NIST 5th PQC Standardization Conference

Novel Schoolbook-Originated Polynomial Multiplication Accelerators for NTRU-based PQC

Yazheng Tu[1], Shi Bai[2], Jinjun Xiong[3], and Jiafeng Xie[1]

[1]: Electrical and Computer Engineering Department, Villanova University

[2]: Mathematics and Statistic Department, Florida Atlantic University

[3]: Computer Science and Engineering Department, University at Buffalo

Outline

- Introduction
- Preliminary
- Point-Wise Multiplier
- SCOPE-I: The First Accelerator
- SCOPE-II: The Second Accelerator
- Evaluation
- Future works

Introduction

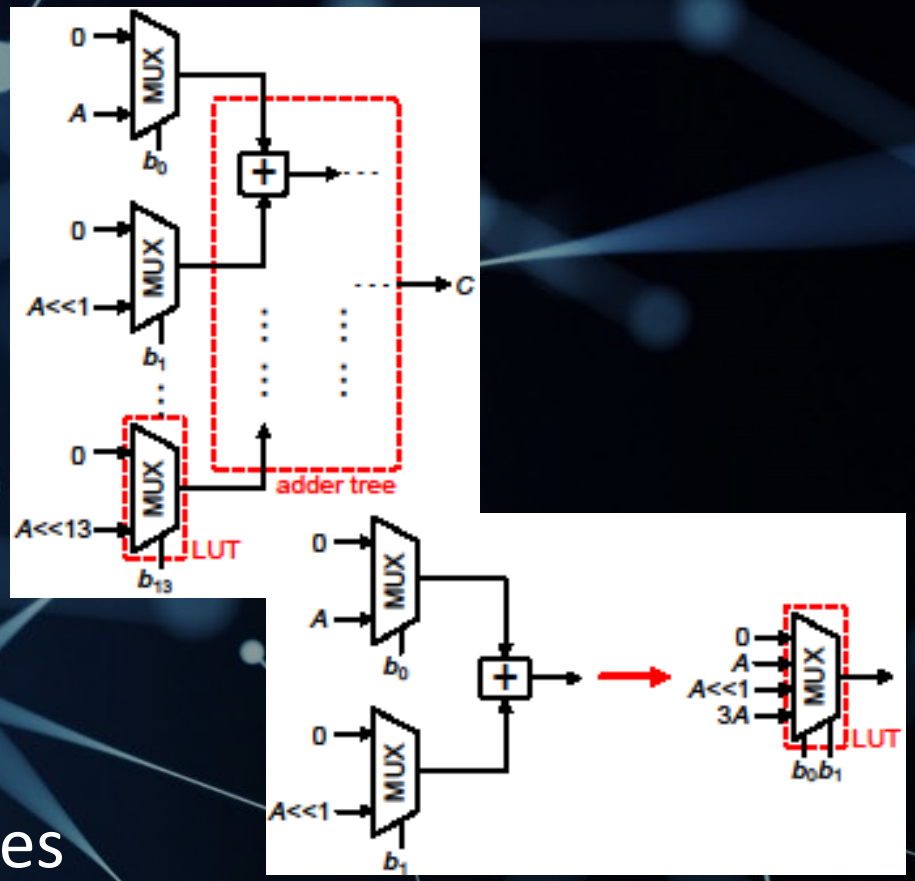
- **Background**
  - NTRU-based PQC is an important branch of lattice-based cryptography
  - Not many specific works carried out
- **Motivation**
  - When a complete polynomial multiplication is needed, solutions other than NTT can be explored
- **Contributions**
  - A novel LUT-based point-wise multiplier combined with modulo reduction
  - A novel polynomial multiplier architecture incorporating the developed point-wise multiplier
  - A TMVP-based accelerator with innovations in algorithm and architecture.
  - A thorough evaluation ensuring the efficiency of the proposed strategy

Preliminary

- **Notations**
  - n: the size of the polynomials; q: the modulus
  - G, D: input polynomials, $G = \sum_{i=0}^{n-1} g_i x^i$, $D = \sum_{i=0}^{n-1} d_i x^i$, where g_i, d_i are coefficients
  - W: the output polynomial, $W = \sum_{i=0}^{n-1} w_i x^i$, where w_i are coefficients
- **NTRU-based PQC**
  - FALCON: Fast Fourier lattice-based compact signatures over NTRU, built on [7]
  - NTRU: a merger of NTRUEncrypt and NTRU-HRSS-KEM
- **Schoolbook-based Polynomial Multiplication**
  - $W = GD \bmod f(x)$; $f(x) = x^n + 1$ for Falcon, $f(x) = x^n - 1$ for NTRU
  - $[W] = [G] \times [D]$, $[W]$, $[D]$ are $n \times 1$ vectors while $[G]$ is a $n \times n$ circulant matrix
- **TMVP-based method**
  - $$\begin{bmatrix} W_0 \\ W_1 \end{bmatrix} = \begin{bmatrix} G_0 & G_2 \\ G_1 & G_0 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \end{bmatrix} = \begin{bmatrix} G_0 & -G_1 \\ G_1 & G_0 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \end{bmatrix} = \begin{bmatrix} G_0(D_0 + D_1) + (-G_0 - G_1)D_1 \\ G_0(D_0 + D_1) + (-G_0 + G_1)D_0 \end{bmatrix}$$

Point-Wise Multiplier (Cont.)

- **Consideration**
  - $C = A \times B$; C: 28 bits, A, B: 14 bits
  - $B = \sum_{j=0}^{13} b_i 2^j$, b_i: bits of B
  - $C = A \times \sum_{j=0}^{13} b_i 2^j = \sum_{j=0}^{13} (A \times 2^j) b_i$
- **Proposal**
  - Multiplication equivalent to adding 14 MUXes
  - Combine neighboring MUXes into a larger one to reduce number of MUXes
  - Trade-off between the size of the MUX and the complexity of summation
  - Problem becomes the finding of an optimal number/ size of MUXes
  - Final recommendation of B: 4, 4, 3, and 3 bits
    - 4 MUXes, each with 2^4, 2^4, 2^3 and 2^3 inputs

