*Sensors and Materials*, Vol. 36, No. 10 (2024) 4269–4282
MYU Tokyo



S & M 3799

# Digital Image Processing under Modified Core Function Based on Residue Number System

Teh-Lu Liao,[1] Yi-You Hou,[2*] Chi-Chun Fang,[1] and Cheng-Yi Chen[3**]

[1]Department of Engineering Science, National Cheng Kung University,
No. 1, University Road, Tainan City 701, Taiwan
[2]Department of Intelligent Commerce, National Kaohsiung University of Science and Technology,
No. 58, Shenzhong Rd., Yanchao Dist., Kaohsiung City 824, Taiwan
[3]Department of Electrical Engineering, Cheng Shiu University,
No. 840, Chengqing Rd., Niaosong Dist., Kaohsiung City 833, Taiwan

(Received May 14, 2024; accepted September 18, 2024)

***Keywords:*** residue number system (RNS), modified core function (MCF), image enhancement, cryptography

In this study, we developed a residue number system (RNS), a numeral system consisting of an arbitrary number of pairwise coprime integers representing a specific integer by its value. We discovered that this system reduced the runtime or training speed of image processing. In addition, we observed that the characteristic operations of the core functions of the decentralized congruential system derived from the proposed system reduced the overall execution time or training speed of image processing. Both the speed of calculation and the properties of the congruential system ensured the accuracy and security of information after distributed processing. Overall, this approach enabled the use of different algorithms on microdevices while ensuring confidentiality. In summary, we developed a system capable of increasing the operation speed of image processing by 50% through core function precomputation, with a modified image data input, a modified core function, and a reverse core function.

## 1. Introduction

Over the past decade, public-key cryptosystems installed on hardware devices have been widely examined.[1–3] Various approaches have been proposed to accelerate the implementation of the residue number system (RNS). To achieve various deciphering goals, a well-known solution has been used to independently conduct calculations for process variables and reconstruct the final results through the Chinese remainder theorem (CRT).[4] To the best of our knowledge, these efforts represent the first attempt to apply CRT in the development of RNSs. In previous solutions,[5–7] RNSs have been used with Montgomery multiplication for calculation. Although this approach facilitates the full implementation of an RNS, the message delivered is never regarded as a binary number. Therefore, messages should be calculated as a whole and cannot be separately computed. An RNS is a number representation system based on modular

*Corresponding author: e-mail: yyhou@nkust.edu.tw
**Corresponding author: e-mail: k0464@gcloud.csu.edu.tw
https://doi.org/10.18494/SAM5156

ISSN 0914-4935 
https://myukk.org/

arithmetic. This system converts large numbers into several smaller numbers, which represent the remainders when the large number is divided by different moduli. This technique is efficient in computer arithmetic because it enables large number computations to be broken down into independent small number operations, thereby increasing computational speed and reducing complexity. In an RNS, modified core functions are used to optimize calculations and data processing, particularly to enhance and accelerate key operations such as addition, subtraction, multiplication, and division within the RNS framework. When the enhanced computational efficiency and data processing capabilities of RNS and modified core function (MCF) are integrated, more efficient and accurate results can be obtained from the processing and analysis of multivariate data.

In recent years, with the development of science and technology, an increasing trend has been observed in the utilization of digital image processing technology in daily activities, such as in facial recognition for access control and in the automatic purchase of food ingredients by smart refrigerators. This degree of convenience has now become an entrenched norm in everyday life. Image processing has spurred many advances in science and technology. In particular, new technologies, such as image processing, often drive synthesis between domains and between cutting-edge and legacy solutions. Software and hardware are tightly intertwined in image analysis, image transmission, and image recording, and image preprocessing has undergone major advances with improvements in deep learning. Consequently, several preprocessing techniques, such as blurring,[8] filtering,[9,10] smoothing,[11] padding,[12] and sharpening,[13,14] have been developed. In addition, with the exponential growth observed in the number of pixels, particularly from 800 × 600 to 1024 × 768, 1920 × 1080, and 4096 × 2160 (i.e., 4K), an increase has also been observed on the load placed on the size of the data itself, with the ability of processors to handle the computational workload playing a key role. Therefore, further research is required to increase the speed of basic digital computation. Technologies such as unmanned driving and computer vision are gradually transforming everyday life, with the problem of high resource consumption, as is the case in artificial intelligence and computer vision, being a key topic in this area. An RNS is a system that uses distributed processing to reduce the overall computational time or training speed of image processing or deep learning, ensuring the accuracy of image processing and processed information.

During the training and execution processes in computer vision, various image processing techniques are typically used. These techniques are computationally expensive and require advanced graphic processing units (GPUs). In addition, the central processing unit should be adapted to support image processing in case the GPU fails.

In this study, we utilized the simplified functions of an RNS for operators in various types of standard algorithm to increase the speed of image processing. To examine the function of hardware-accelerated image processing, we explored various widely used image processing technologies. According to the literature, edge detection, smoothing, and sharpening are the most commonly used techniques for image processing. In this study, we examined the effectiveness of Laplacian operators in edge detection,[15] Sobel operators in sharpening, and Gaussian filters in smoothing.

Although RNSs can be traced back to 496 BC in China, these residual systems did not attract the interest of computer engineers until the 1950s, when their technical applications were

gradually expanded into cryptography, network biosecurity, machine learning, signal processing, and fault-tolerant signal processing. Previous studies have explored RNSs as systems that use a few bits to represent a larger number of bits. These residual number representation systems can considerably shorten the execution speed of certain linear algorithms in signal processing. Therefore, they have been extensively used as a research tool in examining extreme learning machine algorithms. In this study, we examined the correctness of the homomorphism of RNSs in the presence of various kernel operators of image processing.

