the issues of the filter design for FPGA using Matlab are described in [3]. But the only method of the filter testing which is proposed in this book is the usual testbench modeling using the proper signal generation like the step signal generator.

In this paper, the VHDL language is used both for the filter structure description and coefficient searching, as well as the frequency response calculating. This approach is proven by the multiplierless infinite impulse response (IIR) filter synthesis.

II. VHDL PROPERTIES FOR THE DIGITAL FILTER DESIGN

The VHDL language is usually used to describe the application specific digital structure for its configuring into FPGA. But its ability of the mathematical data processing is undervalued. The IEEE library contains the MATH_REAL and MATH_COMPLEX packages. They consist of floating point types, constants, and functions, which are suitable for the complex number processing. These packages are usually used for the VHDL project testbench design. But they have many undervalued features to explore the algorithms of the digital signal processing. Also, the effective routines for the linear equation system solving and discrete Fourier transform are based on these packages. Thus, the possibilities of VHDL language for the mathematical processing and simulation are approaching to ones of the Matlab tool [2,4].

The complex variable $Z = re^{j\varphi}$ is usually used for the digital filter analyzing and synthesis. To get this variable in VHDL, the following function of the vector magnitude r and its angle φ can be used:

```
function Z(r,fi: real) return COMPLEX_POLAR is begin
  return r*exp(COMPLEX_TO_POLAR(MATH_CBASE_J)*fi);
end Z;
```

Getting this function, the complex filter transfer function can be described. For example, the low-pass filter transfer function based on the allpass filter:

$$H(z) = z^{-1} + \frac{a + a(1+b)z^{-1} + z^{-2}}{1 + a(1+b)z^{-1} + az^{-2}}; \qquad (1)$$

is described by the function:

```
function LPF(a, b, r, fi: real) return COMPLEX_POLAR is
begin
   return Z(r,-fi) +
         (COMPLEX_TO_POLAR(COMPLEX'(a,0.0)) +
           a*(1+b)*Z(r,-1.0*fi) + Z(r,-2.0*fi))/
         (COMPLEX_TO_POLAR(COMPLEX'(1.0,0.0)) +
           a*(1+b)*Z(r,-1.0*fi) + a*Z(r,-2.0*fi)) ;
end LPF;
```

The coefficients a, and b of the transfer function can be calculated using the equations [5]:

$$a = \frac{1 - \operatorname{tg} d_f}{1 + \operatorname{tg} d_f}; \; b = -\cos f_C \cdot (1 + a), \qquad (2)$$

where d_f is the transition band, f_C is the passband. It is possible to calculate the transfer function diagram in the frequency space in the VHDL simulator using the following process operator:

```
process(CLK)
    variable p, phas: real :=0.0;
    variable Hz: COMPLEX_POLAR;
begin
     a <= -0.5;  b <= 0.64;              -- filter coefficients
     if CLK='1' and CLK'event then
          phas := phas + 0.001; -- phase (frequency)counter
          p := phas * MATH_PI * 2.0;  -- normalized phase
          m<= trunc(phas)*0.1+0.1; -- vector Z magnitude
          ph <= phas;                 -- frequency signal
     end if;
     Hz := LPF(a, b, m, p);                -- H(z)
     mag <= abs(Hz);                -- magnitude of H(z)
     Phase <= Hz.ARG;               -- phase of  H(z)
```

```
    Logm <= 20.0*log10(abs(Hz));    -- H(z) in decibels
end process;
```

This process generates the waveforms of the magnitude, logarithm magnitude, and phase frequency responses on the screen of the VHDL simulator for a thousand clock cycles. It helps to investigate the pole-zero chart as well. Consider the transfer function of the all pass filter

$$H(z) = \frac{b + a(b+1)z^{-1} + z^{-2}}{1 + a(b+1)z^{-1} + bz^{-2}}. \qquad (3)$$

The values of a and b are derived in (2). The respective pole-zero chart is shown in Fig. 1. The experimental finding of the pole and zero positions consists in sequential exchanging the length r of the vector Z, rotating the angle φ, and calculating the filter response for these values.

Consider b = 0.64, a = –0.5. The magnitude response of the function (3) derived by this method using the Active HDL simulator is shown in Fig.2.

