Circuits Syst Signal Process (2015) 34:3191–3219
DOI 10.1007/s00034-015-9997-4

Approximate Arithmetic for Low-Power Image Median Filtering

M. Monajati · S. M. Fakhraie · E. Kabir

Received: 10 March 2014 / Revised: 31 January 2015 / Accepted: 2 February 2015 /
Published online: 17 February 2015

Abstract In applications related to human senses, such as audio and image processing, computations with limited precision are acceptable. In these areas, a digital system can be implemented using approximate computing that works with sufficient precision. In this paper, we present a method to design 2-bit approximate magnitude comparators that are effectively low cost in terms of power, area and speed. We build larger comparators with adjustable error characteristics. Compared to precise one, our approximate comparators can save power and area up to 7–46 % and 10–50 %, respectively. The structures of our comparators and their error characteristics are presented in this paper. We use these comparators to design different approximate image median filters in order to remove salt and pepper noise. Simulation results show that the output quality of these filters is very similar to that of the precise ones so that the degradation is not noticeable. The approximate filters save up to 30 % of power and area while working 18 % faster than the precise ones.

Keywords Approximate computing · Magnitude comparator · Image processing · Median filter · High speed · Low power · Low cost

Professor Seyed Mehdi Fakhraie passed away on December 7, 2014, aged 54.

M. Monajati (✉) · E. Kabir
Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran
e-mail: mehrnaz.monajati@gmail.com; mehrnaz.monajati@modares.ac.ir

E. Kabir
e-mail: kabir@modares.ac.ir

S. M. Fakhraie
Faculty of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

1 Introduction

The field of image processing is ever expanding. It is used in many diverse applications such as monitoring, medicine, authentication, and space exploration [52]. Noise filtering is an essential component of image processing. The purpose of noise filtering is removing noise and its effects from the original image while keeping the distortion at the minimum possible value.

Image enhancement requires a series of filtering operations. These operations have high computational complexity and generally impose high burdens on speed and power consumption. A common type of image noise is salt and pepper where noisy pixels take either minimum or maximum values of intensities. This appears as black or white dots at random positions of the image [14].

Median filter is the most common rank-order filter used to remove salt and pepper noise [52]. In median filtering, an odd size window is shifted through all pixels of the image. The pixels in this window are ranked by their brightness values to find their median. This median value is assigned to the central pixel of the window. The main arithmetic operation of median filtering is sorting, which consumes a lot of power.

In applications related to human senses, selecting enough precision is sufficient [18]. Clearly, in these applications, using higher precision than the minimum required increases the cost of implementation. In such areas, an imprecise or approximate system that works well enough is quite acceptable. We call such a system an imprecise or approximate system. As said, median filter has high computational complexity. Its implementation is easy, but needs a large number of resources. Therefore, if we can use approximate calculations in the design of a median filter and make trade-offs between accuracy and hardware consumption, the computational complexity is reduced to facilitate real-time processing. As mentioned, the main part of a median filter is the data sorter that consists of a number of magnitude comparators (MCs). Accordingly, design of a simple and low-cost MC is very desirable.

In this paper, we propose an approximate method to design a low-power and low-cost magnitude comparator. To demonstrate this method, we use our comparator to implement a novel median filter. Our approximate median filter (APMF) achieves an acceptable output quality under low-cost requirements. There are two contributions in this paper. The first one is the design of a series of 2-bit imprecise magnitude comparators, 2BIMCs. The second is an architecture that uses them to build large-power efficient approximate MCs.

The remainder of this paper is organized as follows. Section 2 discusses the related work briefly. Our proposed techniques for various imprecise comparator cells are described in Sect. 3. In Sect. 4, we propose and discuss our hardware architectures for different imprecise median filters. In Sect. 4.3, we provide and compare the simulation results of using our proposed filters to remove salt and pepper noise in some typical images. Error evaluations and impact of the proposed filters on images are described in Sect. 4.4. Finally, some conclusions are drawn in the last section.

2 Related Works

The concept of imprecise computation has a rich history. Since the beginning of digital computing, the designers were interested to know what happens if calculations fail. In 1956, Von Neumann [60] proposed a computational model that can obtain reliable results from noisy gates and other uncertain components. That model requires a high level of redundancy (about 10^3–10^4) that seems unrealistic. Since then, a number of imprecise circuits have been designed for different purposes in various levels of circuit and logic [35], architecture [25], data flow [24] and system. Depending on the desire application, approximate calculations exploit a trade-off between accuracy, efficiency and power consumption.

Approximate computing can increase system performance in terms of complexity [7,10,28], area [17,19,31,36,46,51], power consumption [7,17,19,32,36,39], speed [3,7,8,10,31,46,51,53,63], throughput [31] and cost. Energy consumption is one of the most important challenges in the world of computing. Some methods use approximate computing to come up with low-power designs. These methods are voltage over scaling (VOS) [11,26,45,55], biased VOS (BIVOS) [5,6,13,34,42,44], bit width reduction [15,50,54], cutting lower significant parts [36] and hardware simplification [16,21,36,42,43,59]. Majority of these techniques focused on adders, multipliers and their derivative systems. In the rest of this section, we survey some of these methods.

Ning et al. [44] have proposed an Error Tolerant Adder, called ETA, in which the input operands are split into two parts of lower and higher significant bits. The addition process starts from the border between the two parts in opposite directions simultaneously. The more significant bits are added normally from right to left. For lower part, the corresponding input bits are compared from left to right. If both bits are equal to "1," then the checking process is stopped and sum of the remaining bits to the right are set to "1." This is illustrated in Fig. 1. Since carry out is not calculated for the lower part, ETA has better performance than a precise adder in terms of processing

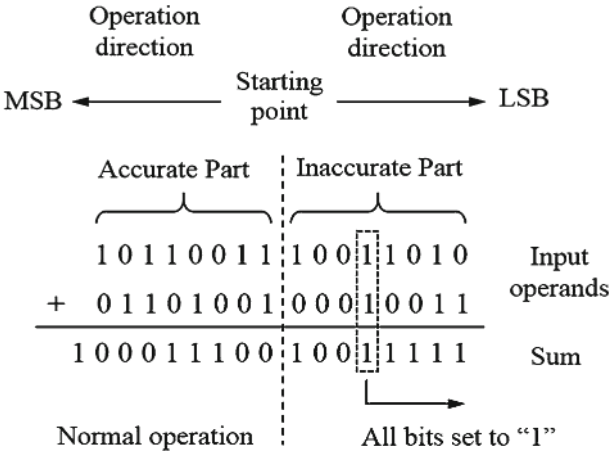


Fig. 1 ETA addition arithmetic [29]

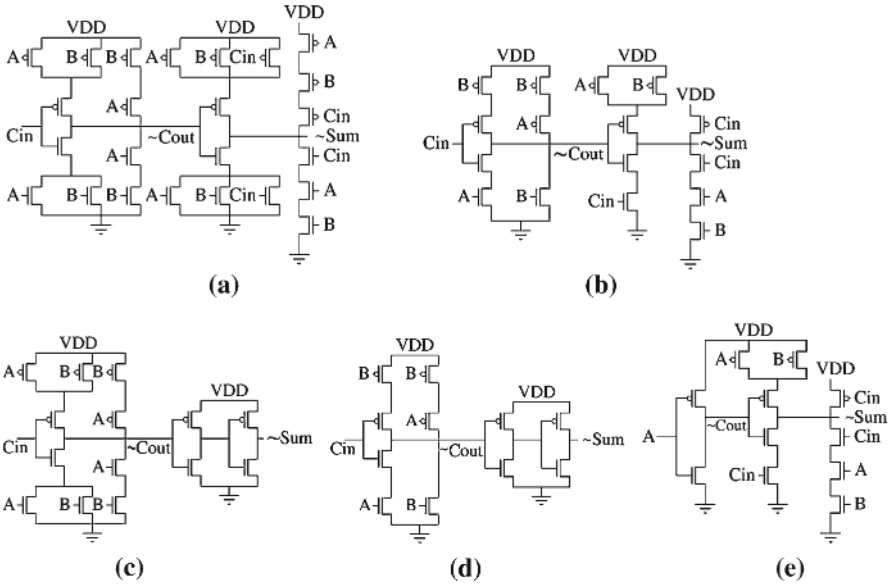


Fig. 2 Different versions of approximate MAs proposed in [15]. **a** Conventional MA, **b** Approximation 1 obtained by reducing some transistors from conventional MA, **c** Approximation 2 in which Sum $= \overline{Cout}$, **d** Approximation 3 in which $Cout$ has a more simplified circuit and Sum $= \overline{Cout}$, and **e** Approximation 4 in which $Cout$ = A and sum is calculated as in Approximation 1

speed, power and area consumption. The optimized and more efficient versions of ETA are ETAII [42], ETAIII [43] and ETAIV [45].

A low-power approximate multiplier architecture is proposed in [41] that uses a 2×2 inaccurate multiplier block resulting from manipulating the logic function of the multiplier. Tong et al. [54] reduced the bit width of the floating point multipliers to save power without affecting the accuracy of programs that operate on low-resolution data. Other works that focus on logic complexity reduction at the gate level are available in [36,48,53]. Some methods that use complexity reduction at the algorithm level to meet real-time energy constraints are reflected in [49,57].

Gupta et al. [17] proposed a logic complexity reduction at the transistor level by removing some of the series connected transistors in the mirror adder (MA) circuit. To limit the error within an acceptable range and maintain a reasonable output quality, they apply the approximation only to the FAs related to the least significant bits. Figure 2 shows the conventional MA and approximated versions proposed by Gupta et al. [15].

