Towards Spatial Fault Resilience in Array Processors

Suraj Sindia* Vishwani D. Agrawal†
Department of Electrical and Computer Engineering
Auburn University, Alabama, AL 36849 USA
*Email: szs0063@auburn.edu †Email: vagrawal@eng.auburn.edu

***Abstract*—Computing with large die-size graphical processors (that need huge arrays of identical structures) in the late CMOS era is abounding with challenges due to spatial non-idealities arising from chip-to-chip and within-chip variation of MOSFET threshold voltage. In this paper, we propose a machine learning based software-framework for in-situ prediction and correction of computation corrupted due to threshold voltage variation of transistors. Based on semi-supervised training imparted to a fully connected cascade feed-forward neural network (FCCFF-NN), the NN makes an accurate prediction of the underlying hardware, creating a spatial map of faulty processing elements (PE). The faulty elements identified by the NN are avoided in future computing. Further, any transient faults occurring over and above these spatial faults are tracked, and corrected if the number of PEs involved in a particle strike is above a preset threshold. For the purposes of experimental validation, we consider a 256×256 array of PE. Each PE is comprised of a multiply-accumulate (MAC) block with three 8 bit registers (two for inputs and one for storing the computed result). One thousand instances of this processor array are created and PEs in each instance are randomly perturbed with threshold voltage variation. Common image processing operations such as low pass filtering and edge enhancement are performed on each of these 1000 instances. A fraction of these images (about 10%) is used to train the NN for spatial non-idealities. Based on this training, the NN is able to accurately predict the spatial extremities in 95% of all the remaining 90% of the cases. The proposed NN based error tolerance results in superior quality images whose degradation is no longer visually perceptible.**

I. Introduction

Scaling of MOS transistor dimensions (thanks to Moore's law) has led to a steady increase in functions offered by microprocessor chips. Additionally, the performance (or speed) offered by these scaled devices has also been exponentially increasing. The unprecedented growth in performance of computers, however, has come at a price of an exponential increase in power density (power per unit area). After a point, roughly starting from the later half of the last decade, manufacturers have restrained from increasing the operating frequency of microprocessor chips. This stalling in frequency has prompted microprocessor industry to shift to an alternative computing paradigm such as parallel computing, where individual computers perform at a slower rate, but manage to accomplish functional tasks concurrently to be counted as an individual computer operating at a much faster rate (being roughly equal to number of parallel processors $\times$ operating frequency of individual processor).

Another possible route to mitigate the increase in power density with successive generations of a microprocessor chip, without stalling frequency scaling, is to downscale the supply voltage. Such a scaling model is popularly referred to as constant electric field scaling [5].

Regardless of the route taken to minimize power density to keep up the performance gains, the semiconductor industry is beginning to hit a road-block attributed to increased manufacturing process related variations. Reference [1] presents an insightful discussion on the trends in frequency and voltage scaling in the face of increased process variation in advanced CMOS technology nodes. Table I (reproduced from [1]) shows scaling trends in CMOS and its impact on energy, and speed in the advanced CMOS nodes. It predicts variability in transistor characteristics, both within-die and across dice as the single most important impediment to performance gains in highly scaled CMOS nodes. Variability in transistor characteristics within the chip has led to a few gates (also referred to in literature as "outliers"), located at spatially disjoint locations to offer delays that are significantly higher, and in many cases, these "outlier" gates lie on the critical path, or paths that would nominally (without any process variation) have had delays that are comparable to critical path delay. Presence of an "outlier" on critical path or close to critical-path leads to an abrupt increase in the delay offered by these paths, consequently reducing the maximum operable frequency at which functionality of the circuit is guaranteed to be correct. However, if one can trade functionality for speed, that is, under the assumption that only a few paths may have these "outliers," then we should still be able to operate the circuit at its maximum speed (as if there were no process variation) alibi with a few errors. The difficulty, however, is that with the advancement of processor nodes, errors due to device variability and transient failures in processing element (PE) may increase to such an extent that for most applications, including image processing where error tolerance techniques have been studied [2], [3], [4], [10], the quality of results obtained may not be satisfactory.

The objectives of this work are to–

- Evaluate the impact of such an "un-guaranteed performance" when we operate circuits at a rate faster than they assuredly can–on common applications in image processing such as low pass filtering, and edge detection.
- Propose an error resilience mechanism based on neural networks (NN) to identify faulty processing elements (PE), and avoid these PE completely from computing all future results if they are degraded beyond a pre-specified threshold.

