Area Efficient Polynomial Arithmetic Accelerator for Post-Quantum Digital Signatures and KEMs

Dina KAMEL, François-Xavier STANDAERT

Lattice-based Post-Quantum Crypto

- Lattice-based schemes defined over the ring $\mathcal{R}_q = \mathbb{Z}_q/(x^n + 1)$ rely on algebraic operations on high-order polynomials.
- High-order polynomial multiplication is very expensive.
  - School-book multiplication complexity is $\mathcal{O}(n^2)$.
- Number Theoretic Transform (NTT) simplifies multiplication of polynomials.
  - NTT complexity is $\mathcal{O}(n\, log(n))$.
- To use NTT, schemes must comply with two conditions:
  - n is a power of 2.
  - Support NTT-friendly primes $q \equiv 1 \bmod 2n$ to achieve fully-splitting of the polynomial ring.

Supported Post Quantum Crypto schemes

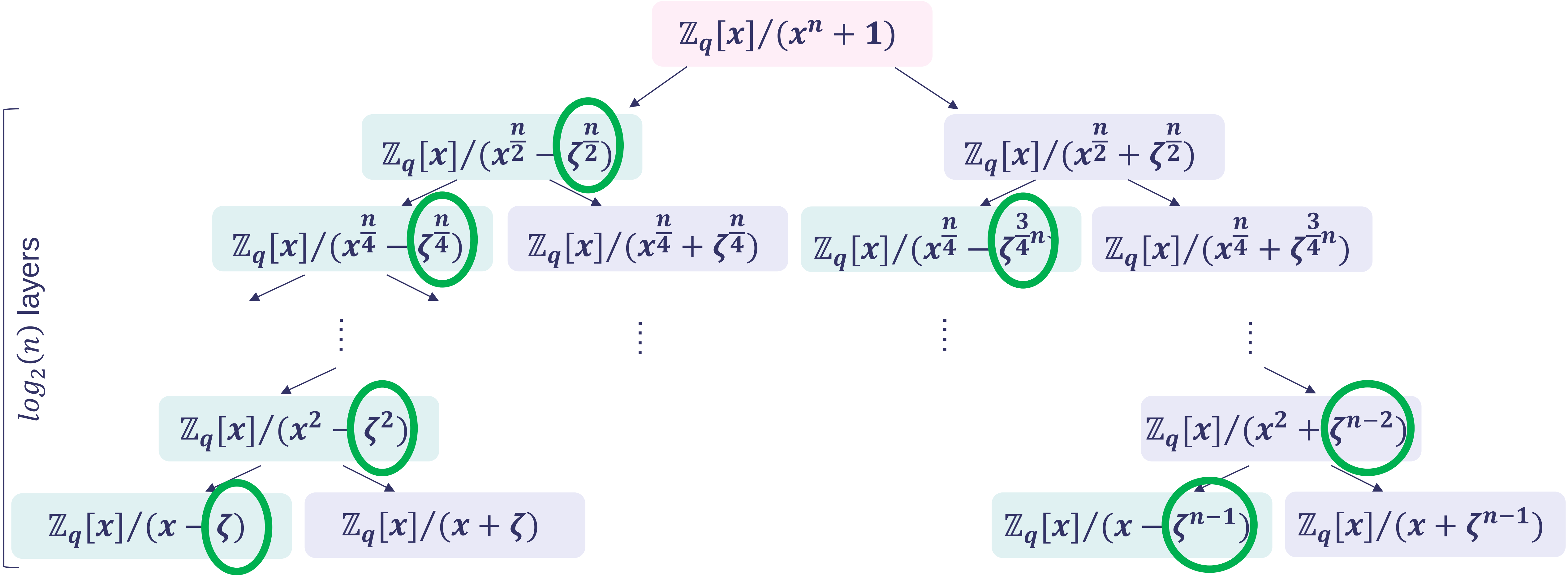
	Purpose	Hardness	λ	n	q	$log_2(q)$
Dilithium	Digital signature	MLWE	2 / 3 / 5	256	8380417	24
Hawk	Digital signature	LIP	1 / 5	512 / 1024	2147473409, 2147389441	62 (31, 31)
Raccoon	Digital signature	MLWE	1 / 3 / 5	512	16515073, 33292289	49 (24, 25)
Kyber	KEM	MLWE	1 / 3 / 5	256	3329	12
Polka	Encryption scheme	LWPR		1024	5939	16

