Hierarchical Synthesis of Approximate Multiplier Design for Field-programmable Gate Arrays (FPGA)-CSRmesh System

Article *in* International Journal of Computer Applications · February 2018

DOI: 10.5120/ijca2018916380

Xiaokun Yang
University of Houston - Clear Lake
21 PUBLICATIONS **46** CITATIONS
SEE PROFILE

Han He
1 PUBLICATION **0** CITATIONS
SEE PROFILE

International Journal of Computer Applications (0975 - 8887)
Volume 180 - No.17, February 2018

Hierarchical Synthesis of Approximate Multiplier Design for Field-programmable Gate Arrays (FPGA)-CSRmesh System

Yunxiang Zhang
University of Houston Clear Lake
2700 Bay Area Blvd
Houston, Texas 77058-1002

Xiaokun Yang
University of Houston Clear Lake
2700 Bay Area Blvd
Houston, Texas 77058-1002

Lei Wu
University of Houston Clear Lake
2700 Bay Area Blvd
Houston, Texas 77058-1002

Archit Gajjar
University of Houston Clear Lake
2700 Bay Area Blvd
Houston, Texas 77058-1002

Han He
University of Houston Clear Lake
2700 Bay Area Blvd
Houston, Texas 77058-1002

ABSTRACT

This paper presents a novel hierarchical synthesis for approximating field-programmable gate array (FPGA) based adders and multipliers. Our proposed work implements the multiplier design with the following contributions: 1) providing four types of single-bit approximate adders to cover a wide range of energy-quality trade-offs, 2) presenting many approximations of multipliers by employing the single-bit adders, 3) substituting corresponding bits of the approximate multiplier by considering the quality constrains of the results. The novel hierarchical synthesis of the approach has been integrated into our prior project, an FPGA-IoTmesh system in the field of fog computing for hardware acceleration. Combining the merits of reconfigurability of FPGAs and long-distance connection of CSRmesh technology, this work creates a diverse range of applications such as approximate designs at the network edge, as well as showing a demo for Internet-of-Things (IoT) connections covering an entire building.

Keywords

approximate designs, field-programmable gate array (FPGA), fog computing, Internet-of-Things (IoT)

1. INTRODUCTION

Today, Internet-of-Things (IoT) is becoming an hypernym for a host of applications ranging from smart home/building, to intelligent farming, to industrial IoT as well as edge/fog/cloud computing to support them. To meet heterogeneous requirements for IoT, therefore, some reconfigurable architectures such as field-programmable gate arrays (FPGAs) have been integrated into conventional servers to improve the flexibility of the entire system. As an example shown in Fig. 1), in this integrated system the FPGA is employed in a fog node to speedup the computation and preprocess data would be transferred to the cloud. In such a way the raw data can be significantly reduced and refined before pushing up to the higher abstract levels such as cloud or fog datacenters.

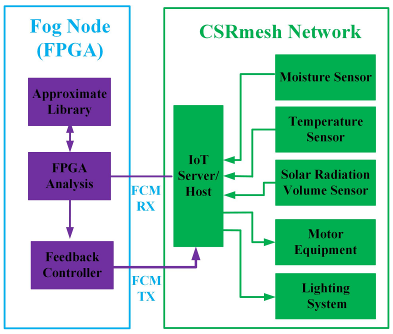


Fig. 1: FPGA-CSRmesh Integrated Fog System

The integration of FPGAs and cloud servers has been proved and evaluated by many implementations such as the Microsoft Bing. Generally Microsoft Bing reaps the benefits in two ways: 1) by embedding 6×8 torus of high-end Stratix V FPGAs into a half-rack of 48 machines, the throughput is increased by 2.25 times, which means that fewer than half as many servers would be needed to sustain the target throughput at the required latency; 2) because the FPGA is able to process requests while keeping latencies low, it is able to absorb more than twice the offered load while executing queries at a latency that never exceeds the software data center at any load [4] [19].
Under this context, in this paper we integrate the FPGA based high-speed computation system with an IoT network, capable of reducing the latency and improving the power efficiency. The latest Bluetooth Low Energy (BLE) mesh technology is used to solve the

problem for extending the connection distance for IoT. Fig. 2 shows the IoT network demo, involving 4 devices, the host controller, the moisture/temperature sensor, the motor equipment, and the light system. All these devices are self-fabricated and used to establish the network [23], [7]. As shown in this figure, the IoT server/host is placed in lab 128 to make connection of all other device. The motor controller, LED and thermal/humidity sensor are set in lab 110, lab 119B, and lab 126. The network is established to connect all the devices together and collect & send all the raw data to the fog nod. In this prototype, the thermal/humidity sensor is over the connection distance to the server/host. Notice that the LED board is thus used as a mesh relay to extend the communication distance to make the connection out of direct point-to-point range. Ideally, the network range can be expanded to a larger area to cover an entire building, farm, and factory, through employing the BLE mesh technology.

