FPGA – based Arbitrary Integer Frequency Divider with 50% Duty Cycle of the Output Signal

Marieta Georgieva Kovacheva, Eltimir Chavdarov Stoimenov and Ivailo Milanov Pandiev

Abstract – **The paper describes modeling and realization of arbitrary frequency divider with integer coefficient. The main advantage of the proposed divider is the fact that the output signal is with 50 % duty cycle for arbitrary value of the divider's coefficient. The device is implemented on VHDL and it is simulated in the simulator ISIM, part of Vivado design suite. The FPGA realization is done on ZedBoard, based on Xilinx Zynq 7020 SoC. For the proposed arbitrary frequency divider an in depth experimental study is conducted. The experiments show very good results for the relative error of the duty cycle in the frequency range from 0 to 1*MHz*.**

Keywords – **Arbitrary integer frequency divider, FPGA, SoC, VHDL, Duty cycle**

I. Introduction

Programmable dividers find excessive application in variety of digital systems like frequency synthesizers, clock generators and demultiplexers [1, 2, 3]. In the general case the frequency of the divider's output signal must be *N* times lower than the input one – f_{in} :

$$f_{out} = \frac{f_{in}}{N}, \quad (1)$$

where the coefficient *N* could be any integer number.

In many cases the frequency dividers are implemented only with even coefficient *N* , as this is sufficient for some application and greatly facilitates the realization.

Another very important parameter for the frequency dividers is the duty cycle - δ of the output signal. It is very desirable for the output signal to preserve the duty cycle of the input, especially when δ =50%. This is very important for applications like clock generation and frequency synthesizers when some specific types of phase detectors are used [15].

On the one hand the design process of a frequency divider with an even coefficient *N* and 50 % duty cycle is a trivial task. Most frequently it is realized with a programmable counter working only on the one edge of the input clock signal. In this case the output signal frequency is given by:

$$f_{out} = \frac{f_{in}}{2N}, \quad (2)$$

where *N* is an arbitrary, integer even number.

On the other hand realization of a frequency divider with an odd coefficient *N* and δ =50% introduces certain design challenges [4, 5]. The problem arises from the fact that the divider's output signal should toggle on the both edges of the input signal. On the other hand, as a general, the digital logic is sensitive only to the one edge of the signal (usually rising), but not for the both. Different solutions of this problem can be found in many sources like [2, 3, 6 ,7].

The authors in [9] introduce a special *duty cycle trimming circuit*. This technique allows the use of standard counters, working on the rising edge of the input signal, which serves as arbitrary frequency dividers. The *duty cycle trimming circuit* is added after the counters to form δ = 50%. A drawback of this approach is the fact that the realization of the *duty cycle trimming circuit* requires specific transistor circuitry and could not be implemented with a standard digital logic.

In [8] the authors propose arbitrary frequency divider with duty cycle which is not exactly, but it is up to 50 %. Unfortunately this is not acceptable in the cases, already mentioned above, when the output signal should have δ exactly 50%.

This paper proposes a frequency divider with an integer and arbitrary coefficient and output signal with δ exactly equals 50%. The divider is implemented on VHDL and uses little resources from the FPGA. The design is simulated, realized and tested on a Xilinx Zynq SoC.

II. Architecture of the Proposed Frequency Divider. VHDL Implementation

As it was already noted, the realization of arbitrary frequency divider with duty cycle of the output signal equals 50 % could be a challenging task. The reason for this is the fact that for odd *N* the output signal must toggle on both - rising and falling edge of the input signal. We are trying to overcome this obstacle by proposing the architecture shown on Fig. 1. There are three counters in the divider's structure. The first counter – *even_cnt* is only necessary when *N* is even. This counter counts to *N*/2 and then restarts from zero.

The other two counters count on the rising and on the falling edge of the input signal *clk_in*, respectively. They are used in the case when *N* is odd. These two counters should count to *N*.

M. Kovacheva is with the Department of Electronics and Electronics Technologies, Faculty of Electronic Engineering and Technologies, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: m_kovacheva@tu-sofia.bg

E. Stoimenov is with the Department of Electronics and Electronics Technologies, Faculty of Electronic Engineering and Technologies, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: e_stoimenov@tu-sofia.bg

I. Pandiev is with the Department of Electronics and Electronics Technologies, Faculty of Electronic Engineering and Technologies, Technical University - Sofia, 8 Kliment Ohridski blvd., 1000 Sofia, Bulgaria, e-mail: ipandiev@tu-sofia.bg

The *reset circuit* block activates the reset signal of the counters when a new value of *N* is programmed. Thus the counters will start counting from zero every time new value of *N* is applied. This is absolutely necessary for the proper operation of the frequency divider.