Number Theoretic Transform

If $n = 2^k$ and $q \equiv 1 \bmod 2n$ then the primitive $2n$ -th root of unity ζ exists s.t $\zeta^{2n} \bmod q = 1$

Number Theoretic Transform

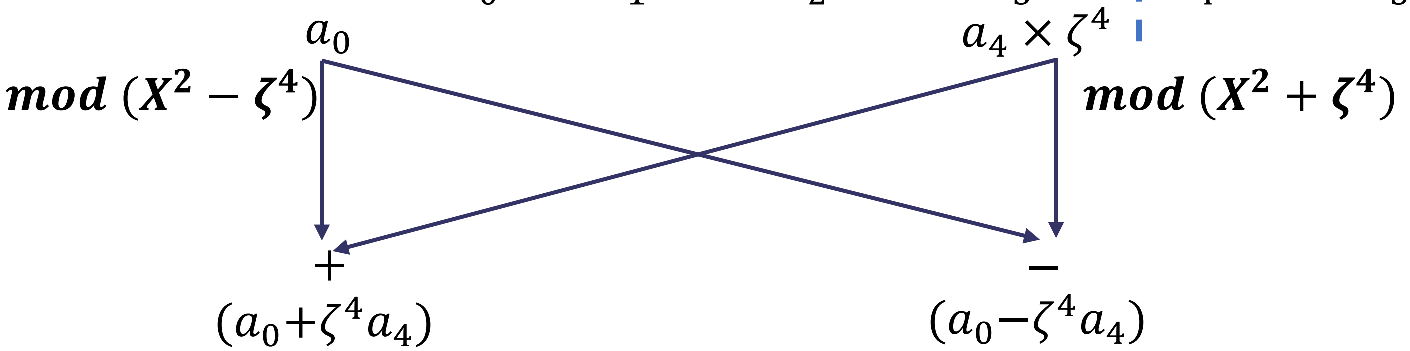
If $n = 2^k$ and $q \equiv 1 \bmod 2n$ then the primitive $2n$ -th root of unity ζ exists s.t $\zeta^{2n} \bmod q = 1$



Source: NTT and its applications in lattice-based cryptosystems: A survey by ZHICHUANG Liang 2022

Number Theoretic Transform

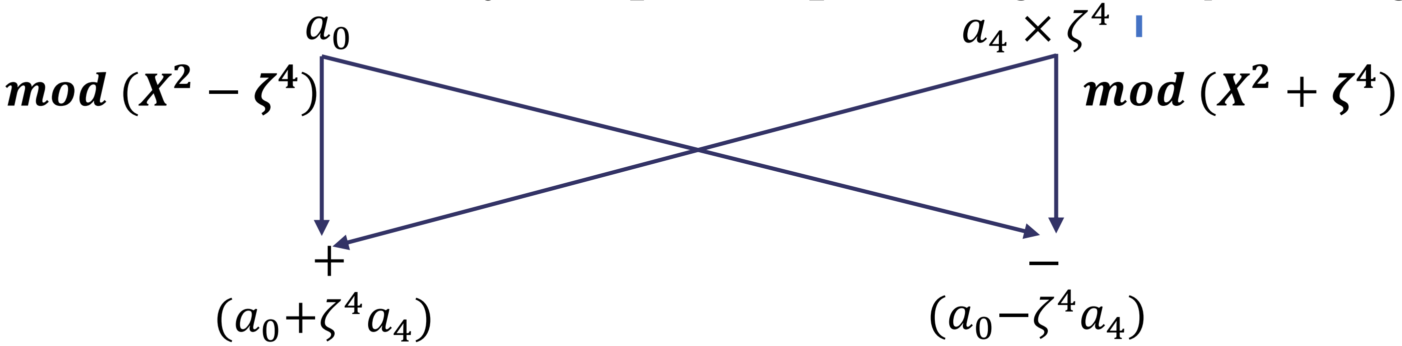
$$a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 \;\; \boldsymbol{mod} \; (\boldsymbol{X^8+1})$$



