Jurnal Teknologi

Full Paper

HARDWARE DESIGN AND IMPLEMENTATION OF GENETIC ALGORITHM FOR THE CONTROLLER OF A DC TO DC BOOST CONVERTER

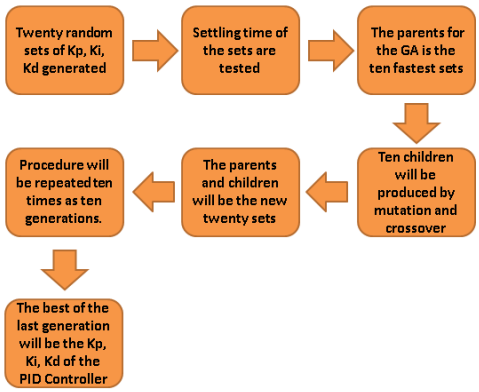
Roderick Yap*, Kevin Lam, Rovi Bugayong, Edward Hernandez, Joey De Guzman

De La Salle University, Manila, Philippines

Article history
Received
02 June 2015
Received in revised form
16 October 2015
Accepted
03 January 2016

*roderick.yap@dlsu.edu.ph

Graphical abstract



Abstract

Controllers for DC to DC Boost Converters have evolved from simple control method to those that involve the use of fuzzy logic controllers. In many implementations, Proportional Integral Derivative (PID) controllers are commonly employed. In this paper, a genetic algorithm for tuning the PID controller of a DC to DC Boost Converter is hardware modelled and implemented on a Field Programmable Gate Array (FPGA) using Verilog as tool for the design entry. The goal of embedding genetic algorithm into the controller is to search for the best PID parameters that will yield fast settling time of the booster at an output of 6V. The hardware implementation allows the controller to tune itself by searching for the best Kp, Ki and Kd values that will give the best settling time. Significantly, this eliminates the need for a separate computer to do the searching routine. Test results of the circuit implemented yielded promising results. When compared to previous researches, the genetic algorithm employed yielded good PID parameters that resulted to a settling time as low as less than 60msec.

Keywords: Genetic Algorithm, Booster PID

1.0 INTRODUCTION

A lot of controllers have been implemented for DC to DC converters. Many of these are implemented using full custom Integrated Circuit Design Approach [1][2]. This type of design approach normally made use of both analog and digital components. Analog designs can be more complicated as a lot of issues such as amplifier gains and phase margins to name a few have to be addressed to guarantee a working hardware implementation. Another approach to designing the controller is to use programmable digital components such as Field Programmable Gate Arrays (FPGA) [3][4][11]. The advantage of using this type of approach is lesser components to deal with since the designer only needs the programming language and the programmable device. Genetic Algorithm has been popularly used for optimization techniques. It basically starts with a random population with each member of the population called Chromosome. The chromosomes evolve through several stages of iterations called generation. The ultimate goal is to search for the best fit chromosome that will serve as the best solution to a problem [6]. In [7], it is used for optimizing in the design of a neural network intended for handwritten numerical recognition. In [8], genetic algorithm was used for solving Job Shop Scheduling problem. In [5], a genetic algorithm was employed for the controller implementation of a DC to DC Boost converter using MATLAB. Instead of dealing heavily on the math, this research focused more on the actual hardware realization and data obtained from the hardware testing.

2.0 SIGNIFICANCE OF THE STUDY

Normally, when implementing genetic algorithm, a computer is used for the heuristic searching operation. In this research, the genetic searching algorithm inclusive of all calculations involved is hardware modelled to implement a standalone system that does not necessitate the use of the computer. The genetic algorithm is applied to a PID controller of a DC to DC Boost converter to allow the system to self-tune during the actual operation. The controller has an output that asserts high to signal the end of the searching operation. It also reveals the best value obtained for the Kp, Ki and Kd of the PID controller intended for the boost converter.

3.0 DESIGN CONSIDERATION

3.1 The Mathematical Algorithm

The goal of the design is to modify an existing PID controller that was successfully implemented in [9] and in [10] to drive a DC-DC Boost converter. In this research, selective design features from these two sources are combined to build a DC-DC Boost converter controller that is tuned using genetic algorithm.

