Design Space Exploration of the Lightweight Stream Cipher WG-8 for FPGAs and ASICs

Gangqiang Yang, Xinxin Fan, Mark Aagaard and Guang Gong

University of Waterloo

g37yang@uwaterloo.ca

Sept 29, 2013
Smart devices are everywhere.
Smart devices are everywhere.

- Lightweight symmetric ciphers.
  - **Block ciphers:** HIGHT, PRESENT, CLEFIA, LED, PRINCE, SIMON/SPECK.
  - **Stream ciphers:** Grain, Trivium, MICKEY, lightweight instances of WG stream cipher family.
Figure 1: The High-Level Hardware Architecture of the Lightweight Stream Cipher WG-8
Overview

1. Introduction to WG-8
2. Hardware architecture of WG-8
   - Preliminaries
   - Behaviour and hardware architecture of WG-8
3. The WG-8 transformation module – WGT-8 ($x^{19}$)
   - Implementation of WGT-8 module using look-up tables
   - Implementation of WGT-8 module using TF1
   - Implementation of WGT-8 module using TF2
   - Implementation of WGT-8 module using TF3
4. The multiplication by $\omega$ module
5. Results
   - Results for FPGAs
   - Results for ASICs
   - Discussions
6. Conclusions
WG-8

- WG-8 is a recently proposed lightweight instance of the WG stream cipher family.
  - Good **randomness properties**, including period, balance, ideal two-level autocorrelation, ideal tuple distribution, and exact linear complexity.
  - Resist to time/memory/data trade-off attack, differential attack, algebraic attack, correlation attack, cube attack and discrete fourier transform (DFT) attack.

Recent work has demonstrated the advantages of tower field constructions for finite field arithmetic in the AES(S-box) and WG-16 ciphers.

Purpose.

Investigate three different tower field constructions in $\mathbb{F}_{2^8}$ and analyze their effect in hardware architectures for WG-8.

Explore the design space and hardware performance for WG-8 on low-cost FPGAs and CMOS 65nm ASICs in terms of area, speed and power consumption.
WG-8 is a recently proposed lightweight instance of the WG stream cipher family.

- Good *randomness properties*, including period, balance, ideal two-level autocorrelation, ideal tuple distribution, and exact linear complexity.

- Resist to time/memory/data trade-off attack, differential attack, algebraic attack, correlation attack, cube attack and discrete fourier transform (DFT) attack.

- Recent work has demonstrated the advantages of *tower field constructions* for finite field arithmetic in the AES(S-box) and WG-16 ciphers.
WG-8 is a recently proposed lightweight instance of the WG stream cipher family.

- Good randomness properties, including period, balance, ideal two-level autocorrelation, ideal tuple distribution, and exact linear complexity.

- Resist to time/memory/data trade-off attack, differential attack, algebraic attack, correlation attack, cube attack and discrete fourier transform (DFT) attack.

Recent work has demonstrated the advantages of tower field constructions for finite field arithmetic in the AES(S-box) and WG-16 ciphers.

Purpose.

- Investigate three different tower field constructions in $\mathbb{F}_{2^8}$ and analyze their effect in hardware architectures for WG-8.

- Explore the design space and hardware performance for WG-8 on low-cost FPGAs and CMOS 65nm ASICs in terms of area, speed and power consumption.
Main contribution

- We proposed four different hardware architectures for WG-8 transformation module (WGT-8).

1. Directly employs $8 \times 8$ look-up table over $\mathbb{F}_{2^8}$.

2. Based on the tower construction $\mathbb{F}_{2^4}$ together with small $4 \times 4$ look-up tables for arithmetic in $\mathbb{F}_{2^4}$.

3. Based on the tower construction $\mathbb{F}_{2^4}$, but with type-I optimal normal basis (ONB) for efficient computations in $\mathbb{F}_{2^4}$.

4. Benefits from the construction $\mathbb{F}_{((2^2)^2)^2}$ enables the simplification of the trace of product of two finite field elements.
Preliminaries

- \( p(x) = x^8 + x^4 + x^3 + x^2 + 1 \), a primitive polynomial of degree 8 over \( \mathbb{F}_2 \).

