FPGA IMPLEMENTATION OF ZIEGLER-NICHOLS CLOSED-LOOP METHOD FOR AUTOMATIC PID PARAMETERS TUNING

Nurotul Auliya'[1], Nanang Sulistiyanto, Ir, M.T.[2], and Mohamad Hairol Bin Jabbar, Dr[3]

Abstract – **A control loop is necessary in order to control a plant or system in order to gain low error system, robust system, or system with fast response depend on the purpose. The most commonly known and used control loop is Proportional-Integral/Proportional-Integral-Derivative. In order to gain the desired output, its parameters, which have different effects, have to be set according to the design requirements. Several methods can be used to determine the parameter; one of them is Ziegler-Nichols closed-loop method. The purpose of this project is to carry out FPGA implementation of Ziegler-Nichols closed-loop method for automatic PID parameters tuning. The commonly used design hardware for digital projects is microcontroller. Microcontroller device resources is limited, we do not know how much device resources this project will take, and to add an additional resources is quite complicated as well, therefore we choose FPGA instead. This project is part of a bigger project which consists of three projects, which are handled by a student each. The most important parts for this project are estimator and controller modules which are located in the FPGA. This is because the estimator's function is to do the steps of the Ziegler-Nichols closed loop method and the controller is necessary because the estimator cannot function if there is no controller. To build and test out the system, it is necessary to begin from the subsystems. If the subsystem's tests are successful, then the probability for the overall system to be success is higher. Experimental results show that the subsystems have been successfully designed, but the overall system could not be applied because the target Spartan 3E FPGA does not have sufficient logic resources on. The first and second objectives was achieved but the third objective was not achieved because this project could not be applied on the target FPGA and therefore this project has not been used on the real tools.**

Keywords – **Auto-tuning PID controller, Ziegler-Nichols, FPGA Implementation**

I. INTRODUCTION

Direct current (DC) motors are the most commonly used in industry. It is used for so many things, starting from tool machines in general, piston pumps, grinding machines, textile machines, paper machines, traction vehicles, steel plants, rolling mill motors, mine hoists, mixers, extruders, ship propulsion, and many others [1] [2].

In order to use DC motors efficiently, a control loop is needed. The most commonly known and used control loop is PI (Proportional-Integral) and PID (Proportional-Integral-Derivative) with percentage of more than 95% [3] [4] [5]. This control loop is also pretty basic so it can be combined with others to build automatic systems [3] [4] [6].

So that the output match with what we want, the controller's parameters have to be set according to the requirements. Each parameter has certain effects when they are changed, so it's quite hard to choose the exact parameters. Thankfully, there are methods to determine the parameter, one of them is Ziegler-Nichols closed-loop method [7] [4]. While the process to search the parameters is quite harsh, this method can provide the commonly smaller Integral of the Absolute value of Error (IAE) and even smallest IAE for third order systems or above compared to other methods [7].

Normally, Ziegler-Nichols closed loop is done manually, but to do it manually, we have to see the response which is displayed at monitor. We also have to manually change the Kp parameter. We have to focus on it, but if we use an automatic system, all the process can be done with only one push. We do not have to see, change the Kp parameter, and we can also do something else during the process. [3]

In order to make Ziegler-Nichols closed loop method done automatically, a tool is necessary as the controller and the estimator for automatization process. The commonly used design hardware is microcontroller, but its resources is limited, we do not know how much device resources this project will take, and to add an additional resources is quite complicated as well, so we choose FPGA instead.

FPGA nowadays has plenty of memory. It can also display the output on monitor thanks to the attached VGA port so we can see how the output turns out not from the numbers in every period of time which is recorded and then converted into graphic which later can be displayed on monitor, but direct from the FPGA to the monitor [8].

The writer, basing on the background, tried to do this project. Hopefully this project will help those who use PID controllers.

II. LITERATURE REVIEW

2.1 Ziegler-Nichols Closed Loop Method

Ziegler-Nichols closed loop method refers to the method by Ziegler-Nichols which is applied on a closed loop system. The basic of closed loop system is as shown in Figure 2-1.

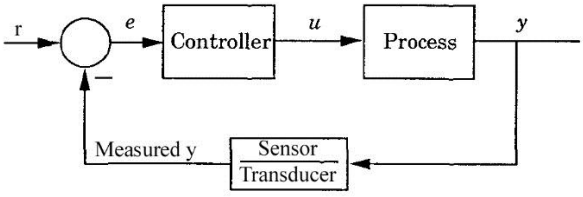

Figure 2-1 *Closed loop [4]*

[1] Nurotul Auliya' is a college student of Teknik Elektro of Brawijaya University in Malang, Indonesia (telephone number: +6285234025426; email: indigo0shadow@gmail.com)

[2] Nanang Sulistiyanto, Ir, M.T. is a lecturer of Teknik Elektro of Brawijaya University in Malang, Indonesia.

