*electronics*

MDPI

*Review*

# Gate-Level Static Approximate Adders: A Comparative Analysis

**Padmanabhan Balasubramanian *, Raunaq Nayar and Douglas L. Maskell**

School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore; nayar.raunaq@ntu.edu.sg (R.N.); asdouglas@ntu.edu.sg (D.L.M.)
* Correspondence: balasubramanian@ntu.edu.sg; Tel.: +65-6790-4745

**Abstract:** Approximate or inaccurate addition is found to be viable for practical applications which have an inherent error tolerance. Approximate addition is realized using an approximate adder, and many approximate adder designs have been put forward in the literature targeting an acceptable trade-off between quality of results and savings in design metrics compared to the accurate adder. Approximate adders can be classified into three categories as: (a) suitable for FPGA implementation, (b) suitable for ASIC type implementation, and (c) suitable for FPGA and ASIC type implementations. Among these, approximate adders, which are suitable for FPGA and ASIC type implementations are particularly interesting given their versatility and they are typically designed at the gate level. Depending on the way approximation is built into an approximate adder, approximate adders can be classified into two kinds as static approximate adders and dynamic approximate adders. This paper compares and analyzes static approximate adders which are suitable for both FPGA and ASIC type implementations. We consider many static approximate adders and evaluate their performance for a digital image processing application using standard figures of merit such as peak signal to noise ratio and structural similarity index metric. We provide the error metrics of approximate adders, and the design metrics of accurate and approximate adders corresponding to FPGA and ASIC type implementations. For the FPGA implementation, we considered a Xilinx Artix-7 FPGA, and for an ASIC type implementation, we considered a 32/28 nm CMOS standard digital cell library. While the inferences from this work could serve as a useful reference to determine an optimum static approximate adder for a practical application, in particular, we found approximate adders HOAANED, HERLOA and M-HERLOA to be preferable.

**Keywords:** approximate computing; approximate adder; digital circuits; logic design; FPGA; ASIC



**Citation:** Balasubramanian, P.; Nayar, R.; Maskell, D.L. Gate-Level Static Approximate Adders: A Comparative Analysis. *Electronics* **2021**, *10*, 2917. https://doi.org/10.3390/electronics10232917

Academic Editors: Alessandro Savino and Akash Kumar

Received: 1 October 2021
Accepted: 20 November 2021
Published: 25 November 2021



## 1. Introduction

Computation-intensive technologies such as artificial intelligence, machine learning, big data and analytics, data mining, cloud computing, Internet-of-Things, etc., often deal with a data deluge, which makes processing using accurate computing techniques expensive in terms of time and resources. In such cases, it would be more feasible and economical if computing is performed such that the results are sufficiently correct, which is called approximate, inaccurate or imprecise computing. For example, in image processing, a minor deterioration in the quality of an image may not be noticeable by a human eye. Another example is when a keyword is input into the Google Search engine, many approximate results are sorted according to how well they match the input keyword and displayed for a user's reference. Google employs approximate computing in their tensor processing units [1], which are application-specific integrated circuits (ASICs) developed for machine learning, used in Google Search, Street View and Photos, among others [2], which achieve a 10× improvement in efficiency [3] than conventional graphics processing unit implementations [4]. An approximate implementation of *k*-means clustering [5], which is a popular method of vector quantization used for cluster analysis in data mining, achieves a 50× energy savings by allowing a 5% loss in classification accuracy when compared with a fully accurate classification [6]. An approximate neural network-based solution to the

problem of branch divergence in single instruction multiple data architectures was found to yield, on average, a 13.6× gain in performance and a 14.8× savings in energy compared to the accurate solution, while providing an accuracy of 96% [7]. Thus, approximate computing is a potential alternative to accurate computing for practical applications, which are inherently error tolerant and helps to reduce standard design metrics such as delay, area, power and energy [8,9].

Approximate computing encompasses hardware, software and memory storage [10–12]. With respect to approximate hardware, research has focused on arithmetic circuits [13] and logic circuits [14]. Within the realm of approximate arithmetic circuits, adders and multipliers have received significant attention, and this is because addition and multiplication are often performed in microprocessors [15] and digital signal processors [16].

This paper discusses approximate adders, which are derived by introducing inaccuracies in an accurate adder. Basically, there are two kinds of approximate adders, namely static approximate adders (SAAs) and dynamic approximate adders (DAAs). Approximation is fixed in an SAA that may produce an accurate sum or an approximate sum corresponding to a specified accuracy in a single clock cycle and guarantees assured savings in design metrics compared to the accurate adder. On the other hand, approximation is variable in a DAA, which may produce an approximate or accurate sum on demand involving single or multiple clock cycles. Generally, DAAs comprise an additional error detection and correction logic (EDCL) to adjust their sum corresponding to a specified accuracy. While EDCL is necessary, nevertheless it represents a design overhead in DAAs. In [17], for a digital video encoding application, it was observed that the reduction in power achieved with a dynamic approximate hardware is similar to that achieved using a static approximate hardware and the reason for this is attributed to the extra EDCL present in the former that is absent in the latter.

In this work, we focus on SAAs. SAAs can be classified into three categories based on their implementation platform as: (a) suitable for FPGA implementation [18,19]; (b) suitable for ASIC type implementation [20–22]; and (c) suitable for both FPGA and ASIC type implementations [23–36]. With respect to ASIC type implementations, full-custom and semi-custom design approaches may be adopted. The former involves a manual transistor-level design, while the latter involves an automated gate-level design where a gate-level approximate adder can be described in a hardware description language (HDL) that can be synthesized using a logic synthesis tool. Additionally, a gate-level design is suitable for an FPGA implementation. Hence, gate-level SAAs, suitable for FPGA and ASIC type implementations, are particularly interesting since they are generic and versatile and they form the focus of this work. The objective of this work is to perform a comparative evaluation of different SAAs from the perspectives of error metrics and design metrics, and provide some inferences about which SAA(s) are better optimized. In the rest of the paper, Section 2 reviews several gate-level SAAs that are suitable for FPGA and ASIC type implementations. Section 3 discusses digital image processing involving the accurate adder and various approximate adders and presents the error metrics of approximate adders. Section 4 gives FPGA- and ASIC-based design metrics of accurate and approximate adders corresponding to the application considered. Section 5 gives the concluding remarks.

## 2. Gate-Level Static Approximate Adders

An SAA is usually partitioned into two parts [37] viz. a precise part where addition is performed accurately and an imprecise part where addition is performed inaccurately. Less significant adder input bits are allotted to the imprecise part and more significant adder input bits are allotted to the precise part. Hence, the precise part is more significant than the imprecise part. A block schematic of the accurate adder and generic architectures of many SAAs are shown in Figures 1 and 2, where the precise and imprecise parts of the approximate adders are highlighted in blue and red, respectively.

