Evaluation of Mobile Network Inference in Nigeria Using Residue Number System

Oluwatunde, S. Johnson, Agbesawa, D.O. , Ogunjimi, L.A. & Longe, O.B (PhD)
Department of Computer Science
Caleb University
Lagos, Nigeria
E-mail: oluwatundesola@yahoo.com

ABSTRACT

This work explores the lesser studied objective of optimizing the multiply-and-accumulates executed during evaluation of the network. In particular, we propose using the Residue Number System (RNS) as the internal number representation across all layer evaluations, allowing us to explore usage of the more power-efficient RNS multipliers and adders. Using results from simulation of our RNS arithmetic block implementations, we show theoretical power advantages of using RNS for an end-to-end evaluator.

Key words: Chinese Remainder Theorem (CRT), Inference, Mobile network, VHDL, RNS

iSTEAMS Cross-Border Conference Proceedings Paper Citation Format
Oluwatunde, S. Johnson, Agbesawa, D.O., Ogunjimi, L.A. & Longe, O.B (2018) Evaluation Of Nigeiran Mobile Network Inference Using Residue Number System. Proceedings of the 13th iSTEAMS Multidisciplinary Conference, University of Ghana, Legon, Accra, Ghana. Vol. 2, Pp 225 -234

1. INTRODUCTION/BACKGROUND OF THE STUDY

A network is also a collection of computers, servers, mainframes, network devices, peripherals, or other devices connected to one another to allow the sharing of data. An excellent example of a network is the Internet, which connects millions of people all over the world. A network consists of multiple devices that communicate with one another. It can be as small as two computers or as large as billions of devices. Networks can be broadly classified as using either a peer-to-peer or client/server architecture.

1.1 Mobile Network

A mobile communications system that uses a combination of radio transmission and conventional telephone and data switching to permit communication to and from mobile users within a specified area. Note: In cellular mobile systems, large geographical areas are segmented into many smaller areas, i.e., cells, each of which has its own radio transmitters and receivers and a single controller interconnected with the public switched telephone network. The most commonly used radio systems are GSM (Global System for Mobile Communication) and CDMA (Code Division Multiple Access). As of September 2017, Verizon, Sprint, and US Cellular use CDMA. AT&T, T-Mobile, and most other providers around the world use GSM, making it the most widely used mobile network technology. LTE (Long-Term Evolution) is based on GSM and offers greater network capacity and speed.

Communications refers to the use of signals to transfer voice, data, image, and/or video information between locations, Nigeria has Africa's largest mobile market, with about 142 million subscribers and a penetration rate of 101%. The initial rapid growth in the number of subscribers had led to problems with network congestion and quality of service, prompting the regulator to impose fines and sanctions on network operators. These operators have responded by investing billions of dollars in base stations and fibre transmission infrastructure to support the increasing demand for data. Although GSM technology still dominates there is a growing shift to services based on LTE. Significant effort has been made to reduce the memory footprint of networks, motivated by the fact that many off-chip memory accesses can dominate energy consumption during evaluation (Zhou et al., 2016; Han et al., 2015). This paper explores the lesser studied objective of optimizing the multiply-and-accumulates executed during evaluation of the network. In particular, we propose using the Residue Number System (RNS) as the internal number representation across all layer evaluations, allowing us to explore usage of the more power-efficient RNS multipliers and adders.

We motivate our optimization with Table 1, which summarizes the large number of multiply-and-accumulates (MACs) required during evaluation of popular networks. Small improvements to the underlying efficiency of the core multiply and accumulate block can have large improvements to the overall network evaluation.

Table 1. Computation Accounting for Popular Mobile Networks

NET	MACs (10^6)	PARAMS (10^6)
ETISALAT	720	60
AIRTEL Nigeria	1550	6.8
GLO Mobile Nigeria	1700	1.25
MTN Nigeria	15300	138

Prior work applying RNS for efficient computation has largely focused on cryptographic applications and general purpose ALUs. In the machine learning domain, various optimizations such as Winograd and FFT transforms have been proposed to speed up network inference. As far as we are aware, this is the first attempt at applying RNS to mobile network inference.