## 2. Materials and Methods

In this study, we examined methods for increasing the computational speed of image processing and calculating the modulus of the product of two separate numbers. We utilized a commonly used equation of fundamental computation in digital image processing. This calculation process has four elements: core function precomputation, a modified image data input, a modified core function, and a reverse core function.

Core function precomputation is used to provide a basis for a modified core function. This step involves calculating frequently used constants that can be calculated once the channels of the RNS are set. These constants can also be used as a hybrid template for many RNSs to operate, thereby minimizing the RNS' memory and space requirements. A modified image input is treated by merging the three elements of RGB colors in a binary power form to offer low memory and space requirements and a certain degree of confidentiality. A modified core function is used to optimize the algorithm. Many channels are used to minimize the data and increase the speed of repetitive arithmetic. A reverse core function is used to transform the RNS expression into a common decimal expression, enabling the retrieval of results for image enhancement.[16]

### 2.1 RNS and core function

In this study, a modified technique was used to shorten the time required to calculate the modulus of the product of two numbers through hardware. The following equation was used to calculate this modulus:[17]

$$X \,(\mathrm{mod}\, M) = X - Y \times M, \tag{1}$$

where $M$ is the modulus of the product of two numbers. RNSs comprise a set of co-prime moduli.[18] In an RNS, the residue of a nonnegative integer $X$ is represented in $N$ channels, where $N$ is the number of moduli $M$ in the system. Each number has a unique representation, with the range of the system being $0 \leq X \leq D$, where $D$ is the product of all channel moduli, also referred to as the dynamic range of the system. Each residue represented in each channel is related only to its own channel modulus. In other words, unlike in traditional decimal numerical systems, in which each integer is weighted, integers in RNSs are not weighted. Therefore, they can be randomly scrambled and still yield the same results as long as the residue and channels are not

mixed. In an RNS, arithmetic homomorphism comprises three processes: addition, subtraction, and multiplication (no division).

Given that the entire operation involves a transformation from a decimal or binary system to an RNS and back to a decimal or binary system, two methods can be used to calculate this transformation. In the first method, the respective residue of the nonnegative integer in each channel is simply selected. In the second method, a technique referred to as CRT is used.[19] This approach facilitates the establishment of an RNS as follows:

$$\begin{cases} X \equiv a_1 (\bmod m_1) \\ X \equiv a_2 (\bmod m_2) \\ \quad \cdots \\ X \equiv a_n (\bmod m_n) \end{cases} M_i = \frac{M}{m_i} t_i M_i \equiv 1 (\bmod m_i),$$
$$X = a_1 t_1 M_1 + a_2 t_2 M_2 + \cdots + a_n t_n M_n + kM \bmod M = \sum_{i=1}^{n} a_i t_i M_i \bmod M, \tag{2}$$

where $a_n$ is the residue of the nonnegative integer $X$ in moduli $m_n$ in each channel and $M$ is the product of all moduli. CRT can be used to easily convert between a decimal or binary system and an RNS.

### 2.2 Core function and precomputation[20]

The following is the core function of an RNS system:[21]

$$C(X) = \sum_{i=0}^{n} w_i \left\lfloor \frac{X}{m_i} \right\rfloor = \sum_{i=0}^{n} (X - x_i) \frac{w_i}{m_i} = X \sum_{i=0}^{n} \frac{w_i}{m_i} - \sum_{i=0}^{n} \frac{x_i w_i}{m_i}, \tag{3}$$

where $x_i$ is the channel residue of $X$ through moduli $m_i$ and $w_i$ represents arbitrary constants referred to as the channel weights of the system. Substituting $M$ into the aforementioned function, the product of the moduli, yields the following:

$$C(M) = \sum_{i=0}^{n} w_i \left\lfloor \frac{M}{m_i} \right\rfloor = \sum_{i=0}^{n} (M - x_i) \frac{w_i}{m_i}. \tag{4}$$

Dividing each side by $M$, the dynamic range of the system, yields the following:

$$\frac{C(M)}{M} = \sum_{i=0}^{n} \frac{w_i}{m_i}. \tag{5}$$

Substituting Eq. (5) into Eq. (3) yields the following common expression for the core function:

$$C(X) = \frac{C(M)}{M} X - \sum_{i=0}^{n} \frac{x_i w_i}{m_i}. \tag{6}$$

This equation is commonly used to describe core functions. Its function is similar to that of a linear plot of $C(X)$ against $X$ with a slope of $C(M)/M$.[22] However, its plot is not perfectly linear because of the noise at the end of the equation, which arises because $w_i$ is not a single independent constant. To improve readability, this noise can be neglected as follows:

$$C(X) = Mag(X) - Noise(X). \tag{7}$$

These results indicate that $Mag(X) = \frac{C(M)}{M}X$ and $Noise(X) = \sum_{i=0}^{n} \frac{x_i w_i}{m_i}$. If noise is assumed to be negligible, then

$$Q = \frac{X}{M} = \frac{Mag(X)}{Mag(M)} = \frac{C(X) + Noise(X)}{C(M) + Noise(M)} \approx \frac{C(X)}{C(M)}. \tag{8}$$

Therefore, using our core function and neglecting noise, we can ignore the quotient $Q$ of the two nonnegative integers $Q$ and $M$, which are presumably very large. Instead, we can calculate the quotient of their counterparts $C(X)$ and $C(M)$ with our own number scale and obtain similar results.

