*Article*

# Lightweight and Error-Tolerant Stereo Matching with a Stochastic Computing Processor

**Seongmo An, Jongwon Oh, Sangho Lee, Jinyeol Kim, Youngwoo Jeong, Jeongeun Kim and Seung Eun Lee ***

Department of Electronic Engineering, Seoul National University of Science and Technology, Seoul 01811, Republic of Korea; ahnseongmo@seoultech.ac.kr (S.A.); ohjongwon@seoultech.ac.kr (J.O.); leesangho@seoultech.ac.kr (S.L.); kimjinyeol@seoultech.ac.kr (J.K.); jeongyoungwoo@seoultech.ac.kr (Y.J.); kimjeongeun@seoultech.ac.kr (J.K.)

* Correspondence: seung.lee@seoultech.ac.kr; Tel.: +82-2-970-9021

**Abstract:** Stereo matching, utilized in diverse fields, poses a challenge to systems in resource-constrained environments due to the significant growth of computational load with image resolution. The challenge is crucial for the systems because fields utilizing stereo matching require short operational time for real-time applications and low power architecture. Stochastic computing (SC) is able to be a valuable approach to address the challenge by reducing the computational load by representing binary numbers with stochastic sequences, which are encoded as a probability value, and by leveraging the concept of mathematical probability. Also, it is possible for a system to be error-tolerant by utilizing the characteristics of stochastic computing. Therefore, in this paper, we propose an approach for lightweight and error-tolerant stereo matching with a hardware-implemented stochastic computing processor. To verify the feasibility and error tolerance of the proposed system, we implemented the proposed system and conducted experiments comparing depth maps with or without stochastic computing by calculating similarities. According to the experimental results, the proposed system indicated no significant differences in output depth maps and achieved an improvement in the depth maps from error-injected input images by an average of 58.95%. Therefore, we demonstrated that stereo matching with stochastic computing is feasible and error-tolerant.

**Keywords:** stochastic computing (SC); stereo matching; error tolerance; processor

check for updates

**Citation:** An, S.; Oh, J.; Lee, S.; Kim, J.; Jeong, Y.; Kim, J.; Lee, S.E. Lightweight and Error-Tolerant Stereo Matching with a Stochastic Computing Processor. *Electronics* **2024**, *13*, 2024. https://doi.org/10.3390/electronics13112024

Academic Editor: José Carlos Castillo

Received: 4 April 2024
Revised: 30 April 2024
Accepted: 15 May 2024
Published: 22 May 2024

## 1. Introduction

Stereo matching, a computer vision technique that estimates 3D structures from 2D images, has been widely employed in various fields such as autonomous driving and augmented reality (AR) to generate depth maps of environments [1]. The primary goal of stereo matching is to determine disparities, which represent the displacements between corresponding features in each pair of stereo images, as illustrated in Figure 1 [2]. However, the process of determining disparities involves comparing all pixel values within a search window against those in a template block, leading to an increase in computational complexity with the rise in image resolution. This presents significant challenges for systems operating in resource-limited environments [3].

Recognizing the limitations of current stereo matching methods, this paper proposes a novel approach to address these challenges with SC. SC is a computing paradigm that employs stochastic sequences to represent and process data. A stochastic sequence is a bit stream of ‘0’s and ‘1’s obtained by comparing a binary number with random numbers, as shown in Figure 2. By approximating a binary number into a probability value between ‘0’ and ‘1’ and leveraging the concept of mathematical probability [4], SC replaces traditional arithmetic units with concise logic gates [5]. This results in outputs with relatively low accuracy but high error tolerance [6], making SC a promising solution for computationally intensive applications such as image processing and artificial intelligence [7–9].

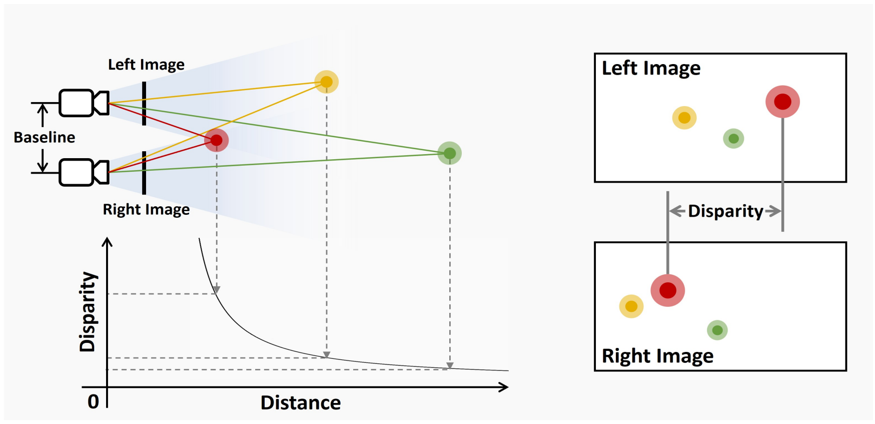

**Figure 1.** An overview of how to obtain the disparity.

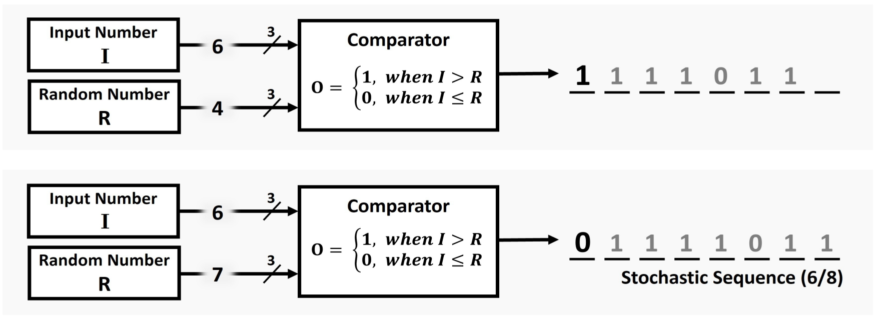

**Figure 2.** The extraction process of the stochastic sequence.

In this paper, we introduce a stereo matching system that includes a hardware-implemented SC unit. The SC unit, featuring a specially designed parallel linear feedback shift register (LFSR), addresses the latency issue of the random number generator and offers benefits in terms of area efficiency and computation time. We connected the SC unit and the ARM-based core via a bus, implemented the SC processor on an FPGA, and performed stereo matching algorithm operations through the SC unit to obtain a depth map.

Our proposed system not only addresses the computational challenges of stereo matching but also improves the error tolerance of the process. We validated the feasibility of our system by comparing the depth map obtained with the SC unit to that obtained without it. Furthermore, we verified the error tolerance of our system by obtaining depth maps from input images with injected errors. Remarkably, when utilizing the SC unit, the output depth map for the input images containing errors improved by an average of 58.95% compared to when not utilizing the SC unit. The contributions of this work are as follows:

- The proposed system shows that there is no significant difference between the depth map obtained with the SC unit and the one obtained without the SC unit.
- The proposed system demonstrates better tolerance to errors when the SC unit is utilized.
- The architecture of the LFSR included in the proposed system is parallelized to achieve area efficiency.

This paper, therefore, presents a significant advancement in stereo matching techniques, offering a solution that is both computationally efficient and error-tolerant. This paper consists of the following. Section 2 outlines the background on the SC and stereo matching algorithm employed in this work. Section 3 introduces related works about the systems utilizing SC. Section 4 demonstrates the proposed overall system architecture and the detailed aspects of the SC unit. Section 5 presents the implementation of the proposed system and shows the experimental results. Lastly, Section 6 concludes this paper.

## 2. Background

### *2.1. Stochastic Computing*

The implementation of SC requires three steps: stochastic number generation, stochastic computing operation, and binary conversion [6]. In stochastic number generation, binary numbers are converted into stochastic sequences with LFSRs and comparators. The LFSR generates pseudo-random numbers by shifting bits and applying a feedback function, which is a linear combination of certain bits in the register. As shown in Figure 3, an LFSR forms a feedback loop consisting of a shift register and an XOR. By utilizing the structure of the LFSR, a pseudo-random number is generated. The stochastic computing operation is used to compute operations with the stochastic sequence, and the binary conversion converts the sequence back into a binary number.

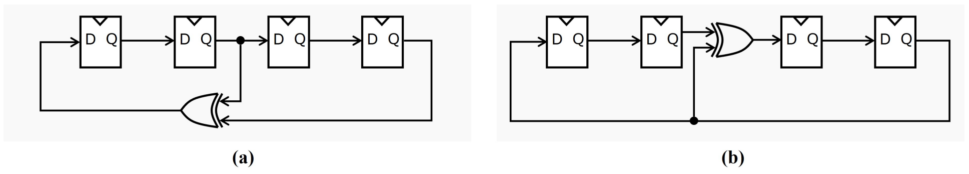

**Figure 3.** Circuits of Fibonacci LFSR and Galois LFSR: (**a**) 4-bit Fibonacci LFSR, (**b**) 4-bit Galois LFSR.

#### 2.1.1. Area Efficiency

With SC, the traditional arithmetic operations are replaced with brief logic gates. For example, a common binary multiplier is replaced by an AND gate, and a common binary adder is replaced by a multiplexer (MUX), which is possible because of the application of the concept of mathematical probability. Since the stochastic sequence refers to a probability value, independently generated stochastic sequences represent independent probability values [10].