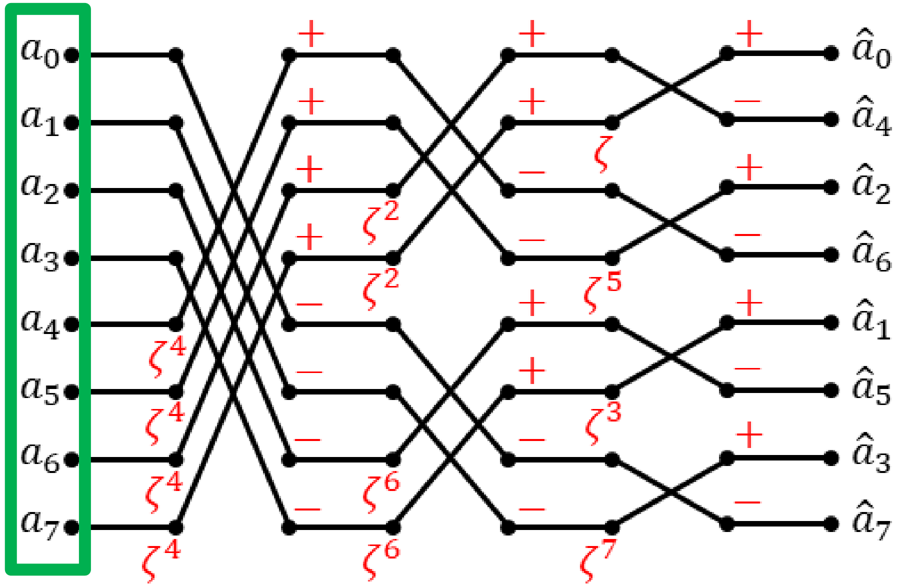
Cooley-Tukey butterfly

Number Theoretic Transform

$$a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 \ \boldsymbol{mod}\ (\boldsymbol{X^8+1})$$



