Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2008, Article ID 634306, 9 pages
doi:10.1155/2008/634306

Research Article

Design of a Mathematical Unit in FPGA for the Implementation of the Control of a Magnetic Levitation System

Juan José Raygoza-Panduro, Susana Ortega-Cisneros, Jorge Rivera, and Alberto de la Mora

Departamento de Electrónica, Centro Universitario de Ciencias Exactas e Ingenierías (CUCEI), Universidad de Guadalajara, Boulevard Marcelino García Barragan 1421, Guadalajara, Jal. 44430, Mexico

Correspondence should be addressed to Juan José Raygoza-Panduro, juan.raygoza@cucei.udg.mx

Received 2 July 2008; Revised 9 October 2008; Accepted 30 October 2008

Recommended by Gustavo Sutter

This paper presents the design and implementation of an automatically generated mathematical unit, from a program developed in Java that describes the VHDL circuit, ready to be synthesized with the Xilinx ISE tool. The core contains diverse complex operations such as mathematical functions including sine and cosine, among others. The proposed unit is used to synthesize a sliding mode controller for a magnetic levitation system. This kind of systems is used in industrial applications requiring high level of mathematical calculations in small time periods. The core is designed to calculate trigonometric and arithmetic operations in such a way that each function is performed in a clock cycle. In this paper, the results of the mathematical core are shown in terms of implementation, utilization, and application to control a magnetic levitation system.

1. Introduction

Mathematical control equations in an FPGA reconfigurable device is an important aspect in the design of arithmetic blocks when implementing control algorithms [1]. A well-known method utilized in the implementation of arithmetic operations in FPGAs is based upon the coordinate rotation digital computer (CORDIC) algorithm [2–6] which has become the standard solution for the implementation of complex operations in FPGAs.

This paper proposes the design of a mathematical unit dedicated to the implementation of control algorithms that involve several sequences of complex mathematical functions calculations.

Traditionally, the development of complex arithmetic functions in FPGA devices has resulted in difficulties to implement such operations. Therefore, the elaboration of mathematical operations in Xilinx FPGAs is proposed through the core generator [7]. The objective of this paper is to explain the development of a core capable of performing mathematical operations such as trigonometric functions in a clock cycle, using an alternative method of the core generator suggested by the manufacturer.

In order to construct such cores, the architecture of the mathematical unit is established by the user with Java software, in which the input and output parameters are defined as well as the functions needed to perform the desired control algorithm. This tool facilitates the users' implementation of mathematical blocks in FPGAs, simplifying the flow design to the adjustment of the interconnection of the required blocks in the main program described in VHDL. This reduces the designer's workload during the implementation stage of control algorithms. The tool is capable of implementing 16 different types of mathematical functions which may be described according to the required algorithm. The maximum number of functions that can be implemented depends on the available resources of the FPGA.

When the VHDL code generator is activated, a window initially appears, asking the characteristic of the input-output variables. The longitude of the input data indicating integer and decimal bits must be specified. At this point, selection of the functions to be implemented according to the control algorithm is made, and finally the code generator creates a file containing the description of each block in VHDL language ready to be synthesized

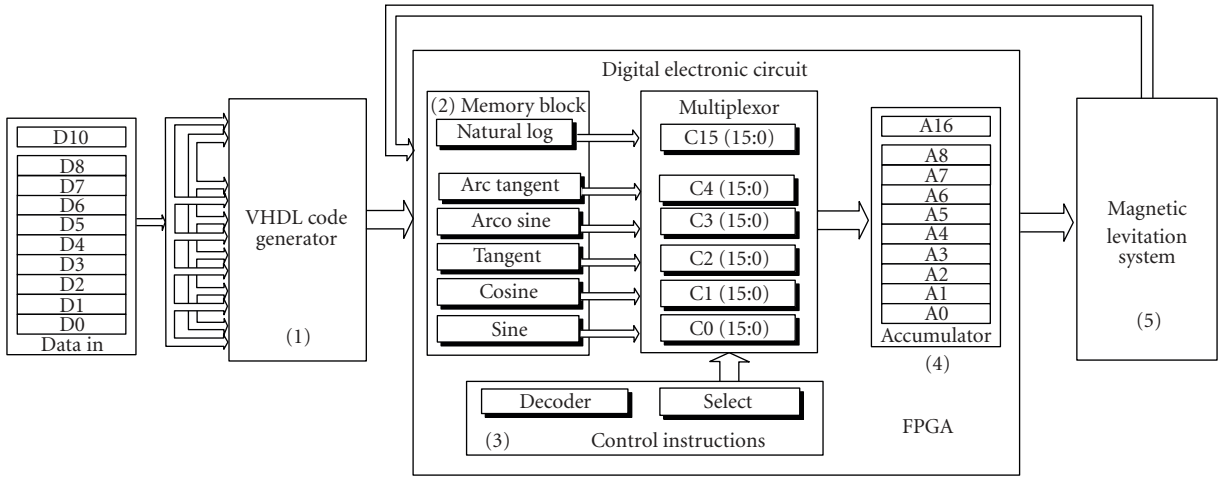

Figure 1: Block diagram of the FPGA control.

by the Xilinx ISE tool [8]. Each function might be an independent module that can be interconnected with the rest of the blocks in order to represent the equations that describe the desired algorithm. Trigonometric functions are implemented in the embedded memory of the FPGA. The advantage of solving complex functions with preloaded tables can be clearly seen in computing time, simplifying the execution of a mathematical function to the transfer of data from memory to the accumulator register.