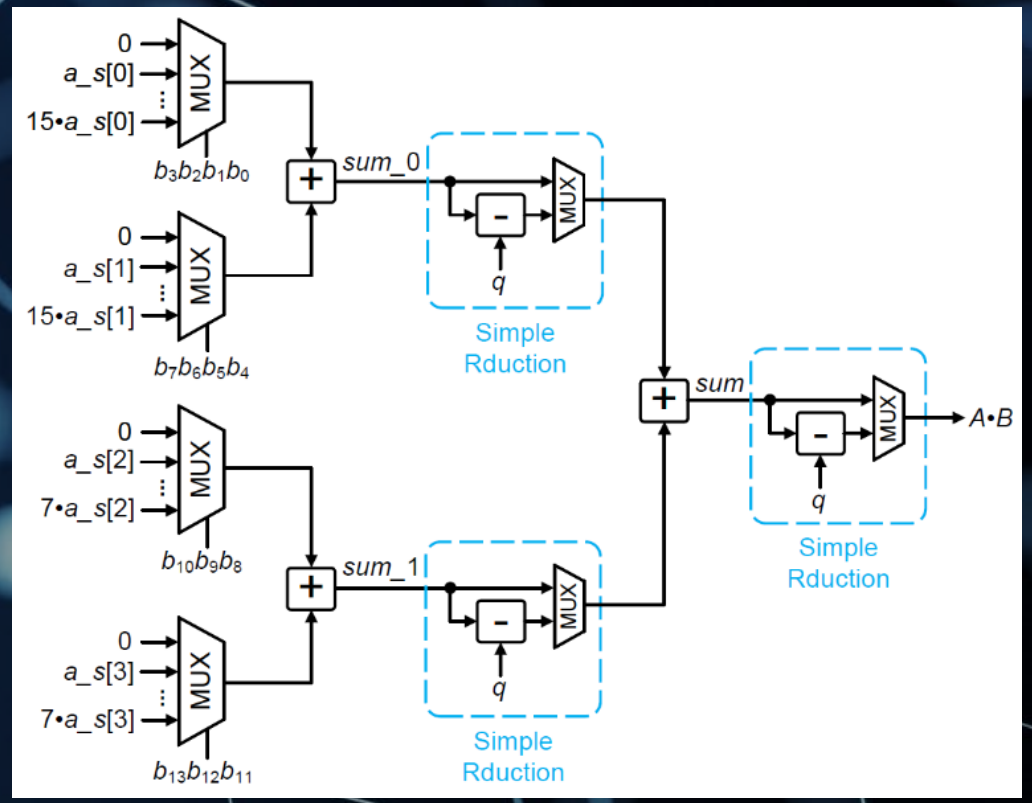


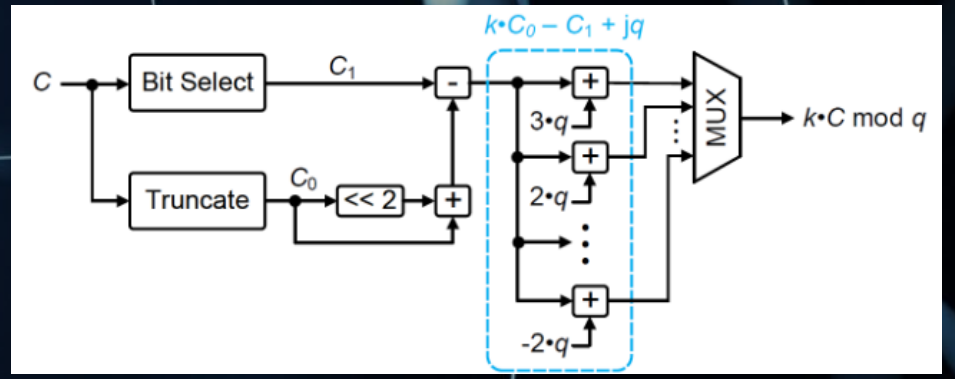
Point-Wise Multiplier (Cont.)

- **Modular Reduction**
  - Traditional methods execute reduction at the end
    - Drawback-1: Leads to difference in bit-width from each MUX
    - Drawback-2: Need to expand all signals to 28 bit
  - Propose to execute reduction at the beginning. **Key benefits** include:
    - Reduce shifted values before feeding to MUXes
    - Scale back to 14-bit
    - In the range [0, q)
    - Only Simple Reduction is needed in the following calculation
    - Halve the bit-width
    - Reduce the critical path

Point-Wise Multiplier (Cont.)

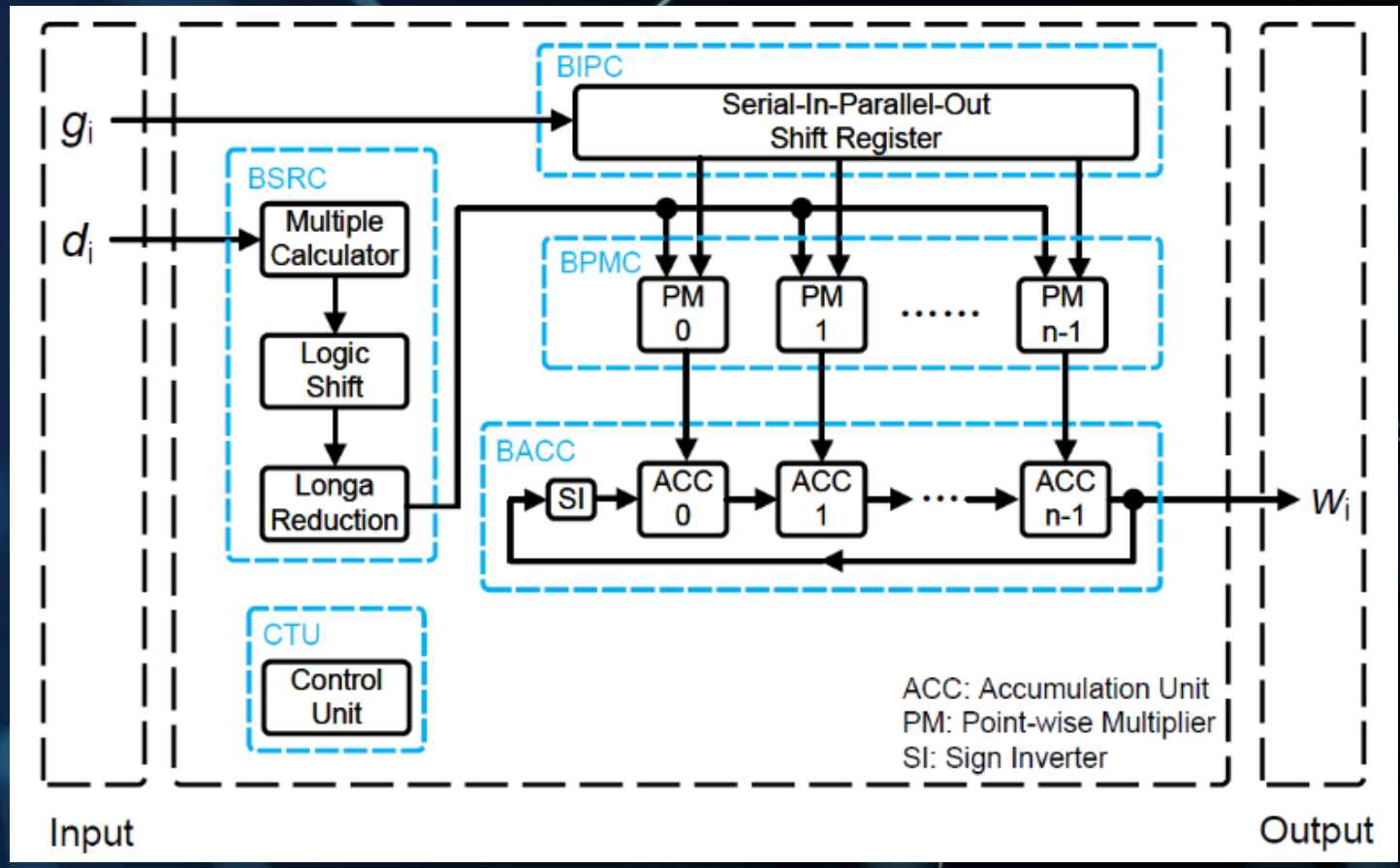
- **Hardware Structure**
  - Components:
    - Two 16-to-1 MUXes
    - Two 8-to-1 MXUes
    - One 2-layer adder tree
    - Three Simple Reduction units
- **Longa Reduction Unit (K-red)**
  - Deploys K-red in [29]
    - $k \cdot C \bmod q$
    - k = 3 for q = 12289, differs for different q
  - Select correct answer from different values
  - C pre-multiplied with modular inverse of k





SCOPE-I: The First Accelerator

- **Proposed Algorithm**
  - Calculates the product of one column of $[G]$ and one d_i at one time
- **Proposed SCOPE-I Overview**
  - Five main components
    - Basic Input Process Component (BIPC)
    - Basic Shift and Reduction Component (BSRC)
    - Basic Point-wise Multiplier Component (BPMC)
    - Basic Point-wise Multiplier Component (BPMC)
    - ConTrol Unit (CTU)
  - Time Complexity: $(n + x)$ cycles
    - x: pipeline register layers
- **Basic Input Process Component (BIPC)**
  - Load and output g_i's
  - Serial-in parallel-out shift register

