AN-HRNS: AN-Coded Hierarchical Residue Number System for Reliable Neural Network Accelerators

*Ｗan-Ju Huang, Hsiao-Wen Fu and Tsung-Chu Huang*

Department of Electronics Engineering, National Changhua University of Education, Changhua, Taiwan

*Abstract—Acceleration, power reduction, reliability and low-overhead are four major goals of neural networks. Residue number systems (RNS) intuitionally can improve all of the goals simultaneously. Especially for reliability-critical applications, the generic ability of modularization empowers the redundant RNS (RRNS) fault tolerance. However, the RRNS still suffers four critical issues including dynamic range (DR), sign detection, limited efficient moduli, and redundancy-decoding complexity. In this paper, we apply the piecewise linear units to limit the DR of next layer, stored sign-and-parity bits to make sign detection efficient, the hierarchical structure to make the best of efficient moduli, and the AN-codes for both sub-RNS error correction and MMR checking. An expanded residue-to-binary converter is developed for highly reduction of the whole multiple-module redundancy decoder. This is the first paper to incorporate AN codes to the RRNS applied in highly-reliable neural networks. From our evaluation under approximately equal bit-length L of moduli in a DR of hkL bits, the g=(hk+2)(hk+1)/2 residue-to-binary converters and g(g-1)/2 comparators in the conventional RRNS can be reduced to an expanded residue-to-binary converter with only k decoders. From the BLER analyses, more than 126 times of MTBF can be achieved.*

*Keywords: Neural network acceleration, redundant residue number system, AN codes, multiple modular redundancy.*

# Introduction

Neural networks (NN) have played a critical role in Artificial-Intelligence (AI) [1]. For operating NNs, modern processors including graphic process units (GPU) are usually equipped with a bunch of multiply-add accumulator (MAC) and fused multiply-add (FMA) cores for acceleration [2]. For most AI applications the learning often requires huge computations for huge datasets, therefore, the NNs suffer four major issues: acceleration, power consumption, area overhead and reliability. Especially in automotive, medical electronics, and many reliability-critical applications, the reliability has become the most critical issue. Even though input-side layers of NNs usually possess self-healing ability, the last layers still suffer logic errors, which are critical for the final decision.

Residue number systems (RNS) has been shown effective and efficient for tackling these issues owing to the generic ability of partitioning operators without carries. Basically, a multiplier in a RNS with *k* moduli can be improved by ***O***(*k*) times in all acceleration, power reduction and area saving [3-5] if neglecting issues about (1) dynamic range (DR), (2) sign detection, and (3) limited efficient moduli. RNS is also suitable for multiple module redundancy (MMR) due to its disjoint modularization. In Fig.1, a redundant RNS (RRNS), *r* redundant moduli are usually added to the *k* moduli, then the MMR can be built by comparing a pair of outputs converted by *k*-combinations out of *s*=*k*+*r* moduli.



Fig. 1 A huge arbitrating decoder for Conventional MMR for a RRNS.

For convenience to explain the arbitrator of RBCs, Fig.2 shows the conventional design for MMR architecture, where the blocks BRC and RBC are separately the binary-residue and residue-binary converters. For simplicity without loss of generality, only one layer is presented. Blocks *%m* denote the modulo-*m* residue generators, which may be integrated with the former blocks. To ensure that at least two correct sets can decide the output in the single module error (SME) model, . This results in . The comparisons can be implemented iteratively or in parallel. For iterative checking using one RBC, the time complexity needs *k* combinations out of *k*+2moduli [6][7], that will crucially impact the acceleration. While in parallel checking schemes, conventional parallel MMR will suffer sets of *k*-moduli RBCs with comparators [8]. Conversion overhead finally becomes the fourth issue of the RRNS.



Fig. 2 A conventional parallel MMRs for a RRNS.

