FPGA ile İnfüzyon Pompası için Akış Kontrol Sistemi

Flow Control Sytem for Infusion Pump Using FPGA

Sude Pehlivan1, Büşra Çorak1, Ahmet Ata1

1Biyomedikal Mühendisliği Bölümü, İzmir Katip Çelebi Üniversitesi, İzmir, Türkiye

{sudepehlivan, ahmetata,busracorak}@ikc.edu.tr

*Özetçe*— Bu projede savunma sanayiden medikale birçok kullanım alanına sahip ve tekrar programlanması mümkün olan FPGA kullanılarak infüzyon pompaları için akış kontrol mekanizması tasarlanması amaçlanmıştır.

Anahtar Kelimeler—infüzyon pompası; FPGA; akış sensörü.

*Abstract*— In this project, it is aimed to design flow control mechanism for infusion pumps using FPGA which has many fields of use from defense industry to medicine.

Keywords—infusion pump; FPGA; flow sensor.

# Giriş

Drug delivery has a crucial role in treatments of various diseases and one of the most common delivery routes is infusion which is provided by a device called infusion pump. An infusion pump delivers fluids such as medicine or nutrient to patients with a pre-defined amount by a physician. There are different kinds of systems for infusion pumps; however, the most common one is syringe infusion pumps that have a piston to pump the fluid out of the syringe. The amount of the fluid must be arranged carefully and the rate of the deliver must be monitored in order to eliminate risk of hyper dose or insufficient treatments. However, since hospitals are crowded places and both nurses and physicians are busy most of the time, the control of these systems must be autonomous that requires a microprocessor. The control mechanism can be based on volume or amount of drops [1]. In this project the control mechanism was chosen as volumetric and it was implemented by using a Hall Effect flow sensor (HEFS) and FPGA hardware circuit.

# Materıals

## YF-S201HALL EFFECT FLOW SENSOR

A HEFS is based on creating an output as electrical pulse that is related to amount of rotation of the wheel inside the sensor. The Hall Effect sensors are based on the idea that magnetic field is directly proportional to the produced voltage. When a conductor that has a shape of a rectangle is subjected to a current along its length, a voltage occurs. As a matter of fact, voltage is the production caused by Lorentz force which is the force an electron is subjected to while passing inside a magnetic field. In the case of HEFS, occurred voltage is the measure of a full rotation and since each created pulse represents a constant volume of fluid, when these pulses are counted total flow can be calculated. From the datasheet, when flow is 120 L/H frequency is 16 Hz which means that pulse frequency is 7.5 times the flow rate and if we multiply frequency with 60 minutes and divide by 7.5, we can get the flow rate information [2],[3].



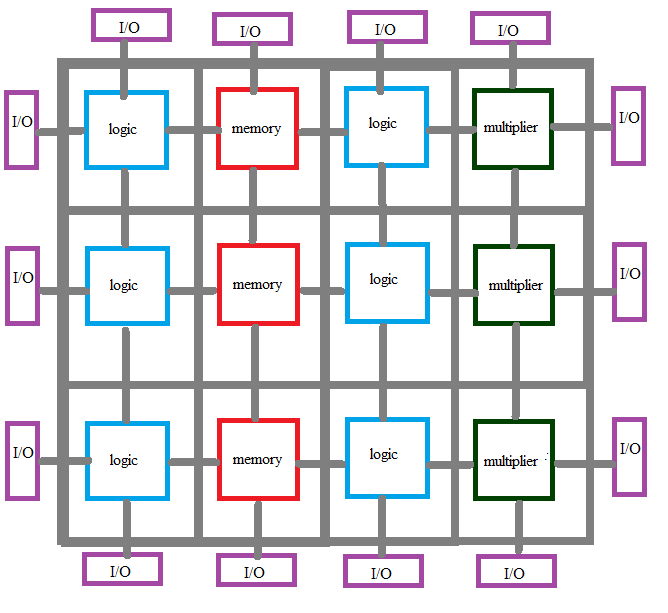
**Figure 1.** Hall Effect Flow Sensor

## FPGA

FPGA stands for field programmable gate arrays and it is an integrated circuit that can be programmed electrically [4]. There are several logic gates which makes it different than other kinds of boards because these logic gates are re programmable by the user. FPGA is slower, needs more area and consumes more power than a standard cell. However, these disadvantages are not that crucial when they are compared with the advantage of providing fast and economic implementation of Moore’s Law [5].

It is important to understand the difference between a FPGA and a microcontroller. The chip of the microcontroller is programmed and can’t be changed. On the other side, FPGA is a blank device that allows the user to implement any kind of program over and over again which is basically being able to design several ASICs in a cost and time effective way [4]. Also, one of the most compelling advantages of the FPGA is that it runs multiple different lines of codes at the same time which provides ability to perform several tasks simultaneously.

