electronics

MDPI

Article

Digital Image Blending by Inexact Multiplication

Padmanabhan Balasubramanian [1,*], Raunaq Nayar [2], Okkar Min [1] and Douglas L. Maskell [1]

[1] School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore
[2] Transport Research Centre, College of Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore
* Correspondence: balasubramanian@ntu.edu.sg; Tel.: +65-6790-4745

Abstract: Digital image blending is commonly used in applications such as photo editing and computer graphics where two images are combined to produce a desired blended image. Digital images can be blended by addition or multiplication, and usually exact addition or multiplication is performed for image blending. In this paper, we evaluate the usefulness of inexact multiplication for digital image blending. Towards this, we describe how an exact array multiplier can be made inexact by introducing vertical cut(s) in it and assigning distinct combinations of binary values to the dangling inputs and product bits. We considered many 8-bit digital images for blending and the blended images obtained using exact and inexact multipliers are shown, which demonstrates the usefulness of inexact multiplication for image blending. For 8 × 8 image blending, one of our inexact array multipliers viz. IAM01-VC8 was found to achieve 63.3% reduction in area, 21% reduction in critical path delay, 72.3% reduction in power dissipation, and 78.1% reduction in energy compared to the exact array multiplier. In addition, IAM01-VC8 achieved 60.6% reduction in area, 9.7% reduction in critical path delay, 64.7% reduction in power dissipation, and 68.1% reduction in energy compared to the high-speed exact 8 × 8 multiplier that was automatically synthesized using a logic synthesis tool. The exact and inexact multipliers were physically realized using 32/28 nm CMOS process technology.

Keywords: approximate computing; arithmetic circuits; multiplier; combinational logic; low power; high speed; CMOS

Citation: Balasubramanian, P.; Nayar, R.; Min, O.; Maskell, D.L. Digital Image Blending by Inexact Multiplication. *Electronics* **2022**, *11*, 2868. https://doi.org/10.3390/electronics11182868

Academic Editor: John Ball

Received: 11 August 2022
Accepted: 8 September 2022
Published: 10 September 2022

1. Introduction

Inexact computing is a promising alternative to exact computing for error-resilient practical applications [1,2] that facilitates higher speed, lesser power dissipation, and greater energy efficiency. Examples of such practical applications include digital image, video and audio processing [3–5], multimedia [6], big data and analytics [7], neuromorphic computing [8], hardware implementation of neural networks for AI and machine learning [9], software engineering [10], memory storage [11], memory systems for multicore processors [12], low-power graphics processing [13], etc. The limits of human perception pave the way for error resilience in many practical applications which involve digital signal processing. For example, minor distortions in digital images and video frames and feeble noise in a digital audio are not discernible by humans due to the innate limitations of human perception.

Inexact computing relates to hardware, software, and memory storage, and inexact hardware covers arithmetic circuits and logic circuits [3,4]. As regards inexact arithmetic circuits, the primary focus has been on the design of inexact adders and multipliers [5] since addition and multiplication are pervasive in general purpose microprocessors, digital signal processors, and application specific processors. In this context, this paper describes the designs of inexact array multipliers (IAMs) and evaluates their usefulness for a digital image-blending application. We compare the performance and design metrics of the exact

array multiplier (EAM) with different IAMs with respect to 8 × 8 image blending. The quality of image blending was quantified using standard signal processing metrics such as the signal-to-noise ratio and the structural similarity index. The error metrics of the IAMs were calculated, and the design metrics of EAM and IAMs were estimated after implementation using a 32/28 nm CMOS process technology.

A preliminary version of this work was presented in [14], and this paper is an extended version that presents additional blended images obtained by experimentation, which consistently demonstrates the superiority of one of our IAMs for image blending. In addition, the energy of the EAM and IAMs were estimated and shown for a comparison in this paper. The remainder of the paper is organized into five sections. Section 2 presents the schematic of an 8 × 8 EAM and describes how vertical cut(s) can be introduced in it and how different combinations of binary values can be assigned to the dangling inputs and product bits to generate different 8 × 8 IAMs. Nevertheless, the procedure of obtaining IAMs from an EAM is generic and it can be extended to address multiplication of any size. Section 3 gives the values of some popular error metrics viz. mean absolute error (also called mean error distance) and root mean square error, calculated for the IAMs. Section 4 discusses image blending and shows examples of blended images obtained using the EAM and IAMs. Section 5 gives the design metrics, namely area, critical path delay, and average power dissipation of EAMs and IAMs. The EAMs and IAMs were described in Verilog hardware description language and synthesized using a logic synthesis tool by using a 32/28 nm CMOS standard digital cell library. Section 6 concludes the paper.

2. Exact and Inexact Array Multipliers

We consider the blending of two 8-bit digital images as a practical application to evaluate the usefulness of IAMs. To blend two 8-bit images, an 8 × 8 multiplier is sufficient. To perform small multiplications, an 8 × 8 multiplication for example, the array multiplier architecture is preferable as it has a regular structure and is thus convenient to layout [15], and it can also be easily pipelined to increase the throughput as needed.

The schematic of an 8 × 8 EAM is shown in Figure 1, where P7 to P0 and Q7 to Q0 represent the multiplier inputs, and M15 to M0 represent the product bits. In the multiplier inputs, bits P7 and Q7 are the most significant while bits P0 and Q0 are the least significant. In the multiplier output, bit M15 is most significant while bit M0 is the least significant. An 8 × 8 EAM generates 64 partial products which can be realized using an array of 2-input AND gates. In Figure 1, P7Q7 up to P0Q0 represent the 64 partial products, which result from the multiplication of input bits P7 to P0 with Q7 to Q0. The EAM incorporates a carry save adder comprising half adders and full adders, where the carry outputs generated from the half adders and/or full adders in one logic level are passed on as carry inputs to the half adders and/or full adders present in the next logic level. A binary half adder adds two input bits and produces a sum bit and any carry overflow, while a binary full adder adds two input bits including any carry input and produces a sum bit and any carry overflow. An 8 × 8 EAM comprises 8 half adders and 48 full adders.