In the modified version of the core function, many calculations involving this function incorporate values that can be precomputed. These values remain constant provided that the number and values of all channel moduli do not change during the calculation process. These precomputed results serve as a look-up table, simplifying the calculation process by avoiding complex modulo operations and division. These values are precomputed as follows:

$$Channel\ weight\ ratio = \frac{w_i}{m_i}, \tag{9}$$

$$C(B_i) = \frac{C(M)\left|M_i^{-1}\right|m_i - w_i}{m_i}, \tag{10}$$

$$deno = \left| -\left\lfloor \frac{1}{Mag(P)} \right\rfloor * M \right|_{P_i}, \text{ and} \tag{11}$$

$$denoB = \left| -\left\lfloor \frac{1}{L * Mag(P)} \right\rfloor * M \right|_{P_i}, \tag{12}$$

where $B_i = M_i \left|M_i^{-1}\right|_{m_i}$ and $L$ is the mean of the bitwise length of $Mag(X)$ and $\left\lfloor \frac{1}{Mag(P)} \right\rfloor$.

The value obtained for $L$ primarily depends on the assumption that $C(M) = 2^h < P$.[23]

$$Y = \left\lfloor \frac{X}{P} \right\rfloor = \left\lfloor \frac{Mag(X)}{Mag(P)} \right\rfloor \approx \lfloor Mag(X) \rfloor * \left\lfloor \frac{1}{Mag(P)} \right\rfloor, \tag{13}$$

with a maximum error of $\left\lceil Mag(X)+\frac{1}{Mag(P)}\right\rceil$. If $L$ is added as a representative of the mean of the bitwise length of $\lfloor Mag(X)\rfloor$ and $\left\lfloor \frac{1}{Mag(P)}\right\rfloor$, then the equation becomes

$$Y=\left\lfloor \frac{L*Mag(X)}{L*Mag(P)}\right\rfloor \approx \lfloor L*Mag(X)\rfloor * \left\lfloor \frac{1}{L*Mag(P)}\right\rfloor \tag{14}$$

with a maximum error of $\left\lceil L*Mag(X)+\frac{1}{L*Mag(P)}\right\rceil$, which is lower than the aforementioned error.

### 2.3 Modification of image RGB values by RNS

In this section, we introduce a novel technique for modifying the data fed to a core function system, specifically with RGB values. Convolution[24] is central to many deep learning algorithms and is constituted by addition and multiplication operations, and occasionally subtraction if the kernel requires negative integers. An RNS is regarded as suitable for convolution if division is not required. Given the ability of RNSs to handle large integers within a dynamic range, our novel technique merges all three RGB values into a single integer to save transfer time and presumably increase the system's efficiency without corrupting individual RGB values. In this study, after the pixels were spilled up and treated, the results were combined into a set (Fig. 1).

To establish a combined set $B(p)$, the following steps must be followed:

Step 1. Divide the image data into pixels.

Step 2. Convert each RGB value into a binary form.

Step 3. Add a number of zeros in between each value to use as a buffer and avoid corrupting other values.

Step 4. Combine each treated pixel $B(p_n)$ into set $B(p)$.

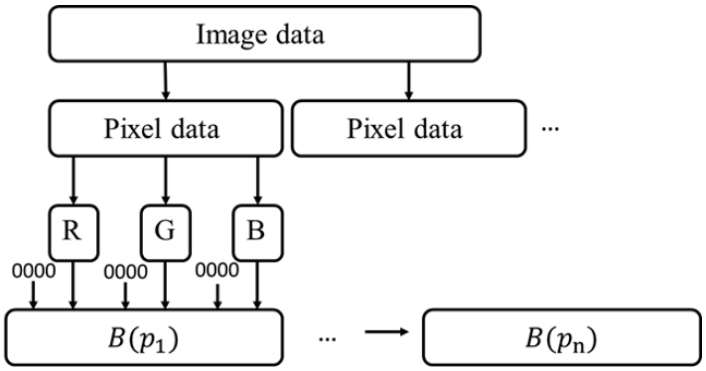



Fig. 1. Structure of the El-Gamal hybrid function.

In Step 3, the number of zeros can be adjusted depending on the kernel size and kernel values used in the convolution of the deep learning algorithm. However, the three sets of zeros must be the same for clarity.

### 2.4 FPGA circuit implementation

As described in this section, we integrated a modified core function into a field-programmable gate array (FPGA), an integrated circuit containing an array of programmed or reprogrammed logic blocks to enable flexible and reconfigurable computation as in computer software. In addition, an intellectual property (IP) core is designed using Vivado high-level synthesis. Units such as a subtractor, an adder, and a multiplier are integrated into the production unit. Python Productivity for Zynq (PYNQ) is a developmental version of the ZYNQ architecture that supports Python. In this study, PYNQ-Z2 was used as a developmental framework for PYNQ (Fig. 2). PYNQ is characterized by its built-in interactive logic that enables compatibility with ARM processors and FPGAs. Users can directly edit programs, and programmable logic circuits can be used as hardware libraries to provide hardware acceleration.

In this study, an FPGA was used to implement the proposed modified core function and the modified digital image data through two image processing techniques. These techniques relied on two distinct kernels for convolution. Figures 3 and 4 depict the flow chart and wiring diagram of the FPGA, respectively. The first step in the process was to establish channels, including the setup of $p_i$, $h$, and $M$. Once these values were established, precomputation was implemented, yielding four constants essential for subsequent calculations in a fixed modulus scenario. Following this initial setup, digital image data were input to implement the modified image data method. Unlike one-time precomputation, this process has to be repeated with each new image input. Subsequently, the modified core function utilized the precomputed values of deno and deno_B, and processed the image data in accordance with the method outlined for the modified image data in an RNS. To access the processed results, the RNS was transformed into a binary or decimal system, and the modifications applied in the image data method were undone.