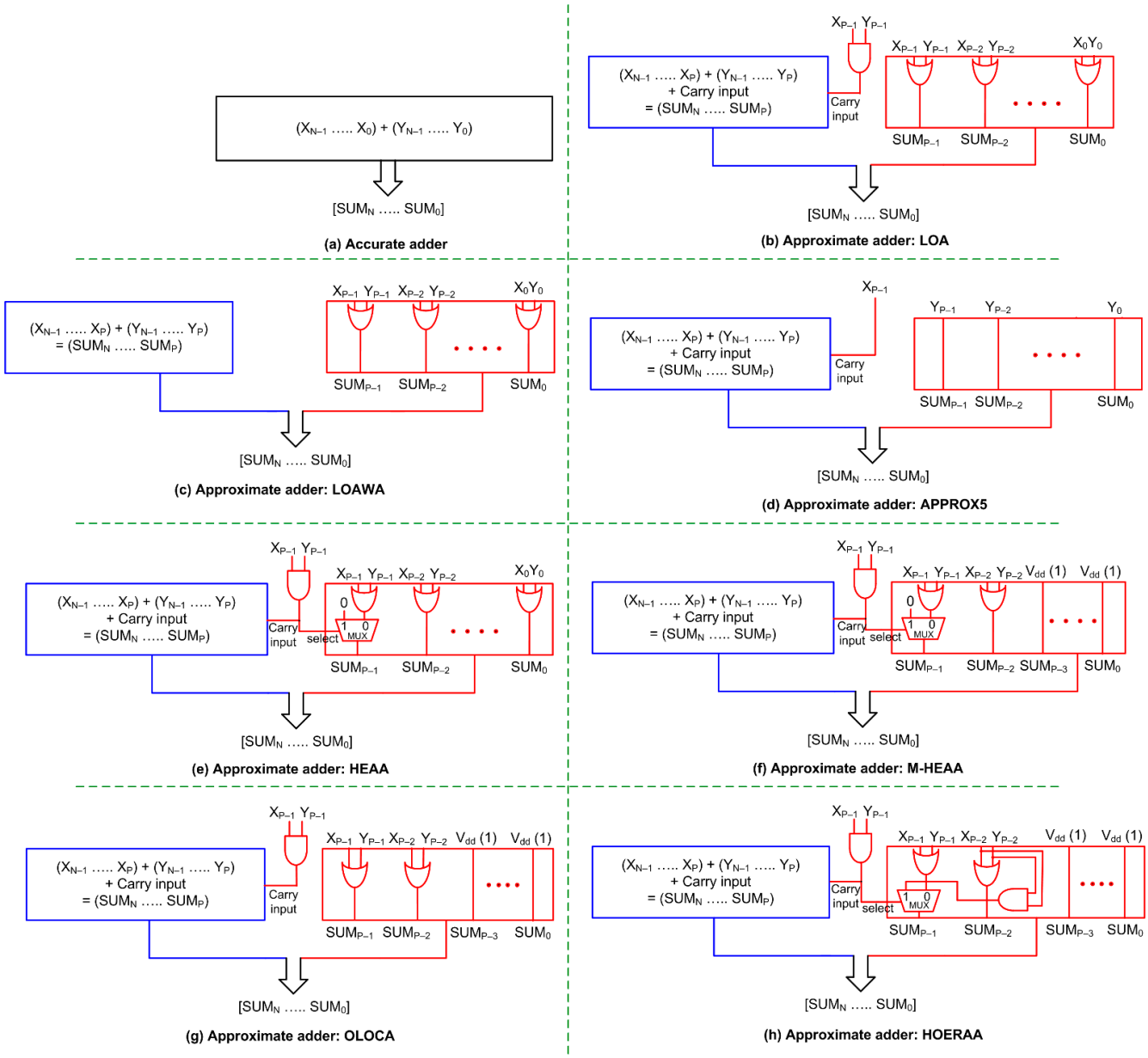


**Figure 1.** Block schematics of accurate adder and some approximate adders: (**a**) Accurate adder; (**b**) LOA; (**c**) LOAWA; (**d**) APPROX5; (**e**) HEAA; (**f**) M-HEAA; (**g**) OLOCA; (**h**) HOERAA.

In Figures 1 and 2, X and Y denote the adder inputs and SUM denotes the adder output. N is the adder size in bits and P is the number of input bits allotted to the imprecise part. Hence, (N–P) input bits are allotted to the precise part. If (N–P) is significantly greater than P, the speed of an approximate adder would be dictated by the speed of its precise part. Given this, for an FPGA implementation, the accurate adder and the precise part of the approximate adders can be described using the addition operator; thereby, the fast carry logic of an FPGA slice can be utilized to realize the accurate and approximate adders in a high-speed fashion. For a semi-custom ASIC type implementation using standard cells, the accurate adder and the precise part of the approximate adders can be described using a high-speed adder architecture such as a carry look-ahead adder (CLA), and they can be synthesized using a logic synthesis tool with speed set as the optimization goal. The precise parts of the approximate adders shown in Figures 1b–h and 2a–f are almost the same, except for the difference pertaining to whether the precise part may incorporate a carry

input or not. Hence, the differences between various approximate adders are primarily attributed to the differences in logic between their imprecise parts.

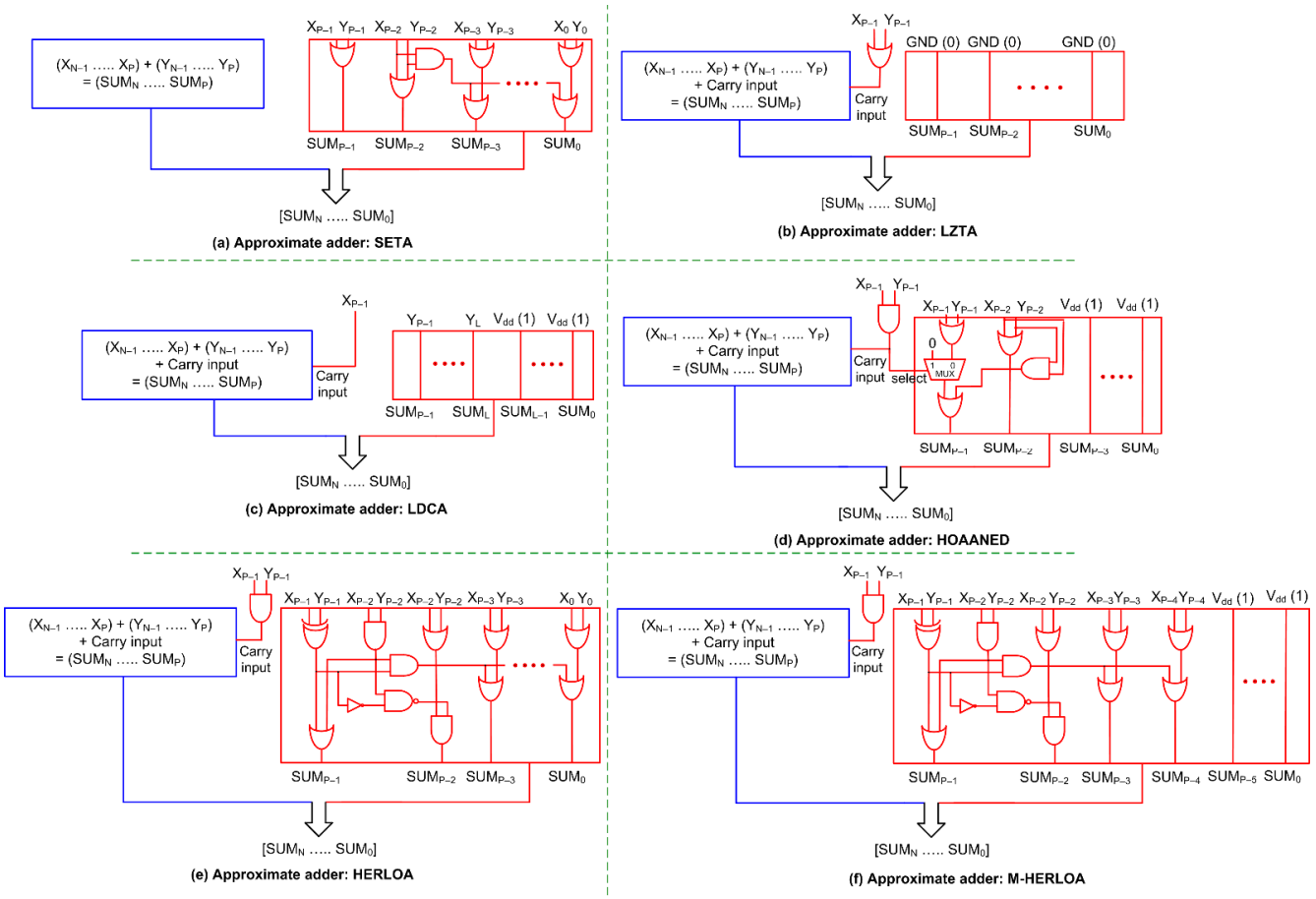


**Figure 2.** Block schematics of approximate adders: (**a**) SETA; (**b**) LZTA; (**c**) LDCA; (**d**) HOAANED; (**e**) HERLOA; (**f**) M-HERLOA.