In a FPGA, usually static RAM (Random access memory) configuration is implemented which provides re-configurability and speed. FPGA consists of several arrays of logic cells that are connected by a structure both connects neighbor cells and busses. Outer side of this structure there are input/output (I/O) peripherals which are connected to busses. I/O peripherals receive information from outside world and provide information to it. Logic cells include combinational logic circuits, multiplexers, FF (flip flop) registers and feedback lines [6].



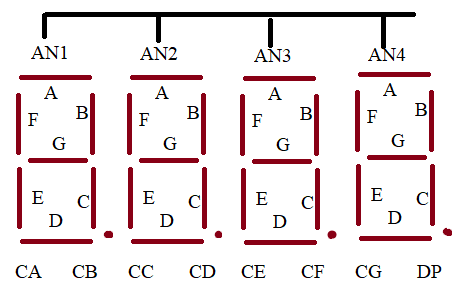
**Figure 2.** Basic Structure of a FPGA [4]

In this project a BASYS 2 FPGA board was used. This board is powered by An USB cable that is connected to a computer. There is also a battery connector for external power supply and a power switch to turn on and off the FPGA [7].

In order to perform a task, fist a ‘bit’ file must be created and then transferred into FPGA. Xilinx ISE/WebPack is used to create the file and Digilent Adept is used to transfer it. After connecting FPGA to computer, Adept must be opened and device must be selected. Afterward, file is browsed and FPGA is programmed.

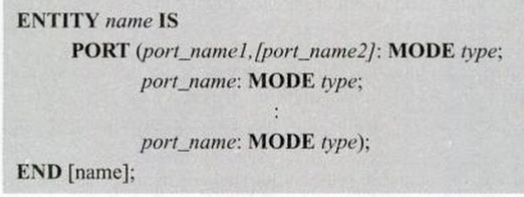
This model uses 25, 50 and 100 megahertz (MHz) oscillators as the internal clock and there are eight switches, eight leds, four buttons and four digit seven segment display [7].

There are eight leds in every seven segment display and also four additional leds to illustrate dots. This means there are 128 outputs and 10 of them corresponds to 0-9 decimal digits. The display is common anode which means all leds of every seven segment is connected together and activated when low. There are four enables for each display that are connected and desired segment can be activated by giving ‘0’ into related one and ‘1’ to others. When multiple segments are used, clock must be arranged such that the time of the appearance of each digit cannot be distinguished by the human eye [7].



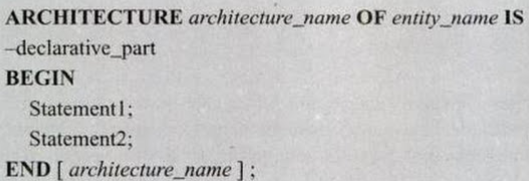
**Figure 3.** Basic Structure of Seven Segment Display [7]

In order to program a FPGA, there are two kinds of ways as schematic or HDL (Hardware Description Language) with two options VHDL and Verilog. VHDL stands for very high speed integrated circuit hardware language and it is used to identify the hardware structure of digital circuits and in this case a FPGA. In VHDL, files are saved as ‘vhd’ or ‘ucf’. In vhd files there are two main parts as ‘entity ‘and ‘architecture’. Entity is used to identify the external interface which means that the inputs and outputs of the system are determined in this stage that can be vectors as well. General structure of the entity is given in Figure 4. First the name of the entity and port the external signals to the external world. Mode can be input or output and type can be std\_logic, std\_logic\_vector, integer and more.



**Figure 4.** Basic Structure of ENTITY

There are two parts of the architecture part as declarative and statement. As shown in Figure 5, name of the architecture is defined by user and it is pointed that this architecture will use the external signals that are defined inside a specific entity. Afterwards, general task of the system is defined.



**Figure 5.** Basic Structure of ARCHITECTURE

It is important to understand some data classes in order to implement the design. Data class can be a ‘constant’ which means the value cannot be modified, ‘variable’ which means that it can take several different values or ‘signal’ which means that it is used as an internal wire that is not in relation with the external environment. Data types are also important and they can be ‘pre-defined’ such as bit\_vector, ‘scalar’ such as integer, ‘composite’ such as array or ‘access’. Operations in VHDL have five types as ‘arithmetic’ such as addition, ‘relational’ such as greater, ‘shift’ such as shl, ‘logical’ such as xor, and ‘miscellaneous’ such as abs. [8]

# METHODS

In this section, written VHDL code is explained.

1. *Seven-segment driver VHDL Module*

This module is written for 4 digit number display in the seven-segments. Firstly, we defined a segment decoder submodule. Then, the clock divider was defined.

### Segment-decoder Module

In this section, codes determine to display the numbers on segments of the Basys2 FPGA board. Four-bit digits were defined for each segment as shown in figure 4.