From the controller mathematical equations borrowed in [9] and [10], these equations were edited and expanded to insert Genetic Algorithm for tuning of the PID controller. These equations are presented in the next paragraph. The goal is to search for the best PID parameters i.e. Kp, Ki, Kd values that can give as low as possible settling time at the desired booster output voltage of 6V which is intended to drive 2 white LEDs. The booster input is placed at 3V. Since the booster is controlled using Pulse Width Modulation (PWM), the output voltage is then based on the duty cycle of the signal driving the power transistor of the booster circuit. In general, the higher the duty cycle of the signal, the higher will be the booster output. Let Ton [9] be the duty cycle value of the PWM signal and Δpulse be the PID controller output.

$$Ton = Ton - \Delta pulse \quad (1)$$

$$\Delta pulse = -Kp * e1 - Ki * e2 - Kd\,(e1-e3) \quad (2)$$

Where e1 is the current error also known as new error, e2 represents the cumulative error and e3 is the previous error. Error e1 here is defined as the difference between the target booster output of 6V and the current output voltage of the booster circuit. Eq. 3 and 4 shows the formula for the error parameters.

$$e1 = 6 - Vout \quad (3)$$

$$e2 = e2 + e1 \quad (4)$$

Vout is the output voltage of the Booster. Elitism was chosen for the genetic algorithm to be used. An initial population of 20 randomly generated PID parameters was generated randomly using MATLAB. The population is collected in a matrix. Each member of the population is a chromosome that represents a combination of Kp, Ki, Kd values. This starting matrix will be known as the initial parents. Likewise, an initial set of error values e1, e2 and e3 are set at 3. The operation starts with the fetching of the first chromosome. Every chromosome is evaluated based on its fitness. The chromosome is expressed in binary format as shown by Eq. 5. The index values indicated show 5 bits allotted for each Kp, Ki and Kd. Eq. 5 shows how the chromosome is used to regulate the booster output in conjunction with Eq. 1. The remaining Chormosome bits i.e. chromosome [9:0] are allotted for the settling time values.

$$\begin{aligned}\Delta pulse = {} & -chromosome(i[24:20]) * e1 \\ & -chromosome(i[19:15]) * e2 \\ & -chromosome(i[14:10]) * (e1 - e3)\end{aligned} \quad (5)$$

Where i = 1 to N with N representing the number of population

A fixed time delay period is allotted by the controller through a series of iterations. This fixed delay period is only terminated if either

a) the booster output has settled at the target value or
b) the maximum limit imposed by the time delay has been reached.

In the case of b), the PID parameter is likely to be considered the least fit for the booster. In the case of a), the PID parameter values together with the settling time in terms of number of iterations, is stored in a memory unit after which the booster is once again placed on a reset state. The fitness function is based on the data provided by the hardware circuit. The choice of the best value is based on the chromosome that will give the fastest settling time at an output of 6V. In principle, if f(t) represents the fitness function of the genetic algorithm, then

$$f(t) = \frac{1sec - settling\ time}{1\ sec} \quad (6)$$

where 1 sec is the manually set maximum waiting time for the booster to settle. The settling time is acquired from the hardware circuit.

The whole cycle is repeated with the next PID parameters fed to Eq. 2. The cycle is continuously repeated until all the 20 parent values are tested on the booster controller model. The values stored in the memory are subjected to a sorting operation. The sorter ranks the data starting with the best value i.e. the one with the shortest settling time which corresponds to the highest fitness value. From the 20 ranked values, the first ten parent values are kept and the remaining 10 are discarded. Out of the first ten, based on random selection, the best 10 undergoes crossover to generate 10 new sets of chromosome values called the offspring. The offspring are

subsequently mutated. The mutated offspring together with the parents will form the new 20-chromosome population which will be subjected to the same procedure undergone by the initial population.

3.2 The Hardware Design

Figure 1 shows the general block diagram of the system. The PID controller represents the hardware implementation of Equations 1 to 5 mentioned in the mathematical algorithm section. It fetches the chromosomes from the memory unit. The settling time is tested using the settling time counter block while settled voltage checker block checks if the output voltage has settled. The obtained settling time and the corresponding Kp, Ki, and Kd are stored back into the memory unit. The whole process is repeated for all subsequent chromosome values. The sorter's task is to rank the PID parameters based on the settling time values. The crossover block, aided by the crossover randomizer block, produces the 10 new offspring values. These new offspring randomly undergo mutation using the mutation block. The mutation block is aided by the mutation randomizer block which basically decide which offspring bit will undergo mutation if any. The entire procedure discussed to this point completes 1 generation. The new set of population i.e. parents and offspring will undergo the same procedure of having its settling time on the PID controlled Booster tested and recorded.

