Revista ELECTRO, Vol. 47, 2025, pp. 64 - 69
https://itchihuahua.mx/revista_electro
ISSN Electrónico: 3061-774X

VERILOG SIMULATION OF A DISCRETE SYSTEM CONTROLLED BY PID

[1]Sáenz-Zamarrón David, [1]Arana-de-las-Casas Nancy Ivette, [2]López-Salas Iraam Antonio, [3]Sortillón-González Patricia Eugenia, [1]Velazco-Soto Diego
[1]Tecnológico Nacional de México / Instituto Tecnológico de Cd. Cuauhtémoc
División de Estudios de Posgrado e Investigación
Av. Tecnológico #137, C.P.31500, Cuauhtémoc, Chih.
[1]narana@itcdcuauhtemoc.edu.mx
[2]Tecnológico Nacional de México / Instituto Tecnológico de Aguascalientes
Departamento de Ingeniería Electrónica
Av. Adolfo López Mateos #1801, C.P. 20255, Aguascalientes, Ags.
[3]Universidad Estatal de Sonora
Departamento de Manufactura e Ingeniería Industrial
Av. Ley Federal del Trabajo, S/N, C.P. 83100, Hermosillo, Son.

RESUMEN

El compensador PID tradicional dentro de los sistemas de control se utiliza ampliamente en ingeniería práctica. El PID puede desarrollarse en hardware dedicado directamente en una oblea de semiconductor. Una posibilidad de fabricación de chips para el público en general es el proyecto Silicluster. En este artículo un controlador PID se diseña y simula utilizando MATLAB-Simulink-Vivado para una planta de tiempo continuo de segundo orden. El sistema dinámico se discretiza y se implementa en un Lenguaje de Descripción de Hardware (HDL) como Verilog y queda listo para la fabricación de un Circuito Integrado de Aplicación Específica (ASIC). En Verilog se reproduce el comportamiento del sistema dinámico en un lazo de control en tiempo real utilizando una estructura ARMA. El sistema tiene una cuantificación limitada por las especificaciones propias del proyecto Silicluster. El ARMA se implementa en aritmética de punto fijo. Se realizaron simulaciones del comportamiento de la respuesta al escalón del sistema.
Palabras Clave: Silicluster, PID, ARMA, Notación en punto fijo.

ABSTRACT

The traditional Proportional-Integral-Derivative (PID) compensator is widely applied in practical control engineering due to its simplicity and effectiveness. It can be implemented in dedicated hardware directly on a semiconductor wafer. A public-accessible initiative for chip manufacturing is the Silicluster project. In this work, a PID controller is designed and simulated using MATLAB-Simulink-Vivado for a second-order continuous-time plant. The dynamic system is discretized and implemented in a Hardware Description Language (HDL), specifically Verilog, preparing it for an Application-Specific Integrated Circuit (ASIC) deployment. A real-time feedback control loop is established in Verilog using an Auto-Regressive Moving Average (ARMA) structure. It is constrained by the quantization limitations defined by Silicluster. The ARMA model is implemented using fixed-point arithmetic to meet hardware constraints. Step-response simulations validate the proposed control design.
Keywords: Silicluster, PID, ARMA, Fixed point-based notation.

1. INTRODUCTION

For several decades, PID control has been a standard methodology for industrial process control. Its enduring popularity stems from its simple structure, robust performance, and ease of implementation. A PID controller can be designed with minimal information about the process model while achieving acceptable performance in both steady-state and transient responses.

Currently, PID controllers are commonly implemented in digital systems operating in discrete time. These controllers can be deployed either in software or hardware. Software-based implementations, however, often are more vulnerable to cybersecurity threats. Hardware-based implementations are typically executed on general-purpose microprocessor units (MPUs). Alternatively, developing the controller in dedicated hardware offers several advantages: increased compactness, greater adaptability to specific applications, improved speed through parallel processing, reduced sensitivity to electromagnetic interference, and enhanced energy efficiency [1].

Digital design and simulation tools are used to program Field-Programmable Gate Arrays (FPGAs), enabling the implementation of discrete-time dynamic systems. In this study, the control technique employed is PID, which has been extensively utilized in recent research. The following section presents relevant examples of both FPGA-based architectures and PID implementations.