Mahdiani et al. [36] proposed an approximate adder named Lower-part-Or-Adder (LOA), as shown in Fig. 3a, in which the approximation is applied to the N less significant bits. In approximate adder, sum is calculated by applying bitwise OR to the individual input bits (lower part). Carry out of approximate adder is generated using just one AND gate for the two most significant bits of the lower part. This method eliminates the carry chain in lower part, which results in faster calculation besides

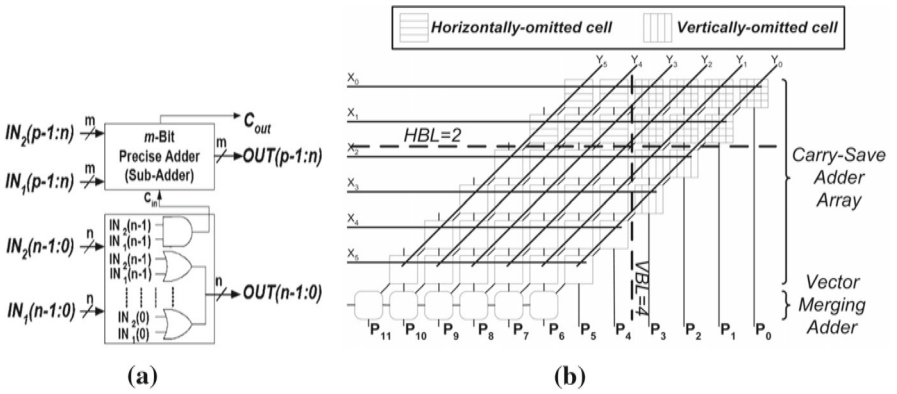


Fig. 3 Approximate computational blocks proposed in [11]. **a** LOA, and **b** BAM

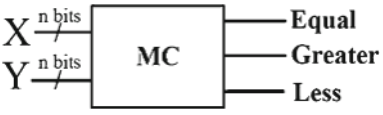


Fig. 4 Magnitude comparator (MC)

the fact that it consumes lower area and power. They also proposed Broken Array Multiplier (BAM), where the approximation is defined by removing the adder blocks in HBL rows from up to bottom and/or in VBL columns from left to right, as shown in Fig. 3b HBL and VBL indicate Horizontal Break Level and Vertical Break Level, respectively.

Data sorting is used in many fundamental processes in computational systems. It is an important building block in digital system design. Magnitude comparator (MC) is the main component in a data sorter and is used in general-purpose and application-specific architectures. Moreover, it is widely used in high-performance systems and has a great impact on their performance and power consumption. Therefore, designing a high-performance, low-cost and compact MC can be achieved using a proper approximation. As far as we know, there is no published work on the design of approximate comparators.

3 Approximate Magnitude Comparators

A magnitude comparator takes two binary numbers as inputs and compares them to determine their relative magnitude. As shown in Fig. 4, for inputs X, Y, the following modes are indicated: Greater, Less, and Equal, which mean $X > Y$, $X < Y$ and X=Y, respectively. In each mode, one output is 1 and the rest are zero.

In this section, several proposed methods for designing approximate comparator are presented and explained in details. The approximate MCs should have fewer logic gates than the precise one.

3.1 Two-Bit Approximate MC

Logic implementation of a circuit affects its speed, complexity, area consumption and power dissipation. Therefore, choosing a proper form for a logic function is very important. Our idea is to manipulate the logic function of MC in order to simplify its hardware by tolerating minor errors in its truth table in a way that its logic function needs fewer number of gates. It should not impose large errors on the circuit, and the final results should be acceptable. We propose some function simplifications of a 2-bit comparator. We show how larger MCs with adjustable error characteristics can be built from these 2-bit ones.

We design approximate logic circuits for less than (L) and greater than (G) outputs of a two-bit comparator (TBC). For this purpose, we change some cells of their related K-maps from 1 to 0 or vice versa in order to decrease the number of required gates. We define four types of approximations for each of L and G, as Ln and Gn. Figure 5 shows the precise and approximate K-maps for L and G. The added/deleted minterms of Ln and Gn with respect to the precise ones are given in Table 1. In all types of approximation, the probability of error is kept as low as 12.5 %, if input values are evenly distributed.

According to Fig. 5, there are four ways to approximate each L or G. Therefore, 16 different combinations of L and G can be used for a 2-bits MC. Hereafter, we mean the type of approximation by TLG. For example, T23 is a 2-bits MC, which uses L2 and G3. Precise and approximate circuit diagrams for L are shown in Fig. 6. As seen, the numbers of gates used in approximate circuits are less than those of accurate one. K-maps and circuit diagrams for imprecise G value are obtained similarly.

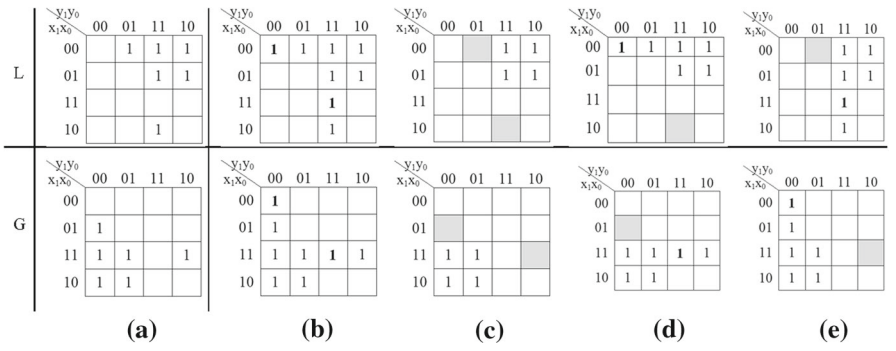


Fig. 5 Approximate Karnaugh map for Ln and Gn. **a** precise, **b** $n = 1$, **c** $n = 2$, **d** $n = 3$, and **e** $n = 4$. Added minterms are *bolded*, while deleted ones *shaded*

Table 1 Added and deleted minterms in different approximate L and G K-maps proposed

	L1	L2	L3	L4	G1	G2	G3	G4
Added minterms	m1, m15	–	m0	m15	m0, m15	–	m15	m0
Deleted minterms	–	m1, m11	m11	m1	–	m4, m14	m4	m14

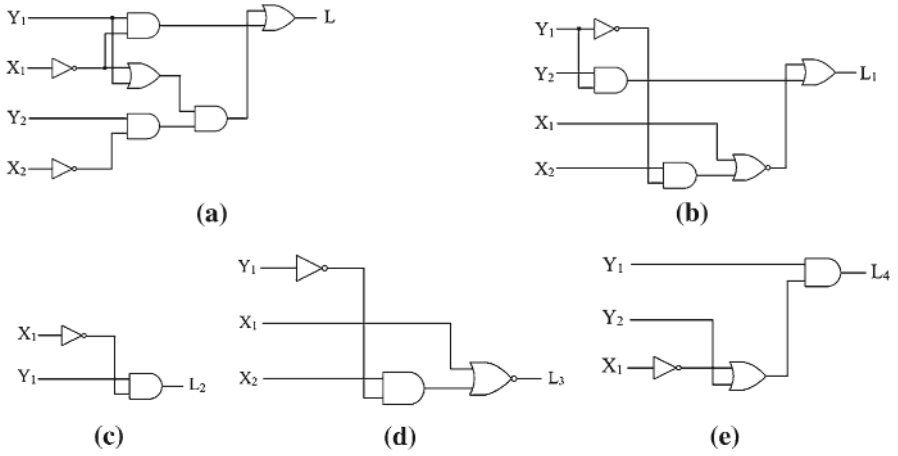


Fig. 6 Logic diagrams for precise and approximate L. **a** Precise, **b** L1, **c** L2, **d** L3, and **e** L4

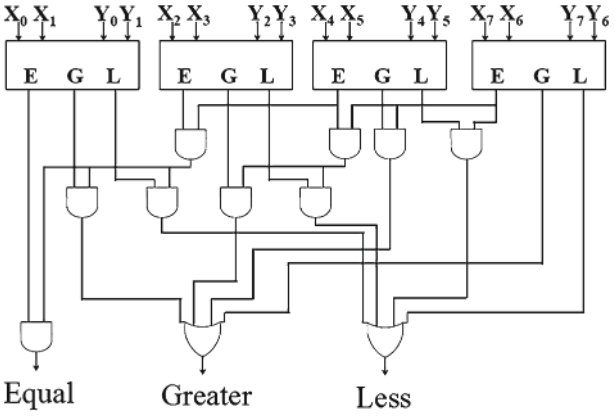


Fig. 7 An 8-bit MC using four 2-bit blocks

3.2 Building Larger Comparators

Larger MCs are built using 2-bit MCs. Generally, to compare the magnitude of X and Y, one starts from the most significant bits. If both input couples are equal, then the next lower significant bits are compared. Figures 7 and 8 demonstrate a method to design an 8-bit MC using four 2-bit.

As described above, the focus of our work is on the circuit design simplification of L and G, so the circuit which checks the equality remains unchanged, see Figs. 7 and 8. We move the circuitry responsible for equality checking out of the 2-bit comparators. In this way, we propose a new architecture for building larger comparators that we call it Regular Magnitude Comparator (RMC).

Our proposed architecture is regular and needs fewer gates than the previous one (Fig. 7), especially when L and G are approximated. It is illustrated in Fig. 9, where

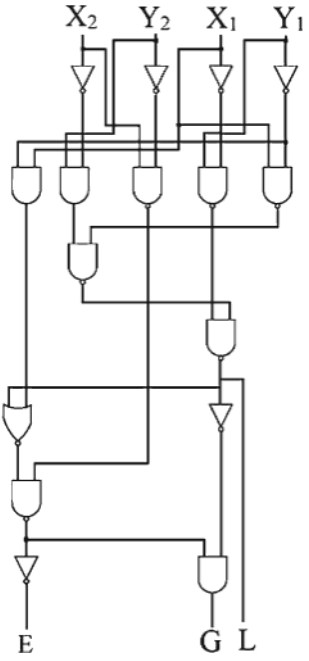


Fig. 8 Logic diagram for computing L, G, and E

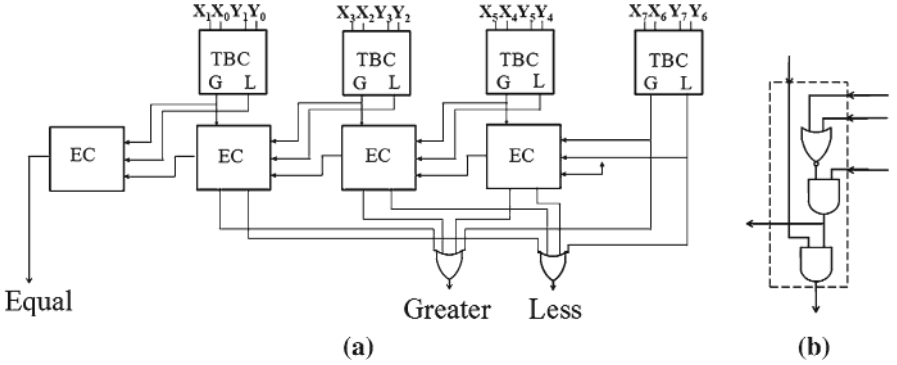
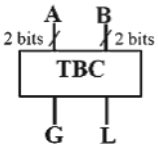


Fig. 9 **a** Block diagram of 8-bit RMC, and **b** EC block

Fig. 10 TBC block



TBC is a 2-bit comparator that determines whether one number is greater or less than the other one (Fig. 10). Each equality check (EC) block checks the equality of the previous more significant bits. In this way, the approximation is simply applied just on one or more TBC blocks, while the rest of the structure remains unchanged.

