EFFICIENT STOCHASTIC COMPUTING-BASED CIRCUITS FOR SERVOMOTOR CONTROLLERS

Nasrin Imanpour[1] and Sayed Ahmad Salehi[2]

[1]Department of Computer Science and Engineering,
University of South Carolina, Columbia, South Carolina, USA
[2]Department of Electrical and Computer Engineering,
University of Kentucky, Lexington, Kentucky, USA

ABSTRACT

Stochastic computing (SC) provides a fault-tolerant and low-cost alternative to conventional binary computing (BC). The capacity of SC to implement complex mathematical functions with simple logic gates creates a path toward the design of efficient hardware architectures. This paper presents a new methodology for the hardware implementation of servomotor controller using SC. We design SC circuits using both quadrature decoder and efficient decoder for implementing servo controller and compare them with traditional BC-based servo controller. The quadrature decoder requires more hardware resources than efficient decoder but can provide position information in PWM form. The FPGA implementation result shows that, compared to BC-based design, quadrature decoder-based design achieves 56.7% savings in area and 33.33% savings in power consumption, and efficient decoder-based design achieves 73.7% savings in area and 33.33% savings in power consumption.

KEYWORDS

Stochastic computing (SC), servomotor controller, quadrature decoder, efficient decoder, stochastic integrator.

1. INTRODUCTION

Servomotor is a rotary or linear actuator that is used for precise control of angular or linear position, velocity, and acceleration. They are used in many systems, including elevators [1], cameras [2], telescopes [3], robotics [4] and various industrial applications [5]. The major advantages of using servomotors are high torque to inertia ratio, accurate positioning accuracy, lightweight and compact design [6]. But they suffer from the disadvantages of requiring more complex drive circuits (controller) and positional encoder, which increase the cost and need of maintenance.

After the development of microprocessors, microprocessor-based servomotor controllers have become more popular than analog controllers [7]. Microprocessor-based controllers can provide a flexible skill but suffers from a long period of development and exhausts many hardware resources. In recent years, FPGA-based hardware implementation technology has proven to be more efficient for many applications including servo controller design [8]. FPGA is only a collection of standard cells which has field programmable characteristics and can reuse IP cores. Therefore, FPGAs can carry out parallel processing and the system can run at a very high speed [8]. In this paper, we discuss efficient design and implementation of servomotor controller on a single FPGA device using stochastic computing (SC).

David C. Wyld et al. (Eds): SIGV, AI & FL, SESBC, MLAEDU, DSCC, NLPTT, SCOM, SCM, CEEE - 2022
pp. 145-156, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.122212

SC is a probabilistic approach for performing computation with simple arithmetic units. The implementation of SC is compatible with modern VLSI design and enhances the ability of FPGA devices. Another advantage of stochastic computing is that its probabilistic feature makes it more tolerant to soft transient errors (such as bit-flips) than binary computing (BC). Because of its advantages, SC circuits have been used for different applications, such as VLSI implementation of deep neural networks [9,10], image processing [11], efficient signal processing architectures [12], and polynomial computation [13]. Although hardware implementation of servomotor controller using BC has been studied in prior works [8], in this paper, we employ SC for the implementation of an efficient servomotor controller. To show the efficiency of the proposed SC circuits, we compare them with traditional BC circuit for servomotor controller.

This paper is organized as follows. In Section 2, we provide some backgrounds on servo controller and SC. Then, in Section 3, we present our proposed work on the SC implementation of servo controller. Results and analysis from simulations are given in Section 4. In Section 5, we conclude the paper.

2. BACKGROUND

Before discussing our work on servo controller, it is necessary to give a brief background on the basics of servo controller and SC.

2.1. Servo Controller

A basic servo system consists of a suitable sensor coupled to a motor shaft to give feedback. A decoder is used to convert the feedback into direction, and position signal. These feedback signals are compared with the command signals and sends out the error signal to the PID controller section. Finally, the PID controller section provides an output voltage to produce required PWM signal, which is supplied to the servomotor. A brief description of each component is given below.

The sensor's output is proportional to the position of the servomotor shaft. The most common type of sensor for servomotor is an incremental encoder. This encoder is made up of a light-emitting diode (LED), a rotating encoder disk, and a light detector. When the disk rotates, the opaque segments of the disk block the light, while the clear segments allow the light to pass through. Thus square-wave like pulses are generated. But an encoder with only one set of pulses cannot indicate the direction of rotation. Therefore, two pulse trains are created which are 90 degree out of phase with each other. Fig. 1 shows an incremental encoder that creates two pulse trains A and B. For example, when the disk rotates in one direction, A leads B, and for the opposite direction, B leads A. As, the pulse trains A and B are 90 degree out of phase, they are known as quadrature signals.