The control algorithm of a magnetic elevation system is presented in order to provide an implementation study for the proposed mathematical unit. This system deals with the levitation of steel objects aided by a controlled electromagnetic force that is equal and opposed to the gravitational force acting on the steel object. This type of control is actually applied in commercial magnetic levitation (MAGLEV) trains [9].

2. Description of the Mathematical Unit

The mathematical unit has been developed with a Java program that generates blocks of mathematical functions in VHDL. The complete system is composed of 5 main modules, as shown in Figure 1, (1) VHDL code generator, (2) RAM or ROM memory block for mathematical operations, (3) control unit for instructions, (4) accumulator registers for results, and (5) magnetic levitation system.

The mathematical unit was functionally designed in VHDL code with instantiation of RAM or ROM memories that were created through the program generator functions, elaborated in Java language, especially for this job which is described in Section 2.1. The memories were programmed with input parameters assigned by the user, allowing the data input to have a suitable format according to the designers needs.

2.1. VHDL Code Generator

As previously mentioned, the proposed mathematical unit is capable of solving trigonometric functions in a clock cycle by using preestablished data tables. To accomplish this, a program was developed in Java language that calculates the values of the trigonometric or mathematical functions within the range of values defined by the user, followed by the creation of tables with the calculated values, and uses a RAM or ROM memory to store these data values and then to translate them into the description hardware language VHDL. The program defines the architecture, entity, and process which automatically adds to the libraries, reducing user time and the definition of each block to only the definition of ranges and precise values of input and output data in its integer and decimal parts.

The software function generator reduces the computational burden to the FPGA by using a standard computer to calculate the possible results of mathematical functions that require only one parameter in the instantiation of a RAM or ROM memory.

The program creates the desired function as an entity in VHDL with an input and an output of the selected size. The VHDL has syntax standards, which are contained in the libraries. The program generates the necessary lines for use by the corresponding libraries. The entity block is also created at the same time, along with the input data, ready to be synthesized by the Xilinx ISE simulator. The list of mathematical function values is calculated with the program code generator.

An example is shown in Table 1. This table corresponds to the calculation of the cosine function, which is implemented in a ROM memory of 16 bits $\times$ 1024 lines. The address bus is identified with the letters a9 to a0, where a0 is the least significant bit. Before executing the program generating code, the data format specifies the required bits for the integer part and the decimal part.

Table 1: Selection of preestablished data for a cosine function.

Address bus				Data bus			
a9-a8	a7-a4	a3-a0	d16	d15-d12	d11-d8	d7-d4	d3-d0
00	0000	0000	0	1000	0000	0000	0000
00	0000	0001	0	0111	1111	1111	1110
00	0000	0010	0	0111	1111	1111	1110
00	0000	0011	0	0111	1111	1111	1010
00	0000	0100	0	0111	1111	1111	1000
00	0000	0101	0	0111	1111	1111	0010
00	0000	0110	0	0111	1111	1110	1110
00	0000	0111	0	0111	1111	1110	0110
00	0000	1000	0	0111	1111	1110	0000
00	0000	1001	0	0111	1111	1101	0110
00	0000	1010	0	0111	1111	1100	1110
00	0000	1011	0	0111	1111	1100	0010

The value of the angle is defined in radians at an interval from 0 to 3.99. In the example in Table 2, this quantity may be defined by the user in the program generator. The calculation of the cosine function is made considering the bits from a6 to a0 as the decimals of the parameter and the bits from a9 to a7 as the integer part. The result of the function is located in the data bus where d0 is the sign bit, d1 to d13 is the decimal part, and from d14 to d16 is the integer part of the data.

The following program code fragment is an example of the result of the VHDL mathematical functions, where numbers 9 and 16 are the defining entrance parameters that were programmed:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_Logic_UNSIGNED.ALL;
entity block is
Port(angle:in std_logic_vector(9 downto 0);
     result:out std_logic_vector(16 downto 0));
end block;
architecture behavior of block is
type func is array (0 to 1024) of std_logic_vector
(16 downto 0);
constant Content: func:=(
B"00000000000100000,"
B"00000000000100000,"
B"00000000000100000,"
B"00000000000100000,"
```

The critical functions programmed in C language turned a floating chain of bits as well as the same operation in inverse form. An example of the code follows:

```
acadena(Number_to_turning, chain_of_exit,
        decimal_of_exit, size_of_exit)
adouble(Chain_to_turning,Number_of_decimal)
acadena(15.25,chain_of_exit,2,6) // chain_of_exit will
have the value of 111101
acadena(15.5,chain_of_exit,2,6) // chain_of_exit will
have the value of 111110
acadena(10.5,chain_of_exit,2,6) // chain_of_exit will
have the value of 101010
acadena(8.75,chain_of_exit,2,10) // chain_of_exit will
have the value of 0000100011
adouble("100011,"2);      // The result is 8.75
adouble("100011,"1);      // The result is 17.5
adouble("100011,"0);      // The result is 35
```

In the program, the "acadena" function transformed the floating value of bits and the "adouble" function converted a floating value of bits. A part of the second version which was generated in Java language follows:

```
import java.io.*;
#1   class seno
#2   {
#3      public static String acadena(double X,
        int enteros,int longitud){
#4          double Y=0.0;
#5      if (X<0)
#6      {
#7          Y=Math.abs(Math.ceil(X));
#8      }else{
#9          Y=Math.abs(Math.floor(X));
#10     }
```

Table 2: Results obtained from the mathematical unit.

Angle degrees	Angle radians	Math unit result	Matlab result	Error obtained
0.00000	0.00000	1.00000	1.00000	0.00000
9.84771	0.17188	0.98527	0.98510	0.00017
14.77157	0.25781	0.96695	0.96692	0.00003
19.69542	0.34375	0.94150	0.94141	0.00009
24.61928	0.42969	0.90910	0.90906	0.00004
29.99076	0.52344	0.86611	0.86609	0.00002
34.91462	0.60938	0.82001	0.81995	0.00006
39.83847	0.69531	0.76785	0.76782	0.00003
95.79138	1.67188	−.10091	−.10083	0.00008
210.38294	3.67188	−.86266	−.86255	0.00012

In order to complete the conversion of the floating value to chain of bits, we followed a 2-stage process; firstly, the whole part becomes a chain of bits, and after the part decimal is turned into a chain of bits. Later they are united in a single decimal number in binary code.

The conversion process starts with the whole part of the function; this requires rounding the smallest number (when positive) or rounding the largest number (when negative). Using the "ceil" function one can obtain the rounding of the number and using the "floor" function one can round the whole part. Since the conversion algorithm uses positive numbers, the "abs" function is used to take the absolute value from the rounded number. The variable "res" keeps the final result from the conversion.

The code generator program allows the usage of RAM or ROM memories and selection of these will depend on the application required. For example, when using ROM memories, these are implemented with the internal resources of the FPGA augmenting the utilization of the circuit; the flexibility of using these memories is their facility to adjust the size of the word and required address for the precise calculations that will be stored in them. When RAM memories are selected, as these are embedded, they do not impact the available resources in the FPGA, allowing a huge logic capacity for other circuit implementation, the disadvantage that it is limited to the implementation of variable arrays in the word longitude and address bus.

With the objective of observing the units behavior during the calculation of different trigonometric functions, a sequence of operations was established for the resolution of the functions with different angles. The obtained results are shown in Table 2. The first column corresponds to the evaluation angles; the second column is equivalent to the first column in radians; the third column shows the results of the cosine function obtained with the mathematical unit presented; the fourth column has the results obtained with Matlab; the last column presents the difference between the value calculated with Matlab and the value obtained with the mathematical unit.

It is important to emphasize that the mathematical function sequence can be carried out to form complete equations which are calculated and stored in a ROM or RAM memory, to be used later in the implementation of individual block control equations, that are capable of being calculated in a clock pulse, optimizing the calculation time.

Table 3: Utilization of the mathematical unit blocks.

Function	Utilization			
	Sel.	Slices	LUTs	TEGs
Cosine	0	11%	8%	6 121
Sine	1	11%	8%	6 282
Square root	2	1%	1%	50
Tangent	3	8%	6%	5 230
Arc cosine	4	3%	3%	2 799
Arc sine	5	3%	3%	2 807
Arc tangent	6	3%	3%	2 952
Exponential	7	5%	4%	3 956
Radians	8	4%	3%	3 303
Hyperbolic tangent	9	2%	1%	1 571
Hyperbolic cosine	10	5%	4%	4 375
Hyperbolic sine	11	5%	4%	4 391
Natural log	12	1%	1%	80
Inverse	13	1%	1%	1 345
Log base 10	14	1%	1%	671
Degrees	15	3%	3%	3 082

2.2. Description of the Mathematical Unit Operation