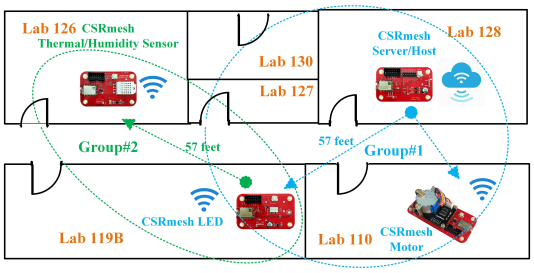


Fig. 2: CSRmesh Network Demo [23]

Integrated the mesh network with the FPGA based fog node, the platform shown in Fig. 1 can be used as a smart home/building system. By approximating the FPGA designs with imprecise components such as adders and multipliers, the raw data collected from the network can be computed faster and lower power consumption. One of the big challenges of this integrated system is the hardware programmability on FPGAs. Basically, FPGA development needs to balance resource cost with algorithm accuracy and extensive hand-coding in register-level transfer (RTL). To fill the gap between software programming and hardware design and provide more RTL choices for different project requirements, we present a basic FPGA design library including five different accuracy levels of components for the beginning of the project: exact design (EX), approximate design #1 (AP1), approximate design #2 (AP2), approximate design #3 (AP3) and approximate design #4 (AP4). More specifically, the main contributions of this paper are:
1) to present five approximation degrees of adders and multipliers and estimate the average/maximum error distances with considering all the possible test cases;
2) to evaluate the tradeoffs between design accuracy and resource cost, by which users can choose computation components depending on different applications and requirements;
3) to employ different approximate components to implement applications on the algorithm level. As a case study, one essential operation of the image processing and computer vision algorithm, the histogram equalization, has been implemented and estimated.

2. RELATED WORK

In the design of energy-efficient digital systems, approximate computing is considered as a possible solution in many application domains. Some of the operations used in this area are intrinsically error-tolerant, such as multimedia, recognition, and data mining [10] [6] [21] and [8]. For these kinds of applications, approximate computing is severed as an important part to reduce the design area, power consumption, and computation delay in the digital systems. This is a tradeoff between accuracy and performance, that we sacrifice accuracy to gain better performance in energy efficiency.
Basically, adders have been commonly considered for the approximate implementation as one of the important components in the circuit design. Some of the approximate adder has been discussed in [12] [9] [14]. These works focused on the subcomponent designs, however, the impact of the approximations on structural implementations has not been considered. In most of the applications, the improvement on system-level has a greater potential to improve the circuit and system performance [25] [24] [22]. The implementation of multipliers usually has three steps: partial product configuration, product accumulation, and the bit shifters. Ref. [13] designed the approximate adder to produce the radix-8 booth encoding 3$\times$ with error reduction. [11] used 2 $\times$ 2 approximate multiplier blocks to compute the final results. And [9] had introduced approximate speculative adders used in the third step of a multiplier.
Recently IBM [17], Intel [16], Microsoft [2], [1], [5] and a lot of companies and research groups [20], [3], [15], [18], [26] proposed some effective results on approximate computing. However, most of them are limited to the software applications. Due to the advantages of the pure hardware implementations, such as the reconfigurability and hardware parallelism, we believe that the FPGA integrated system will be adopted in the future or next-generation of high-performance fog/cloud systems. In this paper, thus, we propose a set of approximate FPGA design components, in order to find a better balance between accuracy and power cost, with providing a wide rang of solutions for different energy-quality tradeoffs corresponding to different applications.

3. PROPOSED WORKS – APPROXIMATE ADDERS AND MULTIPLIERS

In this section, we start from the fundamental single-bit adders' design. Then, four approximate additions are adopted to implement the 4-bit multipliers as a case study. The tradeoffs between accuracy and resource cost are further estimated.

