ON THE APPLICATION OF RESIDUE NUMBER SYSTEMS TO NEURAL NETWORKS AND CRYPTOGRAPHY

DASIP 2018 - CONFERENCE ON DESIGN AND ARCHITECTURES FOR SIGNAL AND IMAGE PROCESSING

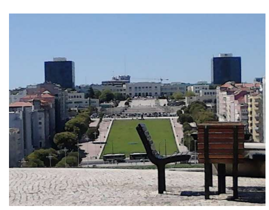
Leonel Sousa, Paulo Martins

October 2018

Where do I come from?

- IST
  - Faculty of Engineering / University of Lisbon
  - ~9000 / ~55000 students

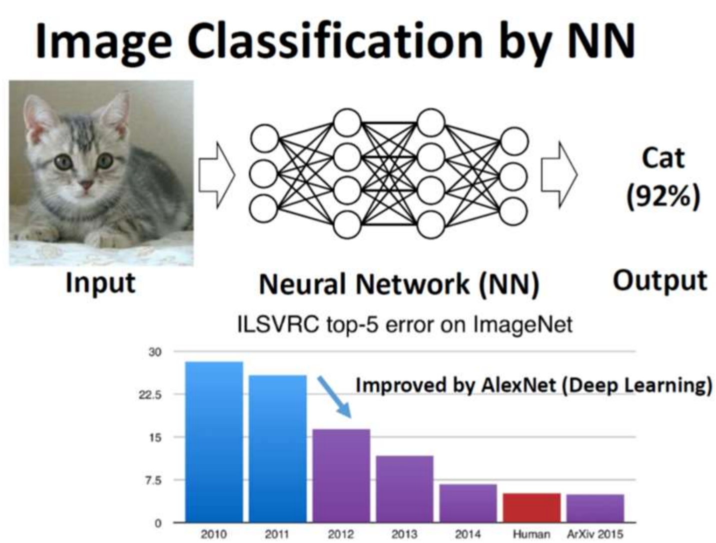
- INESC-ID
  - Research institute
    - 200 PhD Researchers
    - 300 Graduate Students



Motivation

Block Chain: Public Key Cryptography is an essential part of Bitcoin's protocol ensure the integrity of messages created in the protocol

Motivation

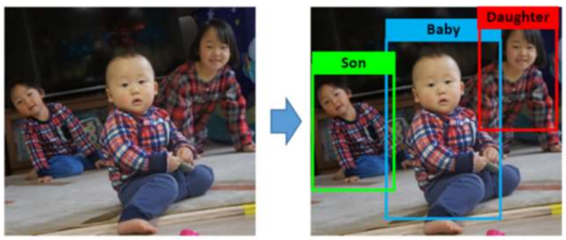


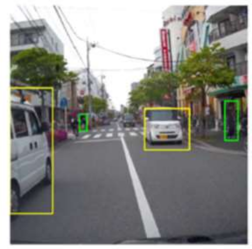
Motivation

Deep Learning: Convolutional Neural Networks

Object Detection

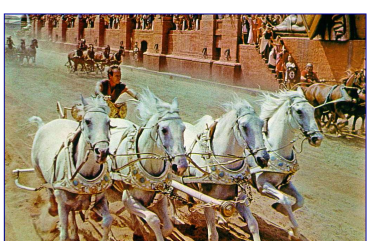
- Detect multiple objects at a time
- High performance-power is necessary





Motivation: Computer Arithmetic

Binary System
Carry chains limit arithmetic scaling



Residue Number System (RNS)
Computation over several channels

Outline

- Residue Number Systems (RNS)
- Public-key Cryptography: RSA, ECC, Lattice-based PQ approach
- Convolutional Neural Networks (CNN)
- Design Automation Tools
- On-going research projects
- Conclusions

RNS: Residue Number System

Based on the Chinese remainder theorem (CRT) widely used for computing with large integers: it allows replacing a computation for which one knows a bound on the size of the result by several similar computations on small integers

C. Chang, A. Molahosseini, A. Zarandi, and T. Tay, "Residue Number System - A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications," IEEE Circuits and Systems Magazine, vol. 15, no. 4, pp. 26-44, November 2015

Residue Number System (RNS)