[3] Mohamad Hairol Bin Jabbar, Dr. is a lecturer of FKEE of UTHM, Malaysia.

The main procedure of Ziegler-Nichols' closed loop method is to set PID parameter to be P controller. At first, Kp is set to be zero then is increased until output signal of *measured y* become oscillated. The Kp is then recorded as Ku (ultimate gain) and the period of oscillation is recorded as Tu (ultimate period). The last step is to determine parameter's value by using Table 2-1.

Table 2-1 Formulas for controller parameters in Ziegler-Nichols closed loop method *[9]*

	K_p	T_i	T_d
P controller	$0.5K_u$	∞	0
PI controller	$0.45K_u$	$\frac{T_u}{1,2}$	0
PID controller	$0.6K_u$	$\frac{T_u}{2}$	$\frac{T_u}{8} = \frac{T_i}{4}$

The setpoint (*r*) used as excitation can be in the form of step signal. The value of this signal has to be small, so the process is not driven too far away from the operating point where the dynamic properties of the process may be different and must not be too small, or it may be difficult to observe the oscillation due to inevitable measurement noise. It is also important that the Ku is found without the control signal being driven to any saturation limit during the oscillation.

2.2 PID Controller

PID algorithm [4] can be written as equation (2-1):

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \quad (2\text{-}1)$$

Where:

e = error
u = control action/control variable
K = gain
Ti = integral time
Td = derivative time
t = time

Control variable is the accumulation of three terms: P-term (proportional toward error), I-term (proportional toward integral error), and D-term (proportional toward derivative error) as pictured in Figure 2-2. Controller parameters are proportional gain K, integral time Ti, and derivative time Td.

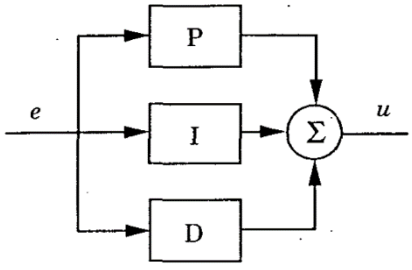

Figure 2-2 *PID algorithm [4]*

In order to apply PID on FPGA, we need to simplify the equation, since integral and derivative is hard to be applied right away, we need to break up the PID equation into equations for P-part, I-part, and D-part.

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \quad (2\text{-}2)$$

$$u(t) = u_P(t) + u_i(t) + u_d(t) \quad (2\text{-}3)$$

Where:

$$u_p(t) = K_p.e(t) \quad , K_P = K \quad (2\text{-}4)$$

$$u_i(t) = K_i\int_0^t e(\tau)\,d\tau \quad , K_i = K : T_i \quad (2\text{-}5)$$

$$u_d(t) = K_d\frac{de(t)}{dt} \quad , K_d = K.T_d \quad (2\text{-}6)$$

Integral of a function can be represented as an area below the line that the function represents. Therefore it can be substituted by accumulation of small squares under the line, which will turn the equation into equation (2-7).

$$u_i(t) = K_i.\left(\sum_{\tau=0}^{t} e(\tau).\Delta\tau\right)$$

$$u_i(t) = K_i.\Delta\tau\left(\sum_{\tau=0}^{t} e(\tau)\right) \quad (2\text{-}7)$$

Derivative can be written in equation (2-8). With the value of t is kept as close to zero as possible.

$$u_d(t) = K_d.\frac{e(n+t)-e(n)}{t} \quad (2\text{-}8)$$

Since $e(t) = r(t) - y(t)$, and the value of $r(t)$ for Ziegler-Nichols closed loop method is constant, we can change equation (2-8) into equation (2-9). By doing this, we can also decrease

$$u_d(t) = K_d.\frac{\left(r(n+t)-y(n+t)\right)-\left(r(n)-y(n)\right)}{t}, \; r(n+t) = r(n)$$

$$u_d(t) = K_d.\frac{\left(-y(n+t)\right)-\left(-y(n)\right)}{t}$$

$$u_d(t) = K_d.\frac{-y(n+t)+y(n)}{t}$$

$$u_d(t) = \frac{K_d}{t}\left(y(n) - y(n+t)\right) \quad (2\text{-}9)$$

2.3 FPGA (Field Programmable Gate Array)

According to Farooq [10], FPGAs are pre-fabricated silicon devices that can be electrically programmed in the field to become almost any kind of digital circuit or system. Normally FPGAs comprise programmable logic blocks which implement logic functions, programmable routing that connects these logic functions, and I/O blocks that are connected to logic blocks through routing interconnect and that make off-chip connections.