Table 2 Different modes of L and G

L	G	State
0	0	A = B
0	1	A > B
1	0	A < B
1	1	?

Table 3 Approximation types containing L = G = 1

Imprecision type	Y_1Y_0	X_1X_0
T11	00	00
T11	11	11
T13	00	00
T14	11	11
T31	11	11
T34	11	11

Table 2 shows different modes of L and G. The case of L=G=0 indicates the equality of both inputs of TBC. In the case of $L \neq G$, G=1 means that $A > B$ and L=1 indicates $A < B$. An ambiguous case is occurred when L=G=1. According to the K-maps in Fig. 5. a, this is not occurred in the precise TBC. In approximate designs, as shown in Table 3, this case occurs only when X is equal to Y and does not cause any problem in the functionality of RMC.

Equations (1) and (2) give the gate count and gate delay complexity for N-bit MC using the unit-gate model. Unit-gate model is a simple model of a circuit that reflects the circuit complexities and is usually used in the literatures [2,12]. In this model, every 2-input basic gate such as AND, OR, NAND, and NOR is considered as one unit gate. Complex gates and multi-input gates are built from 2-input basic gates. The gate count of the circuit equals to the sum of gate counts of its cells. Gate delay of the circuit is equal to sum of the gate delays of the cells located in its critical path.

$$\text{Gate count}_{\text{MC}} = N \times (\text{Gate count}_{\text{TBC}} + \text{Gate count}_{\text{EC}}) + 6 \quad (1)$$

$$\text{Gate delay}_{\text{MC}} = \text{Gate delay}_{\text{TBC}} + 7 \quad (2)$$

where the constant 6 is related to the two 4-bits OR gates and constant 7 is related to the colored gates in Fig. 11. The gate counts of EC and precise TBC are equal to 4 and 8, respectively.

The approximation can be applied to each TBC block. According to Fig. 9, the first TBC on the right is responsible for the most significant bits of inputs and the next ones receive less significant bits. So, total gate count for 8-bit MC is 102. As (1) shows, area consumption and speed of RMC depend on those of TBC. Therefore, simplifying this block and reducing its area leads to smaller and more efficient RMC.

We consider four levels of approximation for TBC. At the first level, only the first TBC on the left is designed approximately, and other blocks are accurate. Therefore,

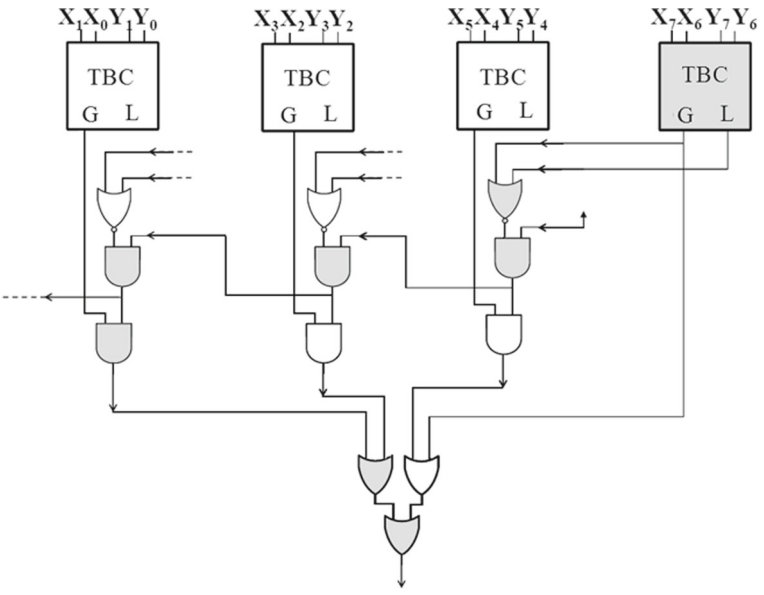


Fig. 11 Critical path of 8-bit RMC

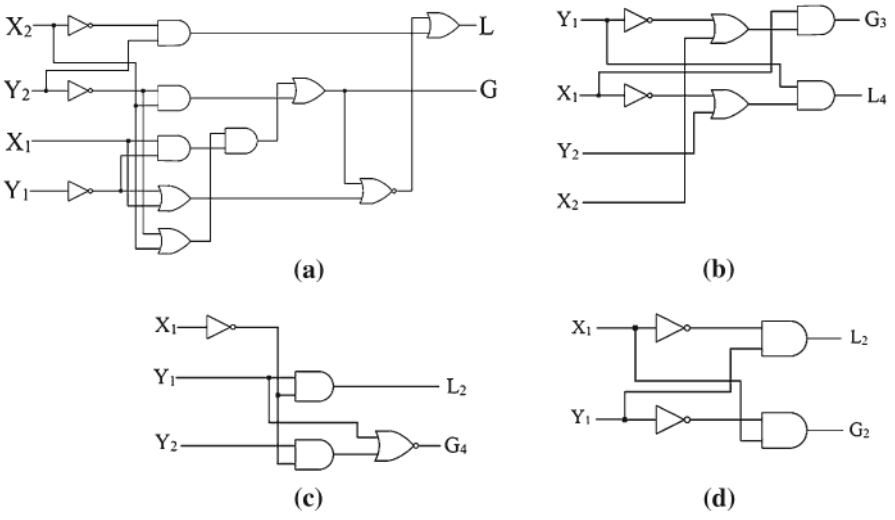


Fig. 12 Logic diagrams of TBC. **a** Precise, **b** T43, **c** T24, and **d** T22

there is 2-bit approximation. Similarly, at the second and third levels, two and three blocks related to the less significant bits are approximated, respectively. At the fourth level, all blocks are designed approximately. This method can also be generalized to the larger magnitude comparators. Clearly, although using higher levels of approximation improves the circuit performance, but it increases the output error rate. Hereafter, we add the number of approximated bits to the end of the TLG. Thus, TLGN denotes an

Table 4 Area of different types of approximate TBC

Approximation type of TBC	Gate count
T31, T41	6
T13, T14, T21	5
T12, T33, T34, T43, T44	4
T23, T24, T32, T42	3
T22	2
Precise	8

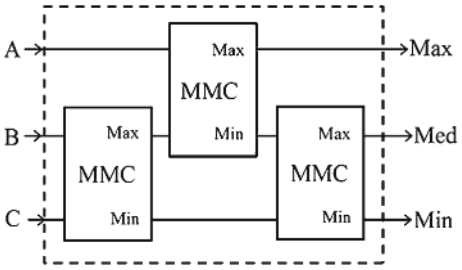


Fig. 13 Ternary data sorter (TDS)

approximate RMC in which the TBC blocks related to N lower significant bits are implemented using TLG approximation.

According to Fig. 12, based on the unit-gate model, a precise TBC has eight gates. We show the area consumption of approximate TBCs in terms of the number of gates in Table 4. Using these approximations reduces the TBC's area from 25 to 75 % of the precise case.

3.3 Optimizing the RMC for Data Sorter

Ternary data sorter (TDS) takes three numbers as inputs and sorts them according to their magnitude. The internal block diagram of TDS is given in Fig. 13. MMC block (Modified MC) sort two numbers. If inputs are same, then both of MMC's outputs will be equal to zero. Therefore, there is no need to design a circuit for equal mode. Figure 14 demonstrates the MMC. SMC is sub-magnitude comparator that determines whether X is greater than Y or not. Due to the value of the output of SMC, we are going to use our proposed comparator to implement a novel median filter to remove salt and pepper noise from gray-level images at high speed and low power cost. The structure of median filter is explained in the next section. In this section, we optimize our proposed comparator to be appropriate for data sorter (DS) that is the main element in the median filter. Multiplexers send the greater and smaller inputs to the relevant outputs. As illustrated in Fig. 15, we remove unnecessary parts of 8-bit RMC and modify it to build an SMC. By using the approximations introduced in Sect. 2, a compact and simplified SMC is obtained. Its area is calculated from (3). It is obvious that using a TBC with fewer gates makes a more compact SMC.

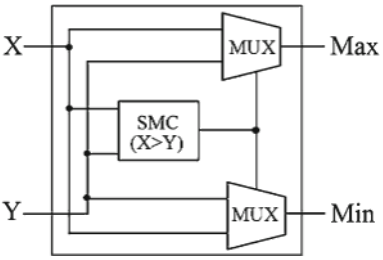


Fig. 14 Modified magnitude comparator (MMC)

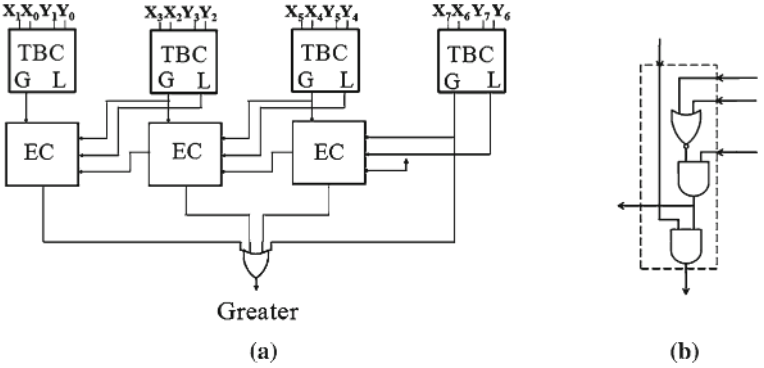


Fig. 15 **a** SMC. **b** Internal structure of EC

However, it should be noted that although employing more approximations leads to a smaller TBC, it may produce such large errors that the result is not acceptable. Thus, it is important to evaluate errors for each approximation.

$$\text{Gate count}_{\text{SMC}} = N \times (\text{Gate count}_{\text{TBC}} + 3) \tag{3}$$

3.4 Synthesis Results of Approximate Comparator

In order to evaluate the speed, area, and power of our approximate comparator, we synthesize it with Nangate 45 nm Open Cell Library using Synopsys Design Compiler. Power consumption is estimated using the power analysis tool of Synopsys with VCD file produced in post-synthesis simulation by applying 100,000 random inputs Figs. 16, 17, 18 and 19 illustrate the synthesis results of our design. LPL notation (lower-part level) indicates the number of bits computed approximately. As LPL increases, the circuit will be much more efficient in terms of area and power consumption. Each increase in the level of approximation leads to about 10 % saving in power and area consumption of the SMC of the SMC. Leakage power is a top concern in deep submicron process technologies (65 nm and below). According to Fig. 18, we can reduce it up to 35 % with respect to the precise one.