The paper is organized as follows. Section II describes the recent efforts in literature where process variation noise in hardware is considered and schemes for tolerance, resilience, and/or mitigation of the same are proposed. Section III de-

 978-1-4673-1074-1/12/$31.00

TABLE I
TECHNOLOGY SCALING PREDICTIONS FOR THE END-OF-CMOS ERA [1]. MANUFACTURING PROCESS VARIATION IS PROJECTED AS THE SINGLE BIGGEST IMPEDIMENT FOR PERFORMANCE AND ENERGY IMPROVEMENT WITH DEVICE SCALING.

High Volume Manufacturing	2006	2008	2010	2012	2014	2016	2018
Technology node	65	45	32	22	16	11	8
Integration capacity	4	8	16	32	64	128	256
Delay= $\frac{CV}{I}$ scaling	≈ 0.7	> 0.7	Delay scaling will slow down				
Energy/Logic Op scaling	> 0.5	> 0.5	Energy scaling will slow down				
Variability	Medium		High		Very High		

scribes the impact of process related threshold voltage variation on CMOS gate delay. Details of the signal processing fabric built to model process variation and a numerical model to capture functional violations due to process variation is presented in Section IV. Section V presents a comparison of image quality obtained by performing common image processing tasks on this signal processing fabric–with, and without, PV degradation. Section VI discusses the proposed neural network based on-line error tolerance scheme for avoiding PE that are degraded beyond a pre-specified threshold. We conclude in Section VII.

II. RELATED WORK

As we saw in the previous section, manufacturing variations in the device tends to slow down a circuit/gate. However, if we choose to operate transistors at the same speed disregarding the prevalent process variation, then the functionality of the circuit in question is no longer guaranteed to be correct. Seminal work on error tolerance under large scale defects and process variation is presented in [2], where in error tolerance techniques such as error-free operation without reconfiguration for high volume applications, error-free operation for all applications through reconfiguration, and error-free operation with reconfiguration and/or degraded performance/capability for high-volume applications are discussed. In [3], [4] authors propose error tolerant design and test schemes for multimedia applications such as image compression and motion estimation is videos. "Soft DSP" techniques such as algorithmic noise tolerance (ANT) [6], [10], which computes the result factoring the prevalent noise, has been proposed to alleviate the impact of degradation in results arising from process variations in advanced CMOS technologies. Another model used for emulating process variation induced or other defects in CMOS circuits using a probabilistic switching framework is due to Palem [7]. In a recent paper [9], we proposed the use of non-linear median filters for off-line image restoration of images degraded by processing on a PV degraded hardware.

The work presented in this paper differs from the current literature on two counts. First, while most of the prevalent work on error-tolerance in image processing has targeted applications where only the relative values of pixels are important. For example, in motion estimation [4] the fact that there is a difference among pixel values is more important than the actual difference. We target image processing applications where the *actual difference is as important as the fact that there exists a difference*. For example, we consider low pass filtering based on spatial convolution of pixel values with a low pass filter mask (see Section V) where in convolution of the image with the mask can accumulate errors over the entire operation. Second, most of the error tolerance schemes use some form of spatial redundancy of all the computing elements, whereas in this work we propose error tolerance with little redundant hardware by relying on "error avoidance" by eliminating the use of faulty PE if they are degraded beyond a pre-specified threshold, and re-using PE that are not as degraded. Elements that are faulty beyond a threshold are identified on-line and continuously by a neural network (NN), thereby working equally well for transient errors due to a PE failing for a brief period of time before recovering. The NN itself can operate on a small footprint additional hardware that can be degraded as the original circuit.

III. MODELING PROCESS VARIATION

A. MOSFET Threshold Voltage Variation

Process variation is a term used in the very large scale integration (VLSI) literature to refer to random local variation of characteristics of two or more transistors that are on the same die that are ideally expected to have identical characteristics. Sources of random variation in an integrated circuit, include (but are not limited to) fluctuation in the number of dopant atoms in the channel of metal oxide semiconductor (MOS) transistor, and edge roughness of the laser beam used in lithography. These manufacturing process variations are effectively captured at the MOS transistor device level as variation in the threshold voltage (V_{th}) of the MOS transistors. Recent research [14] has shown that the threshold voltage variation can be modeled as a normal distribution, $N\left(\mu_{Vth}, \sigma^2_{Vth}\right)$ with variance (σ^2_{Vth}), normalized with respect to its mean value is given by