In Figure 1, vertical cuts labelled as VC0 up to VC13 are shown as examples by blue dashed lines, and any of these vertical cuts may be made in an EAM to generate an IAM. Basically, after making a vertical cut in an EAM, the circuit portion on the right side of the cut will be eliminated and the circuit portion on the left side of the cut would be retained, but with some modification. The deeper the vertical cut, more logic would be eliminated, and hence, more inaccuracy would be introduced in the multiplication. For example, vertical cut VC0 would eliminate just one 2-input AND gate, whereas vertical cut VC2 would eliminate five 2-input AND gates which realize the partial products P2Q0, P1Q1, P1Q0, P0Q1, and P0Q2; one 2-input AND gate that produces M0; and 2 half adders and 1 full adder. Therefore, the choice of a vertical cut should be made commensurate with the extent of inaccuracy that would be deemed acceptable for a practical application. For an illustration, Figure 2 shows the result of a vertical cut VC8 made on the EAM. The circuit portion on the right side of VC8 is eliminated. The full adders shown in rose in

Figure 2 have one of their inputs cut, which are referred to as dangling inputs; these may be assigned a constant binary value of 0 or 1. The full adder that produces product bit M9 in Figure 1 has both its inputs cut in Figure 2, so it is eliminated; the sum output of the full adder present above it is now labelled as M9. The full adder that produces product bit M10 in Figure 1 has one of its inputs cut in Figure 2, so it is converted to a half adder, which is shown in rose. After making the vertical cut VC8 on the EAM, product bits M8 up to M0 will become dangling, and hence, they can be assigned a constant binary value of 0 or 1, as shown in Figure 2. Given these, after making a vertical cut on an EAM, four generalized IAM architectures can be derived, which are given as follows:

- IAM00–binary 0 is assigned to the dangling input of the full adders (shown in rose), and binary 0 is assigned to the dangling product bits (here, M8 up to M0)
- IAM01–binary 0 is assigned to the dangling input of the full adders (shown in rose), and binary 1 is assigned to the dangling product bits (here, M8 up to M0)
- IAM10–binary 1 is assigned to the dangling input of the full adders (shown in rose), and binary 0 is assigned to the dangling product bits (here, M8 up to M0)
- IAM11–binary 1 is assigned to the dangling input of the full adders (shown in rose), and binary 1 is assigned to the dangling product bits (here, M8 up to M0)

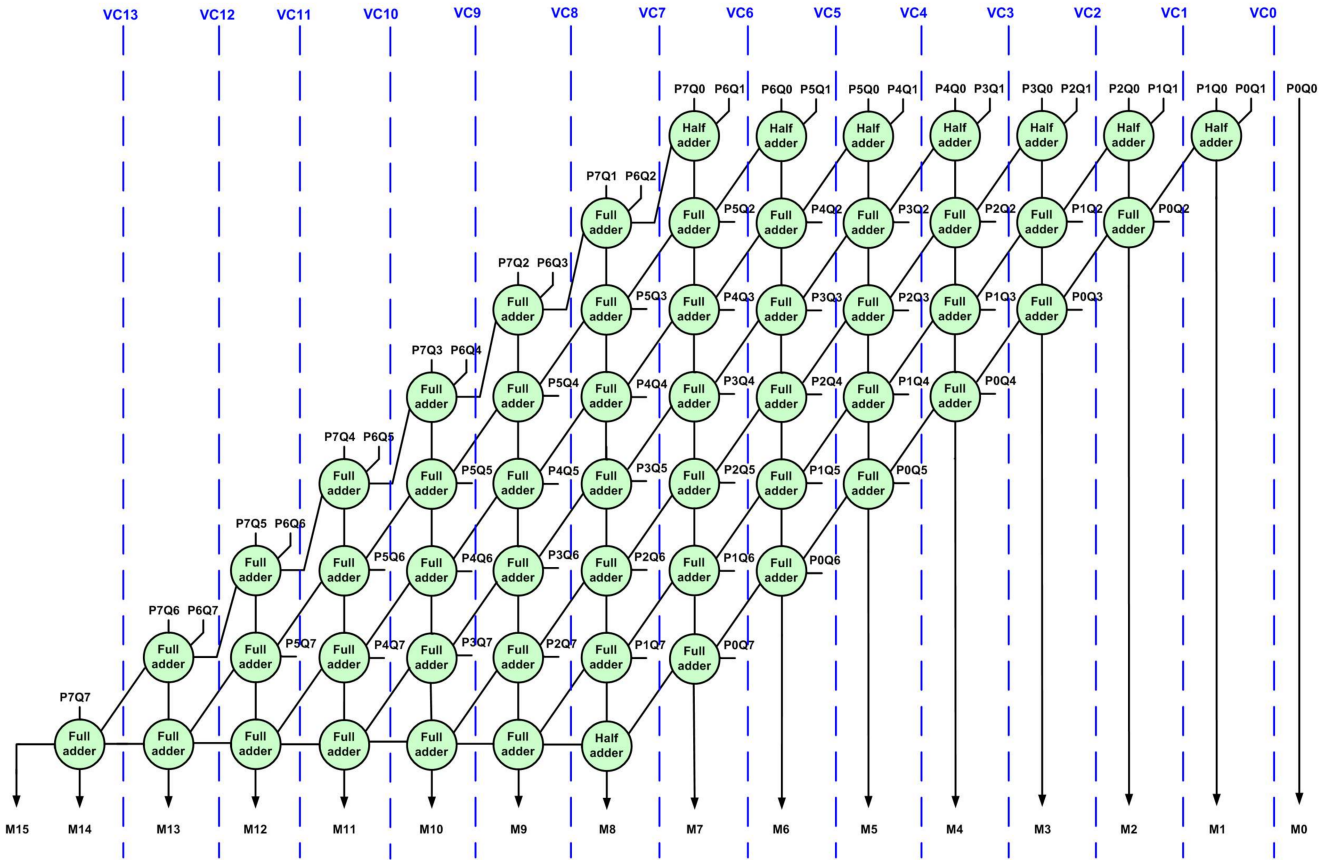

Figure 1. Architecture of an 8 × 8 exact array multiplier (EAM). Example vertical cuts VC0 up to VC13, any of which could be made on an EAM to generate an inexact array multiplier (IAM), are shown as blue dashed lines.