- RNS based on a set of relatively prime moduli: **moduli set**

$$P = \langle m_1, m_2, \cdots, m_N \rangle$$

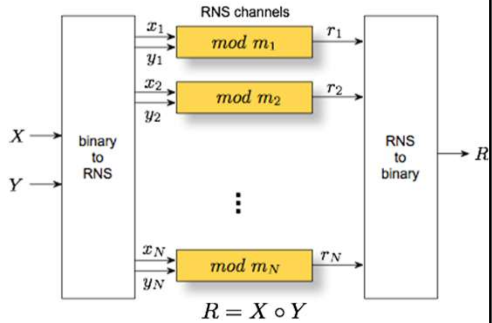
- The dynamic range M is given by:

$$M = m_1 \times m_2 \times \cdots \times m_N$$

- Integer X represented as:

$$X \longrightarrow \{x_1, x_2, \cdots, x_N\}$$
$$x_i = X \ mod \ m_i$$

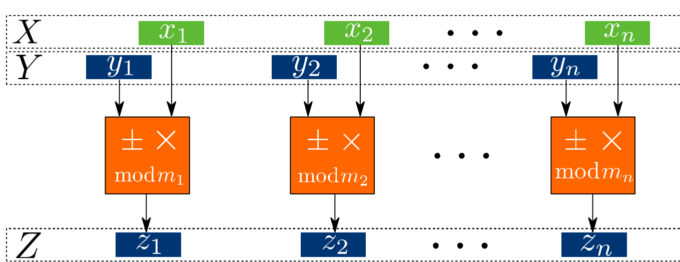
Arithmetic operations (+,-,x,/):



$$\{r_1, r_2, \cdots r_N\} = \{(x_1 \circ y_1) \ mod \ m_1, (x_2 \circ y_2) \ mod \ m_2, \cdots, (x_N \circ y_N) \ mod \ m_N\}$$

RNS

- Parallelism extracted at arithmetic level
- RNS splits arithmetic modulo $M = m_1 m_2 \ldots m_n$ over n rings



RNS

Advantages

- Carry-free between channels
- Fast parallel $+$, $-$, $\times$ and exact $\div$
- Enables side-channel attacks counter measures

Disadvantages

- B2R, R2B , division and modular reduction hard to perform

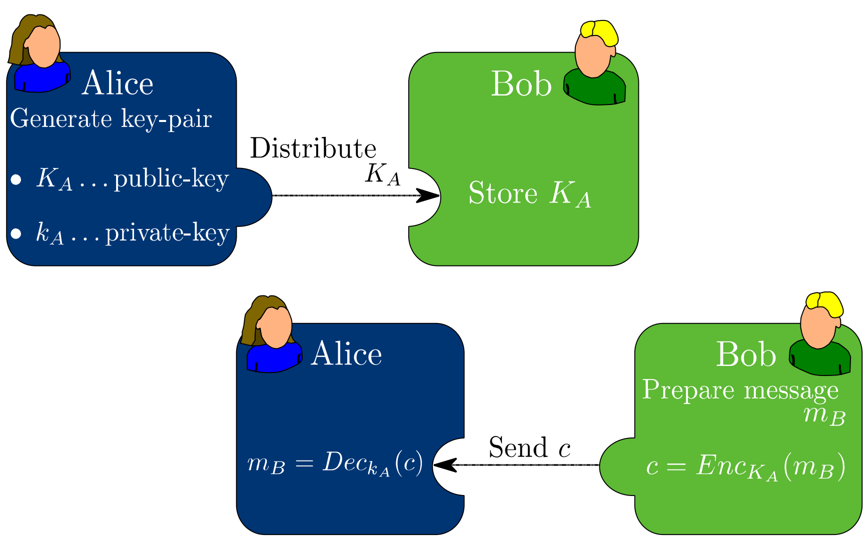
Public-key Cryptography

L. Sousa, S. Antão and P. Martins, "Combining Residue Arithmetic to Design Efficient Cryptographic Circuits and Systems", IEEE Circuits and Systems Magazine, vol. 16, n. 4, pp. 6-32, November 2016.

Cryptographic Algorithms

- Asymmetric/Public Key Encryption algorithms
  - Cipher and decipher a block of data using a private/public key pair
  - Complex mathematical operations, computationally demanding
  - RSA, Elliptic curves, ElGamal, Post-Quantum lattice based crypto
- Digital Signatures
  - Identical to Public Key Encryption algorithms
  - Digital Signature Algorithm (DSA), RSA, Elliptic Curve Digital Signature Algorithm (ECDSA)…

Public-key Cryptography



RSA

RSA Encryption and Decryption operations

$$c = m^e (\text{mod } N) \longrightarrow m = c^d (\text{mod } N)$$

N is thousands of bits wide (e.g. 4096 bits)

$$m = \left(\dots \left(\left(c^{d_{k-1}} \right)^2 c^{d_{k-2}} \right)^2 \dots \right)^2 c^{d_0} \ (\text{mod } N)$$

