UNCONVENTIONAL COMPUTER ARITHMETIC FOR EMERGING APPLICATIONS AND TECHNOLOGIES

IEEE COMPUTER SOCIETY DISTINGUISHED VISITORS PROGRAM (DVP)

https://www.computer.org/web/chapters/dvp

Leonel Sousa
Webinar, July 9, 2020
UNCONVENTIONAL COMPUTER ARITHMETIC FOR EMERGING APPLICATIONS AND TECHNOLOGIES

IEEE COMPUTER SOCIETY DISTINGUISHED VISITORS PROGRAM (DVP)

https://www.computer.org/web/chapters/dvp

Leonel Sousa
Webinar, July 9, 2020
From Lisbon

- IST (Head of the ECE Dept)
  - Faculty of Engineering University of Lisbon
  - ~9000 / ~55000 students

- INESC-ID
  - Research institute
    - 200 PhD researchers and
    - 300 Graduate Students
  - Main research areas
    - Spoken Language Systems
    - Information and Decision Support Systems
    - Interactive Virtual Environments
    - Embedded Electronic Systems
    - Communication Networks and Mobility
Motivation

Cost, speed, size, and energy/operation encoded by color [ITRS03]

- No solution with few major drawbacks as CMOS along all axes
  - spin transistors, superconducting electronics, molecular electronics, resonant tunneling devices, QCA, and optical switches
Nano processing technologies [ITRS15-Beyond CMOS]

- scaled CMOS
  - FinFET non-planar transistor dominant gate design for current 7 nm
- emerging nano-technologies require unconventional number systems & computer arithmetic
Confluence of non-conventional computer arithmetic, new computing paradigms, emergent technologies and applications

- jigsaw puzzle: connecting pieces in the right way to get the whole picture
Outline

1. Logarithmic Residue Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications:
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
LNS

1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications:
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
Simple logarithmic operations come at the cost of more complex +,-

\[
\begin{align*}
\log_b(P \times Q) &= p + q \\
\log_b(P/Q) &= p - q \\
\log_b(P^2) &= 2 \times p = \text{BitShift}(p, 1) \\
\log_b(\sqrt{P}) &= \frac{1}{2} \times p = \text{BitShift}(p, -1)
\end{align*}
\]
LNS

• Addition/subtraction in LSN apply Gaussian Logarithms

\[ G = \log_2(1 \pm 2^\lambda), \quad \lambda = -|q - p| \]

• For high number of bits (32, 64) piecewise polynomial approximation or digit-serial iterative methods are applied
• For subtracting in the LNS domain, co-transformations have to be applied in the critical region \( \lambda [-1, 0] \)
European Logarithmic Microprocessor (ELM)

- 32-bit scalar microprocessor, Register-Memory ISA
  - 16 general-purpose registers, 8 kB L1 data cache
  - two real adders/subtractors operating in 3 clock cycles
  - four combined multiplier/divider/sqrt/integer units operating in 1 clock cycle
  - vector operations use in parallel 4 functional units

- **Fixed-point LNS-based AU**
  - Sign bit and 23 bits fractional component
  - Taylor interpolation for addition and subtraction

- Fabricated with 0.18μm CMOS running at 125MHz, is evaluated against the TMS320C6711 contemporary DSP
  - addition marginally better *multiplications 3.4x faster*
  - division and square root several times faster
LNS: Convolution Neural Networks

• Application of LNS on CNNs allows activation and weights with only 3 bits
  – with almost no loss in classification performance

\[
\text{conv} = \sum_i 2^{\tilde{x}_i + \tilde{w}_i} = \sum_i \text{BitShift}(1, \tilde{x}_i + \tilde{w}_i)
\]

• Accumulation can be done also in the log domain with the approximation

\[
\log_2(1 + x) \approx x \text{ for } 0 \leq x < 1.
\]
1. Logarithmic Number Systems (LNS)

2. **Residue Number Systems (RNS)**

3. Stochastic Computing (SC)

4. Hyper-Dimensional Computing (HDC)

5. DNA Computing

6. Quantum Computing

7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning

8. Conclusions
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:

\[ X \rightarrow \{ x_1, x_2, \cdots, x_N \} \]

\[ x_i = X \mod m_i \]