- \( \mathbb{F}_{2^8} \), the extension field of \( \mathbb{F}_2 \) defined by the primitive polynomial \( p(x) \) with \( 2^8 \) elements. \( \omega \) is a primitive element of \( \mathbb{F}_{2^8} \) and \( p(\omega) = 0 \).

- \( \text{Tr}(x) = x + x^2 + x^2^2 + x^2^3 + x^2^4 + x^2^5 + x^2^6 + x^2^7 \), the trace function from \( \mathbb{F}_{2^8} \to \mathbb{F}_2 \).

- \( l(x) = x^{20} + x^9 + x^8 + x^7 + x^4 + x^3 + x^2 + x + \omega \), the feedback polynomial of LFSR (which is also a primitive polynomial over \( \mathbb{F}_{2^8} \)).

- \( q(x) = x + x^{2^3+1} + x^{2^6+2^3+1} + x^{2^6-2^3+1} + x^{2^6+2^3-1} \), a permutation polynomial over \( \mathbb{F}_{2^8} \).

- \( \text{WGP-8}(x^d) = q(x^d + 1) + 1 \), the WG-8 permutation with decimation \( d = 19 \) from \( \mathbb{F}_{2^8} \to \mathbb{F}_{2^8} \), where \( d \) is coprime to \( 2^8 - 1 \).

- \( \text{WGT-8}(x^d) = \text{Tr}(\text{WGP-8}(x^d)) = \text{Tr}(q(x^d + 1)) \), the WG-8 transformation with decimation \( d = 19 \) from \( \mathbb{F}_{2^8} \to \mathbb{F}_2 \).
**Behaviour of WG-8**

- **WG-8 consists of a 20-stage LFSR with feedback polynomial $l(x)$ followed by WG-8 transformation module ($WGT-8(x^{19})$).**
  - It contains loading and initialization phase, and running phase.

- **Loading and initialization phase.**
  - The loading phase will take 20 clock cycles for loading the initial state $S_i$.
  - The initialization phase will run for 40 clock cycles after the loading phase based on the recursive relation in the picture.

![Diagram of WG-8](image)

$WGP-8(x^{19}):$ WG-8 Permutation Module
Running phase

- The recursive relation for updating the LFSR in the running phase is different from the loading and initialization phase.

- 1-bit keystream is generated from the trace module after each clock cycle.
Hardware architecture of WG-8

- Four main components.
  - 20-stage LFSR.
  - WGT-8(x^{19}). (most important)
  - FSM.
  - Multiplication by \( \omega \) module.

- WGT-8(x^{19}) was further split into WGP-8(x^{19}) and trace modules.
Implementation of WGT-8 module using LUT and Tower Field arithmetic

Figure 2: Look-up table in $\mathbb{F}_{2^8}$

Figure 3: Tower Field 1 construction for $\mathbb{F}_{2^8}$

Figure 4: Tower Field 2 construction for $\mathbb{F}_{2^8}$

Figure 5: Tower Field 3 construction for $\mathbb{F}_{2^8}$
Implementation of WG-8 transformation module using look-up tables

- Pre-compute WGP-8($x^{19}$) and store the results in a $8 \times 8$ look-up table, ($Table(WGP-8)$).

- Pre-compute WGT-8($x^{19}$) = $Tr(WGP-8(x^{19}))$ and store the results in a $8 \times 1$ look-up table, ($Table(WGT-8)$).
The tower construction is used to calculate the multiplication efficiently in $F_{2^8}$.

$2^t$ is calculated by cyclic shift operation using normal basis (NB).

The conversion matrices between TF and NB representations are needed.

$e_1(x)$ is a primitive polynomial.

The arithmetic in $F_{2^4}$ is conducted with the aid of a $4 \times 4$ exponentiation table $T_{exp}$ and a $4 \times 4$ logarithm table $T_{log}$.

- The table $T_{exp}$ stores exponentiation $\alpha^i$, $i = 0, 1, \ldots, 14$.
- The table $T_{log}$ keeps the exponent $i$ for each $\alpha^i$, $i = 0, 1, \ldots, 14$.