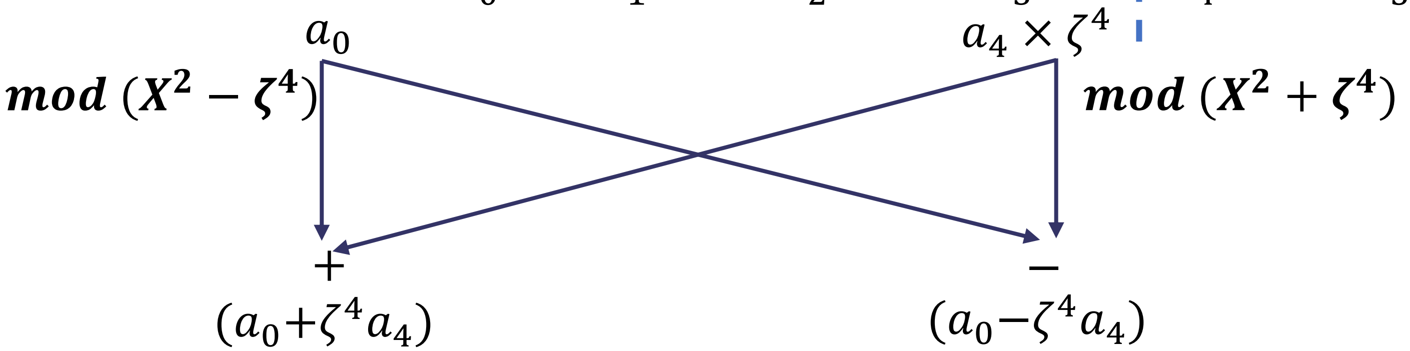
Cooley-Tukey butterfly



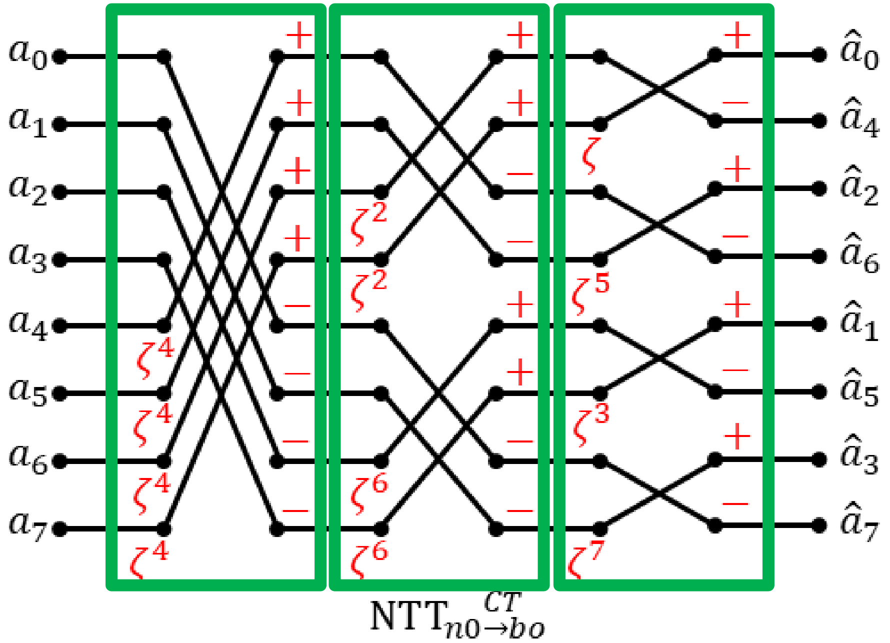
$\mathrm{NTT}^{CT}_{no\to bo}$

Number Theoretic Transform

$$a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 \ \boldsymbol{mod}\ (\boldsymbol{X^8+1})$$

