ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

FIELD PROGRAMMABLE GATE ARRAY BASED PULSE WIDTH MODULATION CONTROLLER

Muzakkir Mas'ud Adamu*, Zakariyya Hassan*, Abubakar Ahmed, Tonga Agadi Danladi**, Bashir Ahmend Danzomo*****

*Depertment of Computer Engineering, **Depertment of Electrical and Electronic Engineering, *** Depertment of Mechanical Engineering
Hussaini Adamu Federal Polytechnic Kazaure, Jigawa State Nigeria.

DOI: 10.5281/zenodo.46521

ABSTRACT

A pulse width modulation (PWM) signal controller is implemented in a digital circuit to control the speed of a DC motor. The PWM controller modules are designed by adopting the very high-speed integrated circuit hardware description language (VHDL) and the Xilinx Spartan-3E starter board, field programmable gate array (FPGA). The use of PWM control for DC motors is widely used due to reliable performance. The starting torque, for example, in a DC motor can be higher several orders in magnitude than that for a comparable size AC motor. PWM control for DC motors enables a higher efficient, wide range of speed control and operation is at speeds less or more than the rated speed. There are many applications of DC motors, which include computers cooling systems, printers, subway trains, airplanes etc. The extensive usage of DC motors in different applications makes PWM speed control systems strongly needed. Hence the use of FPGA based PWM controls are widely used due to robust and reliability circuit operation for this application.

KEYWORDS: Modelling, PMW, Electric Motor, Energy security, picoblaze system generator

INTRODUCTION

The design system has been fortunately fully automated EDA tools support design synthesis, but there is still a great need for efficient techniques to accelerate the design process and this essentially means choosing the right methodology. The first step in designing a system is to choose the design methodology. For the vast majority of applications it is hard to prefer one implementation to the other: bottom-up or top-down. There is always a certain amount of manipulation involved and evaluating between two correct implementations is usually based on the designer's preferences. One may prefer one method for developing a certain type of application while another person prefers the other method. The key idea of both methodologies is the hierarchical propagation of the design units based on behavioural modelling and optimization at each level. The top-down design approach defines the architecture of the whole design as a single unit.

DESCRIPTION OF THE SOFTWARE

Xilinx ISE system in Electronic Design

The Xilinx ISE system is an integrated design environment that consists of a set of programs creating, simulating and implementing digital designs in a FPGA or CPLD target device and allows taking your design from design entry through Xilinx device programming. The ISE Project Navigator manages and processes the design through several steps in the ISE design flow. These steps are Design Entry, Synthesis, Implementation, Simulation/Verification, and Device Configuration. Creating source files done in different formats such as a schematic, or a Hardware Description Language (HDL) such as VHDL can do this. The synthesis step creates netlist files from the various source files that serve as input to the implementation module logic circuit based on a VHDL program. Last step is the schematic under synthesis in the processes pane brings up a hierarchical schematic description of a synthesized circuit; see the report below on figure 1 shows the design summary generated after the synthesis. The report is divided to five parts:

- Top level module project status describes in general the name of the module, the project file, the errors and warnings, routing results related to all signals routed, timing constraints.
- The second part of the report related to devices utilization in the design as the number of slice flip flop used in the design 142 slices from 9312 available, the number of inputs used and so on.

The designer maps the gate level description or netlist to the target design library and optimizes for speed, area or power consumption. The objective is to provide a tool set for FPGA design .

Picoblaze systems generator

The PicoBlaze design is the designation of a series of three free soft processor cores from Xilinx for use in their FPGA and CPLD products. The PicoBlaze design was originally named KCPSM which stands for Constant(K) code Programmable State Machine. When instatiating a PicoBlaze micontroller in VHDL, the respective KCPSM component name must be used for a PicoBlaze processor. The design of embedded systems on FPGA becomes a source task, reduced to the realization of complete hardware-software complex, which requires due attention. The main difficulty during the creation of the software realized microprocessor systems is a file describing the relationships between the system components. In order to use the process of PicoBlaze based systems design, this work proposes an approach to automatically generate a so called “configuration file”, that contain entire description of relationships between the elements of the system. Usually, the configuration file of the system is created manually, with the given of a initial code that need modifications of the reading of Hello World on display to two ligne one reading the result of the frequency and the second ligne to display the result of the duty cycle. The modification of the code of hello world must be save as hello.psm.