Since the precise parts of the approximate adders can be realized in the same manner, the following discussion would deal with the imprecise parts of approximate adders shown in Figures 1b–h and 2a–f, which correspond to LOA, LOAWA, APPROX5, HEAA, M-HEAA, OLOCA, HOERAA, SETA, LZTA, LDCA, HOAANED, HERLOA and M-HERLOA. The approximate adders presented in [24,26] were called LOAWA and HEAA in [30], and we retain the same acronyms here for referencing. Further, the approximate adder constructed using an approximate full adder (AMA5) in [25] was called APPROX5 in [30] and we use the same acronym here for referencing. In the following discussions, OR refers to logical OR, AND (NAND) refers to logical AND (NAND), and XOR (XNOR) refers to logical XOR (XNOR) performed between Boolean literals.

Figure 1b shows LOA [23]. In the imprecise part of LOA, $X_{P-1}$ up to $X_0$ are bitwise OR-ed with $Y_{P-1}$ up to $Y_0$, respectively, to produce the corresponding sum bits $SUM_{P-1}$ up to $SUM_0$. $X_{P-1}$ and $Y_{P-1}$ are AND-ed to provide the carry input to the precise part.

Figure 1c shows LOAWA [24]. The logic corresponding to sum bits $SUM_{P-1}$ up to $SUM_0$ are the same for LOAWA as LOA. However, unlike LOA, there is no carry input provided from the imprecise part to the precise part in LOAWA.

In the case of APPROX5 [25], shown in Figure 1d, $Y_{P-1}$ up to $Y_0$ are forwarded as the corresponding sum bits $SUM_{P-1}$ up to $SUM_0$ using buffers, and $X_{P-2}$ up to $X_0$ are discarded. $X_{P-1}$ is given as the carry input to the precise part.

In the case of HEAA [26], shown in Figure 1e, $X_{P-2}$ up to $X_0$ are bitwise OR-ed with $Y_{P-2}$ up to $Y_0$, respectively, to produce the corresponding sum bits $SUM_{P-2}$ up to $SUM_0$.

$X_{P-1}$ and $Y_{P-1}$ are AND-ed and given as the carry input to the precise part, which also serves as the select input to a 2:1 multiplexer (MUX21). If the select input of MUX21 is 0, the OR of $X_{P-1}$ and $Y_{P-1}$ is produced as $SUM_{P-1}$ and if the select input is 1, $SUM_{P-1}$ is assigned a 0.

The modified version of HEAA is shown in Figure 1f [27], which is referred to as M-HEAA in this paper. The modification pertains to the assignment of a constant 1 to (P–2) least significant sum bits of the imprecise part, i.e., $SUM_{P-3}$ up to $SUM_0$. The rest of the logic of M-HEAA is the same as HEAA. Likewise, OLOCA [28], shown in Figure 1g, is a modified version of LOA in that (P–2) least significant sum bits, i.e., $SUM_{P-3}$ up to $SUM_0$ of the imprecise part of LOA are assigned a constant 1 to obtain OLOCA. Excepting for this, the rest of the logic of OLOCA is the same as LOA.

In the case of HOERAA [30], shown in Figure 1h, $SUM_{P-3}$ up to $SUM_0$ are assigned a constant 1, and $SUM_{P-2}$ is produced by OR-ing $X_{P-2}$ and $Y_{P-2}$ like M-HEAA and OLOCA. Like HEAA and M-HEAA, $X_{P-1}$ and $Y_{P-1}$ are AND-ed and given as the carry input to the precise part and also to the select input of a MUX21. If the select input of MUX21 is 0, the OR of $X_{P-1}$ and $Y_{P-1}$ is produced as $SUM_{P-1}$ and if the select input is 1, the AND of $X_{P-2}$ and $Y_{P-2}$ is produced as $SUM_{P-1}$.

In the case of SETA [31], shown in Figure 2a, the imprecise part does not supply a carry input to the precise part. The OR of $X_{P-1}$ with $Y_{P-1}$ and $X_{P-2}$ with $Y_{P-2}$ produce sum bits $SUM_{P-1}$ and $SUM_{P-2}$, respectively. The AND of $X_{P-2}$ and $Y_{P-2}$ is individually OR-ed with the respective bitwise OR-ed outputs of $X_{P-3}$ up to $X_0$ with $Y_{P-3}$ up to $Y_0$ to produce the corresponding sum bits $SUM_{P-3}$ up to $SUM_0$.

LZTA [32] is shown in Figure 2b, where all the sum bits of the imprecise part, i.e., $SUM_{P-1}$ up to $SUM_0$ are assigned a constant 0. As a result, $X_{P-2}$ up to $X_0$ and $Y_{P-2}$ up to $Y_0$ are discarded, and $X_{P-1}$ and $Y_{P-1}$ are OR-ed and given as the carry input to the precise part.

In the case of LDCA [33], shown in Figure 2c, the imprecise part is subdivided into two sections of size L bits and (P–L) bits, and these two sections are typically equal in size. The sum bits corresponding to the L bit section, i.e., $SUM_{L-1}$ up to $SUM_0$, are assigned a constant 1. In the (P–L) bit section, $Y_{P-1}$ up to $Y_L$ are forwarded as the sum bits $SUM_{P-1}$ up to $SUM_L$ through buffers, and $X_{P-1}$ is given as the carry input to the precise part.

HOAANED [34] is shown in Figure 2d. Just like M-HEAA, OLOCA and HOERAA, $SUM_{P-3}$ up to $SUM_0$ are assigned a constant 1 in HOAANED, and $X_{P-2}$ and $Y_{P-2}$ are OR-ed to produce $SUM_{P-2}$. Like HEAA, M-HEAA and HOERAA, in HOAANED, $X_{P-1}$ and $Y_{P-1}$ are AND-ed and given as the carry input to the precise part and also as the select input of a MUX21. If the MUX21 select input is 0, the OR of $X_{P-1}$ and $Y_{P-1}$ and the AND of $X_{P-2}$ and $Y_{P-2}$ are OR-ed to produce $SUM_{P-1}$; otherwise, the AND of $X_{P-2}$ and $Y_{P-2}$ alone would yield $SUM_{P-1}$.

HERLOA [35], shown in Figure 2e, consists of a unique logic in the imprecise part. $X_{P-1}$ and $Y_{P-1}$ are XOR-ed and $X_{P-2}$ and $Y_{P-2}$ are AND-ed and these two are then OR-ed to produce $SUM_{P-1}$. The XOR of $X_{P-1}$ and $Y_{P-1}$ is complemented and NAND-ed with the AND of $X_{P-2}$ and $Y_{P-2}$, which is then AND-ed with the OR of $X_{P-2}$ and $Y_{P-2}$ to produce $SUM_{P-2}$. The XOR of $X_{P-1}$ and $Y_{P-1}$ and the AND of $X_{P-2}$ and $Y_{P-2}$ are AND-ed and this is individually OR-ed with the respective bitwise OR-ed outputs of $X_{P-3}$ up to $X_0$ with $Y_{P-3}$ up to $Y_0$ to produce the corresponding sum bits $SUM_{P-3}$ up to $SUM_0$. Like LOA, HEAA, M-HEAA, OLOCA, HOERAA and HOAANED, $X_{P-1}$ and $Y_{P-1}$ are AND-ed and given as the carry input to the precise part in HERLOA.

M-HERLOA [36], shown in Figure 2f, is a modification of HERLOA in that the logic corresponding to more significant sum bits of the imprecise part (here, $SUM_{P-1}$ up to $SUM_{P-4}$) are retained the same as HERLOA and the remaining less significant sum bits of the imprecise part (here, $SUM_{P-5}$ up to $SUM_0$) are assigned a constant 1. However, the optimum number of least significant sum bits in the imprecise part, which may be assigned a constant 1 in M-HERLOA is best decided depending on which assignment enables reduced error metrics commensurate with a target application.