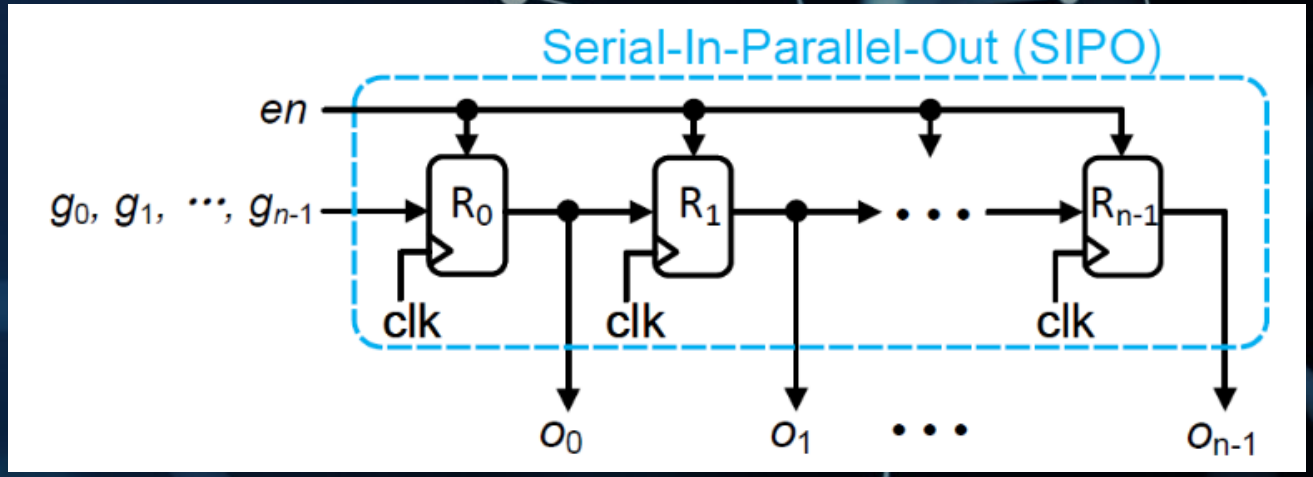


SCOPE-I: The First Accelerator

- **Proposed Algorithm**
  - **Inputs:**
    - $G = \sum_{i=0}^{n-1} g_i x^i$
    - $D = \sum_{i=0}^{n-1} d_i x^i$
  - **Output:**
    - $W = \sum_{i=0}^{n-1} w_i x^i$
  - Parallel calculation
    - Calculates the product of one column of $[G]$ and one d_i at one time
    - Complexity of $O(n)$
  - Serial Input
  - Serial Output

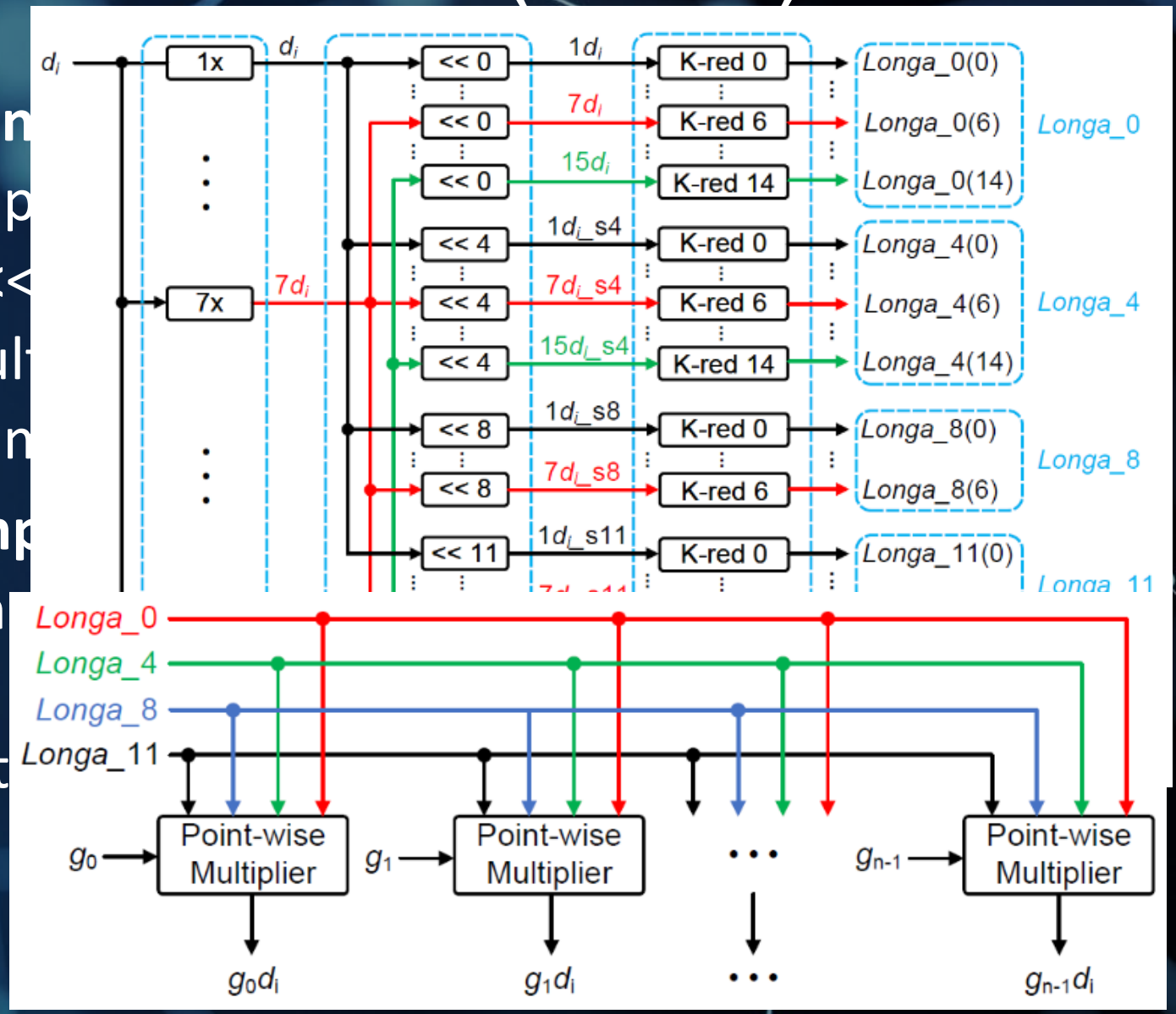
SCOPE-I: The First Accelerator (Cont.)

- **Basic Input Process Component (BIPC)**
  - Responsible load g_i and output g_i's in parallel during calculation
  - Contains n 14-bit registers
  - Load one g_i at each cycle
  - Feed to first register (R_0) and shift the others
  - Take n cycles for loading
  - Signal en stops shifting by disabling the registers
    - Parallel outputs



SCOPE-I: The First Accelerator (Cont.)

- **Basic Shift and Reduction Componen**
  - •Calculate 0-7/15 multiples of the inp
    - •E.g. $7x = 4x + 3x = (x << 2) + (x <<$
  - •Perform logic shift to calculated mul
  - •Execute Longa reduction to shifted n
- **Basic Point-wise Multiplication Comp**
  - Calculates point-wise multiplication
    - Contains n proposed PWM
    - All PWM shares one BSRC output
    - Each PWM takes one different g_i
  - Output n $g_i \cdot d_i$ mod q in parallel



SCOPE-I: The First Accelerator (Cont.)

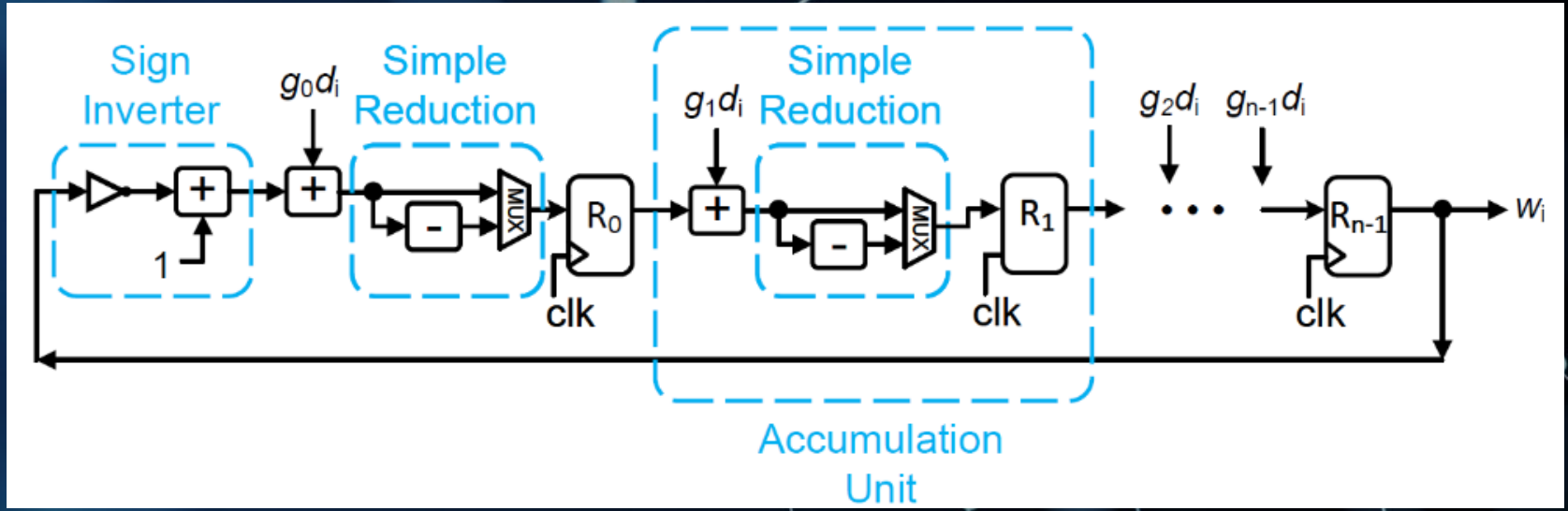
- **Basic ACcumulation Component (BACC)**
  - Accumulates the point-wise products to form the final answer.
  - An example of the accumulation process with n = 4 is provided in the table.
  - Components
    - n 14-bit registers.
    - n full adders.
    - Simple Reductions