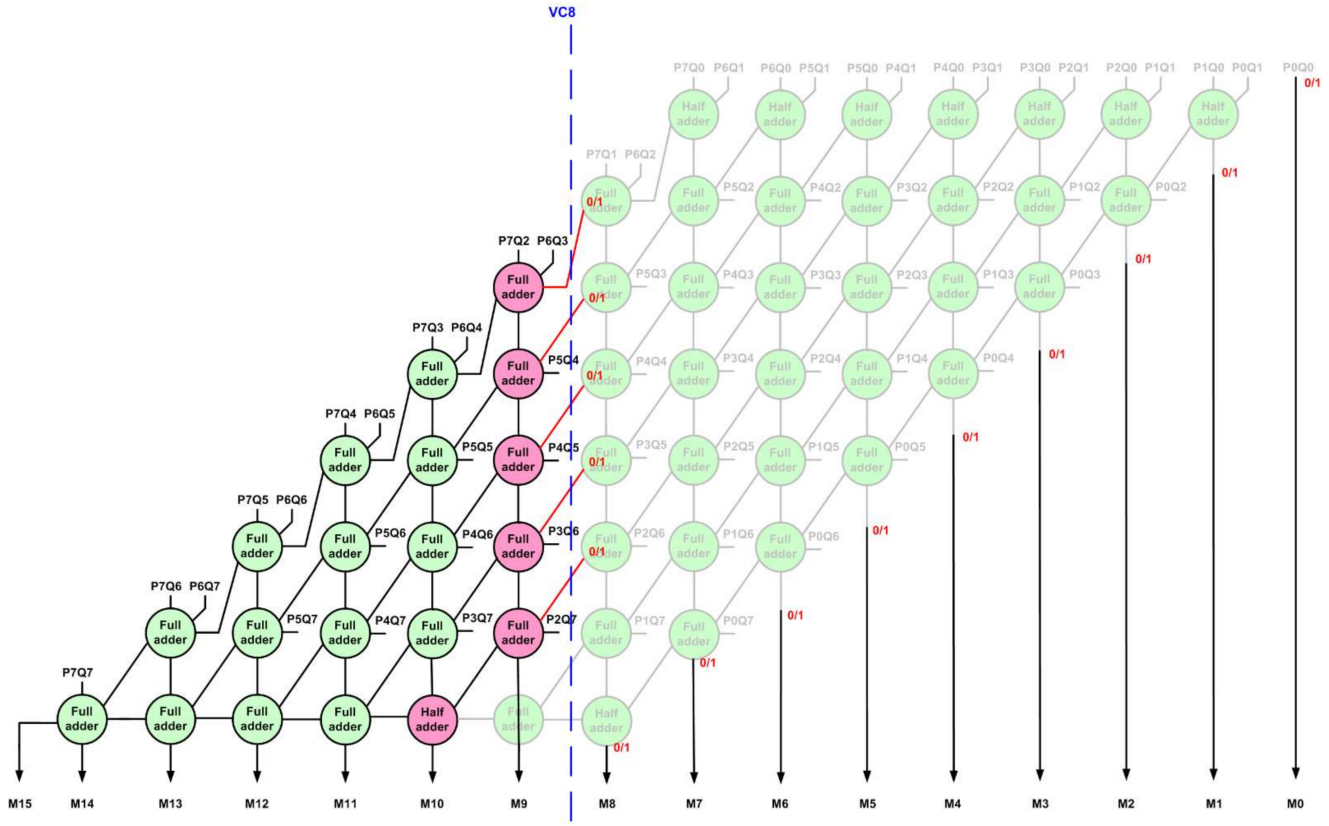

Figure 2. Portrayal of how IAMs are derived by making an example vertical cut VC8 on an 8×8 EAM.

Among the above, IAM00 architecture was presented in [16,17], and IAM01, IAM10, and IAM11 architectures are our propositions. IAM00 and IAM10 architectures have a commonality in that the dangling product bits are assigned a binary 0, and IAM01 and IAM11 architectures have a commonality in that the dangling product bits are assigned a binary 1.

When binary 0 is given as an input to the full adders shown in rose in Figure 2, they would be transformed into half adders. To explain this, let us presume that A, B, and C represent the inputs of a full adder, and SUM and COUT represent its sum and carry outputs, which are expressed by SUM = A⊕B⊕C and COUT = (A⊕B) C + AB, respectively. Supposing one of the inputs, say C, is assigned a binary 0, then SUM = A⊕B and COUT = AB, which represent the equations of a half adder. Therefore, the full adders shown in rose in Figure 2 can be replaced by half adders, which is applicable to IAM00 and IAM01 architectures. Alternatively, if say C is assigned a binary 1 (i.e., binary 1 is given as an input to the full adders shown in rose in Figure 2), then SUM = A⊙B, and COUT = A + B, which can be realized using a 2-input XNOR gate and a 2-input OR gate. Hence, the full adders shown in rose in Figure 2 can be reduced to the combination of a 2-input XNOR gate and a 2-input OR gate, which is applicable to IAM10 and IAM11 architectures.

The IAMs resulting from the vertical cut VC8 made on an 8×8 EAM, after performing the above-mentioned logic optimization, is shown in Figure 3, where Figure 3a depicts examples of IAM00 and IAM01 architectures, and Figure 3b depicts examples of IAM10 and IAM11 architectures. Referring to Figure 3a, IAM00 would have product bits M8 up to M0 assigned a binary 0, and IAM01 would have product bits M8 up to M0 assigned a binary 1. Referring to Figure 3b, IAM10 would have product bits M8 up to M0 assigned a binary 0, and IAM11 would have product bits M8 up to M0 assigned a binary 1.