Figure 4 demonstrates the stochastic computing operations. The probability that two independent events occur at the same time is mathematically the multiplication of the probabilities. Therefore, if two independent stochastic sequences are the inputs of an AND gate, then the output of the gate is equal to the product of the probabilities. Similarly, the probability that an event $A$ or $B$ occurs is the sum of their respective probability values. Therefore, if the inputs of the MUX are independent stochastic sequences, the output is related to the probability of the MUX select signal $S$ being '1' or being '0'. When $S$ is '1', the output of the MUX is the probability of the $S$ and the $A$ to occur simultaneously, $P(A) \times P(S)$. When $S$ is '0', the output is $P(B) \times (1 - P(S))$. Consequently, the final output of the MUX is $P(A) \times P(S) + P(B) \times (1 - P(S))$. If the probability of $S$ is $4/8$, the output of the MUX is equal to the sum of the probabilities scaled to $1/2$. The subtraction operation is also possible by adding an NOT gate, as shown in Figure 4c [11].

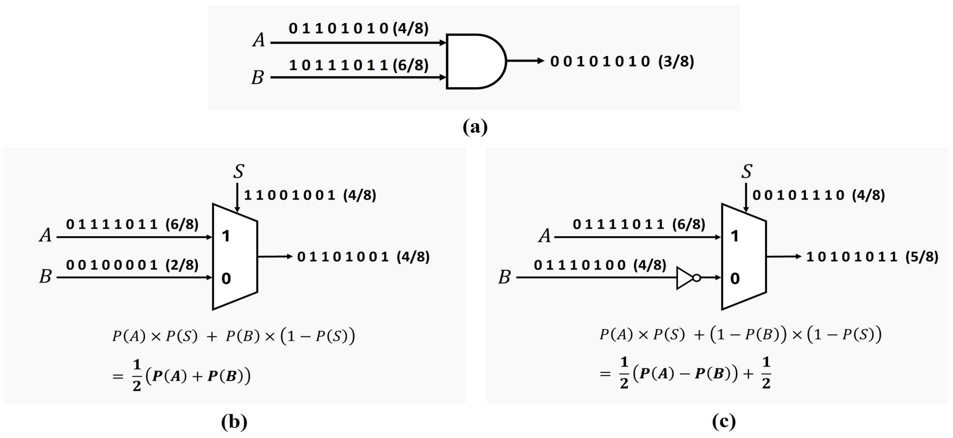

**Figure 4.** Example of stochastic computing operations: (**a**) stochastic multiplication, (**b**) stochastic addition, (**c**) stochastic subtraction.

### 2.1.2. Error Tolerance

Because each digit in a stochastic sequence has the same weight, errors occurring in the sequence have a relatively small influence. For example, if an error occurs in one part of the 256-bit stochastic sequence representing an 8-bit binary number, the difference in the result is only 1/256 [12]. Figure 5 demonstrates a case of bit flip occurring in a stochastic sequence and the comparison of the results. As shown in Figure 5, even if an error occurs in any bit of a stochastic sequence, the impact on the output is relatively small compared to that of a binary number, in which errors of higher-weighted bits have a significant impact [13].

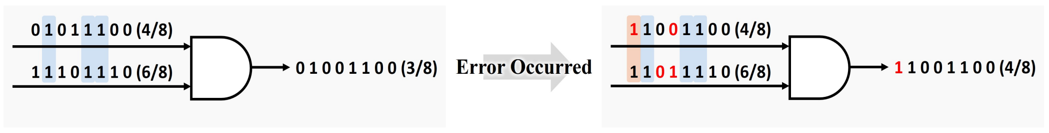

**Figure 5.** Comparison of ideal case and error-occurred case of stochastic multiplication.

## 2.2. *Stereo Matching*

To extract disparities from stereo images, corresponding features between both images are to be detected. One technique for detecting corresponding features is template matching, which involves searching for the most correlated block in an image with a template block in another image [14]. The stereo images have two characteristics: the cameras are horizontally aligned, and searches for correlated blocks are conducted in a single image. As a result, the correlated block exists on the same horizontal line as the template block and has a position within a certain range of x-coordinates. In other words, a search window for the correlated block is determined based on the position of the template block. The disparity is decided by calculating the difference in the x-coordinates of the template block and the most correlated block in the search window [15].

As described in Figure 6, stereo matching consists of four steps. In the cost computation, the matching cost is decided according to the differences in values between the corresponding pixels. The cost aggregation refers to the aggregation of the matching costs to obtain the most correlated block within the search window. In the disparity computation, the disparity is obtained by calculating the distance of the coordinates between the template block and the correlated block selected through the cost aggregation. The disparity refinement aims to rectify the incorrect disparities obtained by stereo matching [16].

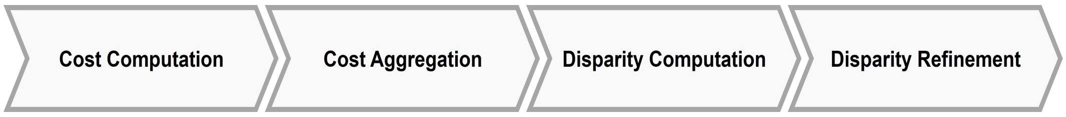

**Figure 6.** Composition of stereo matching.

### 2.2.1. Cost Aggregation

After the cost computation, which calculates the matching cost (in this work, the difference in pixel values) of corresponding pixels in two blocks, cost aggregation proceeds. In the cost aggregation, block-based cost matching functions are commonly employed to aggregate costs [2]. One of the uncomplicated cost matching functions is the sum of absolute difference (SAD). When the blocks are identical, the result is ‘0’. Another function is the sum of squared difference (SSD), which requires multiplication operations resulting in higher computational complexity than the SAD [17]. Normalized cross correlation (NCC) is a method of normalizing the similarity between the two blocks, which is more computationally complicated than the SAD and the SSD as it involves multiplication, division, and square root operations [18]. Equations (1)–(3) describe mathematical expressions for the SAD, SSD, and NCC, respectively.

$$SAD = \sum_{j=1}^{H}\sum_{i=1}^{W}|a_{ij} - b_{ij}| \tag{1}$$

$$SSD = \sum_{j=1}^{H}\sum_{i=1}^{W}(a_{ij} - b_{ij})^2 \tag{2}$$

$$NCC = \sum_{j=1}^{H}\sum_{i=1}^{W}\frac{(a_{ij} - \bar{a})(b_{ij} - \bar{b})}{\sigma_a\sigma_b}$$
$$\left(\text{Where } \bar{a} = \frac{1}{N}\sum_{i,j}a_{ij},\ \bar{b} = \frac{1}{N}\sum_{i,j}b_{ij},\ \sigma_a = \sqrt{\frac{1}{N}\sum_{i,j}(a_{ij} - \bar{a})^2},\ \sigma_b = \sqrt{\frac{1}{N}\sum_{i,j}(b_{ij} - \bar{b})^2}\ \right) \tag{3}$$

Among the three functions mentioned above, the SAD is the most suitable function for cost-effective architectures. This is because the SAD employs only the most uncomplicated operations [19]. Therefore, we utilized the SAD as the cost matching function to make it suitable for an architecture with SC. Figure 7 demonstrates an overview of performing the cost computation and aggregation employing the SAD.

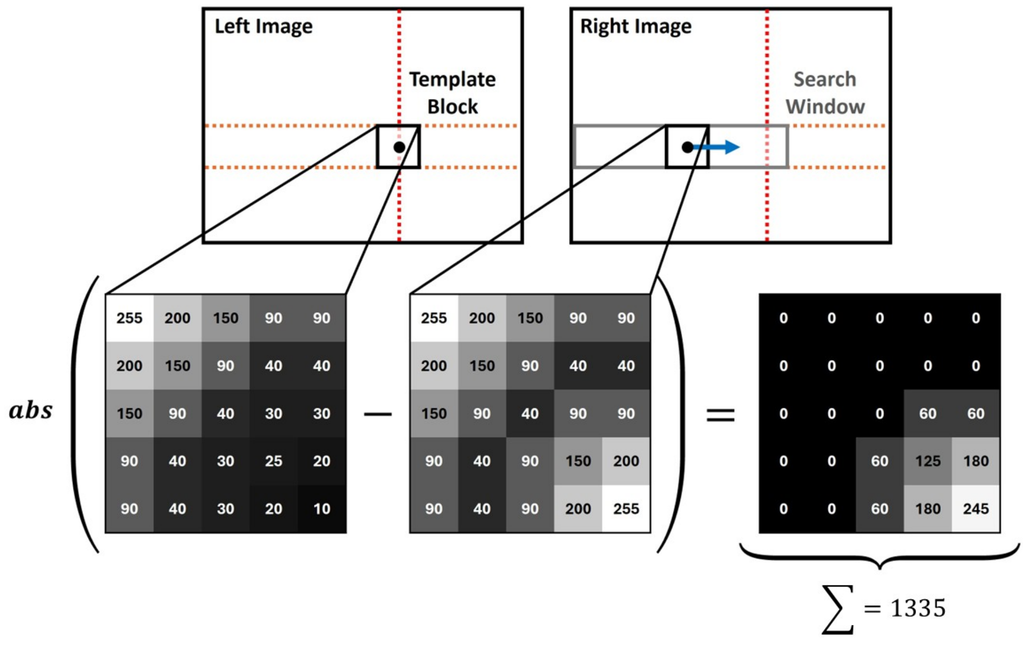

**Figure 7.** Cost computation and aggregation with the SAD.

### 2.2.2. Disparity Computation

The process of disparity computation is a critical step in stereo matching. The disparity is obtained as the difference between a template block and the block having the highest correlation in the search window [20].