Cycle	d_i	R_0	R_1	R_2	R_3
0	d_3	g_0d_3	g_1d_3	g_2d_3	g_3d_3
1	d_2	$g_0d_2 - g_3d_3$	$g_1d_2 + g_0d_3$	$g_2d_2 + g_1d_3$	$g_3d_2 + g_2d_3$
2	d_1	$g_0d_1 - g_3d_2 - g_2d_3$	$g_1d_1 + g_0d_2 - g_3d_3$	$g_2d_1 + g_1d_2 + g_0d_3$	$g_3d_1 + g_2d_2 + g_1d_3$
3	d_0	$g_0d_0 - g_3d_1 - g_2d_2 - g_1d_3$	$g_1d_0 + g_0d_1 - g_3d_2 - g_2d_3$	$g_2d_0 + g_1d_1 + g_0d_2 - g_3d_3$	$g_3d_0 + g_2d_1 + g_1d_2 + g_0d_3$

Accumulation Process of The Proposed First Accelerator (SCOPE-I) with n = 4

SCOPE-I: The First Accelerator (Cont.)

- **Basic ACcumulation Component (BACC)**
  - Output the accumulated values in serial
- **ConTrol Unit (CTU)**
  - Finite State Machine
    - *reset:* After *clr*
    - *load:* n *cycles*
    - *multi:* $(n + x)$ *cycles*
    - *output:* n *cycles*
    - *done:* last until *clr* received again
- **Pipeline register layers:** $x = 4$

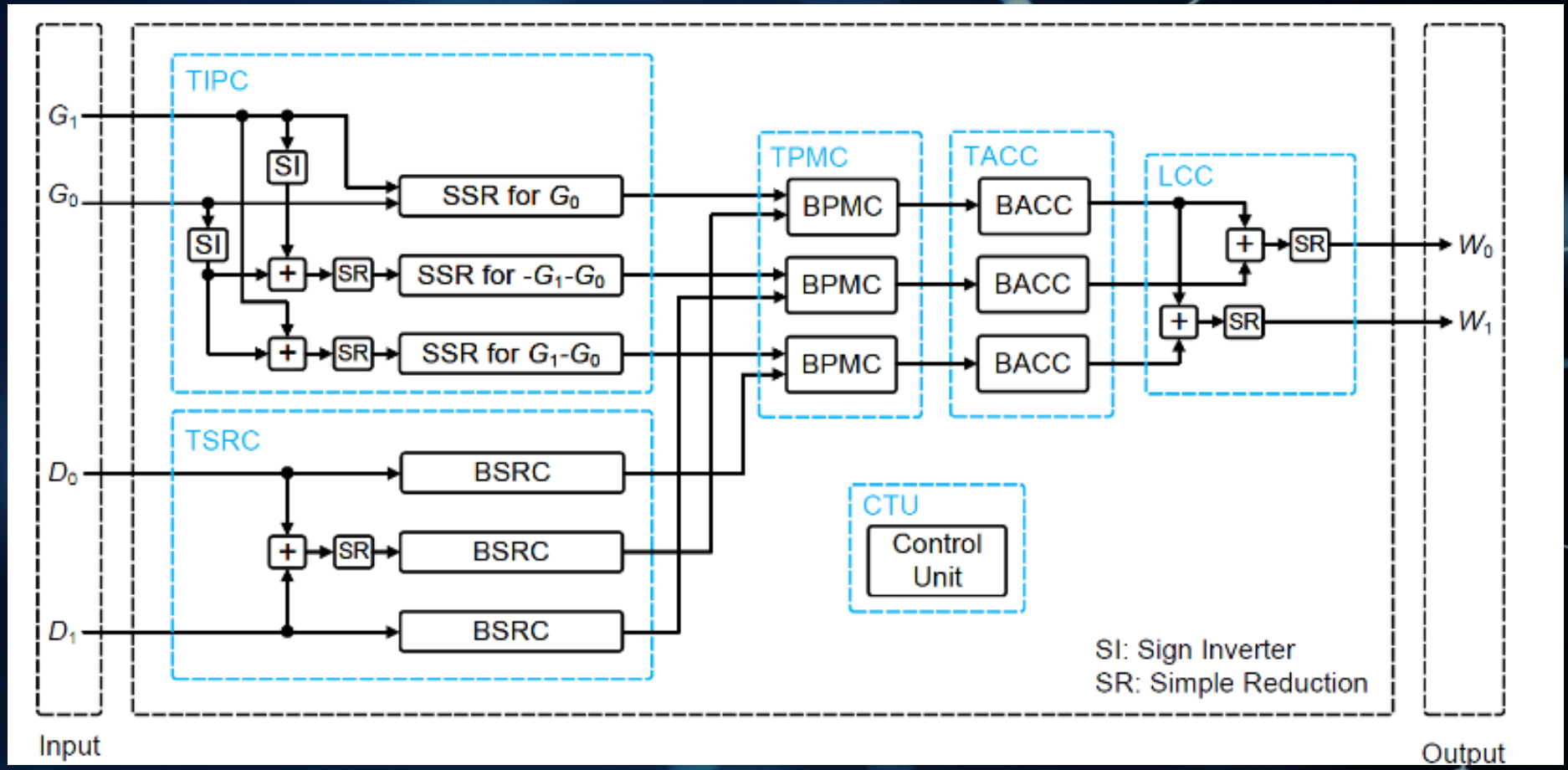


SCOPE-II: The Second Accelerator

- **Proposed Algorithm**
  - $\begin{bmatrix} W_0 \\ W_1 \end{bmatrix} = \begin{bmatrix} G_0 & G_2 \\ G_1 & G_0 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \end{bmatrix} = \begin{bmatrix} G_0 & -G_1 \\ G_1 & G_0 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \end{bmatrix} = \begin{bmatrix} G_0(D_0 + D_1) + (-G_0 - G_1)D_1 \\ G_0(D_0 + D_1) + (-G_0 + G_1)D_0 \end{bmatrix}$
  - **Inputs:**
    - $G = \sum_{i=0}^{n-1} g_i x^i$
    - $D = \sum_{i=0}^{n-1} d_i x^i$
  - **Output:**
    - $W = \sum_{i=0}^{n-1} w_i x^i$
  - $[G_0^i]_j$: the element in the ith row and the jth column of $[G_0]$
  - $[W_0^i]$: the element in the ith row of $[W_0]$

SCOPE-II: The Second Accelerator (Cont.)