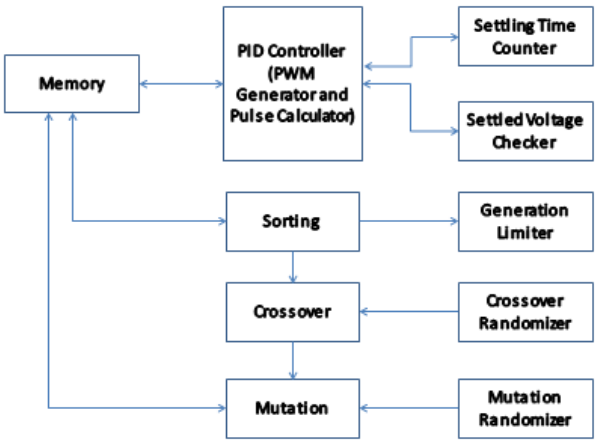


Figure 1 The hardware block diagram

Figures 2 and 3 shows the hardware realization of the PID controller 1st half and 2nd half block respectively. The behaviour is represented by Finite State Machines. As shown in the figures, for the first part, the code starts from the reset state where the 20 initialized chromosomes (Kp, Ki and Kd) sets stored in the memory unit are tested for their settling time. The chromosomes are tested 1 set at a time. The code starts with two solving states. State 00001 solves the integral and the proportional part of the PID equation while state 00010 solves the differential part. Delta pulse calculate solves the change in PWM duty cycle which will then be calculated in state 00100. Pulse Limit state limits the duty cycle of the PWM within the allowable range. PID idle is a waiting state for the booster output to be at a certain voltage value after 2ms prior to sampling the output voltage. The latch state is used to measure the value of the output voltage. This is followed by the next state which solves the error i.e. difference between the target voltage of 6V and the measured value. After getting the error, state 01001 will check if the voltage output is already in the desired 6V output. If not, then it will go back to PID solve 1 state to compute for the new duty cycle needed to bring the output voltage to 6V. If the Voltage output is considered settled by the settled voltage checker module, the code will then go to the settled voltage state whether the previous state is in any from 00000 to 01001. The settling time will then be saved. The code will then go to the delay state to give time for the booster to discharge and will go back to the reset state. The settling times of one set of Kp, Ki, and Kd values will be measured four times for repeatability. Once the procedure is done where 4 settling times are recorded, the settling times will be averaged and the memory will be updated. The address will be incremented for the next chromosome with the same procedures. When all 20 chromosomes have their settling time recorded. What follows is the choosing of the 10 best chromosomes from the 20 recorded values through a sorting block. The top 10 are subjected to crossover and mutation to produce the 10 offspring values. These new sets of chromosomes will undergo the same procedure until a new top 10 is obtained.