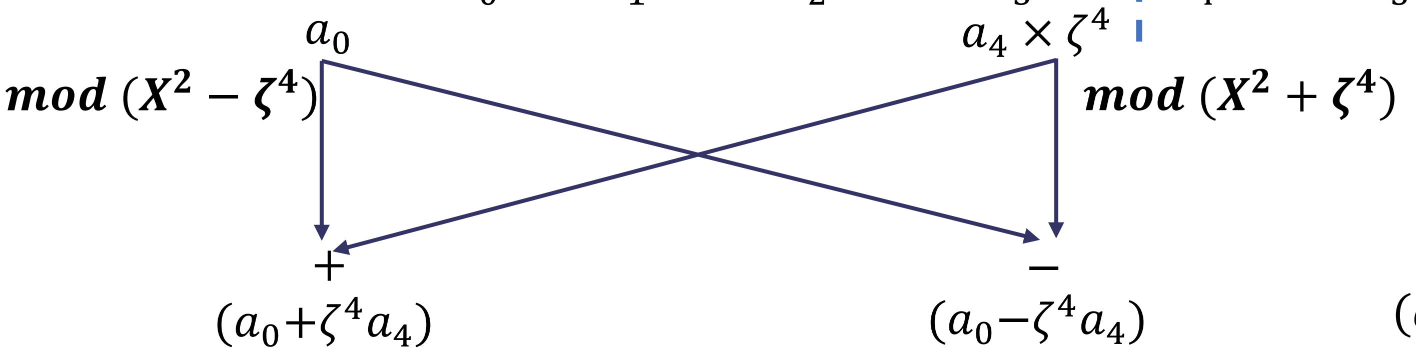


Cooley-Tukey butterfly

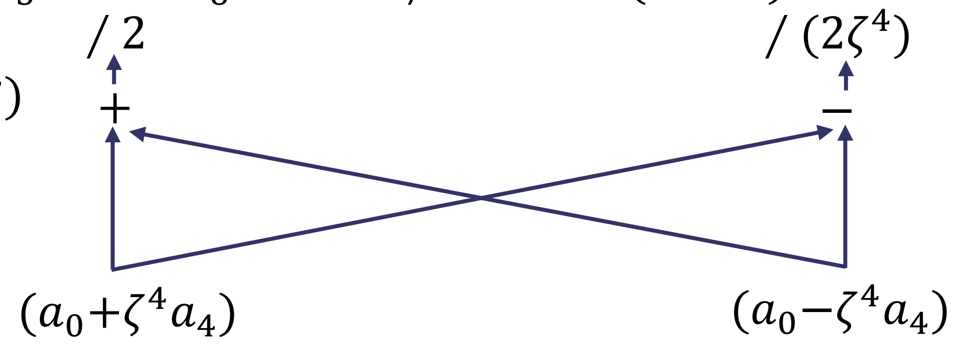


Number Theoretic Transform

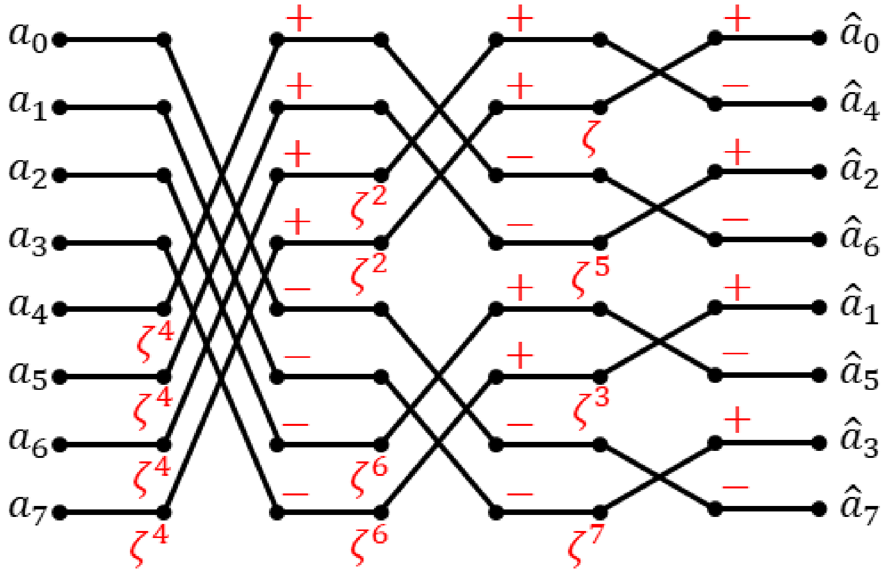
$$a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 \ \boldsymbol{mod}\ (\boldsymbol{X^8+1})$$



