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.

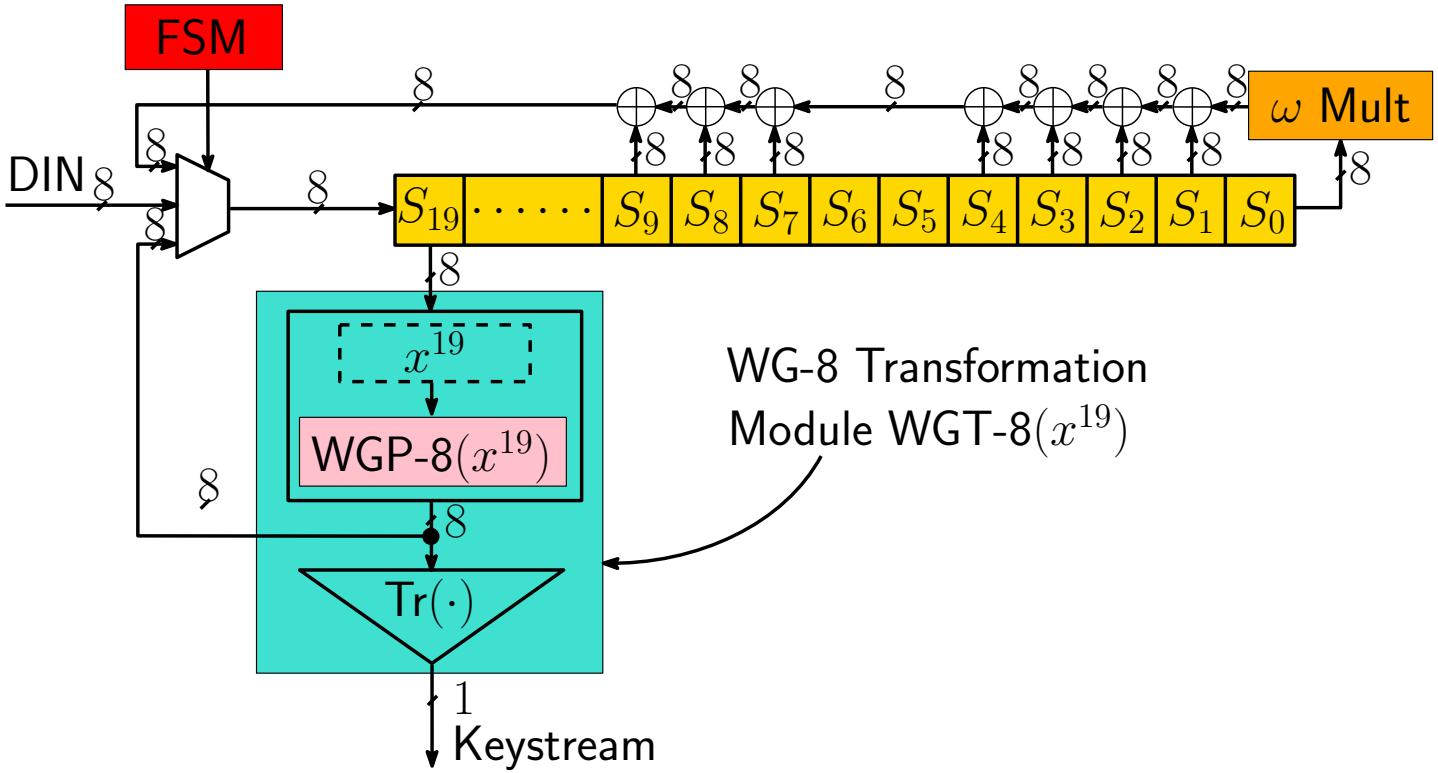


Figure 1: The High-Level Hardware Architecture of the Lightweight Stream Cipher WG-8

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 ω 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.

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.

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.

Main contribution

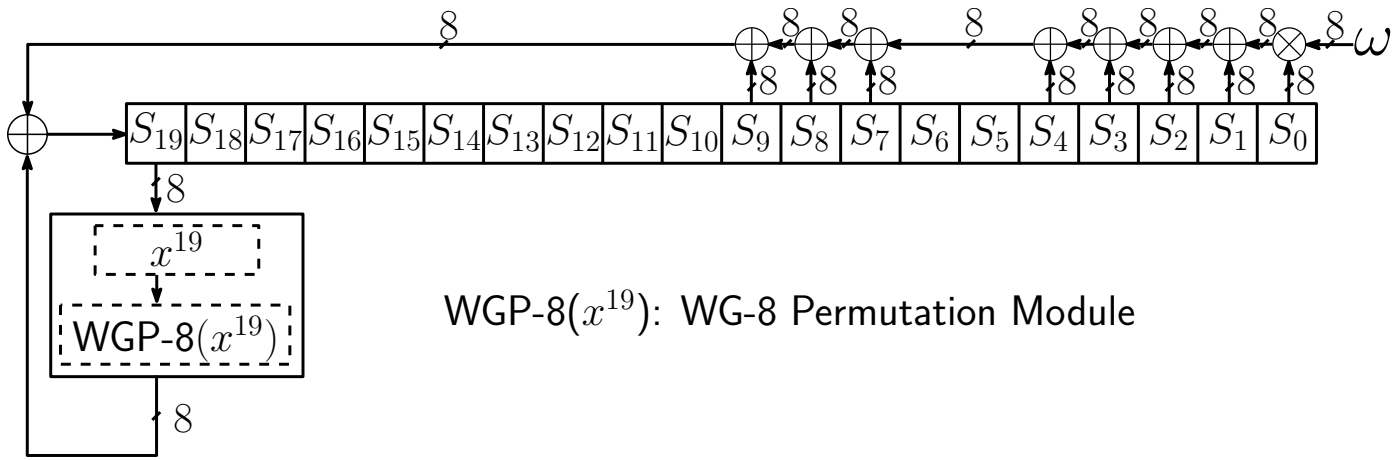
- We proposed four different hardware architectures for WG-8 transformation module (WGT-8).
  1. Directly employs 8×8 look-up table over $\mathbb{F}_{2^8}$.
  2. Based on the tower construction $\mathbb{F}_{(2^4)^2}$ together with small 4×4 look-up tables for arithmetic in $\mathbb{F}_{2^4}$.
  3. Based on the tower construction $\mathbb{F}_{(2^4)^2}$, 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. ω is a primitive element of $\mathbb{F}_{2^8}$ and $p(\omega) = 0$.
- $\mathrm{Tr}(x) = x + x^2 + x^{2^2} + x^{2^3} + x^{2^4} + x^{2^5} + x^{2^6} + x^{2^7}$, 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 $\mathbb{F}_{2^8}$.
- 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 $2^8 - 1$.
- WGT-8$(x^d) = \mathrm{Tr}(\text{WGP-8}(x^d)) = \mathrm{Tr}(q(x^d + 1))$, the WG-8 transformation with decimation $d = 19$ from $\mathbb{F}_{2^8} \to \mathbb{F}_2$.

