Towards Better Standard Cell Library: Optimizing Compound Logic Gates for TFHE

Kotaro Matsuoka*
Yusuke Hoshizuki**
Takashi Sato*
Song Bian*

2021 Nov. 15

*Kyoto University
**AXELL CORPORATION

Agenda

1. TFHE & Standard Cell
2. Preliminary: Blind Rotate
3. Construction of proposed gates
4. DEMO: Workflow for a 16-bit multiplier
5. Conclusion

TFHE & Standard Cell

TFHE & Standard Cell

TFHE: Torus Fully Homomorphic Encryption

- Can evaluates Logic Circuits without decryption.
- Ordinary logic synthesis tools (ex: Yosys) can be used.

Standard Cell: Optimized implementation of small logic functions

- ex: NAND, Full Adder, AOI21, etc.
- Logic synthesis tools will represent the circuit using Standard Cells.

Original TFHE only supports 2-input-1-output gates (ex: NAND) and MUX

- More sophisticated gates can be used in the logic synthesis.

Propose some 3-input or multi-output gates. Around 2x performance benefit in DEMO.

Preliminary: Blind Rotate

Encoded Message Space in Torus

Torus ($\mathbb{T}$): $\mathbb{R} \bmod 1 \in [-\frac{1}{2}, \frac{1}{2})$

- The message space of TFHE
- Group of the fraction parts of real numbers
  - Supports addition (ex.: $\mathbf{0.4 + 0.7 = 1.1 \equiv 0.1 \bmod 1}$)
  - Supports multiplication with integers (ex. $\mathbf{3 \cdot 0.4 = 1.2 \equiv 0.2 \bmod 1}$)
- Need to encode the plaintext space, Binary $\{0, 1\}$, into Torus.

$\mathbb{M}_t = \{-\frac{1}{t}, \frac{1}{t}\}$: Encoded message space

- $t \in \mathbb{N}$. Corresponds $\{0, 1\}$ respectively.
- We use $\mathbb{M}_8$ and $\mathbb{M}_{12}$
  - $\mathbb{M}_8$: Used in the original TFHE implementation
  - $\mathbb{M}_{12}$: More performance benefit but increase the decryption error rate

Blind Rotate

Blind Rotate: The core functionality of TFHE

- Can evaluate Look Up Table (LUT)
  - LUT is represented as a polynomial $\boldsymbol{TV}[\boldsymbol{X}] \in \mathbb{T}[\boldsymbol{X}]/\boldsymbol{X}^{\boldsymbol{N}} + \mathbf{1}$
  - $\boldsymbol{\rho} \in \mathbb{Z}/\boldsymbol{2N}\mathbb{Z}$ is the (encrypted) index input
  - Output: The constant term of $\boldsymbol{X}^{-\rho} \cdot \boldsymbol{TV}[\boldsymbol{X}]$

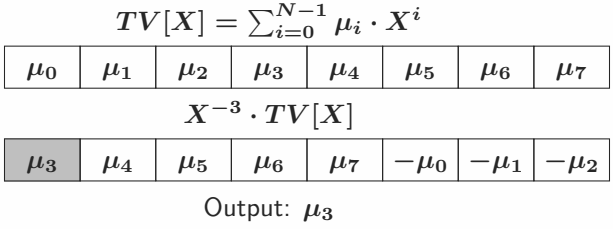

Figure 1: Blind Rotate ($\boldsymbol{\rho = 3, N = 8}$)

Blind Rotate: LUT constraints

Possible LUTs for BR have two constraints.

