2024 Volume 2, Issue 2
Signal and Information Processing
ISSN: 2630-4805

Research Progress and Application of Approximate Computing Technology in Digital Signal Processing

Xu Wang[1,2], Ke Chen[1,2], Chenggang Yan[1,2], Chenghua Wang[1,2], Weiqiang Liu[1,2]*

[1]School of Integrated Circuits, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, Jiangsu Province, China

[2]Key Laboratory of Aerospace Integrated Circuits and Microsystems, Ministry of Industry and Information Technology, Nanjing 211106, Jiangsu Province, China

***Corresponding author:** Weiqiang Liu, liuweiqiang@nuaa.edu.cn

Abstract:

Approximate computing technology has attracted significant attention in the field of signal processing. Complex algorithms and massive data volumes have limited processing speeds and increased hardware consumption in various applications. However, due to the redundancy of signals, precise results are not always necessary, and acceptable results for users are often sufficient. Therefore, the adoption of approximate computing technology can effectively reduce computational load, improve computational efficiency, and enhance system performance. This paper delves into different design levels of approximate computing technology. Firstly, it introduces the characteristics of signal processing applications. Then, it reviews recent research progress in approximate computing technology at both the algorithmic and circuit levels. Additionally, it explores approximate computing solutions in signal processing areas such as communications, video imaging, and radar. Finally, the paper discusses and provides an outlook on the future development directions in this field, offering insights to promote the application of approximate computing technology in signal processing.

Keywords:

Approximate computing
Emerging computing paradigms
Digital signal processing
Research progress

Online publication: December 16, 2024

1. Introduction

Compared to global energy production, the growing demand for computational energy is facing new risks. Currently, the amount of information bits processed and the number of computations performed each year are continuously increasing. It is predicted that by 2050, the number of bits processed by global computing systems is expected to be between 1042 and 1046. The total energy consumption of general-purpose computing continues to grow exponentially, doubling every three years, while global energy production increases linearly at a rate of only about 2% per year. The rise in global computing energy is driven by the growing demand for computation. Although the chip-level energy per bit conversion of computing processor units (such as CPUs, GPUs, FPGAs) has been decreasing over the past 40 years (as indicated by Moore's Law), Moore's Law is currently slowing down as device scaling approaches physical limits. With the continuous growth in energy demand for computing, it is imperative to explore and adopt new computing paradigms that significantly improve energy efficiency [1].

As a potential technology to address the power consumption dilemma, approximate computing has broad application prospects. It replaces traditional computing in an imprecise manner, reducing system power consumption and improving computational performance by sacrificing some computational accuracy, as shown in **Figure 1**. Approximate computing introduces computational error as a new dimension beyond the traditional design dimensions of performance and power consumption in circuit systems. By balancing performance, power consumption, and error in the design space, it achieves a new optimal tradeoff point, providing researchers with a novel design approach [2].

Digital signal processing involves sampling, quantizing, and processing signals in a digital manner, enabling more accurate, flexible, and reliable signal processing through digital algorithms. This offers tremendous opportunities for applications in various fields such as communications, image processing, audio processing, and biomedical engineering. Due to the fault-tolerant characteristics of signal processing applications, it is often unnecessary to pursue absolute or uniquely precise results in practical applications. For example, image processing, as an application related to human perception, can tolerate certain errors in its computations, allowing for a greater degree of fault tolerance in its final results, similar to the imperfect perception of humans themselves. Approximate computing, as a high-performance new computing paradigm, can achieve efficient signal processing with the goal of reducing energy consumption and increasing computational speed. This article combines the latest research on approximate computing technology in the field of digital signal

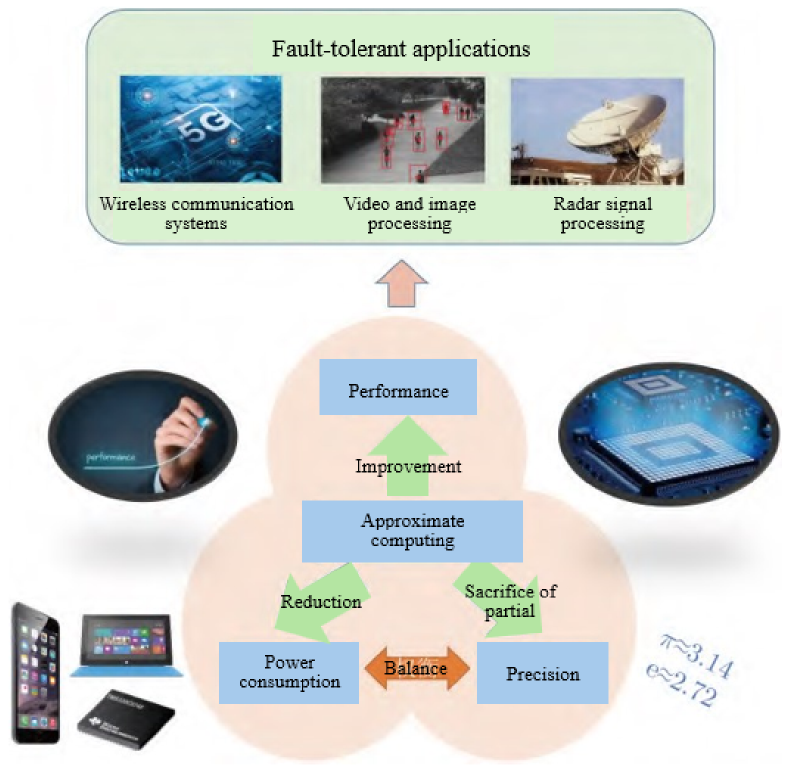

Figure 1. Approximate computing technology and its applications

processing from both academia and industry, introducing and discussing the development, current research status, and future trends of approximate computing chips from the circuit level, algorithm level, and application level.