We compare our comparators with several state-of-the-art implementations with similar objectives to ours (i.e., high-speed operation and power savings). Simulation results are shown in Table 5. Our comparators save more power and are

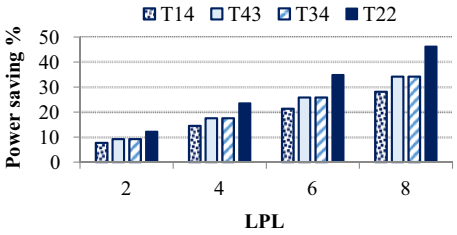


Fig. 16 Percentage of power reduction in various 8-bit approximate SMC with different LPLs. LPL shows the numbers of bits computed approximately

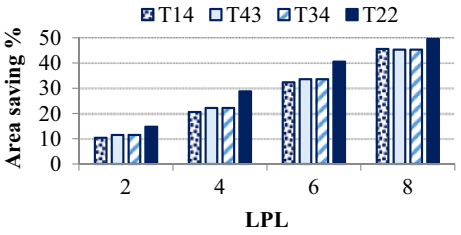


Fig. 17 Percentage of area reduction in various 8-bit approximate SMCs with different LPLs

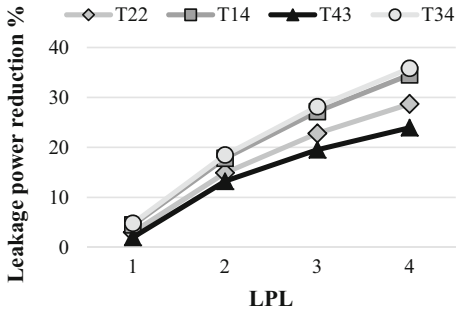


Fig. 18 Percentage of leakage power reduction for different types of 8-bit approximate SMC with different LPLs. LPL shows the numbers of bits computed approximately

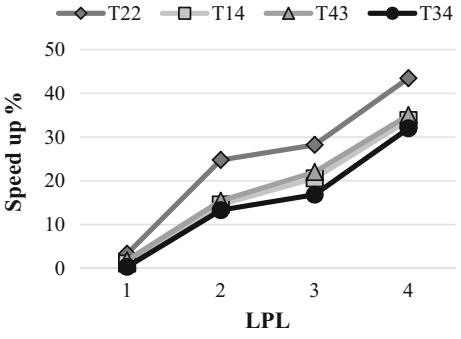


Fig. 19 Percentage of delay reduction in different types of 8-bit approximate SMC with respect to precise one

faster than [1,20,27]. The advantages of approximate comparators are considerable (Table 6).

Table 5 Power-delay product ($\times 10^{-19}$ J) for different kinds of 8-bit approximate SMCs

SMC	LPL = 2	LPL = 4	LPL = 6	LPL = 8
T22	6.66	5.70	5.56	5.23
T14	6.83	6.42	6.09	5.85
T43	6.80	6.37	6.03	5.83
T34	6.90	6.50	6.37	6.02

PDP of the precise one is 6.93×10^{-19} J. LPL shows the numbers of bits computed approximately

Table 6 Comparison on physical properties of several 8-bits comparators

Median filter	[20]	[27]	[1]	Our precise	T22	T14	T43	T34
Power dissipation (μW)	5.72	2.61	2.1	1.96	1.28	1.33	1.31	1.30
Delay (ns)	1.23	0.47	0.38	0.35	0.24	0.25	0.26	0.28

APMFs use 6-bit approximation

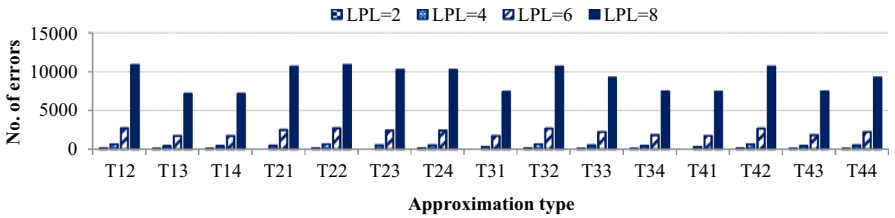


Fig. 20 Number of errors from 65,536 possible cases in different types of approximate SMC

3.5 Error Evaluation of Approximate Comparator

In order to evaluate the errors in the proposed approximate comparators, all possible combinations of two integer numbers in the range of [0–255] are applied to different types of approximate MMCs and their outputs are compared. Then, the mean and maximum error values and the total number of errors are calculated in each case. The results are given in Figs. 20 and 21. As seen, the approximate comparators using the combination of types 1 and 3, 1 and 4 and also 3 and 4 of L and G have the least number of errors. The maximum error value is 85 for all types of approximations. Although these tables provide useful information, but are not sufficient enough. In order to have a proper understanding about each approximation, it is required to know some qualitative information about the error occurrence in it.

Error dispersion is the most important qualitative measure in evaluating approximate comparators. Since the logic functions of approximate MCs are different, each has its own error characteristics.

For this purpose, we have to consider two 8-bit binary numbers as the inputs of MMC and find all possible errors in each approximation type. A series of 8-bit integer pairs between 0 and 255 is applied as inputs to MMC. Then, its output is compared with the correct one.

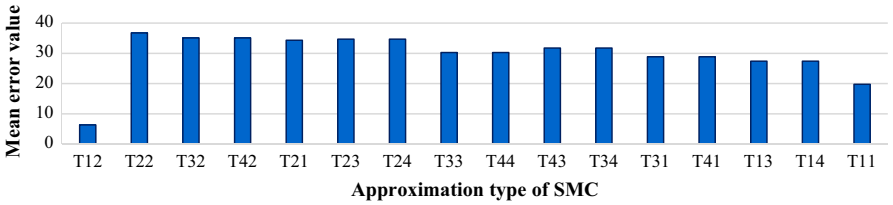


Fig. 21 Mean error values in different types of 8-bit approximate SMCs

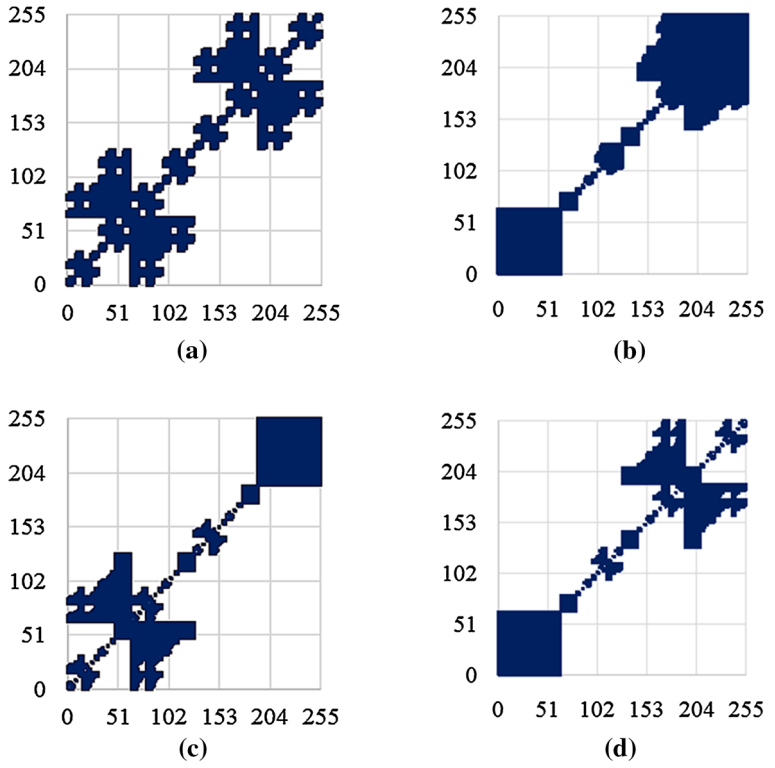


Fig. 22 Error dispersion plots for three types of 8-bit approximate MMCs. **a** T228, **b** T148, **c** T438, and **d** T348

In Fig. 22, we introduce error dispersion plot that represents the error characteristics in a more informative way. For ease of explanation, we divide the plot into four quarters, Q1 to Q4, as shown in Fig. 23.

According to Figs. 22 and 23, the number of errors in Q2 and Q4 is very small and negligible. This is true for all types of approximations. Figure 22 shows that T43 for X and Y greater than 127, Q1, behaves similar to T34 for x and y less than 127, Q3. This is a general rule that the error dispersion for Tnm is symmetric to Tmn with respect to the main diagonal. This can be inferred from the Karnaugh maps for Lm and Gn, shown in Fig. 5. Therefore, if input values are mostly large, then T43 is more appropriate. However, for small input values, it is better to use T14 or T34.

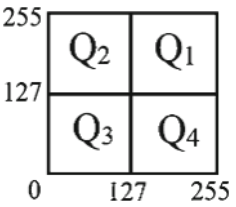


Fig. 23 Partitioning the plot into Q1 to Q4

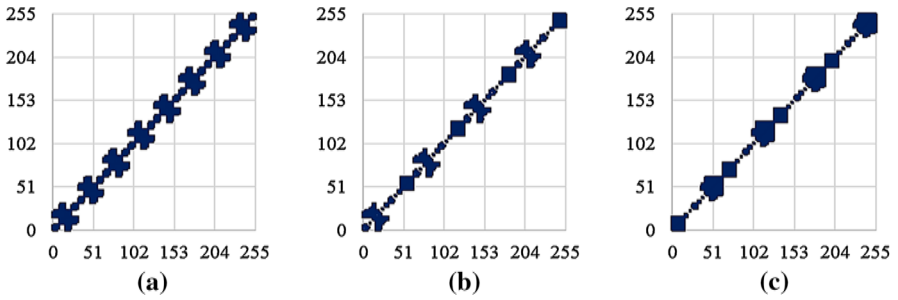


Fig. 24 EDPs for three types of 6-bit approximate MMCs. **a** T226, **b** T436, and **c** T146

Given what has been said, we expect that error dispersion for level 3 of approximation, 6-bit approximations, is much less than that for level four, as Fig. 24 shows. At level 3, the pattern of level 4 is repeated at a lower scale. The reason is that at level 3, the approximation is applied on the six less significant bits and two most significant ones are compared precisely. Thus, if we consider the truth table of level 3, when two most significant bits change, the approximation pattern is repeated from the beginning. Therefore, the period of pattern is 4. This phenomenon is also seen in the lower levels of approximation. It means that for 4-bit and 2-bit approximations, the pattern repetition periods are equal to 16 and 64, respectively. However, it is understood that the patterns of lower levels of approximation are smaller and contain fewer points of the plot than that of larger ones.