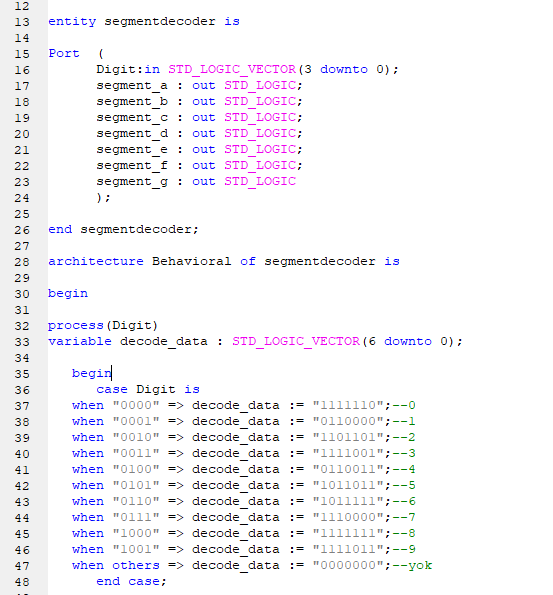


Figure 4. Segment-decoder Module

### Clock Divider Module

In this section, 50 MHz clock of the board was decreased to 3 Hz clock as shown in figure 5. This clock dividing method provides a proper transition between seven-segment displays.

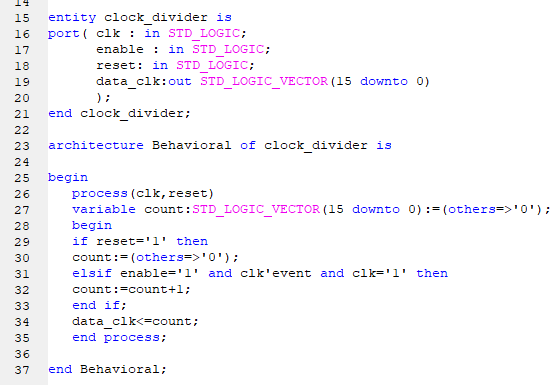


Figure 5. Clock Divider Module

### Top Design of Seven-segment Driver Module

Until top design module, codes displayed the same number on each seven-segment display. In the top design module of seven-segment displays, modulation of codes provides different numbers on seven-segment displays. This part of code is shown in figure 6.

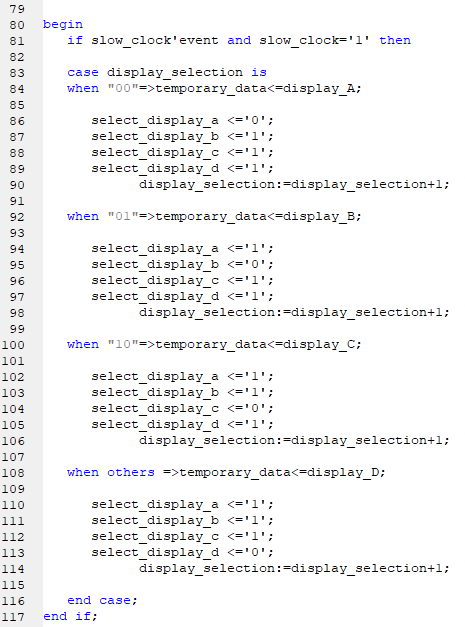


Figure 6. Top Design of Seven-segment Driver Module

1. *Flow Sensor VHDL Module*

We wrote the code for providing a connection between flow sensor and FPGA. This flow sensor’s (YF-S201) output is the number of turns of the propeller which drives the water flowing through it. The effect within the number of propeller revolutions is measured by magnetic sensors.

1. *Counter Pulse Module*

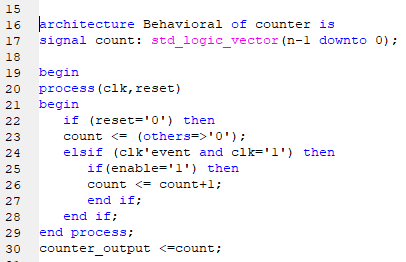


Figure 7. Counter Pulse Module

This module was prepared for taking pulses which comes from the flow sensor. These pulses were collected in 9 bit counter as shown in figure 7.

1. *Flow Rate Calculation Module*

This module was written for flow rate calculation. Flow sensor’s output is calculated as shown in equation 1. Calculation code is shown in figure 8.

(1)

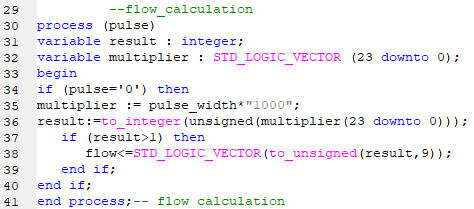


Figure 8. Flow Rate Calculation Module

1. *BCD Converter Module*

Obtained flow rate calculation’s output was assigned to units, tens and hundreds of values for displaying numbers in the seven-segment display as shown in figure 9.