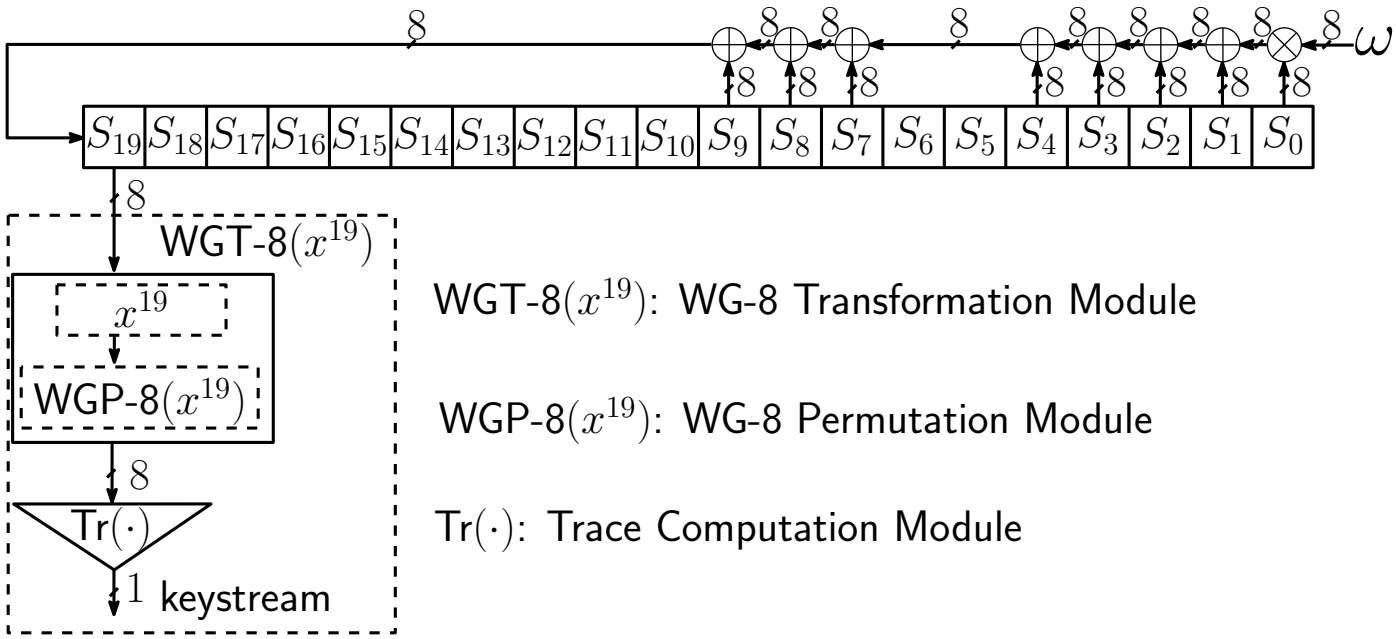
Behaviour of WG-8

- **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.



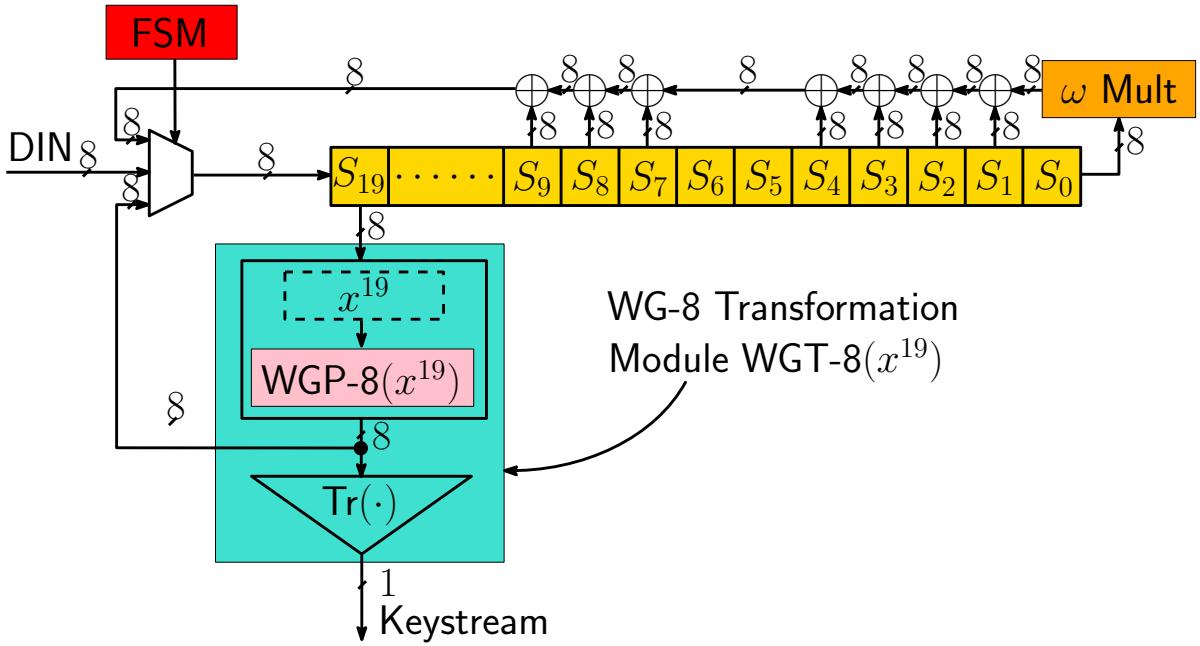
WGP-8(x^{19}): WG-8 Permutation Module

Running phase



- **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 ω 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

