elektrorevue
ISSN 1213 - 1539

Generating PWM Signals With Variable Duty From 0% to 100% Based FPGA SPARTAN3AN

Ziad Nouman, Bohumil Klima, Jan Knobloch

Department of Power Electrical and Electronic Engineering
Brno University of Technology

Email: xnouma00@stud.feec.vutbr.cz, klima@feec.vutbr.cz, xknobl00@stud.feec.vutbr.cz

Abstract *– This article deals with the generation PWM signals with variable duty from 0% to 100% using VHDL and its application in field programmable gate arrays. The article also discusses the usage DCM for decrease the clock frequency. DCM is a digital clock manager that is useful to decrease the skew of clk signal when we want to divide the clk frequency. We used a fixed frequency to produce the input data that generate the PWM signals using one comparator. The comparator compares between two input data. First data is generated using PWM counter and second data is generated by up-down counter using two push buttons. PWM has a fixed frequency and a variable voltage. This voltage value changes for 0V to 2.5 V. Inside signals are monitored on the computer by platform cable usbII and ChipScope program. We need a board fpga and ISE package version14.4.*

1 Introduction

Pulse width modulation (PWM) is a powerful technique for controlling analog circuits with a processor's digital outputs. PWM is employed in a wide variety of applications, ranging from measurement and communications to power control and conversion.

The applications of PWM are used in control systems like DC-DC converters have been explained in references [1], [2], [3] where the PWM signals were used to convert unregulated DC voltage to regulated or variable DC voltage at the output. In AC motor drives, PWM inverters allow to control both frequency and magnitude of the voltage and current applied to the motor.

A PWM signal is not constant, the main parameter is a duty cycle D that is a part of PWM period and describes the proportion of on time to regular interval. The equation (1) describes the duty cycle as the following:

$$D = \frac{\tau}{T} \tag{1}$$

Where: $0 \leq D \leq 1$
Thus the output signal is calculated in equation (2):

$$output = D \times input = \frac{t_{on}}{t_s} \times input \tag{2}$$

Figure 1 shows the duty cycle of PWM.

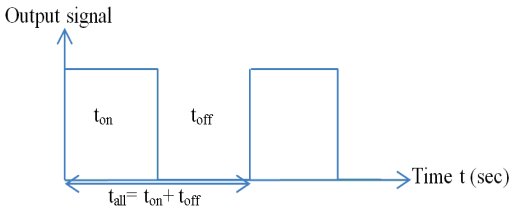


Figure1: PWM with duty cycle

The main advantage PWM is that power loss is very low in the switching devices.

Many digital circuits can generate PWM signals [1]. In this article a simple method is used where a sawtooth waveform and carried data are required. A comparator compares between two values in order to generate pulse width modulation. Hardware Description Language (VHDL) is used to generate the required signals in FPGA.

Engineers use microcontrollers largely to build control system [4], but the microcontrollers are going to disappear with the appearance FPGA. With FPGA, all control system features can be added to a single chip, and new functions can be used by the designer.

This article scans the steps of building a PWM signals using VHDL in FPGA XILINX, these signals can be used in many control systems such as: DC-DC converter, DC motors and DSP system. Board FPGA, ISE software and platform cable USB II are necessary to implement the design.

2 Strategy of building a PWM signals in FPGA using VHDL

The principle of PWM work is required to design PWM. Figure 2 illustrates the principle of work [5]:

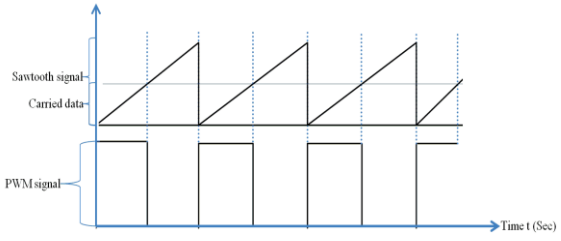


Figure 2: Principle PWM

elektrorevue
ISSN 1213 - 1539

As figure 2 shown the design of processor that generates the signals, which give the output of PWM, requires a comparator that compares between two values. The first value represents the sawtooth signal and the second value represents the data that is entered by using switches or buttons on the FPGA board [5],[6].The input signal can take another form such as sinusoidal signal [6]. When the design requires generating a sinusoidal PWM in FPGA, it must generate LUT using MATLAB and these data must be saved in block memory to control the PWM output.
Figure 3 shows the RTL schematic of the design:

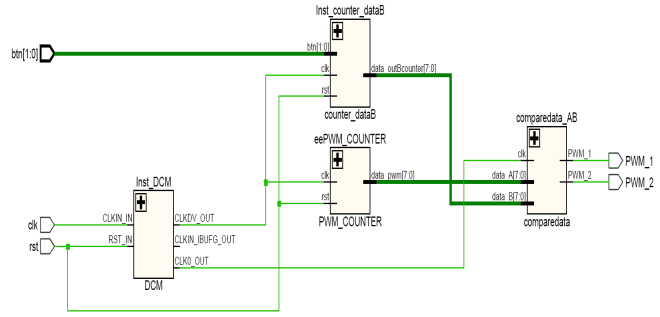


Figure 3: RTL schematic of PWM components

2.1 Digital Clock Manager (DCM)

Figure 4 shows the architecture of DCM in FPGA:

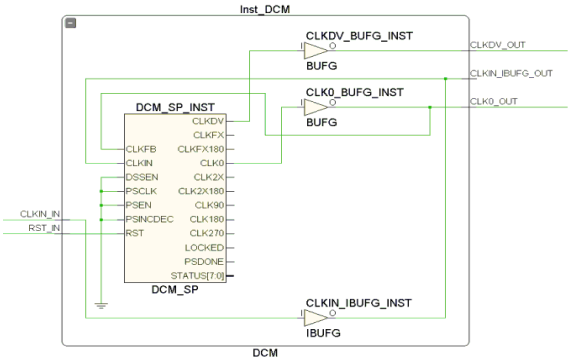


Figure 4: Digital clock manager

The properties of DCM$_s$ [7] depend on the FPGA frequency and the type of this chip. DCM$_s$ provide advanced clocking capabilities to FPGA application.

In this design, pin CLKDV is used to generate clk output with a frequency 3 MHz. This value is used to increase PWM counter.

When CLKIN frequency is 50 MHz in FPGA SPARTAN3- AN, the output frequency of digital clock manager is calculated by equation (3):

$$CLKDV = \frac{CLKIN}{divider} = \frac{50}{15} = 3MHz \qquad (3)$$

CLK0 frequency is the same CLKIN (50 MHz) and it is used to increase or decrease the counter of input data.

2.2 PWM Counter

Figure 5 illustrates the architecture of PWM counter. This counter is used to generate the sawtooth signal using 8bits.

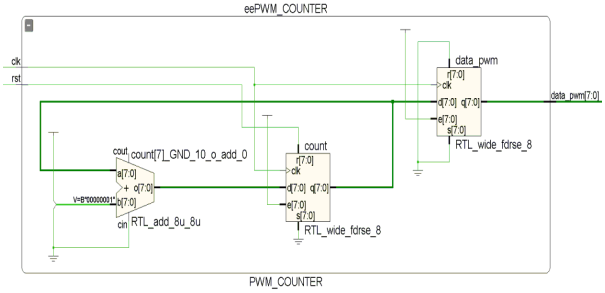


Figure 5: Architecture of PWM counter

The code of this counter is written using VHDL. The counter increases one bit every CLKDV rising and it counts from $(00000000)_2$ to $(11111111)_2$. The maximum value is $(255)_{10}$, when PWM counter arrives to this value, it returns to zero. The counter time from zero to maximum value is 85 µs as the following figure shows:

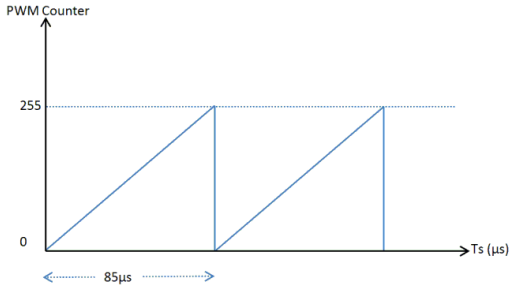


Figure 6: PWM counter signal (triangular signal)

From figure 6, it can deduce the equation (4) that presents the signal frequency:

$$F_s = \frac{1}{T_s} = \frac{1}{85} \approx 11.8\ KHz \qquad (4)$$

This value represents the frequency of PWM signal, but this value is not constant and it can be controlled by CLKDV frequency or CLK0 that can be used as a CLK frequency of PWM counter. This depends on application place.