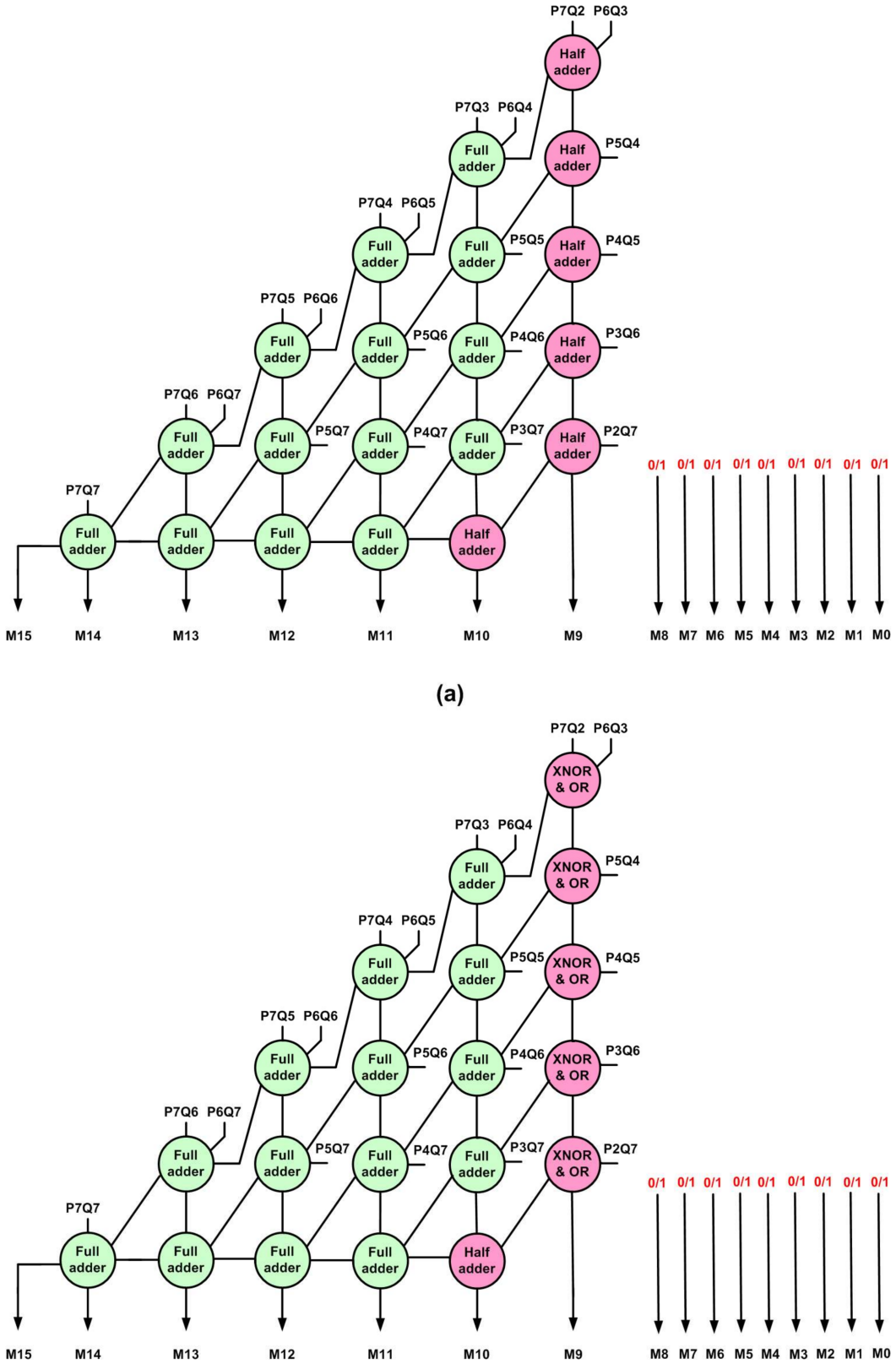

Figure 3. (**a**) Logic schematic of IAM00 and IAM01 architectures resulting from vertical cut VC8 made on the 8 × 8 EAM. (**b**) Logic schematic of IAM10 and IAM11 architectures resulting from vertical cut VC8 made on the 8 × 8 EAM.

3. Error Metrics of Inexact Array Multipliers

We calculated two widely used error metrics for the IAMs, namely the mean absolute error (MAE) and the root mean square error (RMSE). Among these, RMSE is more useful as it effectively quantifies the extent of signal degradation in digital signal processing [18]. An 8×8 multiplier would have a total of 2^{16} distinct inputs; the entire set of distinct inputs was considered to accurately calculate MAE and RMSE for different IAMs. Towards this, Python models of 8×8 EAM and 8×8 IAMs were developed. For each distinct input supplied, the absolute difference between the product produced by the EAM and the product produced by each IAM was calculated to subsequently calculate MAE and RMSE for the IAMs. MAE and RMSE were calculated using Formulas (1) and (2). In Formulas (1) and (2), EAM_Product(P, Q) denotes the product of an EAM with inputs P and Q, and IAM_Product(P, Q) denotes the product of an IAM with the same inputs of P and Q.

$$\text{MAE} = \sum_{P=0}^{2^8-1} \sum_{Q=0}^{2^8-1} |\text{IAM\_Product}(P,Q) - \text{EAM\_Product}(P,Q)| \quad (1)$$

$$\text{RMSE} = \sqrt{\frac{1}{2^{16}} \sum_{P=0}^{2^8-1} \sum_{Q=0}^{2^8-1} \left(\begin{array}{c} \text{IAM\_Product}(P,Q) \\ -\text{EAM\_Product}(P,Q) \end{array} \right)^2} \quad (2)$$

Generally, the reductions in design metrics such as critical path delay, area, and power dissipation for an inexact circuit compared to the exact circuit would depend on the extent of inaccuracy incorporated [19]. The extent of inaccuracy incorporated is typically proportional to the savings in design metrics achievable for an inexact circuit compared to the exact circuit. However, output quality (referring to the quality of blended images here) assumes a greater priority than the savings in design metrics gained for an inexact circuit compared to the exact circuit. Therefore, incorporating an optimum (i.e., maximum allowable) inaccuracy in an inexact circuit commensurate with a target application is more important than achieving significant savings in the design metrics. An optimum inaccuracy implies achieving a good trade-off between output quality and circuit performance in inexact computing. Choosing a less-than-optimum inaccuracy would pave the way for increased accuracy in the computation (here, multiplication) but would also mean including more logic and therefore would result in reduced savings in design metrics for an inexact circuit compared to the exact circuit. On the other hand, choosing a more-than-optimum inaccuracy implies aggressive reduction of logic to greatly reduce the design metrics, but at the expense of compromising on the output quality, which may not be acceptable for a target application. Therefore, a more-than-optimum inaccuracy is not preferable.

For the image blending application, based on repeated experimentation, we found that vertical cut VC8 corresponds to an optimum inaccuracy, as noted in [14]. Given this, vertical cuts VC0 up to VC7 would be construed as leading to less-than-optimum inaccuracy, and vertical cuts VC9 to VC13 would be construed as leading to more-than-optimum inaccuracy. We calculated MAE and RMSE for IAMs corresponding to vertical cuts VC6 up to VC10 covering two less-than-optimum, one optimum, and two more-than-optimum representative inaccuracy conditions; the respective error metrics are given in Table 1. In Table 1, the label of the vertical cut is mentioned as a suffix to the IAM architectures for clarity. Optimum MAE and RMSE values obtained for an IAM corresponding to each vertical cut have been shown in boldface.

From Table 1, it can be seen that the IAM01 architecture consistently enables reduced MAE and RMSE compared to the other IAM architectures for all the vertical cuts. This is possibly because in the IAM01 architecture, even though the dangling inputs of the full adders (which are shown in rose in Figure 2) are assigned binary 0, which may impact the value of product bit M9, this would be compensated by the assignment of a binary 1 to the less significant product bits viz. M8 up to M0. Hence, there is a likelihood for an internal error compensation to happen in the IAM01 architecture, which is the possible

reason for its reduced error metrics compared to other IAM architectures. This suggests that the IAM01 architecture is likely to result in better quality of blended images compared to the quality of blended images obtained using other IAM architectures. The IAM11 architecture shares a commonality with the IAM01 architecture, given the fixed assignment of binary 1 to product bits M8 up to M0. However, in the case of IAAM11 architecture, binary 1 is input to all the full adders (shown in rose in Figure 2), which may exacerbate the error in the internal computation due to the higher significance of product bits M9 up to M16; this is possibly the reason why the IAM11 architecture consistently reports greater MAE and RMSE compared to the other IAM architectures for all the vertical cuts. IAM10 architecture, which also has a constant binary 1 assigned to the full adders (shown in rose in Figure 2), like the IAM11 architecture, reports reduced error in comparison; this is because product bits M8 up to M0 of IAM10 architecture are assigned a binary 0 unlike the IAM11 architecture, thus resulting in relatively less error. In the case of IAM00 architecture, although product bits M8 up to M0 are assigned a binary 0, and a binary 0 is given as an input to the full adders (shown in rose in Figure 2), the assumption of a value of 1 by any significant product bit ranging from M9 up to M16 is likely to overshadow the error arising from the constant assignment of a binary 0 to the less significant product bits M8 up to M0. Therefore, the overall error of IAM00 architecture is likely to be lesser compared to IAM10 and IAM11 architectures, but not lesser than the IAM01 architecture, which inherently features error compensation. Hence, from the perspective of error metrics (given in Table 1), IAM01 architecture is found to be preferable to the other IAM architectures.

Table 1. Error metrics of various inexact array multipliers, obtained by making vertical cuts VC6 to VC10 on the exact array multiplier.