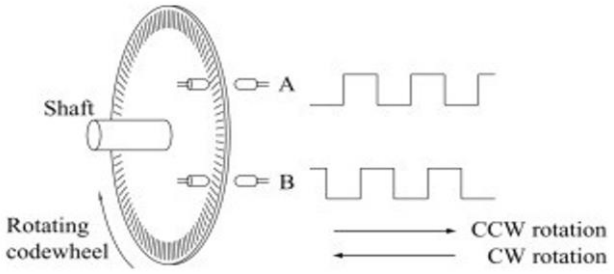

Fig. 1. Optical encoder with two pulse trains A and B [14]

A decoder is employed to

decipher the output signals (pulse train A and B) of the encoder and provide rotor shaft position and direction information. In the next section of this paper, we discuss our decoder implantation along with servomotor controller.

The PID controller is the most widely used algorithm for servomotor control. PID control circuit operation is based on position error. The PID is basically a composition of three individual parts, proportional (P), integral (I), and derivative (D) position loop. All these P, I, and D terms are summed up to provide the desired voltage command. Table I describes the behaviour of different component's output of PID controller in different part of operation. Fig. 2 shows a simple block diagram of PID controller.

We are considering a servomotor, which is pulse width modulation (PWM) controlled. Therefore, the input to the motor should be a PWM signal. To generate PWM signal from the output voltage, provided by PID controller, we need a PWM generator. PWM generator compares the PID output voltage with a sawtooth or triangular like waveform and outputs a PWM signal. Fig. 3 shows the overall block diagram of a servo controller.

2.2. Stochastic Computing (SC)

SC performs arithmetic operations in probabilistic nature. In SC, numbers are represented by random bitstreams of 1's and 0's. Over the last few years, research has been performed to interpret stochastic bitstreams in several ways, these are unipolar, bipolar, inverted-bipolar and ratio of 1's to 0's [15]. In unipolar format, the range of number is $x \in [0, 1]$. For example, in unipolar format, the value $x = 0.4$ can be represented as 0110100100, where 40% of the bits are one and the remainder are zero. In unipolar format, we cannot represent a negative value. Unipolar format can be mapped to extend the range of stochastic numbers [16]. We can map from $x \in [0, 1]$ to the range $y \in [-1, 1]$ using the function $Y = 2X - 1$. This is the called bipolar stochastic representation, and it can represent negative numbers in a natural way. For example, the same bitstream 0110100100 represents the value $y = -0.2$ in bipolar SC. Again, if we want to represent a very large value (> 1), we can use ratio of 1's to 0's, which has a number range $z \in [0, +\infty]$.

In this paper, we are working with PWM signals and we do not need to consider negative numbers. Therefore, we are only interested in unipolar SC format. In unipolar SC, we can perform multiplication operation with logic AND gate. Fig. 4(a) shows the operation of an AND gate on two stochastic bitstreams, where $z = x \times y$. Fig. 4(b) shows the scaled addition operation of a multiplexer performed on two stochastic bitstreams. In this figure, x and y are inputs, ss is the scale factor, and zz is output. They are all represented in unipolar SC, thus $xx, y, s, z \in [0,1]$. The multiplexer circuit computes: $z = s \times y + (1 - s) \times x$.

Table 1. Behaviour of different components

	Rise time	**Overshoots**	**Settling time**	**Steady state error**
Proportional	decrease	increase	Small change	decrease
Integral	decrease	increase	increase	eliminate
Derivative	increase	decrease	decrease	No change

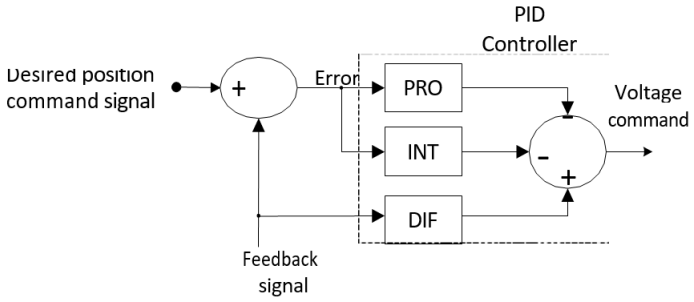

Fig. 2. Simple PID block diagram

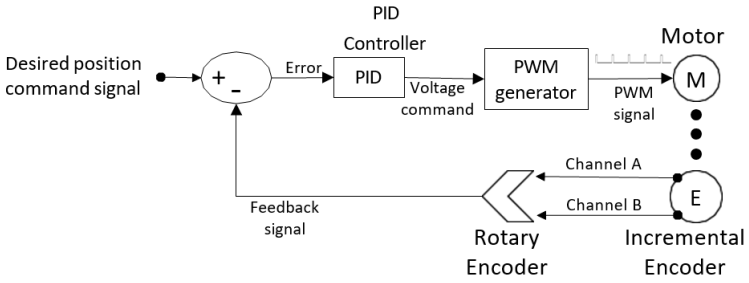

Fig. 3. Block diagram of a basic servo controller

3. PROPOSED WORK

In this section, we describe our implementations of servomotor controller based on both quadrature decoder and alternative efficient decoder. We start with quadrature decoder-based implementation and then move on to the efficient decoder-based implementation.