1. Negacyclic Rotation: $X^{-\rho} \cdot TV[X] = -X^{-(\rho+N)} \cdot TV[X]$
2. Linear Combination: Only $\frac{t}{4}$ degrees of freedom for LUT entries

$X^{-3} \cdot TV[X]$

μ_3	μ_4	μ_5	μ_6	μ_7	$-\mu_0$	$-\mu_1$	$-\mu_2$

$X^{-8} \cdot TV[X]$

$-\mu_0$	$-\mu_1$	$-\mu_2$	$-\mu_3$	$-\mu_4$	$-\mu_5$	$-\mu_6$	$-\mu_7$

$X^{-(3+8)} \cdot TV[X]$

$-\mu_3$	$-\mu_4$	$-\mu_5$	$-\mu_6$	$-\mu_7$	μ_0	μ_1	μ_2

Figure 2: Negacyclic Rotation ($N = 8$)

Blind Rotate: LUT constraints

Possible LUTs for BR have two constraints.

1. Negacyclic Rotation: $X^{-\rho} \cdot TV[X] = -X^{-(\rho+N)} \cdot TV[X]$
2. Linear Combination: Only $\frac{t}{4}$ degrees of freedom for LUT entries
   - There are only $\frac{t}{2}$ possible values (without error) for ρ
   - p : number of inputs, m_i : encoded messages of inputs
   - $a_i \in \mathbb{Z}, b \in [-\frac{t}{2}, \frac{t}{2})$
   - $\rho \approx \lceil 2N \cdot (\sum_{i=0}^{p-1} a_i \cdot m_i + \frac{b}{t} \bmod 1) \rfloor$

ex.) $p = 2, m_i \in \mathbb{M}_8, a_i = 1, b = 1$

$\Rightarrow \rho \approx \lceil 2N \cdot (m_0 + m_1 + \frac{1}{8}) \rfloor \in \{2N \cdot \frac{1}{8}, 2N \cdot \frac{3}{8}, -2N \cdot \frac{3}{8}, -2N \cdot \frac{1}{8}\}$

$TV[X] = \sum_{i=0}^{1} \sum_{j=0}^{\frac{N}{2}-1} \mu_i \cdot X^{i \cdot \frac{N}{2}+j}$

μ_0	μ_0	μ_0	μ_0	μ_1	μ_1	μ_1	μ_1

Output for $\rho = 2N \cdot \frac{1}{8}$ Output for $\rho = 2N \cdot \frac{3}{8}$

Construction of proposed gates

List of proposed gates

- $\mathbb{M}_{\mathbf{8}}$
  - Half Adder (Gao's method)
  - 2BR Full Adder
- $\mathbb{M}_{\mathbf{12}}$
  - 1BR Full Adder (Multi Value)
  - AOI21 and OAI21

List of proposed gates

- $\mathbb{M}_8$
  - Half Adder (Gao's method)
  - 2BR Full Adder
- $\mathbb{M}_{12}$
  - 1BR Full Adder (Multi Value)
  - AOI21 and OAI21

1BR Full Adder

- Full Adder = 3-input XOR gate + 3-input majority gate
  - 1BR Full Adder evaluate this by one Blind Rotate
- Extended from FHEW version[1]
- Construction:
  1. Construct 3-input XOR
  2. Construct 3-input majority gate
  3. Merge above by Multi-value technique

[1] Jean-François Biasse and Luis Ruiz. "FHEW with Efficient Multibit Bootstrapping". In: *Progress in Cryptology – LATINCRYPT 2015*. Ed. by Kristin Lauter and Francisco Rodríguez-Henríquez. Cham: Springer International Publishing, 2015, pp. 119–135. ISBN: 978-3-319-22174-8.