$AB = T_{exp}(T_{log}A + T_{log}B)$. 
Implementation of WGT-8($x^{19}$) module

- **WGP-8 and WGT-8 modules.**
  - $WGP-8(x^{19}) = y + y^{2^3+1} + y^{2^6}(y^{2^3+1} + y^{2^3-1}) + y^{2^3(2^3-1)+1} + 1$, $y = x^{19} + 1$.
  - $WGT-8(x^{19}) = \text{Tr} \left( y + y^{2^3+1} + y^{2^6}(y^{2^3+1} + y^{2^3-1}) + y^{2^3(2^3-1)+1} \right)$.

- A simple $WGT-8(x^{19})$ module described in normal basis (NB).
Implementation of WGT-8 module using TF2 (Tower Field arithmetic)

- $f_2(x)$ is the same as $f_1(x)$.
- $e_2(x)$ is only an irreducible polynomial.
- Type-I optimal normal basis (ONB) exists, and the arithmetic calculation in $F_{2^4}$ is very efficient.

Figure 7: Tower Field 2 construction for $F_{2^8}$
Figure 8: Simple description of WGT-8($x^{19}$) module in normal basis (NB)
The WG-8 transformation module – WGT-8 \((x^{19})\) module for TF1 and TF2.

**Differences between TF1 and TF2.**

- **Multiplication** \(M_8\).
- **Conversion matrices.**
- **Trace function.**
- **Representation of “1” in different tower field.**
Implementation of WGT-8 module using TF3 (Tower Field arithmetic)

The arithmetic operations in $\mathbb{F}_{2^2}$, $\mathbb{F}_{(2^2)^2}$, $\mathbb{F}_{((2^2)^2)^2}$ are all computed using logic directly.

Special trace property can be obtained based on this tower construction.

The trace of multiplication $U$ and $V$ in $\mathbb{F}_{((2^2)^2)^2}$ can be computed as the inner product of $U$ and $V$, i.e,

$$\text{Tr}(UV) = \sum_{i=0}^{7} (u_i \odot_1 v_i)$$

Figure 10: Tower Field 3 construction for $\mathbb{F}_{2^8}$
Hardware architecture of WGT-8($x^{19}$) module for running phase using the trace property

- **WGT-8** ($x^{19}$) can be computed as follows:

  \[
  WGT-8(x^{19}) = \text{Tr}(WGP-8(x^{19})), \quad x \in \mathbb{F}_{2^8}
  \]

  \[
  = \text{Tr} \left( y \oplus_{8} y^{2^3+1} \oplus_{8} y^{2^6} (y^{2^3+1} \oplus_{8} y^{2^3-1}) \oplus_{8} y^{2^3(2^3-1)} y \right)
  \]

  \[
  = \text{Tr}(y \oplus_{8} y^{2^3+1}) \oplus_{1} \text{Tr}(y^{2^6} (y^{2^3+1} \oplus_{8} y^{2^3-1})) \oplus_{1} \text{Tr}(y^{2(2^3-1)} y^{2^6})
  \]

  \[
  = \text{Tr}(y \oplus_{8} y^{2^3+1}) \oplus_{1} \text{Tr}(y^{2^6} \ominus_{8} (y^{2^3+1} \oplus_{8} y^{2^3-1} \oplus_{8} y^{2(2^3-1)}))
  \]

- Two multiplications $y^{2^6} (y^{2^3+1} \oplus_{8} y^{2^3-1})$ and $y^{2^3(2^3-1)} y$ inside the trace function have been replaced by the bitwise AND and XOR operations.

- **Problem:** The **values** of these two multiplications $y^{2^6} (y^{2^3+1} \oplus_{8} y^{2^3-1})$ and $y^{2^3(2^3-1)} y$ are missing.
Hardware architecture of WGP-8\((x^{19})\) module for the initialization phase

- WGP-8 can be computed as follows:

\[
\begin{align*}
\text{WGP-8}(x^{19}) &= q(x^{19} + 1) + 1, \ x \in \mathbb{F}_{2^8} \\
&= (1 \oplus_8 y \oplus_8 y^{2^3+1}) \oplus_8 (y^{2^6} (y^{2^3+1} \oplus_8 y^{2^3-1})) \oplus_8 (y^{2^3(2^3-1)y}).
\end{align*}
\]

- \(y^{2^3+1}\) and \(y^{2^3-1}\) can be obtained from the WGT-8\((x^{19})\) module.