Cooley-Tukey butterfly



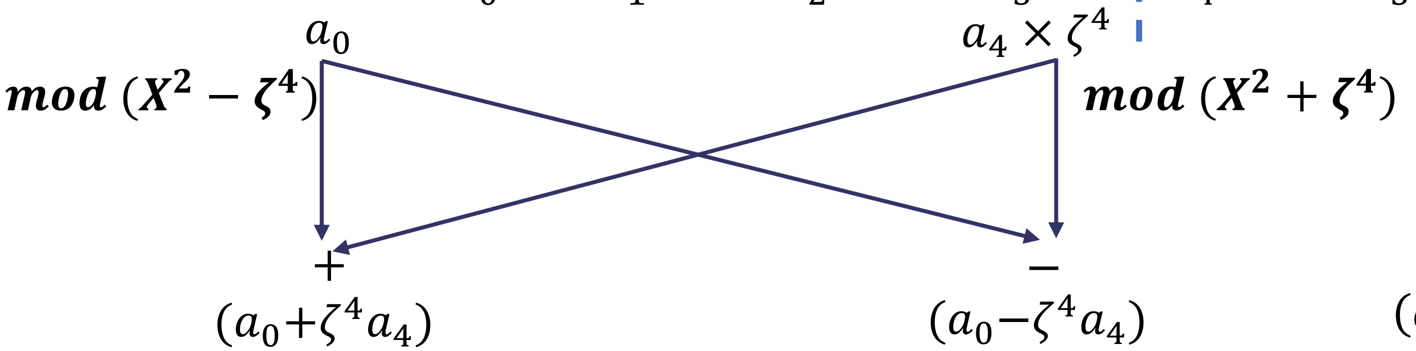
Gentleman-Sande butterfly

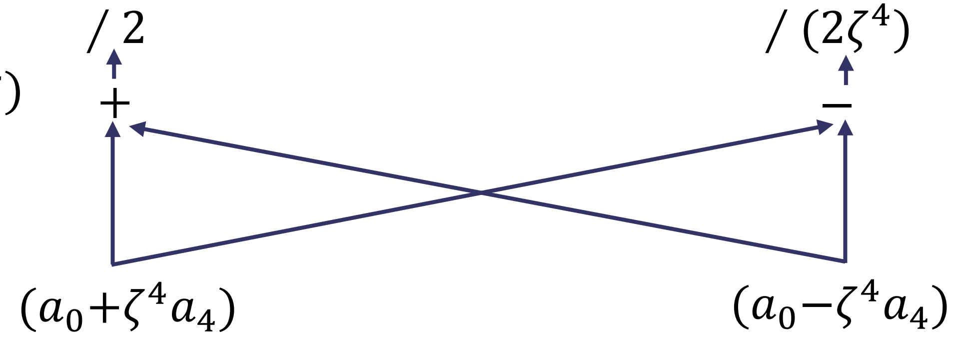


$\mathrm{NTT}_{n0\to bo}^{CT}$

Number Theoretic Transform

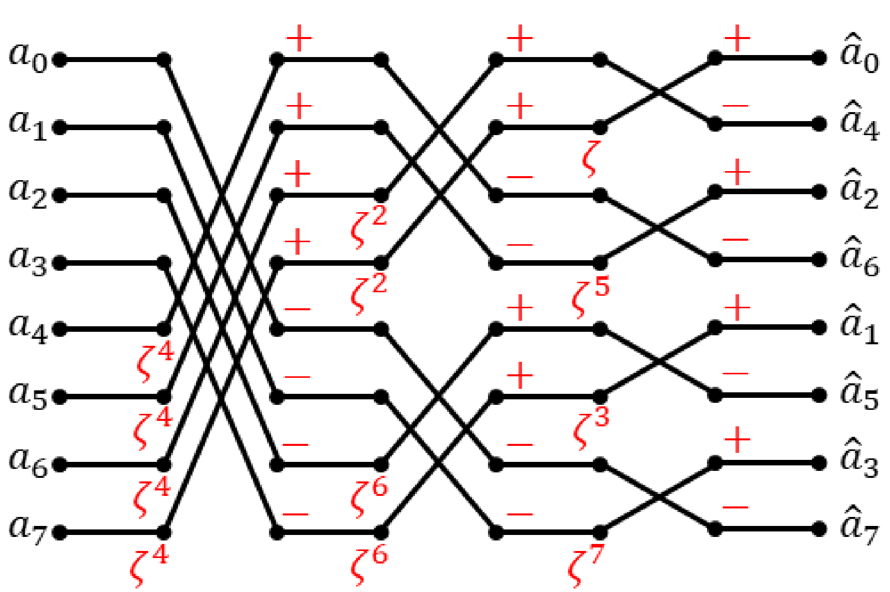
$$a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4 + a_5X^5 + a_6X^6 + a_7X^7 \ \boldsymbol{mod}\ (\boldsymbol{X^8+1})$$