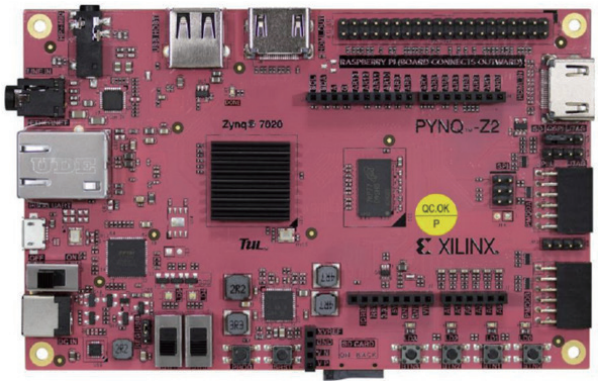



Fig. 2. (Color online) PYNQ-Z2.

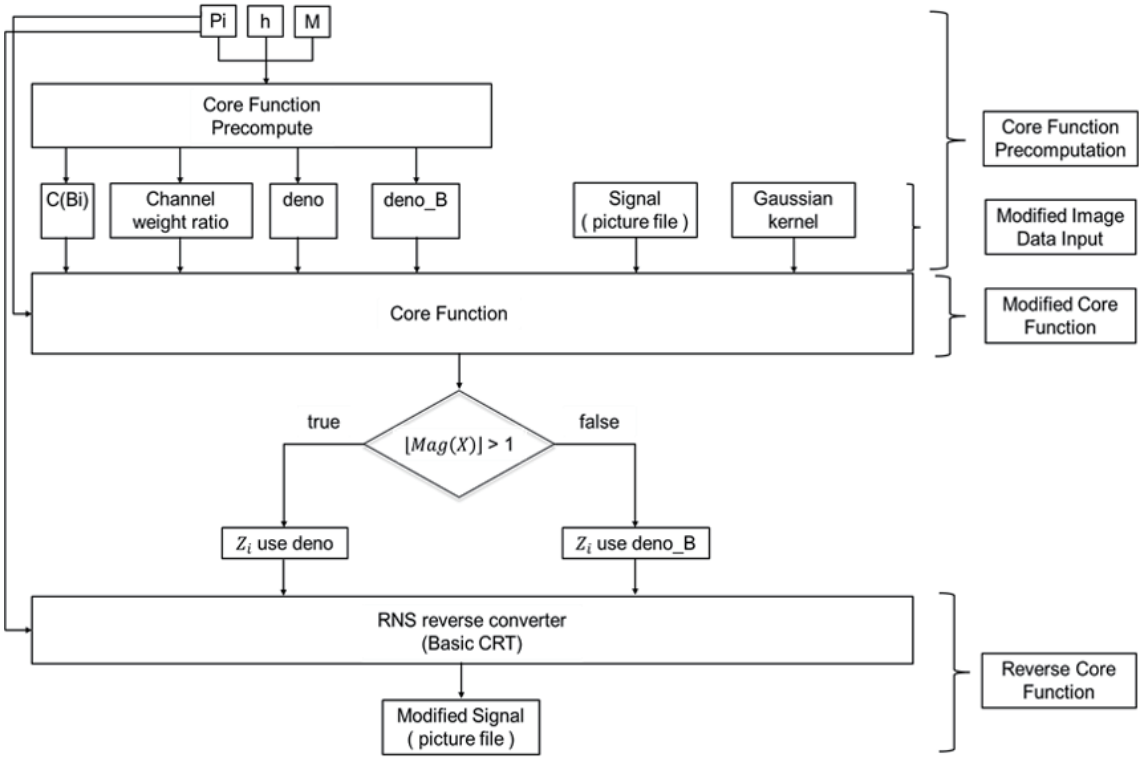



Fig. 3. Flow chart of an FPGA.

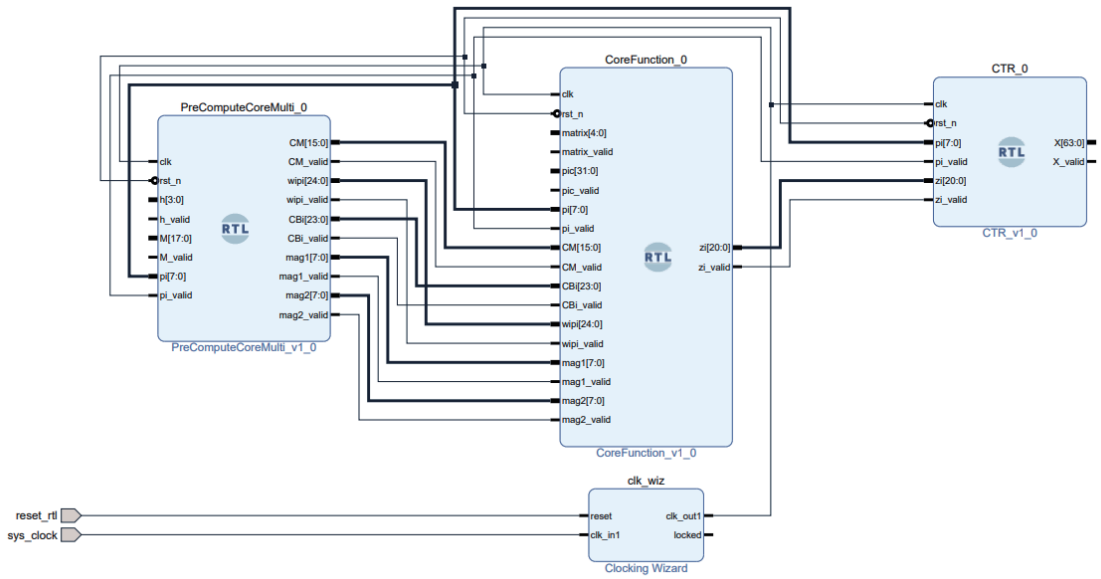



Fig. 4. (Color online) Wiring diagram of an FPGA.