$$\frac{\sigma_{Vth}}{\mu_{Vth}} = \frac{K}{\sqrt{WL}} \tag{1}$$

where K is a proportionality constant that depends on oxide thickness and doping concentration and W and L are width and length of MOS transistor. Plugging in typical numbers for all quantities above, assuming 90nm technology, we have $K = 8.797 \times 10^{-9}$m, $L = 32$nm, $W = 130$nm, which results in a normalized threshold voltage variance of $\frac{\sigma_{Vth}}{\mu_{Vth}}$ **= 13.5%**.

B. Impact of Threshold Voltage Variation on Circuit Functionality and Performance

Delay t_d, offered by a logic gate constructed using MOS transistors is related to threshold voltage of its constituent transistors as follows

$$t_d = \frac{V_{DD}}{(V_{DD} - V_{th})^{\alpha}} t_{D0} \tag{2}$$

where t_{D0} is delay offered by a gate constructed using zero threshold voltage transistors (ns)
V_{DD} is supply voltage (volt)
V_{th} is threshold voltage of MOS transistor (volt)
α is MOS device parameter, value is between 1 and 2. For more advanced technologies, this parameter is closer to 1.

If there is variation in threshold voltage, that can be modeled by a Gaussian random variable as described in Section III-A,

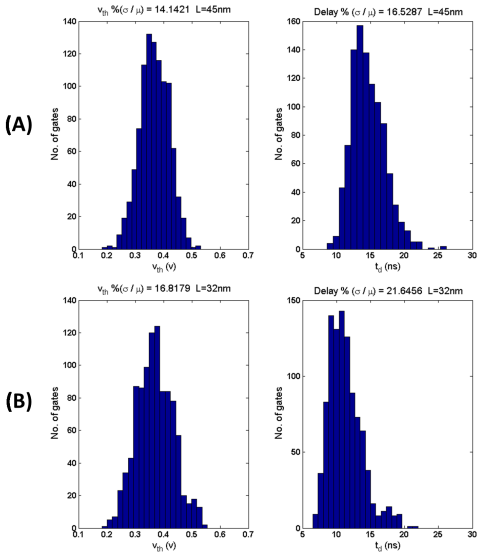

Fig. 1. Histogram of threshold voltage and delay for channel lengths (A) L = 45nm, and (B) L = 32nm.

then the distribution in t_d follows a distribution, that can be obtained by using the random variable transformation specified in equation (2). Plots in Figures 1 show the histogram of threshold voltage (V_{th}) and maximum delay (t_d) offered by 1000 instances of an inverter built in L = 45nm and 32nm. It can be noticed from these plots that the variation in delay is progressively degrading as the technology is advancing. This variation in threshold voltage and delay as a function of technology is summarized in the plot shown in Figure 2. With the passage of technology, variability in transistors is consistently worsening the delay distribution of transistors. That is to say a bigger fraction of transistors will fall outside the acceptable margins of delay. This has a direct consequence of increasing the critical path delays of circuits built in high variability semiconductor processes. To ensure that the circuit functions correctly, one has to reduce the maximum frequency at which the circuit is operated. This leads to a loss in performance/speed offered by the circuit.

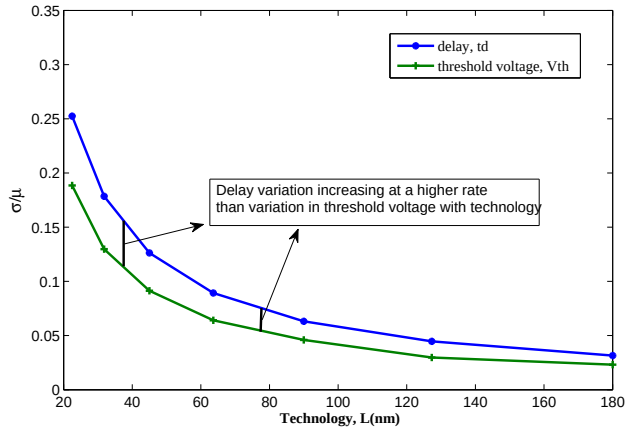

Fig. 2. Variation of delay and threshold voltage as a function of MOS channel length. Notice that as transistors get smaller, the normalized variability in delay offered by a logic gate (here an inverter) is increasing.

