Design of Floating-Point Multiplier Architecture with Adaptive Data Timing Channels

Hoda Ghabeli
Department of Computer Engineering Kerman Branch
Islamic Azad University
Kerman, Iran
h.ghabeli@iauk.ac.ir

***Abstract*— The mantissa multiplication is a significant portion of the floating-point multiplier. In this paper, a multi-path mantissa processing (MPM) floating-point multiplier scheme is proposed. The mantissa computation path is divided into several sub-paths for higher execution speed and energy efficiency. The computation paths are considered based on the characteristics of the input data to reduce the risk of generating approximate results. Different input data can generate variable latency and data time channels. The proposed architecture includes a design of a modified 5-2 compressor for a specific range of floating-point data for utilization in a floating-point multiplier. An adaptive data time channels approach employs a controller to select the appropriate path according to the input data. The proposed design operates in parallel with the exact module and is accurate, which improves the operating speed of the entire circuit.**

Keywords—Arithmetic Circuits, floating-point multiplier, multi-path mantissa processing

I. INTRODUCTION

Convolution is an essential part of the signal processing toolbox and the important operation in convolution is multiplication. In convolution computations, floating-point numbers are often used because of the processing capability and high precision of floating-point numbers. Hence, developing faster, smaller and power-efficient floating-point multipliers is principal to implementing efficient circuits [1]-[3]. Approximate computing is a common operation for high-performance, high-speed and low-power computing [4]. Recently, approximate floating-point multipliers have gained significant attention due to the potential to enhance performance and efficiency in multimedia processing and neural network inference [5]-[8]. Since the multiplication of the mantissa processing step is time-consuming in floating-point multiplication, thus focusing specifically on this step, approximate mantissa computations are proposed to improve floating-point multiplier performance [9]-[11]. Accuracy in the convolution operation is very important and the smallest error in computations leads to significant changes in the final result and performance. Therefore, using the approximate technique with the importance of high accuracy in computations of floating-point is contradictory and using the approximate technique alone is not enough. The main contribution of this paper is predicting a shorter path than the critical path that generates an accurate result with a high probability based on the characteristics of the input data which provides a result faster than the critical path. This satisfies the challenge of time-consuming of the mantissa multiplication step. The circuit is designed with short paths along with a conventional circuit for controlling the correctness of the result in case of a wrong prediction. This satisfies the challenge of the importance of accuracy in the computations of the floating-point. The rest of the paper is structured as follows: Section II provides background and preliminaries of the floating-point multiplier, and reviews the previous approximate floating-point multiplier designs; Section III presents the proposed multi-path mantissa processing (MPM) of the floating-point multiplier scheme. Section IV reports the experimental results, followed by conclusions in Section V.

II. BACKGROUND AND RELATED WORKS

A. floating-point multiplier

The floating-point multiplication of two pairs of operands is a very frequent operation that is needed in many digital signal processing (DSP) applications. The single-path mantissa processing (SPM) floating-point multiplier unit, shown in Fig. 1, executes the equation $P = A \times B$. where, $A = a_{31}a_{30}\ldots a_1a_0$ and $B = b_{31}b_{30}\ldots b_1b_0$ are multiplied and their product is $P = p_{31}p_{30}\ldots p_1p_0$. The floating-point numbers A, B and P are represented as follows:

$$A = (-1)^{A_S} \times (1+A_M)\times 2^{A_E-127} \quad (1)$$

$$B = (-1)^{B_S} \times (1+B_M)\times 2^{B_E-127} \quad (2)$$

$$P = (-1)^{P_S} \times (1+P_M)\times 2^{P_E-127} \quad (3)$$

where $A_S[31]$, $B_S[31]$ and $P_S[31]$, are the sign, of A, B and P respectively. The exponent, of A, B and P are represented by $A_E[30:23]$, $B_E[30:23]$ and $P_E[30:23]$ respectively. The mantissa of A, B and P can be expressed by $A_M[22:0]$, $B_M[22:0]$ and $P_M[22:0]$ respectively. Multiplication of two floating-point numbers A and B is done in three steps, (i) computation of the sign, The sign of the product is obtained as follows:

$$P_S[31] = A_S[31] \oplus B_S[31] \quad (4)$$

where $\oplus$ denote the logical exclusive-OR operation, (ii) taking into account the bias, the exponents of the input operands were added together and the bias was subtracted from the sum of the exponents, and is stored in an 8-bit