3-input XOR

- 3-input XOR $= \boldsymbol{A} \oplus \boldsymbol{B} \oplus \boldsymbol{C}$
- All inputs can be interchanged
  - $\rho \approx \lceil 2N \cdot (\sum a_i \cdot m_i + \frac{b}{t} \bmod 1) \rfloor$
  - $a_A = a_B = a_C = 1, b = 0$
- $\frac{\rho}{2N}$ of yellow and white satisfy:
  $2N \cdot -\frac{3}{12} + N \equiv 2N \cdot \frac{3}{12} \bmod 2N$

Table 1: Output and $\frac{\rho}{2N}$ of 3-input XOR

c\ab	00	01	11	10
0	$0/-\frac{3}{12}$	$1/-\frac{1}{12}$	$0/\frac{1}{12}$	$1/-\frac{1}{12}$
1	$1/-\frac{1}{12}$	$0/\frac{1}{12}$	$1/\frac{3}{12}$	$0/\frac{1}{12}$

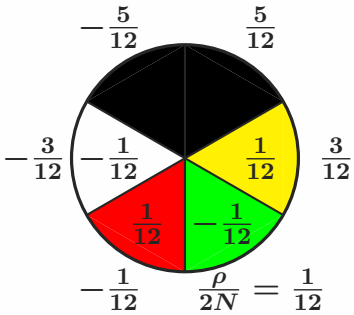

Figure 3: 3-input XOR LUT

3-input majority gate

- 3-input majority gate = $(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$
- All inputs can be interchanged
  - Same as 3-input XOR
  - $a_A = a_B = a_C = 1, b = 0$

Table 2: Output and $\frac{\rho}{2N}$ of 3-input majority gate

c\ab	00	01	11	10
0	$0/-\frac{3}{12}$	$0/-\frac{1}{12}$	$1/\frac{1}{12}$	$0/-\frac{1}{12}$
1	$0/-\frac{1}{12}$	$1/\frac{1}{12}$	$1/\frac{3}{12}$	$1/\frac{1}{12}$

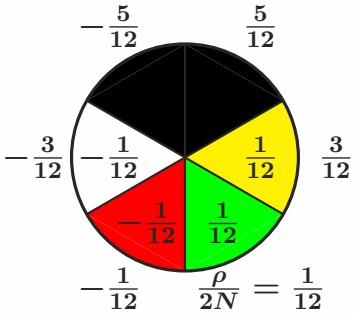

Figure 4: 3-input majority gate LUT

Multi-value technique

- Blind Rotate is a heavy operation.
- $\boldsymbol{TV}[\boldsymbol{X}]$ for 3-input XOR and majority gate have common divisor
  - $\boldsymbol{TV_0}[\boldsymbol{X}] = \sum_{i=0}^{\lfloor \frac{N}{3} \rfloor - 1} \frac{1}{12} \boldsymbol{X}^i$
- $\boldsymbol{TV}[\boldsymbol{X}]$ for 3-input XOR: $\boldsymbol{TV_{xor}}[\boldsymbol{X}] \cdot \boldsymbol{TV_0}[\boldsymbol{X}]$
  - $\boldsymbol{TV_{xor}}[\boldsymbol{X}] = -\boldsymbol{1} + \boldsymbol{X}^{\lfloor \frac{N}{3} \rfloor} - \boldsymbol{X}^{\boldsymbol{2} \cdot \lfloor \frac{N}{3} \rfloor}$
- $\boldsymbol{TV}[\boldsymbol{X}]$ for 3-input majority gate: $\boldsymbol{TV_{majority}}[\boldsymbol{X}] \cdot \boldsymbol{TV_0}[\boldsymbol{X}]$
  - $\boldsymbol{TV_{majority}}[\boldsymbol{X}] = \boldsymbol{1} + \boldsymbol{X}^{\lfloor \frac{N}{3} \rfloor} + \boldsymbol{X}^{\boldsymbol{2} \cdot \lfloor \frac{N}{3} \rfloor}$

$\therefore$ We can share the output of one BR

$\boldsymbol{i} \in \{\boldsymbol{xor}, \boldsymbol{majority}\}$

$$\boldsymbol{X}^{-\rho} \cdot \boldsymbol{TV}[\boldsymbol{X}] = \boldsymbol{X}^{-\rho} \cdot (\boldsymbol{TV_i}[\boldsymbol{X}] \cdot \boldsymbol{TV_0}[\boldsymbol{X}]) = \boldsymbol{TV_i}[\boldsymbol{X}] \cdot \underbrace{(\boldsymbol{X}^{-\rho} \cdot \boldsymbol{TV_0}[\boldsymbol{X}])}_{\text{Blind Rotate}}$$