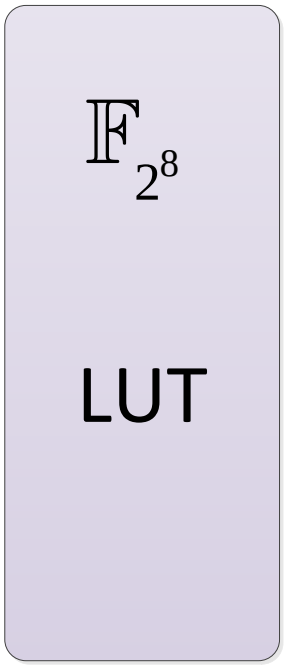


Figure 2: Look-up table in $\mathbb{F}_{2^8}$

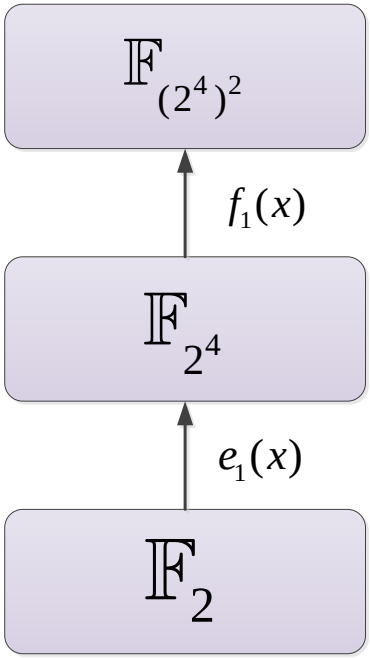


Figure 3: Tower Field 1 construction for $\mathbb{F}_{2^8}$

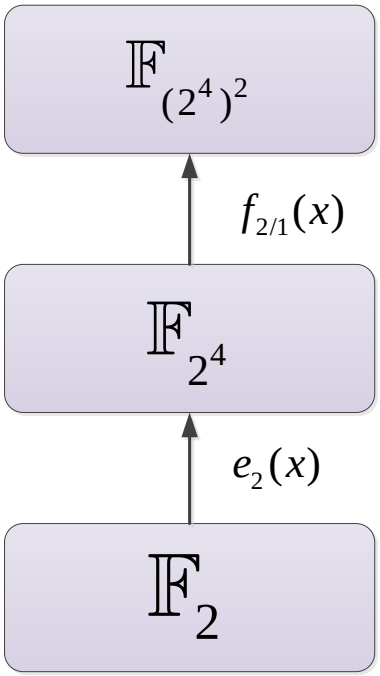


Figure 4: Tower Field 2 construction for $\mathbb{F}_{2^8}$

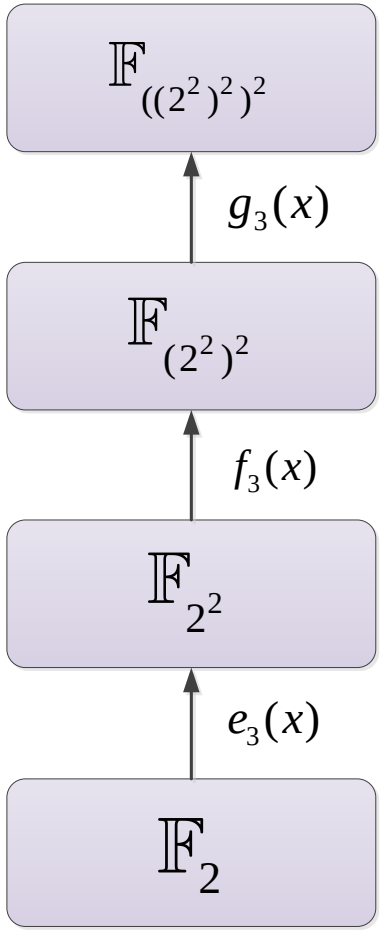
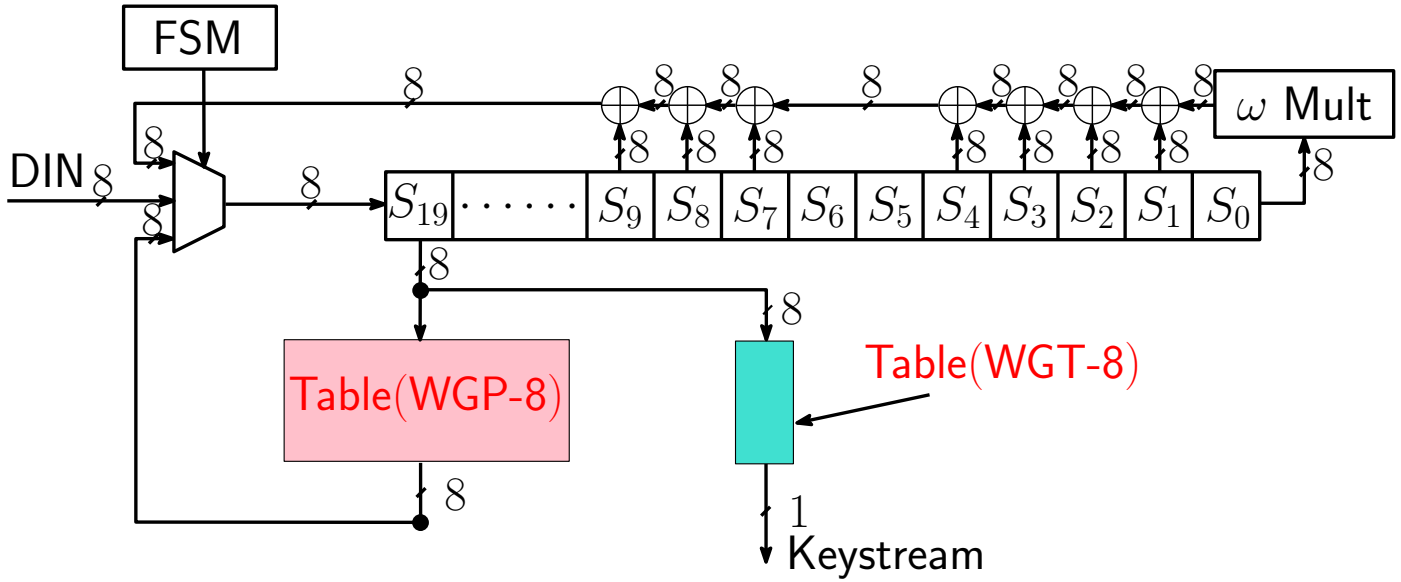


