Error-Tolerant Image Processing Application Based On Stochastic Logic
Checkpoint2 Report
Team Silicon

- **Problem to be addressed**

  Aggressive scaling of semiconductor devices exacerbates issues of reliability. High density of devices leads to more probable occurrences of hard errors during manufacture. In addition, soft errors including particle striking and ray radiation will also have a larger impact on high-density chips compared to chips with less progressive scaling and lower device densities. A particle striking on the chip, for example, can cause a bit flip. Therefore, it is important to develop a method to enhance the reliability of the system.

- **Why does this problem matter?**

  The modern techniques that we adapt now are limited in ensuring absolute reliability. An error-tolerant system is in urgent need to eliminate various external causes of errors. A single transient fault could cause the whole system to perform erroneously. Unreliability in systems can lead to huge losses in all fields, including economic loss of company as well as potential hazards in real-world applications.

- **Idea/solution to be investigated by the project**

  Our idea is to use stochastic logics to implement image processing hardware such as sharpening/smoothing filters and make it more error-tolerant against transient faults.

  Stochastic logic has the nature of error-tolerance and can create analog signal states in digital-based systems. In stochastic logic, a normal input will take advantage of randomness and be transformed to a bit stream. The bit streams or wire bundles are digital, carrying zeros and ones, and the signal is conveyed through the statistical distribution of logical values.

  Advantages of the stochastic logic:

  1. Error tolerance. A single binary value is presented with bit stream which reduces the impact of a single transient fault. In normal operations, if the significant bit of one value is corrupted, it is likely to have a large impact on the result. However, in the stochastic logics, the impact is minimized by equally distributing value weight to all bits in the bit stream.
  2. Fast computation in specific applications. While the stochastic creates more bits than traditional logics, in some cases it will be faster by taking advantage of statistic concept. For example in Figure 1, the multiplication $y = x_1 \cdot x_2$ in directly computed by using a simple AND gate. Another example is scaled addition shown in Figure 2. Using a single MUX, the stochastic logic can perform a weighted average easily.

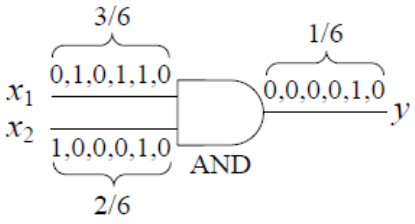


Figure 1. Multiplication in Stochastic Logic

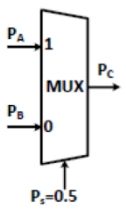


Figure 2. Scaled Addition

➢ Progress so far

1. We finished the design of Sharpening/Smoothing filter. The stochastic filter design itself is trial in the system. It involves only MUX's (8 to 1 and 2 to 1) to calculate scaled addition/subtraction.