- Question: How can we recover the values of these two multiplications \(y^{2^6} (y^{2^3+1} \oplus_8 y^{2^3-1})\) and \(y^{2^3(2^3-1)y}\)?
Hardware architecture of WGP-8($x^{19}$) module for the initialization phase

- WGP-8 can be computed as follows:

$$WGP-8(x^{19}) = q(x^{19} + 1) + 1, \ x \in \mathbb{F}_{28}$$

$$= (1 \oplus_8 y \oplus_8 y^{2^3+1}) \oplus_8 (y^{2^6} (y^{2^3+1} \oplus_8 y^{2^3-1})) \oplus_8 (y^{2^3(2^3-1)}y).$$

- $y^{2^3+1}$ and $y^{2^3-1}$ can be obtained from the WGT-8($x^{19}$) module.

- Question: How can we recover the values of these two multiplications $y^{2^6} (y^{2^3+1} \oplus_8 y^{2^3-1})$ and $y^{2^3(2^3-1)}y$?

- WGP-8 module is only used in the initialization phase.

- we can keep the throughput of the running phase to be 1 bit/cycle, while decreasing the throughput of the initialization phase (i.e., < 1 bit/cycle).

- It can be done by increasing the length of the initialization phase and reuse the existing multipliers.
Figure 11: Reuse the multiplier in two consecutive clock cycles.
Integrated hardware architecture for computing WGT-8 and WGP-8

Figure 12: The Integrated Hardware Architecture for Computing WGP-8($x^{19}$) and WGT-8($x^{19}$)
The multiplication by $\omega$ module

Using finite field arithmetic:

- $X \cdot \omega = x_7 + x_0 \omega + (x_1 \oplus_1 x_7) \omega^2 + (x_2 \oplus_1 x_7) \omega^3 + (x_3 \oplus_1 x_7) \omega^4 + x_4 \omega^5 + x_5 \omega^6 + x_6 \omega^7$. $X \in \mathbb{F}_{2^8}$ under polynomial basis (PB).

- $X \cdot \omega = M \cdot \begin{pmatrix} x_0' & x_1' & \cdots & x_6' & x_7' \end{pmatrix}^T$. Normal basis (NB).

Using the $8 \times 8$ look-up table.
Results for flip-flops, area, speed, and power consumption analysis of FPGA implementations

<table>
<thead>
<tr>
<th>Implementations</th>
<th>Key Size (bits)</th>
<th>IV Size (bits)</th>
<th>Data Rates (bits/cycle)</th>
<th>#FFs</th>
<th>Area (Slices)</th>
<th>Maximum Frequency (MHz)</th>
<th>Dynamic Power (W)</th>
<th>Maximum Throughput (Mbps)</th>
<th>T/A (Mbps/#Slices)</th>
<th>T/P (Mbps/W)</th>
<th>T/(A<em>P) (Mbps/Slices</em>W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WG-8 (LUT)</td>
<td>80</td>
<td>80</td>
<td>1</td>
<td>85</td>
<td>137</td>
<td>190</td>
<td>0.005</td>
<td>190</td>
<td>1.39</td>
<td>38000</td>
<td>277.4</td>
</tr>
<tr>
<td>WG-8 (LUT)</td>
<td>11</td>
<td>207</td>
<td>11</td>
<td>279</td>
<td>5106</td>
<td>19</td>
<td>4.282</td>
<td>209</td>
<td>0.04</td>
<td>48.8</td>
<td>0.010</td>
</tr>
<tr>
<td>WG-8 (TF 1)</td>
<td>1</td>
<td>83</td>
<td>12</td>
<td>306</td>
<td>2369</td>
<td>42</td>
<td>1.686</td>
<td>462</td>
<td>0.20</td>
<td>274</td>
<td>0.12</td>
</tr>
<tr>
<td>WG-8 (TF 2)</td>
<td>11</td>
<td>306</td>
<td>11</td>
<td>470</td>
<td>2795</td>
<td>44</td>
<td>2.399</td>
<td>484</td>
<td>0.17</td>
<td>201.7</td>
<td>0.07</td>
</tr>
<tr>
<td>Grain</td>
<td>80</td>
<td>64</td>
<td>1</td>
<td>–</td>
<td>44</td>
<td>196</td>
<td>–</td>
<td>196</td>
<td>4.45</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Trivium</td>
<td>80</td>
<td>80</td>
<td>1</td>
<td>–</td>
<td>50</td>
<td>240</td>
<td>–</td>
<td>240</td>
<td>4.80</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