2. Research progress of approximate computing in signal processing at the circuit level

At the circuit level, approximate computing methods primarily include CMOS technology based on adjusting input voltage and logical approximate computing based on arithmetic operation units. Probabilistic CMOS technology employs voltage over-scaling (VOS) [3] to reduce energy consumption and critical path delay. This is achieved by maintaining the supply voltage for high-bit circuits while appropriately reducing the supply voltage for low-bit circuits. This approach does not require modifying the original circuit structure and is simple to implement. However, VOS technology may introduce uncontrollable errors, posing significant challenges for subsequent applications. Currently, most hardware-level approximate computing mainly relies on approximate simplified designs of arithmetic operation units and logical function modules. Numerous studies have been conducted domestically and internationally, proposing methods such as approximate adders, approximate multipliers, approximate dividers, and approximate multiply-accumulate units to achieve logical approximation at the circuit level and simplify logical output by reducing the number of gate circuits.

2.1. Approximate adder

The approximate adder, initially applied to asynchronous adders, first appeared in 1996. Nowick [4] significantly reduced the delay of asynchronous adders by introducing an approximate inference adder, improving performance by over 30%. In 2004, Lu [5], a researcher at Intel, proposed the first synchronous speculative approximate adder. By approximating precise logical functions with coarse-grained computations, it effectively increased the clock frequency of microprocessors. Subsequently, researchers designed a series of speculative approximate adders. Studies have found that in practical scenarios, for randomly distributed operand inputs, the carry propagation length of the adder is much shorter than the length of the full carry chain. Therefore, faster and higher-performance adders can be obtained by shortening the carry chain. This includes non-segmented speculative approximate adders and segmented speculative approximate adders. The detailed classification of adder approximation methods is shown in **Table 1**. For non-segmented speculative approximate adders, Esposito *et al.* [6] proposed a new variable-latency speculative adder based on the Han-Carlson parallel prefix topology and presented a new error detection network. Compared to previous methods, this approach reduces the error probability. For segmented speculative approximate adders, Seok *et al.* [7] introduced a new approximate adder method that only uses single input pairs to approximate logical gates. The average error distance and average relative error distance of this adder are significantly better than other approximate adders considered in the literature. Additionally, transistor-level approximate full adders significantly reduce power consumption by decreasing the number of transistors and basic gates. Yan *et al.* [10] presented four low-cost approximate full adders. The proposed and existing approximate full adders are classified into two categories based on error distance. Simulation results demonstrate that compared to existing approximate full adders, both groups of approximate adders achieve significant reductions in power-area-delay

Table 1. Approximation techniques for adders

Approximation method	Related work	Overview
Non-segmented speculative approximation	Literature [6]	Faster and higher-performance adders are obtained by shortening the carry chain.
Segmented speculative approximation	Literature [7–9]	
Transistor-level approximation	Literature [10–12]	Significantly reduce power consumption by reducing the number of transistors and basic gates.

product, power consumption, area, and delay.

2.2. Approximate multiplier

Compared to adders, the circuit structure of multipliers is more complex, and the design difficulty increases accordingly. Multipliers can be divided into fixed-point multipliers and floating-point multipliers. Typically, the approximate design of fixed-point multipliers does not directly start from the transistor level, but rather from the components and algorithmic principles of the multiplier, namely operands, partial product generation, partial product reduction, and final summation. The approximation of operands originates from Mitchell's Logarithmic Multiplier (LM), which converts multiplication into addition in the logarithmic domain. This type of multiplier has very low power consumption [13]. Detailed classifications of approximation methods for fixed-point multipliers are shown in **Table 2**. However, due to the significant precision loss that often occurs when implementing logarithms in compact circuits, this method can only be used in applications with very high error tolerance. A common scheme for approximating the partial product matrix is truncation, and multipliers designed using this method are called truncated multipliers. Nunziata *et al.* [15] investigated an approximate recursive multiplier based on a novel 4×4 multiplier block. Through carry truncation and error compensation, three approximate 4×4 multipliers with different trade-offs between error and precision are designed. These basic blocks are then used to design an 8×8 approximate multiplier. The proposed circuit is implemented in 14 nm FinFET technology and achieves improved performance compared to state-of-the-art circuits. The approximation scheme for partial product generation mainly refers to the Booth algorithm. As the most commonly used signed number algorithm in multiplication, the Booth algorithm has been widely applied due to its ability to effectively reduce the number of partial products. The structure of a multiplier includes a partial product reduction tree, which consists of a large number of adders and occupies more than 50% of the entire multiplier's area. Therefore, it is also necessary to perform an approximate design on the compressors in the multiplier. Zhang *et al.* [20] proposed a novel 4-2 approximate compressor that complements other compressors studied earlier and constructed a hybrid multiplier based on compressors, constant approximation, and error-correcting AND gates. Compared to exact multipliers, the proposed hybrid approximate multiplier achieves excellent trade-offs between precision and performance, with a 66% reduction in power-delay-area product.