Figure 5: Tower Field 3 construction for $\mathbb{F}_{2^8}$

Implementation of WG-8 transformation module using look-up tables

- Pre-compute WGP-8(x^{19}) and store the results in a 8×8 look-up table, (Table(WGP-8)).
- Pre-compute WGT-8(x^{19}) $=$ Tr(WGP-8(x^{19})) and store the results in a 8×1 look-up table, (Table(WGT-8)).



Implementation of WGT-8 module using TF1 (Tower Field arithmetic)

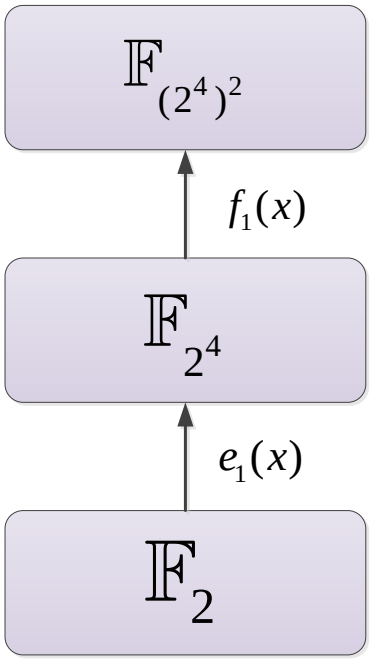


Figure 6: Tower Field 1 construction for $\mathbb{F}_{2^8}$

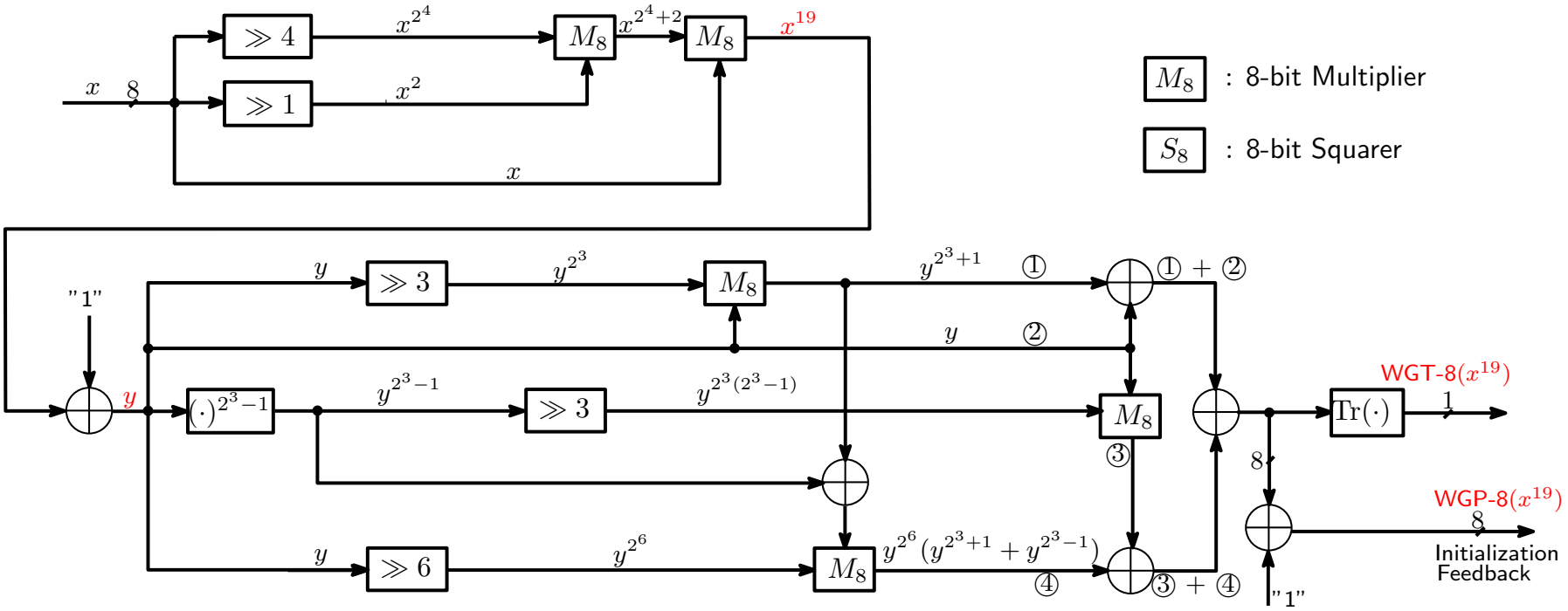
- The tower construction is used to calculate the multiplication efficiently in F_{2^8}.
- 2^t is calculated by cyclic shift operation using normal basis (NB).
- The conversion matrices between TF and NB representations are needed.
- $e_1(x)$ is a primitive polynomial.

- The arithmetic in F_{2^4} is conducted with the aid of a 4×4 exponentiation table T_{exp} and a 4×4 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}) $= y + y^{2^3+1} + y^{2^6}(y^{2^3+1} + y^{2^3-1}) + y^{2^3(2^3-1)+1} + 1, \quad y = x^{19} + 1.$
  - WGT-8(x^{19}) $= \mathrm{Tr}\left(y + y^{2^3+1} + y^{2^6}(y^{2^3+1} + y^{2^3-1}) + y^{2^3(2^3-1)+1}\right).$

- A simple WGT-8(x^{19}) module described in normal basis (NB).