To begin with, we calculate and store the correlation values, i.e., SAD values, for each pair of blocks—the template block and all the blocks within the search window. Once we have the SAD values, we proceed to obtain the disparity. This is accomplished by calculating the difference between the x-coordinate of the template block and that of the block having the minimum result value from the SAD. The block with the minimum SAD value is considered to have the highest correlation with the template block, and thus, its x-coordinate is employed in the disparity calculation.

### 2.2.3. Disparity Refinement

After the disparity computation, the generated disparity map often contains noises such as invalid matches. These are disparities that do not accurately represent the depth information of the scene and are caused by various factors such as occlusions, featureless regions, or repetitive patterns [21].

To address this issue, we implemented methods such as filtering techniques. One common approach is to apply a median filter to the initial disparity maps. The median filter works by replacing each disparity value with the median of the disparities in its neighborhood. This has the effect of preserving the edges while removing isolated noises.

Applying a median filter not only helps in reducing the noise in the disparity map but it also minimizes the computational complexity. This is because the median filter operates in a local neighborhood and does not require knowledge of the entire image. Therefore, the median filter is able to be efficiently implemented even on large images [22].

## 3. Related Work

SC is a non-traditional computing technology that encodes information with finite-length stochastic sequences of '0's and '1's. The probabilistic elements enable the simplification of complex operations and resilience to errors, and various research projects utilizing these properties have been carried out [7,9,23–28]. Particularly, studies have been conducted to enhance the performance of SC and apply it to applications requiring robustness against noise or demanding low power consumption, such as artificial intelligence filtering operations [7,23,24] and image processing [9,25–28].

In [7], Extended Stochastic Logic (ESL) was applied as a method to solve the problem of low accuracy in utilizing stochastic computing for artificial neural networks (ANNs). The study replaced the accumulation process in ANN computing with an ESL-based adder and substituted the conventional activation function with an ESL-based ReLU. This approach resulted in a 48% improvement in accuracy compared to conventional SC-based methods, an 84% reduction in area compared to non-SC-based methods, and a 60% decrease in power consumption. In [23], a parallel SC-based neural network (NN) accelerator was proposed to enhance the fault tolerance. This yielded a 2.8$\times$ improvement in energy efficiency compared to traditional binary computing methods. The authors of [24] conducted research to reduce overhead convolutional neural networks (CNNs) utilizing SC by addressing high parallelism. Pseudo-Sobol sequences were proposed for SC-CNN, and an efficient parallel computation-conversion hybrid convolution architecture was developed, leading to improvements of 41% in energy efficiency and 36% in area.

In [9], non-scaling adders and subtracters were introduced which efficiently performed cascade computations compared to scaling adders. These were verified by applying them to an image sharpening filter. The accuracy of computations and the quality of the sharpened images were measured by peak signal-to-noise ratio and the structural similarity index measure. Ref. [25] proposed a simple method to improve the accuracy of SC by exchanging the wires used in operations and suggests adders and multipliers for SC. These were validated by applying adders and multipliers to the edge detection algorithm, resulting in a reduction in area utilization (64%) and power consumption (96%) compared to the accurate edge detection. In [26], a new technique called approximate stochastic computing (ASC) for improving computation time in image processing was proposed. The research verified its effect on computation time by an edge detection algorithm. Ref. [27] designed a fuzzy noise reduction filter based on SC. The experimental result showed a reduction in the hardware area and power consumption compared to conventional binary implementations. And the proposed design preserved the quality of the results. In [28], the research analyzed the stochastic absolute value function, stochastic tanh function, and one-parameter linear gain functions in SC. The study verified the validity of SC through four basic image processing algorithms: edge detection, frame difference-based image segmentation, median filter-based image enhancement, and image contrast stretching. The result showed SC has noise tolerance and consumes less hardware than the conventional methods.

Table 1 displays the related works regarding stochastic computing utilized for image processing.

**Table 1.** Analysis of related works.

| Source | Proposed Approach | Application | Pros | Cons |
|---|---|---|---|---|
| Temenos, N. et al., 2022 [9] | Non-scaling stochastic computing adder and subtracter. | Sharpening filter. | More accuracy in cascaded computations. Fewer resources are required than in stochastic computing with MUX. | It is impossible to utilize the efficiency that is obtained by utilizing a scaled stochastic computing circuit. |
| Joe, H et al., 2019 [25] | Stochastic computing by the wire exchanging method. | Edge detection algorithm. | It improved the accuracy of stochastic computing. | The hardware circuit design may become complicated, which may increase the power consumption and area of the hardware. |
| R. Seva et al., 2016 [26] | Approximate stochastic computing focusing on image processing. | Edge detection algorithm. | It reduced the long run-time of stochastic computing. | Stochastic computing may not be suitable for applications requiring high accuracy, such as edge detection. |
| S. N. Estiri et al., 2022 [27] | Stochastic computing is applied to a fuzzy noise reduction filter. | Fuzzy noise reduction filter. | Saving in the hardware area and power costs compared to the conventional binary implementation while preserving the quality of the results. | Since the accuracy of the results is not always guaranteed, it can be an important problem in noise filters. |
| P. Li et al., 2011 [28] | They analyzed the stochastic absolute value function, stochastic tanh function, and one-parameter linear gain function in stochastic computing. | Edge detection, frame difference-based image segmentation, median filter-based image enhancement, and image contrast stretching. | It proved stochastic computing is tolerant of soft errors and consumes fewer hardware resources than conventional computing. | As with other applications utilizing SC, it can be a problem if high accuracy is required. |

The following is a summary of how the works introduced in Table 1 address the potential critical drawback of low accuracy in stochastic computing.

In [9], the authors propose an efficient yet simple stochastic computation technique for multipliers and adders by exchanging the wires used for their operation. This design reduces the relative error in computation compared to conventional designs.

In [26], the authors propose a new technique called the approximate stochastic computing (ASC) approach focusing on image processing applications. This approach reduces the computation time of an SC by a factor of 16 at a trade-off of an error percentage of 3.13% in the absolute stochastic value.

And in [27], the authors implement an efficient hardware design for a well-known fuzzy noise reduction filter based on stochastic computing. The filter consists of two main stages: edge detection and fuzzy smoothing. The results demonstrate that the proposed design reduces the relative error in computation compared to the conventional designs and has a smaller area.

## 4. System Architecture

Figure 8 provides a comprehensive illustration of the overall architecture of our proposed system. This system is designed for lightweight and error-tolerant stereo matching, and it incorporates a stochastic computing processor as a key component. The SC processor is composed of two main parts: an SC module and an ARM Cortex-M0 core [11].

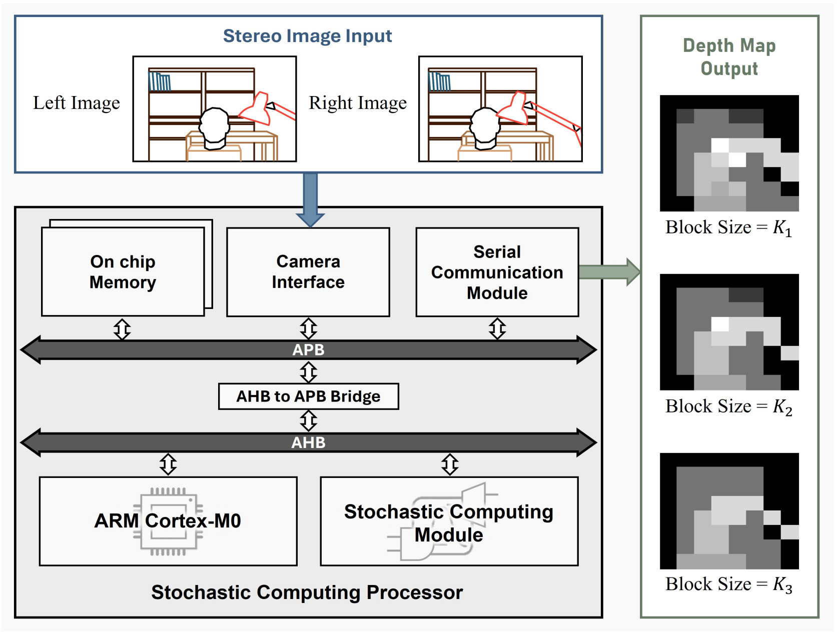

**Figure 8.** The overall architecture of the proposed system.

The SC module is responsible for the stochastic computations required for the stereo matching process. It is designed to efficiently calculate the SAD values, which are crucial for disparity computation in stereo matching. The ARM Cortex-M0 core, on the other hand, serves as the control unit of the system [29,30]. Once the stereo images are received from a camera module via the camera interface, the Cortex-M0 controls the stereo matching process by commanding the SC module to calculate the SAD values.

