Accelerating Deterministic Bit-Stream Computing with Resolution Splitting

M. Hassan Najafi, S. Rasoul Faraji, Bingzhe Li, David Lilja, and Kia Bazargan

International Symposium on Quality Electronic Design (ISQED)
Santa Clara, CA
March 6, 2019
Overview

- Introduction to Stochastic Computing (SC)
- Stochastic Operations
- Deterministic Approaches to SC
- Proposed Design for Multiplication
- Proposed Design for Scaled Addition
- Experimental Results
- Summary
Stochastic Computing (SC)

- A re-emerging computing paradigm, first introduced in 1960s
- A collection of techniques that represent and process numbers with streams of random-bits:
  
  e.g., 1011011100, 11100 -> 0.6
- All digits have the same weight, numbers limited to the [0, 1]

- Advantages:
  - Noise tolerance
    - Stochastic: 0010000011000000 3/16 -> 4/16=0.25
    - Binary: 0.0011=0.1875 -> 0.1011=0.68
  - Low hardware cost
    - Multiplication: AND

- Important weakness for years:
  - Approximate computation
    - Due to random fluctuation and correlation issues

Fig 1. Stochastic multiplication using AND gate
Stochastic Operations

- **AND: Multiplication**  \( Y = X1 \times X2 \)

  ![AND Operation Diagram]

- **MUX: Scaled addition**  \( Y = \frac{X1 + X2}{2} \)

  ![MUX Operation Diagram]
Stochastic Bit-Stream Generation

- Converting from **binary** to **stochastic representation**
  - Set the **Constant Number** register to your **target value**
  - Use a source of generating numbers as the second input of comparator
  - Output a 1 if “**Number Source**” $\leq$ **Constant Number””
Deterministic Approaches to SC

• Recent progress in SC has revolutionized the paradigm  
  [Najafi et al. TVLSI’17] [Najafi and Lilja ICCD’17] [Jenson and Riedel ICCAD’16]

• If properly structured, random fluctuation can be removed  
  – Producing deterministic and completely accurate results

• Independence between bit-streams is provided by:  
  1) Relatively prime stream length  [Najafi et al. TVLSI’17]  
  2) Clock Division  [Jenson and Riedel ICCAD’16] [Najafi and Lilja ICCD’17]  
  3) Rotation  [Jenson and Riedel ICCAD’16] [Najafi and Lilja ICCD’17]  
  4) Direct Low-Discrepancy (LD)  [Najafi et al. ICCAD’18]  
  5) Integrated LD  [Najafi et al. ICCAD’18]
Deterministic Approaches to SC

• Example. Rel. prime length method

\[
\begin{align*}
1/3 &= 100100100100 \\
3/4 &= 111011101110 \\
3/12 &= 100000100100
\end{align*}
\]

\[
\begin{align*}
1/4 &= 1000 1000 1000 1000 \\
3/4 &= 1111 1111 1111 0000 \\
3/16 &= 1000 1000 1000 0000
\end{align*}
\]

• Example. Clock division method

\[
\begin{align*}
1/4 &= 1000 1000 1000 1000 \\
3/4 &= 1110 0111 1011 1101 \\
3/16 &= 1000 0000 1000 1000
\end{align*}
\]

• Example. Rotation method
## Deterministic Approaches to SC

### Example. Direct low-discrepancy method

<table>
<thead>
<tr>
<th>Sobol Seq 1</th>
<th>0</th>
<th>1/2</th>
<th>1/4</th>
<th>3/4</th>
<th>1/8</th>
<th>5/8</th>
<th>3/8</th>
<th>7/8</th>
<th>1/16</th>
<th>9/16</th>
<th>5/16</th>
<th>13/16</th>
<th>3/16</th>
<th>11/16</th>
<th>7/16</th>
<th>15/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobol Seq 2</td>
<td>0</td>
<td>1/2</td>
<td>3/4</td>
<td>1/4</td>
<td>5/8</td>
<td>1/8</td>
<td>3/8</td>
<td>7/8</td>
<td>15/16</td>
<td>7/16</td>
<td>3/16</td>
<td>11/16</td>
<td>5/16</td>
<td>13/16</td>
<td>9/16</td>
<td>1/16</td>
</tr>
</tbody>
</table>

\[
\frac{1}{4} = 1000 1000 1000 1000 \\
\frac{3}{4} = 1101 1110 0111 1011 \\
\frac{3}{16} = 1000 1000 0000 1000
\]