2024 6th Iranian International Conference on Microelectronics (IICM) | 979-8-3315-1137-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/IICM65053.2024.10824412

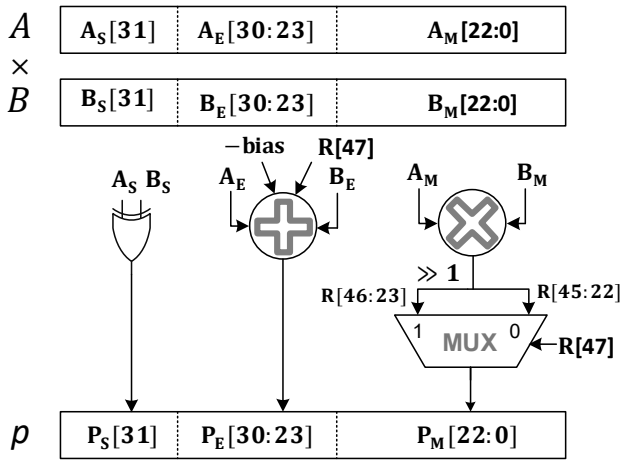

Fig. 1. Conventional single-path mantissa processing (SPM) floating-point multiplier.

register, the exponent of product is expressed as:

$$P_E[7:0] = A_E[7:0] + B_E[7:0] - 127 \quad (5)$$

(iii) non-signed multiplication of mantissa and normalization of the result that the exponent be modified accordingly. The output of the multiplication of mantissa is computed as follows:

$$R = (A_M + 1) \times (B_M + 1) \quad (6)$$

that (6) simplifies as:

$$R = 1 + (A_M + B_M) + (A_M \times B_M) \quad (7)$$

where the M is normalized ($1 \leq M < 2$) and has 23 bits [22:0]. The R signal is generated from the multiplication of the normalized mantissa signals and is composed of 48 bits including the high-order bits R[47] and R[46] for the representation of the integer part. The normalization is done based on R[47] and also the exponent is adjusted depending on the R[47]. If R[47] is '0', the R[45:22] is stored in the register. If R[47] is '1', the output of R is shifted to the right to maintain the normalized mantissa and the result R[46:23] is stored in a register and to compensate this right-shift, the exponent is also increased by one unit [12]. Of the above three steps used in a mantissa multiplication, step (iii) typically requires a higher rate of delay, area and power.

B. Previous Approximate Floating-Point Multiplier

Several approximate floating-point multipliers have been proposed in the literature [9]-[11]. The main idea behind their design is a static and dynamic segmented multiplier for the mantissa. The proposed approximate floating-point multiplier [9] decreases the computations by removing the sum of the value of '1' from (7). As well as multiplication of mantissa part is obtained by segmentation and truncation techniques to reduce hardware. The size of the multiplier and the adder of the mantissa are reduced by discarding the chain of zero bits of mantissas. Although the error rate was small, its result is still approximate. An approximate floating-point multiplier has been proposed in [10], where to reduce the hardware complexity, the product $A_M \times B_M$ in (7) substitutes with a thought-out value. Since the operation of multiplication is a complicated operation, this structure is used to replace multiplication with addition in (7) at the cost of high error distances. This structure utilized a segmented mantissa, leading to smaller delays and power consumption. In [11], the multiplication of the mantissa step has been replaced by one of the input operands as the output of this step. All the above approximate floating-point multipliers operated only in the approximate mode. Previous approximation techniques were proposed regardless of the input data. However, a certain spectrum of numbers is used in image processing applications, such as image filter kernels. This feature can be used in the design of the computing circuit. The convolution process is considered the main step in the convolutional neural network (CNN). What happens in each layer of convolution is placing a filter/kernel on the input image to finally obtain a filtered output that contains its original features. The kernel is a matrix that moves on the input image and its value is multiplied by the value of the input matrix; Finally, it provides a matrix in the output that has features of the image that are considered, such as smoothing, sharpening and edge detection. The values of the kernels are usually in the range of one-digit to two-digit numbers. Table I shows some examples of image filters and related kernels [2].