3.1. Quadrature Decoder based Implementation

3.1.1. Decoder Implementation

We want to implement a decoder which uses a counter that increments or decrements according to the quadrature signals (pulse tarin A & B) from incremental encoder. Fig. 5 shows our circuit implementation of quadrature decoder. This circuit counts all the transitions of the quadrature inputs i.e., both leading and trailing edge of pulse train A and B. We use two delay flip-flops (D-FFs) to generate delayed version of pulse train A and B. All the four signals (A, B, delayed A, delayed B) are fed into a four input XOR gate to generate count enable signal. A and delayed B are fed into a two input XOR gate to generate counter Up/Down signal. Depending on the direction of rotation of the encoder, either pulse train A leads pulse train B or vice versa. If for clockwise (CW) rotation (A leads B), the counter counts up, then for counterclockwise (CCW) rotation (B leads A), counter counts down. To provide the direction signal, we use another D-FF. We check for the B signal in every positive edge of A. When A is leading, the direction is '0', and when B is leading, direction is '1'. Thus, we can determine the encoder rotation direction.

Now we need to convert this position information from counter to time-encoded (PWM) waveform. Fig. 6 shows a small flowchart diagram for PWM generator.

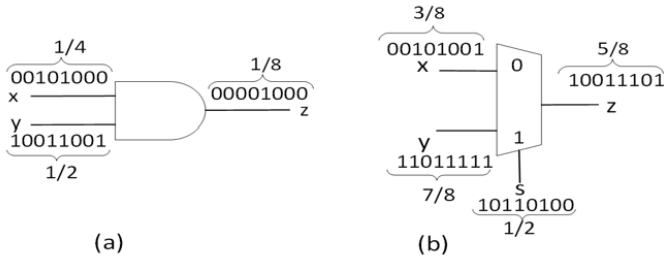

Fig. 4. Basic SC operation of (a) unipolar multiplication, (b) scaled addition

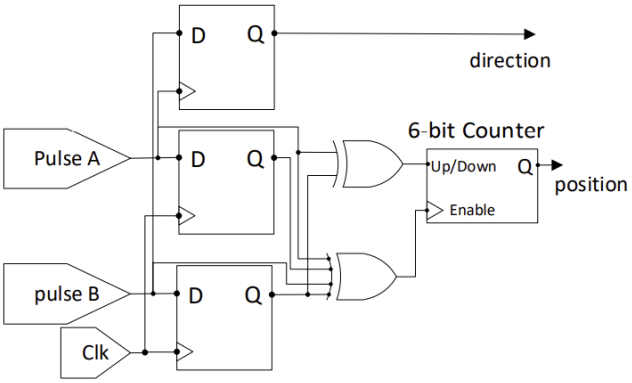

Fig. 5. Digital quadrature decoder

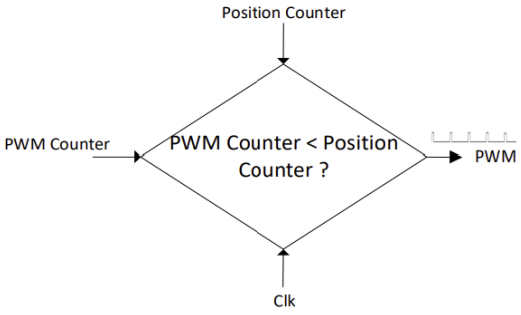

Fig. 6. PWM generator

We introduced a counter named PWM counter and initialized as 0. With every positive edge of clock, this counter goes up till a specific value, thus creating a ramp like signal. The period of PWM waveform depends on the maximum value of this counter and period of clock signal. Initially, the value of PWM counter is less than position counter. But with each clock cycle, the value of PWM counter goes up and eventually crosses position counter sooner or later. Thus, we get a PWM signal that is in high state at the beginning and in low state at the end.

We can use a similar method for desired position command signal and get another PWM signal and direction signal. Then we can calculate the absolute difference of PWM signals between the desired position command signal and feedback signal using a simple XOR gate as shown in Fig. 7. We use another XOR gate to find out if the direction of feedback signal and command signal is same or not. If their direction is same, the XOR output will be low and vice versa.