Once the SAD values are computed, the Cortex-M0 core proceeds to the next step of the process: disparity computation, which computes the disparities based on the SAD values and generates a depth map. The stereo matching operations are performed with a preset block size. Finally, the resulting depth map is transmitted to the outside through the serial communication module.

## 4.1. Stochastic Computing Module

As shown in Figure 9, the stochastic computing module includes a stochastic configuration register and stochastic computing core, which consists of stochastic number generators (SNGs), a stochastic computing unit, and a probability estimator (PE).

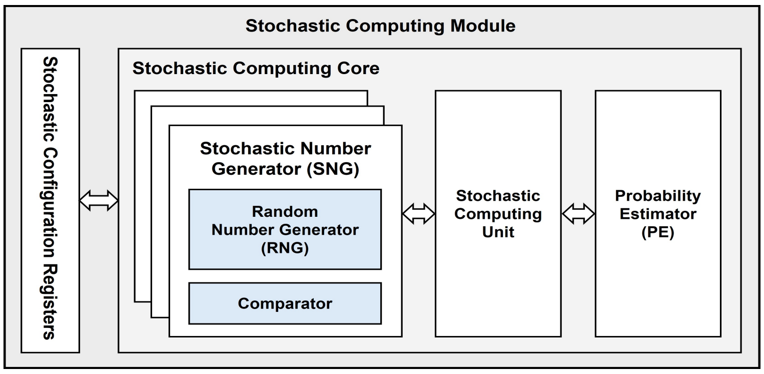
Stochastic Computing Module
Stochastic Configuration Registers
Stochastic Computing Core
Stochastic Number Generator (SNG)
Random Number Generator (RNG)
Comparator
Stochastic Computing Unit
Probability Estimator (PE)

**Figure 9.** A block diagram of the stochastic computing module.

#### 4.1.1. Stochastic Number Generator

In this work, we have employed a parallel architecture of an LFSR in the random number generator (RNG) to effectively address the issue of latency [31]. The latency here is reduced by generating multiple random numbers simultaneously during the stochastic sequence generation process.

Figure 10 provides an illustration of this concept. It shows the architecture of the basic SNG and that of the SNG equipped with a parallel LFSR [32]. The parallel LFSR, as shown in Figure 10b, is capable of generating multiple random numbers at the same time. This is achieved with a relatively low circuit area, especially when compared to an architecture that simply utilizes multiple LFSRs.

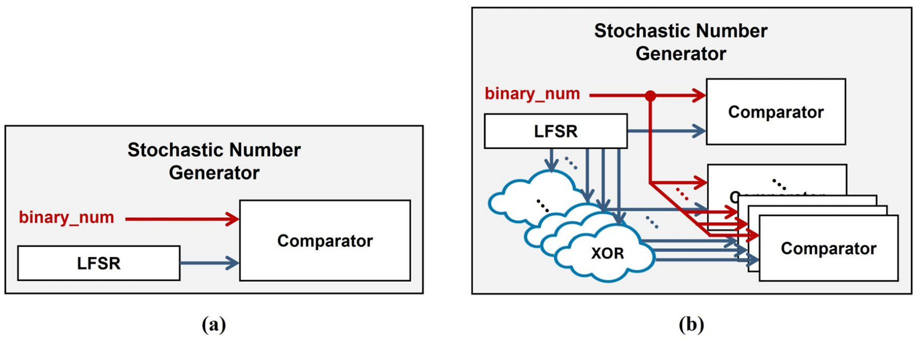

**Figure 10.** (**a**) The basic SNG architecture. (**b**) The SNG with a parallel LFSR.

The basic LFSR outputs a random number in a clock cycle. Therefore, if 256 random numbers are to be obtained through an 8-bit LFSR, 256 clocks are required [33]. The circuit employed in this work, however, requires only one clock cycle by designing the parallel LFSR circuit with 256 stages, as shown in Figure 11. In the circuit, the first random number output of the LFSR becomes the input of the 256th random number output, and the 256th random number output becomes the input of the 255th random number output, and so forth. The parallel LFSR circuit employs only one LFSR, eliminating the risk of loss in error rate due to correlations. Also, the circuit utilizes only XOR gates so that area efficiency is achieved without additional registers.

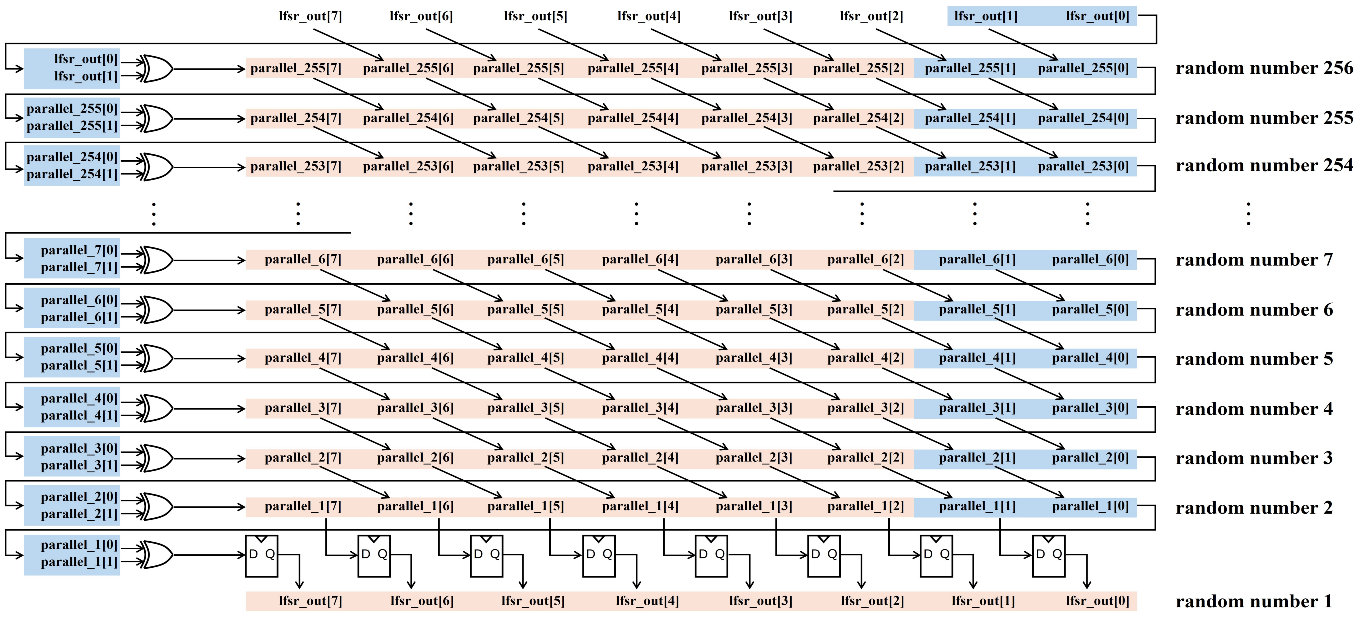
lfsr_out[7] lfsr_out[6] lfsr_out[5] lfsr_out[4] lfsr_out[3] lfsr_out[2] lfsr_out[1] lfsr_out[0]
lfsr_out[0] lfsr_out[1]
parallel_255[7] parallel_255[6] parallel_255[5] parallel_255[4] parallel_255[3] parallel_255[2] parallel_255[1] parallel_255[0]
random number 256
parallel_255[0] parallel_255[1]
parallel_254[7] parallel_254[6] parallel_254[5] parallel_254[4] parallel_254[3] parallel_254[2] parallel_254[1] parallel_254[0]
random number 255
parallel_254[0] parallel_254[1]
parallel_253[7] parallel_253[6] parallel_253[5] parallel_253[4] parallel_253[3] parallel_253[2] parallel_253[1] parallel_253[0]
random number 254
parallel_7[0] parallel_7[1]
parallel_6[7] parallel_6[6] parallel_6[5] parallel_6[4] parallel_6[3] parallel_6[2] parallel_6[1] parallel_6[0]
random number 7
parallel_6[0] parallel_6[1]
parallel_5[7] parallel_5[6] parallel_5[5] parallel_5[4] parallel_5[3] parallel_5[2] parallel_5[1] parallel_5[0]
random number 6
parallel_5[0] parallel_5[1]
parallel_4[7] parallel_4[6] parallel_4[5] parallel_4[4] parallel_4[3] parallel_4[2] parallel_4[1] parallel_4[0]
random number 5
parallel_4[0] parallel_4[1]
parallel_3[7] parallel_3[6] parallel_3[5] parallel_3[4] parallel_3[3] parallel_3[2] parallel_3[1] parallel_3[0]
random number 4
parallel_3[0] parallel_3[1]
parallel_2[7] parallel_2[6] parallel_2[5] parallel_2[4] parallel_2[3] parallel_2[2] parallel_2[1] parallel_2[0]
random number 3
parallel_2[0] parallel_2[1]
parallel_1[7] parallel_1[6] parallel_1[5] parallel_1[4] parallel_1[3] parallel_1[2] parallel_1[1] parallel_1[0]
random number 2
parallel_1[0] parallel_1[1]
D Q
lfsr_out[7] lfsr_out[6] lfsr_out[5] lfsr_out[4] lfsr_out[3] lfsr_out[2] lfsr_out[1] lfsr_out[0]
random number 1

**Figure 11.** Circuit of 256-stage parallel LFSR.

#### 4.1.2. Probability Estimator

Since the data output through the SC unit is still a stochastic sequence, it is necessary to convert the sequence back to the binary number, and the PE performs this conversion. In the probability value of $P = N/L$ encoded from the stochastic sequence, the number of '1's $(N)$ is approximated to a binary number. Therefore, the number of '1's on a stochastic sequence output as a result of the operation is counted, and the conversion to the binary number is performed in consideration of the case of being scaled [34].