MATERIALS AND METHODS

The VHDL simulation serves as a basic for testing complex designs and validating the design prior to fabrication. As a result, to offer support at all levels of description behavioral, structural. A VHDL program can be considered as a description of a digital system, the associated simulator will use this description to produce behavior that will simulate the system. After generating the synthesis step, the implementation will convert the logic design into a physical file that can be downloading on the target device (FPGA). This involves three sub-steps: Translating the synthesis, Mapping and Place and Route. This refers to the programming of the target by FPGA. The digital design flow regardless of technology is a fully automated process. The design consists of several steps and there is a need for a tool set in each step of the process. There are two major tool sets for simulation: the functional simulation tools and Timing simulation tools. Functional simulators verify the logical behavior of a design based on design entry. The design primitives used in this stage must be characterized completely. The timing simulators on the other hand perform timing verifications at multiple stages of the design. In this simulation the real behaviour of the system is verified when encountering the circuit delays and circuit elements in actual device. In general, the simulation information reflects the actual length of the device interconnects see Figure 1.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

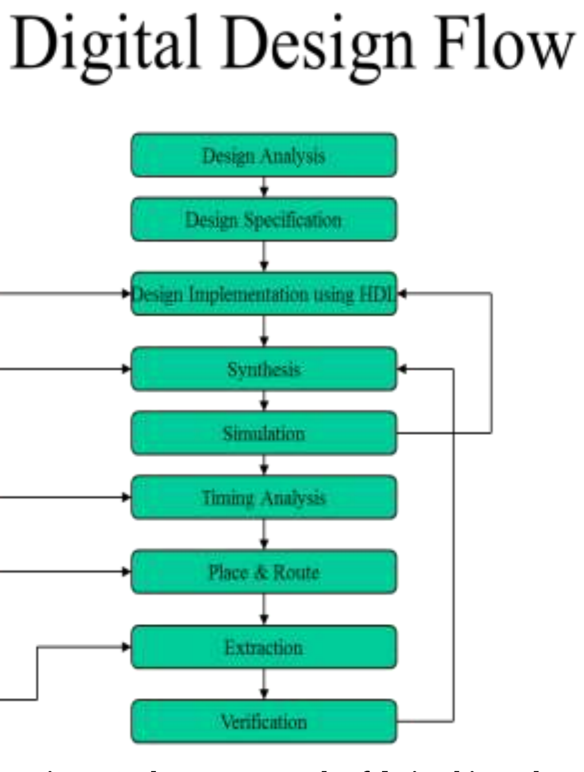


Figure 1: shows an example of design hierarchy

DESCRIPTION OF THE DESIGN SYSTEM AND RESULT

Module to Produce clock divider of 5Hz

The report describes the design and implementation into FPGA of a PWM controller that generates a pulse-width modulated PWM signal to control the speed of DC motor. The design was divided into four stages: the first stage to produce a VHDL code of a clock divider that generates a 5Hz clock signal with one input clock master and one 1-bit output see Figure 2 shows the simulation behavioral of the output of clock divider 5Hz is running high and low continue and Figure 3 proved the value of frequency generated by using oscilloscope connected at the output of the clock.

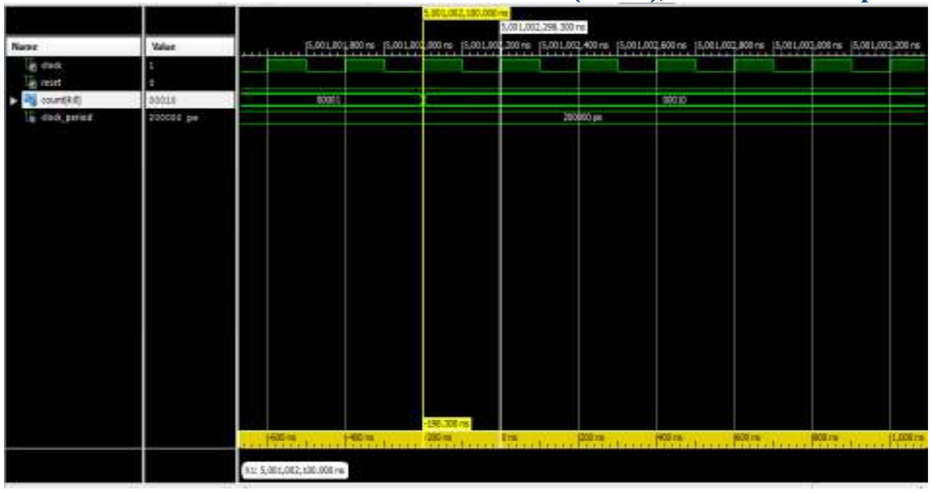
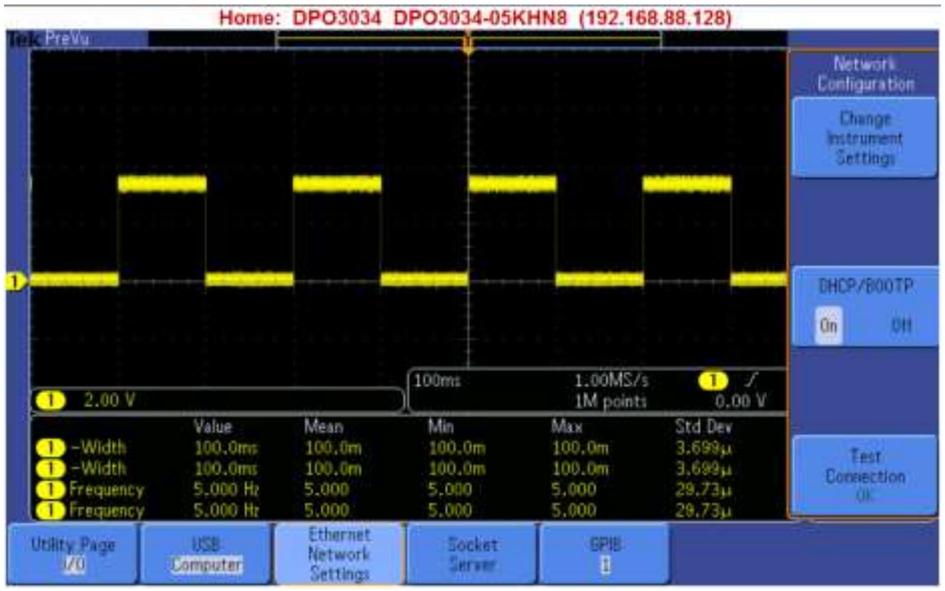