2.3 Control data B

Control data B or counter data B illustrated in figure 3, allows to control the value of data that is carried for comparing with the value of PWM counter and the result will be PWM output signal.

Control data is a counter whose direction can be controlled by two bits. When the bits are “10” the counter counts up and when they are “01” this counter counts down. Two push buttons on FPGA board represent these two bits.

elektrorevue
ISSN 1213 - 1539

Clock signal of this counter comes from CLKO and counts from 0 to 0x3FFFFFF. This means that counter has 26 bits, but the output data that will be compared with the PWM counter signal, must have only 8 bits in order to correspond with the same bits number. The following function can be written in order to solve this problem:

data_outBcounter <= countdatab (25 downto 18);

And the code that generates the carried signal can be written as the following:

```
entity counter_dataB is
    Port (clk  : in  STD_LOGIC;
          btn : in  STD_LOGIC_VECTOR (1 downto 0);
          rst : in  STD_LOGIC;
   data_outBcounter : out  STD_LOGIC_VECTOR (7 downto 0));
end counter_dataB;
architecture Behavioral of counter_dataB is
signal countdatab : std_logic_vector(25 downto 0);
begin
process (clk,rst,btn)
    begin
    if (clk'event and clk='1') then
       if rst= '1' then
         countdatab<=(others=>'0');
       end if;
     if((btn(1)='1') and (btn(0)='0') and (countdatab<X"3FFFFFF"))
         then countdatab<=countdatab+1;
     end if;
    if((btn(0)='1') and (btn(1)='0') and (countdatab>X"0000000"))
    then countdatab<=countdatab -1;
    end if;
data_outBcounter <= countdatab (25 downto 18);
end if;
end process;
end Behavioral;
```

As this code shows, the process depends on the clock signal and the buttons. The output data are generated, if the signal of one button is high. Function (if...end) operates this condition as mentioned.

2.4 Comparator data

The output of comparator will be PWM generator. The scheme of comparator in this design is shown in figure 7:

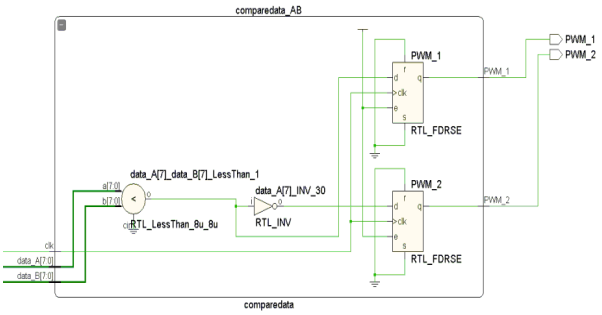


Figure 7: Comparator data and the output of PWM generator

As figure 7 shows, the comparator has three inputs and two outputs: First input, signal data A is coming from output of PWM counter. Second signal, data B is coming from the controlled data output. Third signal, clk signal is coming from DCM.

Comparator compares between data A and data B, if data A is less than data B, the output register will be 1 and the output of PWM1 D-flip-flop will also 1, but the output of PWM2 D-flip-flop will be 0 because the existence of inverter between this D-flip-flop and RTL register. If data A is greater than data B, the output register will be 0 then PWM1 will be 0 and PWM2 will be 1.

Comparator outputs are PWM1 and PWM2 that can be used to control the device (DC motor, DC-DC converter, ADC...).

After writing all codes in VHDL, top level file must be built and the design is synthesized in order to correct the errors of programming if exist.

The scheme of design after creation of top level file is a black box as figure 8 shows:

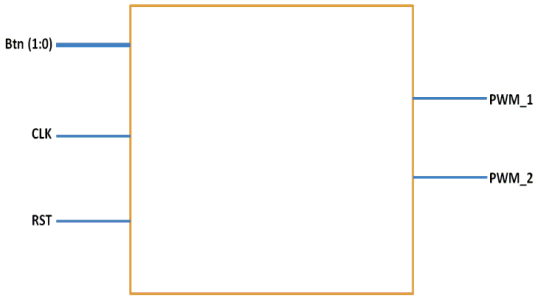


Figure 8: Scheme of top level design with the PWM signals

From figure 8, the input signals and output signals can be seen. Reset signal returns the system to the beginning, and then PWM outputs will be 0. A push button on FPGA board can be configured to reset the system.