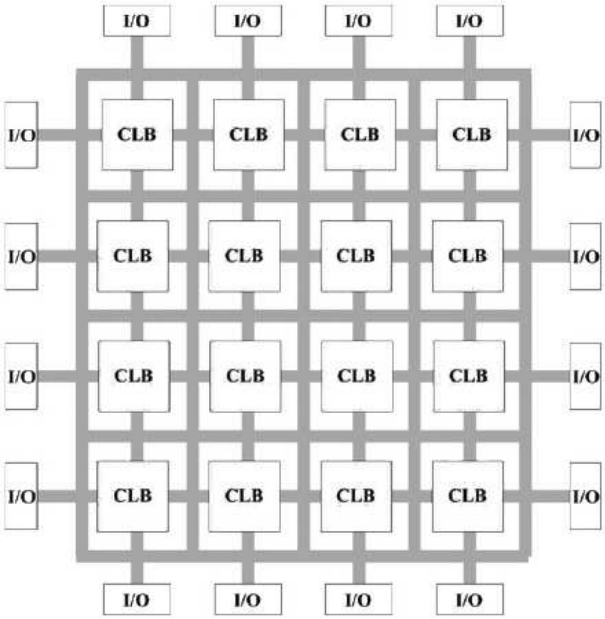

Figure 2-3 *Overview of FPGA architecture [10]*

Figure 2-3 shows a generalized example of a FPGA. I/O represent input/output while CLB is abbreviation of Configurable Logic Blocks. The CLBs and the I/O blocks are interconnected by programmable routing resources.

2.4 *IAE (Integral of the Absolute value of Error)*

After designing and testing a machine, evaluation of the performance is necessary. To do this evaluation, a performance index is necessary. One of the performance indexes that can be used which considers the entire closed loop response is IAE. The equation for Integral of the Absolute value of Error (IAE) is [7]:

$$\text{IAE} = \int_0^\infty |e(t)|.\,dt \qquad (2\text{-}10)$$

The performance index is adequately used to represent the important system specifications instead of a set of indices. The transfer function of a system is considered as an optimal form when the system parameters are adjusted so that the performance index reaches an extreme value. IAE is one of the well-known integral performance indices. [11]

III. METHODOLOGY

3.1 *System Design*

3.1.1 System architecture

Diagram block

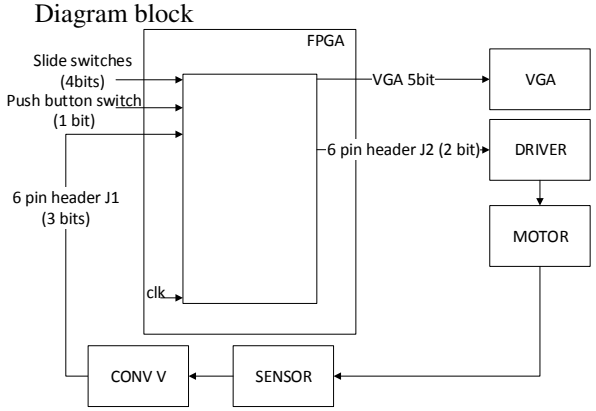

Figure III-1 *Block diagram of overall system*

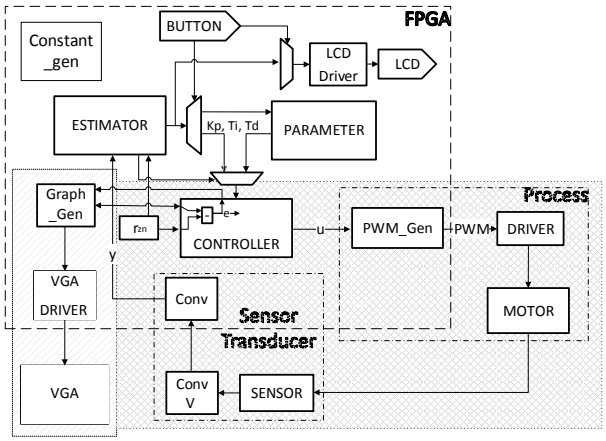

Figure III-2 *Detailed block diagram of overall system*

Where:

r_{zn} = setpoint to apply condition for Ziegler-Nichols closed loop method

u = control action

y = output process or response. In this case, y is already measured

PWM = PWM signal

work belong to other project

work belong to joint project

As shown in Figure III-1 and Figure III-2, the most important part is estimator, which will be in the FPGA.

The necessary ports of the FPGA are one push button, slide switches, four pins (three as input and one as output), VGA port, and LCD. The push button is needed to reset the values and process. The slide switch is used to decide which value will be displayed on the LCD (*Kp*, *Ku*, *Tu*, *Ti*, *Td*), and to decide estimator's mode (idle or searching process). The three pins are used as inputs from the rotary sensor, which number depends on the rotary sensor's output, while the other one is output to driver in PWM signal. The VGA port is used to connect FPGA to VGA so that the graphic can be displayed on the VGA. The LCD is used to display the value of the searched variables.