In this context, [2] developed a closed-loop motion control system integrating a Back Propagation Neural Network (BPNN) with a PID controller implemented on a Xilinx FPGA.

Modelsim and Simulink simulations were used to validate the system's performance, which featured self-tuning of the PID parameters and demonstrated superior real-time responsiveness compared to traditional MPU-bases implementations.

In [3] an FPGA was configured to control both the average temperature and spatial uniformity within a TEM-cell environment. A PID controller with 24 distinct parameters sets were implemented in Verilog using Quartus software. The system achieved a temperature regulation precision of approximately 0.1°C.

In [4] an FPGA based core was developed for real-time adjustment of PID parameters, using a multilayer neural network and the BPNN algorithm. The design was implemented in Verilog through the Vivado platform for ASIC deployment. The

system showed significantly improved rise and settling times, which are critical performance metrics in healthcare applications.

In [5] a methodology was presented for modeling, designing, and implementing an FPGA based PIDF (PID with Feedforward) controller for an industrial system, specifically a physical twin. A digital twin was created to facilitate optimal controller development on a resource-constrained FPGA. The results showed a control error between the digital and physical twin systems of ±0.2%.

In [6], a Digital Pulse Width Modulation (DPWM) architecture was combined with a digital PID controller to regulate a DC-DC converter. Instead of performing multiplications, a look-up table was used to store both the duty cycle ratio and the error value. It effectively replicated PID control behavior for power converters. The architecture was implemented in Verilog and validated through simulation.

In [7], a robot motion control system was developed using a BPNN-PID-based algorithm. The mobile robot, equipped with three omni-directional wheels, employed a closed-loop control system implemented in Verilog on an FPGA. The study demonstrated that the BPNN-PID outperformed the traditional PID algorithm in terms of tracking precision and obstacle avoidance.

In [8], Brushless DC (BLDC) motors were used for speed control, implemented via a PID controller on an FPGA development kit using VHDL. The results demonstrated that the FPGA provided smoother and more accurate speed regulation compared to traditional MPU-based control.

Additional studies have extended PID implementation on hardware platforms. For example, [9] explored the integration of Very Large-Scale Integration (VLSI) with PID controllers for improved temperature regulation.

In [10], an ASIC-based solution reduced sampling and computation delays using fuzzy logic, predictive control, and a neural network PID.

Similarly, [11] implemented an incremental PID with PWM on Xilinx platforms.

While [12] employed a fuzzy PID design in FPGA to achieve faster transient response.

In [13], a PID controller with a low-pass filter (LPF) was developed to enhance stability and reduce overshoot in power systems.

Once the digital control design stage is complete, an ARMA structure is delivered as a controller. The control can then be implemented in HDL for an FPGA or ASIC.

For example, [14] described a many-channel experiment control system based on a FPGA. They carried out Infinite-Impulse-Response (IIR) PID filter. These IIR filters similar to ARMA use fewer FPGA resources than straightforward multiplier-based filters.

These previous works highlight the relevance and evolution of PID control techniques on reconfigurable and application-specific hardware. However, limited attention has been paid to standardizing fixed-point implementations of ARMA-structured PID controllers under public fabrication constraints such as those of the Silicluster project. This paper aims to fill that gap by proposing a design and simulation methodology that prepares a PID-based control system for ASIC integration within these constraints.

In this article, we use a dynamic system to discretize it, design a controller for its implementation in Verilog, and then prepare it for subsequent ASIC manufacturing.

This work is addressed in three main stages: Discretization of a standard second-order continuous-time plant; design of a PID controller; finally, everything will be compressed into the ARMA structure to program it in Verilog.

2. PRELIMINARIES

This section describes the fundamental concepts that support this work.

2.1. Discretization

Discretization encompasses a set of techniques that enable the control of a system using digital computation. A digital controller performs addition and multiplication operations, which allow for numerical integration and differentiation. Consequently, the controller can be represented as an Auto-Regressive Moving Average (ARMA) model, which is the discrete-time equivalent of a continuous-time differential equation.

2.2. PID

The PID controller continuously computes the error e as the difference between the reference input r and the actual output c. This error e tends toward zero as the system approaches steady-state. However, during this convergence, transient dynamics emerge in the closed-loop system.