Inexact Array Multiplier	Mean Absolute Error	Root Mean Square Error
Vertical cut VC6 made on EAM		
IAM00-VC6	192.25	224.032
IAM01-VC6	**101.362**	**132.241**
IAM10-VC6	575.750	587.127
IAM11-VC6	702.75	712.101
Vertical cut VC7 made on EAM		
IAM00-VC7	448.25	513.169
IAM01-VC7	**243.854**	**315.848**
IAM10-VC7	1087.774	1116.071
IAM11-VC7	1342.750	1365.793
Vertical cut VC8 made on EAM		
IAM00-VC8	896.25	1024.757
IAM01-VC8	**484.249**	**628.713**
IAM10-VC8	1664.762	1736.354
IAM11-VC8	2174.798	2230.785
Vertical cut VC9 made on EAM		
IAM00-VC9	1664.25	1916.367
IAM01-VC9	**876.747**	**1146.270**
IAM10-VC9	2444.210	2610.774
IAM11-VC9	3455.707	3583.020
Vertical cut VC10 made on EAM		
IAM00-VC10	2944.25	3435.35
IAM01-VC10	**1514.87**	**1984.46**
IAM10-VC10	3303.20	3656.69
IAM11-VC10	5256.70	5537.27

4. Digital Image Blending

Digital image blending has been considered as a practical application to evaluate the performance of different IAMs versus the EAM. Two digital images of size 512 pixels × 512 pixels were considered for blending, and their grayscale resolution is 8 bits. The blended image is also of size 512 pixels × 512 pixels. The gray values of the images are multiplied pixel by pixel, and the grayscale resolution of the blended image is 16 bits. We considered some standard digital images from [20] viz. *Lena*, *cameraman*, *Einstein*, and *woman with dark hair* for blending with a *mask* image. Figures 4–7 show the blended images obtained via exact and inexact multiplications. Figures 4a, 5a, 6a and 7a show the original images and the blended images obtained by exact multiplication. It was shown in [14] that vertical cut VC8 represents an optimum inaccuracy with respect to image blending. Figures 4b, 5b, 6b and 7b show the blended images obtained by inexact multiplication using different IAMs corresponding to vertical cut VC8 made on the EAM.

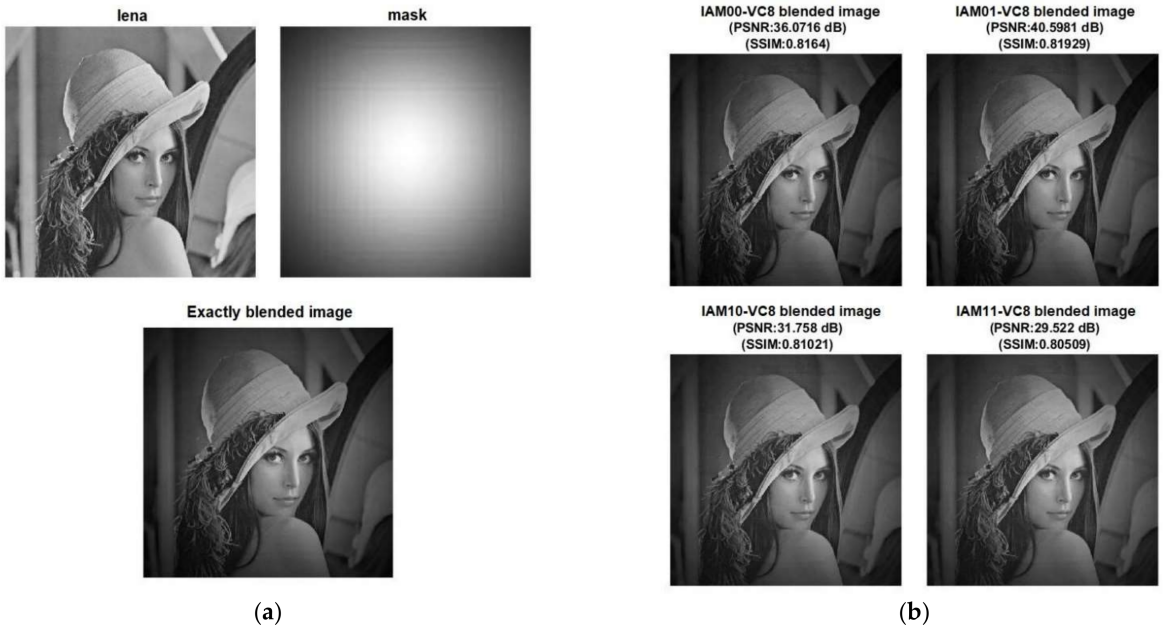

Figure 4. (**a**) *Lena* and *mask* images, and the exactly blended image. (**b**) Blended images obtained using various inexact array multipliers (IAM00-VC8, IAM01-VC8, IAM10-VC8, and IAM11-VC8) based on vertical cut VC8 made on the exact array multiplier.

To measure the quality of blended images, popular figures of merit such as peak signal-to-noise ratio (PSNR) [21] and structural similarity index metric (SSIM) [22] were used. PSNR is measured in decibels (dB). PSNR is infinite for the exactly blended image since no noise is introduced in the exact computation, whereas a finite PSNR results for the inexactly blended images due to the introduction of noise owing to inexact computation. Nevertheless, a high PSNR is preferred for the inexactly blended images, which is indicative of less distortion. Typically, a PSNR greater than 30 dB is preferred for 8-bit digital image processing [21]. SSIM is a measure of the structural similarity between a reference image (i.e., an exactly blended image) and a target image (i.e., an inexactly blended image). SSIM ranges from decimal 0 to 1, with decimal 0 indicating that the reference image and the target image are completely different and decimal 1 indicating that the reference image and the target image are the same. A high SSIM is preferred for an inexactly blended image, which indicates good similarity with the exactly blended image. The visual differences between the inexactly blended images in Figures 4b, 5b, 6b and 7b may be noticed upon close observation. Nevertheless, PSNR and SSIM are scientific metrics which can be used to ascertain the real quality of inexactly blended images obtained using different IAMs and to distinguish the difference in quality between them. From Figures 4b, 5b, 6b and 7b, it can

be seen that the IAM01 architecture (i.e., IAM01-VC8) consistently enables greater PSNR and SSIM for the inexactly blended images compared to the other IAM architectures. This is possibly due to the reduced error metrics (MAE and RMSE) of the IAM01 architecture compared to the other IAM architectures, as evident from Table 1.