The driver is used to pull up the FPGA's output voltage to match motor's maximum input. DC motor in this case is acting as plant. Sensor, which is a rotary sensor, is used to gauge the speed of the DC motor, depend on the output value, it might need to be pulled up or pulled down using Conv V device.

3.1.2 Subsystems

3.1.2.1 *Controller*

As explained in the system architecture, this controller module has been designed by other student in other project. Since this project needs controller, which parameters are to be searched, we also use it in this project.

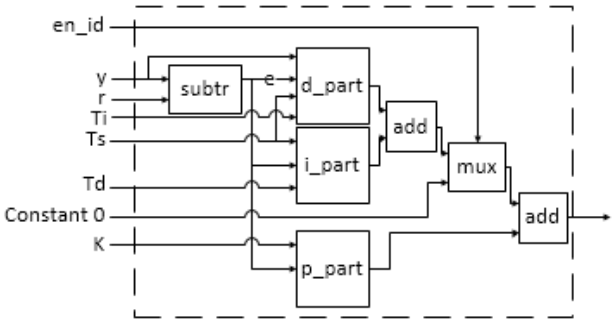

Figure 3-3 *Block diagram for controller*

In Figure 3-3, *r* represent the setpoint. *Ts* represent t in equation (2-9) and $\Delta\tau$ in equation (2-7). d_part, i_part and p_part are using equations (2-9), (2-7), and (2-4) respectively. For whether the controller becomes P controller or PID controller, it depends on *en_id* value.

3.1.2.2 *Estimator*

The function of estimator module is to estimate the parameters of a PID controller which control a process using Ziegler-Nichols closed loop method.

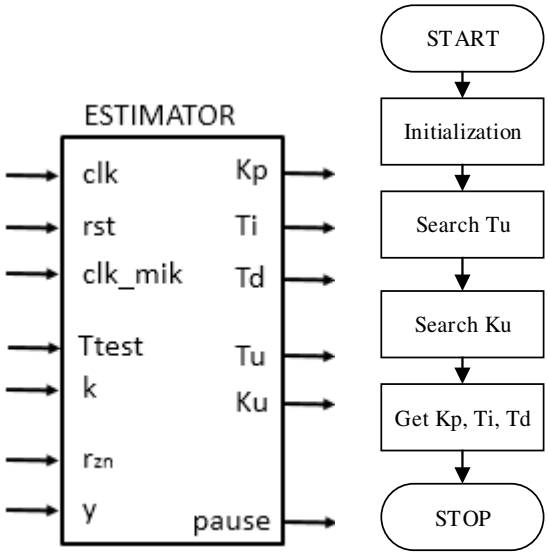

Figure 3-4 *Estimator: (left) module (right) flowchart*

The estimator module shown in Figure 3-4 will mostly use counter and comparation. The *pause* output will be used to control multiplexer so that the parameters value will be saved in the parameter and it will enable the i_part and d_part of the controller.

The flowchart for the program in Figure 3-4 represent Ziegler-Nichols closed loop process, which initial process is to search *Ku* and *Tu*, while using the P controller. Next, it will get the value of *Kp*, *Ti*, and *Td*. In other words, this module can act to estimate the necessary PID parameter for the process.

3.2 Test Method

The testing method is first using simulation. After using simulation, the system is then applied on the FPGA and is tested.

After the subsystem tests are considered successful, we can try assembling the subsystems into the system. The system will be tested using Figure III-5, with transfer function of 1 : 1 for the Process and Sensor module.

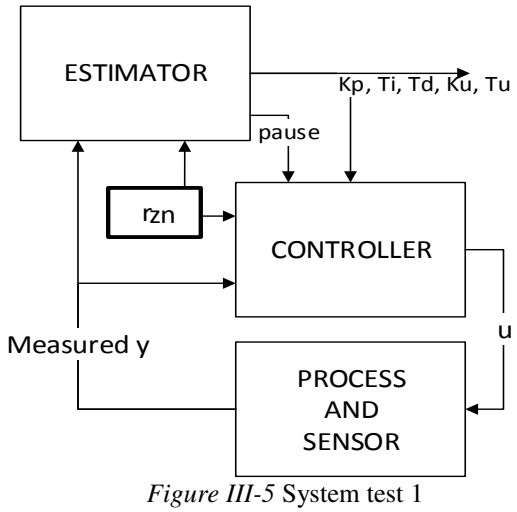

Figure III-5 System test 1

IV. EXPERIMENTATION, RESULTS, AND DISCUSSION

4.1 Subsystem simulation using ISE

In the simulator, we need to set the constant and the clock. For the clock '*clk*', we set the period = 20,000, which means 20,000 ps (20 ns) with 50% duty cycle. For clock '*clk_mik*', we will set the period = 1,000,000, which means 1,000,000 ps (1 µs) with 50% duty cycle. *rst* variable is set to be constant with value of '1' during time 0 ps to 2 ps.