### Example. Integrated low-discrepancy method

<table>
<thead>
<tr>
<th>Sobol Seq 1</th>
<th>0</th>
<th>1/2</th>
<th>1/4</th>
<th>3/4</th>
<th>1/8</th>
<th>5/8</th>
<th>3/8</th>
<th>7/8</th>
<th>1/16</th>
<th>9/16</th>
<th>5/16</th>
<th>13/16</th>
<th>3/16</th>
<th>11/16</th>
<th>7/16</th>
<th>15/16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobol Seq 2</td>
<td>0</td>
<td>1/2</td>
<td>3/4</td>
<td>1/4</td>
<td>5/8</td>
<td>1/8</td>
<td>3/8</td>
<td>7/8</td>
<td>15/16</td>
<td>7/16</td>
<td>3/16</td>
<td>11/16</td>
<td>5/16</td>
<td>13/16</td>
<td>9/16</td>
<td>1/16</td>
</tr>
</tbody>
</table>

0, 1/2, 1/4, 3/4, 0, 1/2, 1/4, 3/4, 0, 1/2, 1/4, 3/4, 0, 1/2, 1/4, 3/4

0, 1/2, 3/4, 1/4, 1/4, 0, 1/2, 3/4, 1/4, 0, 1/2, 3/4, 1/4, 0

\[
\frac{2}{4} = 1010 1010 1010 1010 \\
\frac{3}{4} = 1101 1110 0111 1011 \\
\frac{6}{16} = 1000 1010 0010 1010
\]
Deterministic Approaches to SC

• Challenge
  – Producing **completely accurate** results with any deterministic bit-stream-based method requires
    • $2^i N$ cycles where
      – $i$ is the number of inputs and $N$ is the precision of input data
    • This **long processing time** makes the deterministic methods **energy inefficient** for many applications

• Proposed idea of this work
  – Resolution splitting of input data
    • Exponential reduction in the processing time
    • Significant reduction in the energy consumption
• For multiplication operation we exploit the idea of performing a full precision multiplication by first producing partial products
• Partial products have a lower resolution, so they are represented by exponentially shorter bit-streams

$$Z = X \cdot Y$$

Conventional circuit for bit-stream-based multiplication (K=1)
• **N-bit** binary data is split into **K sub-values** of $M = N/K$-bit resolution, where $K$ is a power of 2

• For example for resolution splitting of two input values $X$ and $Y$ with $K = 2$:

  $X = X_0 \cdot 2^M + X_1$ \quad $Y = Y_0 \cdot 2^M + Y_1$

  $Z = (X_0 \cdot Y_0) \cdot 2^{2M} + (X_0 \cdot Y_1 + X_1 \cdot Y_0) \cdot 2^M + X_1 \cdot Y_1$

• E.g.,

  $X = 10110010$ \quad $X_0 = 1011$ \quad $X_1 = 0010$
Multiplication

• Proposed circuit for bit-stream-based multiplication with resolution splitting of $K = 2$
Multiplication

- **General** proposed circuit for bit-stream based multiplication with resolution splitting of $K$
Multiplication

- Synthesis results of different design approaches for multiplication of 8-bit precision input data

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Design Approach</th>
<th>Area ($\mu m^2$)</th>
<th>Delay (ns)</th>
<th>Power (@MaxF) mW</th>
<th>Energy/cycle (@MaxF)</th>
<th>Operation Cycles</th>
<th>Total Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Conv. Binary</td>
<td>1021</td>
<td>0.80</td>
<td>2.040</td>
<td>1.632</td>
<td>1</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>359</td>
<td>0.38</td>
<td>1.055</td>
<td>0.400</td>
<td>$2^{16}$</td>
<td>26,273.3</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>367</td>
<td>0.48</td>
<td>1.012</td>
<td>0.485</td>
<td>$2^8$</td>
<td>124.3</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>483</td>
<td>0.57</td>
<td>1.315</td>
<td>0.749</td>
<td>$2^4$</td>
<td>11.9</td>
</tr>
<tr>
<td>3</td>
<td>Conv. Binary</td>
<td>2,909</td>
<td>1.30</td>
<td>3.155</td>
<td>4.101</td>
<td>1</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>541</td>
<td>0.42</td>
<td>1.386</td>
<td>0.582</td>
<td>$2^{24}$</td>
<td>9,769,372.8</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>569</td>
<td>0.58</td>
<td>1.169</td>
<td>0.678</td>
<td>$2^{12}$</td>
<td>2,777.0</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>1,351</td>
<td>0.88</td>
<td>1.561</td>
<td>1.373</td>
<td>$2^6$</td>
<td>87.9</td>
</tr>
<tr>
<td>4</td>
<td>Conv. Binary</td>
<td>5,603</td>
<td>1.55</td>
<td>6.002</td>
<td>9.303</td>
<td>1</td>
<td>9.3</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>727</td>
<td>0.43</td>
<td>1.786</td>
<td>0.768</td>
<td>$2^{32}$</td>
<td>3,299,823,374.0</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>872</td>
<td>0.66</td>
<td>1.364</td>
<td>0.900</td>
<td>$2^{16}$</td>
<td>59,028.2</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>3,895</td>
<td>1.22</td>
<td>2.184</td>
<td>2.664</td>
<td>$2^8$</td>
<td>682.0</td>
</tr>
</tbody>
</table>

- Comparing the total energy consumption of the conventional binary and prior deterministic design ($K = 1$) we see more than $16 \times 10^3$, $23 \times 10^5$, and $35 \times 10^7$ times increase when implementing 2-input, 3-input, and 4-input multipliers → So, K=1 is unacceptable for any application!