## 3. Digital Image Processing Using Accurate and Approximate Adders

We considered digital image processing (reconstruction) as a practical application, as in [37], to evaluate the performance of different approximate adders vis-à-vis the accurate adder. We considered many digital images with a grayscale resolution of 8 bits and a spatial resolution of 512 × 512 for experimentation. Image processing was performed as described in [34], whereby an original image was translated into a matrix form which was then processed by computing fast Fourier transform and inverse fast Fourier transform accurately or approximately. The matrix output was subsequently re-translated into a digital image. Integer Fourier transforms were computed wherein multiplication was performed accurately, while addition was performed accurately or approximately. To perform accurate addition, we used the accurate adder and to perform approximate addition, we used different approximate adders individually. We considered a 32-bit addition as in [37], which implies that the size of the accurate adder and approximate adders are 32 bits.

In general, having a small imprecise part in an approximate circuit would reduce the savings in design metrics gained compared to the accurate circuit, while having a big imprecise part in an approximate circuit may make its output quality (here, image quality) unacceptable for a practical application. Therefore, having an optimum imprecise part in an approximate circuit is important as that would pave the way for an acceptable compromise between output quality and savings in design metrics gained by an approximate circuit compared to the accurate circuit [30,34].

It was observed in [20,25] that for digital image processing and digital video encoding applications, the approximation limit may be optimally specified in the range of 7 to 9 least significant bits while considering a 32-bit arithmetic. Here, following a trial-and-error approach, as discussed in [34], the optimum imprecise part of the approximate adders was determined as 10 bits in size and the optimum precise part as 22 bits in size.

Example images *lena* and *cameraman*, which were processed accurately and approximately using accurate and approximate adders, respectively, are shown in Figures 3 and 4 for an illustration. Two figures of merit viz. peak signal to noise ratio (PSNR) [38] and structural similarity index metric (SSIM) [39] were estimated to ascertain the quality of reconstructed images, and they are given above the images in Figures 3 and 4. While PSNR is a figure of merit widely used in digital signal processing, SSIM is a figure of merit of specific relevance for digital image processing. Here, PSNR is used to quantify the signal strength relative to the noise/distortion in an image. A high value of PSNR indicates less distortion in an image. SSIM is estimated by comparing a reference (original) image with a target image. Here, the target image may refer to an accurately or approximately reconstructed image. SSIM ranges from 0 to 1 decimal, with 0 indicating no similarity and 1 indicating a perfect similarity between the reference and target images. Hence, a high value of SSIM is also preferred. A perusal of Figures 3 and 4 would reveal major or minor distortions in the form of grains, spots and/or shaded regions in the images obtained using approximate adders compared to the images obtained using the accurate adder.

The image reconstructed by computing accurate fast Fourier transform and inverse fast Fourier transform involving accurate addition is shown in Figures 3a and 4a, while the images reconstructed by computing approximate fast Fourier transform and inverse fast Fourier transform involving approximate additions are shown in Figures 3b–n and 4b–n, respectively. Due to the accurate computation, PSNR = ∞ for Figures 3a and 4a, and their SSIM = 1.


**Figure 3.** The *lena* image processed accurately and approximately using (**a**) accurate adder and (**b**–**n**) approximate adders.

PSNR and SSIM calculated for the images reconstructed using different approximate adders are given in Tables 1 and 2, respectively. From Figures 3 and 4 and Tables 1 and 2, it is noted that among the approximate adders, HOAANED consistently results in an improved PSNR and this is attributed to its near-normal error distribution characteristic. HOAANED also enables an enhanced SSIM in comparison with many approximate adders, except HERLOA and M-HERLOA. HERLOA and M-HERLOA consistently result in almost the same SSIM, which is greater than the SSIM of images reconstructed using other approximate adders, and this is due to a better approximate logic employed in their imprecise parts. To validate this, an error analysis was performed by supplying one million random input vectors to the accurate adder and approximate adders. The extent of error occurring in the approximate adders relative to the accurate adder was plotted in the form of an error distribution, as shown in Figure 5, which portrays the error magnitudes in terms of their percentage occurrence.

Two well-known error metrics, namely mean absolute error (MAE) and root mean square error (RMSE) were calculated for the approximate adders relative to the accurate adder whose equations are given by (1) and (2). MAE is also called mean error distance in the literature. Nevertheless, RMSE is more important since it better quantifies the extent of signal degradation in digital signal processing [40]. In Equations (1) and (2), L represents the number of input vectors supplied to the adders for calculation of the error metrics and here, L = 1,000,000. The notation ($X_K$, $Y_K$) denotes one set of adder inputs. Accurate_Sum ($X_K$, $Y_K$) represents the sum produced by the accurate adder for a given

input, and Approximate_Sum ($X_K$, $Y_K$) represents the sum produced by an approximate adder for the same input.

$$\text{MAE} = \frac{1}{L}\sum_{K=1}^{L} |\text{Approximate\_Sum}(X_K, Y_K) - \text{Accurate\_Sum}(X_K, Y_K)\,| \quad (1)$$

$$\text{RMSE} = \sqrt{\frac{1}{L}\sum_{K=1}^{L} (\text{Approximate\_Sum}(X_K, Y_K) - \text{Accurate\_Sum}(X_K, Y_K))^2} \quad (2)$$


**Figure 4.** The *cameraman* image processed accurately and approximately using (**a**) accurate adder and (**b**–**n**) approximate adders.

**Table 1.** PSNR (in dB) of various digital images reconstructed using different approximate adders.

| Approximate Adder | *Barbara* | *Boat* | *Einstein* | *Lake* | *Cameraman* | *Peppers* | *Woman* | Average PSNR |
|---|---|---|---|---|---|---|---|---|
| LOA | 32.4863 | 32.5604 | 32.5567 | 32.6313 | 32.1966 | 32.6581 | 32.8121 | 32.5574 |
| LOAWA | 25.1106 | 24.8022 | 25.7325 | 25.2703 | 25.0872 | 25.1460 | 25.2304 | 25.1970 |
| APPROX5 | 31.6881 | 31.8445 | 31.8320 | 31.7789 | 31.3060 | 31.8853 | 32.1200 | 31.7793 |
| HEAA | 30.6490 | 30.5959 | 31.0126 | 30.6447 | 30.6800 | 30.7053 | 30.8507 | 30.7340 |
| M-HEAA | 29.6692 | 29.5523 | 30.1740 | 29.6633 | 29.6510 | 29.6921 | 29.8162 | 29.7454 |
| OLOCA | 32.0496 | 32.1698 | 32.1424 | 32.1815 | 31.8063 | 32.2262 | 32.3729 | 32.1355 |
| HOERAA | 32.9709 | 33.0211 | 33.1791 | 32.9155 | 32.7300 | 33.0998 | 33.2847 | 33.0287 |
| SETA | 25.1447 | 24.8346 | 25.7657 | 25.3066 | 25.1226 | 25.1806 | 25.2653 | 25.2314 |
| LZTA | 30.8740 | 30.9092 | 31.0290 | 30.8975 | 30.9622 | 31.0619 | 30.8768 | 30.9444 |
| LDCA | 31.7570 | 31.9085 | 31.8894 | 31.8521 | 31.3805 | 31.9542 | 32.1818 | 31.8462 |
| HOAANED | 34.7582 | 34.6552 | 34.7908 | 34.7423 | 34.7383 | 34.7416 | 34.7845 | 34.7444 |
| HERLOA | 33.7722 | 33.6949 | 33.9227 | 33.7501 | 33.7766 | 33.8136 | 33.8772 | 33.8010 |
| M-HERLOA | 32.8549 | 32.7319 | 33.1088 | 32.8431 | 32.8210 | 32.8586 | 32.9572 | 32.8822 |