## 3. Experimental Results

In this section, we present the outcomes of the modified binary overlay method for image data. Figure 5 depicts the red component of the first 1000 RGB pixels in the original image. The values of the pixels range from 0 to 255, potentially exposing them to unauthorized decryption. Figure 6 illustrates the data processed using the modified binary overlay method, indicating that the values have become indecipherable.

To demonstrate the amount of chaos introduced by the modified binary overlay method in image data processing, we attempted to reverse the image to its original RGB format and recreate it by using brute force. This process yielded Figs. 7–9, none of which could be recognized by conventional means. This approach can be used to encrypt images.

In this study, we implemented modified image processing through two distinct image enhancement techniques: Laplacian sharpening[25] and Gaussian blur.[26] These techniques are integrated into a modified core function that serves as both a speed booster and a confidentiality tool. The performance results for these techniques are illustrated in Figs. 10–13.

As shown in Figs. 12 and 13, we calculated the increase in operation speed using the following formula:

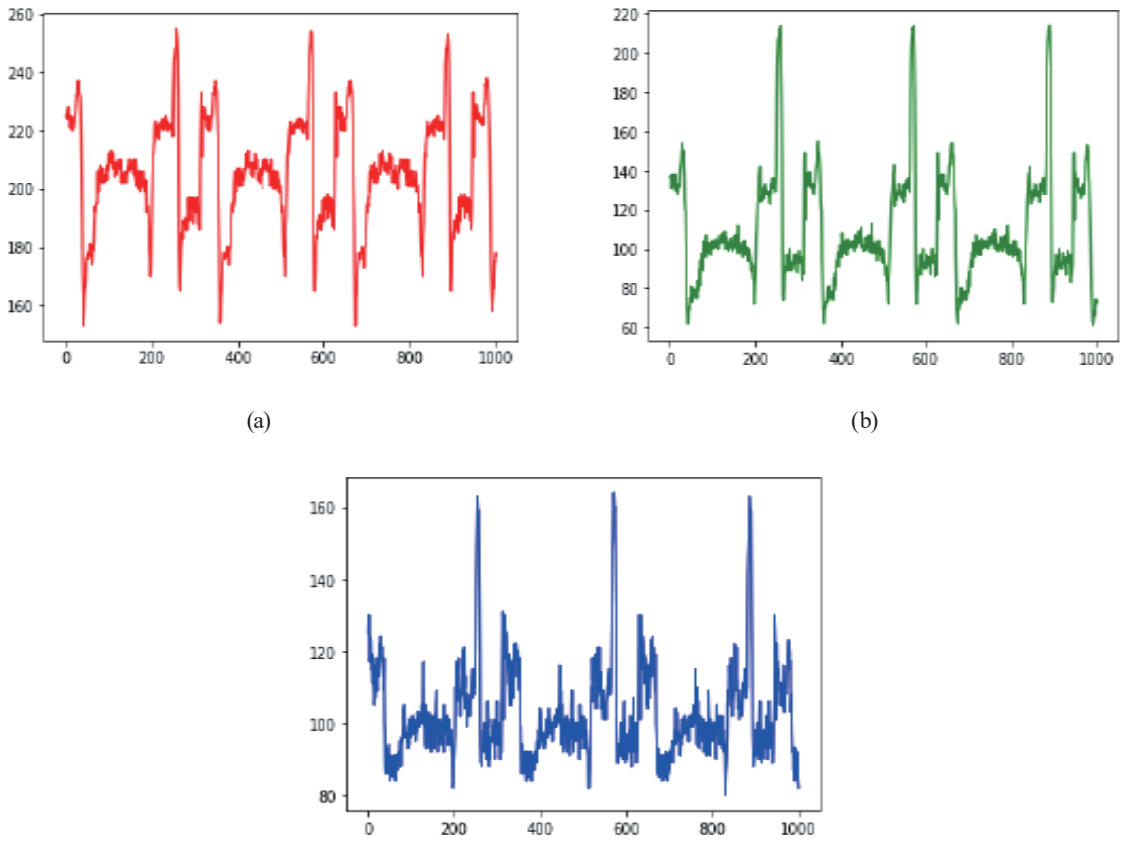



Fig. 5. (Color online) Original distribution of RGB elements (1000 pixels). (a) Red, (b) green, and (c) blue.

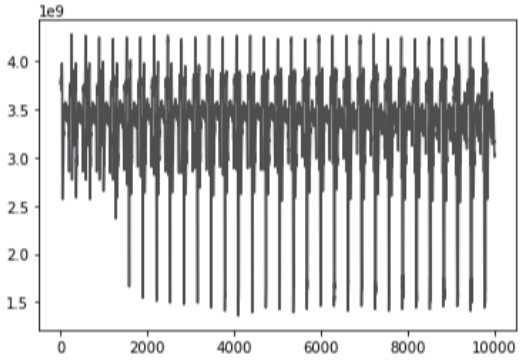



Fig. 6. Modified distribution of RGB elements (10000 pixels).