- Significant reduction with K=2 and K=4
Scaled Addition

• Similarly, for the scaled addition we split the full precision operation to simpler operations with a lower resolution

• Partial additions in a lower resolution are performed on exponentially shorter bit-streams

• The difference with the multiplication operation:
  – The produced output bit-streams are converted to binary radix representation and concatenated to produce the final result

\[ Z = (1 - S) \cdot X + S \cdot Y \]

\[
Z = (1 - S) \cdot (X_0 \cdot 2^M + X_1) + S \cdot (Y_0 \cdot 2^M + Y_1) = \\
((1 - S) \cdot X_0 + S \cdot Y_0) \cdot 2^M + ((1 - S) \cdot X_1 + S \cdot Y_1) \\
= Z_0 \cdot 2^M + Z_1 \]
Scaled Addition

• Conventional circuit for bit-stream-based scaled addition (K=1)
Scaled Addition

- Proposed circuit for bit-stream based scaled addition with resolution splitting of $K = 2$
Scaled Addition

- General proposed circuit for bit-stream based scaled addition with resolution splitting of \( K \)
Scaled Addition

- Synthesis results of different design approaches for **scaled addition** of 8-bit precision input data

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Design Approach</th>
<th>Area ($\mu m^2$)</th>
<th>CP (ns)</th>
<th>Power(@Maxf) mW</th>
<th>Energy/cycle(@Maxf)</th>
<th>Operation Cycles</th>
<th>Total Energy(pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Conv. Binary</td>
<td>159</td>
<td>0.40</td>
<td>0.851</td>
<td>0.341</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>174</td>
<td>0.36</td>
<td>0.652</td>
<td>0.235</td>
<td>$2^9$</td>
<td>120.2</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>191</td>
<td>0.34</td>
<td>0.765</td>
<td>0.260</td>
<td>$2^5$</td>
<td>8.3</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>210</td>
<td>0.34</td>
<td>0.911</td>
<td>0.310</td>
<td>$2^3$</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>Conv. Binary</td>
<td>314</td>
<td>0.60</td>
<td>0.958</td>
<td>0.575</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>187</td>
<td>0.39</td>
<td>0.607</td>
<td>0.237</td>
<td>$2^{10}$</td>
<td>242.4</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>201</td>
<td>0.36</td>
<td>0.776</td>
<td>0.279</td>
<td>$2^6$</td>
<td>17.9</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>230</td>
<td>0.38</td>
<td>0.829</td>
<td>0.315</td>
<td>$2^4$</td>
<td>5.0</td>
</tr>
<tr>
<td>8</td>
<td>Conv. Binary</td>
<td>611</td>
<td>0.74</td>
<td>1.616</td>
<td>1.195</td>
<td>1</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=1</td>
<td>210</td>
<td>0.43</td>
<td>0.563</td>
<td>0.242</td>
<td>$2^{11}$</td>
<td>496.1</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=2</td>
<td>237</td>
<td>0.43</td>
<td>0.729</td>
<td>0.314</td>
<td>$2^7$</td>
<td>40.1</td>
</tr>
<tr>
<td></td>
<td>Bit-stream K=4</td>
<td>324</td>
<td>0.43</td>
<td>0.779</td>
<td>0.335</td>
<td>$2^6$</td>
<td>10.7</td>
</tr>
</tbody>
</table>

- **Significant reduction with K=2 and K=4**
- E.g., for the 8-input scaled addition, **12x** and **46x** reduction in the total energy consumption, with K = 2 and K = 4, respectively.
Experimental Results

- Energy consumption vs. mean absolute error
  - Multiplication

Energy consumption vs. MAE of the implemented
2-input 8-bit precision multipliers

Energy consumption vs. MAE of the implemented
3-input 8-bit precision multipliers
Experimental Results

- Energy consumption vs. mean absolute error
  - Scaled addition

Energy consumption vs. MAE of the implemented
2-input 8-bit precision scaled adders

Energy consumption vs. MAE of the implemented
4-input 8-bit precision scaled adders
Summary

• Important challenge with the recent deterministic methods of processing bit-streams
  – Long processing time and high energy consumption

• We propose a hybrid bit-stream-binary resolution splitting method
  – An exponential reduction in the processing time
  – Significant improvement in the energy consumption

• Evaluated the performance on the multiplication and scaled addition operations

• Energy-efficient implementation of these operations are most useful for applications that can tolerate slight inaccuracy
  – Neural network and image processing
Questions?

M. Hassan Najafi
najafi@louisiana.edu