TABLE I. DIFFERENT COMMON KERNELS FOR LAPLACIAN

Filter	kernel
3 × 3 Laplacian	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ $\begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}$ $\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$

III. PROPOSED MULTI-PATH MANTISSA PROCESSING (MPM) FLOATING-POINT MULTIPLIER DESIGN

Since the multiplication of mantissa is a rather costly step, in terms of delay, compared to other steps, approximate arithmetic is highly preferred for this step. A single path mantissa multiplication is to perform the computation of (7). But there can be shorter paths depending on inputs and mantissa. In the proposed method, a single path of mantissa multiplication is separated into multiple paths to provide the different delays for possible input combinations. The multi-path mantissa processing (MPM) floating-point multiplier is shown in Fig. 2. Unlike the previous methods, where an approximation was created on the formula, in this paper, special paths for a certain group of floating-point values are embedded. Special values are most likely to exist in filters/kernels.

Three-path multiplication of mantissa works under three different inputs for multiplier or multiplicand, (i) normal inputs, (ii) inputs in the range 0 to 96 and (iii) powers of 2 inputs where the mantissa part is zero. Path 1: path 1 is the

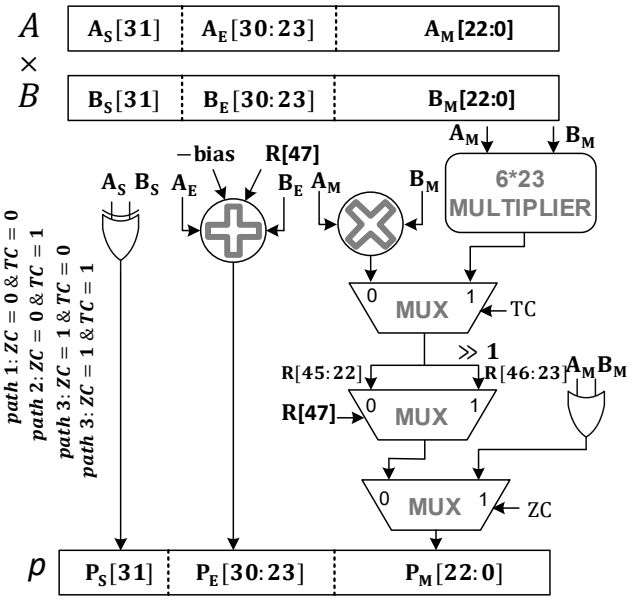

Fig. 2. Multi-path mantissa processing (MPM) floating-point multiplier.

conventional mantissa multiplication. Path 2: path 2 is special for multiplying of maximum 2-digit numbers by multi-digit numbers. Its operational steps are similar to the conventional mantissa multiplication and its subsequent steps are mantissa multiplication and normalization, except that the mantissa multiplication is done proportionally to the certain values of the mantissa (maximum 2-digit numbers). Because the values of the kernels are usually in the range of one-digit to two-digit numbers. Path 3: the mantissa multiplication when the mantissa part of at least one of the multiplicand or multiplier is zero.

The general feature of representing floating-point numbers in the range from 0 to 99 is that the non-zero values of the mantissa are only in the range of bits 17 to 22 and the mantissa is zero in the range of 0 to 16. Therefore, in this case, to multiply the number in this range by another number (6 × 23 multiplier), a 5-2 compressor and a 3-2 compressor or carry-save adder (CSA) are needed, reduction tree as shown in Fig. 3 (a) [13]. But another special feature for the representation of floating-point numbers in the range from 0 to 96 is that bits 22 and 17 of the mantissa never become '1' at the same time, and with this specification, a modified 5-2 compressor can be embedded for 6 inputs illustrated in Fig.3 (b).