Implementation of WGT-8 module using TF2 (Tower Field arithmetic)

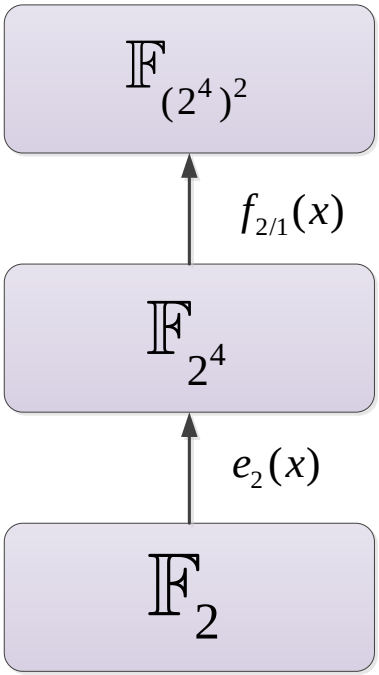


Figure 7: Tower Field 2 construction for $\mathbb{F}_{2^8}$

- $f_2(x)$ 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.

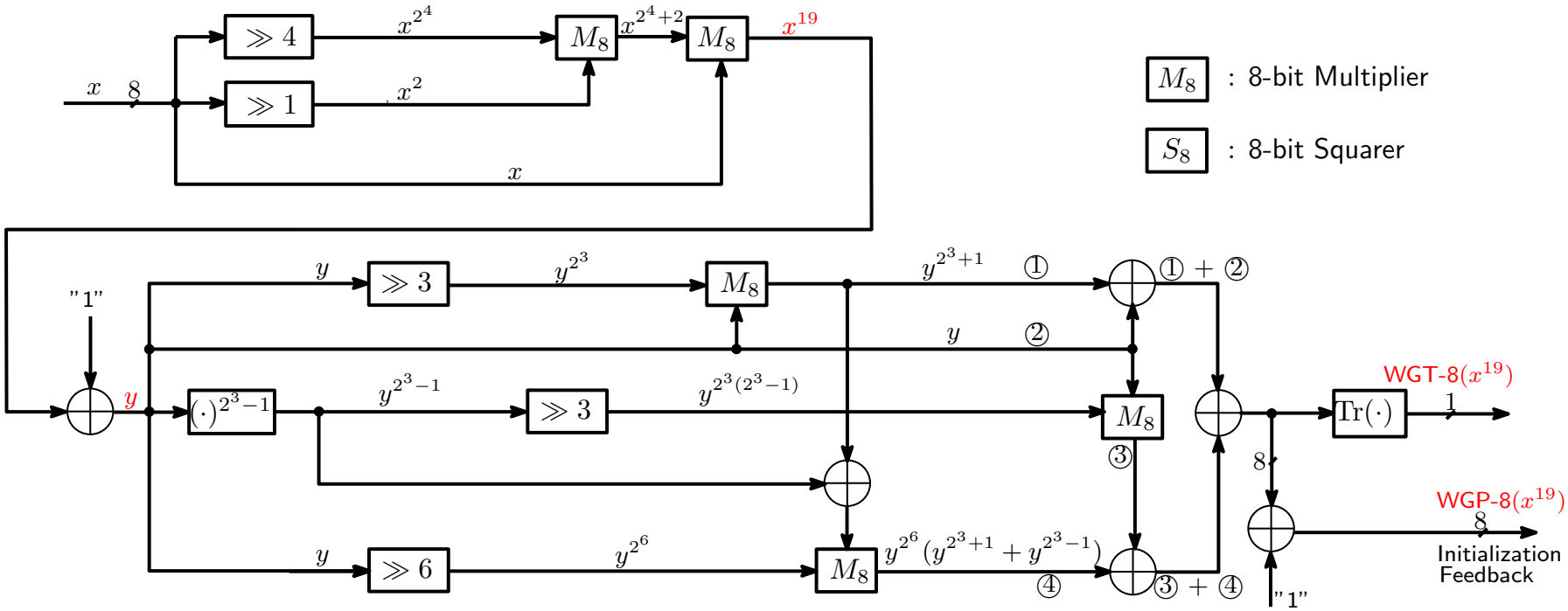


Figure 8: Simple description of WGT-8(x^{19}) module in normal basis (NB)