4 Median Filter

4.1 Precise Median Filters

Median filter is an effective solution to remove impulsive noise while preserving the image edges. This filter works based on moving an odd size window over an image. The window size of 3×3 is most effective over the vast range of noise densities and gives the best results for most commonly used image sizes. The window moves one by one on all image pixels. The desired pixel is located in the center of the window. At each position, the pixel values in the filtering window are sorted in any order, and the median value of them is picked. Finally, the median value is replaced with the desired pixel in the center of the window. Median filters play an important role in signal processing applications such as video and audio processing. Hence, in many cases, real-time high-speed implementations are required. Meanwhile, low-power design is inevitable for

mobile applications. Median filters can be implemented in either software or hardware. Realizing real-time applications just is possible through hardware implementation.

In general, hardware implementation methods for median filter can be divided into three main categories, which are: systolic array [35], sorting network-based architectures [22,33,56] and radix method or threshold decomposition filter (TDF) [23,29,30,47]. Other methods of designing the median filters include the standard median filter (SMF) histogram based filter (HBF), adaptive median filter (AMF), center weighted filter (CWF), bit parallel filter (BPF) and bit serial filter (BSF).

Conventional implementations of median filters are based on sorting networks that arrange the numbers in ascending or descending order and then select median value. This makes them suitable for pipeline processing. Thus, they have high throughput. In addition, their sampling period can be reduced optionally. The task of sorting networks is just comparing and exchanging the numbers, which is very simple. Classic hardware architecture for median filter is based on a 'Bubble Sorting' model [9,48]. For $(N+1)$ inputs, it requires $N \times (2N+1)$ dual input magnitude comparators. Hardware complexity of bubble sorter, in terms of MC numbers, is $O(N^2)$. Although this approach is highly regular, but requires a large number of comparators. An optimized version of bubble sorter is presented in [40,58].

It should be noted that if only the mean value is desired, there is no need to sort the whole window cells. 'Triple Input Sorter'-based algorithm (TIS) is an optimized version of the bubble sort algorithm for the special case of a 3×3 window size and has less exchange of data [62]. Figure 25 displays the structure of a pipelined TIS-based median filter with 3×3 window. In order to implement a moving window, it is necessary to store rows of image. Line buffers are used to hold the elements of columns or rows (depending on the image scan direction). Their length is determined

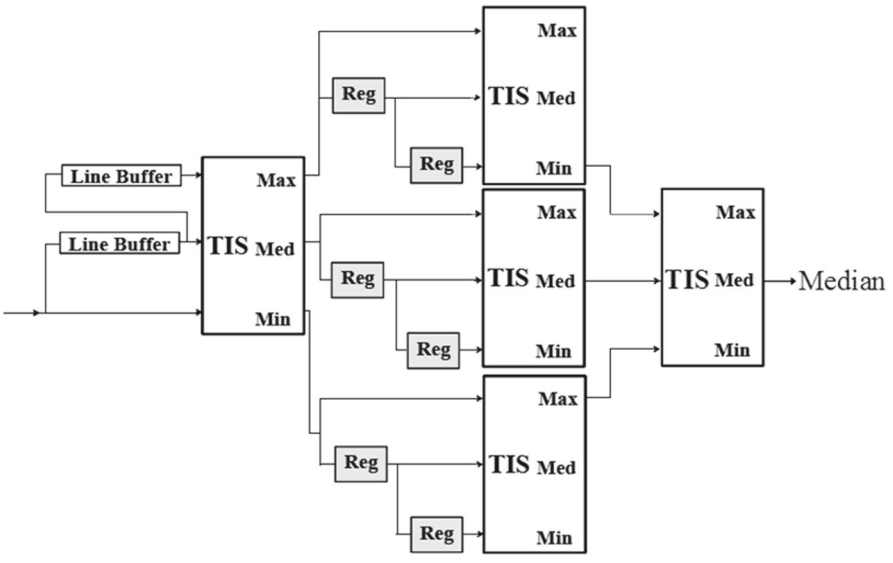


Fig. 25 Pipelined TIS-based sorter implementation

by image length. These buffers simultaneously fed the input values to the median filter. This causes that all pixel values in each window position are processed at the same time [37]. In this case, regardless of the length of the input data, the output pixel is generated in every clock pulse.

4.2 Introducing Approximate Median Filter (APMF)

Computational complexity and delay of the circuit as well as its power consumption and performance depend on its logical function. Therefore, the circuit performance can be improved by discovering a proper logical function. If the complexity of the logic function is less, fewer resources are required to be implemented, thereby reducing switching activity and consequently decreasing the power consumption.

We use our approximate magnitude comparators that were presented in Sect. 3, to implement a low-complexity and low-power median filter. For this purpose, we consider pipelined TIS-based sorter implementation. Employing approximate MMCs in the TIS blocks of median filter leads to erroneous outputs. It is remarkable that median filter is intrinsically error tolerant. According to Fig. 25, since only the median outputs of the last TIS are important, it is likely that the approximation errors affect the other outputs of the TIS blocks. In this case, the median value may be determined correctly. However, in some cases, it is possible that during the sorting, the error occurrence in each TS block compensates by other blocks. Therefore, in total, internal errors might not reduce the quality of final output seriously.

4.3 Synthesis Results of APMFs

To study and analyze the proposed approximate filters in more detail, we consider four types of T22, T43, T14 and T34 as instances. Since the length of line buffers is dependent on the size of the input image and also all window-based spatial filters need such buffers, the synthesis results in this section are without line buffer implementation. We simulate our design using Modelsim and synthesize it with "NanGate 45 nm open cell library" library using Synopsys and its power analysis tool using VCD file that was produced in post-synthesis simulation by applying 100,000 random inputs.

We evaluate the circuit performance of APMFs in terms of dynamic and leakage power consumption, worst-case delay, and power-delay product for all ranges of LPL. The results are given in Figs. 26, 27, 28 and 29 and in Tables 7 and 8. Power consumption of precise filter is equal to 82.96 μW. We can reduce power consumption from 7 to 30 % using 2–6-bit APMFs. As shown in Fig. 26, we can get 10–30 % saving in area consumption. In these cases, the speed of processing is increased from 12 to 18 %. According to Fig. 29, using higher the level of approximation gives better PDP.

We compare our results with those of [4,38], due to their similar objectives. For this purpose, we synthesize them with the technology used in this paper. According to Table 9, our designs have better performance than those of [4,38]. However, as stated before, there is a trade-off between physical properties and the output quality of APMFs. We note that all comparisons are made at the post-synthesis level for all

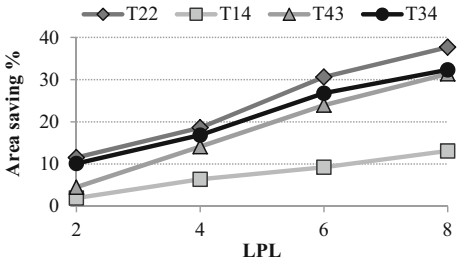


Fig. 26 Area saving for different types of APMFs. LPL shows the numbers of bits computed approximately

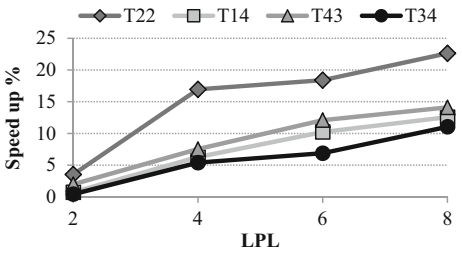


Fig. 27 Percentage of delay reduction in different types of APMFs with respect to precise one. LPL shows the numbers of bits computed approximately

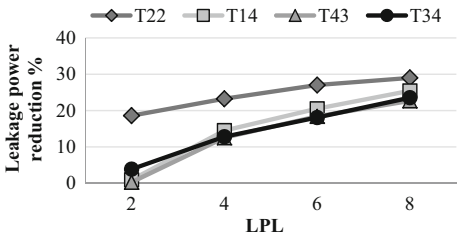


Fig. 28 Percentage of leakage power reduction in different types of APMFs with respect to precise one. LPL shows the numbers of bits computed approximately

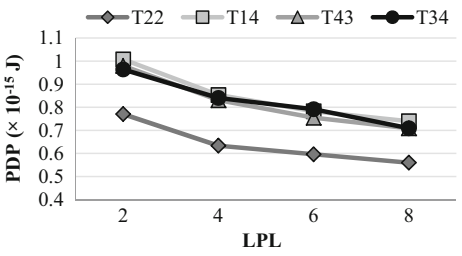


Fig. 29 Power-delay product for different kinds of APMFs. PDP of the precise one is 1.04×10^{-15} J. LPL shows the numbers of bits computed approximately

designs at the block level with a similar loading, as ASIC designers perform for their building blocks, being fully characterized for inclusion in a future chip.

4.4 Error Evaluation of APMFs

In order to examine our design, we add salt and pepper noise to several 512×512 gray-level images. These images are given in Fig. 30. They are used as inputs for different types of APMFs to suppress the noise. In order to evaluate the performance of the

Table 7 Dynamic power consumption (μW) for different kinds of AMPFs

APMF	LPL = 2	LPL = 4	LPL = 6	LPL = 8
T22	63.78	60.92	58.36	57.78
T14	81.06	72.69	69.61	67.54
T43	79.81	71.72	68.57	65.92
T34	77.31	70.93	67.89	63.65

LPL shows the numbers of bits computed approximately. Power consumption of precise filter is equal to 82.96 μW

Table 8 Delay (nS) for different kinds of AMPFs

APMF	LPL = 2	LPL = 4	LPL = 6	LPL = 8
T22	12.08	10.4	10.22	9.69
T14	12.43	11.74	11.24	10.95
T43	12.28	11.58	11.01	10.76
T34	12.47	11.85	11.66	11.14

LPL shows the numbers of bits computed approximately. Delay of precise filter is equal to 12.56 nS

Table 9 Comparison on physical properties

Median filter	[38]	[4]	Our precise	T22	T14	T43	T34
Static (μW)	229.03	254.74	210.72	146.48	176.39	173.28	171.83
Dynamic (μW)	89.47	97.60	82.96	58.36	69.61	68.57	67.89
Delay (ns)	15.07	20.10	12.56	10.22	11.24	11.01	11.66
PDP ($\times 10^{-15}$ J)	1.35	1.96	1.04	0.60	0.78	0.75	0.79

Dynamic power is measured using 100,000 patterns with post-synthesis netlists. APMFs use 6-bit approximation

approximate filters, 8-bit resolution model of them is designed in MATLAB. As the errors of some images cannot be seen in Fig. 30, we show error magnitude in Fig. 31. Based on our experiments, median filters with 2–6-bit approximations give acceptable results with less quality loss so that human eye cannot perceive the difference in their results compared to precise filtering. Thus, we do not show the output images for 2–6 LSBs.