**Table 2.** SSIM (in decimal) of various digital images reconstructed using different approximate adders.

| Approximate Adder | *Barbara* | *Boat* | *Einstein* | *Lake* | *Cameraman* | *Peppers* | *Woman* | Average SSIM |
|---|---|---|---|---|---|---|---|---|
| LOA | 0.8527 | 0.8602 | 0.8440 | 0.8666 | 0.8422 | 0.8447 | 0.8150 | 0.8465 |
| LOAWA | 0.8396 | 0.8464 | 0.8198 | 0.8514 | 0.8181 | 0.8302 | 0.7884 | 0.8277 |
| APPROX5 | 0.8450 | 0.8461 | 0.8318 | 0.8537 | 0.8322 | 0.8284 | 0.8063 | 0.8348 |
| HEAA | 0.9426 | 0.9480 | 0.9370 | 0.9485 | 0.9266 | 0.9471 | 0.9174 | 0.9382 |
| M-HEAA | 0.9362 | 0.9426 | 0.9305 | 0.9426 | 0.9297 | 0.9458 | 0.9086 | 0.9337 |
| OLOCA | 0.8463 | 0.8517 | 0.8373 | 0.8587 | 0.8412 | 0.8359 | 0.8096 | 0.8401 |
| HOERAA | 0.9297 | 0.9358 | 0.9226 | 0.9394 | 0.9113 | 0.9279 | 0.9028 | 0.9242 |
| SETA | 0.8412 | 0.8475 | 0.8213 | 0.8527 | 0.8175 | 0.8319 | 0.7901 | 0.8289 |
| LZTA | 0.8290 | 0.8516 | 0.8234 | 0.8490 | 0.8237 | 0.8287 | 0.7813 | 0.8267 |
| LDCA | 0.8480 | 0.8484 | 0.8349 | 0.8562 | 0.8374 | 0.8313 | 0.8103 | 0.8381 |
| HOAANED | 0.9301 | 0.9361 | 0.9225 | 0.9372 | 0.9072 | 0.9286 | 0.9020 | 0.9234 |
| HERLOA | 0.9619 | 0.9660 | 0.9578 | 0.9663 | 0.9462 | 0.9643 | 0.9445 | 0.9581 |
| M-HERLOA | 0.9601 | 0.9640 | 0.9559 | 0.9648 | 0.9469 | 0.9637 | 0.9423 | 0.9568 |

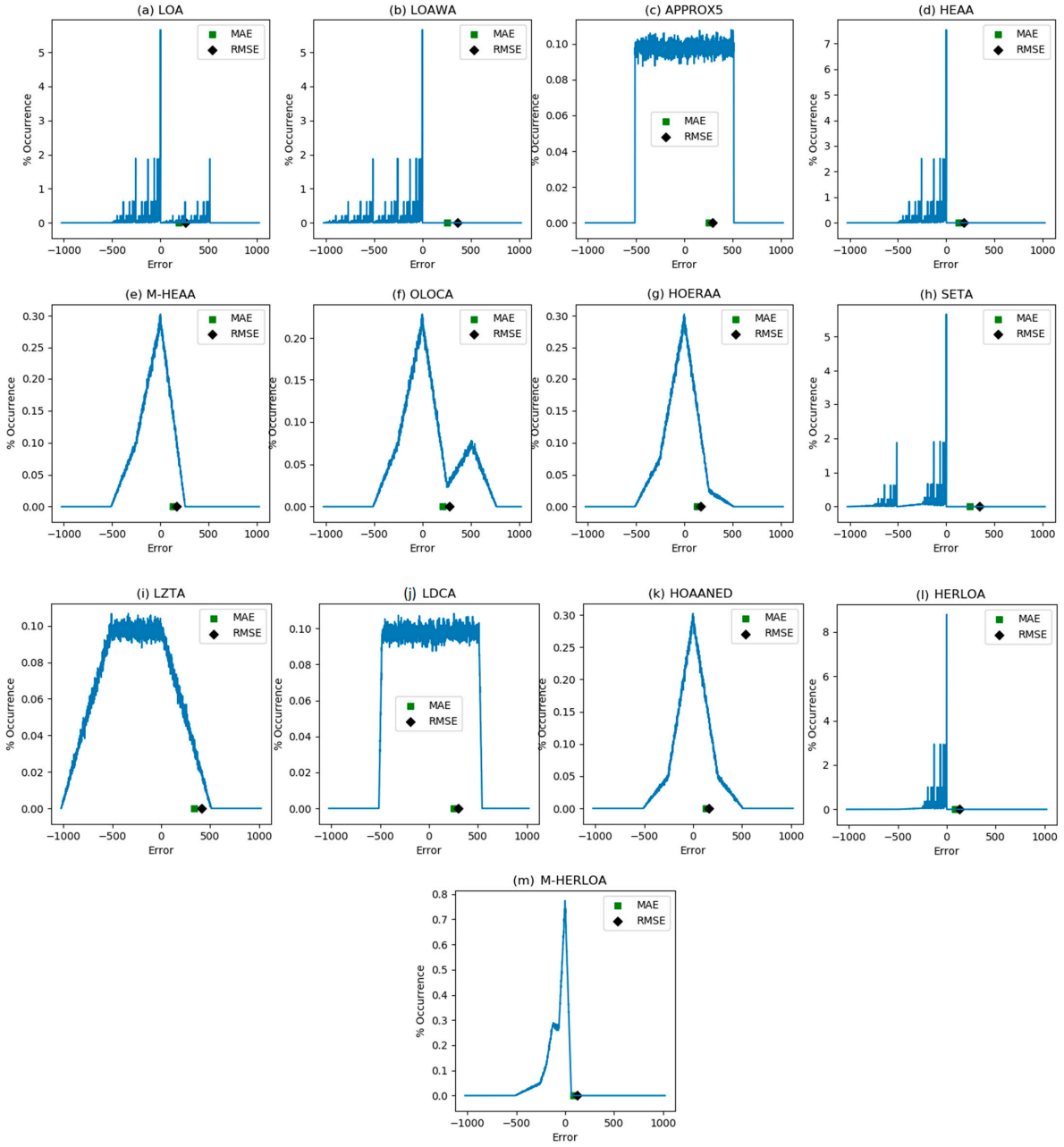


**Figure 5.** Error distribution of 32-bit approximate adders with a 10-bit imprecise part along with a highlight of their MAE

and RMSE: (**a**) LOA; (**b**) LOAWA; (**c**) APPROX5; (**d**) HEAA; (**e**) M-HEAA; (**f**) OLOCA; (**g**) HOERAA; (**h**) SETA; (**i**) LZTA; (**j**) LDCA; (**k**) HOAANED; (**l**) HERLOA; (**m**) M-HERLOA. The error magnitudes are given in the X axis and the percentage of their occurrences is given in the Y axis.

From Figure 5, it is seen that HOAANED has a near-normal error distribution, which is a reflection of the fact that its positive and negative (true) error magnitudes are rather balanced and become almost neutralized on average—this is the reason for the greater PSNR of images reconstructed using HOAANED compared to the PSNR of images reconstructed using other approximate adders, as seen from Table 1.

In Figure 5, HERLOA has a restricted magnitude of error occurrences compared to the other approximate adders, and this may be the reason for the reduced distortions noticed in Figures 3m and 4m compared to Figures 3b–l and 4b–l, respectively. HERLOA does not have a positive error magnitude, and HERLOA is closely followed by M-HERLOA in terms of an optimized error distribution. Although the magnitude of error occurrences is relatively greater in M-HERLOA compared to HERLOA, the former has some positive error magnitudes, which contributes to an overall decrease in its MAE and RMSE.

Figure 6 depicts MAE and RMSE calculated for different approximate adders by considering one million random input vectors. MAE is depicted by the blue bars and RMSE is depicted by the orange bars in Figure 6. In general, approximate adders which include a carry input in their precise part that is supplied from the imprecise part would have less errors compared to approximate adders which have disjoint precise and imprecise parts. This is because a valid carry input supplied from the imprecise part may significantly impact the output of the precise part and, thus, the overall sum. Hence, LOAWA and SETA, which do not feature an internal carry input, have higher MAE and RMSE compared to their counterparts, which feature an internal carry input. LZTA, which is shown in Figure 2b, is worse since the sum bits belonging to the imprecise part of LZTA are assigned a constant 0 and so the information corresponding to the imprecise part may become completely lost during the data processing depending upon the specified inputs. Figure 6 shows that M-HERLOA has less MAE and RMSE compared to other approximate adders, with HERLOA having MAE and RMSE closer to M-HERLOA.