The control signals switch each path into the output of multiplexers. The OR gates are used, which assert the control signal to be one if all the mantissa bits of one of the operands are zero (all zeros), and these computations are done with a parallel prefix algorithm [14]. This circuit checks path 3 where the signal output of the circuit is known as a zero check (ZC). If one of the two mantissa inputs, the multiplier or the multiplicand, has at least 17 zero bits by detection of a zero value circuit, that is the same as the ZC signal circuit, the condition of being in the range from numbers 0 to 99 is correct. If the mantissa bits from position 0 to 16 are all zero and, as well as the output of the NAND gate with the two inputs in the position of bits 17 and 22 of the mantissa is '1 and the condition of being in the range of numbers 0 to 96 is correct, path 2 is selected. The signal control of this path is known as the maximum two-digit numbers check (TC) as

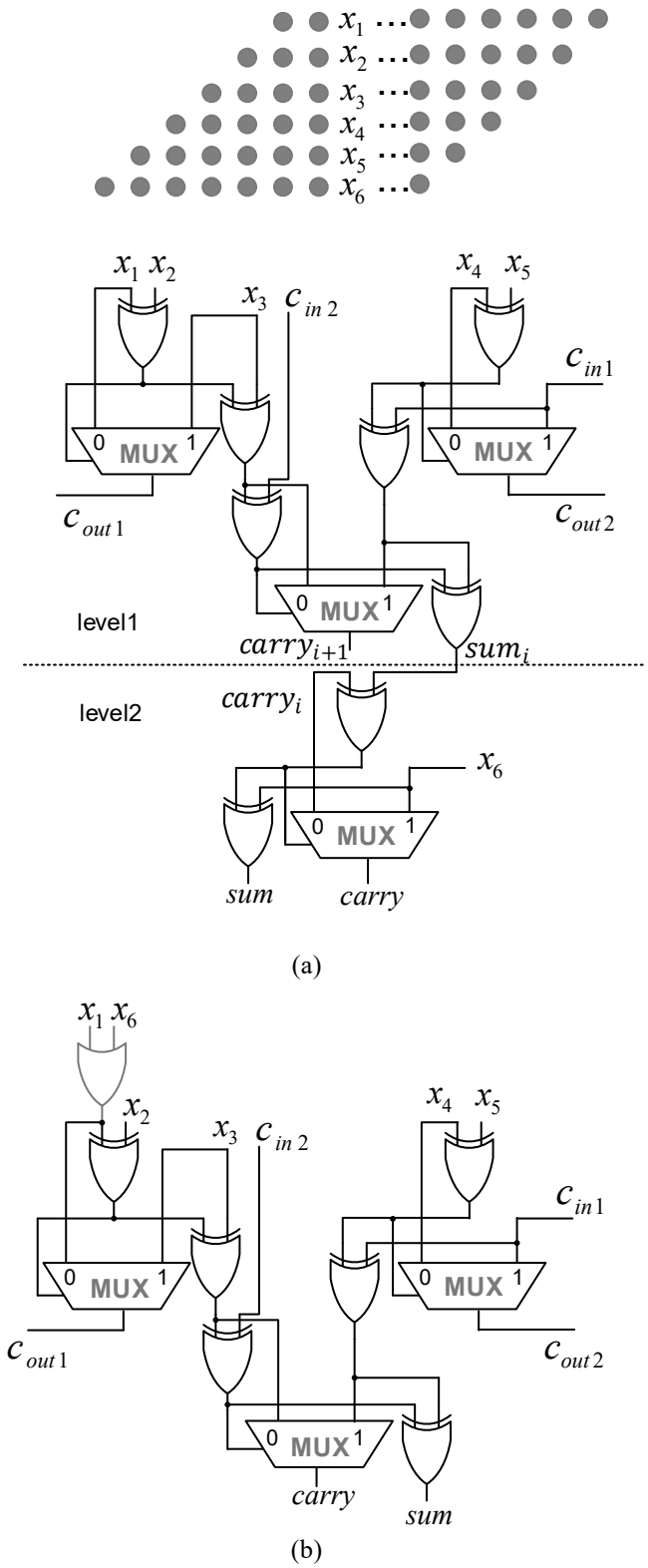

Fig. 3. Reduction tree of a 6 × 23 multiplier, (a) using the 6-2 compressor, (b) using the modified 5-2 compressor.

shown in Fig. 4. When the control signals TC or ZC are high, the path shorter than the critical path is selected.