Figure 2: Test bench on Bevavioral model simulation



	Value	Mean	Min	Max	Std Dev
1 -Width	100.0ms	100.0m	100.0m	100.0m	3.699µ
1 -Width	100.0ms	100.0m	100.0m	100.0m	3.699µ
1 Frequency	5.000 Hz	5.000	5.000	5.000	29.73µ
1 Frequency	5.000 Hz	5.000	5.000	5.000	29.73µ

Figure 3: frequency 5Hz check with oscilloscope

Module produce a counter at the frequancy 5 Hz

The second stage is to produce the VHDL code of 5 bit counter with one clock input from the clock divider and 5-bit output driving the leds for counting. The demonstration of the boards shows some leds are one during the counting see Figure 4. The test bench for the behavioral simulation shows the counter pass from the present state 1 to the next state 2 when the clock changes see Figure 4.

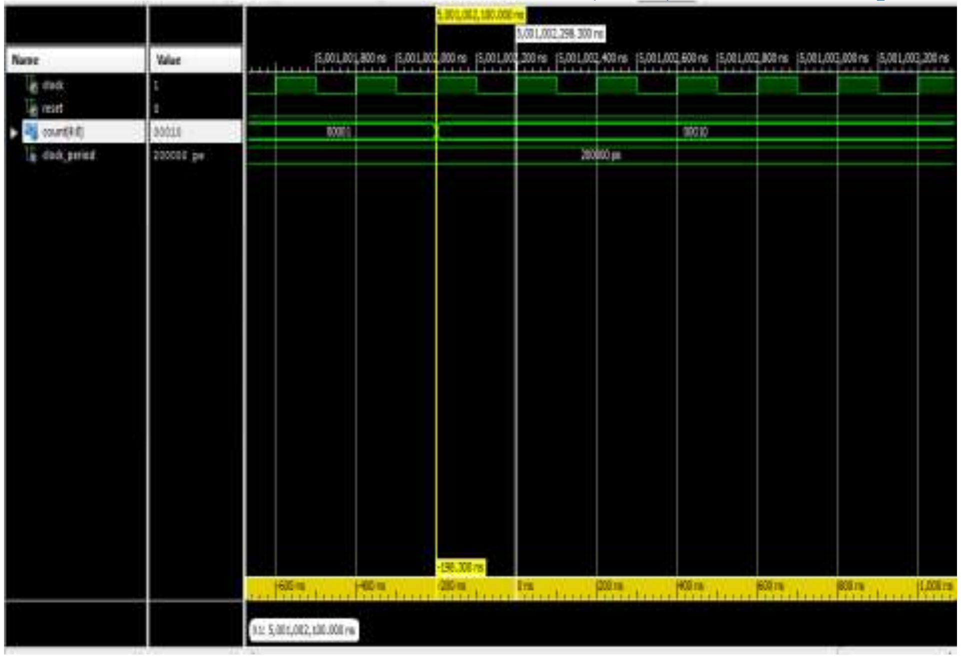


Figure 4: Test bench Behavioural for counter 5-bit

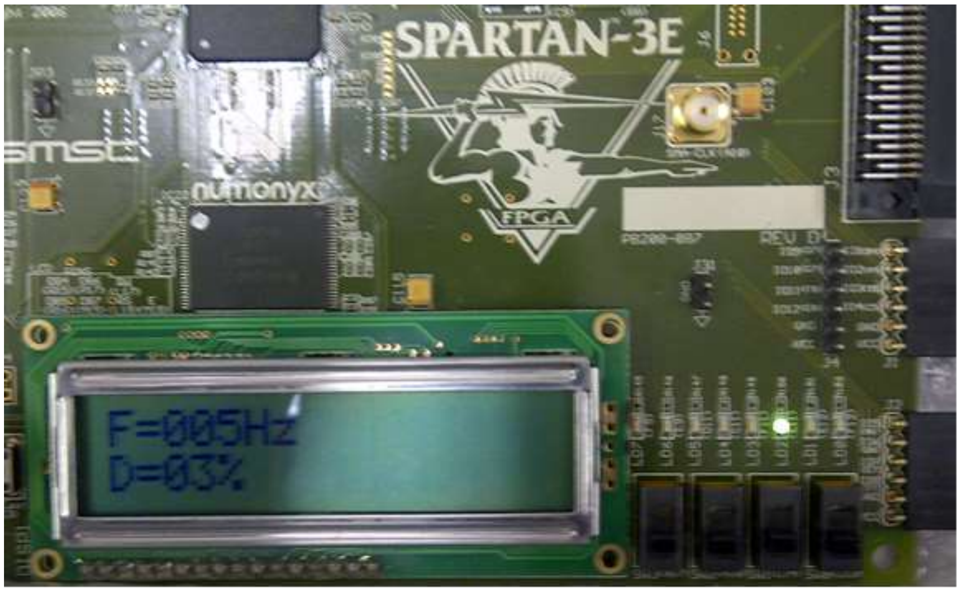


Figure 5: shows the LEDS is running during the counting

Produce a camparator generating a PWM signal

The third stage to produce the VHDL code of a 5-bit comparator with two 5-bits inputs witch have to be compared, and 1-bit output will be the result of the comparison between two inputs one from the counter and the second one will be from the constant treshold at 10000 binary by using this condition:

comp_out <= '1' when (x >= y) else '0';

The results for both comparison will design the PWM signal to drive, and to control the speed of DC motor by generating a pulse-width modulated signal from the output of the comparator,that siganl will be connected to low pass filter to transform that signal to dc voltage. the period of this pulse remains constant and the width of the ducty cyle will change via swicthes.

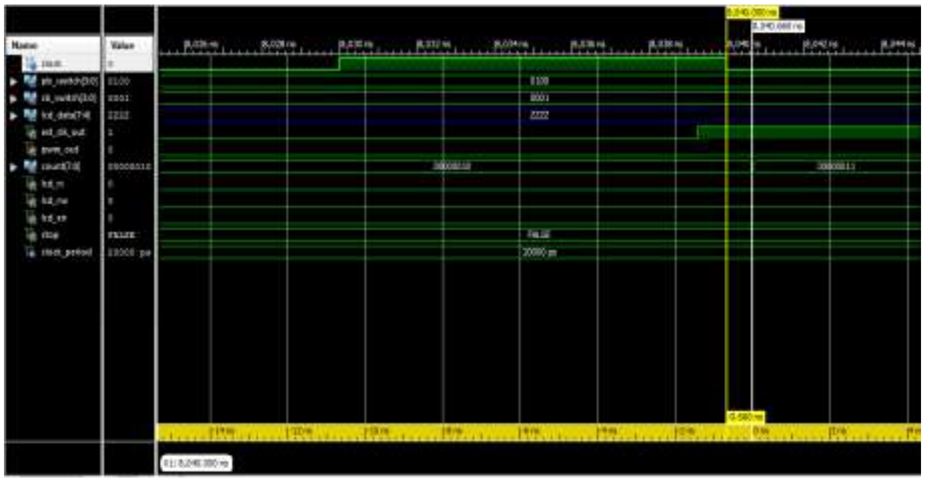