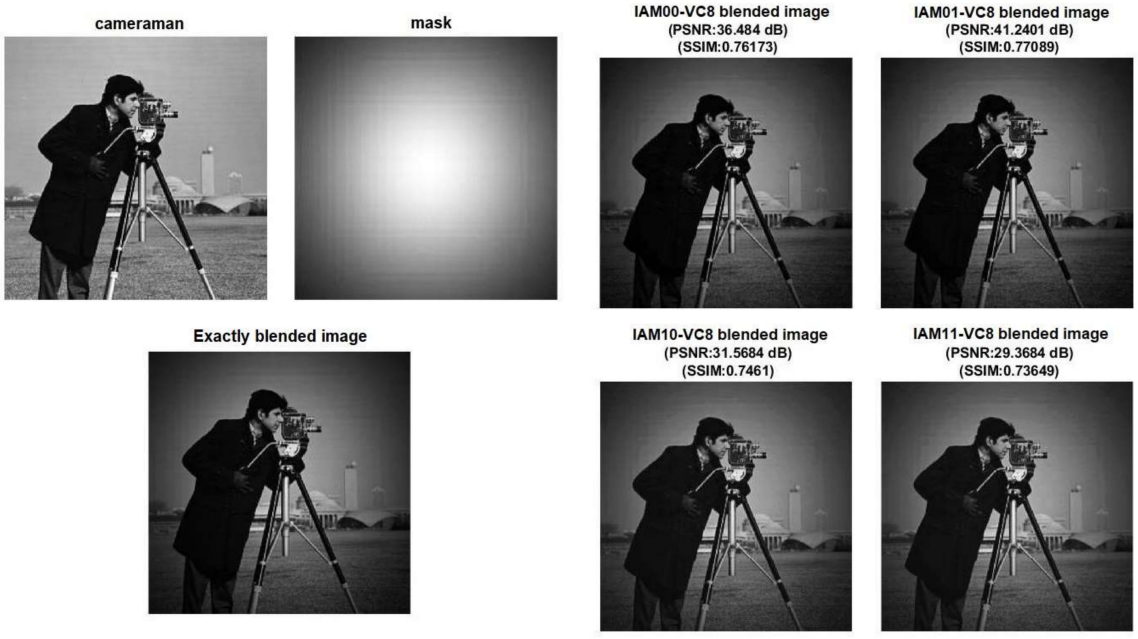

Figure 5. (**a**) *Cameraman* and *mask* images, and exactly blended image. (**b**) Blended images obtained using various inexact array multipliers (IAM00-VC8, IAM01-VC8, IAM10-VC8, and IAM11-VC8) based on vertical cut VC8 made on the exact array multiplier.

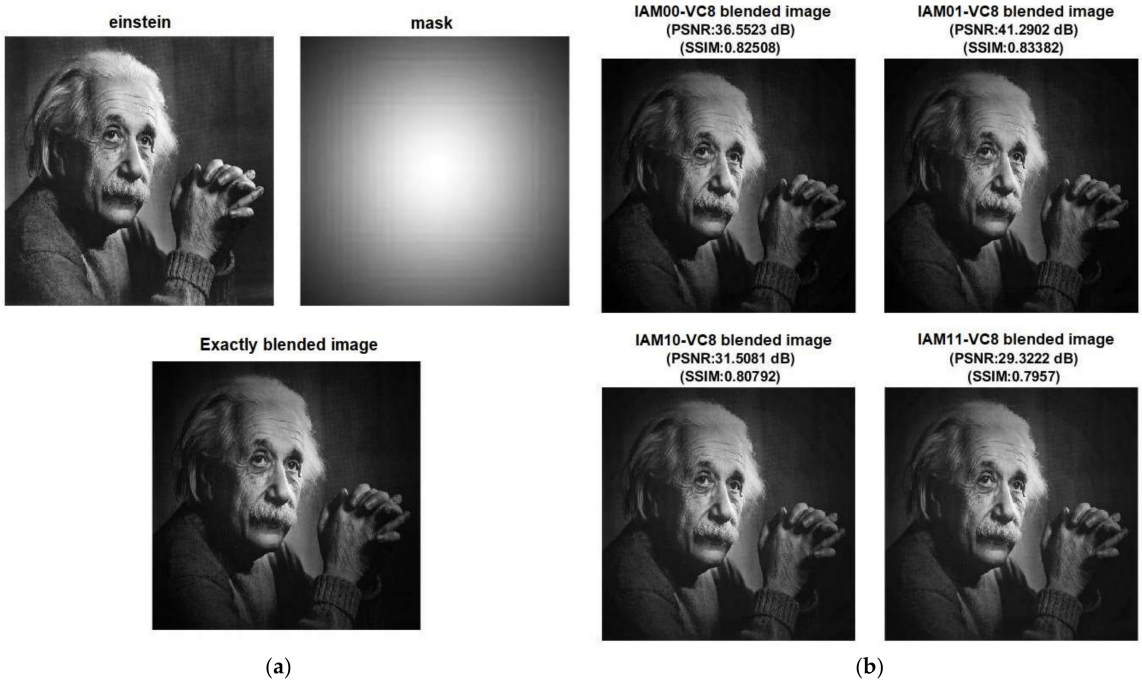

Figure 6. (**a**) *Einstein* and *mask* images, and the exactly blended image; (**b**) Blended images obtained using various inexact array multipliers (IAM00-VC8, IAM01-VC8, IAM10-VC8, and IAM11-VC8) based on vertical cut VC8 made on the exact array multiplier.