1.2. Network Inference

The purpose is to explain how to estimate a regulatory network from calls database. This issue is called network inference. We will restrict ourselves to the very simple case where a large set of calls have been measured on a few number of individuals. The data set is represented by the matrix X:

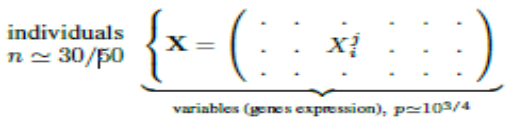

Figure 1: Showing network inference

Even restricting to a small subset of calls database, having n < p is the standard situation. From these data, we want to build a network where:

- the nodes represent the p calls ;
- the edges represent a “direct” and “significant” co-expression between two calls. This kind of relations aims at tracking transcription relations.

The main advantage of using networks over raw data is that such a model focuses on “significant” links and is thus more robust. Also, inference can be combined or compared with or to bibliographic networks to incorporate prior knowledge into the model but, unlike bibliographic networks, networks inferred from one of the model presented below can handle even unknown (i.e., not annotated) calls into the analysis.

1.3 Basics About Network Inference

Even if alternative approaches exist, a common way to infer a network from call database is to use the steps described in Figure 2:

1. The user calculates pairwise similarities (e.g., correlations in the simplest case) between pairs of calls ;
2. The smallest (or less significant) similarities are thresholded (using a simple threshold chosen by a given heuristic or a test or other more sophisticated methods) ;
3. Last, the network is built from the remaining similarities.

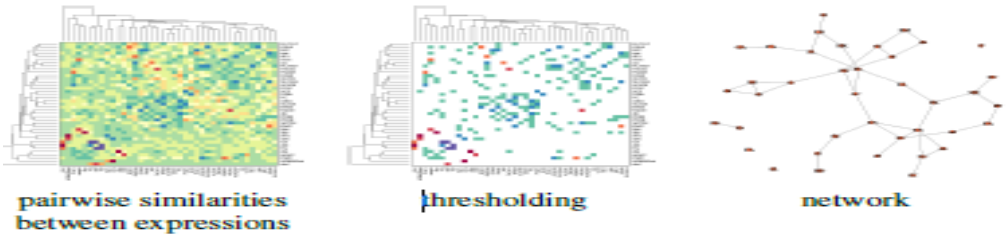

Figure 2: Main steps in network inference

1.4 Residue Number System

The Residue Number System (RNS) is an unconventional or unweighted number system that is defined in terms of relatively prime moduli set $\{m_1, m_2, \ldots, m_n\}$ that is gcd $(m_i, m_j) = 1$ for $I \neq j$. A weighted number X can be represented $X = (x_1, x_2, \ldots, x_n)$ where $xi = X \bmod mi = |X|m_i$, $0 \leq xi \leq m_i$ such a representation is unique for any integer X in the range [0, M - 1] where M is the dynamic range of the moduli set $\{m_1, m_2, \ldots, m_n\}$. The residues are smaller than the original/weighted/converntional number and as a result, the arithmetic operations such as addition, subtraction and multiplication could be carried out independently and parallel between different modules. Therefore, RNS is suitable in an application including addition, subtraction and multiplication widely, such as RSA, digital signal processing, image processing, and fault tolerant.

VHDL (***VHSIC Hardware Description Language***) is a hardware description language used in electronic design automation to describe digital and mixed-signal systems such as field-programmable gate arrays and integrated circuits. VHDL can also be used as a general purpose parallel programming language.

2. BACKGROUND AND EXISTING RELATED WORKS

A simple approach to infer a network from call database is to calculate pairwise correlations between calls and then to simply threshold the smallest ones, eventually, using a test of significance. This approach is known under the term relevance network [Butte and Kohane, 1999, Butte and Kohane, 2000].Let us describe the simple following situations displayed in Figure 3. In this model, a single call, denoted by x, strongly regulates.