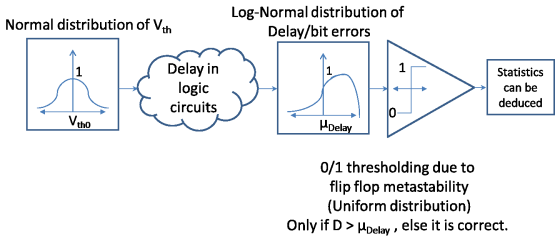

Fig. 3. Block diagram showing the PDFs of different random variables as we traverse different levels of abstraction, starting from transistors to software (or algorithmic level).

IV. Less than Accurate Computing

A. Putting Together the Basic Building Blocks

Addition and multiplication are two basic operations needed for most image processing tasks. For example, convolution, which is a commonly used operation involves a series of multiply-accumulate operations of its two inputs. We build 8 bit addition and multiplication units at a logic gate level, while incorporating faulty behavior as illustrated in Figure 3. Each bit location is computed through a cascaded chain of logic gates. The delay of the gates in each of these logic chains is sampled from Figure 1, depending on the technology used for hardware emulation. Process variation noise added output, y, can be related to its input, x, by a non-linear function f,

$$y = f(x) \tag{3}$$

where

$$x = b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0 \qquad y = b'_7\, b'_6\, b'_5\, b'_4\, b'_3\, b'_2\, b'_1\, b'_0 \tag{4}$$

and in general, for all $i = 1 \cdots 7$

$$b'_i = \begin{cases} b_i & \text{for} \quad t_d \leq t_{d,th} \\ 0\, or\, 1 & \text{for} \quad t_d > t_{d,th} \end{cases} \qquad \text{with equal probability} \tag{5}$$

where t_d is the actual delay offered by the bit line and $t_{d,th}$ is the threshold delay beyond which the bit line enters a meta-stable state, and the final value that the line settles to, can be either 0 or 1 with equal likelihood. The number chosen for $t_{d,th}$ is usually the delay offered by a gate that has its threshold voltage at 3 times the σ_{Vth} away from the mean value V_{tho} given by

$$t_{d,th} = t_d|_{Vth=Vtho+3\sigma_{Vth}} \tag{6}$$

B. Software Emulation

We build a signal processing fabric to deal with images of size 256×256 pixels. PV is captured in software as follows: We emulate a 256 × 256 array of 8-bit *add-store* units. Each *add-store* unit in turn consists of cascaded chain of 1 bit full adders to function as 8-bit adder with their outputs connected to 8-bit register as shown in Figure 4. The full adder consists of AND-OR logic gates, whose delays are sampled from delay distribution described in Section IV-A. We use these arrays of *add-store* units to perform eight bit arithmetic operations such as addition and subtraction. Multiplication is achieved by repeated addition. Further, if any arithmetic operation results in a value in excess of 255, the *add-store*

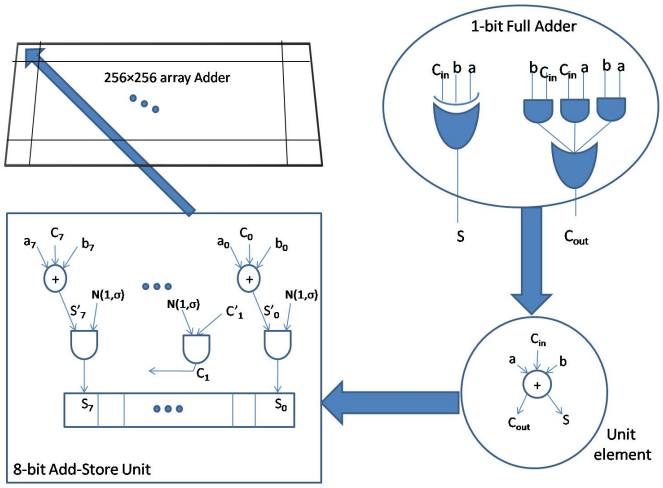

Fig. 4. Synthesis of 256×256 MAC array, with PV noise added for software emulation of PV degraded hardware.