- **Proposed SCOPE-II Overview**
  - Time Complexity: $(n + x)$ cycles, where x is the pipeline register layers

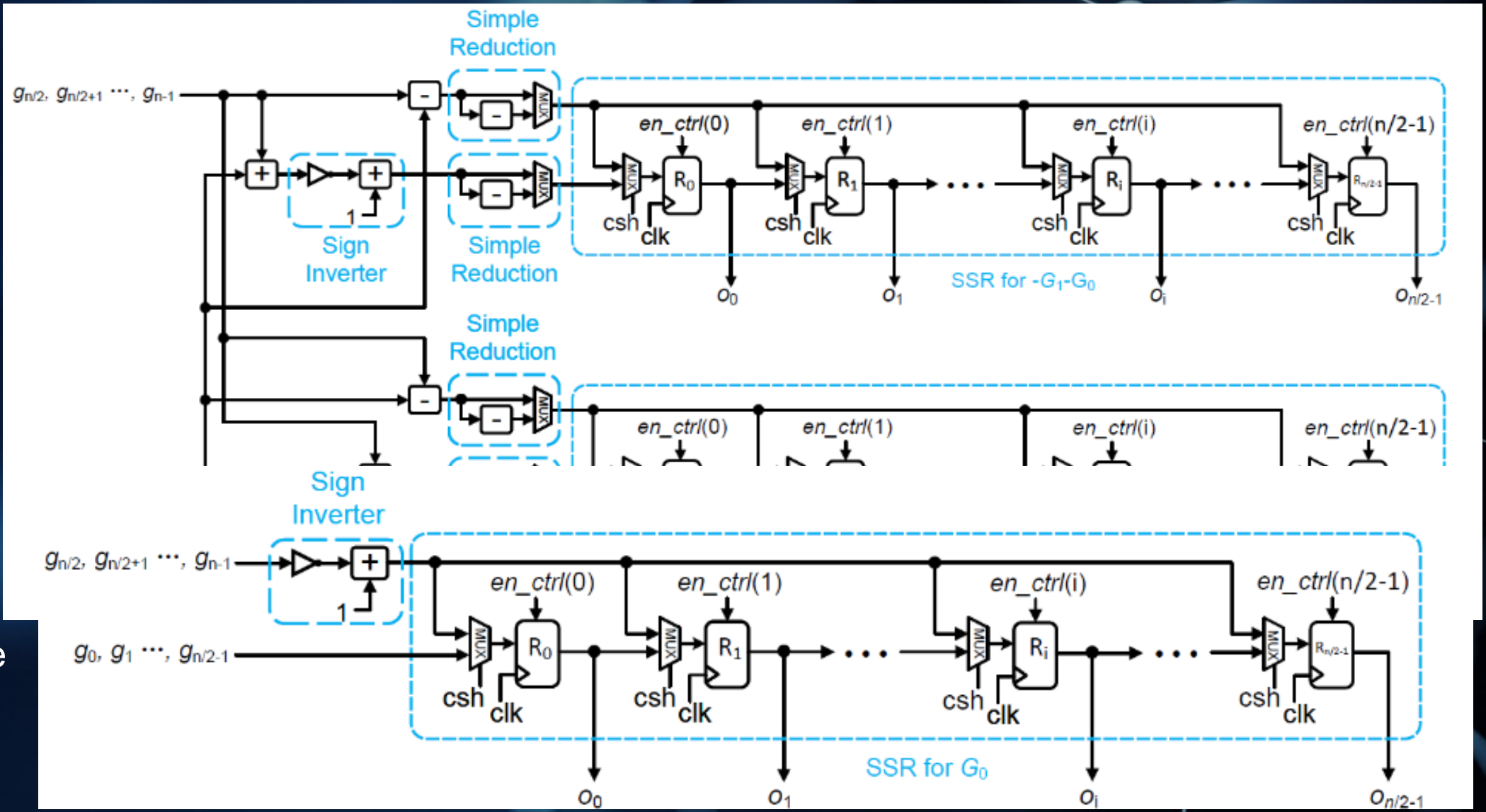


SCOPE-II: The Second Accelerator (Cont.)

- **TIPC**

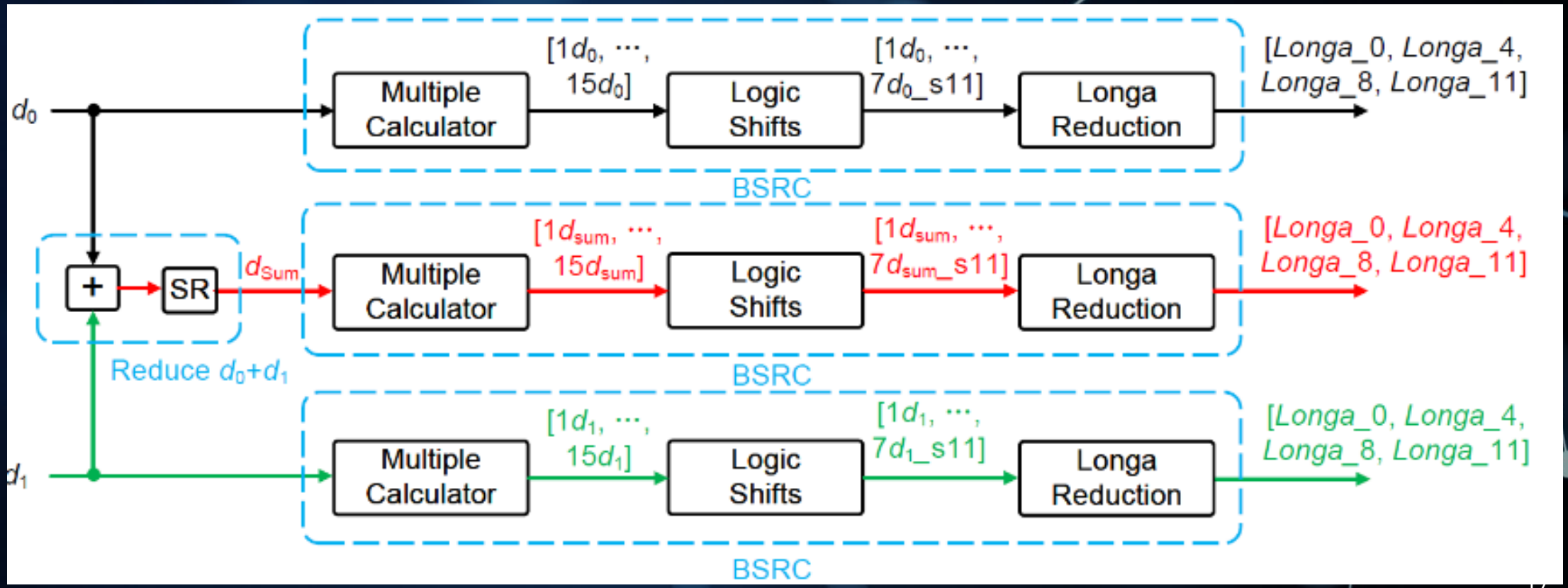
Details of the SSRs for $(-G_1\ -G_0)$ and $(G_1\ -G_0)$