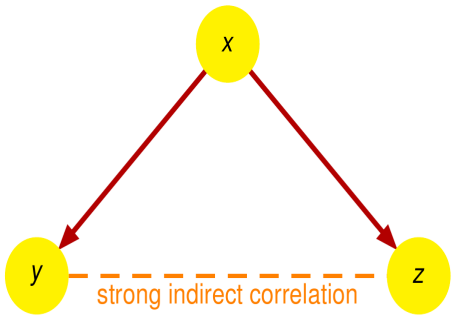

FIGURE 3 – Small model showing the limit of the correlation coefficient to track regulation links : when two calls y and z are regulated by a common call x, the correlation coefficient between the expression of y and the expression of z is strong as a consequence expression between two other genes, y and z.

The correlation between the expressions of x and y and the correlation between the expressions of x and z are strong but, as a consequence, the correlation between y and z is also strong: using the simple mathematical model. The modular and distributive properties of RNS are used to achieve performance improvements especially in the emerging distributed and ubiquitous computing platforms such as cloud, wireless ad hoc networks, and applications which require tolerance against soft error. Secondly, energy efficiency becomes a key driver in the continual densification of Complementary Metal Oxide Semiconductor (CMOS) digital integrated circuits. The high degree of computational parallelism in RNS offers new degree of freedom to optimize energy performance, particularly for very long word length arithmetic such as those involved in the hardware implementation of cryptographic algorithms.

RNS is based on a puzzle introduced by the Chinese mathematician Sun-Tzu, which was later named as Chinese Remainder Theorem (CRT) (Omondi and Premkumar, 2007). Based on CRT, Harvey Garner (Garner, 1959) invented RNS in 1959. It has several interesting number theoretic properties and unique features that can be used to boost up the speed of certain electronic computations (Omondi and Premkumar, 2007). The carry-propagation chain of conventional binary number system was then the main bottleneck of fast arithmetic operation and became the key motivation driving researchers to venture into this alternative number system for which the residue arithmetic operation in each modulus channel is independent and carry-free. Cheney in 1961 (Cheney, 1961) used these features of RNS to design a digital correlator with ten times faster speed than that based on conventional binary number system. This correlator was the first system-level design based on RNS.

A year later, Guffin designed a special-purpose digital computer for solving simultaneous equations using RNS with a great speed advantage (Guffin, 1962). To broaden its applications, researchers are motivated to solve the difficult RNS operations in order to achieve overall performance improvement for general digital computing systems (Merrill, 1964). Therefore, division, overflow detection, sign-detection and magnitude comparison have also come into the limelight of RNS research since 1962 (Szabo, 1962; Keir, Cheney and Tannenbaum, 1962). Meantime, further improvements of the essential RNS arithmetic units, such as modulo adder, multiplier, forward converter and reverse converter (Sasaki, 1967; Banerji and Brzozowski, 1972) remain a hot pursuit.

3. PROPOSED SYSTEM

The proposed methodology for this study consists of four (4) main steps; Moduli set selection, Computational cost, Thread architecture implementation and Adoption of CRT and without CRT. The Chinese Remainder Theorem (CRT) is a technique to reduce modular calculations with large moduli to similar calculations for each of the (mutually co-prime) factors of the modulus. The first description of the CRT is by the Chinese mathematician Sun Zhu in the third century AD. The CRT makes it possible to reduce modular calculations with large moduli to similar calculations for each of the factors of the modulus. At the end, the outcomes of the subcalculations need to be pasted together to obtain the final answer. The big advantage is immediate: almost all these calculations involve much smaller numbers (Henk and Jajodia, 2011).