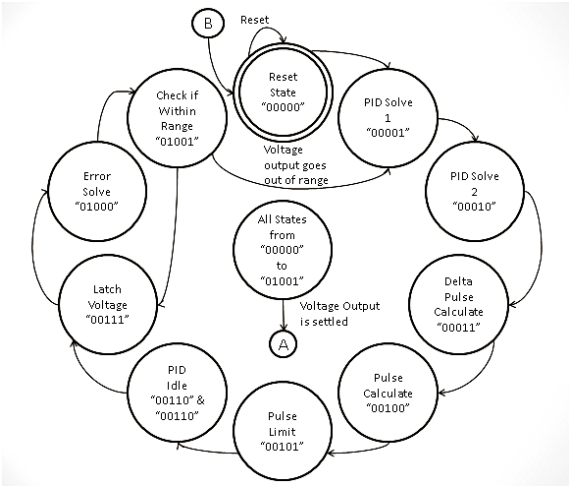


Figure 2 Genetic algorithm based PID controller -1st Half

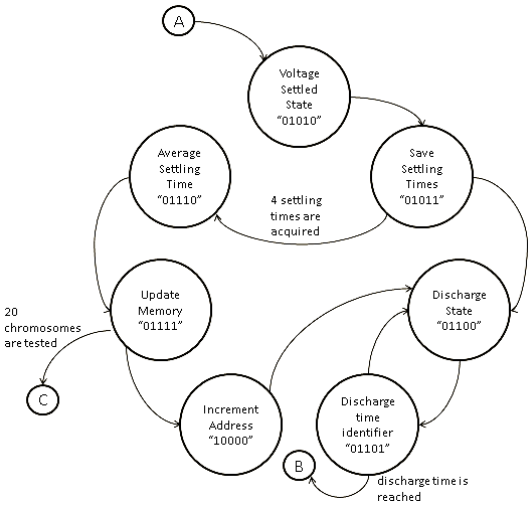


Figure 3 Genetic algorithm based PID controller -2nd Half

Figure 4 shows the FSM for the crossover routine. Starting from reset state that also serves as the wait state, the Crossover code waits until the sorting is finished. Once it receives a signal that sorting is done, it is then directed to crossover routine. Next state is the swap state where the first two inputs which are the parents will crossover, meaning genes Kp or ki or Kd of the chromosomes will swap to produce two new children. After swapping, the address of the memory unit will be incremented for the next two parents. The state comes back to the swap state to crossover again. This procedure will repeat until all parents are swapped. It will then go to the swap finish state which makes an output pin equal to 1 to tell the mutation module to start. It will go back to the wait sorting state and will proceed again only for the next generation.

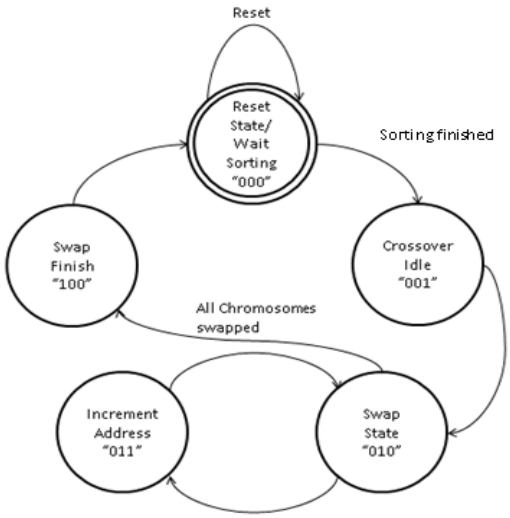


Figure 4 FSM for the crossover routine

Figure 5 shows the FSM for the mutation routine. Starting from reset state that also serves as the wait state, the mutation routine waits until the crossover is finished. Once it receives a signal that crossover is done, it will then be routed to the mutation routine. Next state is the mutation state where the first input which is an offspring from the crossover will mutate. After mutation, the address will be incremented for the next offspring. The state comes back to mutation after passing through mutation idle state to mutate again. This procedure will repeat until all offspring are mutated. It will then go to the mutation finish state to tell the memory that the mutation is finished. The memory will be updated with the parents from the sorter and the children from this module It will go back to the wait crossover state and will proceed again only for the next generation.

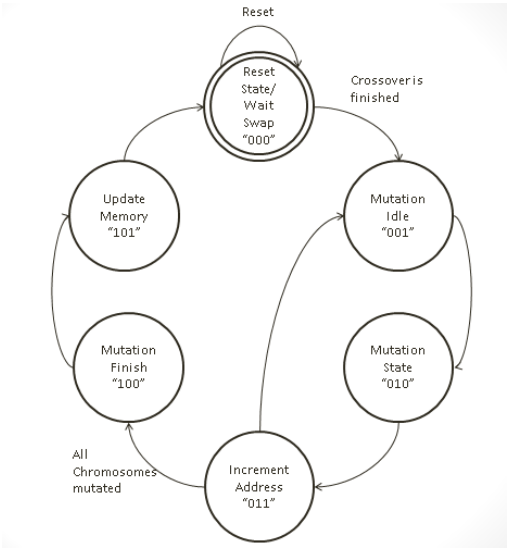


Figure 5 FSM for the mutation routine

Figure 6 shows the circuit used for the randomizer. The rectangular blocks represent D Flip flops. The random numbers are obtained from the exclusive or gates. The circuit shown is a modified version of the circuit shown in [12]. Randomizer is used for the crossover and mutation routine.