3.1 Approximate adders

The sum and carry bits, denoted as Sum and Cout, of the conventional single-bit full adder can be expressed as

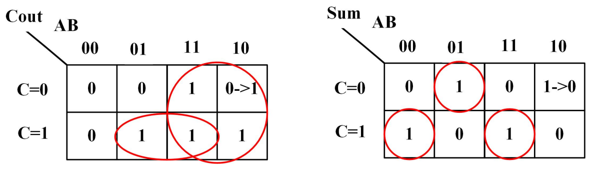
$$Sum(EX) = A'B'C + A'BC' + AB'C' + ABC. \quad (1)$$

$$Cout(EX) = AC + BC + AB. \quad (2)$$

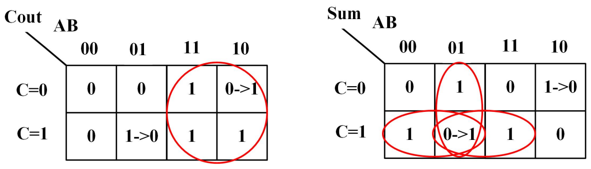
where A and B represent 2 single-bit inputs and C indicates the carry-in bit.
In what follow, we modify the K-map of the basic single-bit adder in order to reduce the gate count. As an example shown in Fig. 3(a), the Sum result is modified from 1 to 0 and the Cout result is changed from 0 to 1 when A=1, B=0 and C=0. After plotting the maximum group of 1's on the map, the algebraic expressions can be simplified as

$$Sum(AP1) = A'BC' + ABC + A'B'C. \quad (3)$$

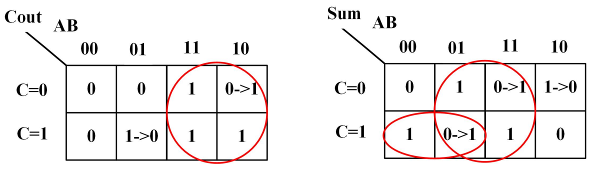
$$Cout(AP1) = A + BC. \quad (4)$$



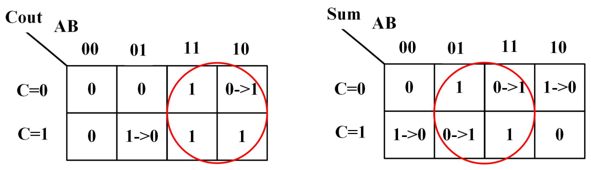
(a) AP1 Adder's K-map



(b) AP2 Adder's K-map



(c) AP3 Adder's K-map



(d) AP4 Adder's K-map

Fig. 3: Algebraic expressions of Approximate Additions

Comparing with the conventional full adder design, the AP1 design simplifies algebraic expressions so as to reduce the hardware cost and power consumption. Likewise, Fig. 3(b), 3(c) and 3(d) also show the modified K-maps of the other three different approximate adders. In the same way the algebraic expressions can be rewritten as

$$Sum(AP2) = A'C + BC + A'B \tag{5}$$

$$Cout(AP2) = A \tag{6}$$

$$Sum(AP3) = B + A'C \tag{7}$$

$$Cout(AP3) = A \tag{8}$$

$$Sum(AP4) = B \tag{9}$$

$$Cout(AP4) = A \tag{10}$$

It can be observed that the AP1 expression costs the largest number of gate count and the AP4 consumes the least in the four approximate designs. In theory, the implementation with more resource cost, is likely to achieve higher accuracy and vice-versa, which will be proved in the following section.

3.2 Approximate multipliers' design and evaluation

Based on the aforementioned adders' design, the 4-bit unsigned multipliers can be implemented in this section. Basically, a binary multiplication requires shifting and adding, as shown in Fig. 4. In this figure, all the product bits, from the least significant bit (LSB) to the most significant bit (MSB), are calculated by exact adders.