Depending on the value of *N* two scenarios are possible. If *N* is even than the switch *SW* is in its left position and the output of the *even_cnt* counter is directly applied to the output of the frequency divider.

If *N* is odd, *SW* is in its right position. In this case the values from the *rise_cnt* and *fall_cnt* counters are taken and after simple logical operations the frequency divider output is derived.

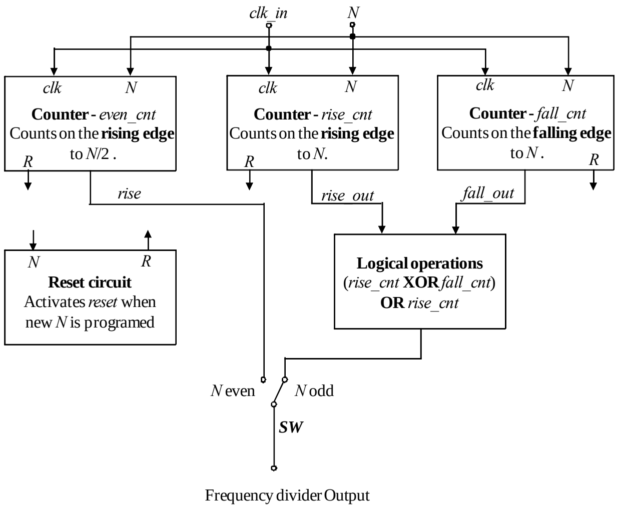

Fig. 1. Architecture of the proposed frequency divider.

The architecture proposed above allows output signal with 50 % duty cycle for any integer *N,* odd or even. The frequency divider with the described architecture is implemented with VHDL in Vivado 2014.1 design suite. Some key parts of the source code are given below.

A) **Counters**. The three counters from the structure are organized as processes in a classical manner. Below the process of the *even_cnt* is given as an example. The *even_cnt* variable increments on every rising edge of the input clock till it reaches N/2. Than the external signal *rise* is toggled.

```
process(clk_in)
variable even_cnt : integer;
begin
if reset ='1' then
even_cnt := 0;
else
   if rising_edge(clk_in) then
      if even_cnt = n/2 then
       even_cnt := 0;
       rise <= not rise;
      else
       even_cnt := even _cnt + 1;
      end if;
   end if;
end if;
end process;
```

The same is valid for the *rise_cnt* and *fall_cnt* counters except that they count to *N.*

B) **Logical operation.** This is a purely combinational block and it is only necessary when an odd *N* is applied. Firstly the block XORs the current values from the *rise_cnt* and *fall_cnt* counters. The result is logically AND-ed with the *rise_cnt* value. The output signal of the block is with 50% duty cycle and could be applied to the frequency divider output.

For simplicity the **Logical operation** and the **SW** blocks are combined in the code below:

```
q_out <= rise when n(0) = '1'
else ((rise_out xor fall_out) or rise_out);
```

C) The **Reset circuit** constantly polls the value of *N.* If a new value is programmed then the *Reset* signal is activated and the counters are cleared. This is absolutely necessary for the proper operation of the frequency divider.

```
process(clk_in,ctrl)
variable n_old,n_new : std_logic_vector (7 downto 0) :=
"00000000";
begin
if rising_edge(clk_in) then
n_new := '0' & ctrl(6 downto 0);
   if n_old /= n_new then
      reset <= '1';
      n_old := n_new;
   else
      reset <= '0';
   end if;
end if;
n <= unsigned (n_new);
end process;
```

Here *ctrl* is an input port which sets the value of *N* . As it can be seen *N* uses only the first 6- bits from the *ctrl* port. Off course the width of *N* could be changed to any desired value.

The presented divider is realized on ZedBoard development kit. The board is equipped with Xilinx Zynq 7Z020 System on Chip (SoC) [10]. The design utilization is shown in Table 1.

TABLE. 1. FPGA RESOURCES UTILIZATION

Resources	Utilization	Utilization, %
Slice LUTs	166	1
Slice register	112	1
IO	18	9
Clocking	1	3

It should be noted that some auxiliary IO ports and logic is added to the design for testing purposes. Nevertheless the utilization of FPGA resources is extremely small.