Sequential computation

Modular Multiplication

Find Q

$Q_{n-1}, Q_{n-2}, \dots, Q_0$

$N_{n-1}, N_{n-2}, \dots, N_0$

$$\begin{array}{rl} & X_{2n-1}, \dots, X_n, X_{n-1}, \dots, X_0 \\ - & Z_{2n-1}, \dots, Z_n, Z_{n-1}, \dots, Z_0 \\ \hline \end{array}$$

s.t. MSBs of X' are 0

$$X' < 2N$$

Montgomery Modular Multiplication

Montgomery Domain

$${}_{M}X = XM (\text{mod } N)$$

Arithmetic modulo *N* is converted to modulo *M*

$$W = {}_{M}X \; {}_{M}Y$$

$$Q = -WN^{-1} \text{ mod } M$$

$${}_{M}Z = \frac{W + QN}{M} \equiv XYM \; (\text{mod } N)$$

$${}_{M}Z < 2N$$

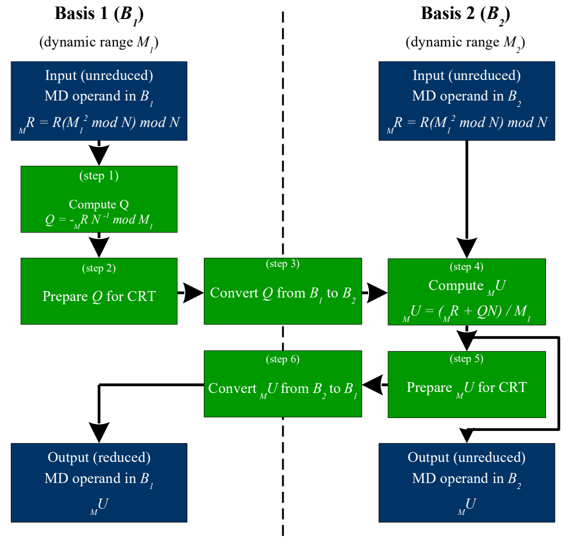
RNS Montgomery Modular Multiplication

- Last part of computation cannot be performed modulo M:

$${}_{M}Z = \frac{W + QN}{M}$$

- Convert $Q = -WN^{-1} \text{ mod } M$ to the second base
- At the end, ${}_{M}Z$ is converted back to the first base

RNS Montgomery Modular Multiplication



RSA

RSA Encryption and Decryption operations: revisited

- ${}_Mc \; = cM_1 \bmod N$
- ${}_Mm \; = M_1 \bmod N$

for $i \leftarrow \{k-1, \ldots, 0\}$

- ${}_Mm \; = \mathrm{MM}({}_Mm, {}_Mm)$ ← Parallel computation

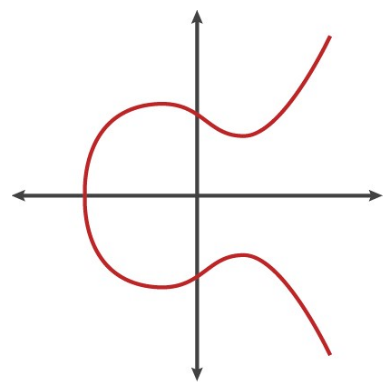
  if $d_i = 1$

  - ${}_Mm \; = \mathrm{MM}({}_Mm, {}_Mc)$ ← Parallel computation

- $m = {}_MmM_1^{-1} \bmod N$

Elliptic Curve Cryptography (ECC)