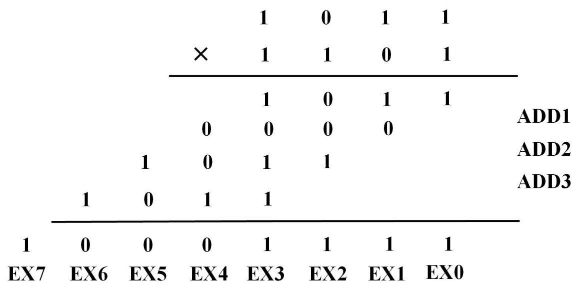
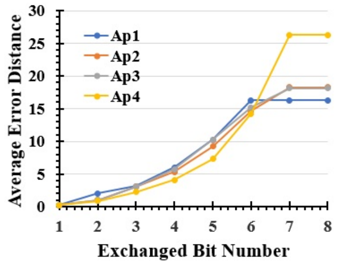


Fig. 4: 4-bit Multiplication

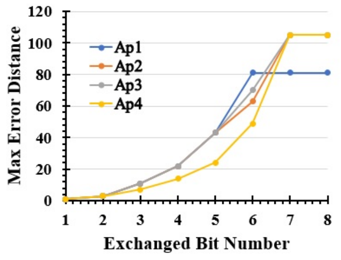
To estimate the accuracy of the approximate multiplications, we replaced the exact single-bit adders by approximate adders (AP1, AP2, AP3, and AP4) from LSB to MSB. The error distance (ED), formulated as $ED = Absolute(R - R*)$ where R represents the exact result and R* indicates the approximate result, is applied to evaluate the multiplications' accuracy. Since the 4-bit multiplication has $2^4 \times 2^4 = 256$ possible combined inputs, the average error distance (AED) can be written as $AED = \frac{\sum_{i=0}^{255} ED_i}{256}$.
Experimental results in Fig. 5(a) demonstrate our expectation, that the AP1 based design achieves the minimum average error distance in all the approximate implementations, it costs the most gate count however. For example, in the case of replacing all the 8-bit additions, the average error distances are 16.25, 18.23, 18.09, and 26.25, using AP1, AP2, AP3, and AP4, respectively.
The maximum error distance (MED) is also applied to estimate the worst case of the approximate designs, which can be formulated as $MED = max\{ED_0, ED_1, ED_2, ..., ED_{255}\}$.
As shown in Fig. 5(b), the worst case for each approximate design happens when all the 8-bit additions are modified. For example, when they are replaced with AP1 and AP2, the maximum error distances are 81 and 105, respectively.

4. IMPLEMENTATION

As a case study, the histogram equalization algorithm is used to estimate the performance of the approximate computations in this section. Generally the histogram equalization is used to enhance the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified histogram.
The pseudocode for implementing the histogram equalization algorithm is depicted in Algo. 1. It basically contains two procedures. In $procedure\#1$ we count the number of each grayscale pixel. Then, the histogram results are computed in $procedure\#2$ as the division of the approximation of multiplications over the size of the