The proportional gain K_P contributes to the immediate system response; the integral gain K_I addresses accumulated past errors; and the derivative gain K_D anticipates future trends based on the rate of change. Together, these three components enable the PID controller to apply corrective actions that stabilize and regulate the overall system dynamics [8].

2.3. Silicluster

Silicluster is a collaborative initiative aimed at fabricating microchips capable of hosting up to 256 independent digital designs. It allocates silicon space for efficient implementation, with each project configured to use a 16-bit digital input, a 16-bit digital output, a 10 MHz clock, Vdd, and Ground, within an area of 130x180 nm (approximately 1,300 logic gates). Designs are modeled using Verilog, an HDL suitable for digital systems. Verilog and VHDL—both supported in the AMD Vivado™ development suite—facilitate design, synthesis, simulation, and verification of ASIC and FPGA implementations.

Once the design is validated, it is passed through a tool chain that translates the HDL code into semiconductor layout layers for chip fabrication. The Efabless platform enables the creation of

custom chips by providing open-source and licensed tools, along with community and industry support.

These foundational concepts—discretization, PID control principles, and the Silicluster platform—form the basis for the proposed methodology. Together, they support the design and hardware implementation of a digital control system suitable for fabrication under constrained silicon environments.

3. DEVELOPMENT

This section presents the analytical procedure used to design the PID controller. The process begins with the formulation of a differential equation representing a Linear Time-Invariant (LTI) system.

$$\frac{d^2y(t)}{dt^2} + \eta\frac{dy(t)}{dt} - \eta^2 m(t) = 0 \quad (1)$$

The corresponding transfer function of the plant is then derived from its differential equation.

$$\boldsymbol{Gp(s)} = \frac{\boldsymbol{Y(s)}}{\boldsymbol{M(s)}} = \frac{\boldsymbol{\eta^2}}{\boldsymbol{s(s+\eta)}} \quad (2)$$

The system's Natural Frequency being η, and Damping of 0.5.

3.1. Discretization

For the system shown in Figure 1, the objective is to determine the discrete PID controller $D(Z)$ such that the design meets a settling time t_s and an overshoot M_p.

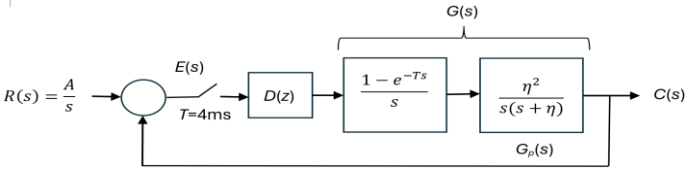

Figure 1. Feedback control system.

The discrete transfer function $G(z)$ can be obtained from Equation (3), based on the continuous-time plant $G_p(s)$ [15].

$$G(z) = \underbrace{\left(1 - e^{-Ts}|_{e^{Ts}=z}\right)}_{\text{zero-order data-hold}} \underbrace{\frac{z[(\eta T-1+e^{-\eta T})z+1-e^{-\eta T}-\eta T e^{-\eta T}]}{(z-1)^2(z-e^{-\eta T})}}_{\text{sampler}} \quad (3)$$

3.2. PID

The design specifications require a specific setting time t_s and an overshoot M_p, with no steady-state errors for a constant input.

Calculation of the β angle.

$$\beta = \tan^{-1}\left(\frac{-\pi}{\ln\left({}^{M_p}/_{100}\right)}\right) \quad (4)$$

Calculation of the σ parameter.

$$\sigma = {}^{4}/_{t_s} \quad (5)$$

Calculation of Natural Frequency.

$$w_n = {}^{\sigma}/_{\cos\beta} \quad (6)$$

Calculation of Damping.

$$\varepsilon = {}^{\sigma}/_{w_n} \quad (7)$$

Calculation of Damped Frequency.

$$w_d = w_n\sqrt{1-\varepsilon^2} \quad (8)$$

With these desired parameters, the corresponding closed-loop pole location $z_1 = r\lfloor\theta$ on plane Z of the discrete plant $G(z)$ is given next.