In this research work, we x our moduli to the structured 4-tuple $\{2^n \pm 1, 2^{n+1} \pm 1\}$. This set can represent numbers in range $[0; M = ((2^{2n} - 1)(2^{2n+2} - 1))/3]$ (Sousa, 2007). We choose the n = 7 moduli set in this work. Every RNS number is stored using 7 + 8 + 8 + 9 = 32 bits, and can fall in the range $[0, ((2^{14} - 1)(2^{16} - 1))/3 = 357886635]$. This is the representational range of a 28-bit unsigned integer. Conversion in and out of RNS is based on CRT Theorems I, II, and III. Unfortunately, with the requisite divisions and iterative algorithms, conversion requires significant arithmetic overhead, as discussed by (Hiasat, 2003). Chinese Remainder Theorem (CRT): translates a residue represented number into its equivalent weighted number. An RNS number can be converted into weighted number X as follows:

Theorem: Let $m_1, \ldots, m_n$ be pairwise coprime (that is $\gcd(m_i, m_j) = 1$ whenever $i \neq j$). Then the system of n equations

$$x = a_1 \pmod{m_1}$$
$$\ldots$$
$$x = a_n \pmod{m_n}$$

has a unique solution for x modulo M where $M = m_1 \ldots m_n$.

Proof: This is an easy induction from the previous form of the theorem, or we can write down the solution directly.

Define $b_i = M/m_i$ (the product of all the moduli except for m_i) and $b_i' = b_i^{-1} \pmod{m_i}$. Then by a similar argument to before,

$$x = \sum_{i=1}^{n} a_i b_i b_i' \pmod{M}$$

Such that

$M = \prod m_i$ i.e $m_1 * m_2 * m_3 * \ldots * m_n$
$b_i =$, $M_i = M/m_i$ (the product of all the Moduli except for mi) and
$b_i' = M_i^{-1}$ = the multiplicative inverse of M_i w.r.t. m
is the unique solution ----------eq. I

NEW CRT1: is a method of converting from Residue Number System to conventional number system

$$X = x_1 + m_1 |k_1 (x_2 - x_1) + k_2 m_2 (x_3 - x_2) + \ldots + k_n m_2 m_3 \ldots m_{n-1} (x_n - x_{n-1}) | m_2 m_3 \ldots m_n$$ ----------eq. II

Such that

$K_1 = |m_1^{-1}| m_2 m_3 \ldots m_n$, $K_2 = |(m_1 m_2)^{-1}| m_3 m_4 \ldots m_n$, … $K_n = |(m_1 m_2 \ldots m_{n-1})| m_n$

NEW CRT2 Definition: It is one of the cheapest converter and allows fast performance of processor. Given moduli set [m1/m2/m3/m4] with Residue digits (x_1, x_2, x_3, x_4)= n where n = j+k.
$X_{1,2} = x_1 + m_1 |k_1(x_2-x_1)|m_2$ for $k_1 = |m_1^{-1}|m_2$
$X_{3,4} = x_3 + m_3 |k_2(x_4-x_3)|m_4$ for $k_2 = |m_3^{-1}|m_4$
$X = X_{12} + M_j|(X_{34}-X_{12})|M_k$ ----------eq. III

At its core, to construct complete end-to-end inference in RNS we need to support two operations: multiply-and-accumulate and ReLU. The choice of moduli allow us to build on prior work for both: (Zimmermann, 1999) proposes digital architecture for multiplication and addition mod 2n ± 1, and (Sousa, 2007) demonstrates comparison of RNS numbers mod 2n ± 1 in VHDL. We use a comparison module to implement the ReLU nonlinearity.

We assume a discrete output for our network { e.g. 10-output image-recognition with CIFAR-10. This allows us to avoid the overhead of conversion out of RNS at the end of the network; instead, with our comparator, we compute a max over final layer softmax values, returning the index with the maximum value. All RNS operations occur in the realm of positive integers with fixed.