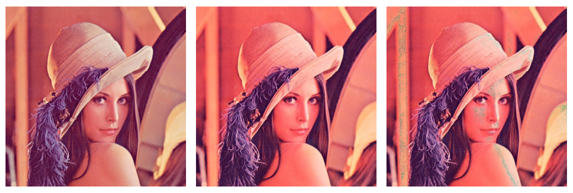


(a) Average Error Distance

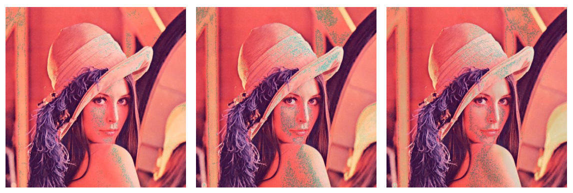


(b) Maximum Error Distance

Fig. 5: Error Data



(a) Original RGB (b) EX Result (c) AP1 PSNR=49.05 MSE=0.81

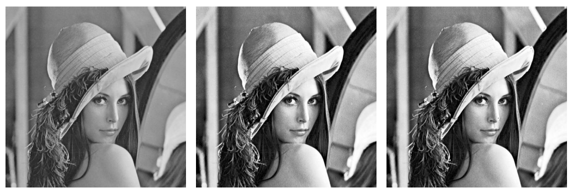


(d) AP2: PSNR=47.34 MSE=1.99 (e) AP3: PSNR=42.13 MSE=3.98 (f) AP4: PSNR=36.07 MSE=16.08

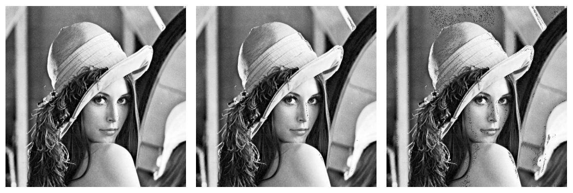
Fig. 6: Equalized RGB Image

Algorithm 1 The Histogram Equalization Algorithm Design with Approximations

```
Input: $Pixel\ array\ p[i](i = 1 - 256)$; $pixel\ number\ s[i]$ of $value = i - 1$;
Output: $Configurated\ pixels : s[j]$;
1: procedure #1 - FIND THE NUMBER OF GRAYSCALE PIXELS:$(s[i] - p[i])$
2:     $initialize : s(0) = p(0)$;
3:     for $(i = 1; i \leq 256; i = i + 1)$ do
4:         $s[i] = p[i] + s(i - 1)$;
5:     end for
6: end procedure
7: procedure #2 - HISTOGRAM EQUALIZATION:$(s[i])$
8:     for $(j = 1; j \leq 256; j = i + 1)$ do
9:         $s[j] = \frac{approx\_comp(s[j]) \times 256}{Height \times Width}$;
10:        if $(s[j] < 256)$ $s[j] = 256$;
11:    end for
12: end procedure
```



(a) Original Grayscale (b) EX Result (c) AP1 PSNR=37.41 MSE=11.79



(d) AP2 PSNR=37.02 MSE=12.91 (e) AP3 PSNR=33.65 MSE=28.06 (f) AP4 PSNR=32.62 MSE=35.53

Fig. 7: Equalized Grayscale Image

image. Finally the regulated results are distributed to each pixel in order to achieve a better image with improved contrast.

Fig. 6 and 7 show the results of rgb and grayscale images, respectively. To demonstrate the difference, we use the peak signal-to-noise ratio (PSNR), a term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation, as one of the performance estimation parameters. It can be formulated as:

$$PSNR = 10 \times \log_{10} \frac{255^2}{MSE}; \qquad (11)$$

where MSE is the mean squared error. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, provided the bit depth is 8 bits, where higher is better.

First, we compare the Fig. 6(a) and Fig. 6(b). It can be observed that the equalized rgb image achieves higher contrast compared with the original image using the exact multipliers. From Fig. 6(c) to Fig. 6(f), the quality of the results has been degraded but the contrast is still enhanced compared with the Fig. 6(a). The higher approximations of the multipliers are employed, the worse quality of the results.

Similarly, Fig. 7 depicts the experimental results of the grayscale images. Note that the higher approximation degrees of the multipliers are employed, the lower of the PSNRs are achieved.

5. EXPERIMENTAL RESULTS

Using the four approximate multipliers in the histogram equalization, the histograms of RGB images and the histograms of grayscale images are shown in Fig. 8 and Fig. 9, respectively. The goal of using the histogram equalization is to make the images to use entire range of values available to them.
Basically, histogram equalization is a nonlinear normalization that stretches the area of histogram with high abundance intensities and compresses the area with low abundance intensities. As an example shown in Fig. 9, the equalized grayscale image is flattened in Fig. 9(b) compared with the original Fig. 9(a). More important, the quality of the equalized results of Fig. 9(c)–Fig. 8(f) are not as good as Fig. 9(b) due to the imprecise multiplications, however, the images are normalized to the original image in Fig. 9(a). In some of the application domains with a tolerance of errors, the imprecise results are acceptable within the quality bound, and the resource cost and power consumption can be significantly reduced due to the approximating designs.

6. CONCLUSION

In this paper, we presented a new methodology to design approximate adders and multipliers that exchange the single-bit computation based on different applications. The experimental results show that our proposed work achieves similar results to the exact design on a very common image processing algorithm, the histogram equalization. As a tradeoff, the hardware resource cost can be significantly reduced due to the imprecise computation on FPGA. Our future work will keep developing more approximate design components in our approximate library and focus on more complicate algorithm on image processing and data mining.

7. REFERENCES

[1] J. Bornholt, T. Mytkowicz, and K. S. McKinley. Uncertain: A first order type for uncertain data. *Proceedings of the 19th Intl. Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 51–66, 2014.

[2] J. Bornholt, T. Mytkowicz, and K. S. McKinley. Uncertain: Abstractions for uncertain hardware and software. *IEEE Micro*, 35(3):132–143, 2015.

[3] V. Chippa, S. Chakradhar, and K. Roy. Analysis and characterization of inherent application resilience for approximate computing. *ACM/EDAC/IEEE 50rd Design Automation Conference (DAC)*, 2013.