III. SIMULATION AND EXPERIMENTAL RESULTS

The workability and the functional parameters of the proposed frequency divider are examined with series of simulation and experimental studies. The behavioral simulation results are obtained with the simulator ISIM, part of Vivado design suite [11].

For simulation purposes the divider coefficient is chosen with an even and an odd value, respectively. For clarity the coefficient is set to 4 and 5, respectively. Fig. 2a shows the simulation results when *N* is equal to 3, and Fig. 2b shows results for *N* equal to 4. It should be noted that the divider coefficient *N* on the oscillograme must be incremented with one to derive the true dividing coefficient. The period of the input clock signal *clk_s* is *10ns*, *q_out* is the divider output signal.

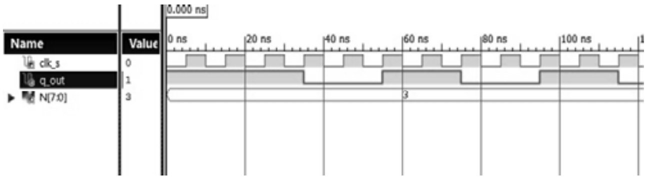

Fig. 2a. Simulation results for *N*=3.

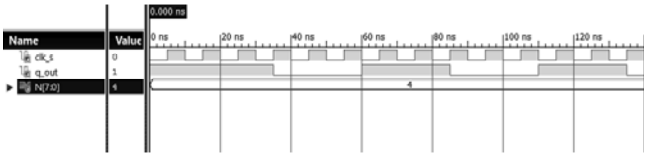

Fig. 2B. Simulation results for *N*=4.

Several experimental studies designed to determine the basic parameters of the proposed frequency divider have been conducted. The test equipment constitutes: pulse Generator – HP 8112A [12]; digital storage oscilloscope – Tektronix TDS1012B [13]; and FPGA development board – ZedBoard, rev. D [14].

The charts on Fig. 3a and 3b show the relative error ε_f of the output frequency versus the frequency of the input signal. Fig. 3a shows the experimental results for even value of *N*, while – Fig. 3b is for an odd *N*. The error ε_f for both cases is less than 2.5% for the frequency range from 0 to 10 M*Hz* and less than 0.5% for frequency range from 0 to 5 M*Hz*.

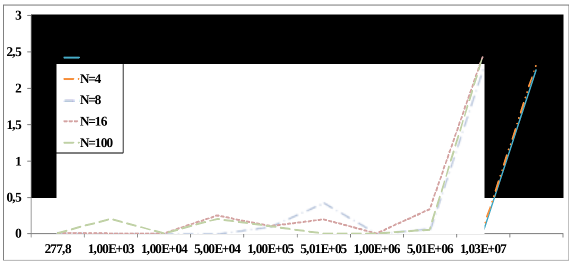

Fig. 3 a. Relative error of the output frequency versus input frequency at even integer value of *N*

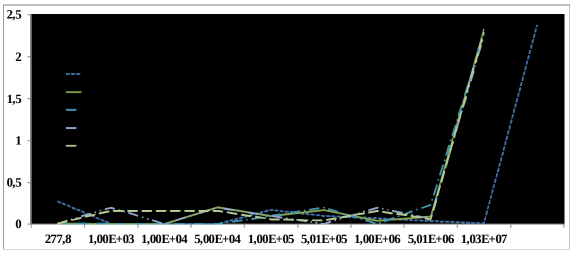

Fig. 3 b. relative error of the output frequency versus input frequency at odd integer value of *N*

Fig. 4a and 4b shows the error ε_δ for the duty cycle of the output signal as a function of the input frequency. The results are shown separately for the even and odd values of *N*.