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

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

\[
\begin{bmatrix}
0 & 209 & 0 \\
209 & 4 & 209 \\
0 & 209 & 0
\end{bmatrix}
\]

RNS Reverse Conversion

\[
\begin{bmatrix}
0 & 4 & 0 \\
4 & 4 & 4 \\
0 & 4 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 5 & 0 \\
5 & 4 & 5 \\
0 & 5 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 6 & 0 \\
6 & 4 & 6 \\
0 & 6 & 0
\end{bmatrix}
\]
RNS: Photonics

- 2 x2 Hybrid Photonic-Plasmonic (HPP) integrated switches
  - fabricated by using Indium Tin Oxide as index modulation material
  - voltage signal controls guidance of light (may operate at 400 GHz), speed is defined by modulators, photodetectors and electronics

- RNS Parallelism (# switches grows with $N^2$) and energy efficiency of integrated photonic high-speed RNS units
RNS

• R(edundant)RNS is used for error detection/correction
  – residues are independent, by introducing redundant moduli, the range of the legitimate moduli is extended to an illegitimate one

• The Processing for Y'all (CREEPY) [2018] core microarchitecture and ISA integrates RRNS centered algorithms and techniques to efficiently assure computational error correction.
  – significant improvements over a non-error correcting binary core
  – novel schemes proposed also for RNS based memory access, extend low power and energy efficient RRNS based architectures
1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. **Stochastic Computing (SC)**
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
• From a continuous-time stochastic process, the value of a bitstream is the #‘1’ bits over the total #bits (9/15=0.6)

\[
\begin{array}{cccccccccccc}
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]

• Multiplication

\[
\begin{array}{c}
I_1 \quad 0;1;1;0;1;0;1;0 (4/8) \\
I_2 \quad 1;0;1;1;1;0;1;1 (6/8) \\
\end{array}
\]

\[
\begin{array}{c}
0;0;1;0;1;0;1;0 (3/8) \\
\end{array}
\]

• Addition

\[
\begin{array}{c}
I_1 \quad 1;1;1;1;1;0;1;1 (7/8) \\
I_2 \quad 0;0;1;0;0;1;1;0 (3/8) \\
I_3 \quad 1;0;0;1;0;1;0;1 (4/8) \\
\end{array}
\]

\[
\begin{array}{c}
1;0;1;0;1;0;1;1 (5/8) \\
\end{array}
\]

• Correctness impacted by correlation between bitstreams
  – e.g. the same stream at the 2 inputs of the * produces the same stream at the output, instead of the square
**SC**

- Converters SC->Binary  Binary->SC

  ![Diagram showing the conversion process]

- Highly non-linear functions (e.g. tanh and max functions in ANNs) require FSM-based SC elements

  ![Diagram of a stochastic counter]

---

Inescid Lisboa
SC-based CMOS Invertible Logic

- Boltzmann Machine

\[
\begin{align*}
    m_i(t + \Delta t) &= \text{sgn}(\text{rnd}(-1, +1) + \tanh(I_i(t + \Delta t))) \\
    I_i(t + \Delta t) &= I_0(h_i + \sum J_{ij} m_j(t))
\end{align*}
\]  

(36)

AND gate

- 5 by 5-bit */divider/factorizer 13x less area than binary for the TSMC 65nm technology
SC: Superconducting Quantum Device

- Adiabatic Quantum-Flux-Parametron logic
  - energy efficiency: ALU RISC V 10x lower energy than CMOS 12nm

- Two characteristics AQFP suitable to implement SC
  - deep pipelining: gate is connected with AC clock signal requiring a clock phase, difficult to avoid RAW hazards with binary computing
  - The opportunity of true RNG using simple buffers
Stochastic Recognition and Mining (StoRM) Processor

- 2D array of Stochastic PE (typically 15x15)
- Binary-to-stochastic units shared across rows/columns
- Implementation on TSMC 65nm: one order of magnitude less circuit area and power consumption
Hyper-Dimensional Computing (HDC)

1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
HDC

• HDC inspired in brain-like operation
  – supported on random high-dimensional vectors, in the order of thousands of bits (10,000-bit vector)
  – alternative to SVM and CNN for supervised classification
  – Associate Memories (AM): pattern X is stored using pattern A as the address, latter X can be retrieved from A or A’ similar to A