3 Test design and monitoring signals

First, timing constraints code must be created. This code allows configuring and using peripheries on FPGA board [8].

The timing constraints code can be written as the following:

```
# PlanAhead Generated physical constraints
NET "btn[1]" LOC = U15;
NET "btn[0]" LOC = T15;
NET "clk" LOC = E12;
NET "PWM_1" LOC = R20;
NET "PWM_2" LOC = T19;
NET "rst" LOC = T14;
```

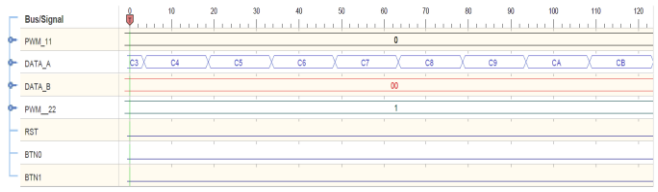
After creating the timing constraints code, it must add the LogiCore IP ChipScope Pro that allows analyzing the design [9]. This IP core is a virtual input/output that can monitor and drive internal FPGA signals in real time.

elektrorevue
ISSN 1213 - 1539

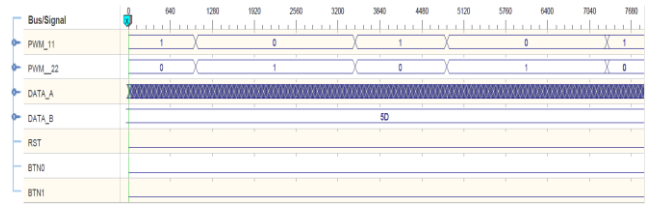
After adding this device, it can generate the programming file to obtain the bit stream that must be downloaded on the FPGA board, and then it can monitor and analyze the signals using ChipScope.

As figure 9-a shows, when the program is downloaded onto FPGA board, the data B is 0 because any button doesn't yet pushed. Value of data A is changing dependent on the PWM counter output. Because data A is greater than data B PWM1 is 0 logic and PWM2 is 1 logic, then the LED (T19) is lighting and LED (R20) is off on the board.

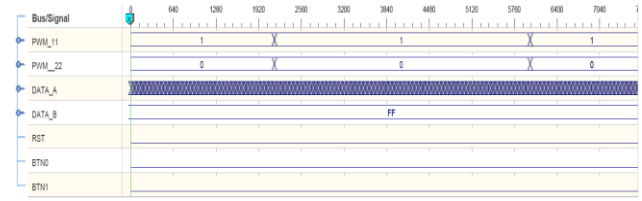
When the up button is pushed, data B will begin to have a value, 0x5D as Figure 9-b shows, when data A is less than 93, PWM1 is 1, and when data A is greater than 93 PWM2 is 1.



(a)



(b)



(c)

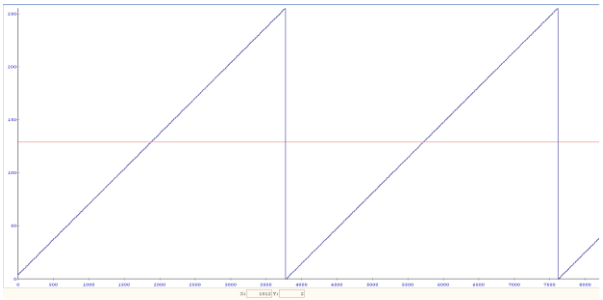
Figure 9: Analyzing of input/output signals of PWM generator: (a) output data when carried signal is equal zero, (b) output data when carried signal is equal 93, (c) output data when carried signal is equal the peak of sawtooth signal.

When data B is 0xFF, PWM1 will be 1 across all the period of output signal and PWM2 will be 0 as Figure 9-c shows.

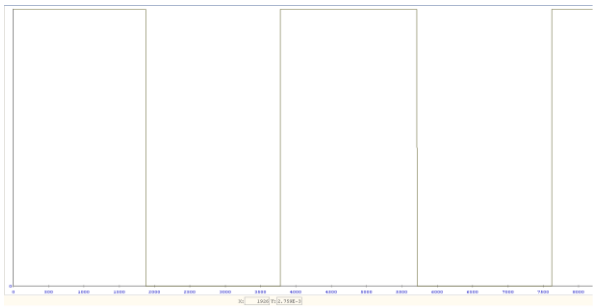
It can also use the BUS PLOT to analyze the data input and data output.