3.1 Comparison in RNS

Though RNS multiplication and addition are operationally intuitive, comparing two RNS numbers is non-trivial. We follow the procedure given by (Sousa, 2007). The crux of the algorithm is reducing comparison to parity (even/odd) checking. To compare two unsigned integers A;B mod M, we compute the difference C = A - B which becomes one of two values:

$$C = \begin{cases} A - B & \text{if } A \geq B \\ M + A - B & \text{if } A < B \end{cases}$$

Because with our chosen moduli M is odd, these two values have different parities. As such, we can compute a comparison given a formula for the mod 2 parity X_P of an RNS number X = (x_1; x_1*; x_2; x_2*). Parity is calculated with the following set of equations:

$$X_1 = x_1^* + (2^n + 1) \times (2^{n-1}(x_1 - x_1^*) \bmod 2^n - 1)$$
$$X_2 = x_2^* + (2^{n+1} + 1) \times (2^n(x_2 - x_2^*) \bmod 2^{n+1} - 1)$$
$$X_P = LSB(X_2) \oplus LSB((X_1 - X_2) \bmod 2^{2n} - 1)$$

Proof is provided by (Sousa, 2007). This is the full-comparator we implement in combinational logic to execute the final layer maximum. In the ReLU, we are able to trim this combinational logic, because we compare with a fixed threshold M/2 (0 in our modulus world). We call this trimmed comparator a half- comparator. In particular, the parity of B = M/2 is fixed and pre-computed, as well as the value of the additive inverse -B = -M/2 we feed into the modulo adder.

The parity-checking combinational logic implemented in Verilog is given by Figure 4. This is used in both the full and half comparator design. We modify the circuit given in (Sousa, 2007), which we suspect, from our testing, does not evaluate correctly in an exhaustive sweep of all approximately 18 million inputs.

We use various optimizations. For example, the choice of modulus allows for use of an inverter to find the additive inverse. It also allows for calculating the remainder with a 16 bit number as a single addition (bottom-right). Additionally, we implement multiplication with 2^n mod 2^{n+1} - 1 as a right rotate.

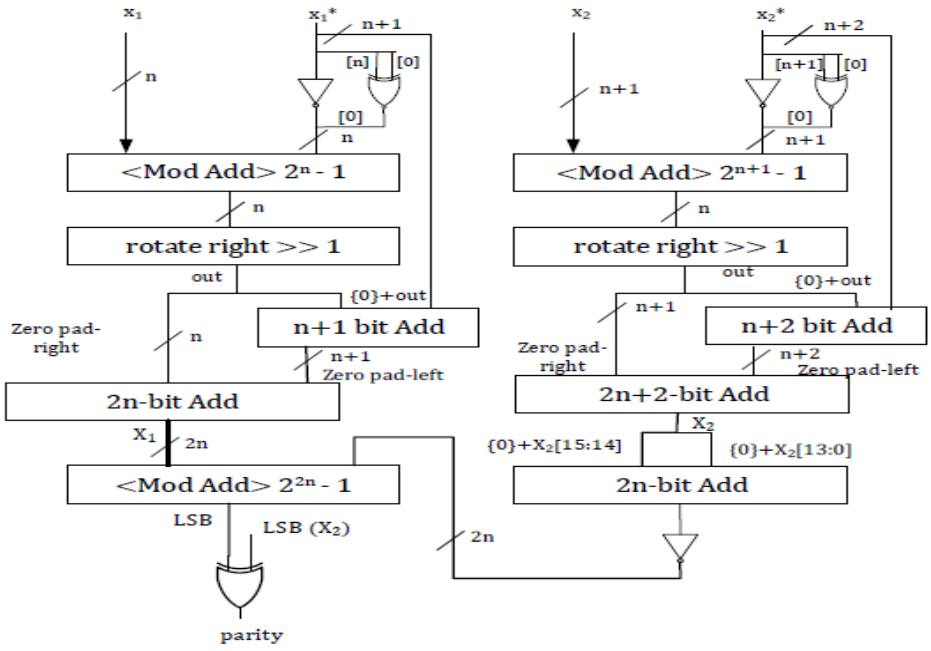

Figure 4. Combinational logic for calculating the parity of an RNS number (x_1, x_1*, x_2, x_2*)