## 5. Implementation and Experiments

### *5.1. Hardware Implementation*

Figure 12 demonstrates the hardware-implemented system. We designed the proposed SC processor, which includes the 8-bit parallel LFSRs, with Verilog HDL and downloaded it to an Altera MAX10, 10M50SCE144C8G FPGA. The SC processor interfaces with the external system through a universal asynchronous receiver/transmitter (UART), a serial communication module that enables serial communications. Furthermore, commands and input data for performing the stereo matching operation are stored through an external memory (4*Mbit* SRAM, CY7C1049GN30). The computation result, the depth map, is transmitted to the external system via UART.

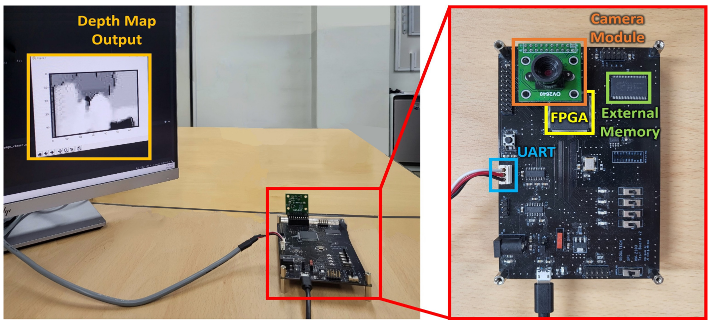

**Figure 12.** The hardware-implemented SC processor.

### *5.2. Experiments*

#### 5.2.1. Experimental Procedures

For the experiments, we employed a stereo image dataset [35]. To make the dataset images suitable for our proposed system, we converted them to grayscale and resized and cropped them [36], setting the resolution of the input images to 160 × 120. Figure 13 shows the employed dataset and the input images of the experiments.

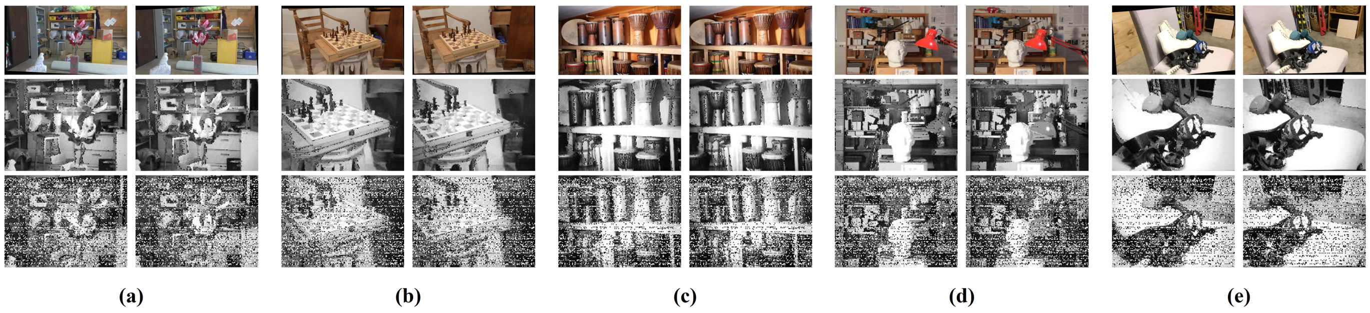

**Figure 13.** The stereo image dataset, noise-free input images, and noise-injected input images: (**a**) artroom, (**b**) chess, (**c**) djembe, (**d**) newkuba, (**e**) skates.

Before conducting the experiments, we performed a software simulation. This allowed us to determine the block sizes that would yield suitable results from the employed stereo matching algorithm. We settled on block sizes of 15, 19, and 23. In the resulting output depth maps, the disparities of each pixel were mapped to values between 0 and 255.

To compare the depth maps obtained through this process, we converted the depth maps to histograms. We then computed the similarity between the histograms with several functions: correlation, chi-square, intersection, bhattacharyya distance, peak signal-to-noise ratio (PSNR), and NCC.

The first experiment out of two experiments compared the depth map obtained utilizing SC with the depth map obtained without utilizing SC on error-free input images to verify the feasibility of the proposed system. The second experiment repeated the first experiment but with 30% salt-and-pepper noise injected into the input images. The depth maps obtained utilizing SC and without utilizing SC on the noisy images were each compared with the depth map obtained without utilizing SC on noise-free images. This was performed to verify the error tolerance of the proposed system.

### 5.2.2. Feasibility Verification

Figure 14 and Table 2 display the results with noise-free input images. In Figure 14, the top figures show the input images, followed by the depth maps obtained when the block size is 15, 19, and 23, respectively, in order. The left depth maps were obtained without SC, and the right ones with SC. When comparing the two depth maps, they have no significant differences visually.

**Table 2.** The comparison results of depth maps from noise-free input images.

| Input Dataset | Block Size | Comparison Method | | | | | |
|---|---|---|---|---|---|---|---|
| | | A * | B * | C * | D * | PSNR (*db*) | NCC |
| artroom | 15 | 0.925 | 1.759 | 0.810 | 0.216 | 16.395 | 0.866 |
| | 19 | 0.891 | 0.956 | 0.726 | 0.235 | 16.337 | 0.868 |
| | 23 | 0.868 | 0.997 | 0.693 | 0.261 | 16.212 | 0.855 |
| chess | 15 | 0.986 | 2.127 | 0.910 | 0.207 | 12.265 | 0.749 |
| | 19 | 0.985 | 2.167 | 0.908 | 0.236 | 12.759 | 0.792 |
| | 23 | 0.988 | 1.454 | 0.915 | 0.232 | 12.864 | 0.803 |
| djembe | 15 | 0.995 | 0.627 | 0.912 | 0.151 | 17.109 | 0.896 |
| | 19 | 0.997 | 0.260 | 0.943 | 0.137 | 17.904 | 0.910 |
| | 23 | 0.997 | 0.210 | 0.916 | 0.145 | 18.922 | 0.926 |
| newkuba | 15 | 0.953 | 2.876 | 0.896 | 0.228 | 16.846 | 0.853 |
| | 19 | 0.954 | 1.286 | 0.857 | 0.234 | 19.357 | 0.910 |
| | 23 | 0.948 | 0.931 | 0.792 | 0.213 | 21.588 | 0.944 |
| skates | 15 | 0.994 | 1.377 | 0.979 | 0.231 | 13.923 | 0.784 |
| | 19 | 0.997 | 0.992 | 0.957 | 0.213 | 14.413 | 0.808 |
| | 23 | 0.997 | 0.690 | 0.962 | 0.206 | 13.707 | 0.759 |

* A: correlation, B: chi-square, C: intersection, D: bhattacharyya distance.

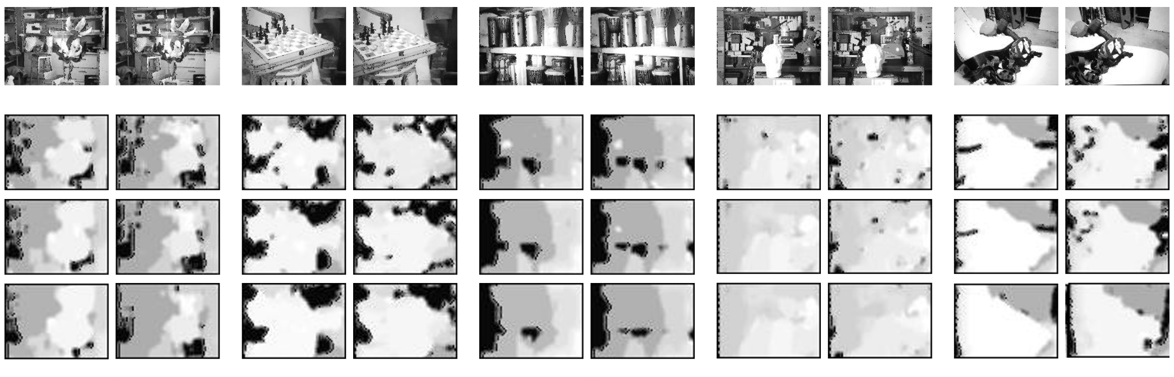

**Figure 14.** The stereo matching results from noise-free input images: (**a**) artroom, (**b**) chess, (**c**) djembe, (**d**) newkuba, (**e**) skates.