Compared to fixed-point numbers, floating-point numbers have the advantage of a wider range of data representation capabilities. However, floating-point operations, especially floating-point multiplication, require a significant amount of hardware resources, making research on approximate floating-point computation particularly urgent. The earliest approximate floating-point multiplier can be traced back to 2000, proposed by Tong *et al.* [22]. They approximate the mantissa multiplier by truncating the mantissa operands, effectively reducing the energy consumption of the mantissa multiplier. However, the error caused by truncating the mantissa operands grows exponentially, while the reduction in power consumption

Table 2. Approximation techniques for fixed-point multipliers

Approximation method	Related work	Overview
Operand approximation	Literature [14]	Extremely low power consumption by converting binary multiplication to addition in logarithmic domain.
Array approximation	Literature [15]	Adjust the output bit width and reduce the power consumption and area of the multiplier by directly discarding some low effective bits of the partial product matrix.
Partial product approximation	Literature [16–18]	Optimize the Booth encoding results using Karnaugh maps to simplify the partial product expression of the Booth algorithm.
Compressor approximation	Literature [19–21]	Significant reductions in power consumption, delay, and transistor count are achieved by breaking the carry chain between the same stages of the compressor.

is linear [23]. To address this issue, Zhang *et al.* [24] proposed a method based on logarithmic conversion. By utilizing the characteristics of logarithmic operations, it converts mantissa multiplication into addition, thereby reducing the demand for hardware resources. This method partially solves the error problem caused by truncating mantissa operands and achieves certain improvements in energy consumption. Besides the logarithmic conversion method, Yin *et al.* [25] proposed another adjustable-precision approximate floating-point multiplier. This multiplier combines an approximate mantissa multiplier with a rounding unit. It approximates mantissa multiplication using a simpler circuit structure, reducing hardware resource consumption. Simultaneously, it rounds the multiplication results to meet specific precision requirements. This approximate floating-point multiplier not only reduces energy consumption to a certain extent but also features adjustable precision, allowing users to perform flexible precision control based on specific needs.

2.3. Approximate dividers

Compared to approximate multipliers, research on approximate dividers started relatively late. In the early 1960s, Mitchell proposed an approximate logarithmic divider that converts binary operands into logarithmic operands, introduces errors, and transforms division operations into subtraction [13]. This operation significantly reduces design complexity and improves performance by sacrificing precision. However, logarithmic dividers introduce large errors, making them unsuitable for applications requiring high precision. Therefore, researchers worldwide have proposed approximate design methods specifically for dividers, mainly including array approximation, operand approximation, and a detailed classification of hybrid array-operand approximation methods for dividers, as shown in **Table 3**.

For array approximation methods, Savio *et al.* [26] presented multiple novel approximate subtractors and utilized them to design a restored array divider. Compared to existing designs, the proposed approximate divider offers significant advantages in area, complexity, and power consumption. Regarding operand approximation methods, Wu *et al.* [28] introduced an energy-efficient approximate divider based on logarithmic transformation and piecewise constant approximation. In this design, the range of conversion between binary and logarithm is extended from [0,1] to [-0.5,1], and a heuristic search algorithm is devised to find the most accurate set of constants to approximate the reciprocal of the divisor by minimizing statistical errors. This design achieves higher output precision compared to the most advanced approximate dividers. For hybrid array-operand approximation methods, Liu *et al.* [30] proposed an approximate hybrid divider. Here, an accurate restored divider unit is used to generate the most significant bits of the quotient for high precision, while other quotient bits are generated using a logarithmic divider to reduce power consumption, area, and delay.

2.4. Approximate multiply-accumulate units

In recent years, with the in-depth study of deep neural networks, approximate multiply-accumulate units (AMACs) have attracted widespread attention. In deep neural networks, convolution operations account for over 90% of the computational workload, and the multiply-accumulate unit (MAC), as one of the primary operations, consumes a significant amount of energy. Since 2017, researchers have begun exploring the use of approximate multipliers and approximate adders for the approximate design of MACs. Approximation methods

Table 3. Approximation techniques for dividers

Approximation method	Related work	Overview
Array approximation	Literature [26, 27]	Approximate design of subtractors in traditional array structures to reduce the complexity of the divider array.
Operand approximation	Literature [28, 29]	Truncation of operands, or truncation starting from the first 1, significantly reduces computation delay and energy with minimal precision loss.
Hybrid array-operand approximation	Literature [30]	Optimization of Booth encoding results using Karnaugh maps to simplify the partial product expression of the Booth algorithm.

for multiply-accumulate units mainly include multiplier approximation, adder approximation, and the detailed classification of approximation methods for merged multiply-add arrays, as shown in **Table 4**.

Yin *et al.* [25] explored the use of asynchronous approximate multiply-accumulate operators and investigated how to leverage the advantages of asynchronous circuits while mitigating their inherent area overhead. By analyzing three approximate MAC architectures with different error rates and area trade-offs, a comparison is made between precise and approximate, synchronous and asynchronous MAC operators. Experiments demonstrate that, under different controlled error rates, the area overhead of asynchronous MACs can be significantly reduced by reusing approximate multipliers. Shriram *et al.* [27] studied the application of approximate adders in the final stage of multiply-accumulate units and proposed a design flow based on synthesis tools. The applied 28 nm CMOS design example shows that this design can achieve a 14% power gain with a marginal decrease in image quality. Wu *et al.* [28] presented a novel approximate multiply-accumulate unit that utilizes static segmentation to compute Y = A×B+C. The proposed architecture employs a unique carry-save adder and segments the three operands A, B, and C to reduce hardware costs. The performance of the proposed approximate multiply-accumulate unit is superior to existing technologies, significantly reducing power consumption.

3. Research progress of approximate computing at the algorithmic level in signal processing

Approximate computing in signal processing algorithms primarily focuses on two directions: approximate filtering and approximate transformation. These studies aim to meet specific application requirements by introducing approximate computing techniques while reducing computational complexity and energy consumption. Approximate filtering is an important research direction widely used in tasks such as denoising, smoothing, and edge detection. Its goal is to reduce computational resources and energy consumption while maintaining filtering effectiveness. On the other hand, approximate transformation plays a crucial role in tasks like frequency domain analysis, compression, and feature extraction. Its objective is to lower computational complexity through approximate computing techniques while preserving the accuracy of transformation results.

