The application of modular arithmetic for matrix calculations

Viktor Kuchukov

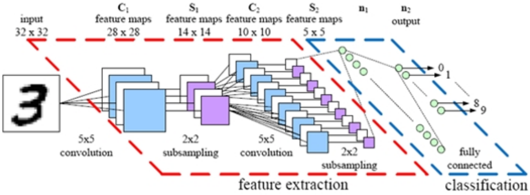
junior researcher
vkuchukov@ncfu.ru

Mikhail Babenko
PhD in Physico-mathematical sciences
Associate Professor
mgbabenko@ncfu.ru

IVANNIKOV ISP RAS OPEN CONFERENCE
MOSCOW, 5-6 DECEMBER, 2019

Array and Matrix

- Digital signal processing
- Artificial Neural Networks



Scientific challenge

- In critical applications, reliable computing systems with the ability to detect and correct errors are required.

Residue Number System (RNS)

In the RNS any number $X \in [0, M)$ have unambiguously represented by a tuple of residues x_i, where for all $i = [1, n]$ x_i is remainder of the division of X by p_i, p_i are coprime numbers (moduli), i.e. $x_i = X \bmod p_i$, $M = \prod_{i=1}^{n} p_i$ — is the dynamic range.

For numbers $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ is executed

$$C = A * B = (a_1 * b_1, a_2 * b_2, \ldots, a_n * b_n)\,, \text{ where } * = \{+, -, \times\}\,.$$

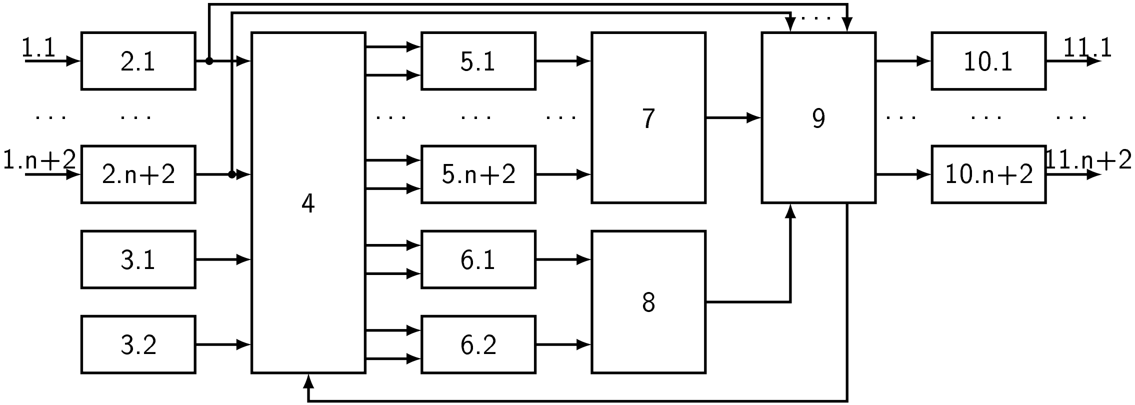
For a representation of negative numbers, the dynamic range is divided into equal parts, and it is possible to represent unambiguously any number X satisfying one of the expressions $\frac{-M-1}{2} \leq \ X \leq \frac{M-1}{2}$ for odd M and $\frac{-M}{2} \leq \ X \leq \frac{M}{2} - 1$ for even M.

To detect and correct an error, two redundant moduli p_{n+1} and p_{n+2} are added to RNS, and the dynamic range of the RRNS will be $P = \prod_{i=1}^{n+2} p_i$.

The number $X = (x_1, x_2, \dots, x_n, x_{n+1}, x_{n+2})$, is valid if $X \in [0, M)$ (legitimate range), but in the case of $X \in [M, P)$ (illegitimate range) it can be said that the number contains an error.

M is represented in the RRNS and it is obvious that $M = (0, \dots, 0, m_{n+1}, m_{n+2})$, where $m_{n+1} = M \bmod p_{n+1}$, $m_{n+2} = M \bmod p_{n+2}$.