4.1.1 Estimator

4.1.1.1 *Signal* y *stable, where* y ≠ r

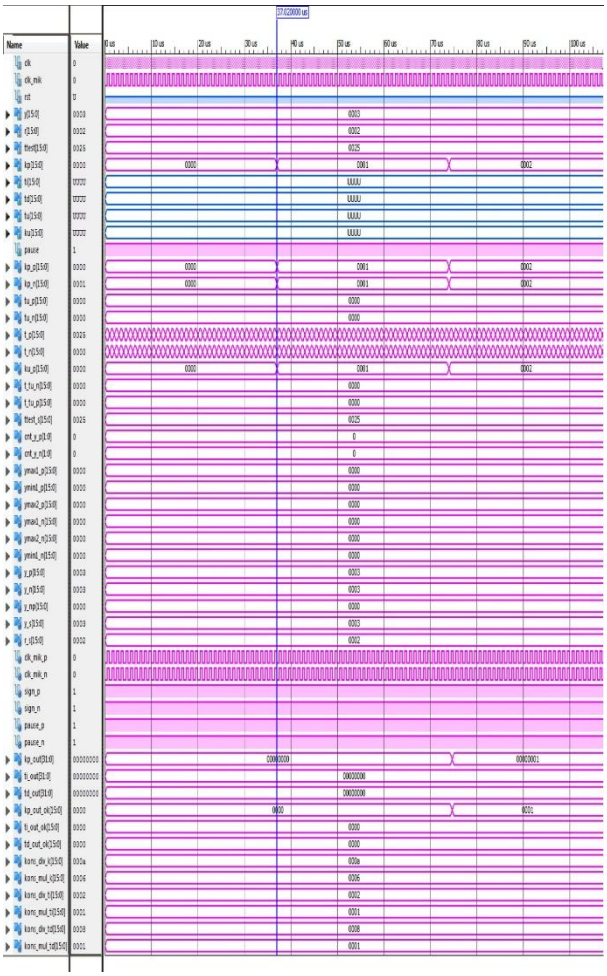

Figure IV-1 *Simulation waveform for Estimator logic block, y ≠ r*

As we can see in Figure IV-1, the output *Kp* firstly change depend on *Kp_p*, that is after the time reaches 37.04 µs, while *Ti*, *Td*, *Tu*, and *Ku* wait for "*(ymax1_p - ymin1_p = ymax2_p -ymin1_p) and ymax1_p /= 0 and pause_p = '1'*" condition to be fulfilled. The value of *pause* will decide whether the controller becomes P controller or PID controller; the value of *pause* must be either '1' or '0'. Therefore, since the value of *Kp* changes right when *t_p* = *ttest* = 25_{16}, and the value of *pause* is not UUUU or XXXX, this test is considered success.

4.1.1.2 *Signal* y *stable, where* y = r

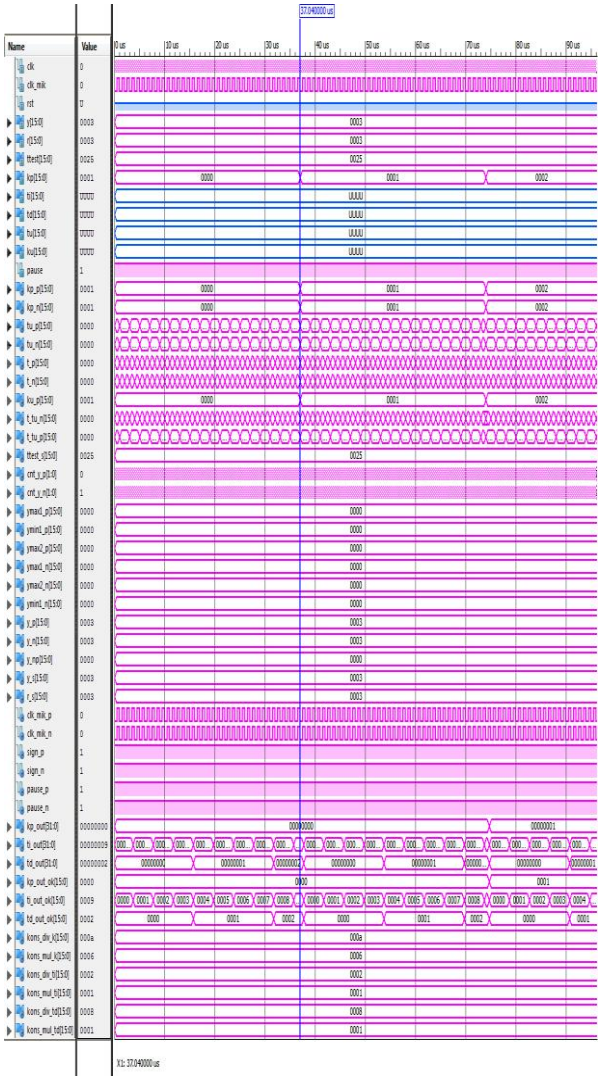

Figure IV-2 *Simulation waveform for Estimator logic block, y=r*

From Figure IV-2 and Figure IV-3, we can tell that the value of *Kp_p* changes 1 *clk* clock after the value of *t_p* turn into 25_{16}. We also know that the value of *t_tu_n* changes after *cnt_y_p*'s value turn into 3_{16} and it will not change if the value of *cnt_y_p* is zero. While the value of *Tu_p* and *Tu_n* is rather messed up because the value of *y* is constantly the same as the value of *r*, this test still can be considered success because they work pretty much fine.