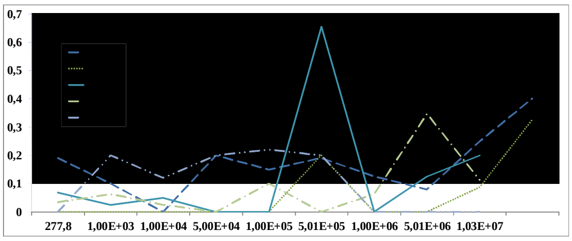

Fig. 4a. Relative error of the duty cycle versus the output frequency at even integer value of *N*

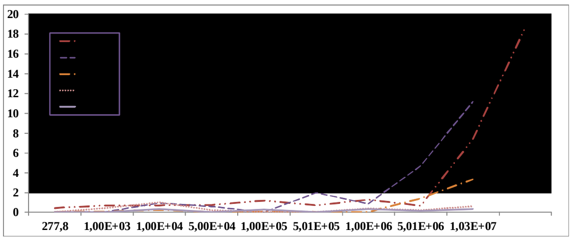

Fig. 4b. Relative error of the duty cycle versus the output frequency at odd integer value of *N*

It can be noted that for even values of *N* the relative error ε_δ is less than 0.7% for the whole frequency range (0-10 M*Hz*). In contrast, for odd *N* the error increases considerably, especially for the small values of *N*. For *N* = 3 ε_δ reaches approximately 18% and for *N*=5 ε_δ it is about 11%. Nevertheless for the frequency range from 0 to 1 M*Hz* the ε_δ is less than 1% and decreases when *N* becomes larger.

Table 2 summarizes the experimental results for the mean and the maximum value of the ε_f and ε_δ at various values of *N*.

TABLE 1. MEAN AND MAXIMUM VALUE FOR ε_f AND ε_δ

N=2			*N*=3		
error	Mean	Max	error	Mean	Max
ε_δ, %	0.17	0.40	ε_δ, %	0.7	18.61
ε_f,%	0.08	2.24	ε_f,%	0.05	2.37
N=4			*N*=5		
error	Mean	Max	error	Mean	Max
ε_δ, %	0	0.33	ε_δ, %	0.84	11.17
ε_f,%	0.02	2.31	ε_f,%	0.09	2.32
N=8			*N*=17		
error	Mean	Max	error	Mean	Max
ε_δ, %	0.05	0.65	ε_δ, %	0.05	3.32
ε_f,%	0.01	2.32	ε_f,%	0.01	2.27
N=16			*N=101*		
error	Mean	Max	error	Mean	Max
ε_δ, %	0.06	0.3458	ε_δ, %	0.20	0.98
ε_f, %	0.01	2.44	ε_f, %	0.01	2.25
N=100			*N*=127		
error	Mean	Max	error	Mean	Max
ε_δ, %	0.12	0.22	ε_δ, %	0.16	0.35
ε_f,%	0.06	2.44	ε_f,%	0.16	2.29

As can be seen in Table 1 the mean value of ε_δ is less than 1% and the maximum value is less than 20% for any integer *N* in the frequency range from 0 to 10M*Hz*. The mean value of ε_f is less than 0.2% and the maximum value is less than 2.5% for any integer value of N in the same frequency range.

The oscillograms on Fig. 5a and 5b gives the input and the output frequency divider signals at N=3 and N=4, respectively. The input signal is with frequency equals to 1M*Hz* and it is applied on Channel 1. Channel 2 shows the form of the output signal.

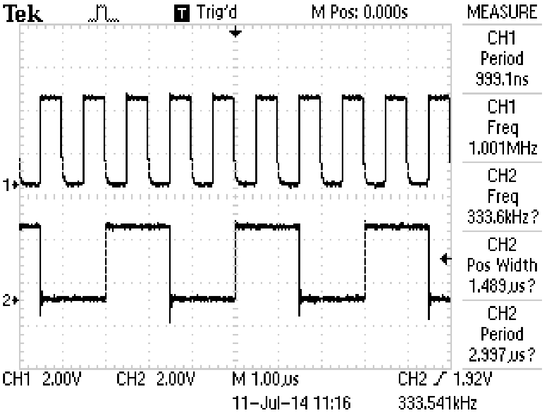

Fig. 5a. Frequency divider input and output signal at *N=3*

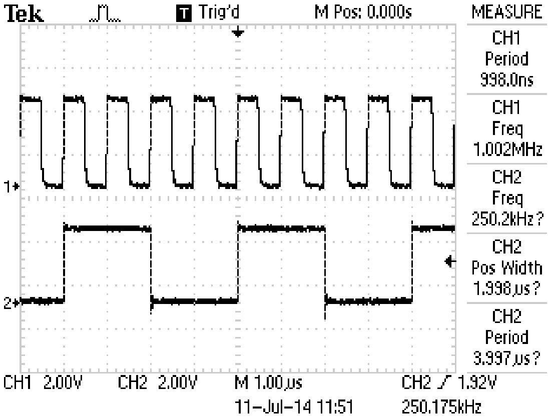

Fig. 5b. Frequency divider input and output signal at *N=4*

The oscilloscope on screen measurements clearly show that the output frequency is N times lower than the input one. Moreover δ of the output signal is almost exactly 50%.