Scheme for detecting, localizing and correcting errors based on RRNS

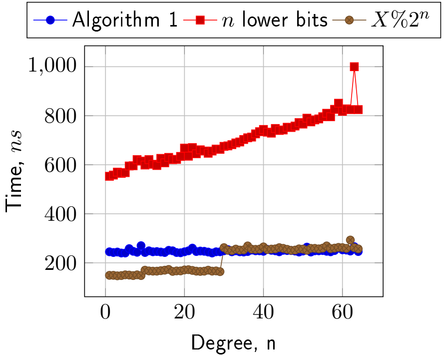


Patent RU2653257 “Modular code error detection and correction device”

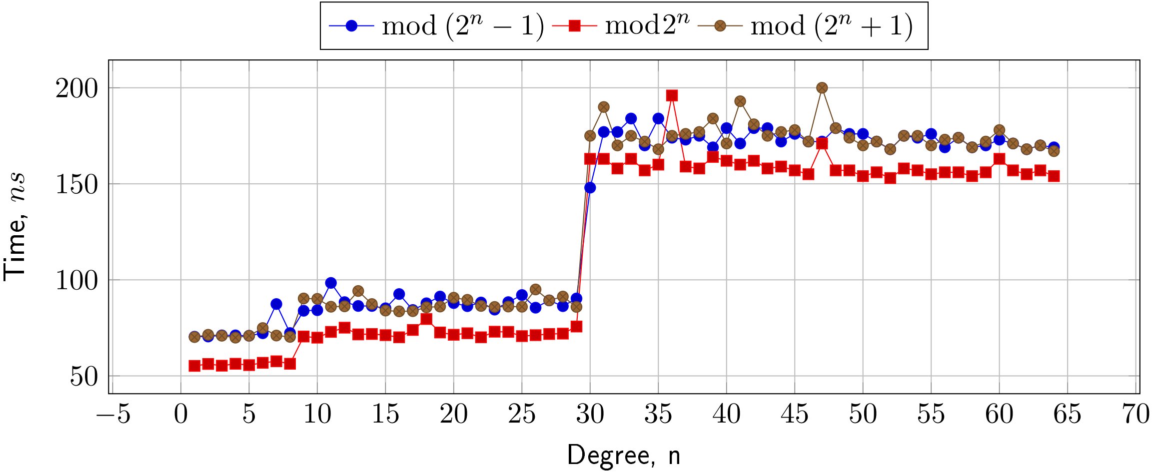
Algorithm 1 Operation $\bmod 2^n$ using the logical AND operator

Input: $X = (x_1, x_2, \ldots, x_n)$, 2^n.
Output: $X \bmod 2^n$.
return $X\&(2^n - 1)$

The modeling of the algorithms took place on a personal computer with a processor Intel i5 and 24 gigabytes of RAM in the programming language Python.



Comparison of methods of finding the remainder of the division by module 2^n.



Comparison of runtime for moduli set $\{2^n-1, 2^n, 2^n+1\}$.

For $X = (x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2})$ with moduli set $\{p_1, p_2, \ldots, p_n, p_{n+1}, p_{n+2}\}$ there are various methods of translation, for example, the method based on the Chinese Remainder Theorem (CRT), the approximate method based on CRT, the method based on the Mixed-Radix Conversion (MRC).

CRT

$$X = \left| \sum_{i=1}^{n+2} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P , \quad (1)$$

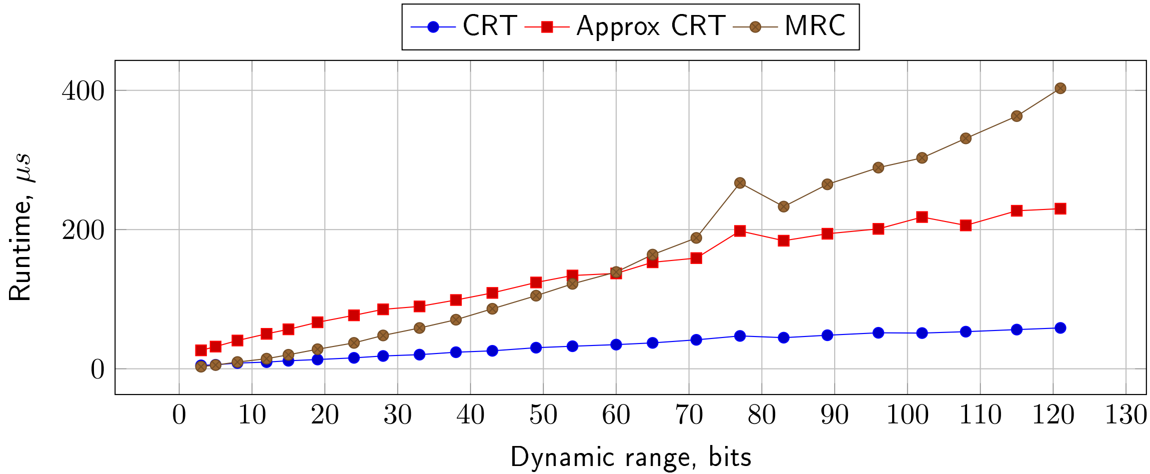
where $P = \prod_{i=1}^{n+2} p_i$ $P_i = P/p_i$, $\left| P_i^{-1} \right|_{p_i}$ — the multiplicative inversion of P_i by module p_i.