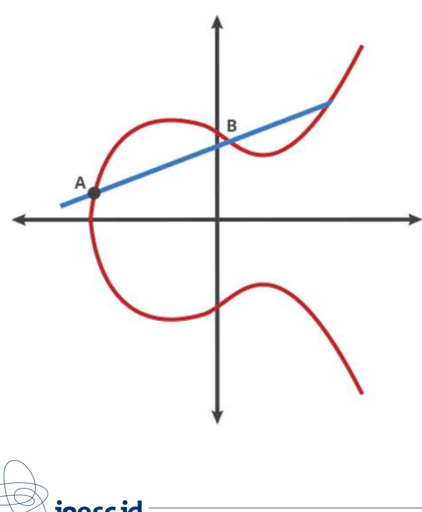
$y^2 = x^3 + ax + b$



- In cryptography, these curves are defined over a finite field $\mathbb{F}_q$

ECC

$C = A + B$

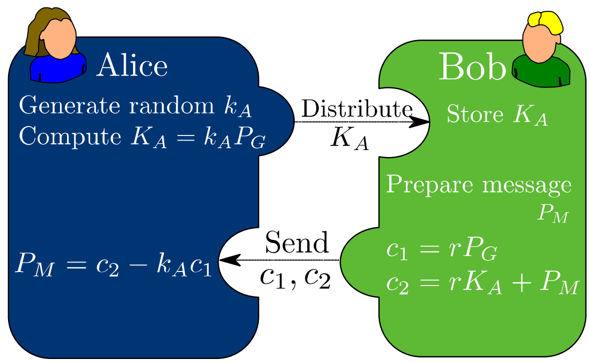


- Point multiplication is defined as

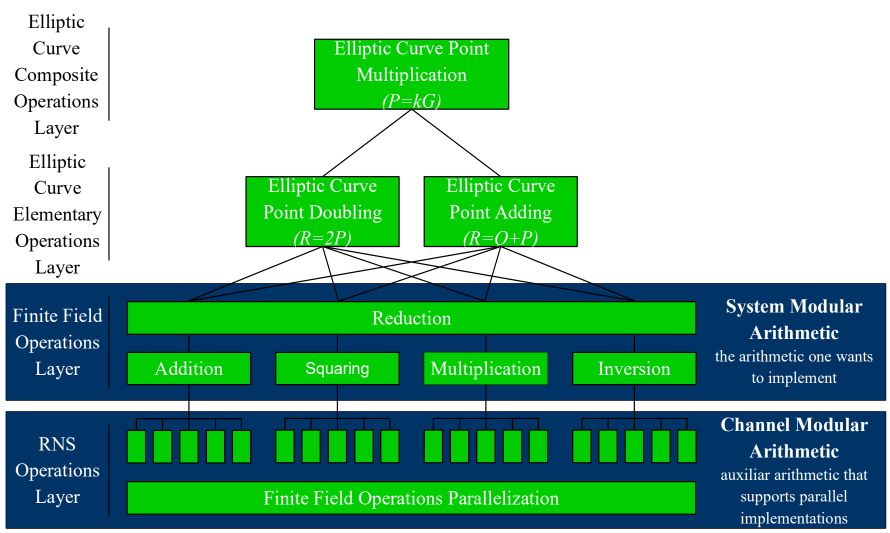
$$Q = sP = \underbrace{P + \cdots + P}_{s\ times}$$

- Given Q and P, it is hard to compute s

ECC Encryption/Decryption Protocol



ECC



ECC: Reducing Bases' Sizes

Mersenne-like Prime

$$q = M_1^2 - 2$$

Decompose X as

$$X = K_X M_1 + R_X$$

$M_1^2 = 2 \bmod q$ leads to

$$XY = (K_X R_Y + K_Y R_X) M_1 + 2K_X K_Y + R_X R_Y$$
$$= V M_1 + U \bmod q$$
$$U, V < 3M_1^2 < M_1 M_2 \Rightarrow \text{Smaller bases required}$$

RNS based EC Point Multiplication

Results for EC Point Multiplication in GPUs with CUDA

Reference	Platform	Lat. [ms]	T.Put [mults/s]	Observations
Szerwinski et. al.	8800 GTS	305	1413	
Bernstein et. al.	8800 GTS	-	3019	ECM factorization
Giorgi et. al.	9800 GTX	-	1972	Library eval.
RNS based	8800 GTS	30.3	3138	12 mults/block
RNS based	285 GTX	29.2	9827	20 mults/block

- An **order of magnitude** improvement in latency
- From **4% to 122% more** throughput.

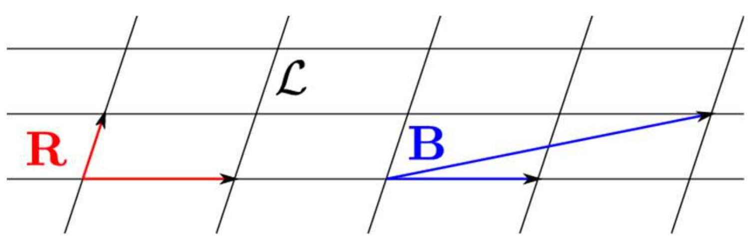
Post Quantum Cryptography

- P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", 35th Annual Symposium on Foundations of Computer Science, 1994

"…algorithms for finding **discrete logarithms** and **factoring integers** on a quantum computer that take a number of steps which is polynomial in the input size, e.g., the number of digits of the integer to be factored."

Lattice-based Cryptography

- A lattice is a repeated arrangement of points

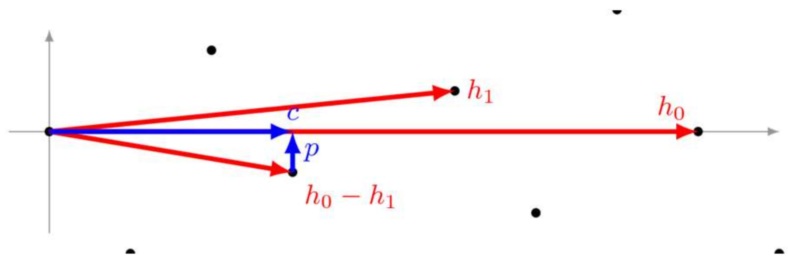


- Matrix $R = (r_1, \dots, r_l)^T$: a basis of $\mathcal{L}$
  - $\mathcal{L} = r_1\mathbb{Z} \oplus \dots \oplus r_l\mathbb{Z}$