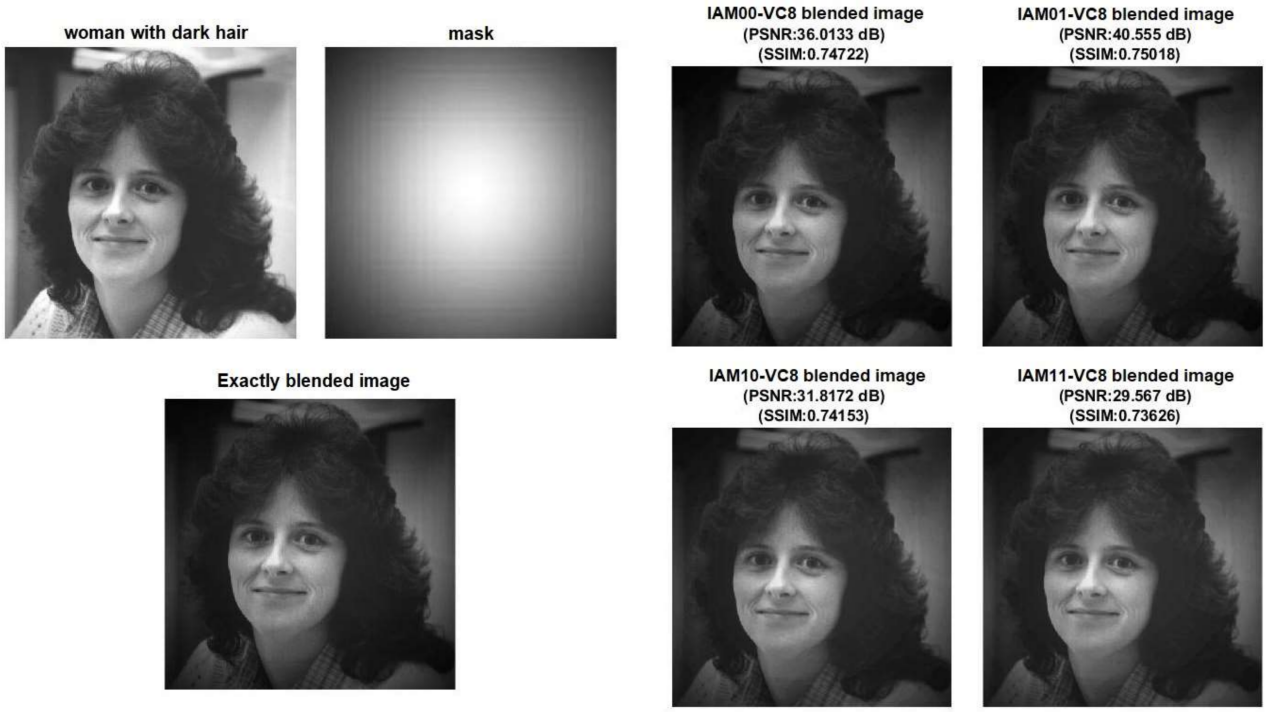

Figure 7. (**a**) *Woman* and *mask* images, and the exactly blended image. (**b**) Blended images obtained using various inexact array multipliers (IAM00-VC8, IAM01-VC8, IAM10-VC8, and IAM11-VC8) based on vertical cut VC8 made on the exact array multiplier.

5. Implementation and Design Metrics

We structurally described an exact 8 × 8 EAM and many 8 × 8 IAMs resulting from vertical cuts VC6 up to VC10 made on the EAM in Verilog HDL. We also behaviorally described an 8 × 8 exact multiplier in Verilog HDL by using the multiplication operator (*). These multipliers were then synthesized for high speed by a logic synthesis tool (Synopsys Design Compiler) using a 32/28 nm CMOS standard digital cell library [23]. The default wire load model (parasitic) was included in the synthesis, and a fanout-of-4 drive strength was assigned to the output ports i.e., to the product bits of multipliers. A typical case high-V_t CMOS process with a supply voltage of 1.05 V and an operating temperature of 25 °C was considered for synthesis and simulation. The gate-level netlists of the multipliers generated by Synopsys Design Compiler were verified by performing functional simulations using Synopsys VCS, and the switching activity was recorded for power estimation. For functional simulations, a test bench comprising about 1000 random input vectors was supplied to the multipliers at a timing of 2.5 ns (400 MHz). After synthesis, the design metrics of the multipliers were estimated, which are given in Table 2. The total area of the multipliers, including cell area and interconnect area, was estimated using Synopsys Design Compiler; the critical path delay was estimated using Synopsys PrimeTime; and the average power dissipation was estimated using Synopsys PrimePower.

From Table 2, it can be seen that IAM00 and IAM01 architectures feature the same design metrics, and IAM10 and IAM11 architectures also feature the same design metrics. As mentioned earlier, the difference between IAM00 and IAM01 architectures is that the dangling product bits (resulting from a vertical cut made on an EAM) are assigned binary 0 in the former and assigned binary 1 in the latter. The same difference manifests between IAM10 and IAM11 architectures. Binary 0 or 1 is assigned to a dangling product bit (arising from a vertical cut) by connecting it to ground or supply using a tie to low (TIEL) or a tie to high (TIEH) standard library cell. Since TIEL and TIEH cells of [23] have the same characteristics; therefore, IAM00 and IAM01 architectures have the same design metrics

since their internal logic is the same, and likewise, IAM10 and IAM11 architectures also have the same design metrics as their internal logic is the same.

Table 2. Standard design metrics of exact and inexact multipliers implemented using a 32/28 nm CMOS standard digital cell library.

Multiplier	Critical Path Delay (ns)	Area (μm^2)	Power Dissipation (μW)
Exact Multiplier	1.75	474.39	144.20
EAM	2.00	509.47	183.80
Vertical cut VC6 made on EAM			
IAM00-VC6 and IAM01-VC6	1.97	334.72	114.70
IAM10-VC6 and IAM11-VC6	1.99	329.62	122.30
Vertical cut VC7 made on EAM			
IAM00-VC7 and IAM01-VC7	1.69	260.41	73.29
IAM10-VC7 and IAM11-VC7	1.70	242.76	88.41
Vertical cut VC8 made on EAM			
IAM00-VC8 and IAM01-VC8	1.58	187.11	50.90
IAM10-VC8 and IAM11-VC8	1.53	179.99	57.68
Vertical cut VC9 made on EAM			
IAM00-VC9 and IAM01-VC9	1.26	138.47	30.68
IAM10-VC9 and IAM11-VC9	1.21	132.32	39.44
Vertical cut VC10 made on EAM			
IAM00-VC10 and IAM01-VC10	0.95	124.08	22.89
IAM10-VC10 and IAM11-VC10	0.96	122.81	27.68

It can be seen from Table 2 that as the order of the vertical cut increases, more logic is reduced and the critical path becomes shorter; hence, the reductions in design metrics increase for the IAMs compared to the exact multiplier and EAM. Compared to the 8×8 EAM, the exact 8×8 multiplier that was described using the multiplication operator and automatically synthesized for high speed by Design Compiler reports 12.5% reduction in critical path delay, 7.4% lesser area occupancy, and 21.5% reduction in power dissipation. It was noted in the previous sections that vertical cut VC8 made on the EAM is acceptable for image blending and the IAM01 architecture is preferable to its counterparts in terms of the quality of blended images and error metrics. Given this, IAM01-VC8 reportedly achieves the following reductions in design metrics compared to the exact multipliers: (i) 9.7% reduction in critical path delay, 60.6% reduction in area, and 64.7% reduction in power dissipation compared to the high-speed exact 8×8 multiplier that was auto-synthesized, and (ii) 21% reduction in critical path delay, 63.3% reduction in area, and 72.3% reduction in power dissipation compared to the 8×8 EAM.

For digital circuits and systems, the product of power dissipation and critical path delay (called PDP), representing energy, serves as a low-power figure of merit. Given this, we calculated PDP for exact and inexact multipliers, plotted in Figure 8. The PDP of the exact multiplier is plotted in blue, the PDP of EAM is plotted in purple, the PDP of IAM architectures corresponding to vertical cut VC6 is plotted in orange, the PDP of IAM architectures corresponding to vertical cut VC7 is plotted in green, the PDP of IAM architectures corresponding to vertical cut VC8 is plotted in red, the PDP of IAM architectures corresponding to vertical cut VC9 is plotted in black, and the PDP of IAM architectures corresponding to vertical cut VC10 is plotted in pink. Obviously, as the order of the vertical cut increases, the PDP of IAMs decreases due to a decrease in their design metrics, as noted from Table 2. From Figure 8, IAM01-VC8, which is suitable for image blending, achieved 68.1% reduction in energy compared to the high-speed exact 8×8 multiplier and 78.1% reduction in energy compared to the 8×8 EAM.