Figure 6: shows the test beach on poste route

Reconfigure the clock divider with different frequancies

The last stage is to reconfigure the clock divider module to support the following output signal frequencies from the push buttons: 5Hz, 5KHz, 500KHz and the rest. As we running from the master clock 50Mz as an input for clock divider and we are looking to generat 5 Hz, so 50Mz/5Hz = 10000000 samples as we will have 50% of the clock will be high and 50% of the clock will be low, that why we need to use half of the number of that samples because 50% will high and 50% will be low that why we need to divide that number by 10000000/2= 500000 as we start from zero we need to take one off that number so the final number will be 4999999 in hexadecimal 4c4b3f, it will be the same calculation for other frequencies 5KHz= 4999 in hexadecimal 001387, for 500KHz=49 in hexadecimal 000031. The code below shows the reconfiguration of the clock divider to support the following output signal set from the push buttons: PB_Switch(3) for reset, pb_switch_ff(0) to control 5Hz, pb_switch_ff(1) for 5KHz and pb_swicth(2) for 500KHz. The system has got a PicoBlaze softcore microcontrller for controlling the LCD display embeded, all data are code in hexadecimal. The fRAM_data describes the values of all frequencies will be display on LCD in hexadecimal. The second data related to duty cycle is reconfigure to support varaiable threshold values set from the slides buttons, by declaring all combinations possible from swiches start from 0000 until 1111. So if we press reset by pushing PB_SWITCH(3) the display shows F = 0 Hz coded in hexadecimal as fRAM_data <= X"2030487A2020", the seconde data will be display on LCD like this D = 0% is coded in hexadecimal as dRAM_data <= X"203025".