• The high number of bits does not improve resolution
  – tolerant to errors and component failure, many patterns equivalent
  – highly structured information, like in the brain, deals with arbitrariness of the neural code
HDC: Arithmetic

- **Componentwise Addition** of a set: sum represents a set individual vectors

- **Multiplication implements** with bitwise logic XNOR
  - bipolar representation, (0, 1) -> (1, -1), $X^*Y = X \text{xnor } Y$
  - Multiplication maps points: $X^*M$ maps $X$ into $X_M$ that is as far from $X$ as the number of 1s in $M$;
    
    $M$ a random vector $\Rightarrow$ multiplication randomizes $X$

\[
d(X_M, X) = \| X_M \times X \| = \| M \times X \times X \| = \| M \|
\]

- multiplication is distributive over addition, and implements a mapping that preserves distance

\[
d(X_M, Y_M) = d(X, Y)
\]
HDC: Nanosystem

• End-to-end brain-inspired HDC nanosystem, using heterogeneous integration of multiple emerging nanotechnologies

  − Monolithic 3D integration of Carbon, Nanotube Field-Effect Transistors (CNFETs) and Resistive Random-Access Memory (RRAM)
  − fine-grained and dense vertical connections between computation and storage layers
  − Integrating RRAM and CNFETs allows to create area-and energy-efficient circuits
IM stores a large collection of random hyper-vectors (items)
- maps symbols to items in the inference phase as trained
- DPUUs combine hyper-vectors sequence according to the algorithm
  - to compose a single hyper-vector per each class.
AM stores the trained class hyper-vectors
- deliver the best prediction according to the Hamming distance ($d_h$).
DNA-based Computing

1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
DNA-based Computing

- With the DNA sticker model, a binary number represented through two groups of single-stranded DNA molecules
  - the memory strand, a long DNA molecule subdivided into non-overlapping segments
  - set of stickers, short DNA molecules, each with the length of a segment, a sticker is complementary to one of those segments
DNA-based Computing

• Example of the bitwise AND operation of 2 n-bit vectors

```
Algorithm 1 AND(Ts1, Ts2, n:in; Td:out)

Require: Pour blank strand of n bits (0...0) in Td
Ensure: bit_stream_in_Td = bit_stream_in_Ts1 \& bit_stream_in_Ts2
1: Combine(Ta, Ts1, Ts2) {Ta: auxiliary Tube}
2: for all bit 0 \leq i < n do
3:   Separate(Ta, i, B[1], B[0])
4:   if B[0] is empty then
5:     Set(Td, i)
6:   end if
7: Combine(Ta, B[1], B[0])
8: end for
```

• DNA ALU was constructed:
  – with 1-bit FA, AND, OR and NAND, decoding and controlling logic
RRNS DNA-based Computing

- RRNS has been applied for overcoming the negative effects caused by the defects and instability of the biochemical reactions and errors in hybridizations
  - applying the RRNS 3-moduli set \( \{2^{n-1}, 2^n+1, 2^{n+1}\} \) to the DNA model leads to one-digit error detection
  - the parallel RRNS-based DNA arithmetic improves the reliability of DNA computing while at the same time simplifies the DNA encoding scheme
1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
Quantum Computing

• A quantum bit (*qubit*), a microscopy unit, such as an atom or a nuclear spin, is a superposition of orthogonal basis states, |0> and |1>

\[ |x\rangle = \alpha |0\rangle + \beta |1\rangle ; \quad |\alpha|^2 + |\beta|^2 = 1 \]

• Generalizing, the state of an *n-qubit* system

\[ \Upsilon = \sum_{b=0,1^n} c_b |b\rangle ; \quad \sum_{b} |c_b|^2 = 1 \]
Quantum Computing

- Single *qubit* gates and respective unitary matrices

(a) Hadamard gate
\[
\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}
\]

(b) Phase gate
\[
\begin{bmatrix} 1 & 0 \\ 0 & e^{j\pi/4} \end{bmatrix}
\]

(c) \(\pi/8\) gate
\[
\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
\]

(d) Pauli-X gate
\[
\begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix}
\]

(e) Pauli-Y gate
\[
\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}
\]