3.1. Approximate filtering algorithms

In recent years, approximate filtering algorithms have garnered significant research attention due to their ability to enhance filtering speed by reducing computational precision and simplifying computational processes. Driven by approximate computing, researchers have conducted a series of explorations targeting approximate FIR filters. For instance, Jiang *et al.* [38] proposed a fixed-point finite impulse response adaptive filter employing an approximate distributed algorithm. This design utilizes a radix-8 Booth algorithm to reduce the number of partial products in the distributed algorithm architecture. Additionally, it approximates the generation of partial products by truncating input data and performing error compensation. To further lower hardware costs, an approximate Wallace tree is utilized for the accumulation of partial products. Consequently, this design significantly reduces delay, area, and power consumption.

Esposito *et al.* [39] presented a quality-scalable

Table 4. Approximation techniques for multipliers and accumulators

Approximation method	Related work	Overview
Multiplier approximation	References [31, 32]	Design approximate MACs by segmenting the multiplication operands.
Adder approximation	References [33, 34]	Reduce power consumption by applying approximate adders to the final carry-propagate adder of signed MAC units, while adjusting input voltage using VOS.
Merged approximation of multiplication and addition arrays	References [35–37]	Design approximate MAC units by inserting accumulation into the multiplication partial product array.

approximate Least Mean Square (LMS) filter where the level of approximation during runtime can be altered by adjusting an external quality knob. This method introduces approximation at the algorithmic level. By freezing the updates of certain coefficients, the filter can automatically enter a low-power approximate mode.

Meanwhile, Di Meo *et al.* [40] proposed a novel approximate implementation method for the Delayed Least Mean Square (DLMS) filter, which updates filter coefficients based on the magnitude of the error signal. Compared to the original DLMS algorithm, the proposed filter achieves a power savings of 53.7%.

Furthermore, Monteiro *et al.* [41] explored the combination of multiplier-free multiple constant multiplication and approximate computing techniques in Gaussian filters. It investigates the impact of three different kernel sizes on image processing. Utilizing a replication strategy, the study evaluates the influence of approximating the least significant bits of adders at various levels. The results demonstrate that all evaluated kernel sizes can reduce power consumption and area.

3.2. Approximate transformations

Approximate transformations primarily include approximate fast Fourier transforms and approximate discrete cosine transforms. The fast Fourier transform is a commonly used method for frequency domain transformation, widely applied in areas such as spectrum analysis, filtering, and signal compression in signal processing. Discrete cosine transforms are extensively used in fields like image and audio compression, as well as signal feature extraction. By introducing moderate approximation techniques into transformation calculations, efficient signal processing can be achieved while reducing computational complexity and resource consumption.

3.2.1. Approximate fast Fourier transform

The Discrete Fourier Transform (DFT) is one of the essential computations in digital signal processing. However, due to its high computational complexity and large computational requirements, it has not been widely used. Since the introduction of the Fast Fourier Transform (FFT) algorithm based on time decimation in 1965, the computation speed of the DFT algorithm has increased by nearly 100 times. The FFT algorithm has experienced rapid development and garnered significant academic attention. Later, Bergland [42] proposed high-radix algorithms based on the 2-point FFT algorithm, such as radix-4 and radix-8, to reduce computational load. Typically, the size of FFT processing is represented using powers of 2. However, the Long Term Evolution (LTE) system of universal mobile communication technology involves 1536-point FFT calculations, increasing the difficulty of hardware design. To address this, Elango and Muniandi [43] optimized the algorithm to reduce the number of multipliers and replaced the precise multipliers of radix-2 butterfly units with approximate multipliers. This results in a 40% increase in logic utilization and a 33% improvement in speed for the FFT processor. As it is difficult to directly correlate the precision of basic units with the overall precision of the FFT processor, approximation design schemes based on basic units face challenges in achieving designs tailored to specific precision requirements, leading to poor portability. Liu *et al.* [44] employed an 8-stage radix-2 single-path delay feedback FFT and a precision adaptive adjustment architecture, along with multi-voltage approximate multiplication and addition. This approach reduces power consumption by 76% for voice keyword recognition while maintaining comparable accuracy. Liu *et al.* [45] presented two approximate bit-width selection algorithms for FFT processors with specific precision requirements. These algorithms facilitate the identification of bit-width combinations at various stages of the FFT processor that meet precision requirements while minimizing resource usage or latency, thereby enhancing hardware performance.

3.2.2. Approximate discrete cosine transform

Computing the traditional 8-point Discrete Cosine Transform (DCT) requires 64 multiplications and 56 additions, making it necessary to investigate fast DCT transformation algorithms. Currently, research on fast DCT algorithms can be broadly classified into two categories. One category focuses on reducing the number of floating-point multipliers and adders in the DCT. In 1977, Chen *et al.* [46] proposed a fast algorithm for DCT using sparse matrix decomposition based on the symmetry of the transformation matrix. This algorithm computes the 8-point DCT with only 16 multiplications and 26 additions. However, this fast algorithm still requires floating-point multiplication, where the power consumption of the multipliers accounts for 40% of the total power consumption, and the hardware complexity represents 45% of the overall hardware complexity. This structure is slow in both hardware and software implementations.