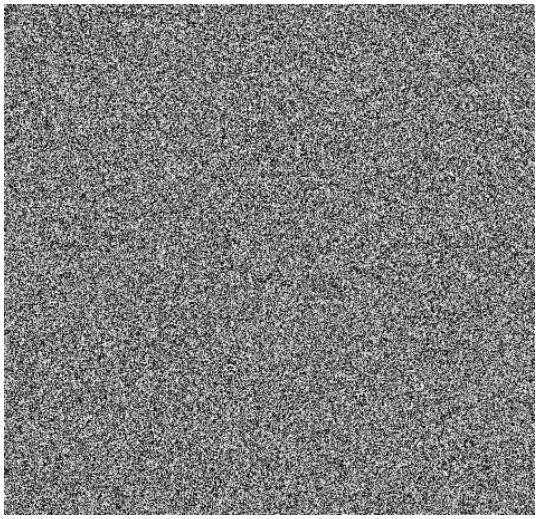


Fig. 7. Reversed image (red) through brute force.

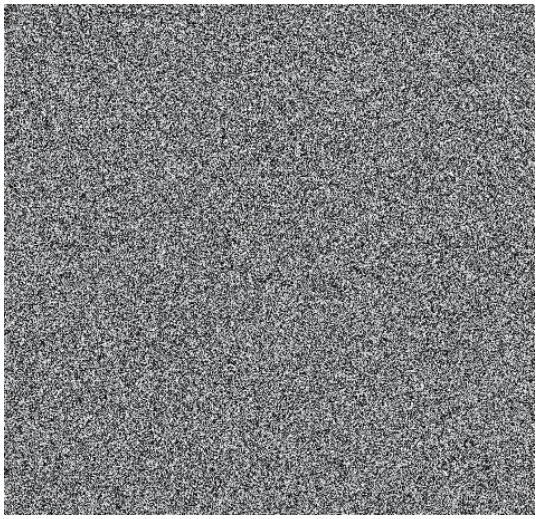


Fig. 8. Reversed image (green) through brute force.

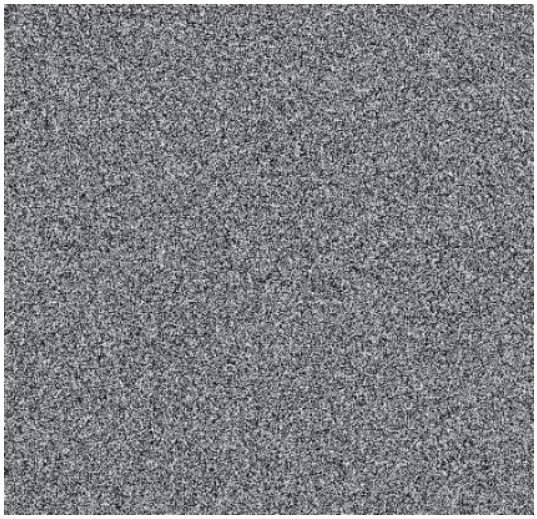
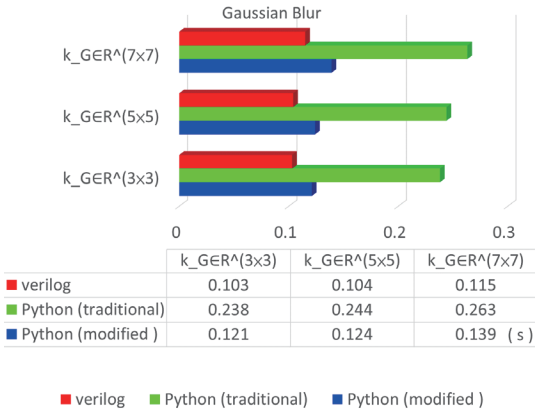
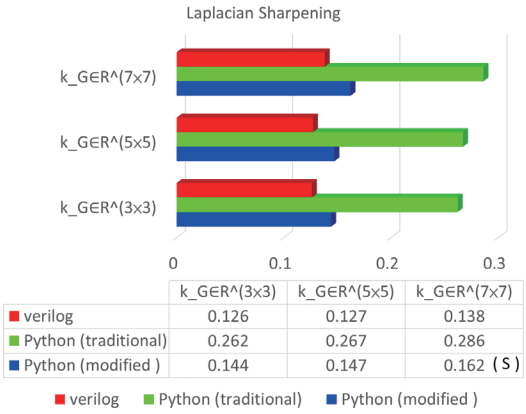


Fig. 9. Reversed image (blue) through brute force.



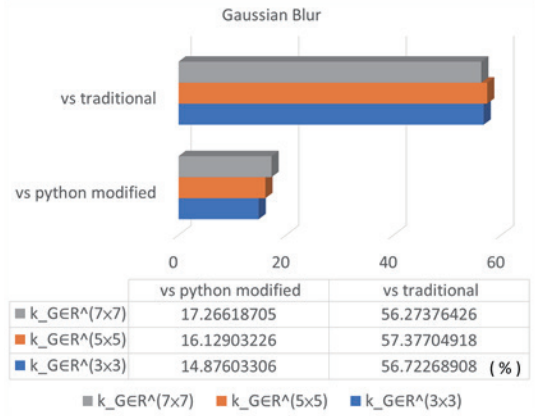

| | k_GER^(3x3) | k_GER^(5x5) | k_GER^(7x7) |
|---|---|---|---|
| verilog | 0.103 | 0.104 | 0.115 |
| Python (traditional) | 0.238 | 0.244 | 0.263 |
| Python (modified ) | 0.121 | 0.124 | 0.139 ( s ) |

Fig. 10. (Color online) Gaussian blur comparison (operation time).