The complex number z_1 is a Polar location with radius r on plane Z.

$$r = e^{-\varepsilon w_n T} \quad (9)$$

And an angle θ of z_1 on plane Z is.

$$\theta = w_n T\sqrt{1-\varepsilon^2} \quad (10)$$

The desired point z_1 in Cartesian coordinates is given by.

$$z_1 = r\cos\theta + r\sin\theta i \quad (11)$$

Figure 2 shows the location of desired point z_1 on plane Z.

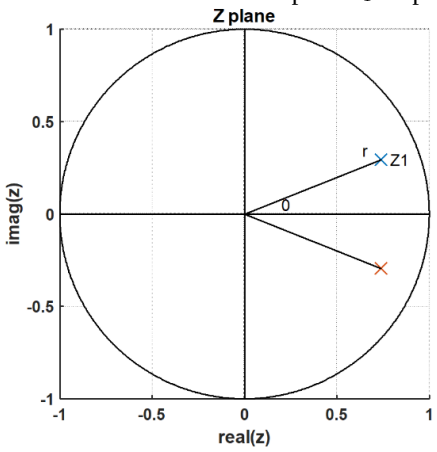

Figure 2. Desired point z_1 on complex plane Z.

It is necessary to transform the desired z_1 point on plane Z to its equivalent s_1 point on Laplace plane S. It is because the PID algorithm runs considering the properties of the Laplace plane.

The complex pole location at s_1 (see figure 3), given the location of z_1 is that of Eq (12).

$$s_1 = {}^{\ln z_1}/_{T} \quad (12)$$

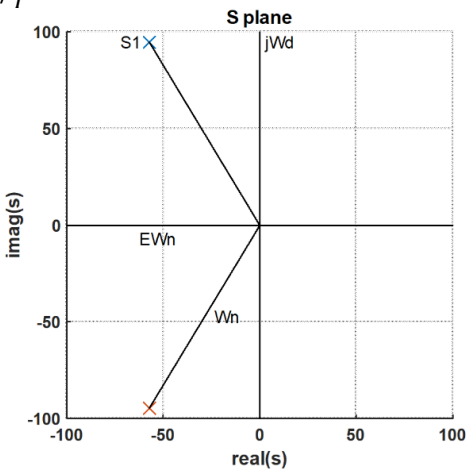

Figure 3. Desired point s_1 on complex plane S.

We use Bilinear Transformation [15].

$$z_1 = \frac{1+\left(\frac{T}{2}\right)s_1}{1-\left(\frac{T}{2}\right)s_1} \quad (13)$$

To evaluate the discretized plant on the desired point s_1.

$$G(s_1) = G(z)|_{z_1} = \left.\frac{(\eta T-1+e^{-\eta T})z+1-e^{-\eta T}-\eta T e^{-\eta T}}{(z-1)(z-e^{-\eta T})}\right|_{z_1} \quad (14)$$

That is because the PID method used here, [16] is based on root-locus properties, it requires the proper stability limit (jw) of the Laplace plane S.

Then we obtain the Magnitude and Angle of the discretized plant evaluated on the desired point s_1.

So, we have the desired point s_1 in Euler's representation.

$$s_1 = |s_1|e^{j\beta} \quad (15)$$

The discretized plant evaluated on the s_1 point, also in Euler way.

$$G(s_1) = |G(s_1)|e^{j\psi} \quad (16)$$

For PID controller design, K_I must be given to satisfy a certain steady-state response. The other two gains: K_D and K_P are computed from equations (17) and (18).

These equations require that s_1 would be complex conjugated. For the case that s_1 is real, sin β=0 and equations cannot be employed.

Calculation of the differential gain.

$$K_D = \frac{sen\psi}{|s_1||G_P(s_1)|sen\beta} + \frac{K_i}{|s_1|^2} \quad (17)$$

Calculation of the proportional gain.

$$K_P = \frac{-sen(\beta+\psi)}{|G_P(s_1)|sen\beta} - \frac{2K_i\, cos\beta}{|s_1|} \quad (18)$$

The Laplace transform of the PID equation yields the transfer function.

$$D(s) = k_p + \frac{k_I}{s} + k_D s \quad (19)$$

The discrete PID controller includes the discrete approximation of the continuous derivative and integral.