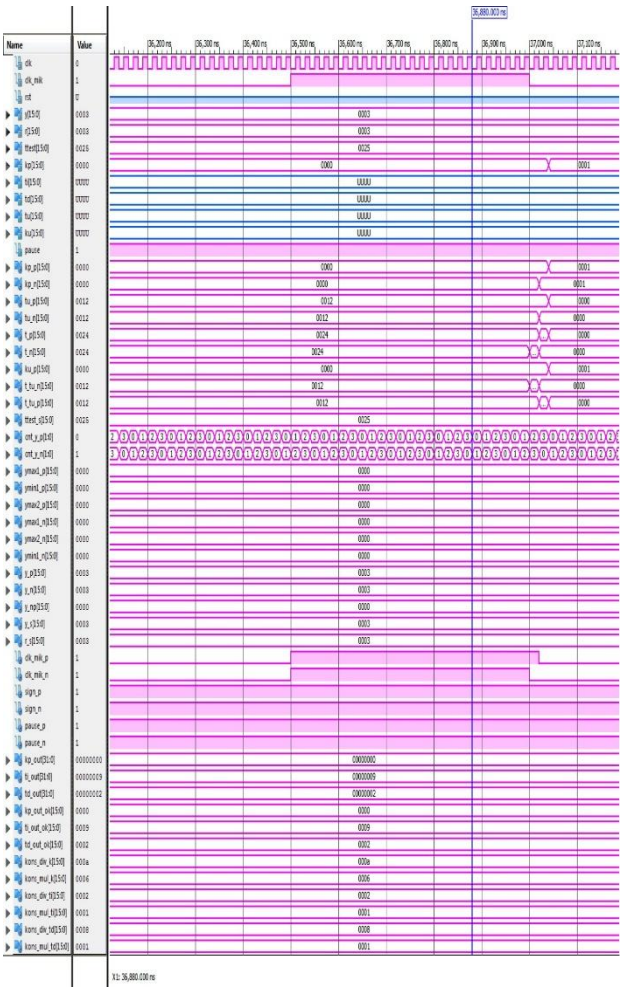

Figure IV-3 *Simulation waveform for Estimator logic block, y=r, zoom*

4.1.1.3 *Signal* y *changing, with constant amplitude*

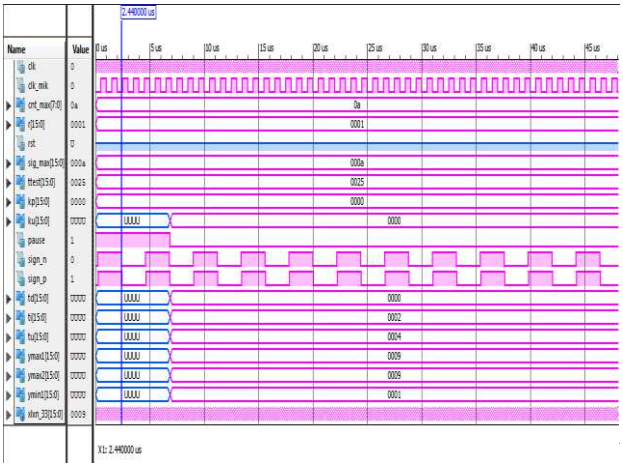

Figure IV-4 *Simulation waveform for estimator logic block, changing y, constant amplitude*

From Figure IV-4 and Figure IV-5 (*xlxn_33* is *y*), we can tell that *ymax1*'s value changes on the first *sign_n*=0 and *sign_p*=1 (2.44 µs), *ymax2*'s value changes on the second one (6.84 µs), and *ymin1*'s value changes at the second *sign_n*=1 and *sign_p*=0 (4.64 µs). After that, since it has fulfilled *"(ymax1_p - ymin1_p = ymax2_p - ymin1_p) and ymax1_p /= 0 and pause_p = '1'"* condition, *Kp* become permanent, *pause* value become '0', and output *Tu*, *Ti*, *Td*, and *Ku* also generate value.