3.1.2. Controller Implementation

Now, we want to implement a controller which is similar to PID controller, but with stochastic method. It is not necessary to have separate circuit blocks for every block (PRO, INT, DIF) in fig. 2 and they can be combined when we design the system in SC. For proportional part, we need to multiply the error signal with a proportional constant. We will have to use a constant number source and convert it to stochastic bitstream using stochastic number generator (SNG). But we want to avoid this because this will add up too much hardware resources. We are going to assume proportional constant Kp is 1. We can design the circuit such that, Kp = 1, provides us with necessary output. Similarly, we are going to assume integral constant and derivative constant, K_i = K_d = 1. Therefore, we do not need any linear feedback shift register (LFSR) and SNG to convert constant numbers to stochastic bit streams. Implementation of stochastic derivative circuit can be very complex and resource consuming. Therefore, we will combine the integral and derivative part together and avoid implementing separate stochastic derivative circuit.

The digital integrator can be expressed as $y(n) = x(n) + y(n-1)$. Fig. 8 shows implementation of a traditional digital integrator. A delay register is used to hold the previous output of the integrator and this delayed output is sent back to the adder to perform the integration function. Fig. 9(a) shows our first approach of implementing stochastic integrator by replicating the accumulator design shown in fig. 8. In fig. 9(a), X represents the stochastic bitstream, which we want to integrate. We used an m-bit register for delay unit, where m is the length of stochastic bitstream. We also used a MUX for the adder operation. After our implementation in Verilog, we see that this kind of implementation does not really work as integrator. Because MUX only does scaled addition in SC and, in this case, the result is just the average of two consecutive error signals which does not serve our purpose. We want our integrator to add up the error to a specific extent determined by integrator constant. There is another way to implement the integral part for our controller [17]. Fig. 8(b) shows the proper implementation of stochastic integrator. An N-bit up/down counter is employed to integrate the input (error signal). Our error signal is in PWM format, and the counter takes input only in positive edge of clock signal. Therefore, if we have nn number of clock cycles between each PWM signal period, we can say the input to the N-bit counter is a n bit stochastic number. For large input value to the counter, the number of '1's should be higher than the number of '0's appeared in the bit-stream. If the input data has large value which means the probability of '1' appears in the bit- stream is high, the output of the up/down counter will go up in proportion to the value of the input data, and vice versa.

Therefore, for a large error, the integral part will give very large value. This will help to figure out if the motor rotation direction should be changed or not. Also, how fast the motor should rotate will be determined from this value. When the error becomes small, the integral part starts to reduce quickly.

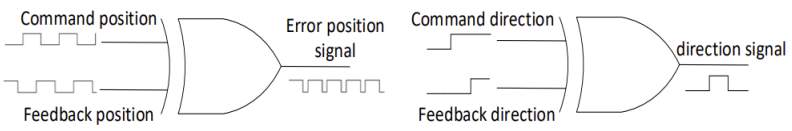

Fig. 7. XOR gate as subtractor of command signal and feedback signal

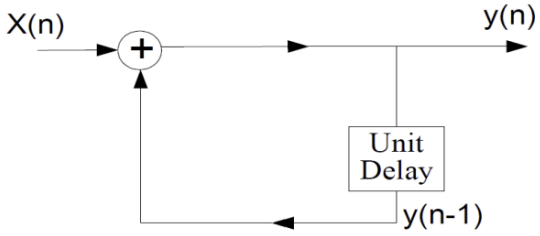

Fig. 8. Accumulator design of a digital integrator

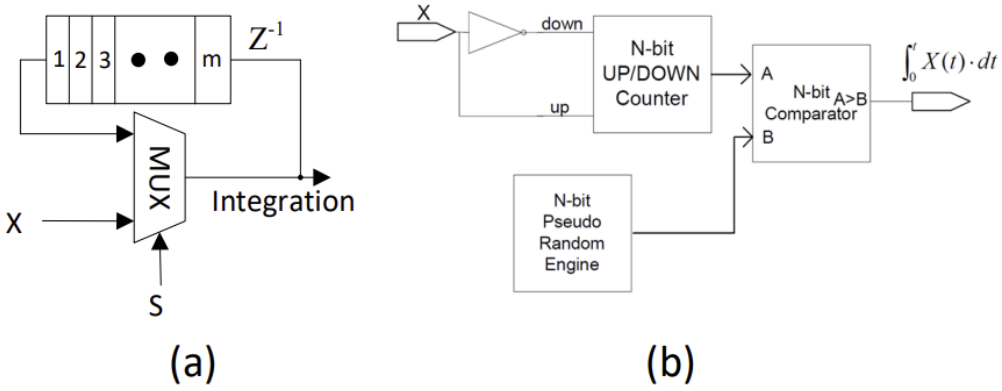

Fig. 9. (a) Wrong approach to stochastic integrator, (b) Right way to implement stochastic integrator [17]

But in this implementation, we need pseudo random engine which will increase circuit complexity. Also, to control the level of integration, K_i should be multiplied with the error signal, which will again increase hardware requirements. To solve this problem, we made a little change to this circuit. Fig. 10 shows our modified integrator.