Upon a more detailed comparison through the similarity values calculated and presented in Table 2, we notably observed the high similarity. This is particularly true for functions such as correlation, intersection, and NCC, which yield a result of '1.000' when computed between identical images, as shown in Table 3. These high values confirmed a significant degree of similarity between the two depth maps. Notably, in the case of correlation, all similarity values were found to be '0.868' or higher, further substantiating this observation. In Figure 15, the similarity values through the above-mentioned functions are expressed as graphs.

**Table 3.** The similarity values when both images are identical.

| Comparison Method | | | | | |
|---|---|---|---|---|---|
| **Correlation** | **Chi-Square** | **Intersection** | **Bhattacharyya Distance** | **PSNR (*db*)** | **NCC** |
| 1.000 | 0.000 | 1.000 | 0.000 | ∞ | 1.000 |

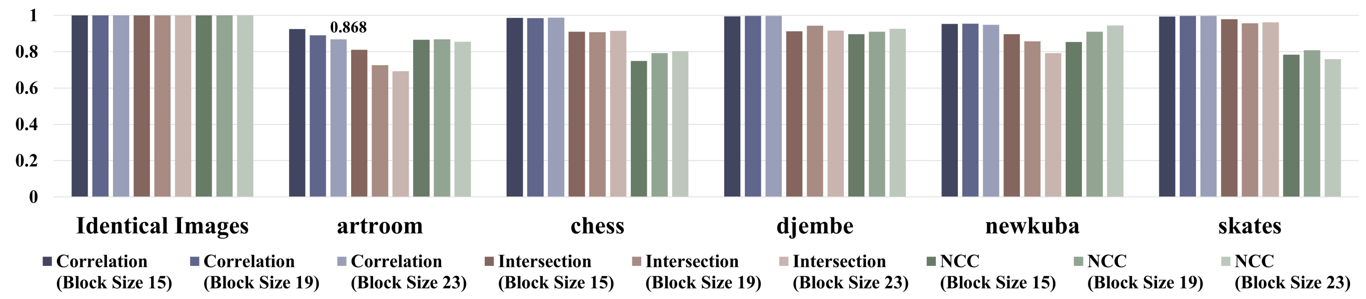

**Figure 15.** Graphs of similarity values between two depth maps obtained from noise-free input images.

Upon analyzing the experimental results in Figures 14 and 15 and Table 2, it is visually observed from Figure 14 that for relatively simple input images such as 'artroom', 'chess', and 'skates', the difference between the depth maps obtained with and without SC is somewhat larger compared to other datasets. This observation is also confirmed through the experimental results displayed in Table 2 and Figure 15. It is inferred that this is due to the fact that the more complex the image being compared in the template matching process (i.e., the more features present), the more advantageous it is to find similar images. Furthermore, through Figure 15, it is confirmed that as the block size increases, the similarity between the two depth maps gradually decreases.

### 5.2.3. Error Tolerance Verification

Figure 16 and Tables 4 and 5 display the results with noise-injected input images. In Figure 16, the top figures illustrate the noise-injected input images, followed by the depth maps obtained with the block sizes. As shown in Figure 14, the left depth maps were obtained without SC, and the right ones with SC. Comparing both depth maps, it is confirmed that more parts remain when the depth map is obtained with SC than when it is without SC, that is, it is more tolerant to the injected errors.

Table 4 shows the similarity values calculated by comparing depth maps obtained without SC from the noise-injected input images and depth maps obtained without SC from the noise-free input images. Conversely, Table 5 shows the similarity values calculated by comparing depth maps obtained with SC from the noise-injected input images and depth maps obtained without SC from the noise-free input images. That is, both depth maps obtained from the noise-injected input images were compared with the same depth map to find out how much improvement is achieved when utilizing SC.

**Table 4.** The comparison results of depth maps without SC from noise-injected input images.

| Input Dataset | Block Size | Comparison Method | | | | | |
|---|---|---|---|---|---|---|---|
| | | **A *** | **B *** | **C *** | **D *** | **PSNR (*db*)** | **NCC** |
| artroom | 15 | 0.125 | 9.448 | 0.065 | 0.800 | 3.890 | 0.285 |
| | 19 | 0.090 | 11.741 | 0.059 | 0.827 | 3.520 | 0.244 |
| | 23 | 0.050 | 22.174 | 0.047 | 0.857 | 3.282 | 0.226 |
| chess | 15 | 0.253 | 4.993 | 0.117 | 0.756 | 3.296 | 0.188 |
| | 19 | 0.289 | 3.950 | 0.144 | 0.743 | 3.114 | 0.152 |
| | 23 | 0.292 | 3.635 | 0.156 | 0.757 | 3.030 | 0.158 |
| djembe | 15 | 0.342 | 3.535 | 0.152 | 0.764 | 4.176 | 0.228 |
| | 19 | 0.293 | 3.693 | 0.145 | 0.775 | 4.197 | 0.347 |
| | 23 | 0.261 | 3.997 | 0.137 | 0.783 | 4.204 | 0.526 |
| newkuba | 15 | 0.034 | 6.100 | 0.044 | 0.812 | 2.920 | 0.246 |
| | 19 | 0.037 | 5.386 | 0.042 | 0.822 | 2.743 | 0.229 |
| | 23 | 0.039 | 5.091 | 0.042 | 0.824 | 2.658 | 0.222 |
| skates | 15 | 0.367 | 26.082 | 0.278 | 0.658 | 4.087 | 0.234 |
| | 19 | 0.379 | 56.550 | 0.298 | 0.676 | 4.417 | 0.439 |
| | 23 | 0.201 | 46.279 | 0.177 | 0.734 | 3.558 | 0.351 |

* A: correlation, B: chi-square, C: intersection, D: bhattacharyya distance.

**Table 5.** The comparison results of depth maps with SC from noise-injected input images.

| Input Dataset | Block Size | Comparison Method | | | | | |
|---|---|---|---|---|---|---|---|
| | | **A *** | **B *** | **C *** | **D *** | **PSNR (*db*)** | **NCC** |
| artroom | 15 | 0.171 | 8.833 | 0.141 | 0.605 | 5.244 | 0.417 |
| | 19 | 0.144 | 11.250 | 0.130 | 0.645 | 4.586 | 0.342 |
| | 23 | 0.098 | 21.865 | 0.106 | 0.707 | 4.068 | 0.285 |
| chess | 15 | 0.342 | 4.695 | 0.255 | 0.542 | 5.162 | 0.445 |
| | 19 | 0.373 | 3.598 | 0.258 | 0.567 | 4.558 | 0.390 |
| | 23 | 0.373 | 3.413 | 0.246 | 0.595 | 4.103 | 0.332 |
| djembe | 15 | 0.153 | 8.474 | 0.068 | 0.885 | 5.307 | 0.240 |
| | 19 | 0.604 | 3.634 | 0.444 | 0.440 | 7.227 | 0.449 |
| | 23 | 0.593 | 3.228 | 0.436 | 0.455 | 7.044 | 0.419 |
| newkuba | 15 | 0.134 | 5.661 | 0.145 | 0.664 | 4.759 | 0.395 |
| | 19 | 0.125 | 4.689 | 0.111 | 0.717 | 4.143 | 0.345 |
| | 23 | 0.137 | 4.526 | 0.102 | 0.745 | 3.877 | 0.326 |
| skates | 15 | 0.595 | 30.309 | 0.691 | 0.475 | 7.383 | 0.568 |
| | 19 | 0.582 | 59.356 | 0.661 | 0.511 | 6.574 | 0.537 |
| | 23 | 0.425 | 46.695 | 0.450 | 0.587 | 5.593 | 0.518 |

* A: correlation, B: chi-square, C: intersection, D: bhattacharyya distance.

Upon comparing Tables 4 and 5, it is evident that the depth maps obtained with SC generally approached the similarity values calculated between identical images. To quantitatively assess the improvement, we calculated improvement rates with a formula: $(Final\ Value - Initial\ Value)/Initial\ Value \times 100\ [\%]$.

The improvement rates of the depth maps for each function, input dataset, and block size were calculated and are presented in Table 6.

In Table 6, the overall average of the improvement rate was approximately 58.95%. The average improvement rates for the block sizes starting from 15 were approximately 51.03%, 62.50%, and 63.33%, respectively. The results indicate a different trend from the previous experiments, where the similarity decreased as the block size increased. This is able to be explained from the fact that the cost matching function, SAD, employed in our work does not consider the position of pixels during calculation. Therefore, while larger block sizes are disadvantageous for performing template matching, the cumulative error in calculation when not using SC for noisy input images increases with block size. This means that it is possible for the degree of improvement when using SC to be made larger. These experimental results allow us to confirm the error tolerance of the proposed system.