The mathematical unit was implemented in a FPGA Virtex II. The results of the utilization are shown in Table 3. The utilization of slices, LUTs, and total equivalent gate (TEG) is presented in independent columns. The column "sel" refers to the instruction code that mathematical unit executes. This makes 16 trigonometric and mathematical functions which may be selected through a control word of 4 bits.

The total circuit utilization is 95% of the available slices in the FPGA and 74% of LUTs, being equivalent in TEG to 58 157 out of 1 000 000 of the available total on the Xilinx Virtex II.

3. Application to the Control of a Magnetic Levitation System

In order to prove the capacities of the mathematical unit, a sliding mode controller [10] was used to regulate a magnetic levitation system. This type of system is used in several applications such as frictionless bearings [11], high-speed MAGLEV passenger trains [12], wind tunnel levitation models [13], molten metal levitation [14], and the levitation of metal slabs during industrial manufacturing process [15]. These systems have natural unstable nonlinear dynamics requiring closed-loop control designs for stabilization. Several control techniques have been applied to the stabilization of MAGLEV systems, such as I/O linearization [16, 17], backstepping [18], and sliding mode control [19], among others. The sliding mode control [10] has been extensively used in electromechanical systems due to its robustness to unknown bounded perturbations. Another characteristic of sliding mode control is the discontinuous nature of its control signal which switches from two states. This is an advantage because it avoids using pulse width modulation (PWM). The drawback of sliding mode control is that the switching signal has an infinite frequency and when implemented with common switching power devices with a frequency around 20 KHz, produces an output phenomenon called chattering; small oscillations around the set-point. Nowadays, there are power devices available with a switching frequency of at least 150 KHz, which common digital signal processor boards cannot support. To take full advantage of such switching devices, one needs high speed digital media such as FPGAs that can support and match high switching frequencies. In this case, the chattering problem is considerably reduced.

3.1. Mathematical Model and Problem Formulation for the MAGLEV System

Figure 2 shows an schematic diagram of a MAGLEV system.

The mathematical model of the MAGLEV system is given in the following equations [17]:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= g - \frac{k_m}{M}\frac{x_3^2}{x_1^2}, \\ \dot{x}_3 &= -\frac{R}{L}x_3 + \frac{1}{L}v, \\ y &= x_1 \end{aligned} \tag{1}$$

with state vector defined as $x = (x_1, x_2, x_3)^T$, where x_1 represents the position of the steel ball of mass M which is positively increasing in the downward position, x_2 is the velocity of the steel ball, x_3 is the current through the coil, v is the input voltage applied to the coil, and y is the output of the system. The constant parameters are the resistance of the coil denoted by R, the inductance denoted by L, g which is the gravitational constant and is considered as a known perturbation term, finally k_m which is the magnetic constant of the electromagnet.

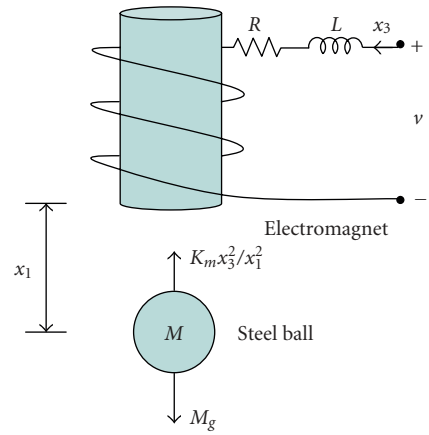

Figure 2: Block diagram of the FPGA control.

The control problem is based upon forcing the output $y = x_1$ to track a reference signal $q(w)$. Therefore, one can consider the following output tracking error:

$$e = x_1 - q(w). \tag{2}$$

3.2. Sliding Mode Output Regulation for the MAGLEV System