The analysis of waveforms in Fig. 2 shows that the poles of the function $H_i(z)$ have a phase ± 2π·ph = ± 2π·0.1645 = = ±1.0336 and are at the points $r_1 = 0.8\angle 1.0336$, $r_2 = 0.8\angle -1.0336$. The zeros of the $H_i(z)$ function have the same phase and are equal to $q_1 = 1.3\angle 1.0336$, $q_2 = 1.3\angle -1.0336$. It should be noted that –cos(1.0336) = = –0.512 ≈ a; 1/0.8 = 1.25 ≈ 1.3 and $0.8^2 = 0.64 = b$, that is, the relation (2) is valid. Besides, one can see that by the magnitude m =1.0 the resulting magnitude $|H_i(z)| = 1.0$ for any frequency, which is true for the all-pass filter.

During the digital filter synthesis, the filter coefficients in the floating point representation are derived using Matlab, Scilab, or any other tool. Then these coefficients have to be quantized. The above VHDL process provides the filter characteristics estimation for any bit width of the quantized coefficients. Moreover, these coefficients can be optimized selecting their least significant bits.

At this process, for example, for the function (1), the quantized coefficient values a, b are adjusted by iterating their values in some convergence circles of a and b. Then, the optimum rounded filter coefficients are selected, which optimize the difference of the calculated function (1) and the frequency response in the filter specification.

The derived filter coefficients are put in the filter model described by VHDL. This model can be tested before and after the synthesis using the testbench, which is available at [6]. By this testing, the inphase REO, and quadrature IMO components of the analytical signal are fed to the inputs of two instances of the same filter (see Fig.3).

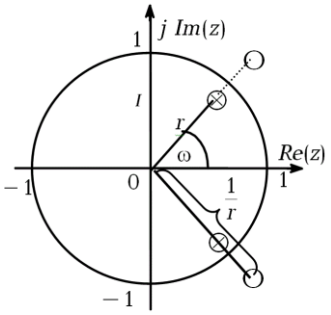

Fig. 1. Pole-zero chart of the all-pass filter

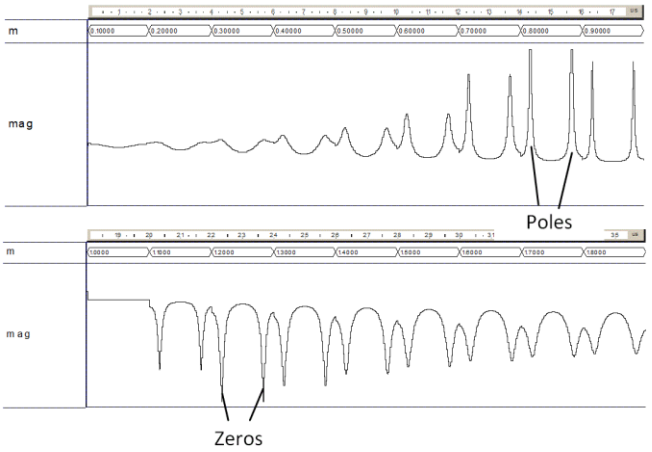

Fig. 2. Testbench for the computation of the all-pass filter response

The response signals RERSP, IMRSP of the filter instances are sampled in the FilterTB_r component after the given delay T_G, which is higher than the estimated group delay of the filter. Then, the value of the frequency response is calculated in the FilterTB_r component as the magnitude and phase of the complex vector.

The frequency of the analytical signal (REO, j*IMO) is exchanged linearly with the period T_G. The results of the modeling are the magnitude and phase responses, which are outputted in the waveform window of the VHDL simulator. This testbench is a very effective one because it not only proves the filter model correctness but generates the frequency response charts, which take into account all the data truncations, roundings, overflows, and saturations.

III. MULTIPLIER-LESS IIR FILTERS

IIR filters provide less complexity and higher filtering effectiveness comparing to the finite impulse response (FIR) filters. But they have less use in the FPGA systems because of the increased data bit width and the limited throughput. The speed of IIR filters is limited by the critical path length, which is usually estimated by the delay in the filter feedback. This delay could not be minimized by the pipelining technique, which is traditionally used for the FIR filter structures [1,3].