- For $n \geq 2$, there are infinite basis

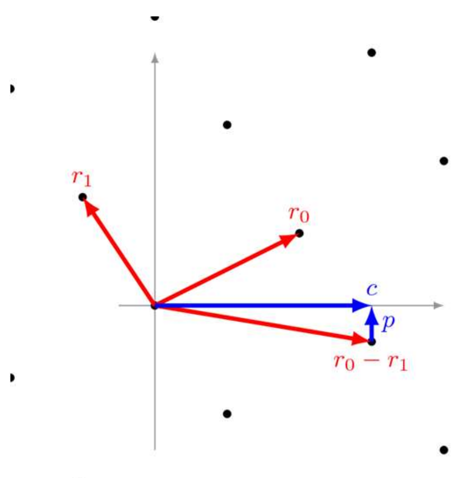
Lattice-based Cryptography

- Encryption corresponds to adding a perturbation p to a lattice point
- (h_0, h_1) is a "bad" lattice base



Lattice-based Cryptography

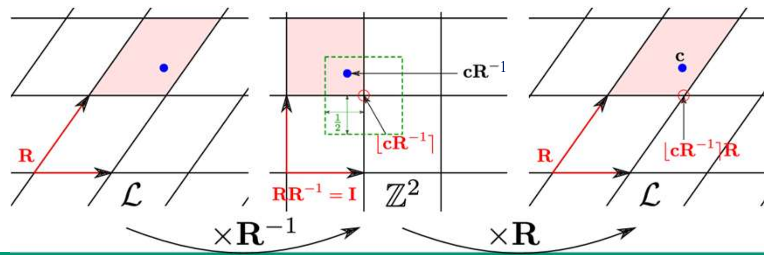
- Decryption corresponds to finding the closest lattice vector u to c and outputting $p = c - u$
- (r_0, r_1) is a "good" lattice base



Lattice-based Cryptography

Babai's Round-off Algorithm

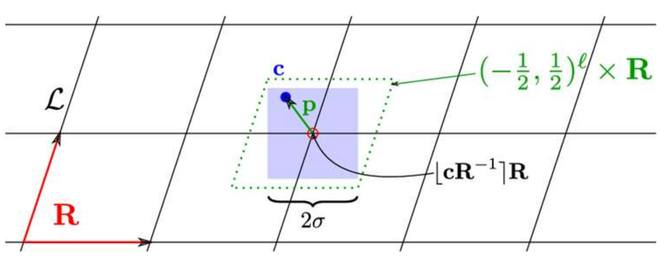
change of basis	rounding components	back to canonical basis
$c \times R^{-1}$	$\lfloor c \times R^{-1} \rceil$	$\lfloor c \times R^{-1} \rceil \times R$



Lattice-based Cryptography

Common Simplification Step

- Use special case of CVP: Bounded Distance Decoding Problem (BDD)
- Babai's Round-off gives the closest vector for a rotated nearly-orthogonal basis R of a lattice



$p = c - \lfloor cR^{-1} \rceil R \text{ modc } m_\sigma$ for $m_\sigma \geq 2\sigma + 1$

Lattice-based Cryptography

GGH-like cryptosystem

- Private-key: good R s.t. $HNF(R) = \left(\begin{array}{c|c} \det R & 0 \ldots 0 \\ \hline * & I_{n-1} \end{array}\right) = B$
- Public-key: B
- Plaintext space: $||p||_\infty \leq \sigma$
- Ciphertext: $c = p \bmod B = (c_1, 0, \ldots, 0)$
- Deciphering: $p = c - \lfloor c \times R^{-1} \rceil R =$
  $c - \left\lfloor c_1 \times \left((R^{-1})_{1,1}, \ldots, (R^{-1})_{1,n} \right) \right\rceil R$

Lattice-based Cryptography

- Babai's algorithm rewritten with integer arithmetic:
  - $u = \lfloor cR^{-1} \rceil R = \left\lfloor cR^{-1} + \frac{1}{2} \right\rfloor R = \left\lfloor \frac{dcR^{-1}}{d} + \frac{1}{2} \right\rfloor R =$

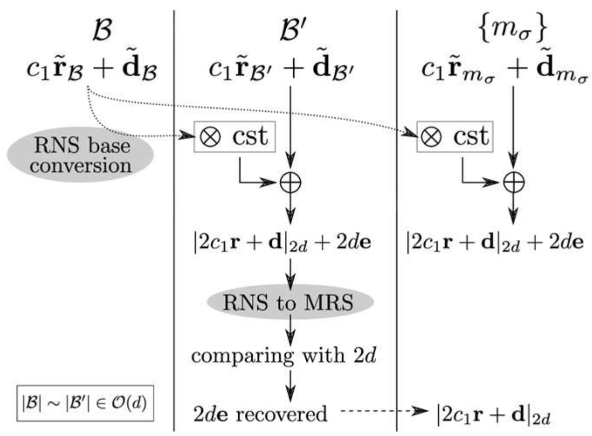
$$\frac{2cdR^{-1} + d - \left(2cdR^{-1} + d \bmod (2d)\right)}{2d} R$$