We use a well-known metric of peak signal-to-noise ratio, PSNR, to quantify the output quality. This metric objectively measures the intensity of the error signal. Suppose f and g are the images with dimensions M $\times$ N obtained from the precise and approximate filters. PSNR is calculated from Equation (4). MSE is mean square error and is described in (5), where f_{ij} implies the intensity of the pixel at ith row and jth column of image. B is the maximum number of bits required to represent the intensity values, which is 8 for gray-level images. We use another metric, namely Mean Structural SIMilarity, MSSIM, which is computed using (6) [61]. To calculate SSIM, both the output images of exact and approximate filters are divided into K windows and the

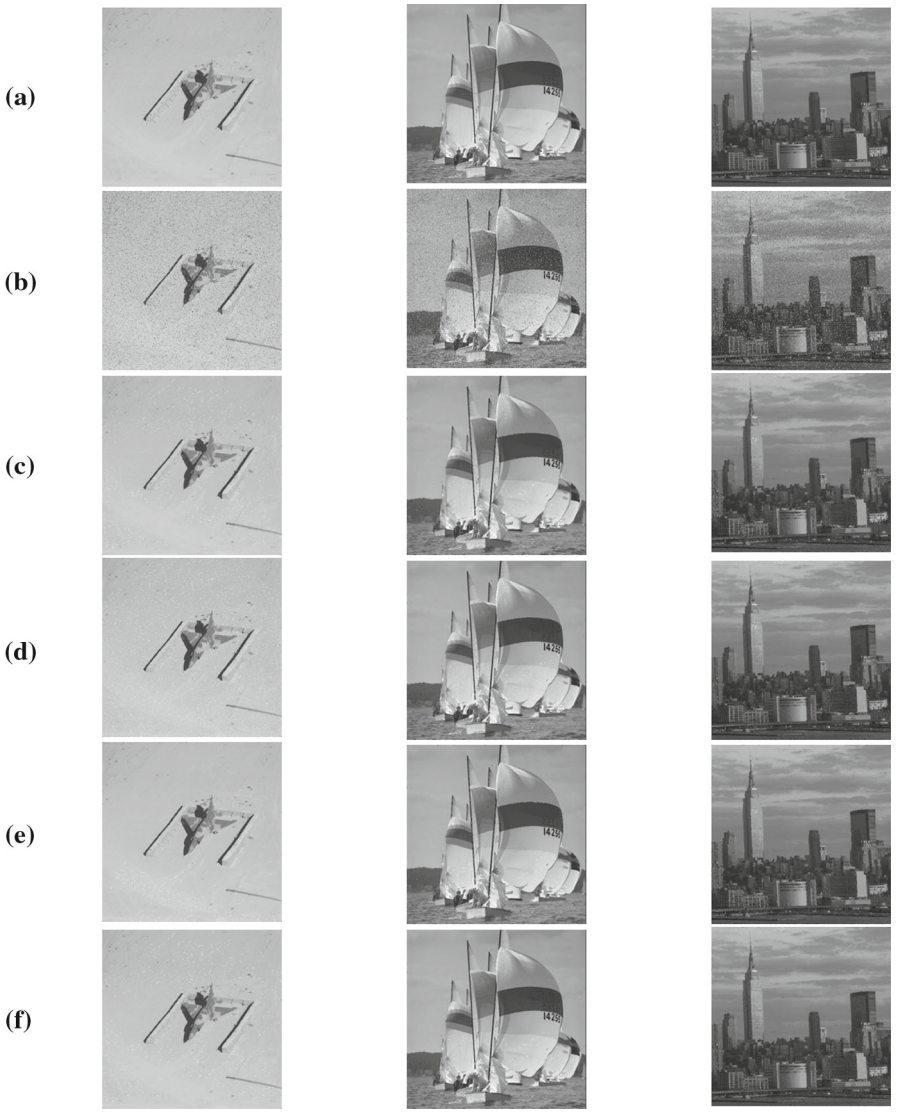


Fig. 30 **a** Original image, **b** noisy image, output images for **c** T22, **d** T14, **e** T43, and **f** T34

structural similarity between them in each window is measured. μ and σ denote the mean and variance of the image intensities, respectively. C1 and C2 are constants. ω is Gaussian weighting function that is normalized to unit sum ($\sum_{i=1}^{N} \omega_i = 1$). MSSIM measures the structural similarity between two images based on human visual perception. On the other words, MSSIM is semi-subjective metric. It gets a value between zero and one. If MSSIM is equal to one, two images are the same.

Figure 32 shows the PSNR of three images illustrated in Fig. 30. PSNR of precise filter for airplane, buildings and boats are 30.64, 36.07 and 31.77 dB, respectively.

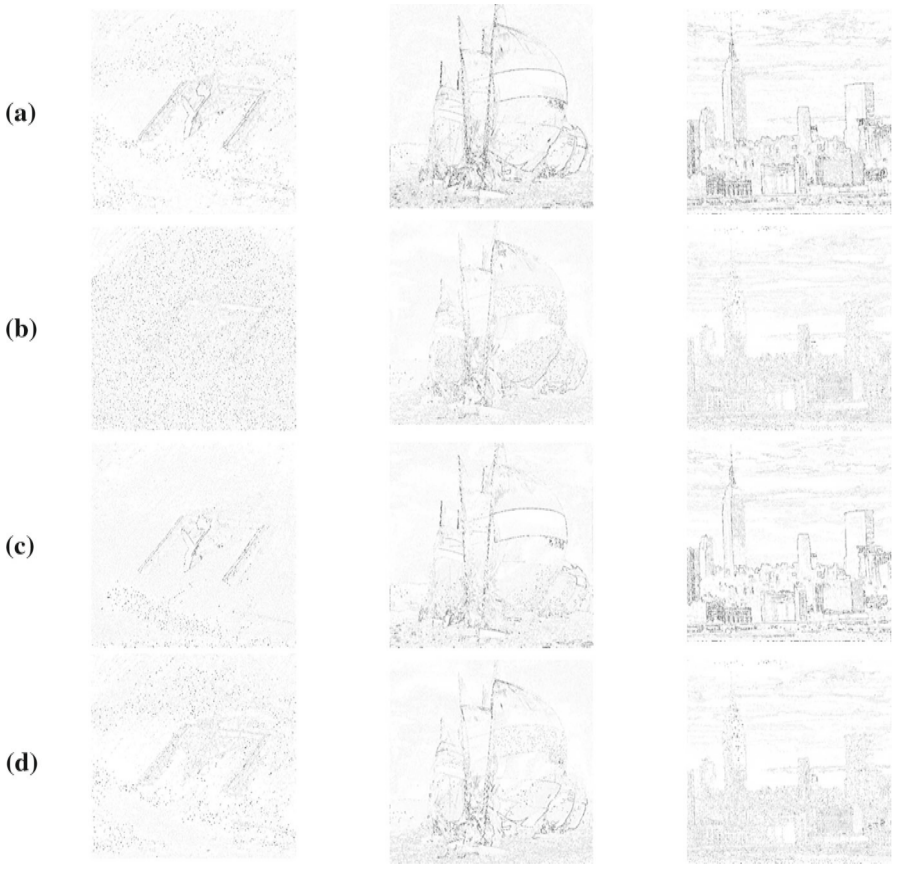


Fig. 31 Absolute error images. Output images for **a** T22, **b** T14, **c** T43 and **d** T34. We note that *darker pixels* in these images represent more error or a worse filter

According to PSNR diagrams, the qualities of output images using APMFs are the same for 2–6 LPLs.

The PSNR values for 2–6-LPL approximations are very close to the PSNR value using the precise filter. Therefore, 2–6 LPL APMFs can be used instead of the precise one and achieve the desired results. The degradation just occurs in 8-LPL approximations. As seen from Figs. 32 and 33, for airplane image, T43 is the best and T14 is the worst, whereas for building, which is darker, T14 is the best and T22 is the worst. Since the performance of these filters is highly dependent on the image type, we analyze the details of their behavior in the following.

According to Fig. 30, T22 results in more errors than other three approximate filters, but performs better in noise removal. The reason must be sought in the EDPs of Fig. 22. As shown in Fig. 34, APMFs exhibit different behaviors in the presence of different kinds of spark noises, i.e., the pixels with intensities of 0 or 255, i.e pepper or salt.

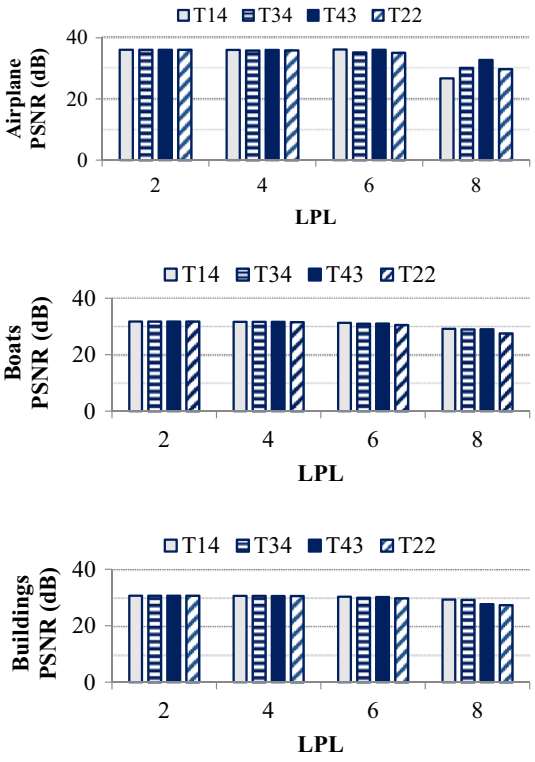


Fig. 32 PSNR for four types of APMFs. LPL shows the numbers of bits computed approximately

$$\mathrm{PSNR}\,(f,g) = 20\log_{10}\left(\frac{2^B-1}{\mathrm{MSE}\,(f,g)}\right) \tag{4}$$

$$\mathrm{MSE}\,(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f_{ij}-g_{ij}\right)^2 \tag{5}$$

$$\mathrm{MSSIM}\,(f,g) = \frac{1}{K}\sum_{i=1}^{K}\mathrm{SSIM}(f,g) \tag{6}$$

$$\mathrm{SSIM}(f,g) = \frac{\left(2\mu_f\mu_g + C_1\right)\left(2\sigma_{fg} + C_2\right)}{\left(\mu_{f+}^2\mu_{f+}^2 + C_2\right)\left(\sigma_{f+}^2\sigma_{f+}^2 + C_2\right)} \tag{7}$$

$$\mu_x = \sum_{i}^{N}\omega_i x_i \tag{8}$$

$$\sigma_x = \left(\sum_{i=1}^{N}\omega_i\left(x_i-\mu_i\right)^2\right)^{\frac{1}{2}} \tag{9}$$

$$\sigma_{xy} = \sum_{i=1}^{N}\omega_i\left(x_i-\mu_x\right)\left(y_i-\mu_y\right) \tag{10}$$