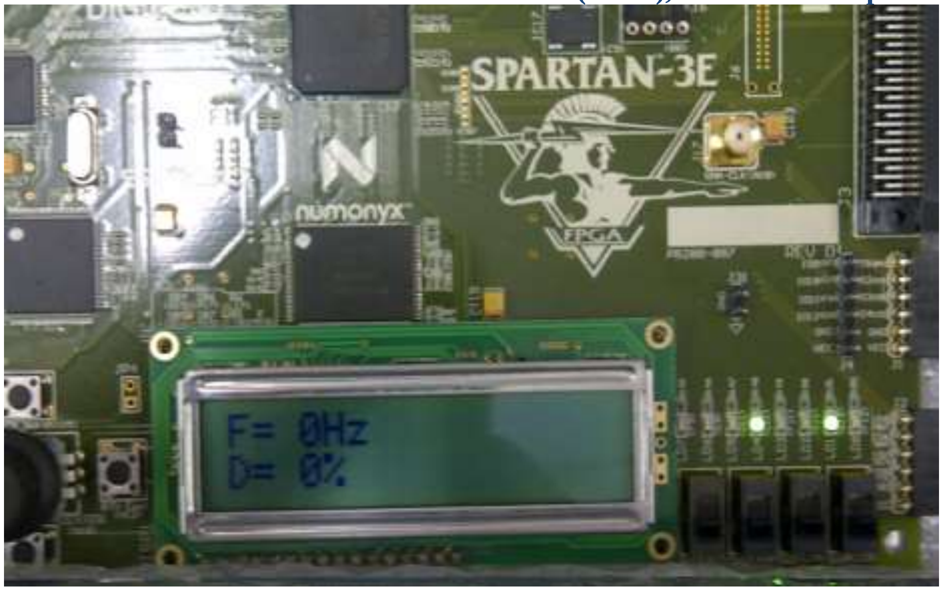


Figure 7: shows the code display on LCD without data

All data are store in microcontroller, generated by FPGA to be display on LCD by selecting the data from push button and swithes button. The VHDL code below shows all frequencies coded in hexadecimal and store in microcontroller as a data to be use by FPGA by using all inputs to select the right data.

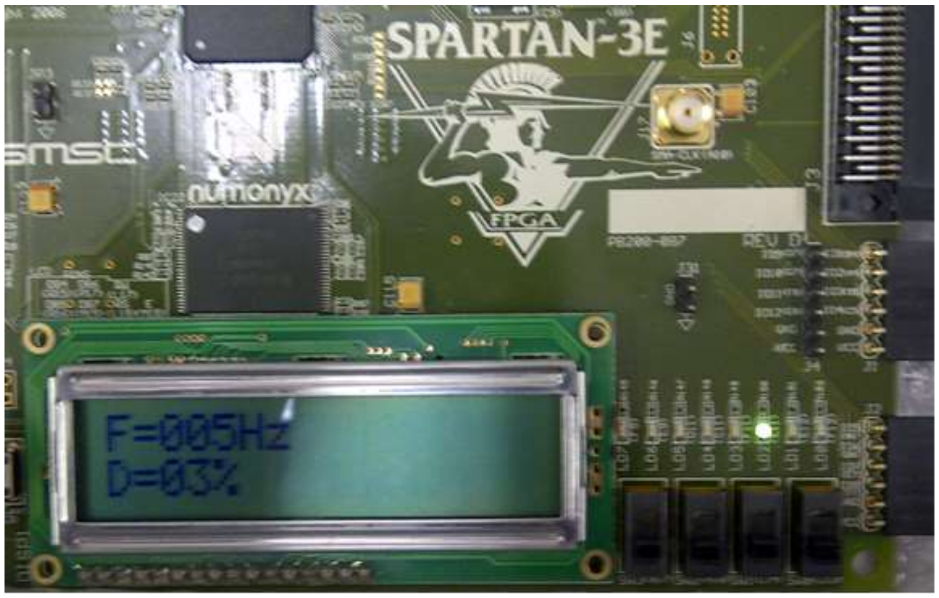
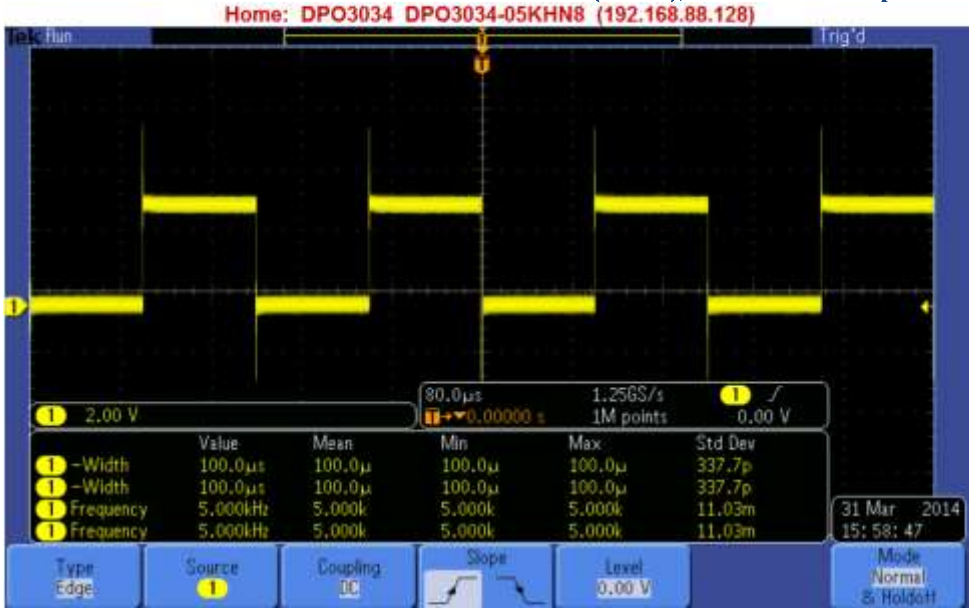