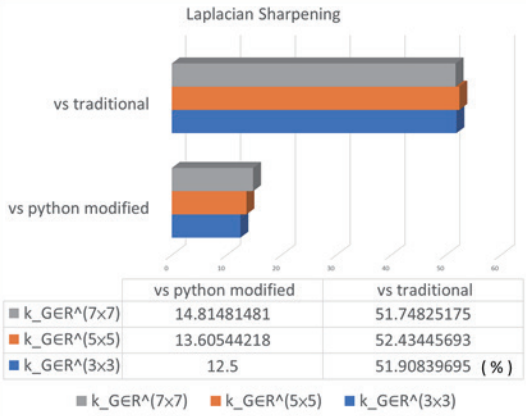

| | k_GER^(3x3) | k_GER^(5x5) | k_GER^(7x7) |
|---|---|---|---|
| verilog | 0.126 | 0.127 | 0.138 |
| Python (traditional) | 0.262 | 0.267 | 0.286 |
| Python (modified ) | 0.144 | 0.147 | 0.162 ( S ) |

Fig. 11. (Color online) Laplacian sharpening comparison (operation time).




| | vs python modified | vs traditional |
|---|---|---|
| k_GER^(7x7) | 17.26618705 | 56.27376426 |
| k_GER^(5x5) | 16.12903226 | 57.37704918 |
| k_GER^(3x3) | 14.87603306 | 56.72268908 ( % ) |

Fig. 12. (Color online) Gaussian blur comparison (operation speed increase).




| | vs python modified | vs traditional |
|---|---|---|
| k_GER^(7x7) | 14.81481481 | 51.74825175 |
| k_GER^(5x5) | 13.60544218 | 52.43445693 |
| k_GER^(3x3) | 12.5 | 51.90839695 ( % ) |

Fig. 13. (Color online) Laplacian sharpening comparison (operation speed increase).

$$\frac{\left(original\ operation\ time - new\ operation\ time\right)}{original\ operation\ time}. \tag{15}$$

Given the importance of the modified core function, by reducing the values of data through multiple channels, we considerably increased the speed of repetitive arithmetic operations. Subsequently, we used the reversed core function to transform the RNS expression into a decimal expression, thereby achieving image enhancement. Figures 14–16 depict the differences between the numbers of pixels obtained using the conventional method and the modified method for image enhancement, indicating a maximum difference or error of ±3, which is considered acceptable.

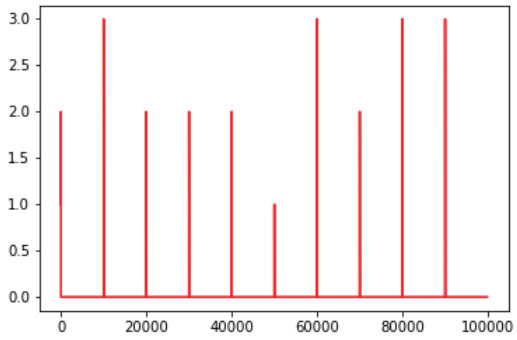



Fig. 14. (Color online) Error difference (pixels, red).

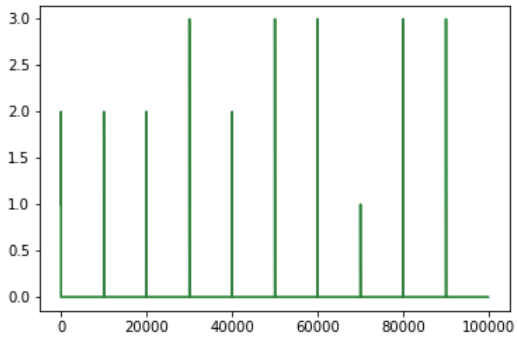



Fig. 15. (Color online) Error difference (pixels, green).

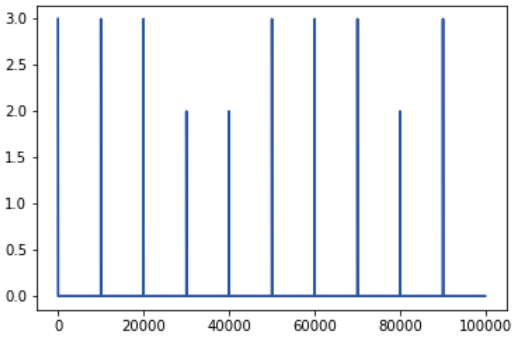



Fig. 16. (Color online) Error difference (pixels, blue).

## 4. Conclusions

In this study, we established a digital image processing framework by using a modified core function based on an RNS. By implementing modified algorithms, we achieved major improvements in Gaussian blur and Laplacian sharpening, thereby increasing operation speed by at least 50%. In addition, we increased the speed of the core function through precomputation to establish a modified core function and applied image data modified binary overlay to this function. In evaluation experiments, we applied our modified core function in image enhancement with image data modification in an FPGA.

## Acknowledgments

This work was financially supported by the National Science and Technology Council, Taiwan, under grants 111-2221-E-006-216 and 111-2218-E-006-009-MBK.