4. ADDITION AND MULTIPLICATION IN RNS

To fully exploit the advantages of transforming the mobile network to the RNS system, efficient modulo arithmetic circuits will be designed using the same moduli set as the comparison, namely ($2^{n1} \pm 1$, $2^{n2} \pm 1$)

4.1 Modulo Addition

In RNS arithmetic has the advantage that each residue operates separately in parallel without any carry propagation from one residue to the other. Our conjugate moduli set requires modulo (2n - 1) addition or multiplication as well as (2n + 1).

First, the modulo (2n - 1) addition can be expressed as conventional n-bit addition if the sum is less than the modulus, while a correction is added if the sum oveflows the modulus as follows:

$$(A+B) \bmod (2^n-1) = \begin{cases} A+B-(2^n-1) \\ \quad = A+B+1 & \text{if } A+B \geq 2^n-1, \\ A+B & \text{otherwise} \end{cases}$$

Since, the output carry (c_{out}) of an n-bit adder can be used to detect the overflow condition which determines whether to increment the sum or not, then such carry can be fed back into the adder as proposed by (Zimmermann, 1999) for (2^n - 1) addition.

$$(A+B) \mod (2^n-1) = (A+B+c_{out}) \mod 2^n$$

where diminished-1 numbers can be used for the inputs, or a correction circuit is added to the output to account for the extra '1'.

Since the addition in both moduli depends on the output carry, then fast parallel prefix adders are the most suitable implementation for the modulo adders. Figure 5 shows the a modulo parallel prefix adder where the inputs are preprocessed into carry generate and propagate signals then a tree of fixed operation propagates the carry in only 3

levels. Each solid circle represents a dot operation which combines the group carry generate and propagate bits. Finally, a modulo end-around carry correction is required to feedback back the output carry (c_{out}) or its complement (c_{out}') according to the designated modulus.

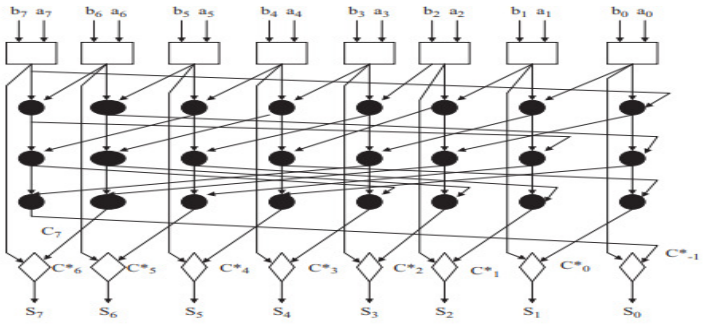

Figure 5: Parallel prefix adder for mod $2^7 - 1$ addition

4.2 Modulo Multiplication

N-bit binary multiplication relies on generating N partial products and accumulating them all to produce the final product. Modulo multiplication relies on the same concept as well as the periodicity of the binary weights which causes the higher order partial products to rotate folding back into n-bit weights. Therefore, the partial products can be generated, similar to in (Zimmermann, 1999), as

$$X \cdot Y \mod (2^n - 1) = \sum_{i=0}^{n-1} x_i \cdot (Y << i) \mod (2^n - 1)$$

where the << operator represents a circular shift. Similarly, an expression for the partial products of the $(2^n + 1)$ modulus can be derived to be

$$PP_i = x_i \cdot y_{n-i-1} \cdots y_0 \overline{y_{n-1}} \cdots \overline{y_{n-i}} + \overline{x_i} \cdot 0 \cdots 01 \cdots 1$$

Such multipliers can be designed in a modular way where a block generates the required partial products according to the selected modulus. Then, a modulo carry save adder tree shown in Figure 6 generates a redundant sum output (PC; PS) which is then added using a modulo adder to produce the final product.