In fig. 10, the N-bit up/down counter value is compared against a constant offset value, which we can choose for our system. By selecting a suitable constant offset value, we can have control over our integration level and avoid using SNGs to generate required constant values. Fig. 11 shows the flowchart of our stochastic integrator operation.

Now we have two PWM signals (error signal and integral signal), and they have some overlapping and non-overlapping 'on' state portion. When the direction is opposite between desired position command signal and feedback signal, we want a PWM signal with large duty cycle (more than a threshold value to indicate motor should turn the other direction). So, we use an OR gate between error signal and integral signal. Therefore, the resultant PWM will have larger duty cycle than both error signal and integral signal as shown in fig. 12(a). If the directions are same, we want a short duty PWM signal. Therefore, we do an AND operation between error signal and integral signal, so that PWM signal will have a positive pulse for only their overlapping portion as shown in fig. 12(b). Fig. 13 shows the process with a small flowchart diagram. Depending on the servo motor type, we can choose the appropriate PWM signal pulse width.

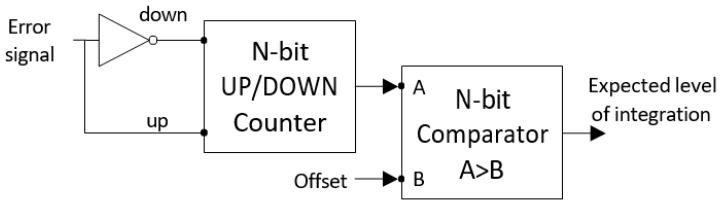

Fig. 10. Modified stochastic integrator

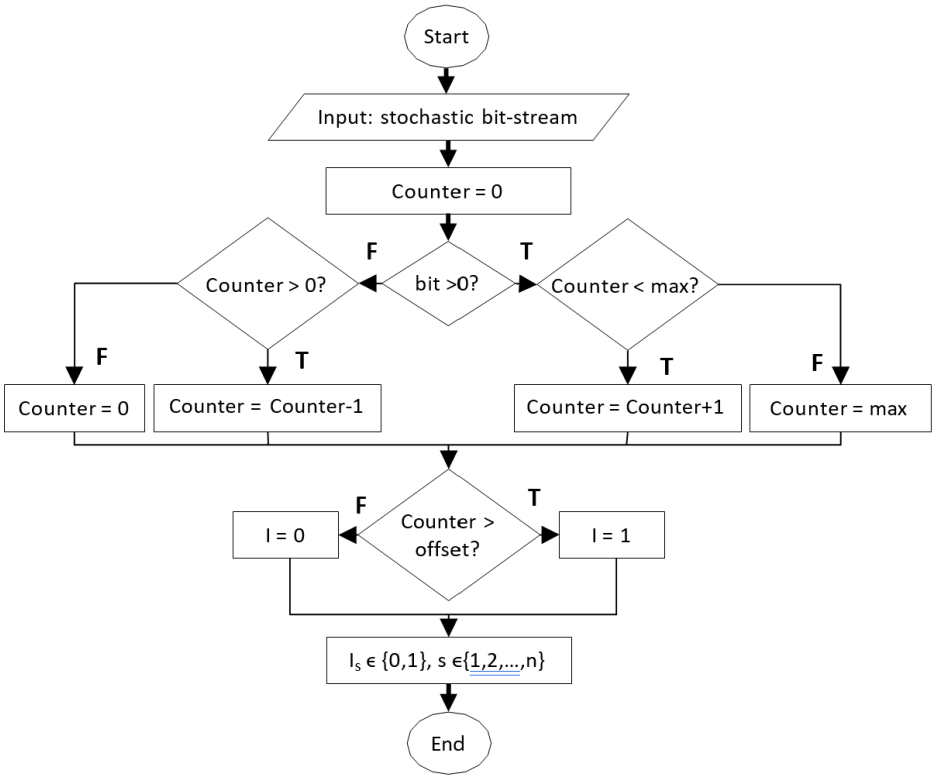

Fig. 11. Flowchart of our stochastic integrator operation

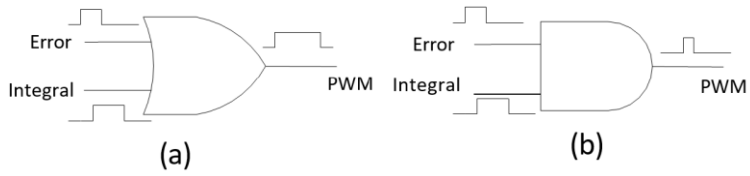

Fig. 12. PWM generation for servomotor using ‘AND’ and ‘OR’ gate

3.2. Efficient Decoder based Implementation

Unfortunately, quadrature decoder requires too many FFs, XOR gates and counters for both feedback and command signals. So, we want an alternative hardware efficient decoder for our servo controller.