One of the effective methods to speed-up the IIR filter in FPGA is to simplify the multiplication by the substituting the hardware multiplier to a set of adders, which add the shifted multiplicand [1,3,7].

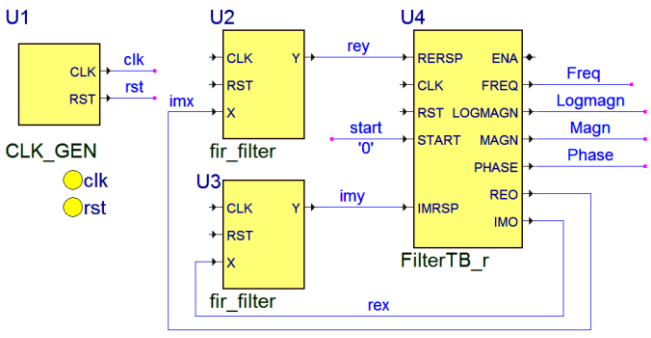

Fig. 3. Testbench for the filter response computation

The modern FPGAs contain the 6-input LUTs, which provide a one-stage network of the three-input adder [8]. In this situation, it is preferable to represent the filter coefficients as the rational numbers in the canonical binary number system:

$$c = k2^p + l2^q + m2^r, \quad (4)$$

where p, q, r are integers, $k,l,m \in \{0, 1, -1\}$.

Due to the usual method, the real coefficient values are derived. Then their truncated values, which are represented in a form (4), are searched near the solution point, which provides the optimum transfer function. This search process is performed by the scanning method [9], or by the evaluation optimization method [7, 10, 11]. In both situations, the searching for the optimum coefficients is performed by the VHDL program.

Because the number of different combinations of values p, q, r, k, l, m in (4) is comparatively small, all the possible coefficients c can be stored in a table or ROM, and these variables form the address word of this ROM. This simplifies the searching process.

The multiplier-less IIR filters, based on the allpass filter like (1), have the minimized number of the coefficients. This filter has the transfer function

$$H(z) = (A_1(z) + A_2(z))/2, \quad (5)$$

where $A_1(z)$, $A_2(z)$ are transfer functions of the allpass filters. These filters are composed of the sections described by the function

$$H_i(z) = \frac{b_i + c_i z^{-1} + z^{-2}}{1 + c_i z^{-1} + b_i z^{-2}} . \quad (6)$$

which originates from (3).

Such filters are stable and immune to small variations of the coefficients [3]. The last feature provides successful searching for the coefficients in the form (4). Moreover, in [11] a Stoyanov-Kawamata filter structure is utilized, which provides the minimum number of elements in the representation (4).

A section which calculates the function (6) with the maximum speed is described by the signal flow graph, which is illustrated by the Fig. 4. Here, the bars represent the register delays, circles represent the adders and coefficient multipliers, truncated circles are input and output of the filter. This graph is derived due to the retiming method described in [1,12,13].

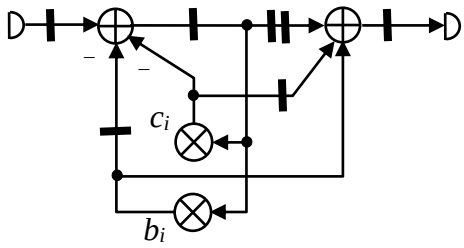

Fig. 4. Signal flow graph for the function (4)

The Fig.4 analysis shows that the respective network has only two multiplication units, it is fully pipelined, the critical path goes through an adder and a multiplier to the coefficient b_i. Therefore, the most performance is achieved when the coefficient b_i has the minimum of terms in its representation (4) or even is equal to zero.

The coefficients in the form (2) can be searched using the VHDL program as well. The iteration of such a program consists in the implementation of the following steps.

Firstly, a set of the coefficients b_i, c_i is selected from a table of coefficients, which are equal to (4). Then, these coefficients are put in the formulas (5) and (6), which are evaluated using the process operator shown in chapter II. The found effective coefficients are put in the respective adder tree, which calculates (4). Finally, the derived filter model is tested by the testbench in Fig. 3. The testing results are compared with the filter specification, and the found effective coefficient set is stored. After several iterations of this algorithm, the optimum set of the filter coefficients is selected. The resulting VHDL filter description and pipelining are performed using the pipelining method shown in [13].