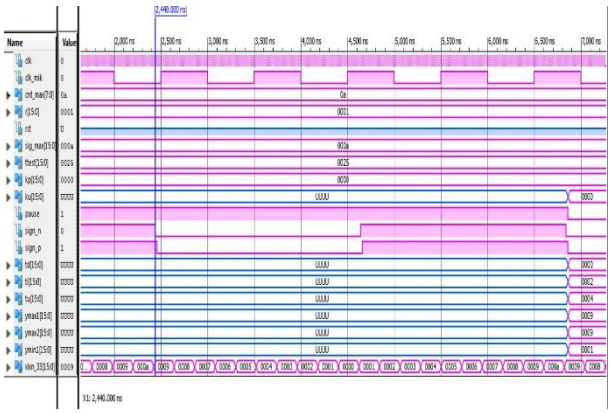

Figure IV-5 *Simulation waveform for estimator logic block, changing y, constant amplitude, zoom*

From the gained values of ymax1, ymin1, and ymax2, we can get Tu = 6.84 µs - 2.44µs = 4.4 µs while the value of Ku is the same as Kp that time, that is 0. From these Ku and Tu's value, we can get Kp, Ti, and Td. As shown in Table IV-1, this logic block can not read and write fractions.

Table IV-1 Supposed and observed values of *Tu*, *Kp*, *Ti*, and *Td*

	Supposed value				Observed value			
Ku	*Tu*	*Kp*	*Ti*	*Td*	*Tu*	*Kp*	*Ti*	*Td*
0	4.4	0	2.2	0.55	4	0	2	0

4.1.1.4 *Signal* y *changing, with changing amplitude*

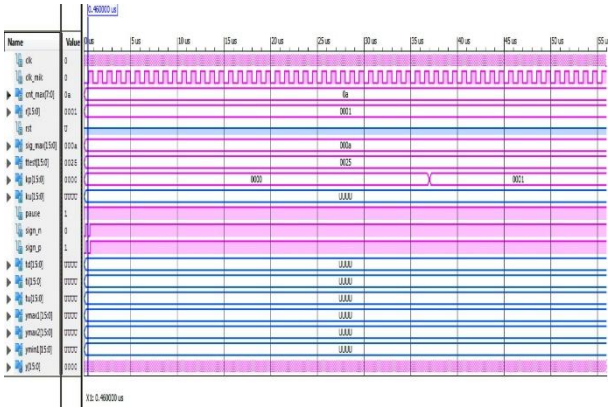

Figure IV-6 *Simulation waveform for estimator logic block, changing y, changing amplitude*

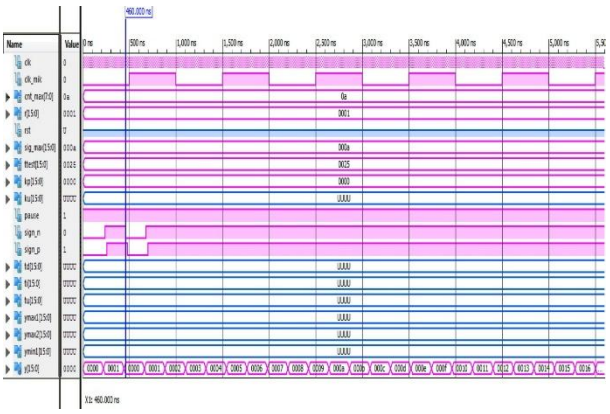

Figure IV-7 *Simulation waveform for estimator logic block, changing y, changing amplitude, zoom*

From Figure IV-6 and Figure IV-7, we can tell that if the amplitude constantly changes, *"(ymax1_p - ymin1_p = ymax2_p -ymin1_p) and ymax1_p /= 0 and pause_p = '1'"* condition will never be fulfilled. Therefore the value of *Kp* keeps increasing and the value of *Ku*, *Tu*, *Ti*, and *Td* remain UUUU.

From the test of constant *y* and changing *y* input, it can be concluded that the estimator module is a success.

4.2 System simulation using ISE

4.2.1 System test 1

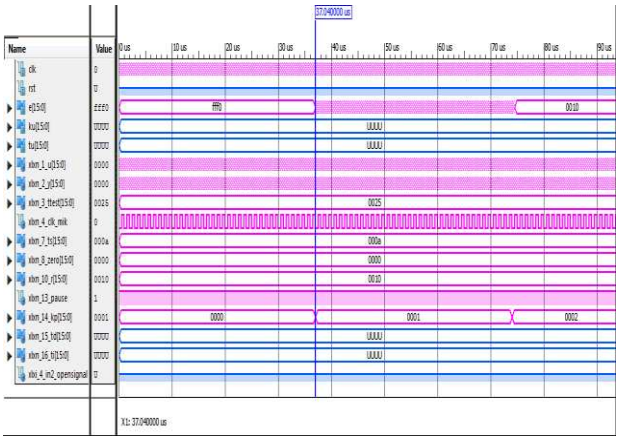

Figure IV-8 *Simulation waveform of system test 1*

From Figure IV-8 and Figure IV-9, we can see that *Kp* changes every 37.04 µs and *pause* have the value of '1'. This shows that the estimator is working fine in the system. It can also be seen that the value of *y* becomes the value of *u* straight away, which shows that the proc_sens (process and sensor) module is working fine in the system. The value of *u* and *e* also changes depend on the value of *y* and *Kp*. This shows that controller module works fine in the system.