In this paper, we apply (1) the piecewise linear units (PLU) to limit the DR of next layer, (2) stored sign-and-parity bits to make sign detection efficient, and (3) the hierarchical structure to make the best of efficient moduli. Furthermore, we incorporate (4) the AN-codes for both sub-RNS correction and MMR checking. we develop an AN-coded hieratical RNS with an expanded RBC (ERBC) for both error checking and error correction in the SME model of sub-RNS, and MMR of the whole RNS. Compared to conventional MMR-based RNSs, the converters with comparators of each output can then be reduced to only small converters with an ERBC. Additionally owing to AN Codes, the system reliability can be improved by 126 times. The rest of this paper is then organized as follows. In Sec. II, basic theorems about AN coded RRNS and state-of-the-art works related to four major issues are reviewed. The proposed technologies are then proposed in Sec. III. Next, evaluation and simulation results are presented. Finally, some conclusions are drawn in Sec. V.

# Reviews on AN Codes and RRNS

*A. Arithmetic Weights*

A *n*-bit integer *x* can be represented by a ternary-coded binary (TCB),

, (1)

where and signs ± separately represent ±1. Distinguishing from a *digital weight* (DW) in representations and the *network* *weights* (NW) of a NN, the minimum count of ± signs in TCBs is defined as the *arithmetic weight* (AW) of *x*,

. (2)

The AW distance (AWD) of two integers *x* and *y*, is defined as ***w***(*x-y*). A number with an AW no more than 2 (3) is called a *nice* (*light*) *number*. The AW error (*w*-AWE) model is constructed under the assumption that the AWD is no more than *w*. Each *n*-bit 2's complemented number can be converted to a unique non-adjacent format (NAF) of a TCB without neighboring non-zero bits.

*B. AN Codes*

For checking an AWE, in 1955 Diamond proposed the concept of AN codes [11]. Namely, the message word *N*, , is encoded to the product codeword *C=AN*, and decoded by =*C/A*. The *w* bits of AW errors (*w*-AWE), *e*, can be located as , where denotes the residue of *x* mod *A*. Note that in the AWE model a single error can be or at location *i*, therefore a *subgroup* (*S*) and a *coset* (*C*) are generated by ×2 respectively from +1 and . Unique residues can be used to identify the error location. If *A* is an odd prime number, and 2 is a *primitive element* for generating a multiplicative group G(*A*) in Galois field, GF(*A*), namely, is called to be *2-primitive*, then G(*A*) is *perfect* to locate the AWE indices of a -bit AN codeword. In this paper, we find that *2-primitivity* is sufficient for *perfectness* but not necessary. For convenience of explanation, we represent a GF(*A*) by generating successive elements by. From GF(3)={0; (1, 2)}, GF(5)={0; (1, 2, 4, 3)} and GF(7)={0; (1, 2, 4) (3, 6, 5)}, where the set in parentheses is *cyclic*, we find that GF(3) and GF(5) are 2-primitive but GF(7) not. For further explanation, we define the multiplicative groups by ×2 from (i.e. 1 and *A*-1) are separately called coset and coset. Then a prime number is called *perfect* for AN codes if the non-zero subset can be separated into only coset and coset. When the cosets are the same and cyclic together, GF(A) is 2-primitive. It is call twin rings when cosets are separate and individually cyclic. Fig.3(a) shows twin rings of GF(23), while (b) illustrates a 2-primitive Mobius ring of GF(13). Note that in a modulo-*A* ring, *A*-1 -1 (mod *A*).

(a) (b)

Fig. 3 (a) Twin of GF(23) and (b) a Mobius ring of GF(13).

Table I lists some composite and prime numbers, where prime numbers are further classified. Note that primes 7, 23, 47, 71 and 79 are perfect but not 2-primitive.

1. Classification of AN Code-related Numbers

|  |  |  |  |
| --- | --- | --- | --- |
| composite | non-perfect | | 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, … |
| prime | 2, 17, 31, 41, 43, 73, 89, 97 |
| perfect | Twin | 7, 23, 47, 71, 79 |
| 2-primitive | 3, 5, 11, 13, 19, 29, 37, 53, 59, 61, 83 |

Table II shows the searched perfect AN-code multipliers *A* for a codeword up to *n* bits. For example, 37 is a perfect multiplier with single-error correcting (SEC) capability of an 18-bit codeword for N≤7084. A natural number *A* with GF(*A*)={0; *S*=(1, …), *C*=(-1, …), *Z*} can be applied for min(|*S*|, |*C*|)-bit AN codes but inefficient if *Z* is not empty [12], where *S* and *C* are called subgroup (1-coset) and coset (-1-coset), separately.