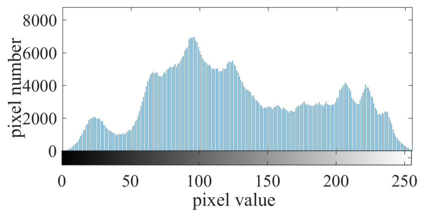
[4] E. S. Chung, A. Putnam, and A. M. Caulfield. A reconfigurable fabric for accelerating large-scale datacenter services. *IEEE Micro*, (3):10–22, 2015.

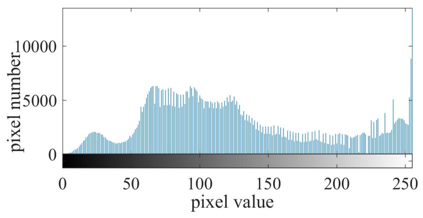
[5] H. Esmaeilzadeh, A. Sampson, and L. Ceze. Architecture support for disciplined approximate programming. *ACM SIGPLAN Notices ASPLOS12*, 47(4):301–312, 2012.

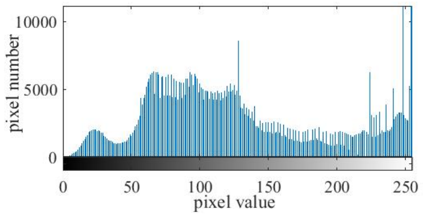
[6] M. Fan, Q. Han, and X. Yang. Energy minimization for online real-time scheduling with reliability awareness. *Elsevier Journal of Systems and Software (Elsevier JSS)*, Vol. 127, PP. 168-176, May 2017.

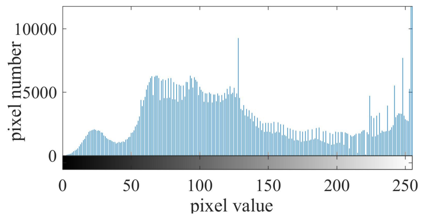
[7] A. Gajjar, Y. Zhang, and X. Yang. Demo abstract: A smart building system integrated with an edge computing algorithm and iot mesh networks. 2017.

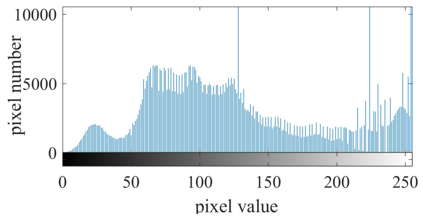
[8] H. He, L. Wu, and X. Yang. Dual long short-term memory networks for sub-character representation learning. *The 15th*


(a) Histogram of Original RGB image


(b) Histogram of Equalized RGB image


(c) Histogram of Equalized RGB image


(d) Histogram of Equalized RGB image


(e) Histogram of Equalized RGB image

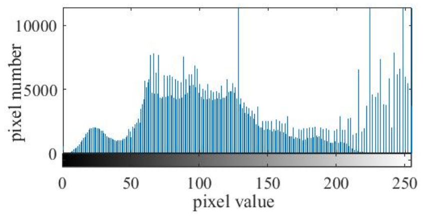
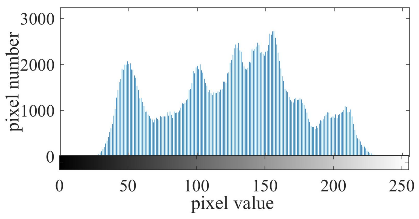
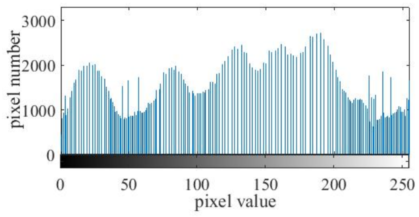

(f) Histogram of Equalized RGB image

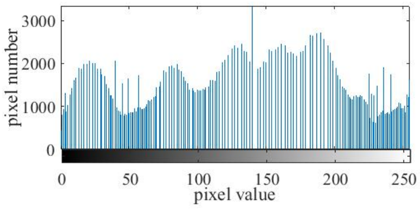
Fig. 8: Comparison of Histograms of RGB Images



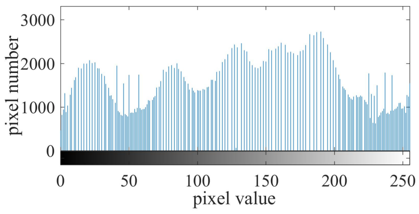
(a) Histogram of Original Grayscale Image