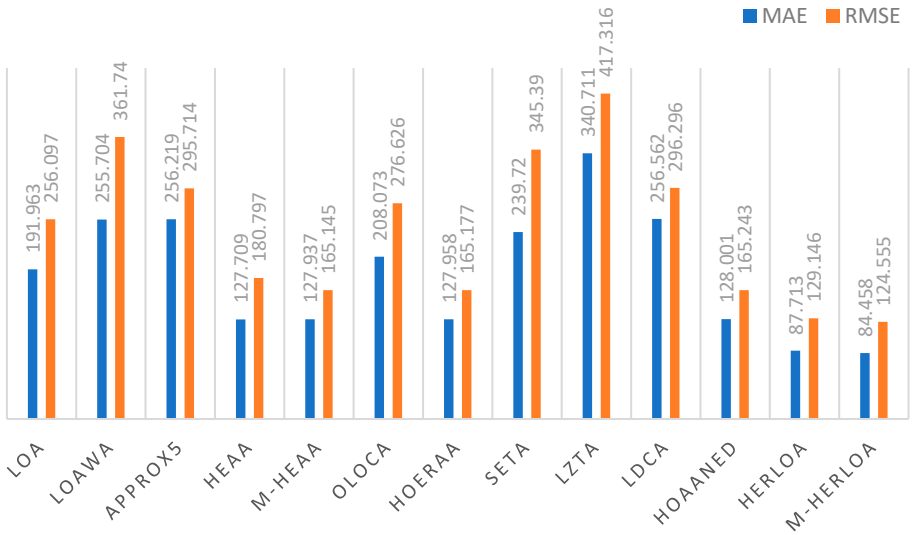


**Figure 6.** Error parameters (MAE and RMSE) calculated for different approximate adders of size 32 bits comprising a 10-bit imprecise part.

To achieve a higher PSNR, HOAANED is preferable and to achieve a higher SSIM, HERLOA and M-HERLOA are preferable. Nevertheless, in terms of the error metrics and image processing figures of merit combined, M-HERLOA may be preferable to its approximate counterparts.

## 4. Accurate and Approximate Adders—Implementation Results

Accurate and approximate adders were implemented commensurate with the digital image processing application discussed using FPGA and ASIC design platforms. About 1000 random input vectors were supplied to the adders to perform functional simulations and their switching activity data was used to estimate total power dissipation.

For the FPGA implementation, the accurate and approximate adders were described behaviorally in Verilog HDL and synthesized and implemented on a Xilinx Artix-7 FPGA (device: xc7a100tcsg324-3) using the Vivado design tool (version: 2018.3). We described the accurate adder and the exact parts of approximate adders using the addition operator in Verilog. As a result, the fast carry logic (CARRY4) inherent in an FPGA slice was utilized to realize high speed addition. Flow_AreaOptimized_high was specified as the synthesis strategy and the default implementation strategy was used. Following an efficient FPGA design practice, a pair of register banks was provided before the adder inputs to eliminate unnecessary input–output (IO) routing delay from dominating the critical path delay. A register bank collects the adder outputs and, thus, the adder is sandwiched between the input and output register banks, with these register banks driven by a common clock. The adders were successfully synthesized and implemented, and the FPGA design metrics obtained after placement and routing are given in Table 3. In Table 3, delay refers to the minimum clock period, which is representative of critical path delay, and power refers to the total on-chip power, which is the sum of the power consumed by clock, signals, logic and IO. The number of slice look-up tables (LUTs) and flip-flops consumed for the implementation of the adders is also given in Table 3.

**Table 3.** Design metrics of accurate and approximate adders implemented on an Artix-7 FPGA.

| Adder | Delay (ns) | LUTs | Flip-Flops | Power (W) |
|---|---|---|---|---|
| Accurate (FPGA) | 2.10 | 32 | 97 | 0.209 |
| LOA | 1.89 | 27 | 97 | 0.198 |
| LOAWA | 1.86 | 27 | 97 | 0.198 |
| APPROX5 | 1.84 | 22 | 88 | 0.200 |
| HEAA | 1.89 | 27 | 97 | 0.199 |
| M-HEAA | 1.87 | 23 | 73 | 0.188 |
| OLOCA | 1.87 | 23 | 73 | 0.187 |
| HOERAA | 1.87 | 23 | 73 | 0.188 |
| SETA | 1.85 | 31 | 97 | 0.199 |
| LZTA | 1.87 | 22 | 69 | 0.184 |
| LDCA | 1.83 | 22 | 78 | 0.195 |
| HOAANED | 1.87 | 23 | 73 | 0.188 |
| HERLOA | 1.89 | 28 | 97 | 0.199 |
| M-HERLOA | 1.90 | 25 | 79 | 0.190 |

From Table 3, we see that, in general, the approximate adders have less delay, consume fewer LUTs and flip-flops and have less on-chip power compared to the accurate FPGA adder. This is because the accurate adder is 32 bits in size, whereas the precise part of the approximate adders is only 22 bits in size, since 10 bits have been allocated to the imprecise part. Hence, the delay of the approximate adders is dominated by the delay of their precise part. Because the imprecise parts of the approximate adders have reduced logic compared to the accurate adder, fewer LUTs and/or flip-flops were required for their implementation and, thus, overall, the approximate adders require lesser resources (LUTs and flip-flops) compared to the accurate adder. For example, M-HERLOA requires 7 LUTs and 18 flip-flops less compared to the accurate FPGA adder in Table 3. Since 6 least significant sum bits were assigned a constant 1 in M-HERLOA, 12 input flip-flops and 6 output flip-flops were not required, thus saving 18 flip-flops compared to the accurate adder. Additionally, the reduction in logic of the approximate adders results in their reduced power consumption compared to the accurate adder. The differences between the resource utilization and power consumption of approximate adders are due

to the differences between their imprecise part logic. The delay is almost the same for the approximate adders and only minor variations are observed between them. This is partly because the precise part of some approximate adders accepts a carry input from the imprecise part, while this is absent in the other approximate adders, and partly due to the area optimized place and route as performed by the FPGA design tool.

In Section 3, in terms of error metrics and/or image processing results, it was noted that HOAANED, HERLOA and M-HERLOA are preferable. From Table 3, it is noted that compared to the accurate FPGA adder, HOAANED has 11% less delay, requires 28.1% fewer LUTs and 24.7% fewer flip-flops, and consumes 10% less power; HERLOA has 10% less delay, requires 12.5% fewer LUTs and consumes 4.8% less power; and M-HERLOA has 9.5% less delay, requires 21.9% fewer LUTs and 18.6% fewer flip-flops, and consumes 9.1% less power.

For an ASIC type standard cell-based implementation, the accurate and approximate adders were described structurally in Verilog HDL. To realize the accurate and approximate adders for high speed, the accurate adder and precise parts of the approximate adders were described using a high speed CLA architecture [41]. The 32-bit accurate adder was described using eight 4-input CLAs, and the 22-bit precise parts of the approximate adders were described using five 4-bit CLAs and a 2-bit CLA. The 2-bit CLA may or may not include a carry input and this depends on the approximate adder architecture considered, i.e., whether the approximate adder may or may not have a carry input supplied from the imprecise part to the precise part. It may be recalled from Section 2 that LOAWA and SETA do not feature an internal carry input from the imprecise part to the precise part, while the rest of the approximate adders do.

The accurate and approximate adders were synthesized for high speed using Synopsys Design Compiler with speed set as the optimization goal and their total area (cells area + interconnect area) was estimated. A 32/28 nm CMOS standard cell library [42] was used for the implementation. A typical case library specification with a supply voltage of 1.05 V and an operating junction temperature of 25 °C was considered. After synthesis, the adders were simulated and their functionality was verified. Subsequently, the switching activity data obtained was used to estimate the total (average) power dissipation using PrimePower. PrimeTime was used to estimate the critical path delay. The adder outputs were assigned a fanout-of-4 drive strength and default wire loads were included. The ASIC-based design metrics are given in Table 4.