Another category of research on fast DCT algorithms aims to reduce hardware overhead by utilizing a multiplier-free integer DCT fast algorithm. One such integer DCT fast algorithm decomposes DCT coefficients into a sparse matrix (where matrix elements only include 0, ±1/2, ±1, or ±2) multiplied by a diagonal matrix. Under image compression conditions, the diagonal matrix can be simply incorporated into the quantization step of the image compression process [47,48]. Therefore, in this case, the complexity of DCT computation is related to the complexity of the sparse matrix. Since the elements in the sparse matrix only consist of powers of 2 such as {0, ±1/2, ±1, ±2}, the computation process becomes multiplier-free. Another integer DCT fast algorithm is based on Multiple Constant Multiplication (MCM). This method multiplies all elements in the matrix by a large value and then rounds them to the nearest integer [49,50]. To further reduce computational complexity, integer multipliers are replaced with shift and add operations.

4. Approximate computing and its applications in signal processing

Approximate computing is widely used in signal processing, including wireless communication systems, video and image processing, and radar signal processing. In wireless communication, approximate computing can enhance system performance and efficiency and reduce computational complexity and power consumption, such as reducing computational requirements in wireless signal modulators and demodulators. In video and image processing, approximate computing can accelerate processing speed, reduce resource consumption, and adapt to various platforms and devices. In radar signal processing, approximate computing can be applied to tasks such as power spectrum estimation, target recognition, and parameter estimation to reduce computational complexity, improve real-time performance and scalability, and meet the demands of complex environments.

4.1. Applications of approximate computing in wireless communication systems

As we enter the era of the Internet of Everything, communication between devices is becoming more frequent, and the amount of data is constantly increasing, making floating-point units (FPUs) extremely important. As the foundation of the Internet of Everything, wireless communication systems widely use FPUs. Given the inherent error tolerance of wireless communication systems, more and more researchers are focusing on the study of approximate FPUs and their application in wireless communication systems to effectively reduce energy consumption within a certain error tolerance range. In the study by Janhunen *et al.* [51], the authors proposed a block floating-point enhanced filter matrix computation unit architecture for multiple-input multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) communication systems. Compared with fixed-point implementations, the block floating-point format can significantly reduce the total circuit area while reducing bit width without degrading bit error rate performance. Wireless communication systems often involve large-scale and complex matrix inversion operations, and orthogonal triangular decomposition is a commonly used solution for matrix inversion. Therefore, another literature [52] proposed a 4x4 matrix design based on floating-point arithmetic, which effectively increases data throughput during orthogonal triangular decomposition. Furthermore, Hu and Koibuchi [53] proposed the use of approximate floating-point compression to accelerate Message Passing Interface (MPI) communication on lossy interconnection networks. By designing an application-level fast approximate compression algorithm and proposing a key bit-flipping recovery scheme optimized for a given bit error rate under lossy interconnection networks, data transmission volume is significantly increased within a certain error range.

4.2. Applications of approximate computing in video and image processing

With the widespread popularity of multimedia applications, the demand for video and image processing is increasingly urgent, leading to significant research interest in the hardware implementation of low-power video and image processing applications. Since images and videos are tolerant of a certain degree of error, many researchers attempt to strike a balance between video and image output quality and energy consumption, significantly improving energy efficiency by sacrificing video and image accuracy to some extent. Park *et al.* [54] proposed an algorithm that can dynamically adjust the bit width of operands in the DCT hardware structure based on the differences in error sensitivity among 64 DCT coefficients in the DCT algorithm. Snigdha *et al.* [55] investigate the approximate feasibility of basic operational units (adders/multipliers) within the DCT based on the Loemer algorithm. Based on the varying impacts of errors introduced by adders/multipliers at different computational stages in the Loemer algorithm on the final output result, they proposed a mathematical model that can inversely derive the appropriate bit width for each operational unit given an output error budget to maximize power savings. However, these methods only approximate some computational units in the JPEG encoder hardware structure, resulting in limited power reduction. Another effective method to reduce power consumption is to introduce low-voltage technology into JPEG encoder circuit design. Pu *et al.* [56] presented a JPEG encoder designed in a 65 nm CMOS process capable of operating at a wide voltage range of 0.4 to 1.2V. To improve throughput, the encoder employs four parallel driver modules and one Huffman encoding module, where each driver module consists of a pair of DCT and quantization modules. The driver modules and Huffman module operate at different voltages and clock frequencies. In the sub-threshold region, the driver modules can operate normally at a minimum voltage of 400 mV and a frequency of 2.5 MHz, while the Huffman module runs at a voltage of 600 mV and a frequency of 10 MHz.

4.3. Applications of approximate computing in radar signal processing

Conventional compressed sensing radar imaging methods not only enable scene imaging but also reduce the required amount of data, i.e., the sampling rate. However, these methods have high computational complexity and greater demand for computer memory. Therefore, the introduction of approximate computing techniques can reduce computational complexity and memory consumption. Fang *et al.* [57] proposed a new compressed sensing synthetic aperture radar (CS-SAR) imaging method that utilizes approximate observation operators to significantly reduce computational complexity and memory consumption, making it suitable for CS-SAR imaging systems with large data volumes or large scenes. The CS-SAR imaging method based on approximate observation is sometimes also referred to as the range-azimuth decoupled CS-SAR imaging method [58]. Compared to traditional CS-SAR imaging methods, the approximate observation-based CS-SAR imaging method can significantly reduce memory consumption and the computational complexity of single steps in iteration. However, since the phase angles of SAR images are always random, this poses difficulties in processing complex-valued SAR images. Li *et al.* [59] presented a magnitude-phase separation method for CS-SAR imaging based on approximate observation. Compared to existing methods, this approach only applies sparse constraints to the magnitude of smooth components, while phase angles remain random, resulting in better reconstruction capabilities. Additionally, due to the inherent low memory requirements of approximate observation, the proposed method requires less memory overhead. In the presence of phase errors, synthetic aperture radar reconstructed images can exhibit defocusing. Li *et al.* [60] proposed a phase error correction method for compressed sensing radar imaging based on approximate observation. Compared to traditional methods, this approach offers better image-focusing capabilities and reduced memory overhead.