IV. Experimental results

The method described above was used to build a set of IIR filters of the order from 5 to 9. These filters were put in the database of the Web tool IIR Filter Generator [13].

This application generates the synthesizable VHDL model of a filter with the given input and output bit width, and stop band frequency. This filter can be the low pass (LP), high pass or half band (HB) filter. Up to 27 models of HB filters with different frequency responses can be generated by this Web tool. Combining them, the filter with excellent characteristics can be built.

In Fig. 5 the frequency responses H_1, H_2 of two HB filters and their sequential connection $H_P = H_1H_2$ are shown. Here, f is the relative frequency, $f \in [0, 1]$. These charts are derived by the testbench shown in Fig.3. The parameters of the synthesized HB filter are shown in Table 1. Here, the hardware volume is given in the configurable logic block slices (CLBS).

The high clock frequency of this filter is derived due to the fact, that for this filter the coefficients $c_i = 0$. As a result, the structure becomes highly pipelined and has small hardware volume.

In Table 1, the results of the synthesis of another HB filter are given. This filter, as well as another filter for comparison, are designed and synthesized on the base of their signal flow graphs and coefficients, given in the references. This filter is one-staged, but it has up to 6 summands in the coefficient representation (2). As a result, the critical path becomes too long, and the maximum frequency is in 6.4 times lower than one for the proposed filter.

Another example is the synthesis of an LP filter with the cut frequency of $0.025f_S$, where f_S is the sampling frequency. The filter structure corresponds to (3), where $A_1(z)$ and $A_2(z)$ are transfer functions of 3-d and 4-th order, respectively. The coefficients found by the VHDL program in the canonical binary number system are equal to

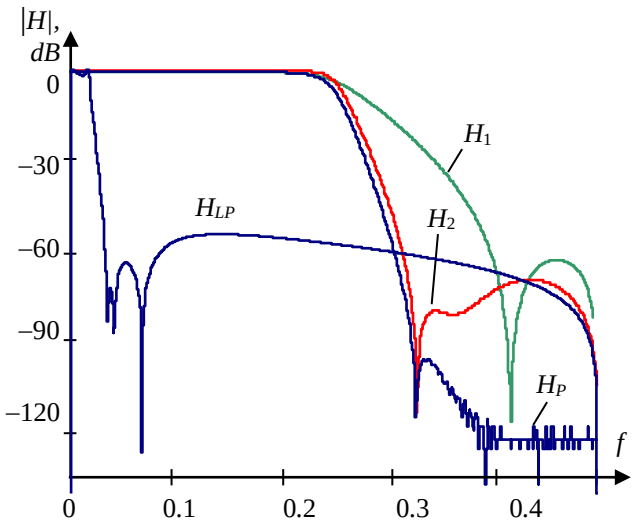

Fig. 5. Frequency response characteristics of the synthesized filters

$c_0 = -1.00\bar{1}01;$

$b_1 = 1.00\bar{1}01;$ $c_1 = -10.0000\bar{1};$

$b_2 = 1.00\bar{1}0\bar{1};$ $c_2 = -10.00\bar{1};$

$b_3 = 1.0000\bar{1}01;$ $c_3 = -10.0000\bar{1}00\bar{1}.$

The resulting filter transfer function HLP is shown in Fig. 5, and its characteristics are given in Table1. This filter has less hardware volume, much higher frequency, and the approximately equal suppression level, comparing to the analogous filter shown in [11]. This is explained by the fact, that the signal flow graph of the analogous filter has much longer critical path due to the complex Stoyanov-Kawamata scheme implemented in.

TABLE I. PARAMETERS OF FILTERS CONFIGURED IN XILINX KINTEX FPGA

Filter	Hardware, CLBS	Max. clock frequency, MHz	Suppression, dB	Reference
HB	203	690	120	—
HB	441	107	106	[15]
LP	179	310	54	—
LP	203	189	57	[11]

V. CONCLUSION

In this paper, it is shown that the VHDL language allows the designer to develop the digital filters without going beyond the editor and simulator. Thus, the possibility of rapid modeling of complex optimization process provides an effective search for optimal structural filter solutions.