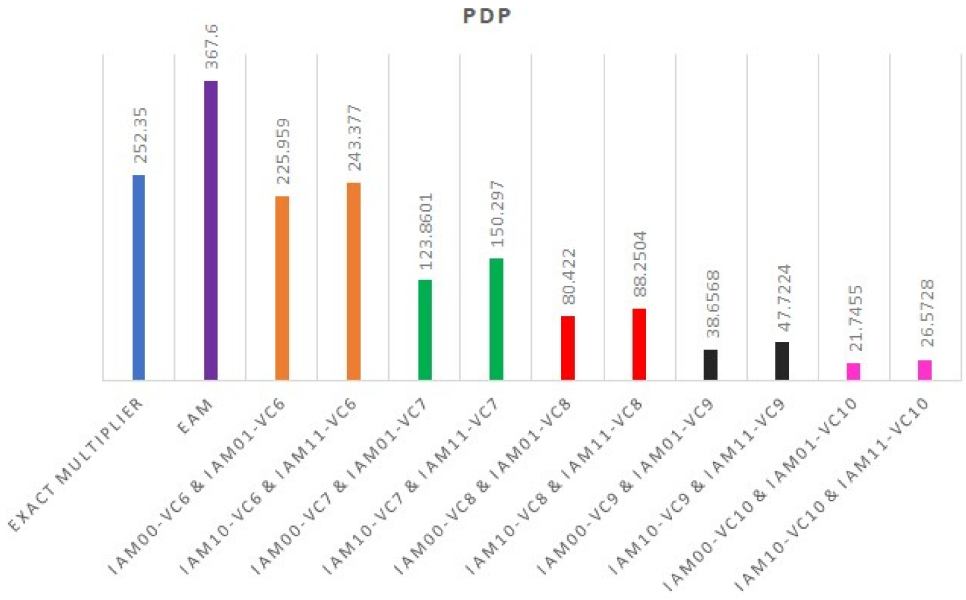

Figure 8. Power-delay product (PDP) of exact and inexact 8 × 8 multipliers.

6. Conclusions

Blending of digital images is commonly performed in computer graphics and photo editing applications. In this work, we analyzed the usefulness of inexact multiplication for image blending by comparison with exact multiplication. Towards this, we considered the blending of 8-bit images exactly and inexactly by employing exact and inexact multipliers, respectively. Different IAMs were derived by making vertical cut(s) on an EAM, and constant binary values were assigned to the dangling inputs and dangling product bits. Among the four IAM architectures, IAM01 architecture was found to be better in terms of having reduced error metrics (MAE and RMSE). Consequently, the IAM01 architecture was found to consistently yield better quality blended images compared to the other IAM architectures, where the quality of blended images has been quantified using standard metrics such as PSNR and SSIM. Vertical cut VC8 made on an EAM was found to be acceptable for 8-bit image blending, and based on the experimentation, the quality of blended images obtained using IAM01-VC8 was found to be acceptable in comparison with the quality of blended images obtained using the exact multiplier. The exact and inexact multipliers were implemented using a 32/28 nm CMOS standard cell library, and it was noted that whilst yielding blended images of acceptable quality, IAM01-VC8 achieved reductions in all the design metrics compared to the exact high-speed 8 × 8 multiplier and the 8 × 8 EAM. Notably, IAM01-VC8 consumes only 32% of the energy of an exact 8 × 8 multiplier and 22% of the energy of an 8 × 8 EAM.

Author Contributions: Conceptualization, P.B. and R.N.; methodology, P.B. and R.N.; software, P.B., R.N. and O.M.; validation, P.B. and R.N.; 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., R.N. and O.M.; 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, grant number MOE2018-T2-2-024.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data are available within the manuscript.

Conflicts 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. 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 May2013; pp. 1–6.
2. 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; pp. 473–475.
3. 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.
4. Zhu, N.; Goh, W.L.; Wang, G.; Yeo, K.S. Enhanced low-power high-speed adder for error-tolerant application. In Proceedings of the International SoC Design Conference, Incheon, Korea, 22–23 November 2010; pp. 323–327.
5. Jayakumar, A.R.H.; Raghunathan, V. Input-based dynamic reconfiguration of approximate arithmetic units for video encoding. *IEEE Trans. VLSI Syst.* **2016**, *24*, 846–857.
6. Breuer, M.A. Multi-media applications and imprecise computation. In Proceedings of the 8th Euromicro Conference on Digital System Design, Porto, Portugal, 30 August–3 September 2005; pp. 1–6.
7. Nair, R. Big data needs approximate computing: Technical perspective. *Commun. ACM* **2015**, *58*, 104. [CrossRef]
8. Panda, P.; Sengupta, A.; Sarwar, S.S.; Srinivasan, G.; Venkataramani, S.; Raghunathan, A.; Roy, K. Cross-layer approximations for neuromorphic computing: From devices to circuits and systems. In Proceedings of the 53rd Annual Design Automation Conference, Austin, TX, USA, 5–9 June 2016; pp. 1–6.
9. Sarwar, S.S.; Srinivasan, G.; Han, B.; Wijesinghe, P.; Jaiswal, A.; Panda, P.; Raghunathan, A.; Roy, K. Energy efficient neural computing: A study of cross-layer approximations. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 796–809. [CrossRef]
10. 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; pp. 164–174.
11. Sampson, A.; Nelson, J.; Strauss, K.; Ceze, L. Approximate storage in solid-state memories. *ACM Trans. Comput. Syst.* **2014**, *32*, 1–23. [CrossRef]
12. Shoushtari, M.; Rahmani, A.M.; Dutt, N. Quality-configurable memory hierarchy through approximation. In Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, Seoul, Korea, 15–20 October 2017; pp. 1–2.
13. Zhang, H.; Putic, M.; Lach, J. Low power GPGPU computation with imprecise hardware. In Proceedings of the 51st Design Automation Conference, San Francisco, CA, USA, 1–5 June 2014; pp. 1–6.
14. Balasubramanian, P.; Nayar, R.; Min, O.; Maskell, D.L. Image blending using approximate multiplication. In Proceedings of the IEEE 32nd International Conference on Microelectronics, Nis, Serbia, 12–14 September 2021; pp. 305–310.
15. Vai, M.M. *VLSI Design*; CRC Press: Boca Raton, FL, USA, 2000; ISBN 978-0849318764.
16. Mahdiani, H.R.; Ahmadi, A.; Fakhraie, S.M.; Lucas, C. Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2010**, *57*, 850–862. [CrossRef]
17. Yamamoto, T.; Taniguchi, I.; Tomiyama, H.; Yamashita, S.; Harz-Azumi, Y. A systematic methodology for design and analysis of approximate array multipliers. In Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems, Jeju, Korea, 25–28 October 2016; pp. 352–354.
18. 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; pp. 47–53.
19. 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]
20. Available online: https://www.imageprocessingplace.com/root_files_V3/image_databases.htm (accessed on 16 May 2022).
21. Gibson, J.D. *Handbook of Image and Video Processing*; Academic Press: Orlando, FL, USA, 2000; ISBN 978-0121197902.
22. Zhou, W.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Processing* **2004**, *13*, 600–612.
23. Synopsys. *Synopsys SAED_EDK32/28_CORE Databook*; Revision 1.0.0; Synopsys: Mountain View, CA, USA, 2012.