5. Reflection and outlook

Currently, approximate computing technology is continuously evolving in the field of signal processing, providing efficient and scalable solutions for processing large-scale signal data. With the widespread adoption of mobile devices and the Internet of Things, there is an increasing demand for low-power and efficient signal processing hardware. In the future, approximate computing technology will focus on developing high-performance signal processing hardware platforms suitable for resource-constrained environments. Simultaneously, as the importance of machine learning in signal processing continues to grow, there will be a greater emphasis on combining approximate computing techniques with machine learning to achieve faster and more accurate signal processing tasks. With the rapid development of edge computing and the Internet of Things, signal processing tasks are no longer limited to central servers or the cloud but are distributed across multiple edge devices. Future approximate computing technology will prioritize distributed and collaborative processing, enabling multiple devices to work together to complete signal processing tasks, improving response speed and system fault tolerance.

However, the larger-scale application and deployment of approximate computing technology in the field of signal processing still face some significant challenges. These challenges primarily involve theoretical issues of error analysis in approximate design, design versatility, and systematic design methodology.

(1) Theoretical issues of error analysis in approximate design: Since approximate design inherently introduces errors, analyzing these errors theoretically can reduce them to some extent and facilitate the selection of the desired level of approximation. However, due to the varying sensitivity of different approximation modules to errors, considering the impact of errors from different modules on the overall system accuracy and determining the relative weight of errors from different modules compared to system errors remain topics of current research in approximate computing. Currently, there are analytical theories for the maximum error of approximate logarithmic multipliers, and related studies compensate for truncation-induced errors through probabilistic analysis. Nevertheless, these error analyses have not yet formed a systematic theoretical model. Therefore, establishing a systematic error model for different modules will aid in selecting arithmetic units with varying degrees of approximation in different application scenarios.

(2) Issues of generality in approximate computing design: The current design optimization of approximate computing mainly focuses on a single design level, such as introducing the idea of approximate computing in specific applications. This approach demonstrates good results in specific applications but has limited applicability and lacks generality. Additionally, research on approximate computing algorithms tends to favor the design of specialized algorithms to meet the needs of specific tasks or domains. While this method can achieve high performance and efficiency, there are limitations in terms of general design. To address these issues, further research on general design methods is needed. This includes integrating approximate computing techniques across multiple levels and pursuing hardware-software co-design. It requires designing general approximate computing methods at the algorithmic level and implementing corresponding support and optimization at the architecture and hardware design levels. Through hardware-software co-design, the energy efficiency potential of approximate computing can be better explored, and efficient, scalable, and general approximate computing solutions can be achieved in various domains.

(3) Issues of systematic design methods for approximate computing: Since approximate computing techniques can be widely applied at different levels of computing systems, including hardware, software, and architecture layers, and the measurement metrics required for approximate computing at different levels are inconsistent (for example, error metrics for the hardware level are not suitable for measuring errors in upper-layer applications), there is currently a lack of systematic design guidance for approximate computing. Therefore, in subsequent research, it is necessary to systematically design and evaluate approximate computing across multiple levels and propose relevant measurement metrics for systematic approximate computing techniques. Meanwhile,

system-level optimization methods can be employed in the systematic design of approximate computing. This includes model-based design space exploration and optimization to find the best configuration and parameter settings for approximate computing while meeting requirements such as performance, power consumption, and accuracy.

6. Conclusion

This article provides a comprehensive overview of approximate computing techniques for signal processing, summarizing recent research progress and applications in this field. By adopting approximate computing techniques, the efficiency and performance of signal processing systems can be effectively improved. Future research directions include further optimizing approximate computing methods and improving approximate error analysis. These research outcomes will offer important references for the study and application of signal processing.

Funding

(1) National Key Research and Development Program of China for Young Scientists (2022YFB4500200)

(2) National Natural Science Foundation of China (62101252, 62022041)

Disclosure statement

The authors declare no conflict of interest.

References

[1] Liu W, Lombardi F, 2022, Approximate Computing, Springer, Cham, 365–368.

[2] Liu W, Lombardi F, Schulte M, 2020, Approximate Computing: From Circuits to Applications. Proceedings of the IEEE, 108(12): 2103–2107.

[3] Chippa VK, Mohapatra D, Roy K, et al., 2014, Scalable Effort Hardware Design. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(9): 2004–2016.

[4] Nowick SM, 1996, Design of a Low-Latency Asynchronous Adder Using Speculative Completion. IEE Proceedings Computers and Digital Techniques, 143(5): 301–307.

[5] Lu SL, 2004, Speeding Up Processing with Approximation Circuits. Computer, 37(3): 67–73.

[6] Esposito D, De Caro D, Napoli E, et al., 2015, Variable Latency Speculative Han-Carlson Adder. IEEE Transactions on Circuits and Systems I: Regular Papers, 62(5): 1353–1361.

[7] Seok H, Seo H, Lee J, et al., 2022, A Novel Efficient Approximate Adder Design Using Single Input Pair Based Computation, 2022 19th International SoC Design Conference (ISOCC), Gangneung-si, Korea, 57–58.

[8] Seo H, Kim Y, 2023, A Low Latency Approximate Adder Design Based on Dual Sub-Adders with Error Recovery. IEEE Transactions on Emerging Topics in Computing, 11(3): 811–816.

[9] Manohar PS, Rohan B, Ramana PVS, et al., 2023, Implementation of Carry Look Ahead Adder with 2-Bit Approximate Adder, 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC), Salem, India, 1543–1547.