Hence the transfer function of a discrete differentiator and integrator uses a digital-filter for differentiation and a digital-filter for trapezoidal rule for integration respectively.

The z-transform of the discrete PID equation yields the transfer function [16].

$$D(z) = K_p + K_I\frac{T}{2}\left[\frac{z+1}{z-1}\right] + K_D\left[\frac{z-1}{Tz}\right] \quad (20)$$

This section established the theoretical groundwork and design steps necessary to implement a discrete PID controller that meets predefined performance specifications. The resulting controller structure, expressed in ARMA format, will be implemented and evaluated in subsequent sections using both floating-point and fixed-point arithmetic representations.

4. RESULTS

To validate and demonstrate the behavior of the proposed control system for the development and implementation of the control algorithm with Silicluster constraints, a software test was performed.

4.1 Numerical evaluation

To evaluate the designed control system, the following parameter values were selected:

T = 4ms, η = 100
M_p=15% overshoot
t_s=0.07 setting time
K_I = 0.1 integral gain
Then, we obtain.

$$G(z) = \frac{7.032z+6.155}{100z^2-167z+67.03} \quad (21)$$

K_D = 0.0031, K_P = 1.1665

$$D(z) = \frac{0.007743z^2-0.01082z+0.003076}{0.004z^2-0.004z} \quad (22)$$

Figure 4 shows the feedback control system in Simulink.

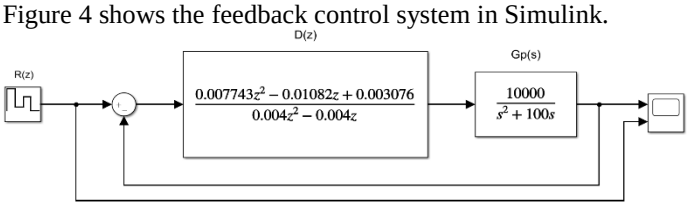

Figura 4. Simulink diagram of the feedback control system.

Figure 5 shows a skyline-type test signal (red) and the PID control system response (yellow).

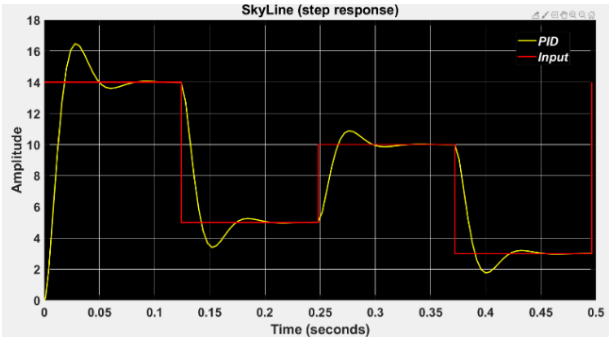

Figura 5. Skyline system response in Simulink.

4.2 ARMA, floating-point notation

The combined transfer function $D(z)G(z)$ was obtained using floating-point representation for all coefficients, as generated by the design algorithm. This formulation results in a third-order ARMA model that encapsulates both the plant and controller dynamics.

$$D(z)G(z) = \frac{C(z)}{E(z)} = \frac{0.1361z^2+0.06508z-0.04732}{z^3-1.67z^2+0.67z} \quad (23)$$

It produces a 3rd. Order polynomial ARMA structure of the controller and plant.

So, with the algorithm we obtain the five parameters presented on the ARMA model Eq (24) and the samples from the error input e, and the output c gotten in real time. In the entire control system, the input samples are those of the setpoint r, while those of the output c, are obtained from the dynamic model of the system.

$$c(k)=a_1*e(k)+a_3*e(k\text{-}2)+a_4*e(k\text{-}3)+b_2*c(k\text{-}1)+b_3*c(k\text{-}2) \quad (24)$$

Revista ELECTRO, Vol. 47, 2025, pp. 64 - 69
https://itchihuahua.mx/revista_electro
ISSN Electrónico: 3061-774X

The control implementation based on this equation requires numerical computations that are constrained by the available hardware resources. The design must adhere to the hardware limitations defined by the Silicluster platform, including a 16-bit digital input and output interface, a 10 MHz clock signal, and a logic gate budget of approximately 1,300 gates.