where $d = \det(R)$

Use RNS Montgomery's reduction

Lattice-based Cryptography

Principle of RNS Montgomery modular reduction



RNS based LBC decryption

Results for LBC decryption in CPUs/GPUs

Execution Times [$\times 10^6$ clock cycles] (Speed-up)				
Method	$n = 400$	$n = 600$	$n = 800$	$n = 1000$
Sequential (i7 4770K)	97.51	283.8	619.4	1222
RNS-GPU (K40c)	22.97 (4.2)	283.8 (3.6)	248.9 (2.5)	512.4 (2.4)
RNS-GPU (GTX 780 Ti)	16.55 (5.9)	59.73 (4.8)	148.2 (4.2)	349.6 (3.5)
4-core RNS-CPU (i7 4770K)	21.05 (4.6)	75.48 (3.8)	189.9 (3.3)	369.7 (3.3)
4-core RNS-CPU (with AVX2) (i7 4770K)	8.668 (11.2)	29.05 (9.8)	74.79 (8.3)	148.5 (8.2)

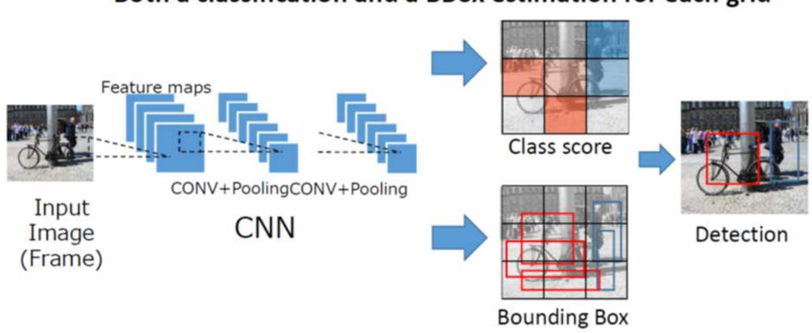
Convolution Neural Networks

H. Nakahara and T. Sasao, "A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018

Deep Learning: CNNs

YOLOv2
(You Only Look Once version 2)

- **Single CNN (One-shot) object detector**
  - **Both a classification and a BBox estimation for each grid**

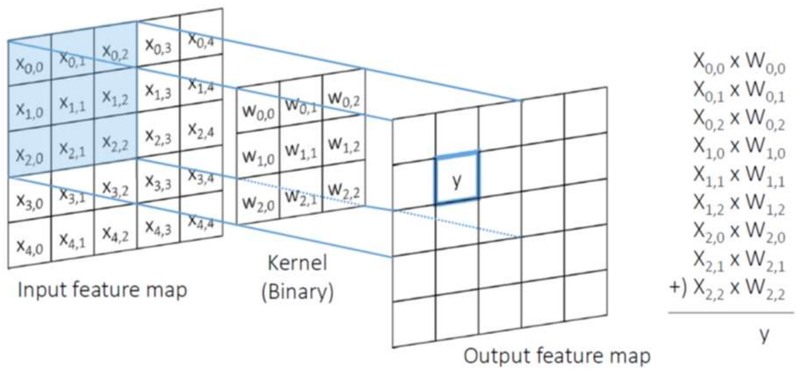


J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv preprint arXiv:1612.08242*, 2016.

Deep Learning: CNNs

2D Convolutional Operation

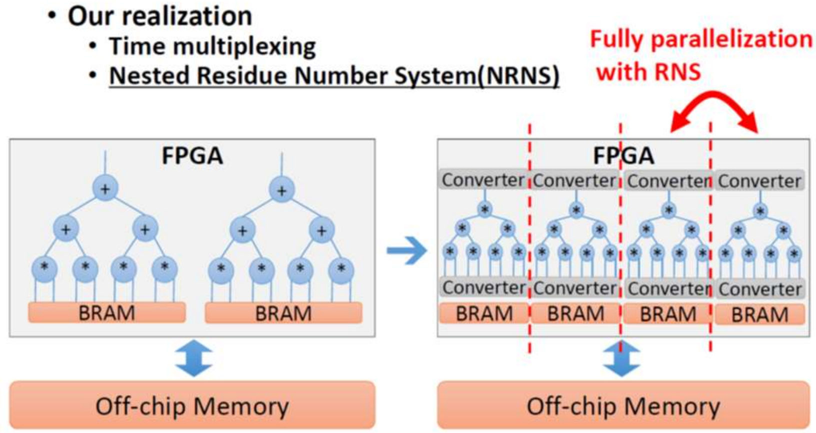
- Computational intensive part of the YOLOv2



Deep Learning: CNNs

Realization of 2D Convolutional Layer

- Requires more than billion MACs
- Our realization
  - Time multiplexing
  - Nested Residue Number System(NRNS)



Nested RNS

Nested RNS

- $(Z_1, Z_2, \ldots, Z_i, \ldots, Z_L) \rightarrow (Z_1, Z_2, \ldots, (Z_{i1}, Z_{i2}, \ldots, Z_{ij}), \ldots, Z_L)$
- Ex: <7,11,13>×<7,11,13>