Details of the SSRs for G_0



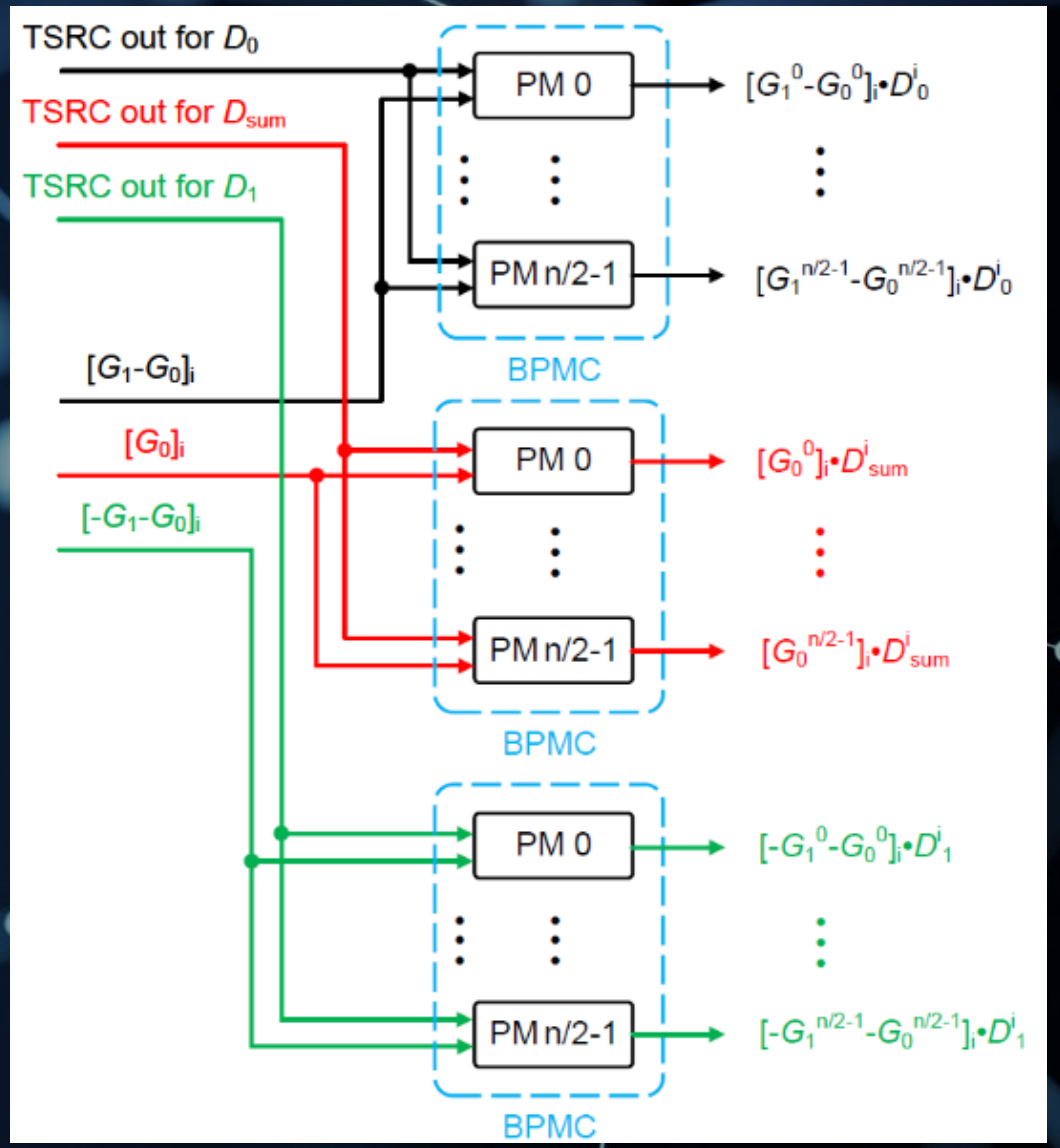
SCOPE-II: The Second Accelerator (Cont.)

- **TMVP Shift and Reduction Component (TSRC)**
  - Responsible for process of D_0, D_1, (D_0 + D_1), respectively.
  - Multiply, shift, and execute Longa Reduction in each BSRC.



SCOPE-II: The Second Accelerator (Cont.)

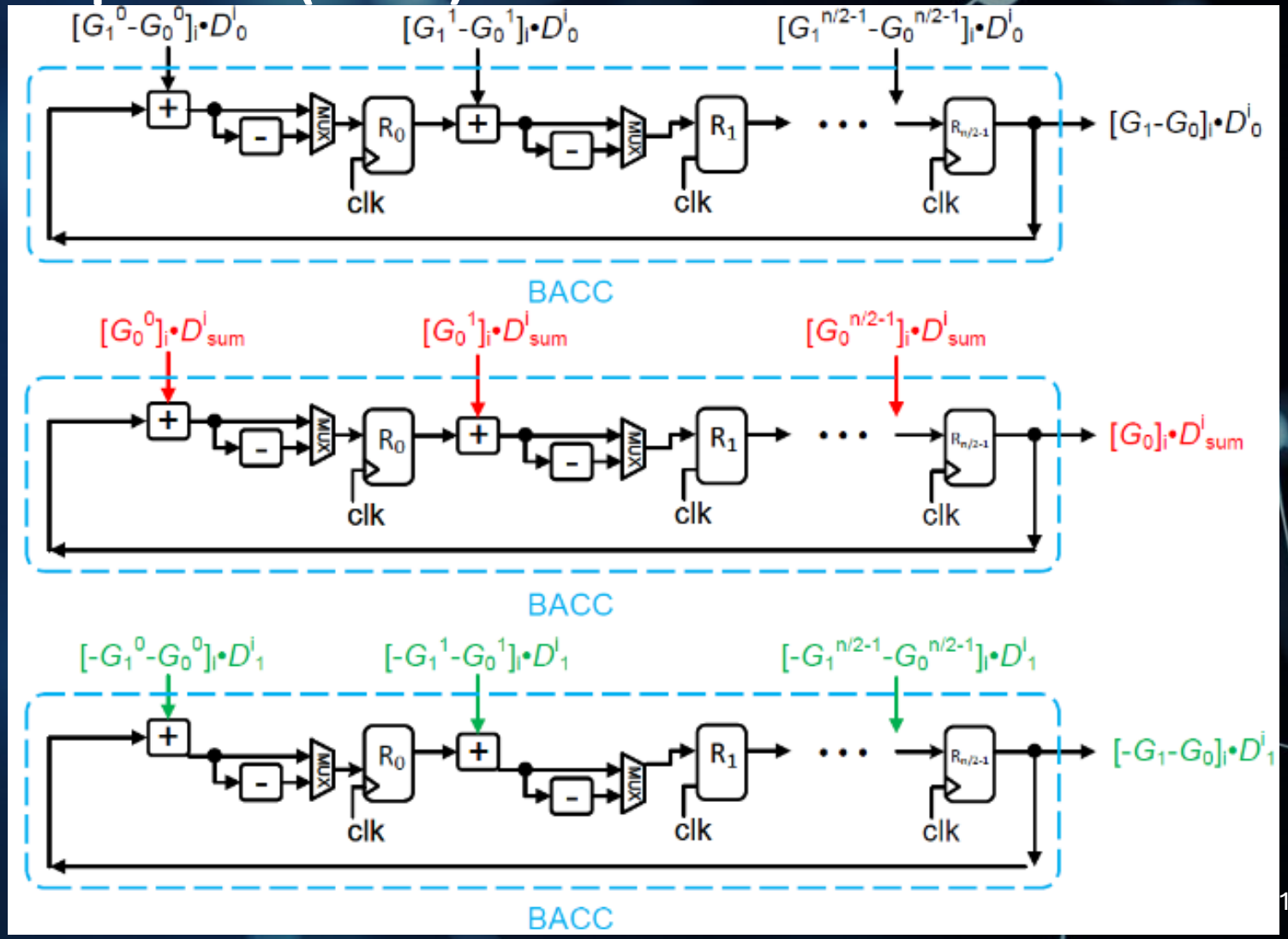
- **TMVP Point-wise Multiplication Component**
  - Contains three individual BPMCs
  - Responsible for process of
    - $[G_1 - G_0]_i[D_0^i]$
    - $[G_0]_i[D_{sum}^i]$
    - $[-G_1 - G_0]_i[D_1^i]$
  - D_{sum}^i are the coefficients of (D_0 + D_1)
  - Receive linear combinations of coefficients
  - Calculate the point-wise products
  - Feed to TACC for accumulation



SCOPE-II: The Second Accelerator (Cont.)