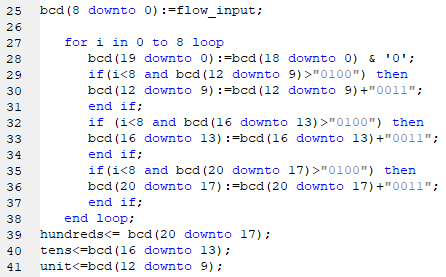


Figure 9. BCD Converter Module

1. *Top Design of Flow Sensor Module*

This module is designed for the combination of flow sensor submodules. Essential assignments were done.

1. *Top Design VHDL Module*

Communication between the top design of the seven segment display module and top design of a flow sensor module arranged. All ports are determined and assigned to the FPGA board by using implementation constraints file (UCF).

# RESULTS AND DISCUSSION

In this project, the seven-segment module worked properly. In clock dividing process we divided 50 MHz FPGA clock to 216 to obtain 3 Hz clock. Because by using only 50 MHz clocks, lines of the seven-segment display are overlapped.

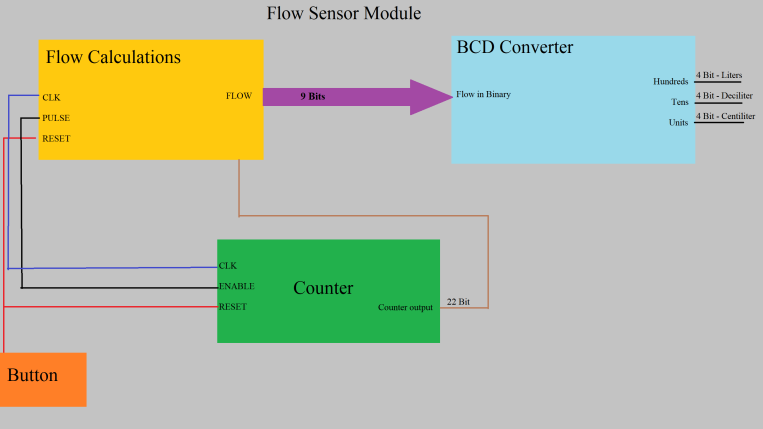


Figure 10. Block diagram of Flow sensor module

We designed figure 10 as a template for the flow sensor module. In this module, we couldn’t assign a button as a reset, because of that when the flow was terminated seven segment display is not reset. That means it cannot be zero after the calculation is done. We couldn’t write an appropriate code for a reset when there are no coming impulses. Firstly, we tried to reset without the button but it didn’t work, then we tried to use the button as an exterior input. But it did not work either

In the flow calculation module, we cannot sure about the result. We see value changes in FPGA very fast. We estimate that the cause of it could be counter and clock didn’t match.

BCD converter worked properly because whatever the output of the sensor is, we see 3 digit number in the seven segment displays. Therefore, the digital output of the sensor becomes decimal numbers.

As a result, our project worked even though its lack of properties. For the proper working of this project, new VHDL modules can be added. While we were doing research about the FPGA and sensors, we discovered a trigger module which is used in some other types of sensors. Maybe we also need this module in this project.

##### Kaynakça

1. Assunção, R., Barbosa, P., Ruge, R., Guimarães, P. S., Alves, J., Silva, I., & Marques, M. A. (2014, February). Developing the control system of a syringe infusion pump. In 2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV) (pp. 254-255). IEEE.
2. Lenz, J. E. (1990). A review of magnetic sensors. *Proceedings of the IEEE*, *78*(6), 973-989.
3. S. Şahin, Y. İşler, C. Güzeliş, “Chapter 19: A Real Time Simulation Platform for Controller Design, Test and Redesing,” *Real-time Simulation Technologies: Principles, Methodologies, and Applications* içinde kitap bölümü, ISBN: 978-1439846650, CRC Press, 461-499, 2012
4. Kuon, I., Tessier, R., & Rose, J. (2008). FPGA architecture: Survey and challenges. Foundations and Trends® in Electronic Design Automation, 2(2), 135-253.
5. Kuon, I., & Rose, J. (2007). Measuring the gap between FPGAs and ASICs. IEEE Transactions on computer-aided design of integrated circuits and systems, 26(2), 203-215..
6. Furtek, F. C., Mason, M. T., & Luking, R. B. (2001). U.S. Patent No. 6,292,021. Washington, DC: U.S. Patent and Trademark Office.
7. Basys, D. (2). Board–Reference Manual.
8. Kaur, G. (2011). VHDL: Basics to programming. Pearson Education India.