The applied control design methodology is a combination of two important control techniques, output regulation theory (ORT) [20] and sliding mode control (SMC) [10]. The advantage of using ORT is that it plays an important role in trajectory output tracking and in the rejection of known disturbances. ORT deals with the problem of finding a control law such that the output of the controlled system can asymptotically track a signal generated by an exosystem and at the same time reject perturbations possible generated by the same exosystem. The nature of the control signal is continuous or smooth, and in this case PWM is required for implementation. When ORT is combined with SMC one obtains a control methodology commonly known as sliding mode output regulation (SMOR) [10] resulting in robust protection against unknown perturbations and avoids the use of PWM as just mentioned before.

The exosystem is proposed as follows:

$$\begin{aligned} \dot{w}_1 &= -\alpha w_2, \\ \dot{w}_2 &= \alpha w_1, \\ \dot{w}_3 &= 0, \\ \dot{w}_4 &= 0 \end{aligned} \tag{3}$$

with initial conditions $w_1(0) = w_2(0) = a$, $w_3(0) = b$, and $w_4(0) = c$, such that, the exosystem generates a reference output tracking signal for an MAGLEV system, which is chosen as $q(w) = w_1 + w_3$, that is, a sinusoidal shape signal with frequency α, peak value of $\sqrt{2}a$, and a dc bias value b.

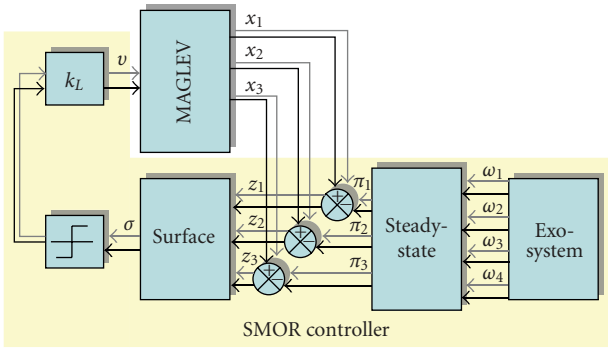

Figure 3: Schematic block diagram of SMOR.

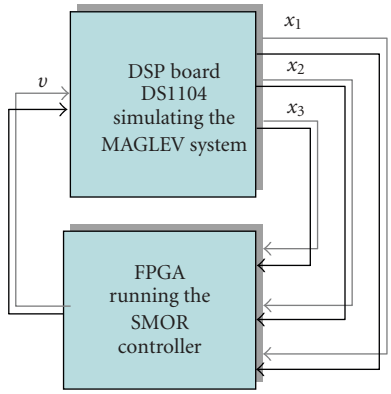

Figure 4: Schematic block diagram of HIL simulation.

The reference signal is chosen in this way in order to test some trigonometric functions of the mathematical unit. In this case, the steel ball will move upward and downward as dictated by the amplitude and frequency of the reference signal.

What follows is the ideal steady state of operation for the MAGLEV system, that is, $\pi(w) = (\pi_1(w), \pi_2(w), \pi_3(w))^T$;

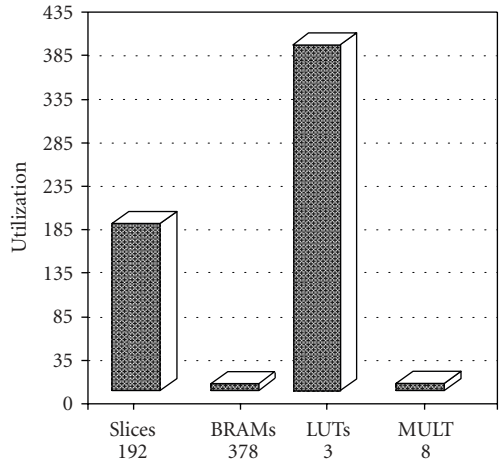

Figure 5: Utilization of the components in the FPGA Virtex II.

this state is such that, if the original states of the MAGLEV, $x = (x_1, x_2, x_3)^T$, are driven to the ideal steady-state, then the output tracking error will asymptotically decay to zero, accomplishing the control objective. In order to find the steady state of operation one must solve the well-known Francis-Isidori-Byrnes [20] equations. In the case of the MAGLEV system results are as follows:

$$\frac{\partial \pi_1(w)}{\partial w} s(w) = \pi_2(w), \tag{4}$$

$$\frac{\partial \pi_2(w)}{\partial w} s(w) = d(w) - \frac{k_m}{M} \frac{\pi_3^2(w)}{\pi_1^2(w)}, \tag{5}$$

$$\begin{aligned} \frac{\partial \pi_3(w)}{\partial w} s(w) &= -\frac{R}{L}\pi_3(w) + \frac{1}{L}c(w), \\ 0 &= \pi_1(w) - q(w) \end{aligned} \tag{6}$$

with $s(w) = (-\alpha w_2, \alpha w_1, 0, 0)^T$. Note that the ideal steady-state value for e is obviously zero. Using this fact, one easily calculates from (6) $\pi_1(w) = w_1 + w_3$, replacing $\pi_1(w)$ in (4) one finds that $\pi_2(w) = -\alpha w_2$. Substituting $\pi_2(w)$ in (5), one reckons the expression for $\pi_3(w)$ as $\pi_3(w) = (w_1 + w_3)\sqrt{(M/k_m)(w_4 + \alpha^2 w_1)}$. The $c(w)$ variable represents the steady-state value for the control input v, but it is not neccesary to calculate such expression when using SMC actions. Let us define the steady-state error as

$$\begin{aligned} z &= (x - \pi(w)) = (z_1 \;\; z_2 \;\; z_3)^T \\ &= (x_1 - \pi_1(w) \;\; x_2 - \pi_2(w) \;\; x_3 - \pi_3(w))^T. \end{aligned} \tag{7}$$

The dynamic equation for (7) with tracking error e (2) can be obtained from (1) as

$$\dot{z}_1 = z_2 + \pi_2(w) - \frac{\partial \pi_1(w)}{\partial w} s(w), \tag{8}$$

$$\dot{z}_2 = d(w) - \frac{k_m}{M} \frac{(z_3 + \pi_3(w))^2}{(z_1 + \pi_1(w))^2} - \frac{\partial \pi_2(w)}{\partial w} s(w), \tag{9}$$

$$\begin{aligned} \dot{z}_3 &= -\frac{R}{L}(z_3 + \pi_3(w)) + \frac{1}{L}u - \frac{\partial \pi_3(w)}{\partial w} s(w), \\ e &= z_1 + \pi_1 - q(w). \end{aligned} \tag{10}$$

Now, one defines the sliding function and control as

$$u = -kL\,\mathrm{sign}(\sigma), \quad \sigma = z_3 + \Sigma_1 z^1, \; k > 0, \tag{11}$$