(f) Pauli-Z gate
Quantum Computing

- Quantum algorithms

```
Quantum Search

Quantum counting

Statistics
mean, median, min

Speed up for some
NP problems

Search for
crypto keys

Fourier transform

Hidden subgroup
problem

Discrete log

Order finding

Factoring

Break cryptosystems
(RSA)
```
1. Logarithmic Number Systems (LNS)
2. Residue Number Systems (RNS)
3. Stochastic Computing (SC)
4. Hyper-Dimensional Computing (HDC)
5. DNA Computing
6. Quantum Computing
7. Applications
   A. Lattice-based Post-Quantum Cryptography
   B. Machine Learning
8. Conclusions
Lattice-based Cryptography

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

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

<table>
<thead>
<tr>
<th>change of basis</th>
<th>rounding components</th>
<th>back to canonical basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c \times R^{-1}$</td>
<td>$\lfloor c \times R^{-1} \rfloor$</td>
<td>$\lfloor c \times R^{-1} \rfloor \times R$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{R} \times c \times R^{-1} & \quad \text{change of basis} \\
\lfloor c \times R^{-1} \rfloor \quad \text{rounding components} \\
\lfloor c \times R^{-1} \rfloor \times R & \quad \text{back to canonical basis}
\end{align*}
\]
**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

\[ p = c - \left\lfloor cR^{-1} \right\rfloor R \mod c m_\sigma \text{ for } m_\sigma \geq 2\sigma + 1 \]
Lattice-based Cryptography

• Babai’s algorithm rewritten with integer arithmetic:

\[ u = \left\lfloor cR^{-1} \right\rfloor R = \left\lfloor cR^{-1} + \frac{1}{2} \right\rfloor R = \left\lfloor \frac{dcR^{-1}}{d} + \frac{1}{2} \right\rfloor R = \frac{2cdR^{-1} + d}{2d} - \frac{(2cdR^{-1} + d \mod (2d))}{2d} \]

where \( d = \det(R) \)

Use RNS Montgomery's reduction
RNS based LBC decryption

Results for LBC decryption in CPUs/GPUs

<table>
<thead>
<tr>
<th>Method</th>
<th>$n = 400$</th>
<th>$n = 600$</th>
<th>$n = 800$</th>
<th>$n = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential (i7 4770K)</td>
<td>97.51</td>
<td>283.8</td>
<td>619.4</td>
<td>1222</td>
</tr>
<tr>
<td><strong>RNS-GPU</strong> (K40c)</td>
<td>22.97 (4.2)</td>
<td>283.8 (3.6)</td>
<td>248.9 (2.5)</td>
<td>512.4 (2.4)</td>
</tr>
<tr>
<td><strong>RNS-GPU</strong> (GTX 780 Ti)</td>
<td>16.55 (5.9)</td>
<td>59.73 (4.8)</td>
<td>148.2 (4.2)</td>
<td>349.6 (3.5)</td>
</tr>
<tr>
<td>4-core RNS-CPU (i7 4770K)</td>
<td>21.05 (4.6)</td>
<td>75.48 (3.8)</td>
<td>189.9 (3.3)</td>
<td>369.7 (3.3)</td>
</tr>
<tr>
<td>4-core RNS-CPU (with AVX2) (i7 4770K)</td>
<td>8.668 (11.2)</td>
<td>29.05 (9.8)</td>
<td>74.79 (8.3)</td>
<td>148.5 (8.2)</td>
</tr>
</tbody>
</table>
ML: CNNs

YOLOv2
(You Only Look Once version 2)

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

ML: CNNs

2D Convolutional Operation

- Computational intensive part of the YOLOv2

\[
\begin{align*}
X_{0,0} \times W_{0,0} \\
X_{0,1} \times W_{0,1} \\
X_{0,2} \times W_{0,2} \\
X_{0,3} \times W_{0,3} \\
X_{0,4} \times W_{0,4} \\
X_{1,0} \times W_{1,0} \\
X_{1,1} \times W_{1,1} \\
X_{1,2} \times W_{1,2} \\
X_{1,3} \times W_{1,3} \\
X_{1,4} \times W_{1,4} \\
X_{2,0} \times W_{2,0} \\
X_{2,1} \times W_{2,1} \\
X_{2,2} \times W_{2,2} \\
X_{2,3} \times W_{2,3} \\
X_{2,4} \times W_{2,4} \\
X_{3,0} \times W_{3,0} \\
X_{3,1} \times W_{3,1} \\
X_{3,2} \times W_{3,2} \\
X_{3,3} \times W_{3,3} \\
X_{3,4} \times W_{3,4} \\
X_{4,0} \times W_{4,0} \\
X_{4,1} \times W_{4,1} \\
X_{4,2} \times W_{4,2} \\
X_{4,3} \times W_{4,3} \\
X_{4,4} \times W_{4,4}
\end{align*}
\]