- We use Synopsys Synplify for synthesis, Xilinx ISE for implementation and Modelsim for simulation.
- All results are obtained after post place-and-route phase and the dynamic power consumption are recorded at a frequency of 33.3 MHz except TF1 (16.7 MHz).
- For serial design, we take advantage of SRL16 in Spartan-3 device, which can reduce the flip-flop numbers and area significantly.
- Using parallel LFSR to achieve data rate from one to eleven bits/cycle.
Results
Results for ASICs

Area, speed, and power consumption results for ASIC implementations.

<table>
<thead>
<tr>
<th>Implementations</th>
<th>Key Size (bits)</th>
<th>IV Size (bits)</th>
<th>Data Rates (bits/cycle)</th>
<th>Area (GE)</th>
<th>Optimum Frequency (MHz)</th>
<th>Total Power (mW)</th>
<th>Maximum Throughput (Mbps)</th>
<th>Optimality</th>
<th>Optimal Optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WG-8 (LUT)</td>
<td>80</td>
<td>80</td>
<td>1</td>
<td>1786</td>
<td>500</td>
<td>0.983</td>
<td>500</td>
<td>0.28</td>
<td>508.6</td>
</tr>
<tr>
<td>WG-8 (LUT)</td>
<td>11</td>
<td>3942</td>
<td>11</td>
<td>3942</td>
<td>610</td>
<td>1.344</td>
<td>6710</td>
<td>1.70</td>
<td>4992.6</td>
</tr>
<tr>
<td>WG-8 (TF 1)</td>
<td>1</td>
<td>7523</td>
<td>1</td>
<td>7523</td>
<td>229</td>
<td>35.8</td>
<td>229</td>
<td>0.03</td>
<td>6.40</td>
</tr>
<tr>
<td>WG-8 (TF 1)</td>
<td>11</td>
<td>42762</td>
<td>11</td>
<td>42762</td>
<td>122</td>
<td>100.1</td>
<td>1342</td>
<td>0.03</td>
<td>13.4</td>
</tr>
<tr>
<td>WG-8 (TF 2)</td>
<td>1</td>
<td>3162</td>
<td>1</td>
<td>3162</td>
<td>260</td>
<td>6.97</td>
<td>260</td>
<td>0.08</td>
<td>37.3</td>
</tr>
<tr>
<td>WG-8 (TF 2)</td>
<td>11</td>
<td>22668</td>
<td>11</td>
<td>22668</td>
<td>205</td>
<td>44.6</td>
<td>2255</td>
<td>0.10</td>
<td>50.6</td>
</tr>
<tr>
<td>WG-8 (TF 3)</td>
<td>1</td>
<td>2981</td>
<td>1</td>
<td>2981</td>
<td>254</td>
<td>7.98</td>
<td>254</td>
<td>0.08</td>
<td>31.83</td>
</tr>
<tr>
<td>WG-8 (TF 3)</td>
<td>11</td>
<td>19882</td>
<td>11</td>
<td>19882</td>
<td>205</td>
<td>53.3</td>
<td>2255</td>
<td>0.11</td>
<td>42.3</td>
</tr>
<tr>
<td>Grain</td>
<td>80</td>
<td>64</td>
<td>1</td>
<td>1126</td>
<td>1020</td>
<td>2.04</td>
<td>1020</td>
<td>0.91</td>
<td>500</td>
</tr>
<tr>
<td>Grain</td>
<td>11</td>
<td>1126</td>
<td>11</td>
<td>1126</td>
<td>1098</td>
<td>2.25</td>
<td>1098</td>
<td>10.73</td>
<td>5368</td>
</tr>
<tr>
<td>Trivium</td>
<td>80</td>
<td>80</td>
<td>1</td>
<td>1986</td>
<td>962</td>
<td>3.88</td>
<td>962</td>
<td>0.48</td>
<td>247.9</td>
</tr>
<tr>
<td>Trivium</td>
<td>11</td>
<td>2028</td>
<td>11</td>
<td>2028</td>
<td>990</td>
<td>4</td>
<td>10890</td>
<td>5.37</td>
<td>2722.5</td>
</tr>
</tbody>
</table>