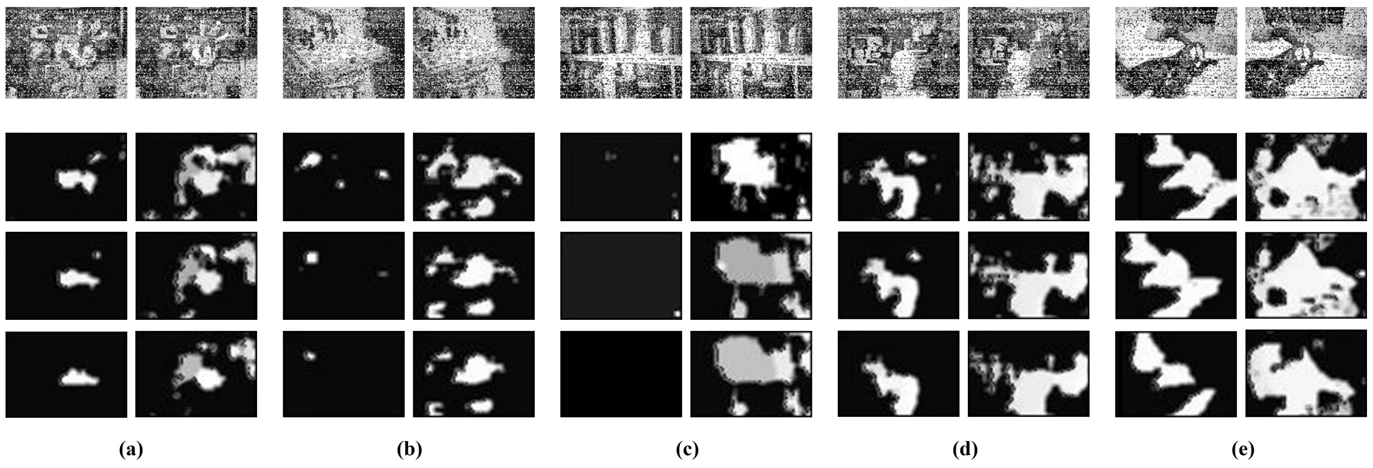

**Figure 16.** The stereo matching results from noise-injected input images: (**a**) artroom, (**b**) chess, (**c**) djembe, (**d**) newkuba, (**e**) skates.

**Table 6.** The improvement rates of depth maps.

| Input Dataset | Block Size | Comparison Method | | | | | |
|---|---|---|---|---|---|---|---|
| | | A * | B * | C * | D * | PSNR (*db*) | NCC |
| artroom | 15 | 36.80% | 6.51% | 116.92% | 24.38% | 34.81% | 46.32% |
| | 19 | 60.00% | 4.18% | 120.34% | 22.01% | 30.28% | 40.16% |
| | 23 | 96.00% | 1.39% | 125.53% | 17.50% | 23.95% | 26.11% |
| chess | 15 | 35.18% | 5.97% | 117.95% | 28.31% | 56.61% | 136.70% |
| | 19 | 29.07% | 8.91% | 79.17% | 23.69% | 46.37% | 156.58% |
| | 23 | 27.74% | 6.11% | 57.69% | 21.40% | 35.41% | 110.13% |
| djembe | 15 | −55.26% | −139.75% | −55.26% | −15.84% | 27.08% | 5.26% |
| | 19 | 106.14% | 1.60% | 206.21% | 43.23% | 72.19% | 29.39% |
| | 23 | 127.20% | 19.24% | 218.25% | 41.89% | 67.55% | −20.34% |
| newkuba | 15 | 294.12% | 7.20% | 229.55% | 18.23% | 62.98% | 60.57% |
| | 19 | 237.84% | 12.94% | 164.29% | 12.77% | 51.04% | 50.66% |
| | 23 | 251.28% | 11.10% | 142.86% | 9.59% | 45.86% | 46.85% |
| skates | 15 | 62.13% | −16.21% | 148.56% | 27.81% | 80.65% | 142.74% |
| | 19 | 53.56% | −4.96% | 121.81% | 24.41% | 48.83% | 22.32% |
| | 23 | 111.44% | −0.90% | 154.24% | 20.03% | 57.20% | 47.58% |

* A: correlation, B: chi-square, C: intersection, D: bhattacharyya distance.

## 6. Conclusions

This work presents a novel approach for lightweight and error-tolerant stereo matching with a hardware-implemented SC processor. SC, which represents and processes data utilizing stochastic sequences and leverages the concept of mathematical probability, replaces traditional arithmetic units with concise logic gates such as AND or MUX. This makes the SC processor lightweight compared to methods that rely on complex 3D CNNs or weighted loss functions used in other works. Furthermore, each digit in a stochastic sequence has the same weight in SC, so even if an error occurs in any bit of the stochastic sequence, the impact on the output is relatively small. This makes SC more error-tolerant compared to methods that may struggle with disparity estimation in occluded regions or unsupervised training.

In this work, we have successfully implemented a system that leverages the characteristics of SC to obtain depth maps through stereo matching operations. For this purpose, we designed an SC processor with Verilog HDL and implemented it on an FPGA. The implemented SC processor includes an SC unit for SC operations, which contains a stochastic number generator composed of a random number generator and a comparator to generate stochastic sequences. The random number generator is designed as a 256-stage parallel LFSR capable of generating 256 random numbers within one clock cycle. This makes it more efficient compared to methods that require a balance between disparity estimation accuracy and efficiency [37–39].

We conducted experiments to verify the feasibility of the proposed system by obtaining depth maps from noise-free input images and to validate its error tolerance by obtaining depth maps from noise-injected images. The experimental results demonstrated that obtaining depth maps from noise-injected input images with SC improved by an average of 58.95% compared to without SC. This indicates that the proposed system exhibits error tolerance without significantly degrading the quality of the output depth map.

The system is expected to be applicable in environments where errors are likely to occur due to instability, such as inside vehicles, and where available resources are limited [40]. However, the system proposed in this paper did not demonstrate an operation time fast enough to be suitable for real-time applications during the experimental process. This is able to be attributed to various delays occurring within the system, such as the delay in data movement through the bus and the delay in re-converting the stochastic sequence into a binary number. Therefore, in future work, it will be necessary to conduct research to implement a system in which it is possible to be applied not only to images but also to video applications. This could be achieved by directly mounting the SC module inside the processor core to reduce the delay from data movement or by further optimizing the architecture of the SC module to reduce delay. This will enhance the versatility and applicability of the system in various real-world scenarios.

**Author Contributions:** Conceptualization, S.A.; methodology, S.L., Y.J. and J.K. (Jeongeun Kim); hardware, J.K. (Jeongeun Kim); software, S.A. and J.K. (Jeongeun Kim); validation, J.O.; formal analysis, S.L.; investigation, S.A., J.O., S.L. and J.K. (Jinyeol Kim); data curation, S.A. and J.O.; writing—original draft preparation, S.A.; writing—review and editing, S.A., J.O., S.L. and Y.J.; visualization, S.L. and J.K. (Jinyeol Kim); supervision, S.E.L. All authors have read and agreed to the published version of the manuscript.

**Funding:** This study was supported by SeoulTech (Seoul National University of Science and Technology).

**Data Availability Statement:** The data presented in this study are available in this article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| SC | stochastic computing |
| AR | augmented reality |
| LFSR | linear feedback shift register |
| FPGA | Field-Programmable Gate Array |
| MUX | multiplexer |
| SAD | sum of absolute difference |
| SSD | sum of squared difference |
| NCC | normalized cross correlation |
| ESL | Extended Stochastic Logic |
| ANN | artificial neural network |
| NN | neural network |
| CNN | convolutional neural network |
| ASC | approximate stochastic computing |

| | |
|---|---|
| SNG | stochastic number generator |
| RNG | random number generator |
| PE | probability estimator |
| UART | universal asynchronous receiver/transmitter |
| PSNR | peak signal-to-noise ratio |