With respect to the input interface, the organization was established as shown in Table 1. The 16-bit input is divided into 3 fields. Sel, Coef, and *r*. The field Selection (Sel), 3-bit length, defines the ARMA's 8-bit Coefficient (Coef) to be introduced into the digital system, that is a_1-a_4, b_2-b_3. Also, it define the frequency divider (MSB and LSB) to obtain the sampling time *T*, given by the clock frequency (Clk=10MHz).

$$T = \frac{1}{\frac{\text{Clk}}{\text{Divider}}} = \frac{1}{\frac{10\text{MHz}}{(9\text{C40})_\text{H}}} = 4\text{msec} \tag{25}$$

When Sel=111, then a 5-bit input setpoint *r*, is introduced.

Next a hardware scheme is developed through Verilog to implement the feedback control system. The 16-bit length output *c* comes from the product of two 8-bit factors.

Table 1. Input digital interface

	Sel	Coef	(bits)	
0	000	a_1	8	0.1361
1	001	a_3	8	0.06508
2	010	a_4	8	-0.04732
3	011	b_2	8	1.67
4	100	b_3	8	-0.6703
5	101	LSB	8	Freq Divider (9C40)H
6	110	MSB	8	
7	111	*r*	5	setpoint

Figure 6 shows the simulation of the step and ramp test signals (red), the discretized plant (blue), and the PID control system response (yellow).

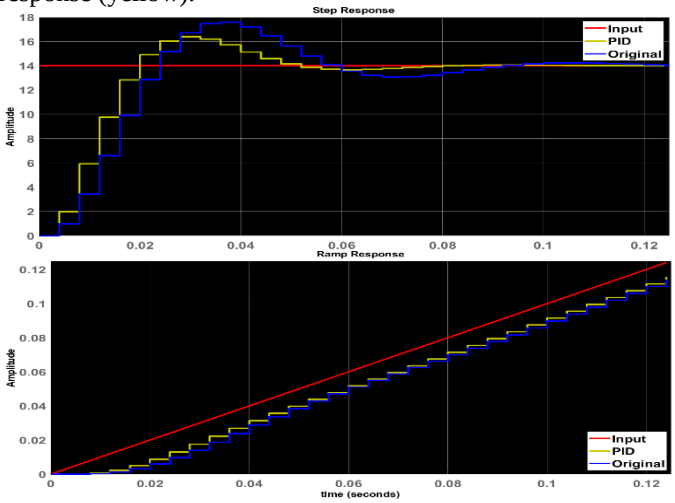

Figura 6. Step and Ramp system response in Matlab.

4.2 Fixed-point notation

ARMA's coefficients in Table 1 are shown in floating-point notation, but, to take advantage of hardware integer multiplication, a fixed-point conversion of the ARMA is developed.

Consider the following product as an example of one of the multiplications generated in the ARMA. a_4 and $e(k\text{-}3)$ are fractional numbers, but, when multiplied by the adequate quantization factor, they become 8-bit integer factors *a* and *e* Eqs (26) and (27); and the multiplication can be carried out with two 8-bit integer factors, generating a 16-bit product a_4*$e(k\text{-}3)$ [15]. But only 8 bits must be returned into the feedback. Therefore, the 8 least significant bits are discarded, producing a truncation error. The truncation error introduces DC bias into the signal and noise.

$$a_4 * e(k-3) = \underbrace{[a_4 2^6 2^2]}_{a\text{ integer}} \underbrace{[e(k-3)2^7]}_{e\text{ integer}} \left[\frac{1}{2^7}\right] \tag{26}$$

$$a_4 * e(k-3) \approx \underbrace{[2a_4 2^6]}_{a\text{ integer}} \underbrace{[2e(k-3)2^7]}_{e\text{ integer}} \tag{27}$$

4.3 Verilog code

Code for the Verilog implementation is shown below, it performs a multiply-and-add operation as the ARMA structure. The 5-bit input setpoint *r* requires a 3-bit serial shift to the left to form an 8-bit integer value, plus an additional multiplication by 2, which forms a 4-bit serial shift to the left.

From the 16-bit multiplication generated in c_{16}, only 8-bits c_8 must be fed back into the equation. To do this, only the 8 most significant bits of the product must be retained. However, since there is still a multiplication factor of 2, a 7-bit serial shift to the right is performed on each product. When the quantities are already 8 bits. Error e_8 can be calculated and the equation's variables can be updated.