- We use Synopsys Design Compiler for synthesis and Cadence SoC Encounter for place & route phase.
- we use TSMC CMOS 65nm LPLV technology.
- All results are obtained after post place-and-route phase and the dynamic power consumption are recorded at their optimum frequency for all designs.
- The Modelsim simulation is run for 2000 clock cycles, including the initialization and running phases.
The **WG-8 (LUT)** method is the **best** hardware solution for the WG-8 stream cipher.

For the three tower field methods, the best one is dependent on the data rates and different metrics.

**Table 1: The best Tower Field method for different metrics**

<table>
<thead>
<tr>
<th>Data Rates</th>
<th>FPGA</th>
<th>ASIC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T/A</td>
<td>T/P</td>
</tr>
<tr>
<td>1</td>
<td>TF 2</td>
<td>TF 3</td>
</tr>
<tr>
<td>11</td>
<td>TF 2</td>
<td>TF 2</td>
</tr>
</tbody>
</table>

**Table 2: The number of multipliers and multiplexers and the area of them**

<table>
<thead>
<tr>
<th>Implementations</th>
<th>Multipliers (Number)</th>
<th>Multiplier (Single area) (slices)</th>
<th>Multiplier (Total areas) (slices)</th>
<th>Multiplexers (Number)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF 1</td>
<td>7</td>
<td>53</td>
<td>371</td>
<td>0</td>
</tr>
<tr>
<td>TF 2</td>
<td>7</td>
<td>29</td>
<td>203</td>
<td>0</td>
</tr>
<tr>
<td>TF 3</td>
<td>5</td>
<td>26</td>
<td>130</td>
<td>6</td>
</tr>
</tbody>
</table>
Discussions

- Compare results with Grain and Trivium.

  - For the FPGA results, Grain and Trivium are better than WG-8(LUT) method in metric of T/A for both date rates one and eleven.

  - For the ASIC results, WG-8(LUT) is 105% better than Trivium in metric of T/P and 138% better in T/(A*P) for data rates one. For data rates eleven, it is 83% better in T/P.

  - For ASIC results, WG-8(LUT) is 2% better than Grain in metric of T/P for data rates one.

- Guaranteed mathematical properties of WG-8.

  - WG-8 has desired randomness properties like period, balance, ideal two-level autocorrelation, ideal tuple distribution, and exact linear complexity.
Conclusions and future work

Conclusions:
- Four implementation methods for WG-8 transformation module (WGT-8) are proposed.
  - LUT: Look-up tables.
  - TF1: Tower construction $\mathbb{F}_{(2^4)^2}$ with small $4 \times 4$ look-up tables in $\mathbb{F}_{2^4}$.
  - TF2: Tower construction $\mathbb{F}_{(2^4)^2}$ with Type-I optimal normal basis (ONB) in $\mathbb{F}_{2^4}$.
  - TF3: Tower construction $\mathbb{F}_{((2^2)^2)^2}$ with special trace property.
- The tower field constructions affect the hardware architecture of WG-8 and also the area of one multiplier.
- Make trade-offs in terms of area, speed, and power consumption.
- Among the three tower field constructions, TF2 with type-I optimal normal basis is the best choice in most cases for WG-8.

Future work:
- Explore more efficient tower field constructions for large size WG stream ciphers, i.e., WG-10, WG-11, WG-14, and other lightweight stream ciphers.
- More architecture optimizations, such as pipelining and reuse techniques can be performed to improve the speed, area and power consumption.
Q & A?