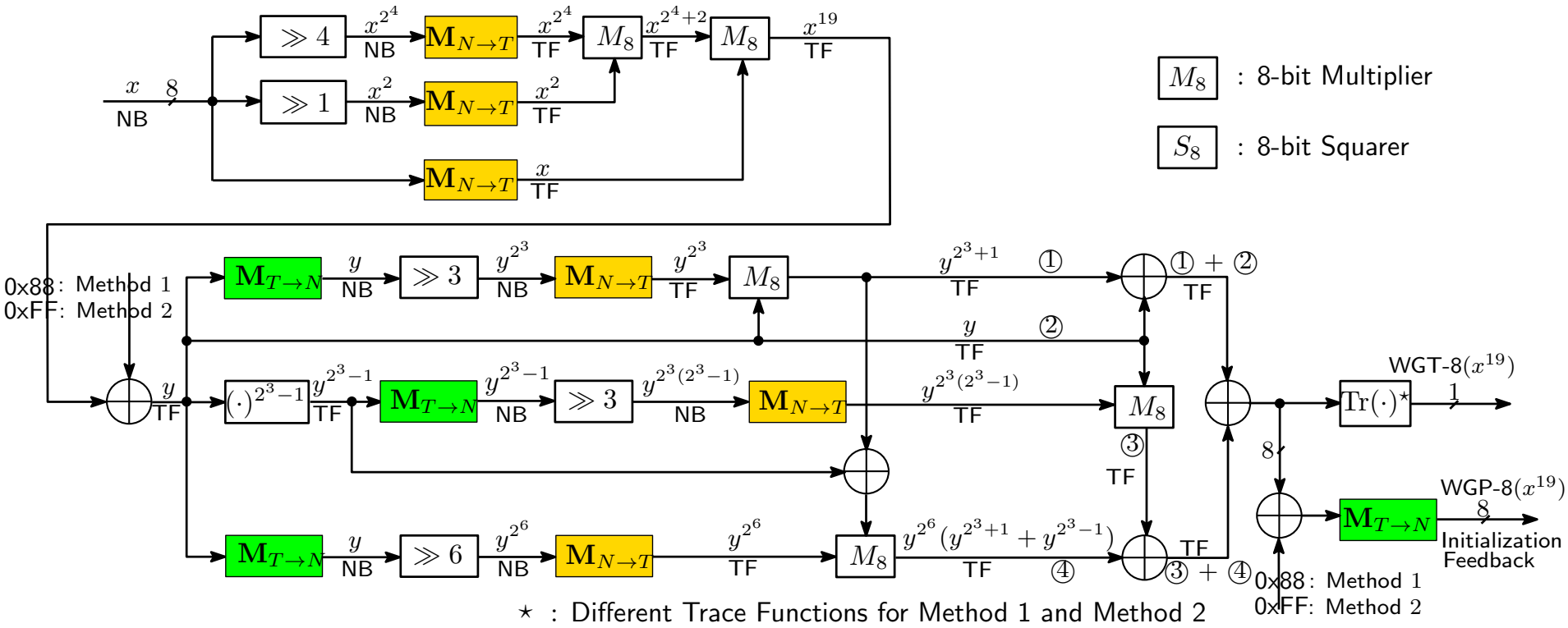


Figure 9: The Hardware Architecture of WGT-8(x^{19}) module for TF1 and TF2.

- 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)

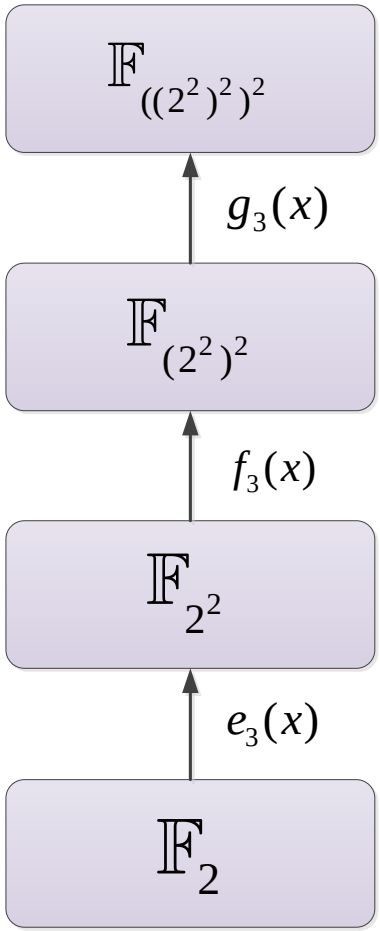
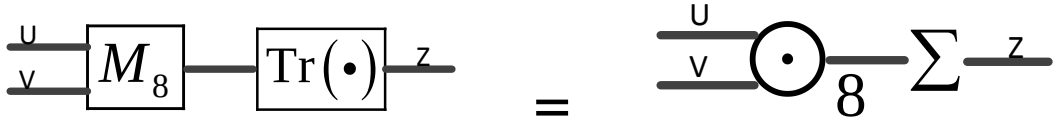


Figure 10: Tower Field 3 construction for $\mathbb{F}_{2^8}$

- 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,

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



Hardware architecture of WGT-8(x^{19}) module for running phase using the trace property

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

$$\begin{aligned}\text{WGT-8}(x^{19}) &= \mathrm{Tr}(\text{WGP-8}(x^{19})),\ x \in \mathbb{F}_{2^8} \\ &= \mathrm{Tr}\left(y \oplus_8 y^{2^3+1} \oplus_8 y^{2^6}(y^{2^3+1} \oplus_8 y^{2^3-1}) \oplus_8 y^{2^3(2^3-1)}y\right) \\ &= \mathrm{Tr}(y \oplus_8 y^{2^3+1}) \oplus_1 \mathrm{Tr}(y^{2^6}(y^{2^3+1} \oplus_8 y^{2^3-1})) \oplus_1 \mathrm{Tr}(y^{2(2^3-1)}y^{2^6}) \\ &= \mathrm{Tr}(y \oplus_8 y^{2^3+1}) \oplus_1 \mathrm{Tr}(y^{2^6} \odot_8 (y^{2^3+1} \oplus_8 y^{2^3-1} \oplus_8 y^{2(2^3-1)})).\end{aligned}$$

- 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:

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

- 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:

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

- 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.