[10] Yan A, Wei S, Li Z, et al., 2023, Design of Low-Cost Approximate CMOS Full Adders, 2023 IEEE International Symposium on Circuits and Systems (ISCAS), Monterey, USA, 1–5.

[11] Lagidi P, Iswarya A, Rajesh G, et al., 2021, Design of 16-Bit and 32-Bit Approximate Full Adder Using Majority Logic, 2021 2nd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 1–5.

[12] Liu B, Xue A, Wang Z, et al., 2023, A Reconfigurable Approximate Computing Architecture with Dual-VDD for Low-Power Binarized Weight Network Deployment. IEEE Transactions on Circuits and Systems II: Express Briefs, 70(1): 291–295.

[13] Mitchell JN, 1962, Computer Multiplication and Division Using Binary Logarithms. IRE Transactions on Electronic Computers, EC-11(4): 512–517.

[14] Kim MS, Del Barrio AA, Oliveira LT, et al., 2019, Efficient Mitchell's Approximate Log Multipliers for Convolutional Neural Networks. IEEE Transactions on Computers, 68(5): 660–675.

[15] Nunziata I, Zacharelos E, Saggese G, et al., 2022, Approximate Recursive Multipliers Using Carry Truncation and Error Compensation, 2022 17th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), Villasimius, Italy, 137–140.

[16] Waris H, Wang C, Liu W, et al., 2022, Hybrid Partial Product-Based High-Performance Approximate Recursive Multipliers. IEEE Transactions on Emerging Topics in Computing, 10(1): 507–513.

[17] Shankar RG, Ananthi DR, 2023, Approximate Booth Multipliers Using Compressors and Counter, 2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 1658–1662.

[18] Liu B, Cai H, Zhang Z, et al., 2023, Multiplication Circuit Architecture for Error-Tolerant CNN-Based Keywords Speech Recognition. IEEE Design & Test, 40(3): 26–35.

[19] Sayadi L, Timarchi S, Sheikh-Akbari A, 2023, Two Efficient Approximate Unsigned Multipliers by Developing New Configuration for Approximate 4: 2 Compressors. IEEE Transactions on Circuits and Systems I: Regular Papers, 70(4): 1649–1659.

[20] Zhang M, Nishizawa S, Kimura S, 2023, Area Efficient Approximate 4-2 Compressor and Probability-Based Error Adjustment for Approximate Multiplier. IEEE Transactions on Circuits and Systems II: Express Briefs, 70(5): 1714–1718.

[21] Xie N, Zhang R, Yan H, et al., 2022, Compressors Evolution Based High Speed and Energy Efficient Approximate Signed Multiplier, 2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT), Nanjing, China, 1–3.

[22] Tong JYF, Nagle D, Rutenbar RA, 2000, Reducing Power by Optimizing the Necessary Precision/Range of Floating-Point Arithmetic. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8(3): 273–286.

[23] Eilert J, Ehliar A, Liu D, 2004, Using Low Precision Floating Point Numbers to Reduce Memory Cost for MP3 Decoding, 2004 IEEE 6th Workshop on Multimedia Signal Processing, Siena, Italy, 119–122.

[24] Zhang H, Putic M, Lach J, 2014, Low Power GPGPU Computation with Imprecise Hardware, 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, USA, 1–6.

[25] Yin P, Wang C, Liu W, et al., 2016, Design and Performance Evaluation of Approximate Floating-Point Multipliers, 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, USA, 296–301.

[26] Savio MMD, Deepa T, Dharshini PD, et al., 2023, Design and Implementation of Approximate Divider for Error-Resilient Image Processing Applications, 2023 Second International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichirappalli, India, 1–5.

[27] Shriram A, Tiwari A, Anil Kumar U, et al., 2022, Power Efficient Approximate Divider Architecture for Error-Resilient Application, 2022 IEEE6th Conference on Information and Communication Technology (CICT), Gwalior, India, 1–6.

[28] Wu Y, Jiang H, Ma Z, et al., 2022, An Energy-Efficient Approximate Divider Based on Logarithmic Conversion and Piecewise Constant Approximation. IEEE Transactions on Circuits and Systems I: Regular Papers, 69(7): 2655–2668.

[29] Saadat H, Javaid H, Parameswaran S, 2019, Approximate Integer and Floating-Point Dividers with Near-Zero Error Bias,

2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, USA, 1–6.

[30] Liu W, Xu T, Li J, et al., 2022, Design of Unsigned Approximate Hybrid Dividers Based on Restoring Array and Logarithmic Dividers. IEEE Transactions on Emerging Topics in Computing, 10(1): 339–350.

[31] Wuerdig RN, Sartori MLL, Abreub A, et al., 2022, Mitigating Asynchronous QDI Drawbacks on MAC Operators with Approximate Multipliers, 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, USA, 1269–1273.

[32] Mishra V, Pandey D, Singh S, et al., 2022, ART-MAC: Approximate Rounding and Truncation Based MAC Unit for Fault-Tolerant Applications, 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, USA, 1640–1644.

[33] Esposito D, De Caro D, Napoli E, et al., 2017, On the Use of Approximate Adders in Carry-Save Multiplier-Accumulators, 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, USA, 1–4.

[34] Wang Z, Wei Q, Xue A, et al., 2022, Low-Power Computing Unit Based on Heterogeneous Approximate Structure for Binary Convolutional Neural Network, 2022 IEEE 16th International Conference on Solid-State & Integrated Circuit Technology (ICSICT), Nanjing, China, 1–3.

[35] Di Meo G, Saggese G, Strollo AGM, et al., 2023, Approximate MAC Unit Using Static Segmentation. IEEE Transactions on Emerging Topics in Computing, (99): 1–12.