\[
y = X_{2,2} \times W_{2,2}
\]
ML: CNNs

Realization of 2D Convolutional Layer

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

Fully parallelization
with RNS
ML: 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: \(\langle 7, 11, 13 \rangle \times \langle 7, 11, 13 \rangle\)

1. Reuse the same moduli set
2. Decompose a large modulo into smaller ones
Example of Nested RNS

- $19 \times 22 (=418)$ on $<7, <5,6,7>_{11}, <5,6,7>_{13}>$

19 × 22

$= <5,8,6> \times <1,0,9>$

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

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

$= <5,0,2>$

$= 418$
ML: Nested RNS

Realization of Nested RNS

Binary 2NRNS

Realized by BRAMs

LUTs

BRAMs and DSP blocks

6-input LUT

<5,6,7> 2Bin

<7,11,13> 2Bin
ML: NRNS based YOLOv2

NRNS based YOLOv2

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

<table>
<thead>
<tr>
<th>Layer</th>
<th># In. Fmaps</th>
<th># Out. F Size</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(Feature Extraction)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv1</td>
<td>3</td>
<td>$128 \times 128$</td>
</tr>
<tr>
<td>Conv2</td>
<td>128</td>
<td>$128 \times 128$</td>
</tr>
<tr>
<td>Max Pool</td>
<td>128</td>
<td>$64 \times 64$</td>
</tr>
<tr>
<td>Conv3</td>
<td>128</td>
<td>$64 \times 64$</td>
</tr>
<tr>
<td>Conv4</td>
<td>128</td>
<td>$64 \times 64$</td>
</tr>
<tr>
<td>Conv5</td>
<td>128</td>
<td>$64 \times 64$</td>
</tr>
<tr>
<td>Max Pool</td>
<td>128</td>
<td>$32 \times 32$</td>
</tr>
<tr>
<td>Conv6</td>
<td>128</td>
<td>$32 \times 32$</td>
</tr>
<tr>
<td>Conv7</td>
<td>128</td>
<td>$32 \times 32$</td>
</tr>
<tr>
<td>Conv8</td>
<td>128</td>
<td>$32 \times 32$</td>
</tr>
<tr>
<td>Max Pool</td>
<td>128</td>
<td>$16 \times 16$</td>
</tr>
<tr>
<td><strong>(Localization+Classification)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv9</td>
<td>128</td>
<td>$16 \times 16$</td>
</tr>
<tr>
<td>Conv10</td>
<td>128</td>
<td>$16 \times 16$</td>
</tr>
<tr>
<td>Conv11</td>
<td>128</td>
<td>$5^2 \times 3 + (5 \times 5)$</td>
</tr>
<tr>
<td>Accuracy (mAP)</td>
<td></td>
<td>69.1</td>
</tr>
</tbody>
</table>
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
### ML: Evaluation

#### Comparison

<table>
<thead>
<tr>
<th></th>
<th>NVidia Pascal GTX1080Ti</th>
<th>NetFPGA-SUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed [FPS]</td>
<td>20.64</td>
<td>3.84</td>
</tr>
<tr>
<td>Power [W]</td>
<td>60.0</td>
<td>3.5</td>
</tr>
<tr>
<td>Efficiency [FPS/W]</td>
<td>0.344</td>
<td>1.097</td>
</tr>
</tbody>
</table>
Conclusions

• Unconventional data representation and arithmetic fundamental for computing on emerging technologies, such as
  - **RNS**: DNA computing; **SC**: quantum devices (AQFP); **HDC**: CNFET, RRAM

• New applications using unconventional arithmetic, namely
  - **LNS**: ML/CNN; **RNS**: Post-Quantum cryptography; **SC**: homomorphic encryption

• For the investigation on non-conventional arithmetic all dimensions of the systems should be considered
  - including not only computer arithmetic theory, but also advances in technology and the demands of emergent applications.
technology
from seed

Thank You for your attention!