3.2.1. Decoder Implementation

According to our new alternative and more efficient decoder, when pulse train *A* leads pulse train *B*, at the rising edge of *A*, *B* is always '0'. When *B* leads *A*, at the rising edge of *A*, *B* is '1'. We use *A* as a clock for an FF and *B* as its input. Then we can detect the direction of the rotation according to the value of *B*. For clockwise (CW) rotation, direction value is '0' and for counterclockwise (CCW) rotation, direction value is '1'. For position information, we only need to generate a signal that has larger duty cycle when we have more pulses of *A* and *B*, and vice versa. So, a simple approach is to OR signals *A* and *B* together. In this way, we only need 1 FF and 1 OR gate for decoder implementation.

We can use a similar method for command signal and get another position signal and direction signal. Therefore, we need two FFs and two OR gates for decoding both command and feedback signals. Also, we need one XOR gate to find whether their directions are same or not. Therefore, we have 3 input signals for the controller, two position signals for command and feedback, and one direction signal as shown in fig. 14. In this figure, f_A, f_B, f_dir, and f_pos are associated with feedback signal, and c_A, c_B, c_dir, and c_pos are associated with command signal.

3.2.2. Controller Implementation

Now, we want to implement a controller with stochastic method, which provides PWM signal for the servomotor. For the controller part, we use counter for determining the pulse width of final PWM. When command position signal is high, counter counts up to increase the pulse width of PWM. If the motor shaft rotation is in the opposite direction, counter counts up at a much faster rate. If the motor shaft rotates in the same direction, counter counts down to decrease the pulse width. Fig. 15 shows the decision tree for the counter operation.