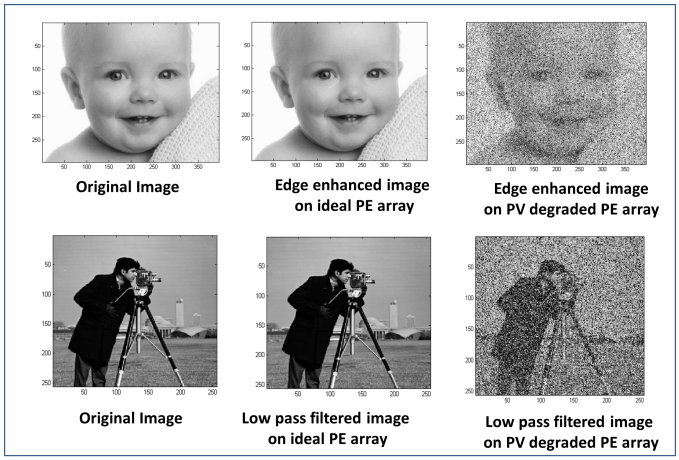

Fig. 5. Comparison of "Camera-man" and "baby-face" image with processing–low pass filtering with mask ξ on "Camera-man" and edge enhancement with HP mask ψ on "baby-face"–carried out on hardware with, and without process variation. Process variation noise added is equivalent to the model developed for 32nm technology node (delay $\sigma/\mu = 13.5\%$).

elements are designed to saturate to 255, thus mimicking real-world scenario where 8-bit gray scale images have a maximum intensity level of 255. For simplicity, we do not use color images and restrict ourselves to the use of gray scale images in all our experiments, as we shall see in the sequel.

V. Image Degradation in Low Pass Filtering and Edge Enhancement

We conducted a low pass filter operation on two test images "cameraman" and "baby face". For both the test images, we repeated the experiment with and without process variation. For low pass filtering, we used the mask:

$$\xi = .25 \times \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \tag{7}$$

From both the images that we subjected to low pass filtering on the processor with process variation noise added, we find that the resulting images tend to have an increased concentration of scattered white and black spots (similar to salt and pepper noise). We also notice that the concentration of white saturated points is more than black points. This can be reasoned out as follows: Low pass filtering which is averaging tends to increase the low level intensities. If there are any MSB bit flips to zero (making the value smaller than what it should be), in the early portions of the processing, their effect gets alleviated over subsequent steps of filtering. However, if there is a MSB bit flip to a 1, this effect tends to be cumulative, often giving rise to an increased concentration of bright spots.

Edge enhancement is an image processing operation where the spatial discontinuities in pixel values of an image are computed, and they are then added to the original image to emphasize these discontinuities; the resulting image is usually a sharper-looking version of the original image. Figure 5 shows edge enhanced image obtained from processing without and with process variation. The degree of sharpness itself depends on the fraction of the edge information added to the original image. Notice the pronounced accumulation of white spots in the edge enhanced image due to the cumulative effect of bits flipping to 1. For high pass (HP) filtering, which is an intermediate step to find the edges in the original image, we use the spatial HP filter mask:

$$\psi = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{8}$$

VI. Error Tolerance Using Neural Network

Neural networks are known to be inherently resilient [8] to faults in their basic computing elements–neurons. This served as the motivation for the use of neural networks for identifying faulty pixels, since the hardware used for building the NN is also assumed to have the same level of process variation as the PE. In the following sections, we shall describe the neural network architecture used for on-line identification of faulty pixel locations, data set and training of NN, and the validation phase of the proposed neural network for different levels of process variation noise and different sets of input images used.

A. Neural Network Architecture

The neural network serves to decide the pixel locations that are faulty across a range of different input images and levels of process variation noise in the underlying hardware. In order to achieve such a pixel-fault identification, we use a fully connected cascade feed-forward neural network (FCCFF-NN) [12]. FCCFF-NN are known to solve hard problem such as $N-$input odd parity problem with a maximum of $\log_2(N+1)$ neurons, while the conventional multilayer feed forward network, with one hidden layer, without cascade, requires as many as N neurons. The choice of FCCFF-NN was critical for this application since we have a total of $8 \times 8 = 256$ inputs and an architecture that is efficient in the number of neurons will allow an area efficient hardware implementation. The neural network used is shown in Figure 6. It has 256 inputs, and 256 outputs. Each input to the NN is a PE output from the most recently computed operation. The outputs of the NN can take values in the range -1 through 1; with -1 implying the computed value by PE is very likely to be incorrect and +1 if the computed value by the PE is very likely to be correct. The neurons in the hidden layer (colored blue in Figure 6) are all bipolar with hard activation function. The output neurons (colored red in Figure 6) are weighted summing blocks of the preceding hidden neurons and the inputs. We use 36 hidden