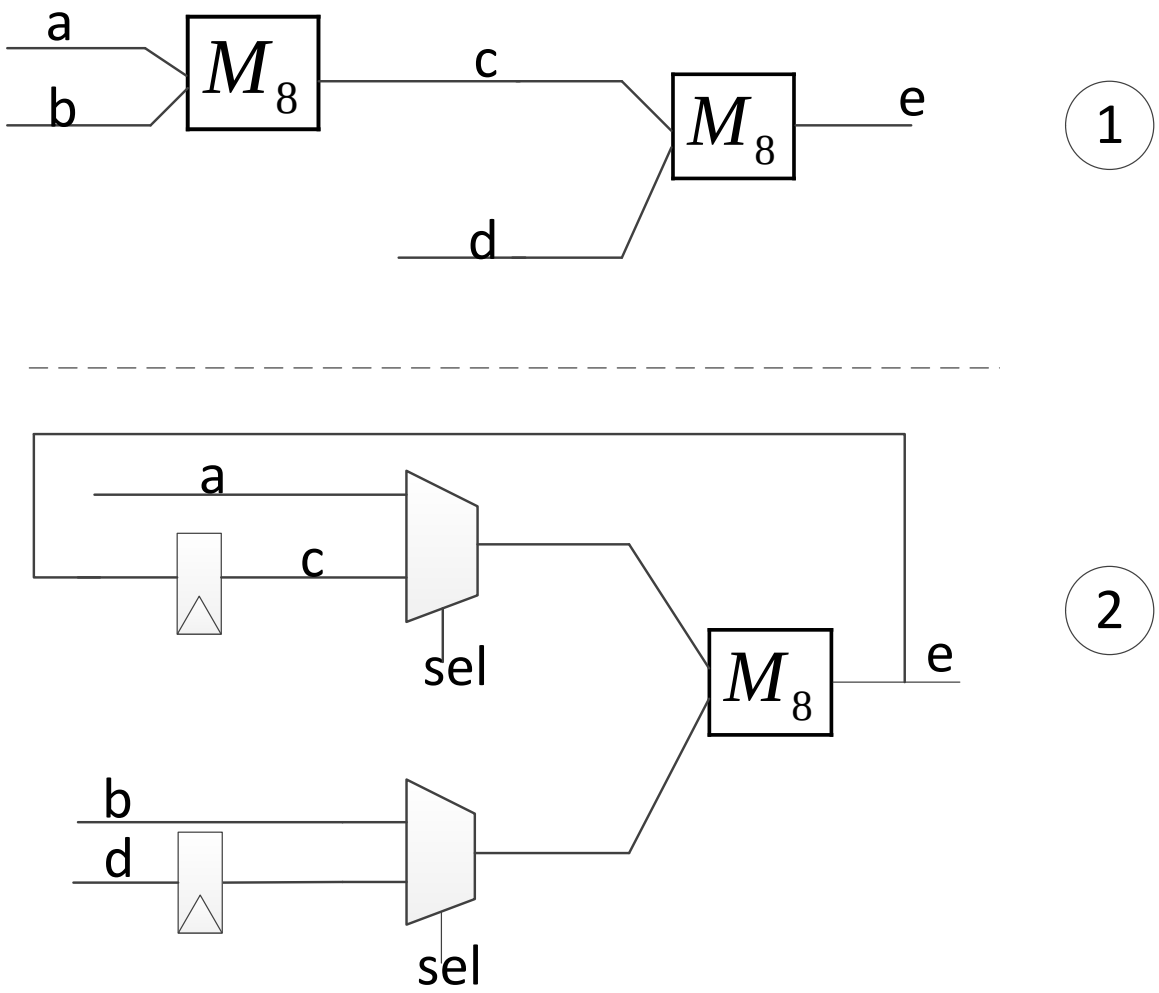


Figure 11: Reuse the multiplier in two consecutive clock cycles.

Integrated hardware architecture for computing WGT-8 and WGP-8

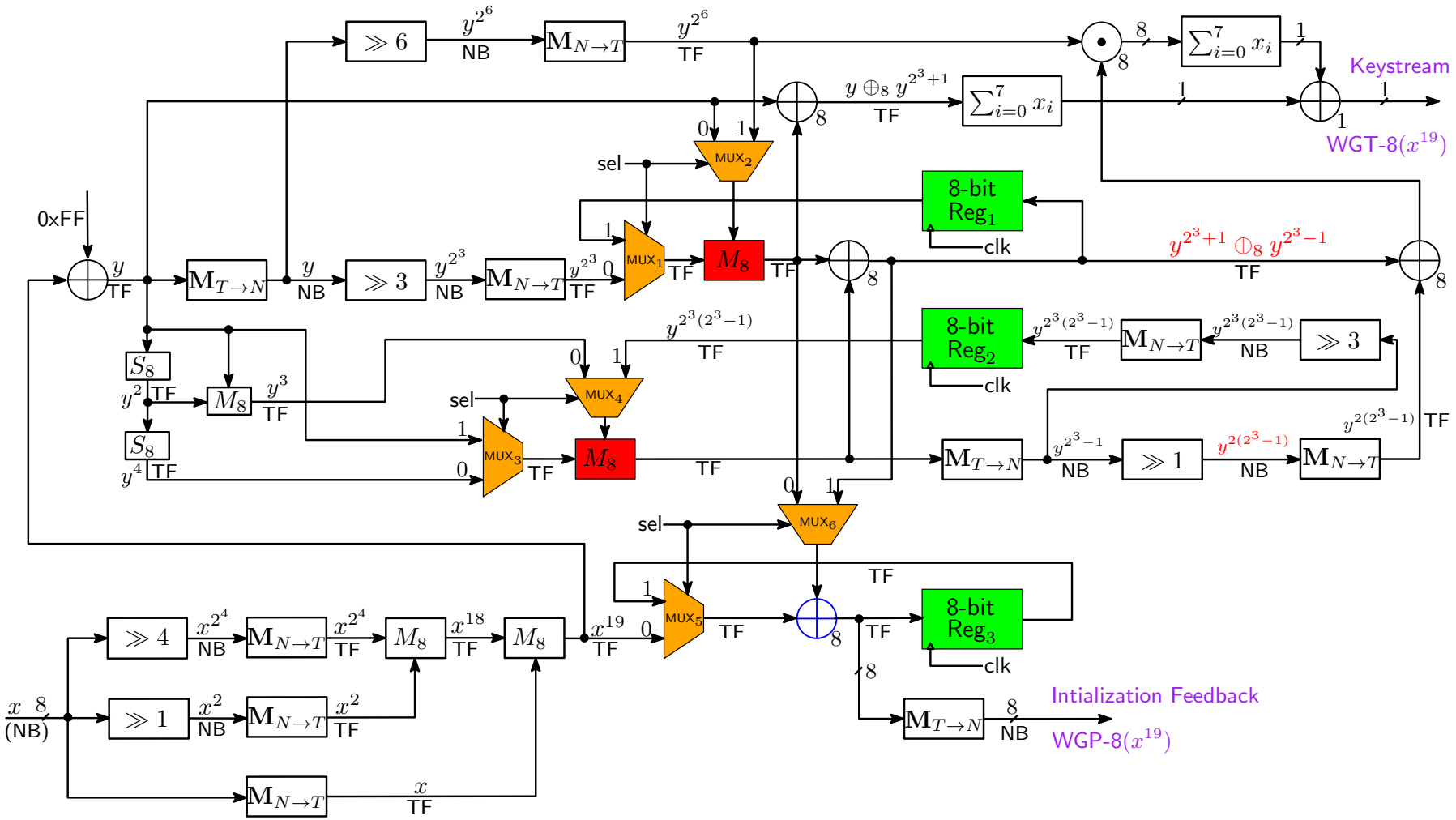
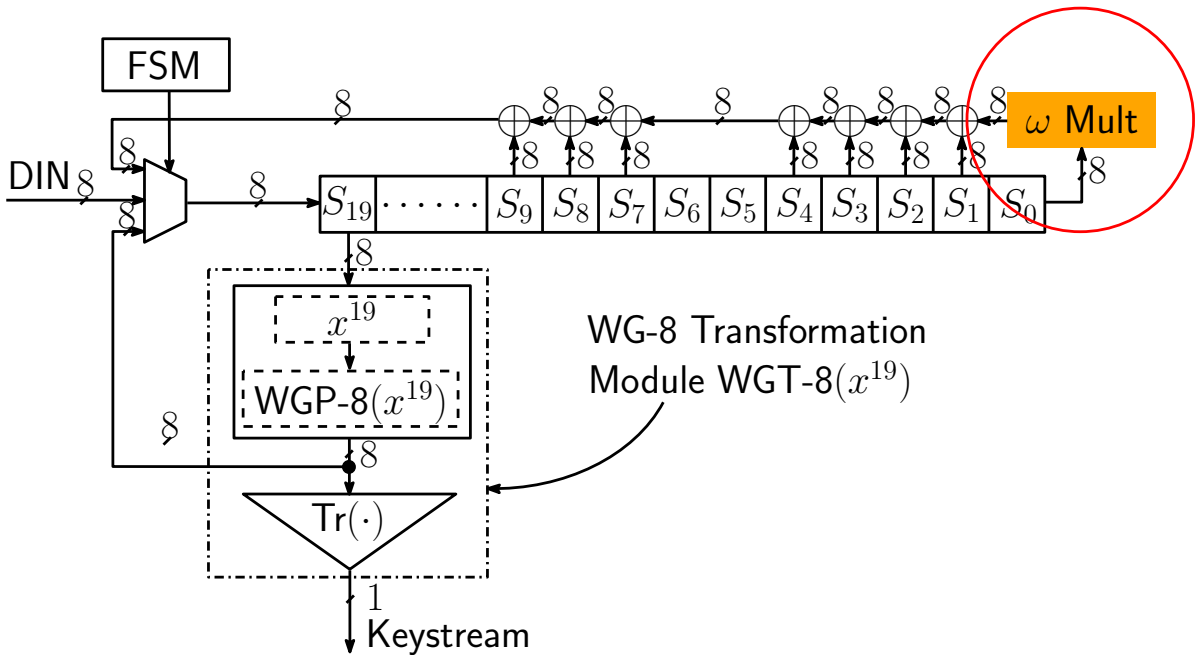