VI. Conclusion

The frequency dividers are very important topic in the mixed-signal and digital electronics. The reason for this is their huge practical importance in a various types of electronic systems. As there are plenty of realizations of such devices, our research shows that there are very few realizations of frequency dividers with arbitrary coefficient and 50% duty cycle of the output signal. The device proposed in this paper tries to fill this gap and introduces a frequency divider with arbitrary integer coefficient and output signal with exactly 50% duty cycle. The experimental studies show that the relative error for the duty cycle is less than 1% in the frequency range from 0 to 1M*Hz.* The error increases to approximately 20% for the frequency range of up to 10M*Hz* only for the odd values of *N*. Nevertheless the output frequency relative error is less than 2.5% for any integer *N*, odd or even, in the whole frequency range 0-10M*Hz*.

Our future efforts will be focused on improvement of the proposed divider structure and finding the reason for the increased duty cycle relative error ε_δ .

References

[1] Universal Serial Bus 3.0 Specification, Revision 1.0, Nov. 2008 [Online], http://www.usb.org.

[2] Y. Moon, S.-H. Lee, and D. Shim, “A divide by 16.5 circuit for 10-Gb Ethernet transceiver in 0.13um CMOS”, IEEE J. Solid-State Circuits, vol. 40, no. 5, pp. 1175-1179, May 2005”

[3] C.-C.Meng, T.-H. Wu, J.-S. Syu, S.-W. Yu, K.-C. Tsung, and Y.-H. Teng, “2.4/5.7-GHz CMOS Dual-Band Low-IF Architecture Using Weaver-Hartley Image-Rejection Techniques”, IEEE Trans. Microw. Theory Tech., vol 57, no. 3, pp. 552-561, March 2009.

[4] MAO Weiyong, QI Zhongyang,WANG Lan. Design for decimal frequency divider based on FPGA. Joural of GuiLin College of Aerospace Technology, vol.53 (1), Jan. 2009, pp.30 - 32.

[5] ZHOU Dongcheng, WANG Yongbin, ZHENG Yaping. “Design of Fractional-N Frequency Synthesizer based on FPGA,” Electronic Measurement Technology, vol.29(3), Mar. 2006, pp.79 – 93.Measurement Technology, vol.29(3), Mar. 2006, pp.79 – 93.

[6] S. -C. Tseng, H.-J. Wei, J. –S Syu, C. Meng, K.-C. Tsung, and G.-W. Huang, “True 50% Duty-Cycle High-Speed Divider with Modulus of Odd Numbers”, Proc. Asia Pacific Microwave Conference, pp. 305-308, Dec. 2009

[7] R. Magoon and A. Molnar, “RF Local Oscillator Path for GSM Direct Conversion Transceiver with True 50% Duty Cycle Divide by Three and Active Third Harmonic Cancellation”, Proc. IEEE RFIC Symposium, pp. 23-26. 2002.

[8] S. Zhang, C. Zhao, “Design for realizing arbitrary fractional divider based on FPGA which duty cycle is up to 50%”, Proceedings of the 2nd International Symposium on Computer, Communication, Control and Automation (ISCCCA-13), Atlantis Press, France, 2013.

[9] B. Sangjin, C. Son, J. Kim “Simple odd number frequency divider with 50% duty cycle”, IEICE Electronics Express, Vol.9, No15, pp. 1249-1253, 2012.

[10] Zynq-7000 All Programmable SoC Overview, Xilinx, Application Note: DS190 (v1.6), December 2, 2013.

[11] ISim User Guide, Xilinx, UG660 (v14.1), April 24, 2012.

[12] Operating, Programming and servicing Manual, HP 8112A 50MHz Programmable Pulse Generator, Hewlett Packard, Germany, January 1992.

[13] Digital Storage Oscilloscopes, TDS1000B Series, TDS2000B Series, Tektronix.

[14] ZedBoard, Getting started guide, Ver. 7, Avnet, January 2014.

[15] 74HC/HCT4046A Phase-locked-loop with VCO, Product specification, Philips Semiconductors, November 1997.