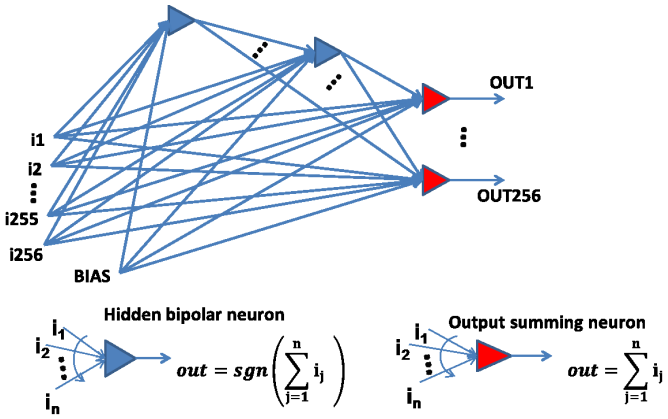

Fig. 6. Fully connected cascade feed-forward neural network. Bipolar neurons with hard activation are used for hidden layer, while output neurons are weighted summers of outputs from all the hidden neurons and inputs.

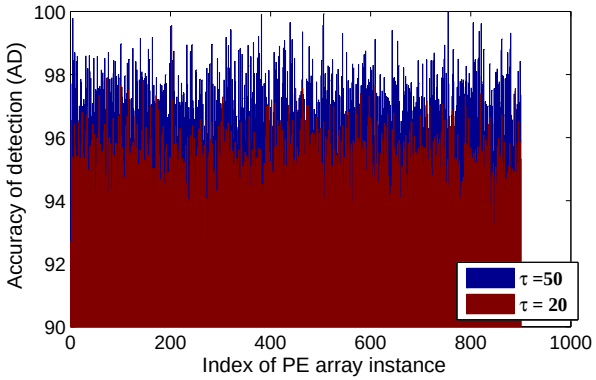

Fig. 7. Accuracy of detection of faulty pixels in different instances (1–900) by the neural network for a threshold $\tau = 20$ and $\tau = 50$.

neurons connected as a fully connected cascade, in addition to the 256 output summing blocks (alternately referred to as output neurons in the paper). Transfer functions of both these type of neurons is shown in Figure 6. We will next describe the data sets used for training and the training procedure imparted to the neural network for faulty pixel identification.

B. Content Rich Data Sets for Training

There are three control parameters that can be varied in order to gather sufficiently content rich data for setting up the NN training. First, the original input image that is to be processed on the PE array. Second, the underlying hardware used for processing the images. Finally, the operation to be performed on this image-either low pass filtering or edge enhancement.

As for the first control parameter, we used 10 images, two of which are shown in Figure 5. For diversity in the second control parameter, we created 1000 instances of the PE array shown in Figure 4. Gates in each instance of the PE array are assigned random delays sampled from the PDF of gate delay distribution at the 32nm technology node (for example, delay distribution of an inverter in 32nm node is shown in Figure 1). Out of these 1000 instances, 100 instances are used for processing images to generate sufficient data for training. Finally, for the last control parameter, we use five of the ten images for low pass filtering (we will refer to it as LP group), and the rest for edge enhancement (referred to as EE group). Each of the five images from LP and EE group are processed on all 100 instances of the PE array. Thus we have a total of $100 \times (5 + 5) = 1000$ images for training the NN. Next to serve as a reference (ideal output from PE array) in the training phase, we process each of the five images from both the groups on a fault-free PE array resulting in 10 images.

C. Training the Neural Network

From the outputs of the original PE array of size 256×256, blocks of size 8×8 are used as input to the NN. Each output of the NN has a possible range of [-1,1]. The outputs of the NN are trained to take values of -1 if there is a difference beyond a threshold τ between PV degraded output and the ideal output, and are trained to take the value 1 if the difference is less than this threshold τ. The value τ can be set any value between a minimum 1 and a maximum of 255. Setting a value of $\tau = 1$ would imply the margin for error is tight, and that any PE whose computation leads to even one intensity level different from the ideal value will be considered faulty. Conversely, if τ is set to a higher value then PE whose computed results are within τ intensity levels of the ideal value are considered fault-free.

We use neuron by neuron (NBN) training algorithm [13], which has significantly improved convergence when compared to the conventional error back propagation training for FCCFF-NN. The NBN training kit is available online at [11].