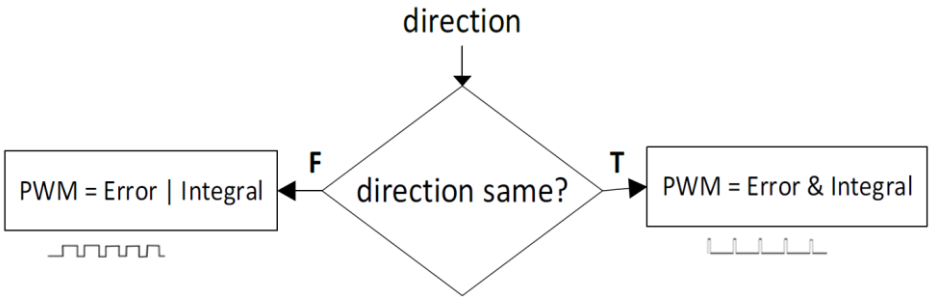

Fig. 13. Final PWM waveform generation for servo motor

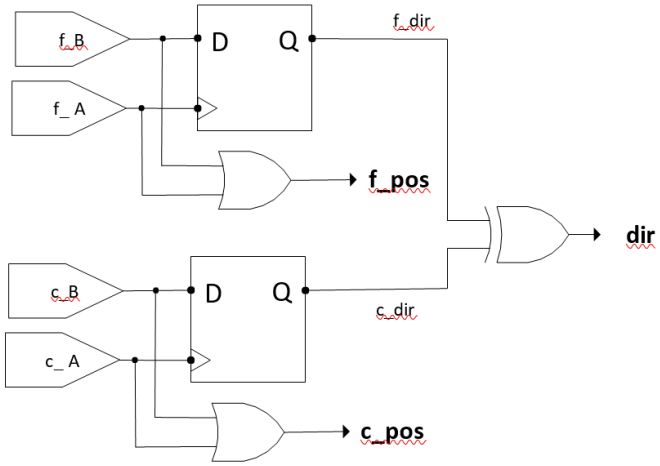

Fig. 14. Efficient decoder for command and feedback signals

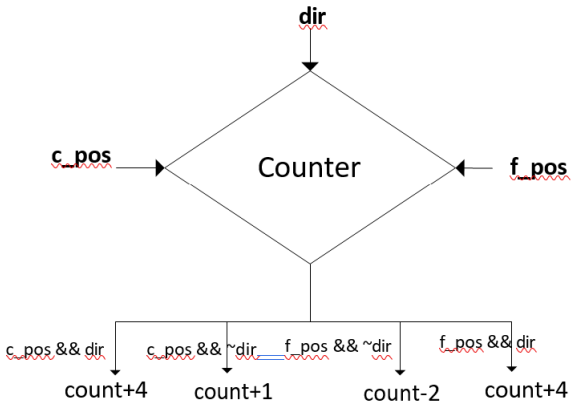

Fig. 15. Counter decision tree

From fig. 15 we see that, when direction of motor shaft and command signal are opposite (dir), counter counts up at a very fast rate (count+4) to generate high duty cycle PWM signal for the servomotor. When the width of the PWM reaches the threshold value, motor will try to correct its direction of rotation. For same direction (~dir), counter counts up (count+1) or down (count-2) depending on the state of command signal and feedback signal respectively. We can select the counter increase or decrease rate according to the motor type and system, to provide us with required PWM signal. We also need to generate a ramp like signal to compare against the counter as shown in fig. 6. For example, we can implement a ramp signal generator that has a period of 20ms. Therefore, we can generate our PWM signal with 20ms period.

For this implementation, we need less hardware resources than quadrature decoder-based implementation, especially because we used simpler stochastic decoder.

4. SIMULATION AND ANALYSIS

For the proposed designs, we perform simulations to estimate the hardware complexity of the designs by implementing them in Xilinx Artix-7 through Vivado HLS software. In order to evaluate the performance and hardware complexity of the proposed SC based controllers, we

compare them with a BC based PID controller. For BC based implementation, we employed the PID architecture proposed in [18] with quadrature decoder. Fig. 16 shows the PID controller design.

We present the simulation results achieved by Vivado HLS 2019.2. We use Verilog to model our circuits and select Xilinx Artix-7 chip xc7a35tcpg236-1 as the target FPGA device. We have shown a comparison of hardware resources (area), power, and speed in fig. 17, between our 1st stochastic approach with quadrature decoder, 2nd stochastic approach with efficient decoder and a digital deterministic approach. In fig. 17, we considered only dynamic power consumption. Implementation of servomotor controller using SC, compared to BC, can reduce the required area for 1st SC approach and 2nd SC approach by 56.7% and 73.7% respectively and it can save dynamic power consumption by 33.33% for both SC methods. As the results show, SC achieves better performance compared to BC for hardware realization of servomotor controller.

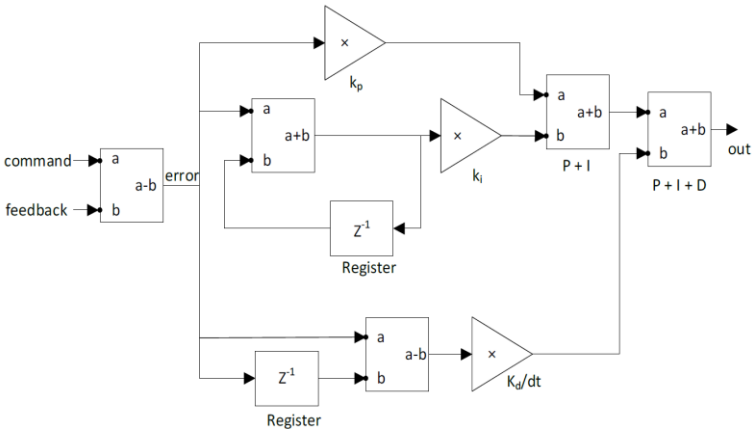

Fig. 16. Block diagram of a digital PID controller

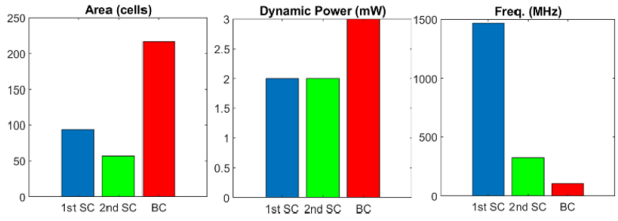

Fig. 17. Comparison of area, power, and spread for different implementation

5. CONCLUSION

In this paper, we studied two SC based servo controller architectures. From our simulation results, we can claim that SC based architectures provide more efficient circuits compared to BC based architecture for servomotor controller. Also, our 2nd SC approach is more hardware efficient than 1st SC approach because it requires simpler decoder circuit. But our 1st SC approach requires simpler controller part because we don't have to generate ramp signal and comparator, as our error signal is already in PWM waveform. Therefore, speed is much higher for our 1st SC approach. In our SC based controller circuits, we did not use a separate derivative part because it may complicate the design too much and it is not mandatory for our design. But

adding a separate derivative part might provide a more flexible control over our designs. In our future work we will try to incorporate an efficient stochastic derivative part in our circuits.