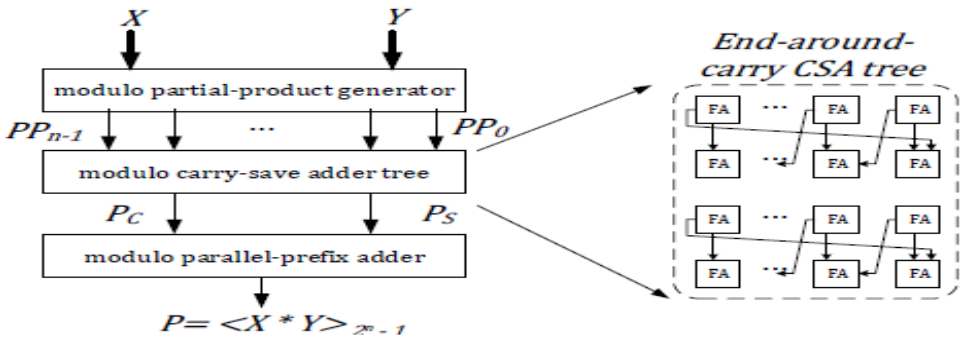

Figure 6: Mod $2^n - 1$ multiplier architecture showing the module carry save adder

5. RESULT AND DISCUSSION

5.1. RNS Power Consumption

We built several building blocks for RNS-based network inference using Bluespec SystemVerilog and synthesized them using a commercial LP65nm CMOS process. Table 2 shows the power and frequency of operation of the RNS blocks as well as their 32-bit counter-parts. It is worth noting that the multiplier consumes almost half the power of the 32-bit block with a positive slack allowing for higher frequency of operation.

Table 2. Synthesis results

BLOCK	P (mW)	f (MHz)	SLACK(ps)
ADDER32	1.05	625	15.9
ADDERRNS	1.18	625	17.6
MULTIPLIER32	3.04	250	7.1
MULTIPLIERRNS	1.56	250	95.4
CONVERTTORNS	2.6	250	1.1
RELU-RNS	0.88	156	109.5
COMPARERNS	1.67	156	93.1

5.2. Maintaining a Modulus Integer Network

A limitation using RNS is the necessity to maintain positive integer weights and activations within a given modulus M. To demonstrate the feasibility of this, and obtain a rough estimate of accuracy degradation when imposing such constraints, we train different flavors of a 8-layer (7 CN/1 FC) network on the Street- View House Numbers (SVHN) dataset (Netzer et al., 2011). We denote a (W, A)-FP/INT network as a network with W-bit weights and A-bit activations in either floating point or integer, respectively. Note that negative integers are interpreted as their respective positive value in a wrap-around modulus.

We first trained (32, 32)-FP. We used a set of shadow floating point weights, initialized to (32, 32)-FP, and truncate these shadow weights in the forward pass to generate our (6, 6)-FP network (gradients get passed to the shadow weights). In our (32, 32)-INT and (6,6)- INT networks, we modify this truncation operation to be a suitable affine transformation to fit our bit width and desired range. Note that our activation function in the integer network changes to compare with M/2. Networks were trained for 15 epochs, with data augmentation, 50% last layer dropout, and selecting the best model-checkpoint with highest validation accuracy. Note that a 6-bit integer is able to fit within each modulus of our RNS representation. As expected, reducing bitwidth increases error. Moving to integer networks appears to slightly decrease accuracy. The precise reason for this is unclear; perhaps, something wonky is occurring with the gradient updates and gradient magnitude. Networks were implemented in Tensorow or Tensorpack.