1. Multipliers for perfect AN codes

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *N* | 5 | 6 | 9 | 11 | 14 | 18 | 23 | 26 | 29 | 30 | 33 | 35 | 39 | 41 | 50 | 51 | 53 | 55 | 65 |
| *A* | 11 | 13 | 19 | 23 | 29 | 37 | 47 | 53 | 59 | 61 | 67 | 71 | 79 | 83 | 101 | 103 | 107 | 121 | 131 |
| ***w***(*A*) | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 3 | 3 |

For additive-preserving, (*n*, *k*)-AN codes encode *k* bits of message *N* by multiplying about *n*-*k* bit multiplier *A* for locating 2*n* 1-AWEs. Therefore, and the code rate will be approaching to

. (3)

However, for multiplicative-preserving, *A* should satisfy the idempotence, . It means that *A* times *A* will be overflow over *A*. This implies that and the code rate will be less than 50%.

Although AN codes had been studied a little earlier than RNS, from our conjecture, owing to the inefficient multiplicative-preserving issue, the later developments were almost focused on the RNS.

*C. Residue number systems*

Eq.(4) shows the preserving theorem for a RNS of *k* moduli ,

, (4)

where preserved operators can be and , but .

The modular additions were early implemented in [13][14]. Then the modular multiplication arose a lot of interests when conjugate pairs were took as moduli [15][16]. Since modulus requires one extra state or bit, and modulus will diminish one state and suffer the diminished-1 issue for modular multiplication [17][18][19].

An integer can then be represented by in the RNS, where denotes and converted back by the Chinese remainder theorem (CRT),

, (5)

where , , and . Note that will be a constant with merits for logic simplification. A lot of literature were devoted to RBCs [20][21] and moduli selecting. The RBC can be speeded up by Barret reduction [9] if is light, or by approximate CRT in [22],

, (6)

where is a constant decimal that can also be taken for logic simplification, and for a real number *x*.

As for moduli selecting, due to lightweight, , and are the most favorite forms of *L*-bit moduli for converter design [23]. Although a lot of researchers devoted themselves to moduli searching, an efficient set of moduli could not be searched systematically until the work in [4].

*D. Issues of RRNSs*

RNS is suitable for variable precision arithmetic and multiplication-rich datapaths like RSA encryption and RNS NNs [4][3]. However, there still exist three critical issues of all RNSs and one issue due to decoding the RRNS.

(1) *DR Inflation (DRI)*

The DRI issue may be neglected or hardly occurred in previous work. The authors of the most recent work [5] extend the multiplicity *L* of the modulus to overcome the DRI issue, but the acceleration is then quite limited, and the seldom-used circuitry for bit extension still takes the overhead. In this paper the DR is rectified in advance by a piecewise linear units (PLUs) [10] such that there is no DRI in the next layer.

(2) *Limited Efficient Moduli*

In [24] the authors propose an expansion-weight ratio (EWR, ) as a heuristics for moduli-selecting automation. However, from our experience, efficient moduli are quite limited. Therefore, in this paper the hierarchical RNS [25] with the ordered forms of , , , , and only is applied since we find that the forms can be reused in the submodules.

(3) *Sign Detection*

In a DR-aware system, the DRI can be avoided by careful design without overflow such that sign detection is unnecessary in a RNS. However, when the RNS is applied in a NN, the activation functions need magnitude comparisons so that the sign detection is required.

(4) *Combinatory Complexity*

Different from the majority decoder of a TMR, the MMR of a RRNS need to compare all pairs of residue sets. As a result, the combinatory complexity becomes a big issue.

# Proposed Technologies

## The model weights and input datasets are assumed pretrained and normalized such that the model weights and the activation function outputs can be quantized to and bits and denoted as . Without loss of generality, a L-layer deep neural network (DNN) can be represented as

, (7)

where , and are respectively the *l*-th layer activation outputs, weights and activation functions. The proposed AN-Coded hierarchical RNS (AN-HRNS) for DNNs is shown in Fig.4.



Fig. 4 A Proposed AN-HRNS architecture.