Figure 8: shows the frequency of 5 Hz and the Led for counting

All data was display on LCD, controlled by push buttons and slides button was check and confirms by using oscilloscope connected to the output generation the signal for PWM (Figure 7,8,9,10).



	Value	Mean	Min	Max	Std Dev
1 -Width	100.0µs	100.0µ	100.0µ	100.0µ	337.7p
1 -Width	100.0µs	100.0µ	100.0µ	100.0µ	337.7p
1 Frequency	5.000kHz	5.000k	5.000k	5.000k	11.03m
1 Frequency	5.000kHz	5.000k	5.000k	5.000k	11.03m

Figure 9: frequency of 5KHz

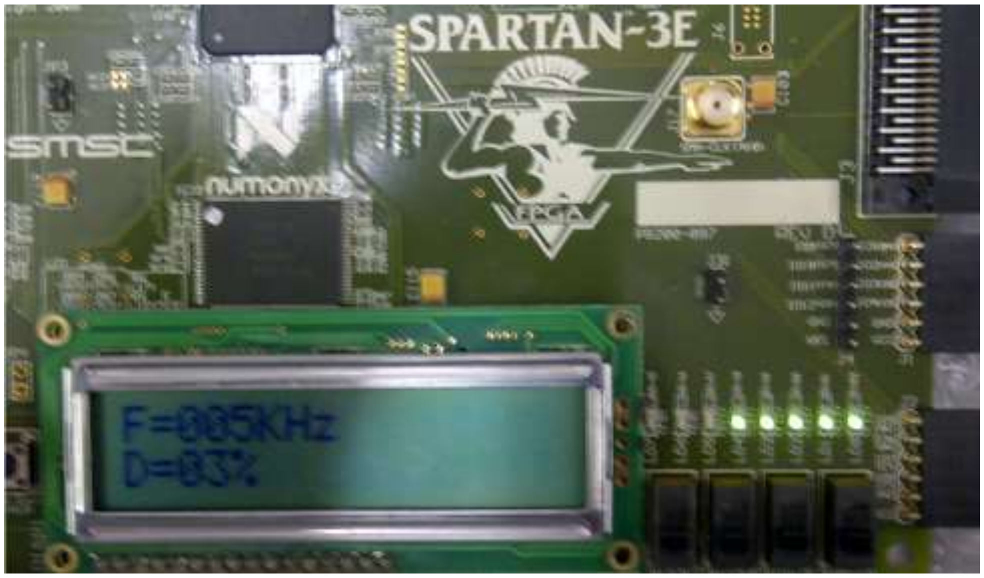
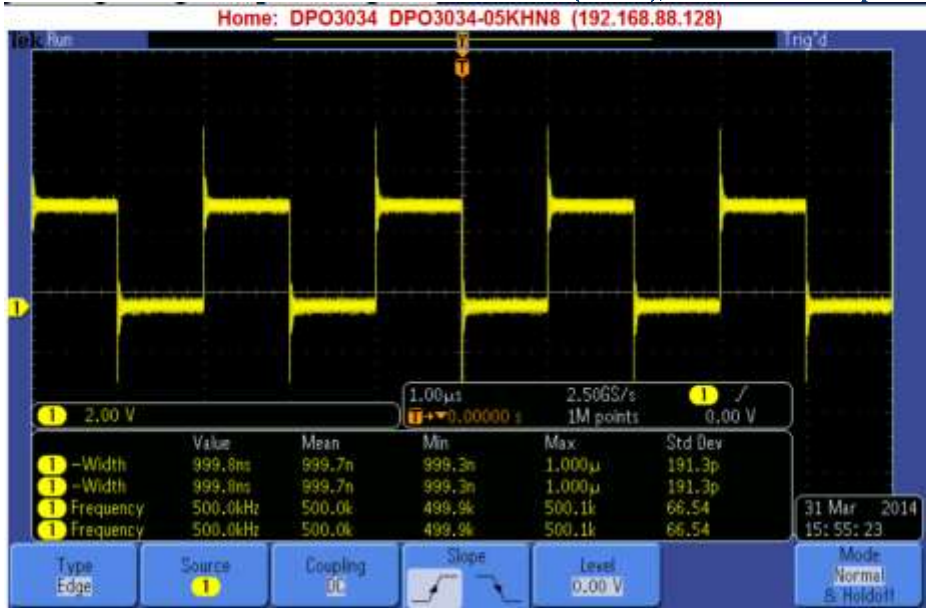