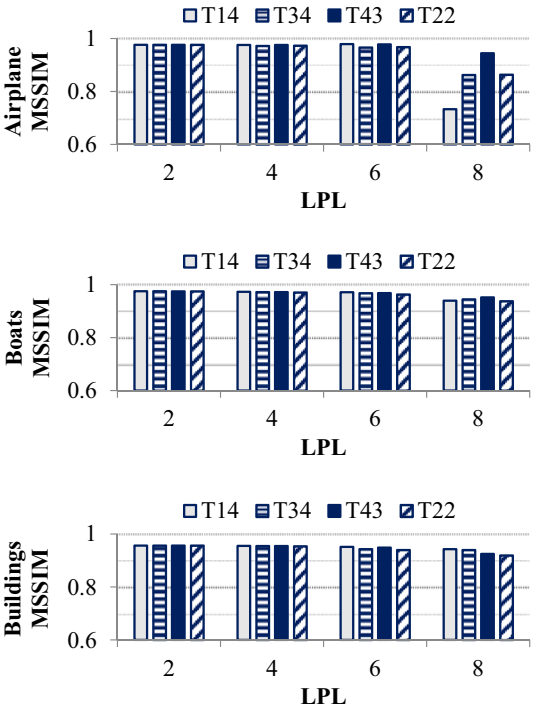


Fig. 33 MSSIM for some types of APMFs. LPL shows the numbers of bits computed approximately

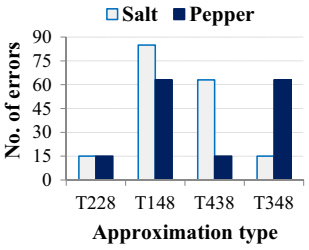


Fig. 34 Number of errors occurred in noisy case. *White* and *black bar* show the remaining salt noisy pixels and pepper noisy pixels after different types of approximate filtering

In the presence of white dots, T22 and T34 perform better, but for black dots, T22 and T43 are the best. However, this is highly dependent on the type of the image, and one must use the information from Figs. 20 and 22 to select the best filter type for the desired application. According to these figures, the number of noisy points that are not removed using T22 is less than that of the other APMFs, whereas the total number of errors for this filter is the most. Thus, although T22 removes salt and pepper noise better, due to the scattered distribution of errors, the output image will be more blurry. So, this filter can be effective for the image with fewer edges and low contrast.

Figures 30 and 31 are consistent with the results shown in Fig. 34. It is observed that T22 suppresses the noise better, and the error is more occurring around edges. It is visible from airplane images that T14 is weaker against pepper noise points. Furthermore, the building images show that T14 and T34 do not perform well in dark

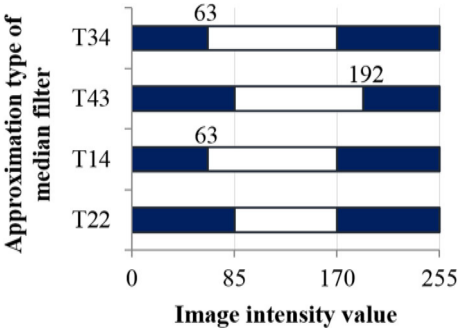


Fig. 35 The range of numbers susceptible to error during salt and pepper noise removal by median filter. The error may occur in any part of the image with intensity values in the color range. *White intensity* ranges are error-free

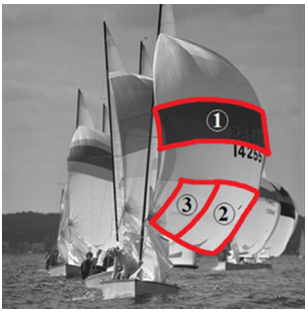


Fig. 36 Selected regions specified in image of boats

regions. It should be noted that when the difference between the brightness of noisy points and that of the image is high, i.e., peppers in a lighter background and salts in a darker background, all types of APMFs will remove the noise completely. This occurs when the brightness values lie in Q2 and Q4 in Fig. 23, and as we illustrated in Sect. 3, these regions are error-free. Note that depending on the image type, one can select appropriate APMF.

From EDPs, we can find the regions that APMFs work accurately. For salt and pepper noise suppression, it is important to know about the regions in which APMFs remove noise precisely. For a better view, we give the regions with high risk of error and also error-free regions in Fig. 35. The approximate filter may not remove pepper noise from any part of the image that its intensity values are placed at the bottom of the colored area in Fig. 35. The same is true for salt spots and the top of the shaded area. Therefore, although the number of erroneous black dots in T14 and T34 is more than that in T43 or T22, but if the image intensity is between 63 and 85, these filters will remove the black dots completely, whereas the outputs of T43 and T22 contain the error.

Similarly, we can judge the filters in the case of white noisy points. This phenomenon can be observed well for airplane image in Figs. 30 and 31. According to Fig. 34, the number of white noisy points that T43 cannot remove is more than that of T34, but since most regions of the image have an intensity between 170 and 190 where T43 works without error, its output image quality is much better than that of T34. As seen in images of boats, all types of APMFs could remove the noise of the sky part, completely.

The reason is that the image intensity range in this region is between 124 and 170, which is error-free for all types of APMFs. The image intensities of the Region (1), as shown in Fig. 36, are between 35 and 55. These values are within the common error range for all APMFs. However, since the errors of T22 or T43 in this range are much less than the other two, they remove almost all noisy points. The pixel values in Region (2) of Fig. 36 are between 192 and 215, which are common error ranges in all of APMFs. Since T22 and T34 have much less error in this range, it is expected that these two filters remove the noise better than others. This is confirmed in Figs. 26 and 30. As the intensity value of the Region (3) of Fig. 36 is in the range of 172–190 in which T43 works error-free, the corresponding region in the output image of this filter is without error. However, this is a common region of error between three other APMFs.

According to Fig. 34, T14 has much more error than the others. Therefore, the output quality of this filter is worse than T22 and T34. The remained noisy points of T22 and T43 in Region (3) are the same.

5 Conclusion

In this paper, we proposed a method to design approximate 2-bit magnitude comparators that can be more effective in terms of power, area and speed than conventional precise one. Large comparators are implemented easily using 2-bit comparators. Simulations show that power and area consumption of 8-bit approximate comparators are decreased from 7 to 46 and 10 to 50 % respectively, with respect to the precise one, when using different approximation structures. Although there is a trade-off between output quality and power, our design is useful for error resilient applications such as image/video processing.

We used our comparators to design image median filters to remove salt and pepper noise at higher speed and lower power. The error behaviors of approximate comparators and median filter were discussed. Simulation results showed that the output quality of approximate median filters with 2–6 LPLs is very similar to that of precise filter, and the degradation is not visible. Their implementation can provide savings up to 30 % in terms of power and area consumption, and we can get 18 % speed up with respect to a precise filter.

We evaluated our designs with some state-of-the-art implementations (three comparators and two filters) with similar objectives to ours. Simulation results showed that our designs are faster and have better performance in terms of area and power consumption.

References

1. S. Abdel-Hafeez, A. Gordon-Ross, B. Parhami, Scalable digital CMOS comparator using a parallel prefix tree. IEEE Trans. Very Large Scale Integr. Syst. **21**(11), 1989–1998 (2013)
2. D. Alnajiar, Y. Ko, T. Imagawa, H. Konoura, M. Hiromoto, Y. Mitsuyama, M. Hashimoto, H. Ochi, T. Onoye, Coarse-grained dynamically reconfigurable architecture with flexible reliability, in *IEEE International Conference on Field Programmable Logic and Applications* (FPL'09), pp. 186–192 (2009)

3. L. Budin, D. Jakobović, M. Golub, Genetic algorithms in real-time imprecise computing. J. Comput. Inf. Technol. **8**(3), 249–257 (2004)
4. C. Chakrabarti, Sorting network based architectures for median filters. IEEE Trans. Circuit Syst. II Analog Digit. Signal Process. **40**(11), 723–727 (1993)
5. L.N. Chakrapani, B.E.S. Akgul, S. Cheemalavagu, P. Korkmaz, K.V. Palem, B. Seshasayee, Ultra-efficient (embedded) SOC architectures based on probabilistic CMOS (PCMOS) technology, in *Design, Automation and Test in Europe* (DATE '06), 6–10 March, pp. 1–6 (2006)
6. S. Cheemalavagu, P. Korkmaz, K.V. Palem, B.E.S. Akgul, L.N. Chakrapani, A probabilistic CMOS switch and its realization by exploiting noise, in *IFIP International Conference on Very Large Scale Integration* (VLSI-SoC), pp. 535–541 (2005)
7. X. Cheng, M.S. Hsiao, Region-level approximate computation reuse for power reduction in multimedia applications, in *ACM International Symposium on Low Power Electronics and Design* (ISLPED '05), pp. 119–122 (2005)
8. H. Chishiro, A. Takeda, K. Funaoka, N. Yamasaki, Semi-fixed-priority scheduling: new priority assignment policy for practical imprecise computation, in *IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications* (RTCSA), 23–25 August, pp. 339–348 (2010)
9. L. Christopher, W. Mayweather III, S. Perlman, A VLSI median filter for impulse noise elimination in composite or component TV signals. IEEE Trans. Consum. Electron. **34**(1), 262–267 (1988)
10. J. Epps, Fast approximate computation of non-uniform DFTs for biological sequence analysis. Electron. Lett. **45**(8), 429–430 (2009)
11. D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N.S. Kim, K. Flautner, Razor: circuit-level correction of timing errors for low-power operation. IEEE Micro **24**(6), 10–20 (2004)
12. F. Firouzi, A. Azarpeyvand, M.E. Salehi, S.M. Fakhraie, Adaptive fault-tolerant DVFS with dynamic online AVF prediction. Microelectron. Reliab. **52**(6), 1197–1208 (2012)
13. J. George, B. Marr, B. Akgul, K. Palem, Probabilistic arithmetic and energy efficient embedded signal processing, in *IEEE/ACM International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*, pp. 158–168 (2006)
14. R. Gonzalez, P. Wintz, *Digital Image Processing* (Addison-Waley, Reading, 1987)
15. A. Gupta, S. Mandavalli, V.J. Mooney, L. Keck-Voon, A. Basu, H. Johan, B. Tandianus, Low power probabilistic floating point multiplier design, in *IEEE Computer Society Annual Symposium on VLSI* (ISVLSI'11), 4–6 July, pp. 182–187 (2011)
16. V. Gupta, D. Mohapatra, S.P. Park, A. Raghunathan, K. Roy, IMPACT: IMPrecise adders for low-power approximate computing, in *International Symposium on Low Power Electronics and Design* (ISLPED'11), 1–3 August, pp. 409–414 (2011)
17. V. Gupta, D. Mohapatra, A. Raghunathan, K. Roy, Low-power digital signal processing using approximate adders. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(1), 124–137 (2013)
18. J. Han, M. Orshansky, Approximate computing: an emerging paradigm for energy-efficient design, in *18th IEEE EuropeanTest Symposium* (ETS), pp. 1–6 (2013)
19. B. Heck Ferri, N. Perrin, A. Ferri, Adaptive length IIR filters implemented with imprecise computing, in *American Control Conference* (ACC '07), 9–13 July, pp. 2887–2892 (2007)
20. L. Hing-Mo, T. Chi-Ying, A MUX-based high-performance single-cycle CMOS comparator. IEEE Trans. Circuits Syst. II Express Briefs **54**(7), 591–595 (2007). doi:10.1109/TCSII.2007.899856
21. H. Jiawei, J. Lach, Exploring the fidelity-efficiency design space using imprecise arithmetic, in *16th Asia and South Pacific Design Automation Conference* (ASP-DAC), 25–28 January, pp. 579–584 (2011)
22. V. Jimenez-Fernandez, D. Martinez-Navarrete, C. Ventura-Arizmendi, Z. Hernandez-Paxtian, J. Ramirez-Rodriguez, Digital circuit architecture for a median filter of grayscale images based on sorting network. Int. J. Circuits Syst. Signal Process. **5**(3), 297–304 (2011)
23. J. Juhola, P. Haavisto, O. Vainio, T. Raita-aho, Y. Neuvo, On VLSI implementation of median based field rate up-conversion, in *IEEE International Symposium on Circuits and Systems*, 1–3 May, pp. 3042–3045 (1990)
24. Z. Kedem, V. Mooney, K. Krishna, Muntimadugu, A. Devarasetty, P.D. Parasuramuni, optimizing energy to minimize errors in dataflow graphs using approximate adders. *Paper Presented at the International 11th Conference on Compilers, Architectures, and Synthesis for Embedded Systems* (CASES), Arizona, USA (2010)
25. Z.M. Kedem, V.J. Mooney, K.K. Muntimadugu, K.V. Palem, an approach to energy-error tradeoffs in approximate ripple carry adders, in *International Symposium on Low Power Electronics and Design (ISLPED'11)*, 1–3 August, pp. 211–216 (2011)