## References

1 E. F. Brickell: Proc. CRYPTO1989 (1990) 368–370.
2 M. Shand and J. Vuillemin: Proc. 11th IEEE Symp. Computer Arithmetic (1993) 252–259.
3 S. E. Eldridge and C. D. Walter: IEEE Trans. Comput. **42** (1993) 693. https://doi.org/10.1109/12.277287

4 J. J. Quisquater and C. Couvreur: Electron. Lett. **18** (1982) 905. https://doi.org/10.1049/el:19820617
5 K. C. Posch and R. Posch: IEEE Trans. Parallel Distrib. Syst. **6** (1995) 449. https://doi.org/10.1109/71.382314
6 S. Kawamura, M. Koike, F. Sano, and A. Shimbo: Advances in Cryptology- EUROCRYPT (2000) 523. https://doi.org/10.1007/3-540-45539-6_37
7 H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura: Proc. Cryptographic Hardware and Embedded Systems (2001) 364–376.
8 T. Indriyani, M. I. Utoyo, and R. Rulaniingtyas: J. Phys.: Conf. Ser. **1477** (2020) 2755. https://doi.org/10.1088/1742-6596/1477/5/052056
9 D. Duce, I. Herman, and B. Hopgood: Comput. Graphics Forum **21** (2002) 43. https://doi.org/10.1111/1467-8659.00565
10 P. Milanfar: IEEE Signal Process Mag. **30** (2013) 106. https://doi.org/10.1109/MSP.2011.2179329
11 J. R. Jensen and K. Lulla: Geocarto Int. **2** (1986) 65. https://doi.org/10.1080/10106048709354084
12 A. D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee: Proc. 2019 IEEE Int. Conf. Image Process. (ICIP, 2019) 4275–4279.
13 Y. Yongbin, Y. Nijing, Y. Chenyu, and N. Tashi: J. Syst. Eng. Electron. **30** (2019) 448. https://doi.org/10.21629/JSEE.2019.03.02
14 M. R. Metwalli, A. H. Nasr, O. S. Farag Allah and S. El-Rabaie: Proc. 2009 Int. Conf. Computer Engineering & Systems (2009) 63–70.
15 H. L. Garner: IRE Trans. Electron. Comput. **EC-8** (1959) 140. https://doi.org/10.1109/TEC.1959.5219515
16 L. Hong, Y. Wan, and, A. Jain: IEEE Trans. Pattern Anal. Mach. Intell. **20** (1998) 777. https://doi.org/10.1109/34.709565
17 A. F. Tenca and C. K. Koc: IEEE Trans. Comput. **52** (2003) 1215. https://doi.org/10.1109/TC.2003.1228516
18 F. J. Taylo: Computer **17** (1984) 50. https://doi.org/10.1109/MC.1984.1659138
19 Thu Van Vu: IEEE Trans. Comput. C-34 (1985) 646. https://doi.org/10.1109/TC.1985.1676602
20 Y. Kong, S. Asif, and M. A. U. Khan: Appl. Algebra Eng. Commun. Comput. **27** (2015) 1. https://doi.org/10.1007/s00200-015-0268-1
21 J. Gonnella: IEEE Trans. Signal Process. **39** (1991) 69. https://doi.org/10.1109/78.80766
22 N. Burgess: Proc. 16th IEEE Symp. Computer Arithmetic (2003) 262–269.
23 D. D. Miller, J. N. Polky, J. R. King: Proc. 26th Midwest Symp. Circuits and Systems (1986) 385–389.
24 Q. Li, Q. Wang, and X. Li: IEEE Trans. Geosci. Remote Sens. **59** (2021) 8693. https://doi.org/10.1109/TGRS.2020.3047363
25 D. F. Shen, C. W. Chiu, and P. J. Huang: Proc. 2006 IEEE Int Conf. Multimedia and Expo (2006) 457–460.
26 H. Kong, H. C. Akakin, and S. E. Sarma: IEEE Trans. Cybern. **43** (2013) 1719. https://doi.org/10.1109/TSMCB.2012.2228639


## About the Authors

Teh-Lu Liao (Member, IEEE) received his B.S. degree (Hons.) from the Department of Engineering Science, National Cheng Kung University (NCKU), Taiwan, in 1982, and his M.S. and Ph.D. degrees (Hons.) from the Department of Electrical Engineering, National Taiwan University, Taiwan, in 1984 and 1991, respectively. He joined the Department of Engineering Science, NCKU, as an associate professor in 1991 and was promoted to full professor in 1999. His research interests include control theory, multi-agent systems, signal and image processing, chaos communication systems, and cryptosystems. He is a member of the IEEE and a Fellow of the Chinese Automatic Control Society, Taiwan. (tlliao@ncku.edu.tw)

Yi-You Hou received his Ph.D. degree from the Department of Engineering Science, National Cheng Kung University, Taiwan, in 2008. At present, he is an associate professor in the Department of Intelligent Commerce, National Kaohsiung University of Science and Technology, Taiwan. His main research interests are in system control and biomedical engineering. (yyhou@nkust.edu.tw)

Chi-Chun Fang received his E.E. degree from National Tsing Hua University, Taiwan, in 2020 and his M.S. degree from National Cheng Kung University, Taiwan, in 2022. (kennieg4a@gmail.com)

Cheng-Yi Chen received his M.S. and Ph.D. degrees from the Department of Mechanical Engineering of National Sun Yat-Sen University, Taiwan, in 1994 and 1999, respectively. From 1999 to 2001, he was an associate professor of the Department of Automation Engineering of Kao Yuan Institute of Technology. From 2001 to 2009, he was an associate professor in the Department of Electrical Engineering of Cheng Shiu University. Since July 2009, he has been a professor in Cheng Shiu University. His research interests are in nonlinear control, automation control, motion control, embedded control system design, and smart systems. (k0464@gcloud.csu.edu.tw)