Original modulus

$<7,<5,6,7>_{11},<5,6,7>_{13}> \times <7,<5,6,7>_{11},<5,6,7>_{13}>$

1. **Reuse** the same moduli set
2. **Decompose** a large modulo into smaller ones

Nested RNS

Example of Nested RNS

- 19x22(=418) on $<7,<5,6,7>_{11},<5,6,7>_{13}>$

19×22

=<5,8,6>×<1,0,9>

Binary2NRNS Conversion

$=<5,<3,2,1>_{11},<1,0,6>_{13}> \times <1,<0,0,0>_{11},<4,3,2>_{13}>$

Modulo Multiplication

$=<5,<0,0,0>_{11},<4,0,5>_{13}>$

Bin2RNS on NRNS

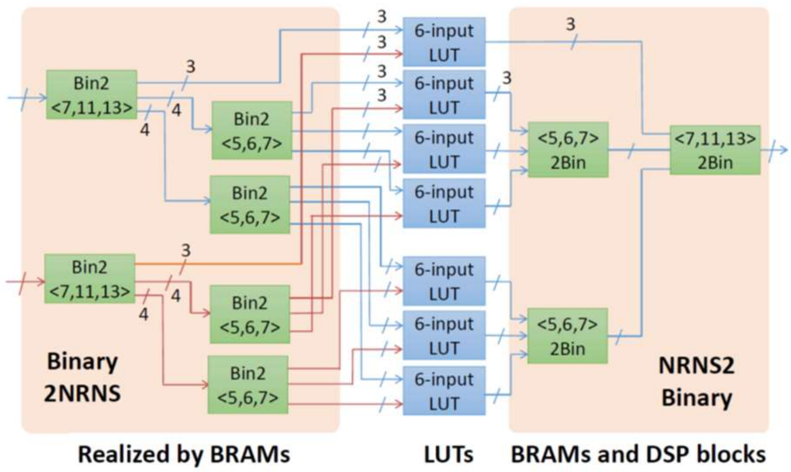
=<5,0,2>

RNS2Bin

=418

Nested RNS

Realization of Nested RNS



NRNS based YOLOv2

NRNS based YOLOv2

- Framework: Chainer 1.24.0
- CNN: Tiny YOLOv2
- Benchmark: KITTI vision benchmark
- mAP: 69.1 %

Layer	# In. Fmaps	# Out. F Size
(Feature Extraction)		
Conv1	3	128×128
Conv2	128	128×128
Max Pool	128	64×64
Conv3	128	64×64
Conv4	128	64×64
Conv5	128	64×64
Max Pool	128	32×32
Conv6	128	32×32
Conv7	128	32×32
Conv8	128	32×32
Max Pool	128	16×16
(Localization+Classification)		
Conv9	128	16×16
Conv10	128	16×16
Conv11	128	$5^2 \times 3 + (5 \times 5)$
Accuracy (mAP)	69.1	

Implementation

Implementation

- FPGA board: NetFPGA-SUME
  - FPGA: Virtex7 VC690T
  - LUT: 427,014 / 433,200
  - 18Kb BRAM: 1,235 / 2,940
  - DSP48E: 0 / 3,600
- Realized the pre-trained NRNS-based YOLOv2
  - 9 bit fixed precision (dynamic range: 30 bit)
- Synthesis tool: Xilinx Vivado2017.2
  - Timing constrain: 300MHz
  - 3.84 FPS@3.5W → 1.097 FPS/W

Evaluation

Comparison

	NVivia Pascal GTX1080Ti	NetFPGA-SUME
Speed [FPS]	20.64	3.84
Power [W]	60.0	3.5
Efficiency [FPS/W]	0.344	1.097

Design Automation

S. Antão and L. Sousa, "The CRNS framework and its application to programmable and reconfigurable cryptography," ACM Transactions on Architecture and Code Optimization, vol. 9, no. 4, pp. 33:1–33:25, 2013.

Design Automation

Implementation Details

- For a processor with word length of $2l$ bits, choose moduli of the form:

$$m_i = 2^l - c_i$$

- The following congruence is valid:

$$z = z_0 + 2^l z_1 \equiv z_0 + c_i z_1 \pmod{m_i}$$

- Enabling fast reductions if c_i is small

Design Automation

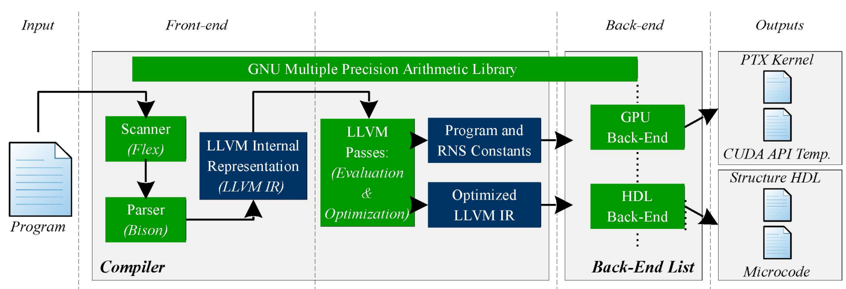
Implementation Details

- To maximize the hardware reuse in Montgomery's Reduction algorithm:
  - Associate each Processing Element (PE) with a channel from the first base and with a channel from the second base
  - PEs can be SIMD channels on CPUs, threads on GPUs, or hardware elements on FPGAs