IV. ANALYSIS AND EXPERIMENTAL RESULTS

All designs and architectures have been described in VHDL. The proposed designs are compared with approximate designs [9] and [10] using analysis and synthesis by the Synopsys design compiler for Faraday's 90nm

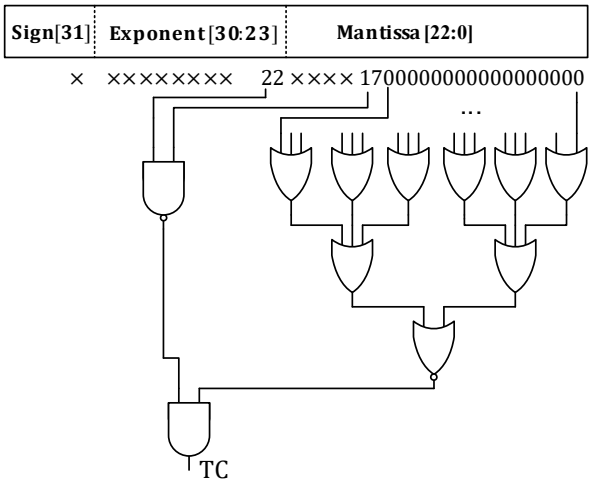

Fig. 4. The circuit structure of a TC signal.

technology library and for single-precision floating-point format. A Dadda tree multiplier is considered for the multiplication of the mantissa stage of all designs to evaluate the impact of using the proposed designs in the floating-point multipliers. Table II shows the path delay circuits and amount of delay for the mantissa processing step in three paths of the MPM using the modified 5-2 compressor. The shortest delay path of the timing channel is when the multiplicand or the multiplier is a power of 2 where all the values of the mantissa are zero and the signal ZC is high and path 3 is selected. Path 2 is to reduce the delay compared with a SPM conventional floating-point multiplier; this path is selected when the TC is enabled. Path 1 (i.e., conventional SPM floating-point multiplier) is proposed to generate the exact result. Path 1 is selected when path 2 and path 3 are unselected where the ZC and TC signals have values of '0'. In the proposed MPM floating-point multiplier, path 1 is in the critical path. However, it is expected that the utilization of alternative shorter paths with a very high probability of being selected will decrease the delay. Path delay measurement of average delay for the multiple timing channel is proposed in [14]-[17]. In the multiple timing channel circuits, the average delay is computed as:

$$T_{avg} = Pro_{critical\ path} \times T_{critical\ path} + (1 - Pro_{critical\ path}) \times T_{non-critical\ path} \quad (8)$$

where $Pro_{critical\ path}$ represents the probability of the event of the critical path and $T_{critical\ path}$ is the delay of the critical path [15].

The simulation results of the delay, area and power consumption for two of the proposed MPM floating-point multipliers using the modified 5-2 compressor and using the 6-2 compressor and two of the SPM floating-point multipliers using the modified 5-2 compressor and using the 6-2 compressor of this paper and the approximate floating-point multiplier of [9] and [10] and conventional floating-point multiplier are given in Table III. The use of the 6-2 compressor of Fig.3 (a) and (b) in the SPM floating-point multiplier results in an approximate floating-point multiplier that is comparable to the previous approximate circuits. The approximate floating-point multiplier shows a significant reduction in delay, area and power consumption compared with an exact design. The SPM floating-point multiplier using the modified 5-2 compressor and using the 6-2 compressor have better area, power and delay characteristics than the conventional floating-point multiplier, but a lower error distance is expected. The SPM floating-point multiplier using the modified 5-2 compressor and using the 6-2 compressor have 22.49% and 14.02% less power consumption in comparison to the [9]. The circuit area required by the SPM floating-point multiplier using the modified 5-2 compressor is on average 4.19%, less than the one required by the [9].