Unlike the use of common design tools like Matlab, the VHDL simulator provides the operation control of the filter, taking into account both coefficient and data bit width and method of arithmetic operation implementation, as well as features of its configuring in FPGA.

A very effective VHDL testbench is proposed which not only proves the filter model correctness but generates the frequency response charts, which take into account all the data truncations, roundings, overflows, and saturations.

It was proven, that if the coefficients of the multiplierless filter have no more than three summands in their representation, then its pipelined implementation in FPGA has both the highest clock frequency and small hardware volume. The examples of the IIR multiplierless filter design show the effectiveness of the VHDL language use.

REFERENCES

[1] S. A. Khan, “Digital Design of Signal Processing Systems. A Practical Approach,” UK, Wiley, 2011.

[2] F. F. Daitx, V. S. Rosa, E. Costa, P. Flores, S. Bampi, “VHDL Generation of Optimized FIR Filters”, 2-nd Int. Conf. on Signals, Circuits and Systems, 7-9 Nov. 2008, pp.1-5.

[3] U. Meyer-Baese, “Digital Signal Processing with Field Programmable Gate Arrays”, 4-th Ed., Springer, 2014.

[4] P. J. Ashenden and J. Lewis, “The Designer’s Guide to VHDL”. Morgan Kaufmann, 2008.

[5] P. P. Vaidyanathan, P. Regalia and S.K. Mitra, “The Digital All-Pass Filter: A Versatile Signal Processing Building Block” Proc. IEEE. vol. 76. no. 1. pp. 19–37. Jan. 1988.

[6] A. Sergiyenko, “Testbench for the filter testing” Kiev: I. Sikorsky’s KPI, 2012. [Online]. Available: http://kanyevsky.kpi.ua/en/useful-ip-cores/testbench-for-the-filter-testing/

[7] L. D. Milic and M. D. Lutovac, “Design of multiplierless elliptic IIR filters with a small quantization error” IEEE Trans. on signal processing. vol. 47, no. 2, pp. 469–479. Feb. 1999.

[8] S. Churiwala, Ed. Designing with Xilinx FPGAs Using Vivado, Springer. 2017.

[9] A. T. Mingasin, “Digital filter synthesis for the high speed system on the chip” Digital Signal Processing. no. 2. pp. 14 – 23. 2004, in Russian.

[10] S.-T. Pan, “CSD-Coded Genetic Algorithm on Robustly Stable Multiplierless IIR Filter Design” Mathematical Problems in Engineering, Hindawi Publ, vol. 2012, Article ID560650, 15 P.

[11] V. I. Anzova, J. Yli-Kaakinen, T. Saramaeki, “An Algorithm for the Design of Multiplierless IIR Filters as a Parallel Connection of Two All-Pass Filters”. IEEE Asia Pacific Conf. on Circuits and Systems, APCCAS. 2006. pp. 744-747.

[12] M. Potkonjak, J. M. Rabaey “Maximally and Arbitrarily Fast Implementation of Linear and Feedback Linear Computations”, IEEE Trans. On Computer Aided Design of Integrated Circuits and Systems, vol. 19, pp. 30-43, Jan. 2000

[13] A. Sergiyenko, A. Serhiyenko, A. Simonenko, “A method for synchronous dataflow retiming”, 1-st IEEE Ukraine Conference on Electrical and Computer Engineering (UKRCON), pp. 1015-1018, 2017.

[14] A. Sergiyenko, “VHDL design of multiplier-free IIR filters“, Kiev: I. Sikorsky’s KPI, 2016. [Online]. Available: http://kanyevsky.kpi.ua/GEN_MODUL/APgen/APMF_help.php

[15] K. S. Yeung, S.C. Chan, “The Design and Multiplier-Less Realization of Software Radio Receivers With Reduced System Delay”. IEEE Trans. On Circuits and Systems – Regular Papers, vol. 51, pp. 2444–2449. Dec. 2004.