26. Z.M. Kedem, V.J. Mooney, K.K. Muntimadugu, K.V. Palem, An approach to energy-error tradeoffs in approximate ripple carry adders, in *2011 International Symposium on Low Power Electronics and Design (ISLPED)* 1–3 August, pp. 211–216 (2011)
27. M. Kim, J.-Y. Kim, H.-J. Yoo, A 1.55 ns 0.015 mm 2 64-bit quad number comparator, in *IEEE International Symposium on VLSI Design, Automation and Test (VLSI-DAT'09)*, pp. 283–286 (2009)
28. T. Le, M. Glesner, A flexible and approximate computing approach for time–frequency distributions. IEEE Trans. Signal Process. **48**(4), 1193–1196 (2000)
29. C.L. Lee, C.-W. Jen, Bit-sliced median filter design based on majority gate. IEEE Proc. G Circuits Devices Syst. **139**(1), 63–71 (1992)
30. T.-W. Lee, J.-H. Lee, S.-B. Cho, FPGA Implementation of a 3/spl times/3 window median filter based on a new efficient bit-serial sorting algorithm. in *IEEE 7th Korea–Russia International Symposium on Science and Technology*, pp. 237–242 (2003)
31. K. Lengwehasatit, A. Ortega, DCT computation based on variable complexity fast approximations. in *International Conference on Image Processing (ICIP 98)*, October 4–7, pp. 95–99 (1998)
32. T. Li-Ping, C. Jia-Ming, H. Wei-Fen, S. Wei-Kuan, An imprecise computation model in reducing power consumption of flash memory for portable devices, in *28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, 28–30 September, pp. 62–63 (2004)
33. C.C. Lin, C.J. Kuo, Two-dimensional rank-order filter by using max-min sorting network. IEEE Trans. Circuits Syst. Video Technol. **8**(8), 941–946 (1998)
34. J.W.S. Liu, S. Wei-Kuan, L. Kwei-Jay, R. Bettati, C. Jen-Yao, Imprecise computations. IEEE Spec. Issue Real-Time Syst. **82**(1), 83–94 (1994)
35. C. Long-Wen, J.H. Lin, A bit-level systolic array for median filter. IEEE Trans. Signal Process. **40**(8), 2079–2083 (1992)
36. H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans. Circuits Syst. **57**(4), 850–862 (2010)
37. H.R. Mahdiani, S.M. Fakhraie, C. Lucas, Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors. IEEE Trans. Neural Netw. Learn. Syst. **23**(8), 1215–1228 (2012)
38. T. Matsubara, V.G. Moshnyaga, K. Hashimoto, A low-complexity and low power design of 2D-median filter. ECTI Trans. Comput. Eng. Comput. Inf. Technol. **5**(2), 1–9 (2011)
39. D. Mohapatra, V.K. Chippa, A. Raghunathan, K. Roy, Design of voltage-scalable meta-functions for approximate computing, in *Design, Automation & Test in Europe Conference and Exhibition (DATE)*, 14–18 March, pp. 1–6 (2011)
40. B. Morcego, J. Frau, A. Català, Suavizado de Imágenes en Tiempo Real mediante Filtrado por Mediana Utilizando Arrays Sistólicos, in *VII Conference on Design of Circuits and Integrated Systems (DCIS)*, pp. 545–546 (1992)
41. V.G. Moshnyaga, K. Hashimoto, An efficient implementation of 1-D median filter, in *IEEE 52nd International Midwest Symposium on Circuits and Systems (MWSCAS'09)*, pp. 451–454 (2009)
42. Z. Ning, G. Wang Ling, Y. Kiat Seng, An enhanced low-power high-speed adder for error-tolerant application, in *12th International Symposium on Integrated Circuits (ISIC '09)*, 14–16 December, pp. 69–72 (2009)
43. Z. Ning, G. Wang Ling, W. Gang, Y. Kiat Seng, Enhanced low-power high-speed adder for error-tolerant application, in *International SoC Design Conference (ISOCC'10)*, 22–23 November, pp. 323–327 (2010)
44. Z. Ning, G. Wang Ling, Z. Weija, Y. Kiat Seng, K. Zhi Hui, Design of low-power high-speed truncation-error-tolerant added and its application in digital signal processing. IEEE Trans. Very Large Scale Integr. VLSI Syst. **18**(8), 1225–1229 (2010)
45. Z. Ning, G. Wang Ling, Y. Kiat Seng, Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications, in *International SoC Design Conference (ISOCC'11)*, 17–18 November, pp. 393–396 (2011)
46. S. Nocco, S. Quer, A probabilistic and approximated approach to circuit-based formal verification. J. Satisfiabil. Boolean Model. Comput. **5**, 111–132 (2008)
47. J. Nousiainen, J. Isoaho, O. Vainio, Fast implementation of stack filters with VHDL-based synthesis and FPGAs, in *IEEE Winter Workshop on Nonlinear Digital Signal Processing*, pp. 5.2–5. 6 (1993)
48. K. Oflazer, Design and implementation of a single-chip 1-D median filter. IEEE Trans. Acoust. Speech Signal Process. **31**(5), 1164–1168 (1983)

49. K. Oflazer, Design and implementation of a single-chip 1-D median filter. IEEE Trans. Acoust. Speech Signal Process. **31**(5), 1164–1168 (1983). doi:10.1109/tassp.1983.1164203
50. Y. Pang, K. Radecka, An efficient algorithm of performing range analysis for fixed-point arithmetic circuits based on SAT checking, in *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, 15–18 May, pp. 1736–1739 (2011)
51. Y. Pang, K. Radecka, An efficient algorithm of performing range analysis for fixed-point arithmetic circuits based on SAT checking, in *IEEE International Symposium on Circuits and Systems (ISCAS'11)*, 15–18 May, pp. 1736–1739 (2011)
52. J.C. Russ, *The Image Processing Handbook* (CRC Press, Boca Raton, 2010)
53. L. Shih-Lien, Speeding up processing with approximation circuits. Computer **37**(3), 67–73 (2004)
54. J.Y.F. Tong, D. Nagle, R.A. Rutenbar, Reducing power by optimizing the necessary precision/range of floating-point arithmetic. IEEE Trans. Very Large Scale Integr. VLSI Syst. **8**(3), 273–286 (2000)
55. G.V. Varatkar, N.R. Shanbhag, Energy-efficient motion estimation using error-tolerance, in *International Symposium on Low Power Electronics and Design (ISLPED'06)*, 4–6 October, pp. 113–118 (2006)
56. K. Vasanth, S.N. Raj, S. Karthik, P.P. Mol, Fpga implementation of optimized sorting network algorithm for median filters, in *International Conference on Emerging Trends in Robotics and Communication Technologies (INTERACT)*, 3–5 December, pp. 224–229 (2010)
57. K. Vasanth, V.J.S. Kumar, A. Yogalakshmi, A. Sivasangari, A simple decision based neighborhood filter for the removal of salt and pepper noise. In *International Conference on Emerging Trends in Computer Science and Information Technology (ICETCSIT), India*, 30–31 March (2013)
58. M.A. Vega-Rodríguez, J.M. Sánchez-Pérez, J.A. Gómez-Pulido, An FPGA-based implementation for median filter meeting the real-time requirements of automated visual inspection systems, in *10th Mediterranean Conference on Control and Automation* (2002)
59. A.K. Verma, P. Brisk, P. Ienne, Variable latency speculative addition: a new paradigm for arithmetic circuit design, in *Design, Automation and Test in Europe (DATE '08)*, 10–14 March, pp. 1250–1255 (2008)
60. J. Von Neumann, Probabilistic logics and the synthesis of reliable organisms from unreliable components. Autom. Stud. **34**, 43–98 (1956)
61. Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)
62. K.-C. Wu, D. Marculescu, Power-planning-aware soft error hardening via selective voltage assignment. IEEE Trans. Very Large Scale Integr. VLSI Syst. **22**(1), 136–145 (2014)
63. H. Yoshimoto, D. Arita, R. Taniguchi, Real-time communication for distributed vision processing based on imprecise computation model, in *International Parallel and Distributed Processing Symposium, IEEE Computer Society*, pp. 128–133 (2002)