Table 3. SVHN Test Error Rate

NETWORK	SVHN TEST ERROR
(32, 32)-FP	3.95%
(6, 6)-FP	6.69%
(32, 32)-INT	4.54%
(6, 6)-INT	7.07%

5.3. Estimation of the RNS Break-Even Point

Use of RNS incurs overhead proportional to the output size, because of the comparatively expensive ReLU-RNS modules. If we have a Y * X fully-connected layer, we can compare the relative energy costs associated with performing the corresponding MACs and ReLU in RNS or non-RNS:

$$Y \times E_{RNSReLU} + XY \times (E_{RNSMult} + E_{RNSAdd}) > Y \times E_{ReLU} + XY \times (E_{Mult} + E_{Add})$$

E_X is the energy per operation for hardware block X.
Given our simulation results, this simplifies to:

$$X > \frac{E_{ReLU} - E_{RNSReLU}}{(E_{RNSMult} + E_{RNSAdd}) - (E_{Mult} + E_{Add})} \approx 0.98$$

This hints that it could be possible to achieve energy savings through RNS on FC layers of any size, because of our ReLU overhead/MAC savings ratio. It is demonstrable that the same result applies for a CN layer, in which we would replace X with $C_{in}K_XK_Y$, the size of channel-output filter. Note that this estimation is ignoring costs of memory accesses. Though, because of our choice of moduli, the amount of data being shuffled is similar in both systems.

6. CONCLUSION

In this research work, we outlined use of the Residue Number System to perform inference on mobile networks. Using our single-block implementation power estimates, we showed theoretical analysis of the advantages of RNS for an end-to-end system. In addition, this research, which will provide the following expected contributions amongst others:

1. The speed of the communication channel will tremendously be increased due to the carry-free property of RNS.
2. Congestion in the communication channel will be reduced as partial representation of actual data will be transmitted, therefore reducing the data traffic in the communication channel.
3. This will increase the general throughput of the communication system.
4. There will be a reduction in energy and memory consumption.
5. Generally the quality of service is expected to improve.

REFERENCE

1. Abdelhamid, M. & Koppula, S, (2017), Applying the Residue Number System to Network Inference, Cornell University
2. Courbariaux, M & Bengio, Y. (2016) Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. CoRR abs/1602.02830(2016)
3. Guffin, R.M., (1962).“A computer for solving linear simultaneous equations using the residue number system,” IRE Trans. Electron. Comput. vol. 11, no. 2, pp. 164–173, Apr. 1962.
4. Han, S., Mao, H., & Dally, W. J. (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and human coding. arXiv preprint arXiv:1510.00149, 2015.
5. Henk C. A, Sushi Jajodia, (2011) Chinese Remainder Theorem Encyclopedia of Cryptography and Security pp.201-202. (2011). 978-1-4419-5905-8. Springer US.
6. Hiasat, A. (2013) Residue number system to binary converter for the moduli set (2^n -1; 2^n +1), Journal of systems architecture, 2003.
7. Jidaw Systems (2013), GSM Operators in Nigeria. http://www.jidaw.com/telecom/gsm.html
8. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Andrew Y. N. (2011). Reading digits in natural images with unsupervised feature learning. In NIPS workshop on deep learning and unsupervised feature learning, volume 2011, pp. 5.
9. NCC (July 2018), Subscriber Statistics.
10. Omondi, A., and Premkumar, B., (2007). Residue Number Systems: Theory and Implementations. London: Imperial College Press, 2007.
11. Piestrak, S. J. (1994) Design of residue generators and multioperand modular adders using carry-save adders. IEEE Transactions on Computers, 43(1):68{77, Jan1994. ISSN 0018-9340. doi: 10.1109/12.250610.
12. Sousa, L. (2007) Efficient method for magnitude comparison in RNS based on two pairs of conjugate mod-
13. uli. In Computer Arithmetic, ARITH'07. 18th IEEE Symposium on, pp. 240{250. IEEE, 2007.
14. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., & Zou, Y. (2016) DoReFa- Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprintarXiv:1606.06160.
15. Zimmermann, R. (1999) Efficient VLSI implementation of modulo (2^{n1} ± 1) addition and multiplication In Computer Arithmetic, Proceedings. 14th IEEE Symposium on, pp. 158{167. IEEE.