where sign is the typical signum function, with $\Sigma_1 = (\Sigma_{1,1}\Sigma_{1,2})$ and $z^1 = (z_1, z_2)^T$.

Making use of a rigorous stability analysis by means of a Lyapunov function [10], one finds a stability condition for gain k:

$$k > \left| \left(\frac{1}{L}\right) v_{eq}(z, w) \right|, \tag{12}$$

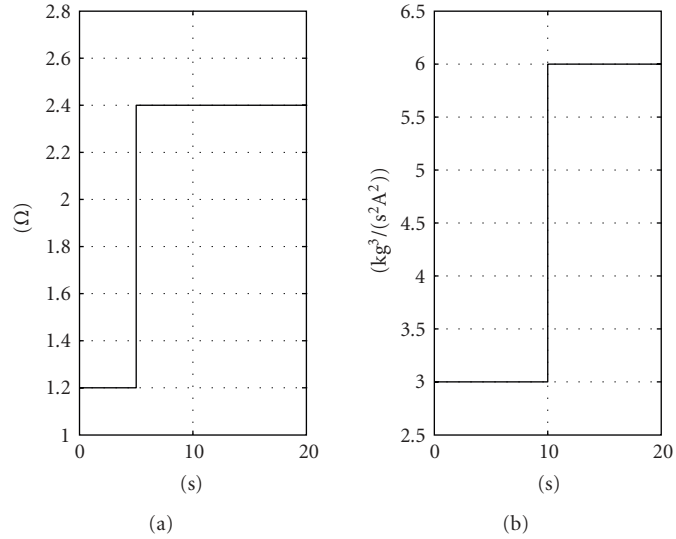

Figure 6: (a) Resistance variation, (b) magnetic constant variation.

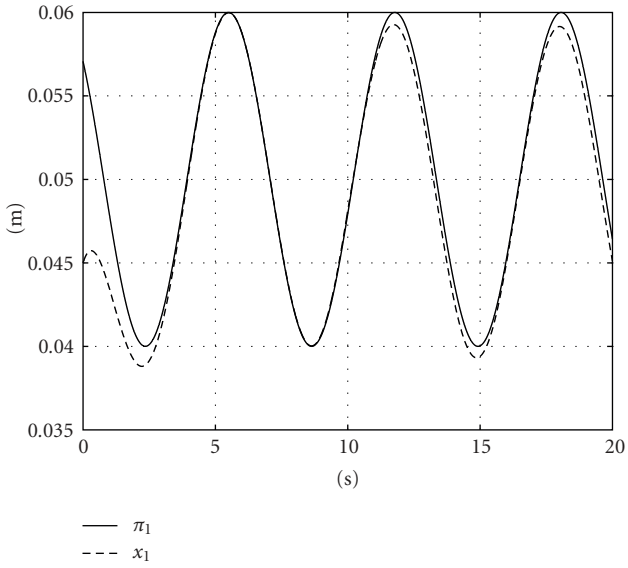

Figure 7: Output tracking signal.

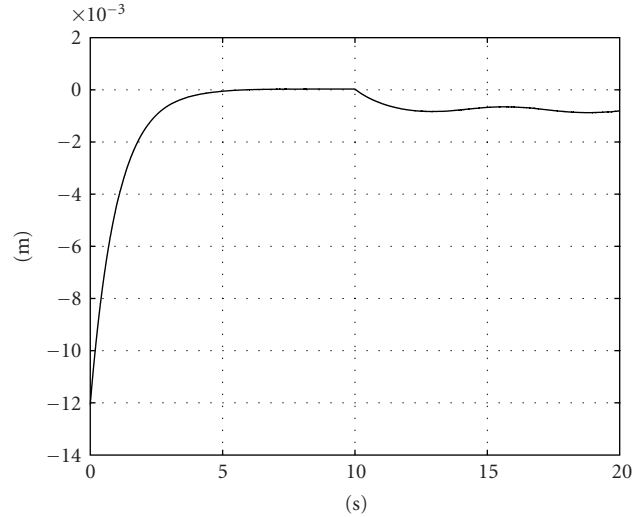

Figure 8: Output error signal.

where $v_{eq}(z, w)$ is a solution of $\dot{\sigma} = 0$, namely,

$$v_{eq} = R(z_3 + \pi_3) + L\left(\frac{\partial \pi_3}{\partial \omega}\right)s(\omega) - L\sum_{1,1}\left(z_2 + \pi_2 - \left(\frac{\partial \pi_1}{\partial \omega}\right)s(\omega)\right) - L\sum_{1,2}\left(d(\omega) - \left(\frac{k_m}{M}\right)\frac{(z_3 + \pi_3)^2}{(z_1 + \pi_1)^2} - \left(\frac{\partial \pi_2}{\partial \omega}\right)s(\omega)\right). \tag{13}$$

If condition (12) is satisfied then $\sigma = 0$ is guaranteed, implying that z_3 can be calculated from (11) as

$$z_3 = -\Sigma_1 z^1. \tag{14}$$

That is, the differential equation (10) is unnecessary as its solution (14) is now known. The remaining differential equations for z_1 and z_2 are obtained by replacing (14) in (8) and (9). This residual dynamic is known as the sliding mode dynamic. This dynamic is made stable by the proper choice of Σ_1. An easy way to stabilize the sliding mode dynamic is by using its linear approximation at the origin as shown here:

$$\begin{aligned} \dot{z}^1 &= (A_{11} - A_{12}\Sigma_1)z^1 + \text{H.O.T}, \\ \dot{w} &= Sw, \\ e &= z_1 + \pi_1(w) - q(w) \end{aligned} \tag{15}$$

with A_{11} and A_{12} being as proper dimensions matrices obtained from linear approximation, and where H.O.T. stands for higher-order terms, that vanish at the origin. Now, Σ_1 is chosen so that the matrix $(A_{11} - A_{12}\Sigma_1)$ is Hurwitz or has

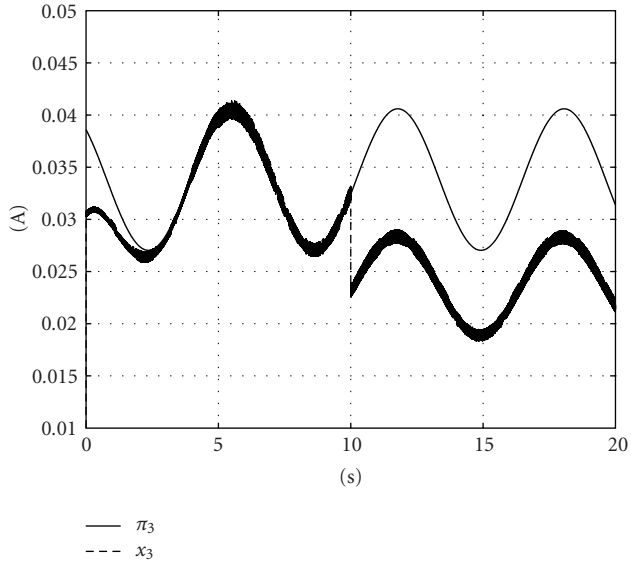

Figure 9: Current signals.

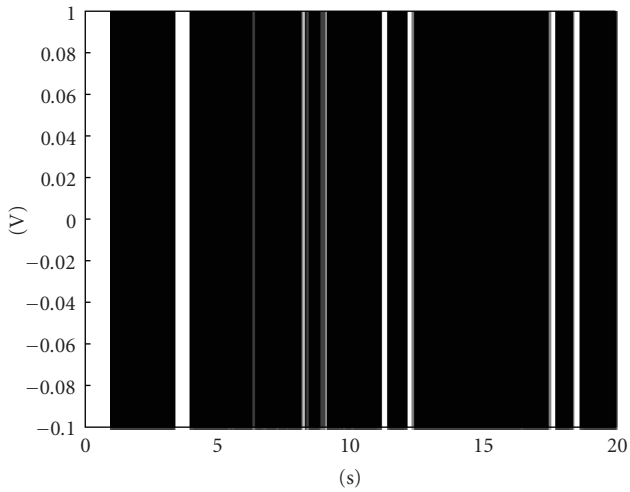

Figure 10: Voltage control signal.

negative real part poles. In this case $\lim_{t\to\infty} z^1(t) = 0$ and as a consequence by (14) z_3 tends to zero too. By continuity, using $\pi_1(w) = w_1 + w_3$ one finally finds that the output tracking error e asymptotically tends to zero, satisfying the control objective. Finally a closed-loop block diagram is presented in Figure 3.

4. Control Algorithm Implementation Results

The control algorithm was tested using an FPGA virtexII XE2V1000-4fg256, and the plant dynamic was simulated using the DSP board DS1104 from DSPACE. This type of simulation is known as hardware-in-the-Loop (HIL) simulation [21]. HIL simulation is a real-time simulation form. It differs from real-time simulation by the addition of a hardware component in the loop as an FPGA. This technique is increasingly being used in the development and testing of complex real-time embedded systems. Moreover, the complexity of the plant dynamic under control is commonly simulated in a graphical environment as SIMULINK from Matlab. In our case the plant dynamic was created in SIMULINK and then downloaded to the DSP board DS1104 in order to arrange the I/O ports. Figure 4 shows a simple diagram of the HIL simulation that was performed.

4.1. FPGA Implementation Results