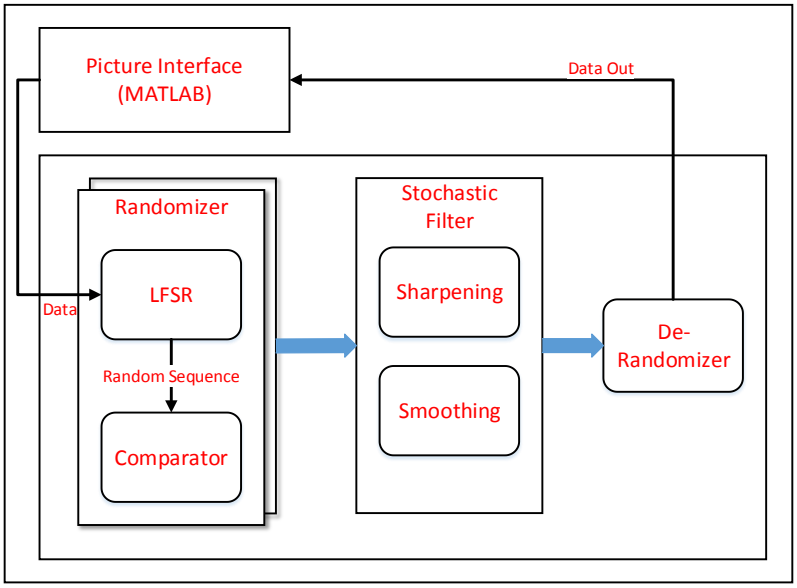
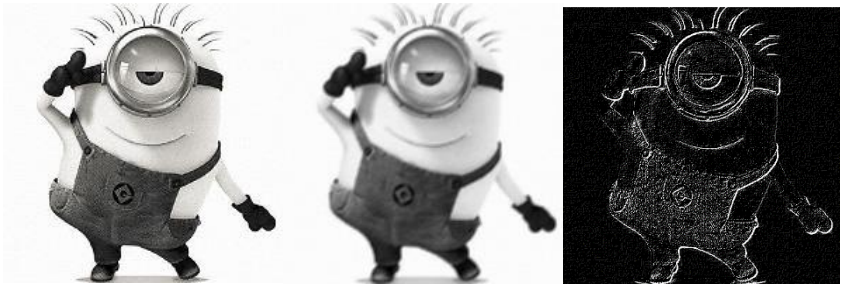
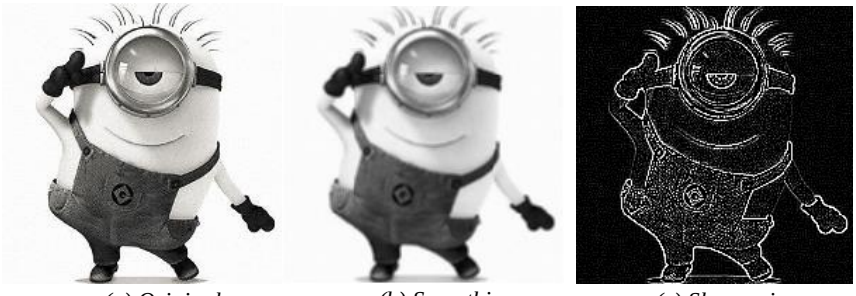


Figure 3. Stochastic Logic Architecture



(a) Original *(b) Smoothing* *(c) Sharpening*

Figure 4. Smoothing and Sharpening Results with Stochastic filter



(a) Original *(b) Smoothing* *(c) Sharpening*

Figure 5. Smoothing and Sharpening Results with Custom Baseline filter

2. Evaluate clock period/synthesis area

   We evaluated speed/area of the two designs. Based on the comparison in Table 1, the clock period and area of stochastic design are both smaller than baseline design. However, the stochastic design will need more cycles to execute (2^8 in our case). After further optimization, we managed to keep the Period∗Area∗#Cycle roughly the same for baseline design and stochastic design (if only count the core module).

Table 1. Comparison of Synthesis Results

	Baseline	Stochastic	Stochastic (core only)
Clock Period (ns)	7.8	1.8	1.8
Synthesis Area (μm^2)	49687	26796	1063
Total Cycle	N	$N \cdot 2^8$	$N \cdot 2^8$
Period∗Area∗#Cycle	387559N	12347596N	489830N

➢ Future works

1. Error-tolerant testing
   - We plan to inject errors manually and observe the resulting pictures. To compare the erroneous pictures with original picture, we need a method to evaluate the difference between the pictures. We plan to use SSIM and PSNR metrics for evaluation.
   - The PSNR block computes the peak signal-to-noise ratio, in decibels, between two images. The higher the PSNR, the better the quality of the compressed, or reconstructed image. A PSNR of 50-70dB is considered a good image.
   - The Structural SIMilarity (SSIM) index is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of ideal quality. An SSIM index of 1 implies the two images are equal and any value from 0.7 to 1 is considered a good quality image.
2. Result Analysis

➢ Issues

1. Do we need to compare image similarity processed by Stochastic/baseline design? These two approaches have similar algorithms but differences will occur due to errors such as quantization error.
2. In the error-tolerant test, do we just randomly flip pixel bits or do we have to create some certain noise such as Salt-and-pepper noise?

Figure 6. An Image With Salt-and-pepper Noise