## Linear and DR-Preserving Activation Function

The linearity of rectified linear units (ReLUs) and piecewise linear units (PLUs) [10] as shown in Fig.5(a) and (b) can preserve the distributive law. In this paper WA PLUs are selected as activation functions such that rectification of the sums within only bits can limit the external DR of the l-th layer within bits, where is the input node count of the -th layer. Note that the 2’s complement product of a -bit multiplicand and a-bit multiplier takes bits. The sign detection or magnitude comparison to decide the PLU range can be simplified by Eq’s.(5) and (6). Note that the PLU functions are AN-preserving, that is, the output is also a A-multiple if A divides the input *x*.

(a)  (b)

Fig. 5 (a) A ReLU and (b) a PLU functions for RNSs.

## Hierarchical Structure for Best Moduli Selection

Owing to the application of PLUs, the DRs can be aware and the moduli can be planned in advance. Additionally, we find that the moduli of the external RNS system can still be applied again in its subsystems. For convenience of explanation, let’s take the most regular case as an example. The simplest and most efficient approach is to choose {*, ,* } in order as the high-level moduli, and {, , } as the low-level moduli set. To begin with, for Layer , *H* is selected as if five (*h*=5) moduli are all selected. The submodules will take the high-level residues as inputs and finally generate the output residues. Therefore the DR of the subsystems can be limited to bits, where *A* is the AN-Code multiplier. *L* can be selected as bits if the first k moduli are selected. Inputs are converted to RNS{} by BRCs as shown in BRC\*. Each residue can be looked as internal binary numbers and encoded to AN codes by multiplying . Thereafter, the multiplier of AN codes is sometimes denoted as *a* for distinguishing from activated output *A* in a DNN. The *i-*th products are then converted to sub-RNS{}.

For example, a trained model of NN-784-16-10 is quantized to W8A8, namely 8-bit weights and 8-bit activation outputs. For the first layer, the DR will be 8+8-1+24.6. Therefore, if *h*=5, {32, 31, 33, 29, 35} are selected as the moduli of the first layer. The five submodules will be protected by multiplying *A*. Taking the largest modulus 35 for explanation, *A* should be able to locate (*A*-1)/2 AWE positions, therefore, . *A=23* is selected according to Table II. If 23 is used for self-checking only, the aliasing rate will be 1/23. If it is used to locate a single AWE, it can corrected an 11-bit integer. As for the submodules, since more modular operations are executed in submodules, the moduli count *k*=3 is selected for taking the best. The internal DR will be bits. Therefore, the moduli {512, 511, 513} are selected.

## Sign Detection Techniques for Comparision in PLU

Sign detection as well as associated overflow issues have been studied for more than half a century [26]. The basic approach is still to simplify the calculations of residue-to-binary conversion. Fortunately, except of residue variables, all the other moduli and parameters are constant and can be simplified into few additions in advanced as explained in Eq.(5). In this paper, the convertors can be further simplified owing to our hierarchical moduli selecting strategy.

## AN-Code Checkers/Decoders and Expanded RBC

(1) *AN-Code Decoders*

After *L* layers, the outputs are then converted by RBC\* as shown in Fig.4. Since the DNN are composed of multiplications, additions and integer-linear functions, correct outputs should be divisible by *a*. Blocks ***/a*** shows a AN codes decoder. In this paper, we consider only partial (unary) encoding of binary inputs for multipliers. Actually, in most inference machines, the weights are constant and can be optimized into logic circuitry by technology mapping. The paths along the network are dynamic and tend to be infected by errors. Therefore it is reasonable to consider only the data-paths for encoding. Taking 13N codes as an example without loss of generality, Table III shows the error-quote-remainder table, where *AN + e = AN + Q + R*. Then a 13N decoder can be designed as shown in Fig.6.



Fig. 6 A decoder of 13N codes.

1. Error-Q-R Table for logic optimization

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AWE | 1 | 2 | 4 | 8 | 16 | 32 | -1 | -2 | -4 | -8 | -16 | -32 |
|  | 0 | 0 | 0 | 0 | 1 | 2 | -1 | -1 | -1 | -1 | -2 | -3 |
| R | 1 | 2 | 4 | 8 | 3 | 6 | 12 | 11 | 9 | 5 | 10 | 7 |
|  | 000 | 000 | 000 | 000 | 001 | 020 | 111 | 111 | 111 | 111 | 110 | 101 |

## (2) Double-Mode Expanded RBC

To make the best use of the redundancy, all the decoded residues *xi* with the error signals *Ei* will feed to the expanded RBC (ERBC) as shown in Fig.7. Assume that any modulus in can be redundant. When any integer for all , the converted binary numbers will be equal. For example, 20 can be represented as (2, 0, 6, 9) in RNS(3, 5, 7, 11). Once if the last residue is faulty, the number can still be converted by (2, 0, 6, 0) in RNS(3, 5, 7, 1). In the MMR mode, the one-hot error codes choose the corresponding modulus and residue to be 1 and 0 respectively. In the ECC mode, all residues are corrected and fed into the ERBC.