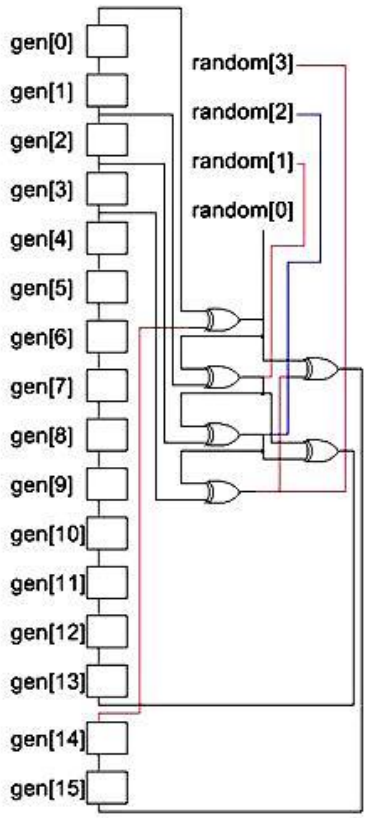


Figure 6 Randomizer

4.0 RESULTS AND DISCUSSION

In order to verify the validity of the design, the genetic algorithm together with the PID controller model was subsequently implemented on a Field Programmable Gate Array (FPGA). A booster circuit was likewise built and interfaced to the FPGA. Figure 7 shows the booster circuit to be controlled. The circuit will drive 2 white LEDs for a total of 6V output.

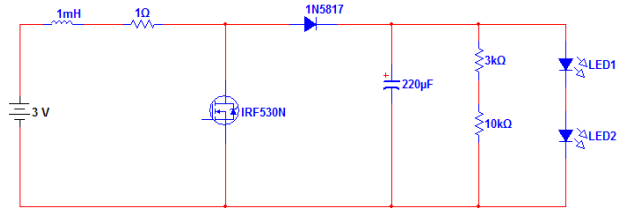


Figure 7 The booster circuit

Fig. 8 shows the picture of the prototype representing the whole system. Three sets of twenty chromosome values were randomly generated using MATLAB as the initial parent values. Embedded in the Chromosome values are the settling time which are all initially set to 0. The PWM frequency was set at 100kHz.

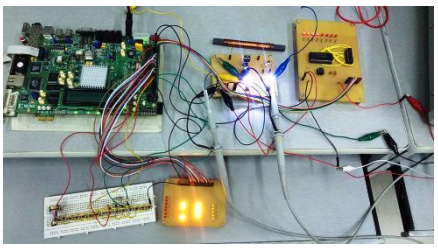


Figure 8 Prototype for hardware verification

The genetic algorithm was run and evaluated on the hardware set up. For every chromosome value tested, the PID controller inside the FPGA outputs the corresponding PWM signal to the booster circuit of Figure 7. The output voltage from the booster is fed back to the FPGA in a digital format through an analog to digital converter. An internal counter inside the FPGA serves as timer in measuring the settling time of the booster output. Figure 9, 10 and 11 show the respective summary result of the 3 sets of 20 randomly generated chromosome values. In order ensure integrity of data, all settling time must be within a period of 100msec. This means all values at the timer column are in excess of 100. As for example, a timer value of 145 means the settling time is 45msec. The top 5 chromosomes shown in each table are picked from the 10 best values obtained during the searching routine. The top 10 chromosomes of the last generation are played again 8 times. The system will then rank them based on the performance results to yield the top 5 fittest chromosomes along with their average settling time. The figure show the top 5 best performing chromosome values for each set of test. In order to ensure the repeatability of the values obtained, each chromosome will undergo 4 repeated trials using oscilloscope this time to measure the settling time. Figure 12 shows a sample waveform obtained from the oscilloscope. Experimental data obtained show promising results since all the settling time data obtained for every trial are close enough and these data in terms of the best values agree with the results obtained from the FPGA. This guarantees repeatability.

Rank	kp	ki	kd	Timer	Settling time (in ms)				Average	Vout
					1st	2nd	3rd	4th		
0	17	5	19	145	45	45	45	45	45	6.03
1	19	4	26	145	55	55	55	55	55	5.95
10	22	2	30	145	57	57	57	57	57	5.95
11	15	16	28	145	55	55	55	55	55	6.03
100	7	17	28	145	57	55	57	55	56	5.95

Figure 9 Results for 1st batch of randomly generated chromosomes

Rank	kp	ki	kd	Timer	Settling time (in ms)				Average	Vout
					1st	2nd	3rd	4th		
0	18	7	26	146	55	55	55	55	55	5.96
1	18	7	26	146	55	55	55	55	55	5.96
10	22	3	28	156	62	62	62	62	62	5.96
11	21	7	28	167	64	64	64	64	64	6.03
100	20	7	28	167	63	61	62	61	61.75	5.96

Figure 10 Results for 2nd batch of randomly generated chromosomes

Rank	kp	ki	kd	Timer	Settling time (in ms)				Average	Vout
					1st	2nd	3rd	4th		
0	13	12	18	145	45	50	50	50	48.75	6.03
1	20	3	25	145	60	55	55	50	55	6.03
10	22	3	27	155	70	65	60	70	66.25	6.03
11	22	1	25	155	70	65	65	70	67.5	6.02
100	20	5	25	155	65	65	65	70	66.25	6.02