## References

1. Domínguez-Morales, M.J.; Jiménez-Fernández, Á.; Jiménez-Moreno, G.; Conde, C.; Cabello, E.; Linares-Barranco, A. Bio-Inspired Stereo Vision Calibration for Dynamic Vision Sensors. *IEEE Access* **2019**, *7*, 138415–138425. [CrossRef]
2. Hallek, M.; Boukamcha, H.; Smach, F.; Atri, M. Real Time Stereo Matching Using Two Step Zero-Mean SAD and Dynamic Programing. In Proceedings of the 2018 15th International Multi-Conference on Systems, Signals & Devices (SSD), Yasmine Hammamet, Tunisia, 19–22 March 2018; pp. 1234–1240. [CrossRef]
3. Lin, X.; Wang, J.; Lin, C. Research on 3D Reconstruction in Binocular Stereo Vision Based on Feature Point Matching Method. In Proceedings of the 2020 IEEE 3rd International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 27–29 September 2020; pp. 551–556. [CrossRef]
4. Liu, S.; Gross, W.J.; Han, J. Introduction to Dynamic Stochastic Computing. *IEEE Circuits Syst. Mag.* **2020**, *20*, 19–33. [CrossRef]
5. Shivanandamurthy, S.M.; Thakkar, I.G.; Salehi, S.A. A scalable stochastic number generator for phase change memory based in-memory stochastic processing: Work-in-progress. In Proceedings of the Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis Companion, New York, NY, USA, 13–18 October 2019. [CrossRef]
6. Kim, J.; Jeong, W.S.; Jeong, Y.; Lee, S.E. Parallel stochastic computing architecture for computationally intensive applications. *Electronics* **2023**, *12*, 1749. [CrossRef]
7. Chen, K.C.; Wu, C.H. High-Accurate Stochastic Computing for Artificial Neural Network by Using Extended Stochastic Logic. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–4. [CrossRef]
8. Jang, S.Y.; Yoon, Y.H.; Lee, S.E. Stochastic Computing based AI System for Mobile Devices. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 4–6 January 2020; pp. 1–2. [CrossRef]
9. Temenos, N.; Sotiriadis, P.P. Modeling a Stochastic Computing Nonscaling Adder and its Application in Image Sharpening. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 2543–2547. [CrossRef]
10. Kang, S.; Pan, R.; Zhao, K.; Dong, J.; Zhao, Y. Implementation of Stochastic Computing-based Image Compression System Using Probabilistic Switching Behavior of RRAM. In Proceedings of the 2022 15th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Beijing, China, 5–7 November 2022; pp. 1–6. [CrossRef]
11. Kim, J.; Jeong, Y.R.; Cho, K.; Jeong, W.S.; Lee, S.E. Reconfigurable Stochastic Computing Architecture for Computationally Intensive Applications. In Proceedings of the 2022 19th International SoC Design Conference (ISOCC), Gangneung-si, Republic of Korea, 19–22 October 2022; pp. 61–62. [CrossRef]
12. Ma, C.; Zhong, S.; Dang, H. High Fault Tolerant Image Processing System Based on Stochastic Computing. In Proceedings of the 2012 International Conference on Computer Science and Service System, Nanjing, China, 11–13 August 2012; pp. 1587–1590. [CrossRef]
13. Zhang, Z.; Wang, R.; Zhang, Z.; Zhang, Y.; Guo, S.; Huang, R. Circuit Reliability Comparison Between Stochastic Computing and Binary Computing. *IEEE Trans. Circuits Syst. II Express Briefs* **2020**, *67*, 3342–3346. [CrossRef]
14. Liu, Z.; Song, B.; Guo, Y.; Xu, H. Improved Template Matching Based Stereo Vision Sparse 3D Reconstruction Algorithm. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 4305–4310. [CrossRef]
15. Perri, S.; Frustaci, F.; Spagnolo, F.; Corsonello, P. Design of Real-Time FPGA-based Embedded System for Stereo Vision. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5. [CrossRef]
16. Zhao, J.; Liang, T.; Feng, L.; Ding, W.; Sinha, S.; Zhang, W.; Shen, S. FP-Stereo: Hardware-efficient stereo vision for embedded applications. In Proceedings of the 2020 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, 31 August–4 September 2020; IEEE: New York, NY, USA, 2020; pp. 269–276.
17. Choi, C.H.; Kim, Y.; Ha, J.; Moon, B. Haar Filter Hardware Architecture for the Accuracy Improvement of Stereo Vision Systems. In Proceedings of the 2021 18th International SoC Design Conference (ISOCC), Jeju Island, Republic of Korea, 6–9 October 2021; pp. 401–402. [CrossRef]
18. Gani, S.F.A.; Miskon, M.F.; Hamzah, R.A. Depth Map Information from Stereo Image Pairs using Deep Learning and Bilateral Filter for Machine Vision Application. In Proceedings of the 2022 IEEE 5th International Symposium in Robotics and Manufacturing Automation (ROMA), Malacca, Malaysia, 6–8 August 2022; pp. 1–6. [CrossRef]
19. Chen, F.; Liu, X.; Yu, H.; Ha, Y. CLIF: Cross-Layer Information Fusion for Stereo Matching and its Hardware Implementation. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5. [CrossRef]
20. Yang, L.; Wang, B.; Zhang, R.; Zhou, H.; Wang, R. Analysis on Location Accuracy for the Binocular Stereo Vision System. *IEEE Photonics J.* **2018**, *10*, 1–16. [CrossRef]

21. Jia, T.; Ma, J.; Li, W.; Zhang, Y.; Zeng, Z.; Huang, J. Implementation of Real-Time Stereo Matching System Based on Speckle Structured Light. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; pp. 1537–1542. [CrossRef]
22. George, G.; Oommen, R.M.; Shelly, S.; Philipose, S.S.; Varghese, A.M. A Survey on Various Median Filtering Techniques For Removal of Impulse Noise From Digital Image. In Proceedings of the 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), Tiruchengode, India, 2–3 March 2018; pp. 235–238. [CrossRef]
23. Zhang, Y.; Lin, S.; Wang, R.; Wang, Y.; Wang, Y.; Qian, W.; Huang, R. When Sorting Network Meets Parallel Bitstreams: A Fault-Tolerant Parallel Ternary Neural Network Accelerator based on Stochastic Computing. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1287–1290. [CrossRef]
24. Hu, A.; Li, W.; Lv, D.; He, G. An Efficient Stochastic Convolution Accelerator Based on Pseudo-Sobol Sequences. In Proceedings of the 17th ACM International Symposium on Nanoscale Architectures, Virtual, OR, USA, 7–9 December 2022; pp. 1–6.
25. Joe, H.; Kim, Y. Novel stochastic computing for energy-efficient image processors. *Electronics* **2019**, *8*, 720. [CrossRef]
26. Seva, R.; Metku, P.; Kim, K.K.; Kim, Y.B.; Choi, M. Approximate stochastic computing (ASC) for image processing applications. In Proceedings of the 2016 International SoC Design Conference (ISOCC), Jeju, Republic of Korea, 23–26 October 2016; pp. 31–32. [CrossRef]
27. Estiri, S.N.; Jalilvand, A.H.; Naderi, S.; Najafi, M.H.; Fazeli, M. A Low-Cost Stochastic Computing-based Fuzzy Filtering for Image Noise Reduction. In Proceedings of the 2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC), Pittsburgh, PA, USA, 24–25 October 2022; pp. 1–6. [CrossRef]
28. Li, P.; Lilja, D.J. Using stochastic computing to implement digital image processing algorithms. In Proceedings of the 2011 IEEE 29th International Conference on Computer Design (ICCD), Amherst, MA, USA, 9–12 October 2011; pp. 154–161. [CrossRef]
29. Jeong, Y.; Oh, H.W.; Kim, S.; Lee, S.E. An Edge AI Device based Intelligent Transportation System. *J. Inf. Commun. Converg. Eng.* **2022**, *20*, 166–173. [CrossRef]
30. Cho, K.N.; Oh, H.W.; Lee, S.E. Vision-based Parking Occupation Detecting with Embedded AI Processor. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 10–12 January 2021; pp. 1–2. [CrossRef]
31. Laskin, E. *On-Chip Self-Test Circuit Blocks for High-Speed Applications*; University of Toronto: Toronto, ON, Canada, 2006.
32. Salehi, S.A. Low-Cost Stochastic Number Generators for Stochastic Computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 992–1001. [CrossRef]
33. Frasser, C.F.; Linares-Serrano, P.; Ríos, I.D.d.l.; Morán, A.; Skibinsky-Gitlin, E.S.; Font-Rosselló, J.; Canals, V.; Roca, M.; Serrano-Gotarredona, T.; Rosselló, J.L. Fully Parallel Stochastic Computing Hardware Implementation of Convolutional Neural Networks for Edge Computing Applications. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 10408–10418. [CrossRef] [PubMed]
34. Lin, Z.; Xie, G.; Xu, W.; Han, J.; Zhang, Y. Accelerating Stochastic Computing Using Deterministic Halton Sequences. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 3351–3355. [CrossRef]
35. Scharstein, D.; Hirschmüller, H.; Kitajima, Y.; Krathwohl, G.; Nešić, N.; Wang, X.; Westling, P. High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth. In *Pattern Recognition*; Jiang, X., Hornegger, J., Koch, R., Eds.; Springer: Cham, Switzerland, 2014; pp. 31–42.
36. Park, J.; Shin, J.; Kim, R.; An, S.; Lee, S.; Kim, J.; Oh, J.; Jeong, Y.; Kim, S.; Jeong, Y.R.; et al. Accelerating Strawberry Ripeness Classification Using a Convolution-Based Feature Extractor along with an Edge AI Processor. *Electronics* **2024**, *13*, 344. [CrossRef]
37. Poggi, M.; Pallotti, D.; Tosi, F.; Mattoccia, S. Guided stereo matching. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 979–988.
38. Bi, W.; Chen, M.; Wu, D.; Lu, S. EBStereo: Edge-based loss function for real-time stereo matching. *Vis. Comput.* **2023**, *40*, 2975–2986. [CrossRef]
39. Lee, S.H.; Kanatsugu, Y.; Park, J.I. MAP-based stochastic diffusion for stereo matching and line fields estimation. *Int. J. Comput. Vis.* **2002**, *47*, 195–218. [CrossRef]
40. Han, C.Y.; Jeong, Y.S.; Lee, S.E. Simulation-Based Fault Analysis for Resilient System-On-Chip Design. *J. Inf. Commun. Converg. Eng.* **2021**, *19*, 175–179. [CrossRef]