Figure 7 shows the Verilog Vivado skyline response of the system with 8-bit integer arithmetic. The 1-bit clock input Clk and the variables Sel (3-bits) and Coef (8-bits) can be seen. The input in red is the same skyline as in Figure 5. The output in green follows the input but is restricted due to the size restrictions of the hardware where it is implemented.

```
r8 ← r«4
c16 =  (a1*2^6*e(k))>>7+(a3*2^6*e(k-2))>>7+(a4*2^6*e(k-3))>>7+
       (b2*2^6*c(k-1))>>7+(b3*2^6*c(k-2))>>7;
c8=c16[7:0];
e8 = r8-c8;            //error
//Variable update
e(k)=e8;    c(k-1)=temp;
```

The results demonstrate that the PID controller designed via ARMA formulation performs effectively under simulation and through hardware constraints. The system achieves the desired transient response, with a settling time of 0.068 seconds and an overshoot of 16%, closely aligning with design specifications. Truncation error represents quantization noise and degrades the signal-to-noise ratio, as seen in the first step of Figure 7 .The use of fixed-point arithmetic and the efficient use of Verilog enable the controller's feasibility for implementation in resource-limited ASIC environments.

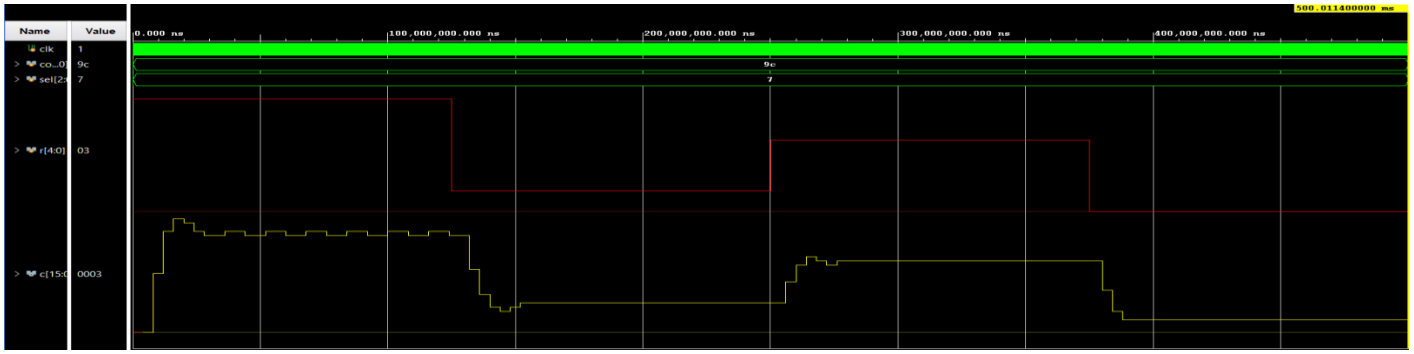

Figure 7. System Response in Verilog.

5. CONCLUSIONS

In this work, a continuous-time second-order plant was successfully discretized and controlled using a PID algorithm. The control system was designed in MATLAB and Simulink, then implemented in Verilog using fixed-point arithmetic under the Vivado environment. The digital controller was evaluated under hardware constraints imposed by the Silicluster platform. Despite resource limitations—including word-length restrictions and limited logic gate availability—the control system maintained functional equivalence with the original simulation model. Step and skyline input tests confirmed satisfactory dynamic response, with settling time and overshoot close to the desired specifications. These results validate the feasibility of implementing ARMA-based digital PID control in resource-constrained environments. The proposed design is thus ready for ASIC integration under open-source fabrication initiatives such as Silicluster. This simulation is important because it will eventually allow for an ASIC chip that performs specialized control tasks, optimizing resources for this purpose, since general-purpose microprocessors are mostly used for control tasks.

ACKNOWLEDGEMENTS

We acknowledge the Load Trail company and the Graduate School of the Cuauhtémoc City, Institute of Technology by its support to this research.