In Table 4, we see that all the approximate adders have the same delay and this is because their precise parts were realized for high speed using a common CLA architecture. The areas of approximate adders, however, differ and this is due to the differences in the logic composition of their imprecise parts. Consequently, their power dissipation also differs. To assign a constant 1 to some least significant sum bits in M-HEAA, OLOCA, HOERAA, LDCA, HOAANED and M-HERLOA, tie-to-high (TIEH) standard cells were used and to assign a constant 0 to some least significant sum bits in LZTA, tie-to-low (TIEL) standard cells were used. TIEH and TIEL standard cells of [42] have the same design attributes. Given that HOAANED, HERLOA and M-HERLOA are preferable, from Table 4, it is noted that HOAANED, HERLOA and M-HERLOA have 17.9% less delay compared to the accurate CLA. Further, compared to the accurate CLA, HOAANED occupies 24.7% less area and dissipates 28.2% less power, HERLOA occupies 21.5% less area and dissipates 21.5% less power, and M-HERLOA occupies 23.1% less area and dissipates 26.7% less power.

Power-delay product (PDP), which is representative of energy and considered as a low power figure of merit, was calculated for accurate and approximate adders corresponding to FPGA and ASIC type implementations and normalized, which is shown in Figure 7. To normalize the PDP, the highest PDP corresponding to an adder (i.e., accurate adder) was considered as the baseline and this was used to divide the PDP of all the adders corresponding to FPGA and ASIC type implementations separately. The green and blue bars shown in Figure 7 represent the normalized PDP corresponding to FPGA and ASIC

type implementations, respectively. Power and delay are preferred to be less for a digital design and, hence, PDP is also preferred to be less. In Figure 7, the approximate adders are found to have less PDP compared to the accurate adder, meaning the former are more energy efficient than the latter.

**Table 4.** Design metrics of accurate and approximate adders synthesized using a 32/28 nm CMOS standard digital cell library.

| Adder | Delay (ns) | Area ($\mu m^2$) | Power ($\mu W$) |
|---|---|---|---|
| Accurate (CLA) | 1.17 | 564.60 | 94.33 |
| LOA | 0.96 | 428.36 | 71.77 |
| LOAWA | 0.96 | 413.37 | 68.86 |
| APPROX5 | 0.96 | 424.58 | 73.54 |
| HEAA | 0.96 | 430.65 | 71.49 |
| M-HEAA | 0.96 | 422.32 | 66.11 |
| OLOCA | 0.96 | 420.03 | 66.38 |
| HOERAA | 0.96 | 430.38 | 68.82 |
| SETA | 0.96 | 419.68 | 72.94 |
| LZTA | 0.96 | 415.56 | 63.14 |
| LDCA | 0.96 | 420.07 | 68.05 |
| HOAANED | 0.96 | 425.36 | 67.73 |
| HERLOA | 0.96 | 443.28 | 74.01 |
| M-HERLOA | 0.96 | 433.94 | 69.11 |

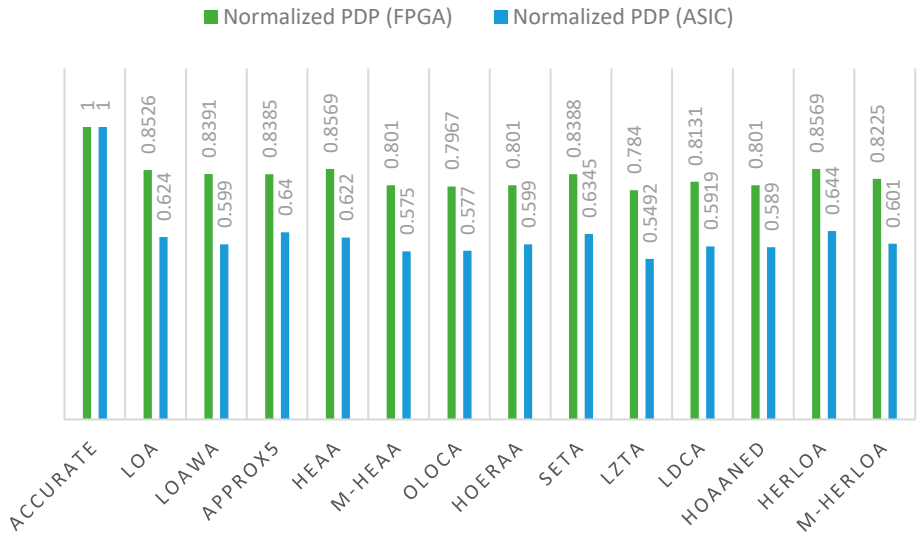


**Figure 7.** Normalized PDP of accurate and approximate adders corresponding to FPGA and ASIC type implementations.

The normalized PDP plots of the adders corresponding to FPGA and ASIC type implementations indicate a similar trend. Among the adders, LZTA is very energy efficient. However, the image processing results shown in Figures 3 and 4 and Tables 1 and 2, and the error distribution and error metrics given in Figures 5 and 6, clearly show that LZTA is not preferable. In approximate computation, output quality assumes higher precedence than savings in design metrics gained compared to accurate computation. Given this, LZTA is not preferable, although it may have a high energy efficiency. On the contrary, HOAANED, which enables a higher PSNR, and HERLOA/M-HERLOA, which enable a higher SSIM, are preferred and they report a significant improvement in energy efficiency compared to the accurate adder. From Figure 7, we observe that HOAANED, HERLOA and M-HERLOA achieve 19.9%, 14.3% and 17.5% reduction in PDP, respectively, compared to the accurate adder for an FPGA implementation, and 41.1%, 35.6% and 39.9% reduction in PDP, respectively, compared to the accurate CLA for an ASIC-type implementation.

## 5. Conclusions

A comparative analysis of different gate-level SAAs, suitable for both FPGA and ASIC type implementations, has been performed in this work. Digital image processing was considered as an example application and the image processing results were shown. The error metrics of approximate adders corresponding to the image processing application were calculated and provided for a comparison. Further, the design metrics of accurate and approximate adders commensurate with the target application were provided corresponding to FPGA and ASIC type implementations. The objective is to identify those approximate adders that would facilitate an acceptable compromise between output quality and savings in design metrics compared to the accurate adder. In this context, HOAANED, HERLOA and M-HERLOA are found to be preferable.

**Author Contributions:** Conceptualization, P.B., R.N. and D.L.M.; methodology, P.B., R.N. and D.L.M.; software, P.B. and R.N.; validation, P.B., R.N. and D.L.M.; investigation, P.B. and R.N.; resources, D.L.M.; data curation, P.B. and R.N.; writing—original draft preparation, P.B.; writing–review & editing, P.B.; visualization, P.B. and R.N.; supervision, P.B. and D.L.M.; project administration, P.B. and D.L.M.; funding acquisition, D.L.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Ministry of Education (MOE), Singapore under an academic research fund Tier-2 grant number MOE2018-T2-2-024.