In terms of average delay, the MPM floating-point multiplier using the 6-2 compressor has a 16.6% improvement while the MPM floating-point multiplier using the modified 5-2 compressor has a 17.38% improvement, in comparison to the conventional floating-point multiplier. In terms of delay, power consumption and area, the MPM floating-point multiplier using the modified 5-2 compressor has a 2.06%, 3.49% and 4.05% improvement respectively compared to the MPM floating-point multiplier using the 6-2 compressor. This is while the 6-2 compressor covers the multiplication of a number in the range of 0 to 99 with any other number, and the modified 5-2 compressor covers the multiplication of a number in the range of 0 to 96 with any other number.

Table IV shows the power-delay product (PDP) and area-delay product (ADP) in the proposed designs and the previous approximate design, as well as the conventional floating-point multiplier. The SPM floating-point multiplier using the modified 5-2 compressor and using the 6-2 compressor shows a significant improvement in terms of PDP and ADP compared to the exact multiplier. The ADP and PDP computations for the MPM floating-point multiplier designs are based on the average delay. Ideally, the average delay is also obtained by considering the probability of critical and non-critical paths in (8). However, the average delay for the proposed MPM architectures is obtained regardless of the probabilities in Table IV. Therefore, PDP and ADP values are close to the conventional floating-point multiplier and there is no improvement in APD and PDP for these designs compared with others. Considering the values of the kernels in the maximum range of two-digit numbers, path 2 will have more probability than path 1, and it is expected that this will reduce the delay, ADP and PDP of the proposed MPM floating-point multipliers.

V. CONCLUSION

This paper proposes useful computations in floating-point multiplication considering the input data. Here, the SPM floating-point multiplier can be used in the approximate region and the MPM floating-point multiplier can be used in the exact region. The SPM or MPM floating- point multiplier uses a 6×23 reduction tree of the multiplier in two designs: 1) using the 6-2 compressors and 2) using the modified 5-2 compressor. Modification of an existing 5-2 compressor design that changes the circuit profile such that delay, area and power result decreased in comparison with the 6-2 compressor. This compressor can be suitable for kernels/filters because the coefficients of the kernel are in the range of two-digit integers, consistent with the proposed compressor. Furthermore, a single path of the SPM floating-point multiplier is separated into multiple paths of the MPM

TABLE II. PATH DELAY CIRCUITS FOR MANTISSA PROCESSING IN THREE PATHS

Path #	Path control signals	Mantissa processing	Delay (ns)
	ZC TC		
Path 1	0 0	Level 1: 5-2 compressor and CSA Level 2: 5-2 compressor Level 3: CSA Level 4: CSA Level 5: CPA (Carry Propagate Adder) Level 6: normalization	4.37
Path 2	0 1	Level 1: Modified 5-2 compressor and CSA Level 2: CPA Level 3: normalization	3.18
Path 3	1 0 1 1	Level 1: Twenty-three OR gates	2.88

TABLE III. EXPERIMENTAL RESULTS FOR FLOATING-POINT MULTIPLIERS

Floating-point multiplier	Power (µw)	Area (μm^2)	Delay (ns) (critical path)
SPM using the 6-2 compressor	2984.8	8024.3	3.06
SPM using the modified 5-2 compressor	2690.9	7181.5	2.99
[9]	3471.7	7495.8	2.71
[10]	730.86	1930.2	2.04
MPM using the 6-2 compressor	11114.6	25934.7	4.37
MPM using the modified 5-2 compressor	10819.2	25098.2	4.37
Conventional SPM	8844.0	19462.1	4.20

TABLE IV. PDP, ADP AND T_{AVG} OF THE ARCHITECTURES

Floating-point multiplier	(T_{avg}) Delay(ns)	(PDP) Power-delay-product	(ADP) Area-delay-product
SPM using the 6-2 compressor	3.06	9133.5	24554.20
SPM using the modified 5-2 compressor	2.99	8043.1	21472.53
[9]	2.71	9408.3	20238.72
[10]	2.04	1490.9	3937.62
MPM using the 6-2 compressor	3.50	38901.1	90771.52
MPM using the modified 5-2 compressor	3.47	37542.6	87090.72
Conventional SPM	4.20	37144.8	81740.49

floating-point multiplier to provide different outputs for possible input combinations. The circuit is designed with high-probability short paths along with an additional circuit for controlling the correctness of the result. Finally, the result analysis shows that the proposed design can improve the average delay of the floating-point multiplier more than previous approximate floating-point multiplier designs.