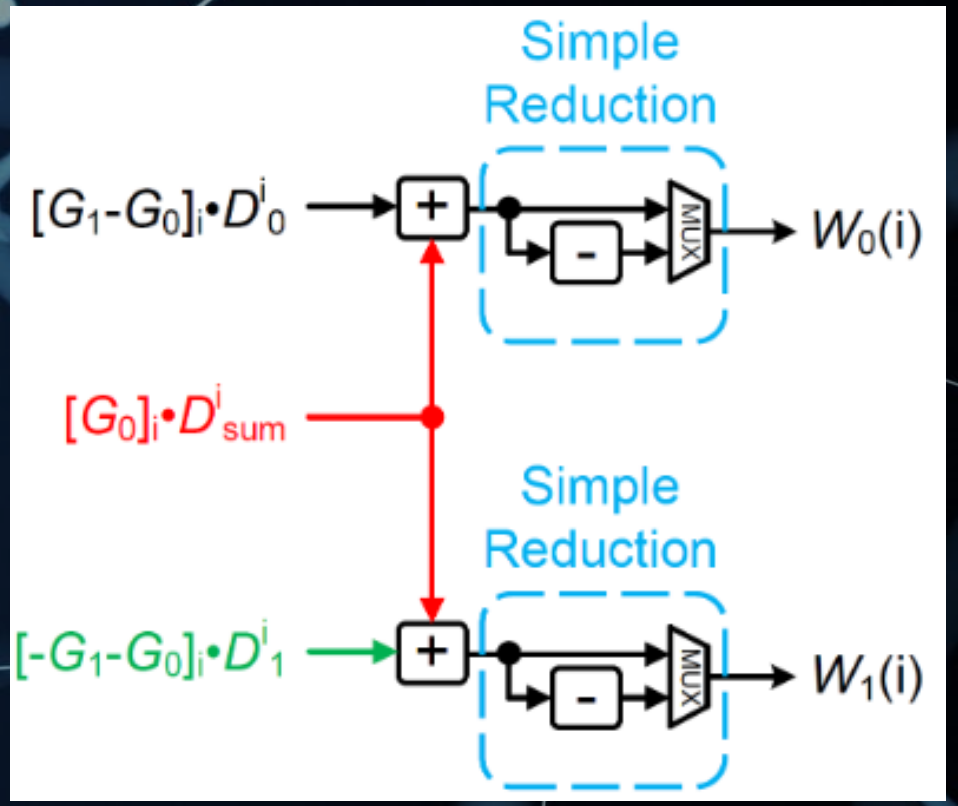
- **TMVP ACcumulation Component (TACC)**
  - $n/2$ cycles
  - **Output**

    $W_{\frac{n}{2}-1}, W_0, \ldots, W_{\frac{n}{2}-2}$



SCOPE-II: The Second Accelerator (Cont.)

- **Linear Combination Component (LCC)**
  - Execute two additions to produce final results
    - $[G_1 - G_0]_i[D_0^i] + [G_0]_i[D_{sum}^i]$
    - $[G_1 - G_0]_i[D_0^i] + [-G_1 - G_0]_i[D_1^i]$
  - Works with the TACC synchronously
  - Two Simple Reductions involved
  - Outputs two coefficients at the same time
  - Outputs scaled to range $[0, q)$



Evaluation: Complexity Analysis

- **SCOPE-I**
  - n registers in BIPC
  - n registers, 2-to-1 MUXes are used in BACC
  - $3n$ 8-to-1, n 16-to-1, and $3n$ 2-to-1 MUXes in BPMC
  - $(7n + 1)$ adders in BPMC and BACC, 200 adders in BSRC
  - $(n + 4)$ cycles
- **SCOPE-II**
  - $3n/2$ registers,2-to-1 MUXes, and Sign Inverters in TIPC
  - $3n/2$ registers and 2-to-1 MUXes in TACC
  - $3n/2$ registers and 2-to-1 MUXes in TSRC
  - $(n/2 + 4)$ cycles

Evaluation: FPGA-based Implementation

- **Falcon**
  - n = 512, q = 12289
  - Artix-7 (XC7a200t) and Ultrascale+ (XCZU9EG-FFVB1156-2)
- **NTRU**
  - n = 701, q = 2^{13}; n = 821, q = 2^{12}
  - Zynq Ultrascale+(XCZU9EG-FFVB1156-2) and Zynq-7000 (xc7z100ffg1156-2)
- Other schoolbook or similar designs
  - n = 256, q = 12289; n = 256, q = 2^{12}
  - Kintex-7 (xc7k480tffv1156-3), Virtex Ultrasclae+ (xcvu9p-flga2577-3-e), Zynq Ultrascale+ (XCZU9EG-FFVB1156-2), Virtex-7 (xc7v2000tflg1925-2L), and Zynq-7000 (xc7z100ffg1156-2)

Evaluation: Comparison

- **Comparison With The Existing Works (FALCON)**

Design	n	Method	LUT	FF	Slice	DSP	BRAM	Fmax[1]	Latency[2]	Delay[3]	ELUT[4]	EADP[5]	EADPR[6]
Zynq Ultrascale+													
[27]	512	NTT	14,327	7,314	NA	4	2	314	2,100	6.7	16,895	112,992	NA
SCOPE-I	512	SB	88,267	35,159	14,598	0	0	525	516	1.0	88,231	86,718	30.30%
SCOPE-II	512	TMVP	157,686	84,226	26,937	0	0	529	260	0.5	157,686	77,502	31.41%
Artix-7													
[27]	512	NTT	14,500	7,287	NA	4	2	142	2,100	14.8	16,371	242,103	NA
SCOPE-I	512	SB	97,322	35,159	15,163	0	0	254	516	2.0	97,322	197,709	18.34%
Kintex Ultrascale+													
[30]*	512	NTT	22,648	15,030	NA	16	24	200	782	3.9	34,456	134,723	NA
SCOPE-I	512	SB	88,185	35,237	14,734	0	0	507	516	1.0	88,185	89,750	33.38%
SCOPE-II	512	TMVP	154,688	87,439	24,503	0	0	410	260	0.6	154,688	98,095	27.19%

Note: Due to the relatively large resource usage of the proposed second accelerator (TMVP-based), we don't implement it on the Artix-7 device.
SB: schoolbook.
*: The performance listed is an estimation since no specific data for $n = 512$ is provided in this work.
[1]: Fmax: Maximum frequency. Unit: MHz
[2]: Latency: Calculation latency (number of cycles). We roughly estimated the NTT-based polynomial multiplication in [27] as 2,100 for $n = 512$.
[3]: Delay = Latency/Fmax. unit: μs.
[4]: ELUT: Equivalent LUT, following [22]. 1 DSP = 102.4 Slices (7 series)/51.2 Slices (UltraScale+); one 18K BRAM = 116.2 Slices (7 series)/58.1 Slices (UltraScale+). UltraScale+ has 8 LUTs in one Slice/CLB while 7 series contains 4 LUTs in one Slice/CLB.
[5]: EADP: Equivalent ADP. EADP = #ELUT×delay (since the Slice number is not available for all designs, we use LUT as the main resource usage metric).
[6]: EADPR: EADP reduction (based on the same FPGA device with the same n).

PGA-based Implementation

- **Comparison With The Existing Works (FALCON)**
  - SCOPE-I and SCOPE-II exhibit 30.30% and 31.41% lower EADP than [27], respectively, on Zynq Ultrascale+
  - SCOPE-I maintains an 18.34% lower EADP than [27] on Artix-7
  - 33.8% and 27.19% more efficient in EADP compared to [30]
  - SCOPE-I is 4.15x faster in latency cycles compared to [27]
  - SCOPE-I has 1.67x higher frequency than [27]

[27] L. Beckwith, D. T. Nguyen, and K. Gaj, “High-performance hardware implementation of lattice-based digital signatures.” Cryptology ePrint Archive, Paper 2022/217, 2022. https://eprint.iacr.org/2022/217.
[30] B. Li, Y. Yan, Y. Wei, and H. Han, “Scalable and parallel optimization of the number theoretic transform based on FPGA,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023.

Evaluation: Comparison (Cont.)