Figure IV-9 *Simulation waveform system test 1, zoom*

From Figure IV-9, we can tell that the value of *u* changes after half *clk* time. It also changes to the value of *e* multiply *kp* after half *clk* time. Meanwhile, the value of *y* seems to become the same as the value of *u* without delay.

We can not do IAE performance index yet, because until the time reaches 90 µs (the maximum time in Figure IV-8), the value of *Ku* and *Tu* have not been found, which means the value of *Kp*, *Ti*, and *Td*, which are the parameters, which we intend to apply the performance index on, have not been found yet.

4.2.2 Synthesize of system test 1

This system has a huge problem, that is "WARNING:Xst:1336 - (*) More than 100% of Device resources are used", meaning that in order to apply the system, more device resources are needed. The detail is shown in Figure IV-10.

```
Device utilization summary:
---------------------------

Selected Device : 3s500efg320-4

 Number of Slices:                     6998  out of   4656   150% (*)
 Number of Slice Flip Flops:          10428  out of   9312   111% (*)
 Number of 4 input LUTs:               4249  out of   9312    45%
    Number used as logic:              4021
    Number used as Shift registers:     228
 Number of IOs:                          50
 Number of bonded IOBs:                  50  out of    232    21%
    IOB Flip Flops:                      32
 Number of GCLKs:                         3  out of     24    12%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used
```

Figure IV-10 *Device utilization summary: Spartan 3E Starter Kit*

In order to solve the device resources problem, we tried to synthesize it to a Spartan 6 FPGA. The result is that the problem is solved.

V. CONCLUSIONS

5.1 Conclusions

The reached conclusion is that while the subsystems have been successful, the system itself could not be applied on FPGA Spartan 3E Starter Kit because it does not has sufficient device resources, with the system test 1 used up 150% of Slices and 111% of Slice Flip-Flop.

Parts of the objectives was not achieved because the project is not fully completed. The objectives that was achieved are the first objective, that is to design an estimator logic block to be integrated with PID controller for closed loop system on FPGA, and the second objective, that is to apply Ziegler-Nichols closed loop method using the estimator and the PID controller, with both are successful though they are unable to read and write decimal fractions. The third objective, that is to evaluate the performance in terms of IAE, was not achieved because this project has not been used on the real tools yet.

In order to fully complete this project, we need to be able to apply the system on FPGA with the process, sensor, and all as well as to evaluate the performance of the overall system.

For the future works, it will be better if the estimator can also read and write decimal fractions, which means having a better accuracy, it will also be good if the estimator can search for all three parameters that is for P controller, PI controller, and PID controller, which we can choose one of them depend on the wanted controller.

REFERENCES

[1] Grupo WEG, DC Motors, Brazil: www.weg.net, 2013.

[2] Teco, DC MOTORS, Texas: Teco-Westinghouse, 2000.

[3] K. J. Åström, Control System Design Lecture notes for ME 155A, Santa Barbara: Department of Mechanical & Environmental Engineering University of California, 2002.

[4] K. J. d. T. H. Åström, PID Controllers, 2nd Edition, Alexander Drive: Instrument Society of America, 1934.

[5] B. C. Kuo, Automatic Control Systems Sixth Edition, New Jersey: Prentice-Hall, 1962.

[6] F. Haugen, SCE1106 Control with Implementation, Norway: TechTeach, 2009.

[7] A. d. M. S. Zomorrodi, Comparison of PID Controller Tuning Method, Iran: Department of Chemical and Petroleum Engineering Sharif University of Tecnology, 2013.

[8] Xilinx, Digilent Nexys 2 Board Reference Manual, Pullman: Digilent, 2011.

[9] F. Haugen, Ziegler-Nichols' Closed-Loop Method, Skien, Norway: TechTeach, 2010.

[10] U. d. Farooq, Tree-Based Heterogeneous FPGA Architectures, New York: Springer Science+Business Media, 2012.

[11] Z. Y. C. I. P. Jiao, Distributed-Order Dynamic Systems, Springer.com, 2012.

[12] W. Bolton, Mechatronics Electronic control systems in mechanical and electrical engineering Third Edition, Harlow, England: Pearson, Prentice Hall, 2003.

[13] Cytron, B106 Rotary Encoder User's Manual, Johor: Cytron Technologies, 2010a.

[14] Cytron, SPG50 Series, Cytron Technologies.

[15] Cytron, B106 Series Datasheet, Johor: Cytron Technologies, 2010b.

[16] Xilinx, Spartan-3E Starter Kit Board User Guide, Pullman: Digilent, 2006.