[36] Liu B, Zhang Z, Cai H, et al., 2022, Self-Compensation Tensor Multiplication Unit for Adaptive Approximate Computing in Low-Power CNN Processing. Science China Information Sciences, 65(4): 149403.

[37] Liu B, Zhang R, Shen Q, et al., 2023, W-AMA: Weight-Aware Approximate Multiplication Architecture for Neural Processing. Computers and Electrical Engineering, (111): 108921.

[38] Jiang H, Liu L, Jonker PP, et al., 2019, A High-Performance and Energy-Efficient FIR Adaptive Filter Using Approximate Distributed Arithmetic Circuits. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(1): 313–326.

[39] Esposito D, Di Meo G, De Caro D, et al., 2018, Quality-Scalable Approximate LMS Filter, 2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Bordeaux, France, 849–852.

[40] Di Meo G, De Caro D, Petra N, et al., 2022, A Novel Low-Power DLMS Adaptive Filter with Sign-Magnitude Learning and Approximated FIR Section, 2022 17th Conference on Ph. D. Research in Microelectronics and Electronics (PRIME), Villasimius, Italy, 217–220.

[41] Monteiro M, Seidel I, Grellert M, et al., 2022, Exploring the Impacts of Multiple Kernel Sizes of Gaussian Filters Combined to Approximate Computing in Canny Edge Detection, 2022 IEEE 13th Latin America Symposium on Circuits and System (LASCAS), Puerto Varas, Chile, 1–4.

[42] Bergland G, 1969, Fast Fourier Transform Hardware Implementations: An Overview. IEEE Transactions on Audio and Electroacoustics, 17(2): 104–108.

[43] Elango K, Muniandi K, 2020, VLSI Implementation of an Area and Energy Efficient FFT/IFFT Core for MIMO-OFDM Applications. Annals of Telecommunications, 75(5/6): 215–227.

[44] Liu B, Ding X, Cai H, et al., 2021, Precision Adaptive MFCC Based on R2SDF-FFT and Approximate Computing for Low-Power Speech Keywords Recognition. IEEE Circuits and Systems Magazine, 21(4): 24–39.

[45] Liu W, Liao Q, Qiao F, et al., 2019, Approximate Designs for Fast Fourier Transform (FFT) with Application to Speech Recognition. IEEE Transactions on Circuits and Systems I: Regular Papers, 66(12): 4727–4739.

[46] Chen WH, Smith C, Fralick S, 1977, A Fast Computational Algorithm for the Discrete Cosine Transform. IEEE Transactions on Communications, 25(9): 1004–1009.

[47] Potluri US, Madanayake A, Cintra RJ, et al., 2014, Improved 8-Point Approximate DCT for Image and Video Compression Requiring Only 14 Additions. IEEE Transactions on Circuits and Systems I: Regular Papers, 61(6): 1727–1740.

[48] Da Silveira TLT, Canterle DR, Coelho DFG, et al., 2022, A Class of Low-Complexity DCT-Like Transforms for Image and

Video Coding. IEEE Transactions on Circuits and Systems for Video Technology, 32(7): 4364–4375.

[49] Xing Y, Zhang Z, Qian Y, et al., 2018, An Energy-Efficient Approximate DCT for Wireless Capsule Endoscopy Application, 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 1–4.

[50] Cai L, Qian Y, He Y, et al., 2021, Design of Approximate Multiplierless DCT with CSD Encoding for Image Processing, 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 1–4.

[51] Janhunen J, Pitkanen T, Silven O, et al., 2011, Fixed-and Floating-Point Processor Comparison for MIMO-OFDM Detector. IEEE Journal of Selected Topics in Signal Processing, 5(8): 1588–1598.

[52] Amin-Nejad S, Basharkhah K, Gashteroodkhani TA, 2019, Floating Point Versus Fixed Point Tradeoffs in FPGA Implementations of QR Decomposition Algorithm. European Journal of Electrical Engineering and Computer Science, 3(5): 127.

[53] Hu Y, Koibuchi M, 2021, Accelerating MPI Communication Using Floating-Point Compression on Lossy Interconnection Networks, 2021 IEEE 46th Conference on Local Computer Networks (LCN), Edmonton, Canada, 355–358.

[54] Park J, Choi JH, Roy K, 2010, Dynamic Bit-Width Adaptation in DCT: An Approach to Trade Off Image Quality and Computation Energy. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18(5): 787–793.

[55] Snigdha FS, Sengupta D, Hu J, et al., 2016, Optimal Design of JPEG Hardware Under the Approximate Computing Paradigm, 2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, USA, 1–6.

[56] Pu Y, De Gyvez JP, Corporaal H, et al., 2010, An Ultralow-Energy Multi-Standard JPEG Co-Processor in 65 nm CMOS with Sub/Near Threshold Supply Voltage. IEEE Journal of Solid-State Circuits, 45(3): 668–680.

[57] Fang J, Xu Z, Zhang B, et al., 2014, Fast Compressed Sensing SAR Imaging Based on Approximated Observation. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7(1): 352–363.

[58] Jiang C, Zhang B, Fang J, et al., 2014, Efficient ℓq Regularisation Algorithm with Range-Azimuth Decoupled for SAR Imaging. Electronics Letters, 50(3): 204–205.

[59] Li B, Liu F, Zhou C, et al., 2018, Mixed Sparse Representation for Approximated Observation-Based Compressed Sensing Radar Imaging. Journal of Applied Remote Sensing, 12(3): 035015.

[60] Li B, Liu F, Zhou C, et al., 2017, Phase Error Correction for Approximated Observation-Based Compressed Sensing Radar Imaging. Sensors, 17(3): 613.