Figure 12: The Integrated Hardware Architecture for Computing WGP-8(x^{19}) and WGT-8(x^{19})

The multiplication by ω 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).
  - $X \cdot \omega = \mathbf{M} \cdot (x'_0, x'_1, \cdots, x'_6, x'_7)^T$. Normal basis (NB).
- Using the 8×8 look-up table.

Results for flip-flops, area, speed, and power consumption analysis of FPGA implementations

Spartan-3(xc3s1000)											
Implementations	Key Size (bits)	IV Size (bits)	Data Rates (bits/ cycle)	#FFs	Area (Slices)	Maximum Frequency (MHz)	Dynamic Power (W)	Maximum Throughput (Mbps)	Optimality		
									T/A (Mbps/ #Slices)	T/P (Mbps/ W)	T/(A*P) (Mbps/ Slices*W)
WG-8 (LUT)	80	80	1	85	137	190	0.005	190	1.39	38000	277.4
WG-8 (LUT)			11	207	398	192	0.016	2112	5.31	132000	331.7
WG-8 (TF 1)			1	83	678	19	0.671	19	0.03	28.3	0.042
WG-8 (TF 1)			11	279	5106	19	4.282	209	0.04	48.8	0.010
WG-8 (TF 2)			1	83	343	42	0.339	42	0.12	123.9	0.36
WG-8 (TF 2)			11	306	2369	42	1.686	462	0.20	274	0.12
WG-8 (TF 3)			1	114	436	49	0.267	49	0.11	183.5	0.42
WG-8 (TF 3)			11	470	2795	44	2.399	484	0.17	201.7	0.07
Grain	80	64	1	–	44	196	–	196	4.45	–	–
Trivium	80	80	1	–	50	240	–	240	4.80	–	–

- 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.

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

CMOS 65nm										
Implementations	Key Size (bits)	IV Size (bits)	Data Rates (bits/ cycle)	Area (GE)	Optimum Frequency (MHz)	Total Power (mW)	Maximum Throughput (Mbps)	Optimality		
								T/A (Mbps/ #GE)	T/P (Mbps/ mW)	T/(A*P) (Mbps/ GE*mW)
WG-8 (LUT)	80	80	1	1786	500	0.983	500	0.28	508.6	0.285
WG-8 (LUT)			11	3942	610	1.344	6710	1.70	4992.6	1.27
WG-8 (TF 1)			1	7523	229	35.8	229	0.03	6.40	0.001
WG-8 (TF 1)			11	42762	122	100.1	1342	0.03	13.4	0.0003
WG-8 (TF 2)			1	3162	260	6.97	260	0.08	37.3	0.012
WG-8 (TF 2)			11	22668	205	44.6	2255	0.10	50.6	0.002
WG-8 (TF 3)			1	2981	254	7.98	254	0.08	31.83	0.011
WG-8 (TF 3)			11	19882	205	53.3	2255	0.11	42.3	0.002
Grain	80	64	1	1126	1020	2.04	1020	0.91	500	0.44
Grain			11	1126	1098	2.25	12078	10.73	5368	4.77
Trivium	80	80	1	1986	962	3.88	962	0.48	247.9	0.12
Trivium			11	2028	990	4	10890	5.37	2722.5	1.342

- 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.

Discussions

- 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

Data Rates	FPGA			ASIC		
	T/A	T/P	T/(A*P)	T/A	T/P	T/(A*P)
1	TF 2	TF 3	TF 3	TF 2/3	TF 2	TF 2
11	TF 2	TF 2	TF 2	TF 3	TF 2	TF 2/3

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

Implementations	Multipliers (Number)	Multiplier (Single area) (slices)	Multiplier (Total areas) (slices)	Multiplexers (Number)
TF 1	7	53	371	0
TF 2	7	29	203	0
TF 3	5	26	130	6

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 4×4 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?