REFERENCES

[1] R.-F. Fung, J.-H. Lin, "Vibration Analysis and suppression Control of an Elevator String Actuated by a PM Synchronous Servo Motor," Journal of Sound and Vibration (1997) 206(3), 399-423.

[2] D. C. Schuurman, D. W. Capson, "Robust direct Visual Servo Using Network-Synchronized Cameras," IEEE Transactions on Robotics and Automation, Vol. 20, No. 2, April 2004.

[3] M. Buttu, A. Orlati, G. Zacchiroli, M. Morsiani, F. Fiocchi, F. Buffa, G. Maccaferri, G. P. Vargiu, C. Migoni, S. Poppi, S. Righini, A. Melis, "Diving into the Saedinia Radio Telescope minor servo system," Proc. SPIE 8451, Software and Cyberinfrastructure for Astronomy II, 84512L (24 September 2012).

[4] Huang, T., Mei, J., Li, Z., Zhao, X., and Chetwynd, D. G. (October 6, 2004). "A Method for Estimating Servomotor Parameters of a Parallel Robot for Rapid Pick-and-Place Operations." ASME.J. Mech. Des. July 2005; 127(4): 596-601.

[5] G. Ellis and R. D. Lorenz, "Comparison of motion control loops for industrial applications," Conference Record of the 1999 IEEE Industry Applications Conference. Thirty-Forth IAS Annual Meeting (Cat. NO.99CH36370), Phoenix, AZ, USA, 1999, pp. 2599-2605 vol.4, doi: 10.1109/IAS. 1999.799205.

[6] L. Lindner, O. Sergiyenko, V. Tyrsa and P. Mercorelli, "An approach for dynamic triangulation using servomotors," 2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE), Istanbul, 2014, pp. 1926-1931, doi: 10.1109/ISIE.2014.6864910.

[7] Y. F. Li and C. C. Lau, "Development of fuzzy algorithms for servo systems," in IEEE Control Systems Magazine, vol. 9, no. 3, pp. 65-72, April 1989, doi: 10.1109/37.24814.

[8] F Zhaoyong Zhou, Tiecai Li, T. Takahashi and E. Ho, "FPGA realization of a high-performance servo controller for PMSM," Nineteenth Annual IEEE Applied Power Electronics Conference and Exposition, 2004. APEC '04., Anaheim, CA, USA, 2004, pp. 1604-1609 Vol.3, doi: 10.1109/APEC.2004.1296079.

[9] A. Ardakani, F. L. Primeau, N. Onizawa, T. Hanyu, W. J. Gross, "VLSI Implementation of Deep Neural Network Using Integral Stochastic Computing", IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 25, Issue 10, Oct. 2017.

[10] H. Sim and J. Lee, "A New Computing Multiplier with Application to Deep Convolution Neural Network", 54th ACM/EDAC/IEEE Design Automation Conference, 2017.

[11] P. Li, D. J. Lilja, W. Qian, K. Bazargan, and M. D. Riedel, "Computation on Stochastic Bit Streams Digital Image Processing Case Studies," IEEE Trans. VLSI Syst., vol. 22, no. 3, pp. 449–462, Mar. 2014.

[12] S. A. Salehi and D. D. Dhruba, "Efficient Hardware Implementation of Discrete Wavelet Transform Based on Stochastic Computing," 2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Limassol, Cyprus, 2020, pp. 422-427, doi: 10.1109/ISVLSI49217.2020.00083.

[13] S. A. Salehi, Y. Liu, M. D. Riedel, and K. K. Parhi, "Computing Polynomials with Positive Coefficients using Stochastic Logic by Double-NAND Expansion," GLSVLSI'17 Proceedings of the on Great Lakes Symposium on VLSI 2017 Pages 471-474.

[14] Lynch, K., Marchuk, N. and Elwin, M., 2015. Embedded computing and mechatronics with the PIC32 microcontroller. Newnes.

[15] Alaghi, A., 2015. The Logic of Random Pulses: Stochastic Computing (Doctoral dissertation).

[16] B. R. Gaines, "Stochastic computing," Proc. AFIPS Spring Joint Computer Conf., pp. 149–156, 1967. doi:10.1145/1465482.1465505.

[17] Zhang, D., 2006. Stochastic Approach to Digital Control Design and Implementation in Power Electronics.

[18] Geramipour, Arezou, et al. "Design of FPGA-based digital PID controller using Xilinx SysGen® for regulating blood glucose level of type-I diabetic patients." Int J Mechatron Electr Comput Technol 3.7 (2013): 56-69.