Fig. 7 An ERBC.

According to the central limit theorem as shown in Fig.8, most data are centralized far from the DR edges, i.e., a part of residues can be converted to its binary numbers. Therefore the ERBC can be operated at MMR mode for any error with an aliasing rate of 1/*a* or ECC mode for the 1-AWE output correction.



Fig. 8 Analysis of data centralization.

# Evaluations

## Acceleration and Key Features

The DR is usually limited by the first layer. For example, if the quantization is set to W8A3, the DR of the NN-784-16-10 for MNIST bench dataset will be limited by the first layer to 256x256x784, so {32, 31, 33, 29, 35} is a proper set for external moduli. For reducing the aliasing rate in MMR mode, AN-multiplier *A* is as big as possible, but for output 1-AWE correcting in ECC mode, 23 is proper for *A*. When *A* is set to 23 for the sub-RNS, the internal DR will be 24.3 bits. Therefore, the moduli {512, 511, 513} are selected.

An AN-HRNS synthesizer is developed using Python in Xilinx® Pynq-z2 platform. Generated fixed-point c++ codes are synthesized in HLS for generating Verilog codes. A part of specific converter IPs are coded in Vivado. Fig.9 shows the layout in all-programmable SoC Zynq7020/Pynq system for experiments.

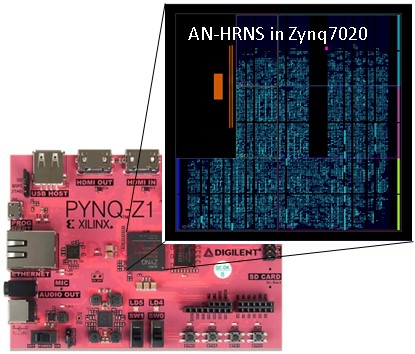


Fig. 9 Layouts in Zynq7020.

As shown in Table IV, only the authors in [5] considered the DRI issue in NNs. Owing to hierarchical moduli selection, efficient and optimized moduli can be selected in this work. Compared to the 26-bit multiplications in a binary number system, our multipliers take reduce the critical paths from 52 to only 14. No doubt that we can achieve more than 2.7 times of acceleration rate finally. When a pair of converted binary numbers are mutually checked, the aliasing rate in [3-5] is about 1/DR, while a higher aliasing rate up to 1/*a* in this work. The last three rows show that conventional RRNS require full RBCs with comparators. But our AN-HRNS needs only *h* sets of k-converters with only a ERBC.

1. omparison with previous work on RRNS.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Work | [3] | [4] | [5] | Ours |
| Consider DRI | No | No | Yes | Yes |
| Acceleration rate | NA | 2.5 |  | 2.7 for 5 moduli |
| Checking | Mutual | Mutual | Mutual | AN-code self |
| Aliasing Rate | NA | NA | 1/DR | 1/*A* |
| Redundant module | 2 | 2 | 2 | Hierarchical |
| *k*-moduli converters |  |  |  | h sets of k-converter  with a ERBC |
| #comparators |  |  |  |

For comparing the critical paths and area overhead, the NNs are also implemented in a 0.18 CMOS technology. Fig.10 shows the layout. Table V shows the comparisons on area, delay and power dissipation of three implementations of the NN784-16-15-Softmax inference macnine for MNIST hand-written digit recognition. Column QNN shows the quantized neuronet and ANRRNS shows the results by AN-encoding one of our previous work in [4]. Column *This work* shows the experimental results in this paper. We can find that the speed is promoted according to best moduli selection and hierarchical structure. Actually, owing to the hierarchical structure, the AN codes can help protect the subsystem modules effectively and efficiently.