**Conflicts of Interest:** The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

## References

1. Available online: https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu (accessed on 12 November 2021).
2. Available online: https://research.google/pubs/pub41694/ (accessed on 12 November 2021).
3. Available online: https://www.tomshardware.com/news/google-tensor-processing-unit-machine-learning,31834.html (accessed on 13 March 2018).
4. Jouppi, N.P.; Young, C.; Patil, N.; Patterson, D.; Agrawal, G.; Bajwa, R.; Bates, S.; Bhatia, S.; Boden, N.; Borchers, A.; et al. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture, Toronto, ON, Canada, 24–28 June 2017.
5. Hartigan, J.A.; Wong, M.A. A k-means clustering algorithm. *J. R. Stat. Soc. Ser. C Appl. Stat.* **1979**, *28*, 100–108.
6. Chippa, V.K.; Mohapatra, D.; Roy, K.; Chakradhar, S.T.; Raghunathan, A. Scalable effort hardware design. *IEEE Trans. VLSI Syst.* **2014**, *22*, 2004–2016. [CrossRef]
7. Grigorian, B.; Reinman, G. Accelerating divergent applications on SIMD architectures using neural networks. *ACM Trans. Archit. Code Optim.* **2015**, *12*, 1–23. [CrossRef]
8. Han, J.; Orshansky, M. Approximate computing: An emerging paradigm for energy-efficient design. In Proceedings of the 18th IEEE European Test Symposium, Avignon, France, 27–30 May 2013.
9. Roy, K.; Raghunathan, A. Approximate computing: An energy-efficient computing technique for error resilient applications. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI, Montpellier, France, 8–10 July 2015.
10. Saadat, H.; Parameswaran, S. Hardware approximate computing: How, why, when and where? In Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, Seoul, Korea, 15–20 October 2017.
11. Sampson, A.; Deitl, W.; Fortuna, E.; Gnanapragasam, D.; Ceze, L.; Grossman, D. EnerJ: Approximate data types for safe and general low-power computation. In Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, San Jose, CA, USA, 4–8 June 2011.
12. Shoushtari, M.; Rahmani, A.M.; Dutt, N. Quality-configurable memory hierarchy through approximation. In Proceedings of the 14th International Conference on Compilers, Architecture, and Synthesis for Embedded Systems, Taipei, Taiwan, 9–14 October 2011.
13. Jiang, H.; Liu, C.; Liu, L.; Lombardi, F.; Han, J. A review, classification, and comparative evaluation of approximate arithmetic circuits. *ACM J. Emerg. Technol. Comput. Syst.* **2017**, *13*, 1–37. [CrossRef]
14. Scarabottolo, I.; Ansaloni, G.; Constantinides, G.A.; Pozzi, L.; Reda, S. Approximate logic synthesis: A survey. *Proc. IEEE* **2020**, *108*, 2195–2213. [CrossRef]
15. Hennessy, J.; Patterson, D. *Computer Architecture: A Quantitative Approach*, 5th ed.; Morgan Kaufmann: Burlington, MA, USA, 2003; ISBN 9780123838735.
16. Wanhammar, L. *DSP Integrated Circuits*, 1st ed.; Academic Press: Cambridge, MA, USA, 1999; ISBN 9780127345307.

17. Raha, A.; Jayakumar, H.; Raghunathan, V. Input-based dynamic reconfiguration of approximate arithmetic units for video encoding. *IEEE Trans. VLSI Syst.* **2016**, *24*, 846–857. [CrossRef]
18. Prabakaran, B.S.; Rehman, S.; Hanif, M.A.; Ullah, S.; Mazaheri, G.; Kumar, A.; Shafique, M. DeMAS: An efficient design methodology for building approximate adders for FPGA-based systems. In Proceedings of the Design, Automation and Test in Europe, Dresden, Germany, 19–23 March 2018.
19. Perri, S.; Spagnolo, F.; Frustaci, F.; Corsonello, P. Efficient approximate adders for FPGA-based data-paths. *Electronics* **2020**, *9*, 1529. [CrossRef]
20. Gupta, V.; Mohapatra, D.; Park, S.P.; Raghunathan, A.; Roy, K. IMPACT: Imprecise adders for low-power approximate computing. In Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design, Fukuoka, Japan, 1–3 August 2011.
21. Yang, Z.; Jain, A.; Liang, J.; Han, J.; Lombardi, F. Approximate XOR/XNOR-based adders for inexact computing. In Proceedings of the 13th IEEE International Conference on Nanotechnology, Beijing, China, 5–8 August 2013.
22. Zhang, T.; Liu, W.; McLarnon, E.; O'Neill, M.; Lombardi, F. Design of majority logic (ML) based approximate full adders. In Proceedings of the IEEE International Symposium on Circuits and Systems, Florence, Italy, 27–30 May 2018.
23. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-inspired computational blocks for efficient VLSI implementation of soft-computing applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 850–862. [CrossRef]
24. Albicocco, P.; Cardarilli, G.C.; Nannarelli, A.; Petricca, M.; Re, M. Imprecise arithmetic for low power image processing. In Proceedings of the 46th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 4–7 November 2012.
25. Gupta, V.; Mohapatra, D.; Raghunathan, A.; Roy, K. Low-power digital signal processing using approximate adders. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2013**, *32*, 124–137. [CrossRef]
26. Balasubramanian, P.; Maskell, D. Hardware efficient approximate adder design. In Proceedings of the IEEE Region 10 Conference, Jeju, Korea, 28–31 October 2018.
27. Balasubramanian, P.; Maskell, D.L.; Prasad, K. Approximate adder with reduced error. In Proceedings of the IEEE 31st International Conference on Microelectronics, Nis, Serbia, 16–18 September 2019.
28. Dalloo, A.; Najafi, A.; Garcia-Ortiz, A. Systematic design of an approximate adder: The optimized lower part constant-OR adder. *IEEE Trans. VLSI Syst.* **2018**, *26*, 1595–1599. [CrossRef]
29. Lu, Q.; Gharehbaghi, A.M.; Fujita, M. Approximate arithmetic circuit design using a fast and scalable method. In Proceedings of the IFIP/IEEE 27th International Conference on Very Large Scale Integration, Cuzco, Peru, 6–9 October 2019.
30. Balasubramanian, P.; Maskell, D.L. Hardware optimized and error reduced approximate adder. *Electronics* **2019**, *8*, 1212. [CrossRef]
31. Lee, J.; Seo, H.; Kim, Y.; Kim, Y. Approximate adder design with simplified lower-part approximation. *IEICE Electron. Express* **2020**, *17*, 20200218. [CrossRef]
32. Lee, J.; Seo, H.; Kim, Y.; Kim, Y. Design of a low-cost approximate adder with a zero truncation. In Proceedings of the International SoC Design Conference, Yeosu, Korea, 21–24 October 2020.
33. Seo, H.; Kim, Y. A new approximate adder with duplicate-constant scheme for energy efficient applications. In Proceedings of the IEEE International Conference on Consumer Electronics–Asia, Seoul, Korea, 1–3 November 2020.
34. Balasubramanian, P.; Nayar, R.; Maskell, D.L.; Mastorakis, N.E. An approximate adder with a near-normal error distribution: Design, error analysis and practical application. *IEEE Access* **2021**, *9*, 4518–4530. [CrossRef]
35. Seo, H.; Yang, Y.S.; Kim, Y. Design and analysis of an approximate adder with hybrid error reduction. *Electronics* **2020**, *9*, 471. [CrossRef]
36. Balasubramanian, P.; Nayar, R.; Maskell, D. An approximate adder with reduced error and optimized design metrics. In Proceedings of the 17th IEEE Asia Pacific Conference on Circuits and Systems, Penang, Malaysia, 22–26 November 2021.
37. Zhu, N.; Goh, W.L.; Zhang, W.; Yeo, K.S.; Kong, Z.H. Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing. *IEEE Trans. VLSI Syst.* **2010**, *18*, 1225–1229.
38. Gibson, J.D. *Handbook of Image and Video Processing*; Gibson, J.D., Bovik, A., Eds.; Academic Press: Orlando, FL, USA, 2000; ISBN 978-0121197902.
39. Zhou, W.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612.
40. Chan, W.-T.J.; Kahng, A.B.; Kang, S.; Kumar, R.; Sartori, J. Statistical analysis and modeling for error composition in approximate computation circuits. In Proceedings of the 31st IEEE International Conference on Computer Design, Asheville, NC, USA, 6–9 October 2013.
41. Balasubramanian, P.; Maskell, D.L. Factorized carry lookahead adders. In Proceedings of the IEEE 14th International Symposium on Signals, Circuits and Systems, Iasi, Romania, 11–12 July 2019.
42. Synopsys SAED_EDK32/28_CORE Databook. Revision 1.0.0. January 2012. Available online: https://www.synopsys.com/community/university-program/teaching-resources.html (accessed on 21 July 2021).