The system is declared as an entity of three inputs that represents the position of the ball x_1, the velocity of the ball x_2, the current through the coil x_3, and the output voltage v closed loop with the MAGLEV system. The internal variables used for the calculation of the equations use a word of 32 longitude bits—15 bits to represent the integer part, 16 bits for the decimal part, and 1 bit to represent the sign. The variable v corresponds to the final calculation of the system and has a word longitude of 64 bits—4 for integers, 1 for signs, and 59 for decimals, providing the necessary accuracy for the stability of the system. The total processing time of the calculations of one cycle in the FPGA is 202 nanoseconds, representing maximum processing speed of up to 21 nanoseconds. Figure 5 shows the utilization of the components in the FPGA virtexII XE2V1000-4fg256, with 3% of slices, 3% of LUTs, 7% of RAMs, and 20% of multipliers. The device has sufficient resources available to implement additional circuits.

4.2. Closed-Loop System in Implementation Results

The nominal parameters of the MAGLEV system are $k_m = 3\,\text{kgm}^3/\text{s}^2\text{A}^2$, $M = 0.14\,\text{kg}$, $g = 9.8\,\text{m/s}^2$, $R = 1.2\,\Omega$, $L = 1 \times 10^{-3}\,\text{H}$. The constant values of the exogenous signals (2) are $a = 0.0070716$, $b = 0.05\,\text{m}$, and $c = 9.8$. Taking the nominal parameters of the MAGLEV system, the following pairs of matrices are calculated:

$$A_{11} = 013920, \qquad A_{12} = 0 - 579.6551. \tag{16}$$

The control parameters that appear in (11) are as follows:

$$k = 100, \qquad \Sigma_{1,1} = -12.9423, \qquad \Sigma_{1,2} = -12.2492. \tag{17}$$

The matrix Σ_1 in (11) is calculated using the LQR function provided in Matlab.

To verify the robustness properties, some plant parameter variations are introduced which can be seen in Figure 6, where R and k_m may change up to 100% from their nominal values. It is worth to mention that the perturbation term generated by the variation of R satisfies the matching condition [10], but not the variations on k_m.

Figure 7 shows the tracking of the output signal where can be appreciated a good performance for $0 \le t < 5$. But for $5 \le t < 10$ where the perturbation term due to the variation in R is present, and the output still performs well due to the matching conditions. Finally, the unmatched perturbation

term due to the variation of k_m, appearing at $t \geq 10$ adversely affects the MAGLEV system but the output still performs well.

Figure 8 shows the output tracking error where can be appreciated the transient and steady-state responses can be observed.

Figure 9 shows π_3, which represents the ideal steady-state behavior of the current. It can be seen that the current becomes different to π_3 for $t \geq 10$ due to the unmatched perturbation.

Finally, Figure 10 shows the voltage input signals where the discontinuous nature of the control signal can be appreciated. The main advantage of having discontinuous control signals is that it avoids the use of PWM as mentioned in [10], therefore, facilitating a straightforward implementation of the control action.

5. Conclusion

This work has presented the results of a program generator for VHDL code developed in Java language and designed to implement a mathematical unit prototyped and implemented in reconfigurable FPGA circuits from Xilinx. The mathematical unit was used to implement the control algorithm of a magnetic levitation system, accomplishing the requirements of speed and precision necessary to operate under nominal conditions. The code generator tool allows the implementation of blocks containing complex operations which may be grouped in the same memory, letting operations to run in a clock pulse, based on the calculation of functions through preestablished tables. Moreover, the HIL simulation test platformed has facilitated the verification of the results obtained when the physical plant is not available.

References

[1] S. Ortega-Cisneros, J. J. Raygoza-Panduro, J. Suardíaz Muro, and E. Boemo, "Rapid prototyping of a self-timed ALU with FPGAs," in *Proceedings of International Conference on Reconfigurable Computing and FPGAs (ReConFig '05)*, p. 8, Puebla City, Mexico, September 2005.