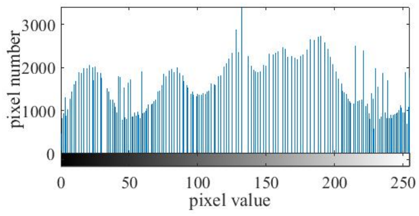
(b) Histogram of Equalized Grayscale Image



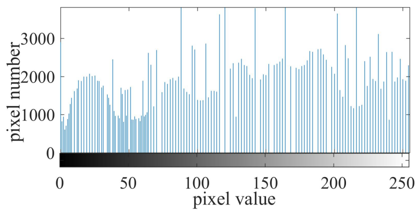
(c) Histogram of Equalized Grayscale Image



(d) Histogram of Equalized Grayscale Image



(e) Histogram of Equalized Grayscale Image



(f) Histogram of Equalized Grayscale Image

Fig. 9: Comparison of Histograms of Grayscale Images

Intl. Conference on Information Technology - New Generations (ITNG-2018), 2018.

[9] J. Huang, J. Lach, and G. Robins. A methodology for energy-quality tradeoff using imprecise hardware. *DAC 2012*, pages 504–509, 2012.

[10] J.Han and M. Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. *IEEE ETS*, 2013.

[11] P. Kulkarni, P. Gupta, and M. Ercegovac. Trading accuracy for power with an underdesigned multiplier architecture. *24th IEEE Intl. Conf. on VLSI Design*, pages 346–351, 2011.

[12] J. Liang, J. Han, and F. Lombardi. New metrics for the reliability of approximate and probabilistic adders. *IEEE Transactions on Computers*, 62(9):1760–1771, 2013.

[13] S. Lu. Speeding up processing with approximation circuits. *Computer*, 37(3):67–73, 2004.

[14] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. Modeling and synthesis of quality-energy optimal approximate adders. *ICCAD 2012*, pages 728–735, 2012.

[15] S. Misailovic, M. Carbin, S. Achour, and Z. Qi. Chisel: Reliability- and accuracy-aware optimization of approximate computational kernels. *Proceedings of the 2014 ACM Intl Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA)*, pages 309–328, 2014.

[16] A. K. Mishra, R. Barik, and S. Paul. iact: A software-hardware framework for understanding the scope of approximate computing. *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014.

[17] R. Nair. Big data needs approximate computing: Technical perspective. *ACM Communications*, (1):58–104, 2015.

[18] G. Pekhimenko, D. Koutra, and K. Qian. Approximate computing: Application analysis and hardware design. www.cs.cmu.edu/ gpekhime/Projects/15740/paper.pdf.

[19] A Putnam, A. M. Caulfield, and E. S. Chung. A cloud-scale acceleration architecture. *2016 49th Annual IEEE/ACM Intel. Symposium on Microarchitecture (MICRO)*, pages 506–511, oct. 2016.

[20] M. Shafique, R. Hafiz, and S. Rehman. Cross-layer approximate computing: From logic to architectures. *ACM/EDAC/IEEE 53rd Design Automation Conference (DAC)*, 2016.

[21] J. Thota, P. Vangali, and X. Yang. Prototyping an autonomous eye-controlled system (aecs) using raspberry-pi on wheelchairs. *Intl. Journal of Compt. Applications (IJCA)*, 158(8):1–7, 2017.

[22] X. Yang and J. Andrian. A high performance on-chip bus (msbus) design and verification. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst. (TVLSI)*, 23(7):1350–1354, July 2015.

[23] X. Yang and X. He. Demo abstract: Establishing a ble mesh network with fabricated csrmesh devices. *The Second ACM/IEEE Symposium on Edge Computing (SEC)*, July 2017.

[24] X. Yang and W. Wen. Design of a pre-scheduled data bus (dbus) for advanced encryption standard (aes) encrypted system-on-chips (socs). *The 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 506–511, Jan 2017.

[25] X. Yang, W. Wen, and M. Fan. Improving aes core performance via an advanced ibus protocol. *ACM Journal*

on Emerging Technologies in Computing, 14(1):61–63, Jan 2018.

[26] X. Yang, N. Wu, and J. Andrian. Comparative power analysis of an adaptive bus encoding method on the mbus structure. *Journal of VLSI Design*, 2017:1–7, May 2017.