Design Automation

Compute with Residue Number System

- Obtains in seconds a fully functioning RNS based implementation
- RNS details are transparent to the user

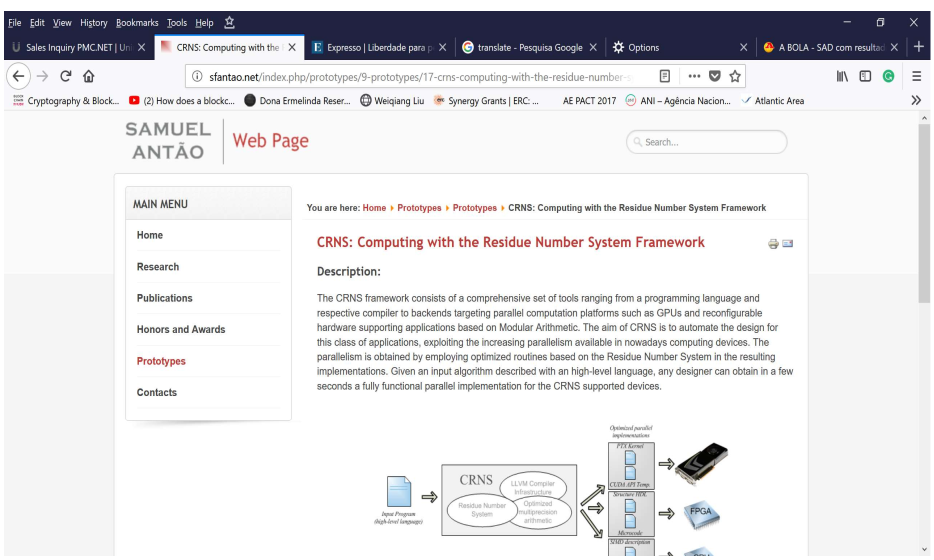


Compute with Residue Number System (DSL)

- Algorithm described using an extended version of C
- Available at http://sfantao.net/index.php/prototypes

```
(outputA) = (int1024 inputA)
{
  /*modulo:*/
  int N = 0xbf4325a19 ...  c92e820f7;
  /*exponent:*/
  int D = 0x922ad9c67 ...  c1c856434;
  /*mask:*/
  int M = 0x400000000 ...  000000000;

  /*compute the exponentiation*/
  int I = inputA % N;

  int R = I;
  while(M)
  {
    int M2 = M & D;

    R = (R * R) % N;

    if(M2)
      R = (R * inputA) % N;

    M = M >> 1;
  }

  outputA = R;
}
```


CRNS



On-going Projects

https://futuretpm.eu/

FutureTPM (2018-2020)

- FutureTPM aims to design QR cryptographic algorithms
  - Symmetric Cryptography
  - Asymmetric Cryptography
  - Privacy-protecting primitives, such as Direct Anonymous Attestation
- Hardware, **Software**, and Virtual TPM FutureTPM
- Standardization within TCG, ISO/IEC and ETSI
- Run-Time Risk Assessment and Vulnerability Analysis
  - E.g. via side-channel attacks

Post qUaNtum Cryptography TooLbox (PUNCTuaL)
PUNCTual(2018-2019): UPMC-ULisboa

- Cryptographic solutions mainly based on public key using the RSA and ECC primitives.
  - These primitives are insecure if a quantum computer is produced
- PUNCTuaL provides tools and methodologies for PQ cryptography on which secure and efficient protocols can be underpinned
- Design of the tools will be based on industrial use-cases
  - protocols used to secure the digital market, telecommunications and the Internet of Things (IoT).

Conclusions

- Residue Nmber System improves performance of multiple cryptosystems
- RNS helpful in updating inefficient RSA systems to ECC, and also in moving to quantum-resistant cryptography
- RNS is also helpful for Attificial Intelligence, namely CNNs
- Applicable to a wide-range of platforms, since channel bit-width can be tailored for the target architecture (also embedded systems)
  - (e.g. GPUs, CPUs with SIMD, FPGAs)


Thank You
for your attention!
technology
from seed
inesc id
lisboa
Instituto de Engenharia de Sistemas e Computadores
Investigação e Desenvolvimento em Lisboa
58