A Power Efficient Network Coding Accelerator

Mattis Hasler – TU Dresden
Problem

• Very cool thing called „Random Linear Network Coding (RLNC)“
• Very high computation complexity
• Need to be done in every router
• Too much power consumption on general purpose hardware
• Custom hardware?
Solution

• Tomahawk MPSoC platform
• Ultra low power
• High power efficiency
• <500 MHz → high parallelity
• RDMA 128bit in and out
• SRAM 2 cycle access
• Tensilica LX 5 at the core
  ▪ Two 128bit data memory interfaces
What is Network Coding

**Traditional Approach**
- Data broken into pieces
- k-piece data set → k pieces
- All pieces needed
- Only these pieces will do

**RLNC**
- Mixtures created from pieces
- Any node can create mixtures
- Many mixtures possible
- Any k mixtures will do
Random Linear Network Coding

\[
\begin{pmatrix}
C_1 \\
C_2 \\
C_3 \\
C_4 \\
C_5 \\
C_6
\end{pmatrix}
= 
\begin{pmatrix}
\alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} \\
\alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} \\
\alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} \\
\alpha_{4,1} & \alpha_{4,2} & \alpha_{4,3} & \alpha_{4,4} & \alpha_{4,5} & \alpha_{4,6} \\
\alpha_{5,1} & \alpha_{5,2} & \alpha_{5,3} & \alpha_{5,4} & \alpha_{5,5} & \alpha_{5,6} \\
\alpha_{6,1} & \alpha_{6,2} & \alpha_{6,3} & \alpha_{6,4} & \alpha_{6,5} & \alpha_{6,6}
\end{pmatrix}
\begin{pmatrix}
P_1 \\
P_2 \\
P_3 \\
P_4 \\
P_5 \\
P_6
\end{pmatrix}
\]
Math basics

• Parameters
  ▪ Generation size
  ▪ Field size – Finite Field with \( q = 2^x \)

• Basic operations
  ▪ Encoding: \( X = CM \)
  ▪ Decoding: \( M = C^{-1}X \)

• Finite field arithmetic
  ▪ Multiplication
  ▪ Addition
  ▪ Inversion
Math basics for Finite Fields

• Finite Fields from prime number $p$ and exponent $n$
  ▪ $GF(p^n)$

• Symbol in finite fields as polynomials
  ▪ $A(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$

• Irreducible Polynomial
  ▪ Polynom that is not the product of two polynomials of positive degree

• Example:
  ▪ $GF(2^8), a = x^2 + 1, b = x^3 + x + 1, p = x^8 + x^4 + x^3 + x + 1$
Math basics for Finite Fields

• Addition
  ▪ Elementwise in the basic Field GF(2)
  ▪ → XOR
  ▪ \(a + b = a - b\)

• Multiplication
  ▪ Normal multiplication, reduced with irreducible polynom

• Inversion
  ▪ Finding number that holds: \(a^{-1}a = 1\)
  ▪ One idea: \(a^{q-1} = 1 \rightarrow a^{-1} = a^{q-2}\) (252 multiplication \(\otimes\))
  ▪ Other idea: translating \(GF(2^8)\) to \(GF((2^4)^2)\) and back
Polynomial = <primitive polynom>
def multiply(a, b):
    p = 0
    while a && b:
        if b[0]:    # lowest bit
            p = p ^ a
            b = b >> 1    # divide by 2
        c = a[-1]   # highest bit
        a = a << 1    # multiply by 2
        if c:
            a = a ^ POLYNOM
    return p

b   a
0b100101  0b110

37   6   6*2**0   0b110
18   12
9    24   6*2**2   0b11000
4    48
2    96
1    192   6*2**5   0b11000000

222  0b11011110

Russian Peasant Multiplication

<table>
<thead>
<tr>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b100101</td>
<td>0b110</td>
</tr>
<tr>
<td>37</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>24</td>
</tr>
<tr>
<td>4</td>
<td>48</td>
</tr>
<tr>
<td>2</td>
<td>96</td>
</tr>
<tr>
<td>1</td>
<td>192</td>
</tr>
<tr>
<td>222</td>
<td>0b11011110</td>
</tr>
</tbody>
</table>
TIE design – Multiplication

\[ p = a \times b \]

Critical path:
8\((a_i \rightarrow a_{i+1}) + (a_7 \rightarrow p_7)\)
9\((\text{XOR} + \text{MUX})\)
Extention to SIMD

- Multiplication of a lot of symbols with the same coefficient
- With 128bit memory interface → 16x SIMD
- ...or 8x SIMD on 16bit calculation
Finite Fields Inversion

- Transformation from $GF((2^n)^2)$ to $GF(2^n)$
  - Homomorph for addition and multiplication
  - Using a precalculated transformation matrix based on $P(x) = x^2 + x + p_0$
  - $A(x) = a_0 + a_1x$ from $GF((2^n)^2)$
    - $a_0$ from $GF(2^n)$
    - $a_1$ from $GF(2^n)$
- Inversion: $A \cdot B = 1$
  - $b_0 = (a_0 + a_1)\Delta^{-1}$
  - $b_1 = a_1\Delta^{-1}$
  - $\Delta = a_0^2 + a_0a_1 + p_0a_1^2$
TIE design – Inversion

- Transform with matrix $T$
- Combine 2 $GF(2^4)$ values
  - Inversion with lookup table
- Transform with matrix $T^{-1}$
TIE size

• Original
  ▪ unmodified LX5

• multiplier
  ▪ Hardware multiplier

• SIMD
  ▪ 16x parallel multiplier

• FLIX
  ▪ Flix option
Speedup

• Inversion
  ▪ Compared against $a^{-1} = a^{253}$ in plain-C implementation
Energy efficiency

- ODROID UX3
- Tensilica LX5
Unrolled Multiplication

Transformed Invert

Network Coding

Any k mixtures will do

Summary

| 37 | 12 |
| 18 | 24 |
| 9  | 48 |
| 4  | 96 |
| 2  | 192|
| 1  | 222|