Approximate CRT

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \cdot \frac{\left| P_i^{-1} \right|_{p_i}}{p_i} \right|_1 = \left| \sum_{i=1}^{n} x_i \cdot k_i \right|_1 , \quad (2)$$

where $k_i = \frac{\left| P_i^{-1} \right|_{p_i}}{p_i}$

Convert from RNS



Comparison of methods of transfer from RNS to positional numeral system.

Scalar product

Algorithm 2 Calculation of the scalar product

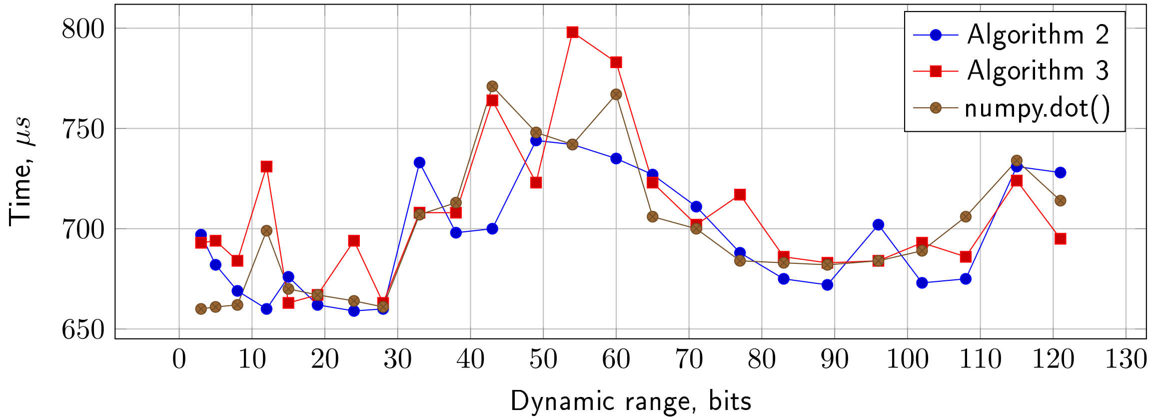
Input: $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$.
Output: $S = \langle X, Y \rangle$.

S=0
for $i = 1$ to n **do**
  $S = S + x_i \cdot \ y_i$
end for
return S

Algorithm 3 Calculation of the scalar product with a binary tree

Input: $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$.
Output: $S = \langle X, Y \rangle$.

for $i = 1$ to $\lceil \frac{n}{2} \rceil$ **do**
  $S_{0,i} = x_{2i-1} \cdot y_{2i-1} + x_{2i} \cdot y_{2i}$ *//parallel*
end for
for $i = 1$ to $\lceil log_2 n \rceil - 1$ **do**
  for $j = 1$ to $\lceil n/2^{i+1} \rceil$ **do**
    $S_{i.j} = S_{i-1,2j-1} + S_{i-1,2j}$ *//parallel*
  end for
end for
return $S_{\lceil log_2 n \rceil - 1, 1}$



Comparison of scalar product algorithms.

Conclusion

- The simulation of translation operations in the RNS showed the effectiveness of standard Python language tools.
- Hardware-based methods of translation from RNS in software implementation showed worse results than the method based on the Chinese Remainder Theorem.
- It can be concluded that the interaction of FPGA with computers, RNS allows achieving the required level of reliability of calculations.
- Further development of the research will be directed to the application of the residue number system for artificial neural networks and the implementation of the obtained parallel algorithms by means of GPUs on CUDA.

Thanks for your attention

Viktor Kuchukov

junior researcher
vkuchukov@ncfu.ru

Mikhail Babenko
PhD in Physico-mathematical sciences
Associate Professor
mgbabenko@ncfu.ru