1. Comparison with previous work on RRNS.

|  |  |  |  |
| --- | --- | --- | --- |
| Work | QNN (binary) | ANRRNS | Ours |
| Area () | 93,644 | 1,591,867 | 2,823,006 |
| Delay (ns) | 90.23 | 37.30 | 16.26 |
| Power dissipation (mW) | 1,474 | 1,784 | 78.50 |

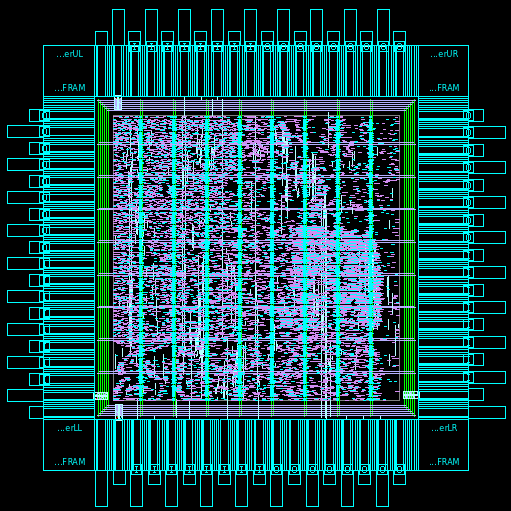


Fig. 10 Layouts in a cell-based technology.

Fig. 11 shows the accuracy test results. Two synapses of each layers in each layers of NN784-16-10 are randomly injected with no more than two AWEs with a SNR under the AWGN model. The curve of QNN shows that the accuracy will suffer serious risk without any design for reliability.

Fig. 11 Cross validations for QNN, ANRRNS and AN-HRNS.

## Reliability Improvement

Block error rate (BLER) analysis is an effective approach to estimate the upper/lower bound of the failure rate () of a system of fully independent/dependent blocks during a period (T) and the MTBF can then be estimated as . Assume the outputs of multiplier and adder suffers times of additive white Gaussian noise (AWGN), and the input wires have a lower noise infection rates (). To estimate the improvement on the NNs, in this paper only a MAC is taken as a block due to simulation time complexity. Fig.12 shows the infection points of a 16-bit MAC with A=47.



Fig. 12 Results of BLER simulation.

Fig.13 shows BLER simulations for uncoded (in red) and partial AN-coded (in blue) when =1, 0.1 and 0.01, separately. We find that the block reliability can be improved up to 126 times when and SNR=12 dB.



Fig. 13 Results of BLER simulation.

For a system with *n* blocks, the system failure rate ( will be the same as the block error rate (BLER = ) if all block are fully correlated. To the other extreme, if all blocks are independent, . Since the mean time between failure (MTBF) is proportional to , the improvement for MTBF will be the improvement for BLER for both extreme cases. That is, the MTBF can be improved up to 126 times.

# Conclusions

Considering conversion cost, dynamic range, acceleration and reliability, in this paper we proposed an AN-coded hierarchical residue number system. Two-level hierarchy makes the moduli searching easy and efficient, preserves the AN-coding theory, and provides a checking mechanism for multiple modular redundancy. In this preliminary work, from detailed and robust evaluations, our AN-HRNS architecture can achieve about three times of acceleration. From BLER analyses for either highly correlated error distribution or in a less-noisy environment, the improvement on the mean time between failures of the whole system can be up to 126 times.