AOI21

- AOI: AND OR Inverter
- $\neg((A \wedge B) \vee C)$
  - c is not interchangeable
    - Must be treated differently from a and b
    - $a_A = a_B = 1$, $a_C = 2, b = 1$

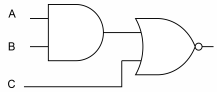

Figure 6: AOI21

Table 3: Output and $\frac{\rho}{2N}$ of AOI21

c\ab	00	01	11	10
0	$1/-\frac{3}{12}$	$1/-\frac{1}{12}$	$0/\frac{1}{12}$	$1/-\frac{1}{12}$
1	$0/\frac{1}{12}$	$0/\frac{3}{12}$	$0/\frac{5}{12}$	$0/\frac{3}{12}$

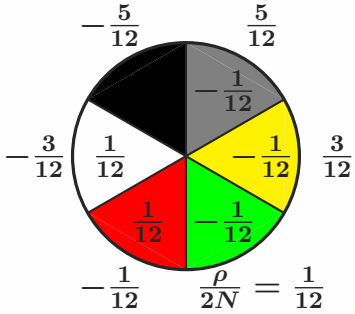

Figure 5: AOI21 LUT

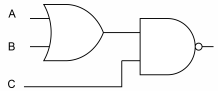

Figure 8: OAI21

- OAI: OR AND Inverter
- $\neg((A \vee B) \wedge C)$
  - $a_A = a_B = 1$, $a_C = 2, b = -1$
  - Same $TV[X]$ but use a different pie

Table 4: Output and $\frac{\rho}{2N}$ of OAI21

c\ab	00	01	11	10
0	$1/-\frac{5}{12}$	$1/-\frac{3}{12}$	$1/-\frac{1}{12}$	$1/-\frac{3}{12}$
1	$1/-\frac{1}{12}$	$0/\frac{1}{12}$	$0/\frac{3}{12}$	$0/\frac{1}{12}$

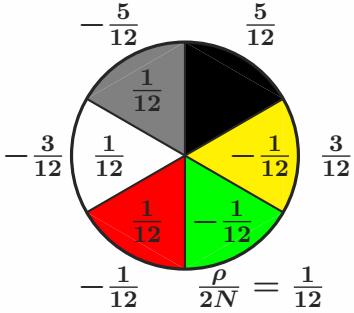

Figure 7: OAI21 LUT

DEMO: Workflow for a 16-bit multiplier

List of Software

- oveus-tfhe[2]: Our implementation derived from TFHEpp[3]
- Yosys[4]: Logic synthesis tool
- Sudachi[5]: Homomorphic logic circuit execution engine

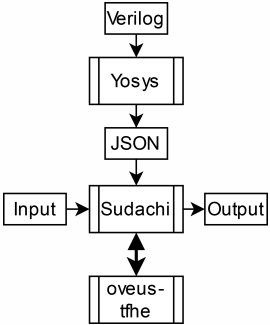

Figure 9: Workflow

[2] https://github.com/axell-corp/oveus-tfhe
[3] https://github.com/virtualsecureplatform/TFHEpp
[4] https://github.com/YosysHQ/yosys
[5] https://github.com/virtualsecureplatform/Sudachi

Conclusion

Conclusion

- We proposed some 3-input or multi-output gates
  - Demonstrated proposed gates give performance benefit in the real environment
  - Implementation: `https://github.com/axell-corp/oveus-tfhe`
- There is room for more sophisticated standard cells
  - More than depth 2 BR functions
    - SWAP gate in the implementation
  - $t > 12$ cases
    - 4-input AND and OR in $\mathbb{M}_{16}$ in the implementation