[2] J. S. Walther, "A unified algorithm for elemetary functions," in *Proceedings of the AFIP Spring Joint Computer Conference (SJCC '71)*, vol. 38, pp. 379–385, AFIPS Press, Montvale, NJ, USA, 1971.

[3] F. Cardells-Tormo and J. Valls-Coquillat, "Optimisation of direct digital frequency synthesisers based on CORDIC," *Electronics Letters*, vol. 37, no. 21, pp. 1278–1280, 2001.

[4] T. C. Chen, "Automatic computation of exponentials, logarithms, ratios and square roots," *IBM Journal of Research and Development*, vol. 16, no. 4, pp. 380–388, 1972.

[5] C.-S. Wu, A.-Y. Wu, and C.-H. Lin, "A high-performance/low-latency vector rotational CORDIC architecture based on extended elementary angle set and trellis-based searching schemes," *IEEE Transactions on Circuits and Systems II*, vol. 50, no. 9, pp. 589–601, 2003.

[6] H. Dawid and H. Meyr, "VLSI implementation of the CORDIC algorithm using redundant arithmetic," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '92)*, vol. 3, pp. 1089–1092, San Diego, Calif, USA, May 1992.

[7] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no. 3, pp. 330–334, 1959.

[8] Xilinx, 2007, http://www.xilinx.com.

[9] M. Ono, Y. Sakuma, H. Adachi, et al., "Train control characteristic and the function of the position detecting system at the Yamanasi Maglev test line," in *Proceedings of the 15th International Conference on Magnetically Levitated Systems and Linear Drives (MAGLEV '98)*, pp. 184–189, Mt. Fuji, Yamanashi, Japan, April 1998.

[10] V. I. Utkin, A. G. Loukianov, B. Castillo-Toledo, and J. Rivera, "Sliding mode regulator design," in *Variable Structure Systems: From Principles to Implementation*, A. Sabanovic, L. Fridman, and S. Spurgeon, Eds., p. 1943, IEE, London, UK, 2004.

[11] P. Allaire and A. Sinha, "Robust sliding mode control of a planar rigid rotor system on magnetic bearings," in *Proceedings of the 6th International Symposium on Magnetic Bearings (ISMB '98)*, pp. 577–586, Boston, Mass, USA, August 1998.

[12] Hyung-Woo Lee, Ki-Chan Kim, and Ju Lee, "Review of maglev train technologies," *in IEEE Transactions on Magnetics*, vol. 42, no. 7, pp. 1917–1925, 2006.

[13] R. J. M. Muscroft, D. B. Sims-Williams, and D. A. Cardwell, "The development of a passive magnetic levitation system for wind tunnel models," *SAE Transactions: Journal of Passenger Cars: Mechanical Systems*, vol. 115, no. 6, pp. 415–419, 2006.

[14] K. Im and Y. Mochimaru, "Numerical analysis on magnetic levitation of liquid metals, using a spectral finite difference scheme," *Journal of Computational Physics*, vol. 203, no. 1, pp. 112–128, 2005.

[15] B. V. Jayawant and D. P. Rea, "New electromagnetic suspension and its stabilization," *Proceedings of the Institution of Electrical Engineers*, vol. 115, no. 4, pp. 549–554, 1965.

[16] A. El Hajjaji and M. Ouladsine, "Modeling and nonlinear control of magnetic levitation systems," *IEEE Transactions on Industrial Electronics*, vol. 48, no. 4, pp. 831–838, 2001.

[17] W. Barie and J. Chiasson, "Linear and nonlinear state-space controllers for magnetic levitation," *International Journal of Systems Science*, vol. 27, no. 11, pp. 1153–1163, 1996.

[18] Z.-J. Yang, K. Kunitoshi, S. Kanae, and K. Wada, "Adaptive robust output-feedback control of a magnetic levitation system by K-filter approach," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 1, pp. 390–399, 2008.

[19] F.-J. Lin, L.-T. Teng, and P.-H. Shieh, "Intelligent sliding-mode control using RBFN for magnetic levitation system," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 3, pp. 1752–1762, 2007.

[20] A. Isidori and C. I. Byrnes, "Output regulation of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 35, no. 2, pp. 131–140, 1990.

[21] B. Lu, X. Wu, H. Figueroa, and A. Monti, "A low-cost real-time hardware-in-the-loop testing approach of power electronics controls," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 2, pp. 919–931, 2007.