##### References

1. A. Bosio *et al*., "A Reliability Analysis of a Deep Neural Network," IEEE Latin American Test Symp., Chile, 2019.
2. M. Cheng, L. Xia, Z. Zhu, Y. Cai, Y. Xie, Y. Wang, H. Yang. "TIME:A Training-in-memory Architecture for RRAM-based Deep Neural Net-works," IEEE Trans. CAD of IC and Sys-tems, 2018, pp.1-14.
3. K. Phalakarn and A. Surarerks, "Alternative Redundant Residue Number System Construction with Redundant Residue Representations," Int’l Conf. Comp. & Comm. Sys. (ICCCS), Nagoya, 2018, pp. 457-461.
4. T.-C. Huang. "Self-Checking Residue Number System for Low-Power Reliable Neural Network," IEEE Asian Test Symp., pp.37-42, Dec. 2019.
5. N. Samimi, M. Kamal, A. Afzali-Kusha and M. Pedram, "Res-DNN: A Residue Number System-Based DNN Accelerator Unit," in IEEE Trans. on CAS I: Regular Papers, vol. 67, no. 2, pp. 658-671, Feb. 2020.
6. C. Su *et al*., "An algorithm for scaling and single residue error correction in RNSs," IEEE Trans. Comp., 39(8): 1053-64, Aug. 1990.
7. R. S. Katti, "A new residue arithmetic error correction scheme," in IEEE Transactions on Computers, vol. 45, no. 1, pp. 13-19, Jan. 1996.
8. T. Fatt Tay *et al.*, "A non-iterative multi-residue digit error detection & correction algo. in RRNS," IEEE Trans. Comp. 65(2):396-408, Feb. 2016.
9. P. Barrett, "Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor," in Advances in Cryptology, CRYPTO 86, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1987, vol. 263, pp. 311–323.
10. Nicolae, A., "PLU: The Piecewise Linear Unit Activation Function", *arXiv e-prints*, 2018.
11. J. M. Diamond, "Checking codes for digital computers," Proc. IRE, vol. 43, pp. 487-488, Apr. 1955.
12. R. T. Chien, "On linear residue codes for burst errors correction," IEEE Trans. Inform. Theory, vol. IT-lO, pp. 127-133, Apr. 1964.
13. T. R. N. Rao, "Error-checking logic for arithmetic-type operations of a processor," IEEE Trans. Computers, vol. C-17, pp.845-849, Sep. 1968.
14. N. Garcia, "Error correction in parallel binary adders," Proc. Southeastern Symp. on System Theory B-4, Virginia Polytechnic Institute, Blacksburg, Va., May 1969.
15. R. Zimmermann, "Efficient VLSI implementation of modulo addition and multiplication," Proceedings 14th IEEE Symposium on Computer Arithmetic (Cat. No.99CB36336), Adelaide, SA, Australia, 1999, pp. 158-167.
16. J.-C. Bajard, L.-S. Didier, and P. Kornerup, "Modular Multiplication and Base Extension in Residue Number Systems," Proc. 15th IEEE Symp. Computer Arithmetic, N. Burgess, ed., pp. 59-65, June 2001.
17. C. Efstathiou, H. T. Vergos, G. Dimitrakopoulos and D. Nikolos, "Efficient diminished-1 modulo multipliers," in IEEE Trans. on Computers, vol. 54, no. 4, pp. 491-496, April 2005.
18. J. W. Chen and R. H. Yao, "Efficient modulo 2n + 1 multipliers for diminished-1 representation," in IET Circuits, Devices & Systems, vol. 4, no. 4, pp. 291-300, July 2010.
19. W. Hong, R. Modugu and M. Choi, "Efficient Online Self-Checking Modulo 2^n+1 Multiplier Design," in IEEE Trans. on Computers, vol.60, no.9, pp.1354-1365, Sept. 2011.
20. T. R. N. Rao, "Error-checking logic for arithmetic-type operations of a processor," IEEE Trans. Computers, vol. C-17, pp.845-849, Sep. 1968.
21. C. Efstathiou *et al.*, "Efficient diminished-1 modulo multipliers," in IEEE Trans. Comp., vol. 54, no. 4, pp. 491-496, April 2005.
22. N. N. Kucherov, et al. "A high-speed RBC based on approximate CRT," Conf. of Russian Young Researchers in EEE, pp. 325-328, 2018.
23. T. R. N. Rao, "Biresidue Error-Correcting Codes for Computer Arithmetic," IEEE Trans. Comp., vol.C-19, pp.398–402, May 1970.
24. H.-W. Fu, T.-Y. Chen, C.-D. Tsai, M.-W. Shen and T.-C. Huang. "AN-Coded Redundant Residue Number System for Reliable Neural Networks," 2021 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), June 17, 2021.
25. H. M. Yassine, "Hierarchical residue numbering system suitable for VLSI arithmetic architectures", Proc. IEEE International Symposium on Circuits and Systems, vol. 2, pp. 811-814, May 1992.
26. D. K. Banerji and J. A. Brzozowski, "Sign Detection in Residue Number Systems," in IEEE Transactions on Computers, vol. C-18, no. 4, pp. 313-320, April 1969.