Figure 10-a shows the signal of data in (A) and signal of data in (B), data A represents the sawtooth signal that will be compared with data B. Data B represents the carried signal.

Figure 10-b shows the output signal of PWM generator that is produced by the comparison between data A and data B. As it is seen, when the value of sawtooth signal is greater than the carried signal, PWM1 will be 0, and when the carried signal is greater than sawtooth signal the PWM1 signal will be 1.



(a)



(b)

Figure10: Analyzing data using bus plot in ChipScope core: (a) sawtooth signal with the carried signal, (b) PWM signal.

4 Conclusion

In this article, the generation of PWM signals is discussed using VHDL based on FPGA. A board SPARTAN3AN is used as a hardware and ISE14.4 XILINX is used as software. A PWM counter and the algorithm are designed that allow to enter the data using two push buttons on the board. The comparator is necessary to compare between the data that generate PWM signals. The generated PWM signals have a fixed frequency (11.8 KHz) depended on the frequency of sawtooth, and a variable duty cycle that changes from 0% to 100%. But the frequency can be changed using DCM manager, without editing the program. After creating the design, the data is analyzed using IP CORE CHIPSCOPE.

These signals can be used to drive a DC motors, DC-DC converter, AC motors, and robots. But the applications of FPGA don't have limitations in these parts of control system, and it can be used instead of microcontrollers in the DSP systems, communications… .

Acknowledgment

This work was supported by the European Regional Development Fund under project No. CZ.1.05/2.1.00/01.0014 and by the faculty project FEKT-S-11-14 "Utilization of new technologies in the power electronics".

elektrorevue
ISSN 1213 - 1539

References

[1] MUNTEAN, N, Gavris M a Cornea O.: *Dual input hybrid DC-DC converters*. EUROCON - International Conference on Computer as a Tool (EUROCON), 2011 IEEE: Conference.27-29 April 2011, ISSN: 978-1-4244-7486-8, 1-4.Available.from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5929268

[2] AGNIHOTRI, P., Kaabouch, N., Salehfar, H., Wen-Chen, HU.: *FPGA-based combined PWM-PFM technique to control DC-DC converters*. North American Power Symposium (NAPS), 2010 IEEE Conference .Sept. 2010, ISSN:978-1-4244-8046-3,1-6 available from: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5929268

[3] EFTICHIOS, K., Apostolos, D.,Kostas, K.: *High-frequency pulse width modulation implementation using FPGA and CPLD ICs*. Journal of Systems Architecture 2006, vol:52, issue:332–344. Available from: http://www.sciencedirect.com/science/article/pii/S1383762105001049#

[4] SHRUTI, S., Jageshwar, R., Amit, A.: *Controlling DC Motor using Microcontroller (PIC16F72) with PWM*. International Journal of Engineering Research. 01 Dec. 2012, Vol .1, Issue No.2, s. 45-28. Available from: http://www.ijer.in/ijer/publication/v1s2/p%2045-47%20ijer_jageshwar.pdf

[5] KRÁL, J.: *Řešené Příklady ve VHDL Hradlová Pole FPGA pro začátečníky*. Praha: BEN-Technical literature, 2010. ISBN 978-80-7300-257-2.

[6] PONG P. *FPGA Prototyping by VHDL examples Xilinx SPARTAN-3 VERSION*. Hoboken, New Jersey,USA: John Wiley & Sons,Inc, 2008. ISBN 978-0-470-18531-5.

[7] XILINX. Spartan-3 Generation FPGA User Guide: Extended Spatan-3E, and Spartan-3 FPGA Families .UG331, V1.8. June 13, 2011 .available from: http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

[8] Xilinx.Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334. V1.1. June 19, 2008. Available from : http://www.gta.ufrj.br/ensino/EEL480/spartan3/ug334.pdf

[9] XILINX. Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334. V14.3. October 16, 2012. Available from :http://www.xilinx.com/support/documentation/sw_manals/xilinx14_3/chipscope_pro_sw_cores_ug029.pdf