Figure 10: shows the 5 KHz and 3% of duty cycle

The second data related to duty cycle is reconfigure to support varaiable threshold values set from the slides buttons, by declaring all combinations possible from swiches from 0000 until 1111. The data was converted on hexadecidal and stored on FPGA.



	Value	Mean	Min	Max	Std Dev
1 -Width	999.8ns	999.7n	999.3n	1.000µ	191.3p
1 -Width	999.8ns	999.7n	999.3n	1.000µ	191.3p
1 Frequency	500.0kHz	500.0k	499.9k	500.1k	66.54
1 Frequency	500.0kHz	500.0k	499.9k	500.1k	66.54

Figure 11: frequency 500 KHz

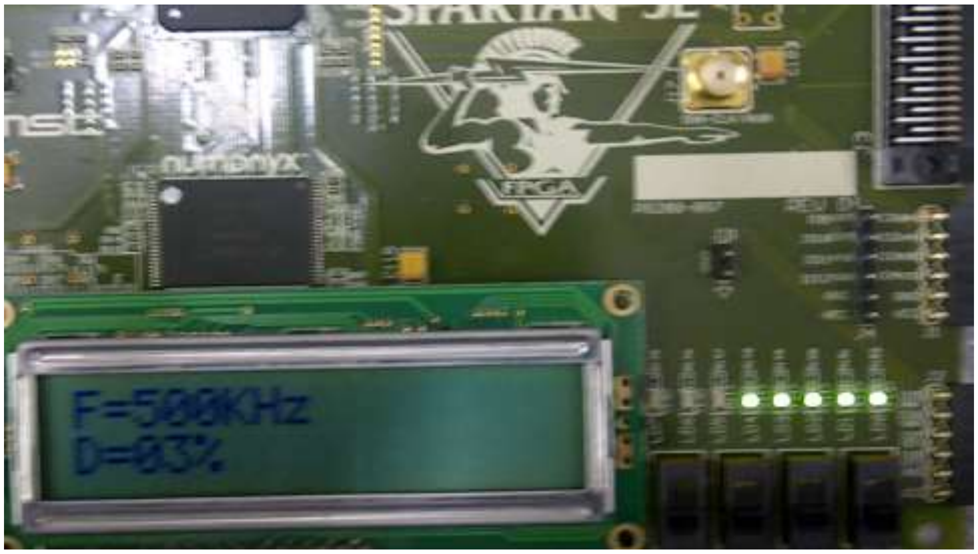
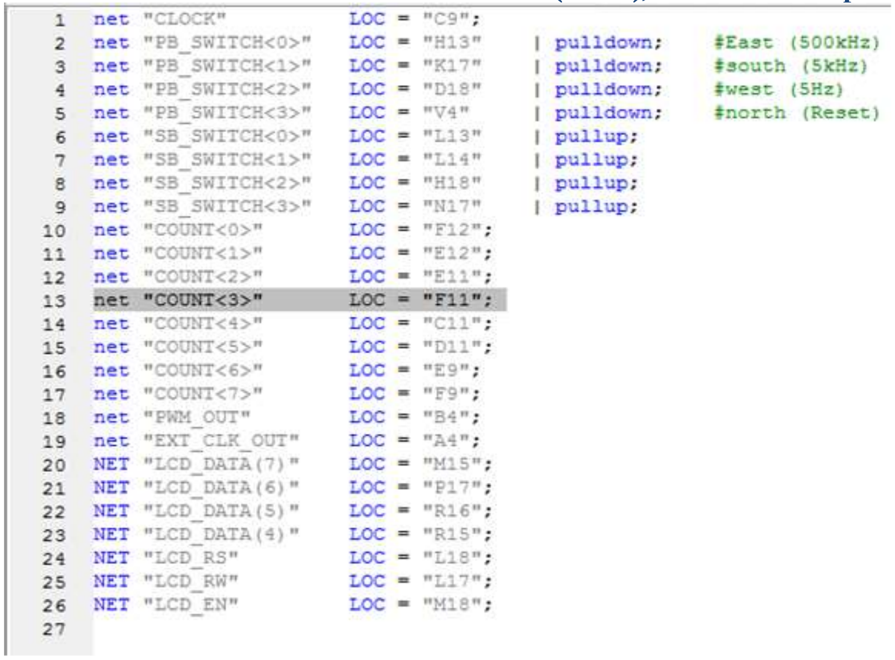