REFERENCES

[1] V. Arunachalam, and K. Sivasankaran. Microelectronic Devices, Circuits and Systems. Springer Singapore, 2021.

[2] T. D. Pham, "Kriging-Weighted Laplacian Kernels for Grayscale Image Sharpening," in IEEE Access, vol. 10, pp. 57094-57106, 2022.

[3] C. Wang, T. Li and D. Xin, "Multiple Kernel Extreme Learning Machine With Kernel Alignment Regularization," 2023 11th International Conference on Information Systems and Computing Technology (ISCTech), Qingdao, China, 2023, pp. 322-327.

[4] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," 2013 18th IEEE European Test Symposium (ETS), Avignon, France, 2013, pp. 1-6, doi: 10.1109/ETS.2013.6569370.

[5] S. Rezaei, R. Omidi, and A. Azarpeyvand, "Logarithm-approximate floating-point multiplier," Microelectronics Journal 127, 2022, p. 105521.

[6] A. Towhidy, R. Omidi, and K. Mohammadi, "On the design of iterative approximate floating-point multipliers," IEEE Transactions on Computers 72.6, 2022, pp. 1623-1635.

[7] C. Yan, X. Zhao, T. Zhang, J. Ge, C. Wang and W. Liu, "Design of High Hardware Efficiency Approximate Floating-Point FFT Processor," IEEE Transactions on Circuits and Systems I: Regular Papers 2023, pp. 4283-4294.

[8] Z. Li, Z. Lu , W. Jia, R. Yu, H. Zhang, G. Zhou, Z. Liu, and G. Qu, "Efficient Approximate Floating-Point Multiplier With Runtime Reconfigurable Frequency and Precision," IEEE Transactions on Circuits and Systems II: Express Briefs (2024).

[9] G. Di Meo, G. Saggese, A.G.M. Strollo, D. De Caro, N. Petra, "Approximate Floating-Point Multiplier based on Static Segmentation," Electronics 11, no. 19, 2022, p. 3005.

[10] L. Tegazzini, G. Di Meo, D. De Caro, and A.G. Strollo, "Design of a Hardware-Efficient Floating-Point Multiplier with Dynamic Segmentation," In 2024 19th Conference on Ph. D Research in Microelectronics and Electronics (PRIME), pp. 1-4. IEEE, 2024.

[11] M. Imani, D. Peroni and T. Rosing, "CFPU: Configurable floating point multiplier for energy-efficient computing," 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 2017, pp. 1-6.

[12] I.C. Society, IEEE Std 754TM-2008 (Revision of IEEE Std 754–1985), IEEE Standard for Floating-Point Arithmetic, vol. 2008, no. August. 2008.

[13] C. H. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits," IEEE

Trans. Circuits Syst. I, Reg. Papers, vol. 51, no. 10, pp. 1985–1997, Oct. 2004.

[14] H. Ghabeli, "Architecture Level Design of Sub-Word Multipliers for Variable-Sized and Variable-Signed Operands," 2024 20th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP). IEEE, 2024.

[15] I.-C. Lin, Y.-M. Yang, and C.-C. Lin, "High-performance low-power carry speculative addition with variable latency," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 23, no. 9, pp. 1591–1603, Sep. 2015.

[16] H. Ghabeli, A. Sabbagh Molahosseini, A.A Emrani Zarandi, and L. Sousa, "Variable latency carry speculative adders with input-based dynamic configuration," Computers & Electrical Engineering 93, (2021), p. 107247.

[17] H. Ghabeli, A. Sabbagh Molahosseini, and A.A Emrani Zarandi, "New multiply-accumulate circuits based on variable latency speculative architectures with asynchronous data paths," Majlesi Journal of Electrical Engineering 16, no. 2 (2022), pp. 41-53.