D. Validation of Neural Network Training

The NN training imparted is now validated on a new data set. This is drawn as follows: A fresh set of 10 images not used in the training phase is chosen. These are split again into LP group and EE group. Each of the five images from both the groups are processed on the remaining 900 instances of the PE array that was created earlier. This amounts to every array instance being tested with 10 image processing operations–five for LP filtering and the rest for edge enhancement.

We now characterize performance of the NN with two figures of merit: 1) accuracy of detection (AD) and 2) mis-prediction (MP). For a given array instance, if N_{act} is the number of PE that are faulty, that is PE whose computed value is different from the desired value by more than threshold τ (We used $\tau = 20$ for all instances); and N_{detect} is the number of PE that was found by the NN to be faulty accurately, then we define, accuracy of detection (AD) to be

$$AD = \frac{N_{detect}}{N_{act}}. \tag{9}$$

We define mis-prediction (MP) to be the ratio of number of PE, $N_{mis-pred}$, that were found to be faulty incorrectly by the NN to the total number of PE that are actually fault free. We therefore have

$$MP = \frac{N_{mis-pred}}{N_{tot} - N_{act}}, \tag{10}$$

where N_{tot} is the total number of PE in the array. In our experiments, we have $N_{tot} = 65536$.

Figure 7 shows bar graph of the accuracy with which individual PE were identified as faulty by the NN for each

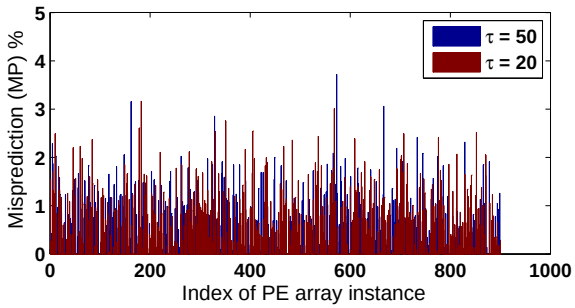

Fig. 8. Rate of mis-prediction (in %) by NN for threshold $\tau = 20$ and $\tau = 50$ for all instances. The means are 0.5% and 0.3% for $\tau = 20$ and $\tau = 50$, respectively.

instance (plotted along the x-axis). Mean accuracy is 95%. We repeat the experiment with another value of $\tau = 50$, and find that the performance of the NN improves to a mean accuracy of 97%. This can be reasoned as follows: difference between actual and desired PE values become discernible as the difference increases; neurons in the neural network, similar to the human eye, can perceive differences better when it is greater. Figure 8 shows a bar graph of mis-prediction for different array instances by the NN. Again, higher threshold $\tau = 50$ results in much smaller mis-prediction. We next look at an on-line error tolerance scheme using this trained NN.

E. On-Line Error Tolerance

PE identified to have been faulty by the NN is precluded from future computation, instead re-using PE whose performance is acceptable. Such a scheme while requires additional time due to PE re-use, it offers significant benefits in the quality of images perceived. Figure 9 shows a test image "Lena" processed (LP filtered) on a PE array with and without precluding faulty elements as predicted by NN. We see significant improvement in visual quality when NN based decision is used for excluding PE that are faulty and re-using PE's that are deemed to be good by NN. Further NN monitors PE outputs after each operation, and continuously updates list of PE that are faulty, which allows the scheme to be resilient against transient errors.

VII. Conclusion and Future Work

A neural network based on-line error tolerance scheme for countering process variation related degradation in array processors was proposed. The key take-away from this work is the demonstration of spatial tolerance without having to increase redundancy in the processing elements. Instead, we use an intermediate layer of intelligence such as neural network to identify *which computation is good enough and which is not.* While neural networks can serve to identify the faulty elements in array processors, greater investigation is required on the possibility of using NN itself as PE [2] in the array so that one can leverage the error resilient properties of neural network.