Figure 12: shows the 500 KHz and 3% of duty cycle

Pin assignment and hardware implementation

The arbitrary mapping is described in a separate file, you just have to assign all elements of the vector to the right pins of the FPGA. I used the following mapping, in file name for instance constraints.ucf, Here is the mapping of all signals used in the design system on the pins of the board.


```
net "CLOCK"          LOC = "C9";
net "PB_SWITCH<0>"   LOC = "H13"   | pulldown;    #East (500kHz)
net "PB_SWITCH<1>"   LOC = "K17"   | pulldown;    #south (5kHz)
net "PB_SWITCH<2>"   LOC = "D18"   | pulldown;    #west (5Hz)
net "PB_SWITCH<3>"   LOC = "V4"    | pulldown;    #north (Reset)
net "SB_SWITCH<0>"   LOC = "L13"   | pullup;
net "SB_SWITCH<1>"   LOC = "L14"   | pullup;
net "SB_SWITCH<2>"   LOC = "H18"   | pullup;
net "SB_SWITCH<3>"   LOC = "N17"   | pullup;
net "COUNT<0>"       LOC = "F12";
net "COUNT<1>"       LOC = "E12";
net "COUNT<2>"       LOC = "E11";
net "COUNT<3>"       LOC = "F11";
net "COUNT<4>"       LOC = "C11";
net "COUNT<5>"       LOC = "D11";
net "COUNT<6>"       LOC = "E9";
net "COUNT<7>"       LOC = "F9";
net "PWM_OUT"        LOC = "B4";
net "EXT_CLK_OUT"    LOC = "A4";
NET "LCD_DATA(7)"    LOC = "M15";
NET "LCD_DATA(6)"    LOC = "P17";
NET "LCD_DATA(5)"    LOC = "R16";
NET "LCD_DATA(4)"    LOC = "R15";
NET "LCD_RS"         LOC = "L18";
NET "LCD_RW"         LOC = "L17";
NET "LCD_EN"         LOC = "M18";

```

Figure 13: Pin assignment

CONCLUSION

Softcore system is a one of the significant technology in present days. However the industrial automation application is one of the key issues in developing PicoBlaze. The utilization of softcore technology is novel and might enhance the existed Testing and automation system. With this project we are proposed to optimizing the memory, simultaneously increase the frequency and the duty cycle for accurate results with minimum time .Due to increasing the performance no need to spend much more money for other controlling elements, automatically the cost of the product will be get normal rates. A PicoBlaze based Embedded System was successfully designed and implemented in this project.

REFERENCES

[1] H. C. Roth, L. K. John, Digital System design using VHDL, Cengage Learning, 2008, ISBN 9780495244707.
[2] M.A. Karim, X. Chen, (2008), Digital Design-basic concepts and principles, CRC Press, ISBN: 978-1-4200-6131-4.
[3] M. Cristea, A. Dinu, D. Nicula, (2001), A practical Guide to VHDL Design, ISBN 9733113598.
[4] R. E. Haskell, D. M. Hanna, Digital Design Using Digilent FPGA Boards –VHDL I Active-HDL Editon, LBE Books Rochester Hills, 2009.
[5] S. Yalamanchili, (2001), Introductory VHDL-from simulation to synthesis, prentice Hall, ISBN: 013080982-9.
[6] T. L. Floyd, (2003), Digital Fundation with VHDL, prentice Hall, ISBN: 0-13-099527-4.
[7] Toolbox.xilinx.com,. "Xilinx 1.5I Software Manuals". N.p., 2015. Web. 17 Dec. 2015.
[8] W. Kafig, (2010), VHDL 101: Getting Started, Newnes, ISBN-13: 978185617047