Figure 11 Results for 3rd batch of randomly generated chromosomes

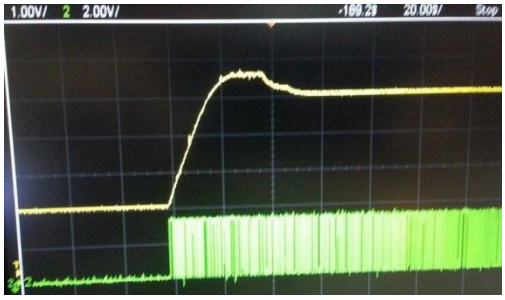


Figure 12 Camera shot of the waveform settling time from Oscilloscope

5.0 CONCLUSION

In this paper, a genetic algorithm was proposed for the control of a DC to DC boost converter. the goal was to investigate if the entire system can be hardware modelled and subsequently implemented as a standalone system without needing the use of a separate computer to do the heuristic searching operation. The system was successfully implemented in hardware using an FPGA with Verilog as design entry tool. The searching algorithm proved successful because results show a settling time as low as less than 60msec. This is in contrast with [9] which obtained a settling time of 108msec because the PID tuning method was done manually. For further studies, we recommend a bigger initial population for the searching operation. This will mean bigger hardware and memory requirement for the system. It is also recommended that the ranking operation of the genetic algorithm be inclusive of both settling time and booster output value closest to 6V.

Acknowledgement

The author wishes to thank the University Research Coordination Office (URCO) of De La Salle University for funding this research. Special thanks also to Mr. Gerard Ely Faelden for the help on Genetic Algorithm.

References

[1] Yeong-Tsair Lin, Wen-Yaw Chung. 2005. A Monolithic CMOS Step-Down DC-DC Converter, *Circuits and Systems*, 2005. 48th Midwest Symposium on, Aug 2005

[2] Wan-Rone Liou et al. 2007. A High Efficiency Dual-Mode Buck Converter IC For Portable Applications Communications, *Circuits and Systems*, 2007. ICCCAS 2007. International Conference on, July 2007

[3] Islam ,M. Murshidul, Allee , David R., Konasani , Siva, and Rodriguez, Armando A. Rodriguez, 2004. A Low-Cost Digital Controller for a Switching DC Converter with Improved Voltage Regulation, *IEEE Power Electronics Letters*, 2(4): 121-124.

[4] Dumalag, Lawrence and Yap, Roderick. 2012 FPGA Based Multi-Decision Level PID Controller for Boost Converter, *5th AUN/SEEDNET Regional Conference on Information and Communications Technology*.

[5] K.Sundareswaran et. al. 2010. Robust Controller Identification for a Boost Type DC-DC Converter Using Genetic Algorithm. *2008 IEEE Region 10 Colloquium and the Third ICIIS, Kharagpur*.

[6] Pengfei Guo et al. 2010. The Enhanced Genetic Algorithms for the Optimization Design, *2010 3rd International Conference on Biomedical Engineering and Informatics (BMEI 2010)*.

[7] Tai-Shan Yan et. al. 2007. Research on Handwritten Numerical Recognition Method Based On Improved Genetic Algorithm AND Neural Network, *Proceedings of the 2007 International Conference on Wavelet Analysis and Pattern Recognition, Beijing, China*.

[8] Hongze Qiu et. al. 2009. A Genetic Algorithm-based Approach to Flexible Job-shop Scheduling Problem. *Natural Computation, 2009. ICNC '09. Fifth International Conference*.

[9] Dumalag, Lawrence. 2010. FPGA Based Multi-Decision Level PID Controller for Boost Converter, A Thesis Document, *De La Salle University, Manila, Philippines, August, 2010*

[10] Dee, Robert Isaac et. al. 2010. Development of an FPGA-based controller for the implementation of a DC-DC boost converter, A Thesis Document, *De La Salle University*.

[11] Caldo, R. and Yap, R. 2013. Design, development and implementation of a fuzzy logic controller for DC-DC Buck and Boost converter in an FPGA, *Control, Automation and Information Sciences (ICCAIS), International Conference on, November 2013*.

[12] Pseudo-Random Number Generation Routine for the MAX765x Microprocessor [available online: https://www.maximintegrated.com/en/app-notes/index.mvp/id/1743]