- **Comparison With The Existing Works (NTRU)**

Design	n	q	Method	LUT	FF	Slice	DSP	BRAM	Fmax[1]	Latency[2]	Delay[3]	ELUT[4]	EADP[5]	EADPR[6]
Zynq Ultrascale+														
[10]	701	2^{13}	SB	71,028	18,994	11,661	0	0	223	701	3.14	71,028	223,276	NA
SCOPE-I	701	2^{13}	SB	87,190	41,843	15,069	0	0	577	705	1.22	87,190	106,532	52.29%
SCOPE-II	701	2^{13}	TMVP	150,266	59,677	26,920	0	0	549	354	0.64	150,266	96,893	56.60%
[10]	821	2^{12}	SB	72,430	21,172	11,300	0	0	236	821	3.48	72,430	251,970	NA
SCOPE-I	821	2^{12}	SB	74,760	44,360	12,268	0	0	556	825	1.48	74,760	110,930	55.98%
SCOPE-II	821	2^{12}	TMVP	122,677	64,132	22,863	0	0	513	414	0.81	122,677	99,002	60.71%
Zynq-7000+														
[9]	701	2^{13}	SB	1,463	NA	NA	0	86	76	247,104	3,251.37	21,449	69739901.81	NA
[10]	701	2^{13}	SB	71,321	19,554	20,270	0	0	201	701	3.49	71,321	248,736	NA
SCOPE-I	701	2^{13}	SB	87,191	41,845	25,339	0	0	452	705	1.56	87,191	135,995	45.33%
SCOPE-II	701	2^{13}	TMVP	153,170	58,980	45,710	0	0	416	354	0.85	153,170	130,342	47.60%
[9]	821	2^{12}	SB	1,463	NA	NA	0	86	76	338,664	4,456.11	21,449	95580784.23	NA
[10]	821	2^{12}	SB	71,990	21,202	11,647	0	0	210	821	3.91	71,990	281,447	NA
[28]	821	2^{12}	SB	56,218	21,406	NA	0	0	70	821	11.73	56,218	659,357	NA
SCOPE-I	821	2^{12}	SB	74,773	44,360	22,272	0	0	438	825	1.88	74,773	140,840	49.96%
SCOPE-II	821	2^{12}	TMVP	126,409	63,325	36,336	0	0	436	414	0.95	126,409	120,031	57.35%

Evaluation: Comparison (Cont.)

- **Comparison With The Existing Works (NTRU)**
- On Zynq Ultrascale+ Device:
- SCOPE-I has 52.29% and 55.98% less EADP than [10] for n = 701 and n = 821
- SCPOE-II has 56.6% and 60.71% less EADP for respective n
- On Zynq-7000 Device:
- For n = 701, SCOPE-I and SCOPE-II at least 45.33% and 60.71% less EADP
- For n = 821, SCOPE-I and SCOPE-II at least 49.96% and 57.35% less EADP

•[9] P. Choi and D. K. Kim, “Lightweight polynomial multiplication accelerator for NTRU using shared SRAM,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 12, pp. 4574–4578, 2023.
•[10] P. He et. al, “HPMA-NTRU: High-performance polynomial multiplication accelerator for ntru,” in IEEE Int. Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1–6, 2022.
•[28] Z. Qin, R. Tong, X. Wu, G. Bai, L. Wu, and L. Su, “A compact full hardware implementation of PQC algorithm NTRU,” in 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), pp. 792–797, 2021.

PGA-based Implementation

- **Comparison With The Existing Schoolbook or Similar Designs**

Design	n	q	Method	LUT	FF	Slice	DSP	BRAM	Fmax[1]	Latency[2]	Delay[3]	ELUT[4]	EADP[5]	EADPR[6]
Kintex-7														
[11]	256	7,681	SB	20,000	18,000	8,000	128	0	260	258	1.0	72,429	71,872	NA
SCOPE-I	256	12,289	SB	57,339	20,736	16,523	0	0	449	260	0.6	57,339	33,203	48.93%
SCOPE-II	256	12,289	TMVP	115,108	52,016	33683	0	0	345	132	0.4	115,108	44,041	38.72%
Virtex Ultrascale+														
[11]	256	7,681	SB	19,000	18,000	3,300	128	0	298	258	0.9	71,429	61,841	NA
SCOPE-I	256	12,289	SB	54,085	20,748	9,330	0	0	571	260	0.5	54,085	24,627	60.18%
SCOPE-II	256	12,289	TMVP	100,725	52,271	16,564	0	0	528	132	0.3	100,725	25,181	59.28%
Zynq Ultrascale+														
[12]	256	2^{13}	TM4	4,550	NA	NA	44	10	588	726	1.2	27,220	33,609	NA
SCOPE-I	256	2^{13}	SB	30,814	14,873	5,438	0	0	607	260	0.4	30,814	13,199	60.73%
SCOPE-II	256	2^{13}	TMVP	53,698	21418	8,766	0	0	540	132	0.2	53,698	13,126	60.94%
Virtex-7														
[12]	256	2^{13}	TM4	4,330	NA	NA	44	10	476	726	1.5	27,000	41,181	NA
SCOPE-I	256	2^{13}	SB	30,577	14,883	9,266	0	0	435	260	0.6	30,577	18,276	55.62%
SCOPE-II	256	2^{13}	TMVP	53,867	21,509	15,918	0	0	418	132	0.3	53,867	17,011	58.69%
Zynq-7000														
[12]	256	2^{13}	TM4	4,550	NA	NA	44	10	400	726	1.8	27,220	49,405	NA
SCOPE-I	256	2^{13}	SB	30,582	14,874	9,162	0	0	476	269	0.6	30,582	17,283	65.02%
SCOPE-II	256	2^{13}	TMVP	53,877	21,544	15,343	0	0	409	132	0.3	53,877	17,388	64.80%

SB: Schoolbook. TM4: Toom-Cook-4.

Evaluation: Comparison (Cont.)

- **Comparison With The Existing Schoolbook or Similar Designs**
  - **For prime modulo**
    - SCOPE-I demonstrates 48.93% and 60.18% less EADP on Kintex-7 and Virtex Ultrascale+ devices, respectively.
    - SCOPE-II demonstrates 38.72% and 59.28% less EADP on Kintex-7 and Virtex Ultrascale+ devices, respectively.
  - **For power-of-2 modulo**
    - Proposed designs on Zynq Ultrascale+ and Zynq-7000 devices have 60.73% and 64.80% less EADP than [12].

•[11] D.-e.-S. Kundi et. al, “Ultra high-speed polynomial multiplications for lattice-based cryptography on FPGAs,” IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 4, pp. 1993–2005, 2022.

•[12] J. Wang et. al, “A high-throughput Toom-Cook-4 polynomial multiplier for lattice-based cryptography using a novel Winograd-schoolbook algorithm,” IEEE Transactions on Circuits and Systems I: Regular Papers, 2023.

Conclusion & Future Works

- **Conclusion:**
  - The proposed design strategy be seen as an alternative solution to the NTT-based polynomial multiplication for the NTRU-based (or other lattice-based) PQC when n is relatively small.
  - For large n (such as n = 1, 024), however, the implementation will be very large and hence unsuitable for practical applications.

- **Future Works:**
  - New solutions to deploy the proposed strategy
  - Deploying the proposed SCOPE in the actual cryptoprocessor building
  - New polynomial multiplication implementation strategies

References

- [9] P. Choi and D. K. Kim, “Lightweight polynomial multiplication accelerator for NTRU using shared SRAM,” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 70, no. 12, pp. 4574–4578, 2023.
- [10] P. He, Y. Tu, A. Khalid, M. O’Neill, and J. Xie, “HPMA-NTRU: High-performance polynomial multiplication accelerator for ntru,” in 2022
  IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), pp. 1–6, 2022.
- [11] D.-e.-S. Kundi, Y. Zhang, C. Wang, A. Khalid, M. O’Neill, and W. Liu, “Ultra high-speed polynomial multiplications for lattice-based
  cryptography on FPGAs,” IEEE Transactions on Emerging Topics in Computing, vol. 10, no. 4, pp. 1993–2005, 2022.
- [12] J. Wang, C. Yang, F. Zhang, Y. Meng, S. Xiang, and Y. Su, “A high-throughput Toom-Cook-4 polynomial multiplier for lattice-based cryptography using a novel Winograd-schoolbook algorithm,” IEEE Transactions on Circuits and Systems I: Regular Papers, 2023.
- [27] L. Beckwith, D. T. Nguyen, and K. Gaj, “High-performance hardware implementation of lattice-based digital signatures.” Cryptology ePrint Archive, Paper 2022/217, 2022. https://eprint.iacr.org/2022/217.
- [28] Z. Qin, R. Tong, X. Wu, G. Bai, L. Wu, and L. Su, “A compact full hardware implementation of PQC algorithm NTRU,” in 2021
  International Conference on Communications, Information System and Computer Engineering (CISCE), pp. 792–797, 2021.
- [29] P. Longa and M. Naehrig, “Speeding up the number theoretic transform for faster ideal lattice-based cryptography,” in Cryptology and Network Security: 15th Int. Conf., pp. 124–139, 2016.
- [30] B. Li, Y. Yan, Y. Wei, and H. Han, “Scalable and parallel optimization of the number theoretic transform based on FPGA,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2023.


THANK YOU