REFERENCES

[1] J. Kulisz, F.A. Jokiel. (2024). Hardware Implementation of the PID Algorithm Using Floating-Point Arithmetic. Electronics, 13, 1598. https://doi.org/10.3390/electronics13081598

[2] J. Wang, M. Li, W. Jiang, Y. Huang, R.A. Lin. (2022). Design of FPGA-Based Neural Network PID Controller for Motion Control System. Sensors 22, 889. https://doi.org/10.3390/s22030889

[3] V. Dubreuil and A.V. Osintsev. (2019). Designing Multiple PID Controllers Based on an FPGA for Controlling the Temperature of TEM-cell Surfaces. International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). Novosibirsk, Russia, pp. 0194-0198, doi: 10.1109/SIBIRCON48586.2019.8958396.

[4] S.S. Chauhan, C. Paramasivam and V. P. Meena. (2024). FPGA IP Core for DC Motor Control With Adaptive Neural Network PID Tuning, and High-Resolution Encoder Interface. IEEE Access, vol. 12, pp. 169356-169369, doi: 10.1109/ACCESS.2024.3491995.

[5] P. Jain, S. Bhat and V. Ryali. (2024). Digital twin based PID control modeling, design, and development for FPGA implementation. International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE). Bangalore, India, pp. 1-7, doi: 10.1109/IITCEE59897.2024.10467314.

[6] V. Radhika, K. Srinivasan, B.B. Sharmila, (2021). Design of digital pulse width modulator architecture with digital PID controller for DC-DC converter using FPGA. Analog Integr Circ Sig Process 107, 299–307. https://doi.org/10.1007/s10470-020-01794-8

[7] J. Liu, M. Liu, D. Pei and H. Sun. (2019). FPGA Implementation of Family Service Robot Based on Neural Network PID Motion Control System. International Conference on Electronic Engineering and Informatics (EEI). Nanjing, China, pp. 304-308, doi: 10.1109/EEI48997.2019.00073.

[8] D. Bushuev. (2022). PID Control of Motor Using FPGA. Technology and Communication. Vaasan Ammattikorkeakoulu University of Applied Sciences.

[9] R. Pradhan, S. Jena, S. Ojha and T. K. Sahoo. (2024) Integration of VLSI with Traditional PID Controller for Temperature Control in Electronic Devices. IEEE 1st International Conference on Advances in Signal Processing, Power, Communication, and Computing (ASPCC). Bhubaneswar, India, pp. 187-192. doi:10.1109/ASPCC62191.2024.10881846.

[10] G.M. Sung, P.Y. Chiang and Y.Y. Tsai. (2021). Predictive Direct Torque Control ASIC with Fuzzy Voltage Vector Control and Neural Network PID Speed Controller. IEEE International Future Energy Electronics Conference (IFEEC), Taipei, Taiwan, pp. 1-5. doi: 10.1109/IFEEC53238.2021.9661986.

[11] M. Liu, H. Zhang, Y. Zhang and C. Yuan. (2022). Design and Performance Analysis of ZYNQ Based Incremental PID-PWM Controller. IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, pp. 1123-1126, doi: 10.1109/EEBDA53927.2022.9744964.

[12] F. Caisheng Fan, L. Xue-Dong, L. Feiyu, L. Yuxin. (2022). Realization of Fuzzy PID Controller on FPGA for Source Measurement Unit. ICITEE '22: Proceedings of the 5th International Conference on Information Technologies and Electrical Engineering. Pages 445-451. https://doi.org/10.1145/3582935.3583010

[13] Y. Liang and X. Liu. (2023). A Method for Improving Transient Response of the Source Measure Unit Based on PID+LPF Controller. IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), Shenyang, China, pp. 748-751. doi: 10.1109/ICPECA56706.2023.10076250.

[14] D.T. Schussheim, K. Gibble. (2023). A many-channel FPGA control system Available to Purchase. Rev. Sci. Instrum. 94, 085101. https://doi.org/10.1063/5.0157330

[15] Phillips C.L., Troy Nagle H., Chakrabortty A. (2014). Digital Control System Analysis and Design. Ed. Prentice Hall Inc. Fourth Edition.

[16] Phillips C.L., Parr J.M. (2011). Feedback Control Systems. Ed. Prentice Hall 4th. Edition. New Jersey, USA