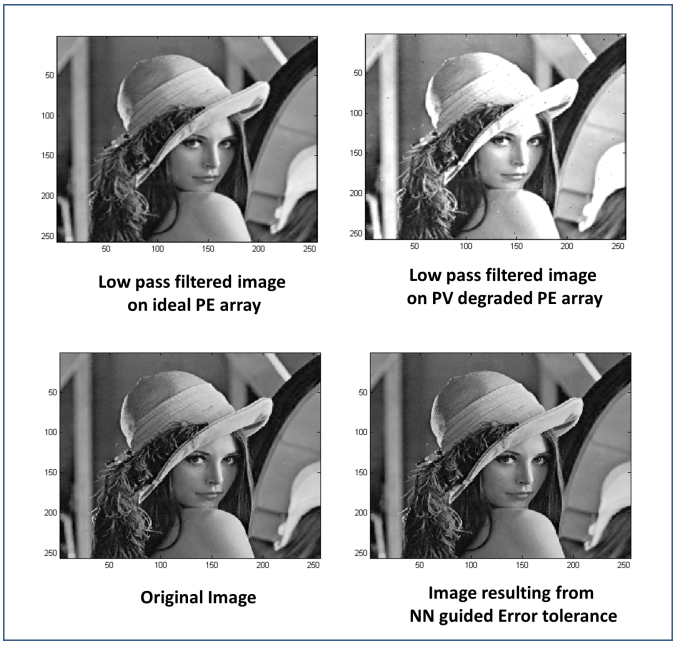

Fig. 9. A test image "Lena" illustrating improved visual quality with achieved with the NN based error tolerance. Notice the increased white spots in the PV degraded image (right top) due to the accumulated effect of MSBs flipping.

References

[1] S. Borkar, "Design Perspectives on 22nm CMOS and Beyond," in *Proc. 46th ACM/IEEE Design Automation Conf.*, July 2009, pp. 93–94.

[2] M. A. Breuer, S. K. Gupta, and T. M. Mak, "Defect and Error Tolerance in the Presence of Massive Numbers of Defects," *IEEE Design & Test of Computers*, vol. 21, no. 3, pp. 216–227, May-June 2004.

[3] I. S. Chong and A. Ortega, "Hardware Testing for Error Tolerant Multimedia Compression Based on Linear Transforms," in *Proc. 20th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Oct. 2005, pp. 523–531.

[4] H. Chung and A. Ortega, "Analysis and Testing for Error Tolerant Motion Estimation," in *Proc. 20th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems*, Oct. 2005, pp. 514–522.

[5] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS Scaling for High Performance and Low Power - the Next Ten Years," *Proc. IEEE*, vol. 83, no. 4, pp. 595–606, Apr. 1995.

[6] R. Hegde and N. R. Shanbhag, "Energy-Efficient Signal Processing via Algorithmic Noise-Tolerance," in *Proc. International Symp. Low Power Electronics and Design*, 1999, pp. 30–35.

[7] K. V. Palem, "Energy Aware Computing Through Probabilistic Switching: A Study of Limits," *IEEE Trans. Computers,*, vol. 54, no. 9, pp. 1123–1137, Sept. 2005.

[8] M. N. O. Sadiku and M. Mazzara, "Computing with Neural Networks," *IEEE Potentials*, vol. 12, no. 3, pp. 14–16, Oct. 1993.

[9] S. Sindia, F. F. Dai, V. D. Agrawal, and V. Singh, "Impact of Process Variations on Computers used for Image Processing," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 2012.

[10] G. V. Varatkar and N. R. Shanbhag, "Error-Resilient Motion Estimation Architecture," *IEEE Trans. Very Large Scale Integration Systems*, vol. 16, no. 10, pp. 1399–1412, Oct. 2008.

[11] B. M. Wilamowski, "Neuron by Neuron Trainer 2.0." http://131.204.128.91/NNTrainer/index.php, accessed on Oct. 10, 2011.

[12] B. M. Wilamowski, D. Hunter, and A. Malinowski, "Solving Parity-N Problems With Feedforward Neural Networks," in *Proc. International Joint Conf. Neural Networks*, volume 4, July 2003, pp. 2546–2551.

[13] B. M. Wilamowski and H. Yu, "Neural Network Learning Without Backpropagation," *IEEE Trans. Neural Networks*, vol. 21, no. 11, pp. 1793–1803, Nov. 2010.

[14] X. Yuan, T. Shimizu, U. Mahalingam, J. S. Brown, K. Z. Habib, D. G. Tekleab, T.-C. Su, S. Satadru, C. M. Olsen, H. Lee, L.-H. Pan, T. B. Hook, J.-P. Han, J.-E. Park, M.-H. Na, and K. Rim, "Transistor Mismatch Properties in Deep-Submicrometer CMOS Technologies," *IEEE Trans. Electron Devices*, vol. 58, no. 2, pp. 335–342, Feb. 2011.