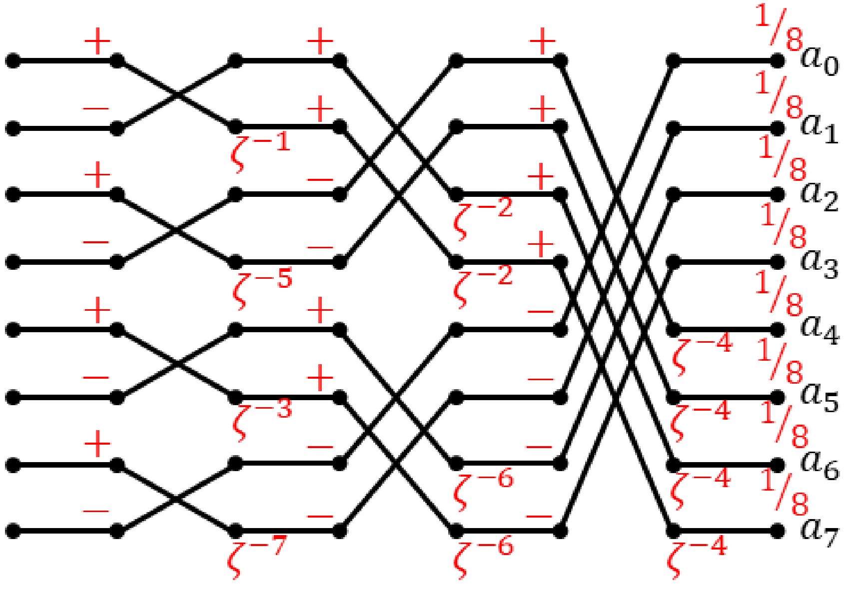


Cooley-Tukey butterfly

Gentleman-Sande butterfly

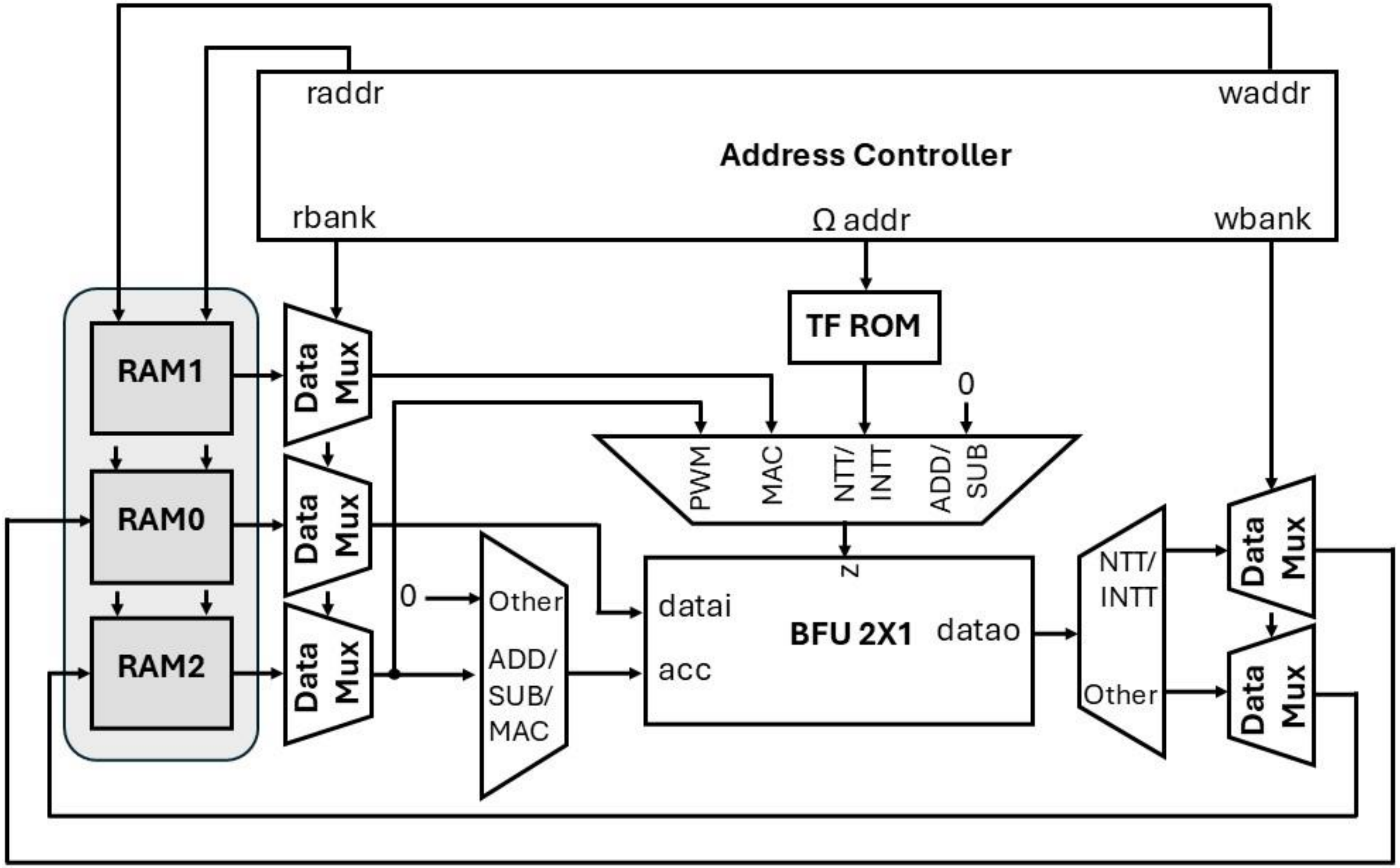


$\mathrm{NTT}_{n0\to bo}^{CT}$

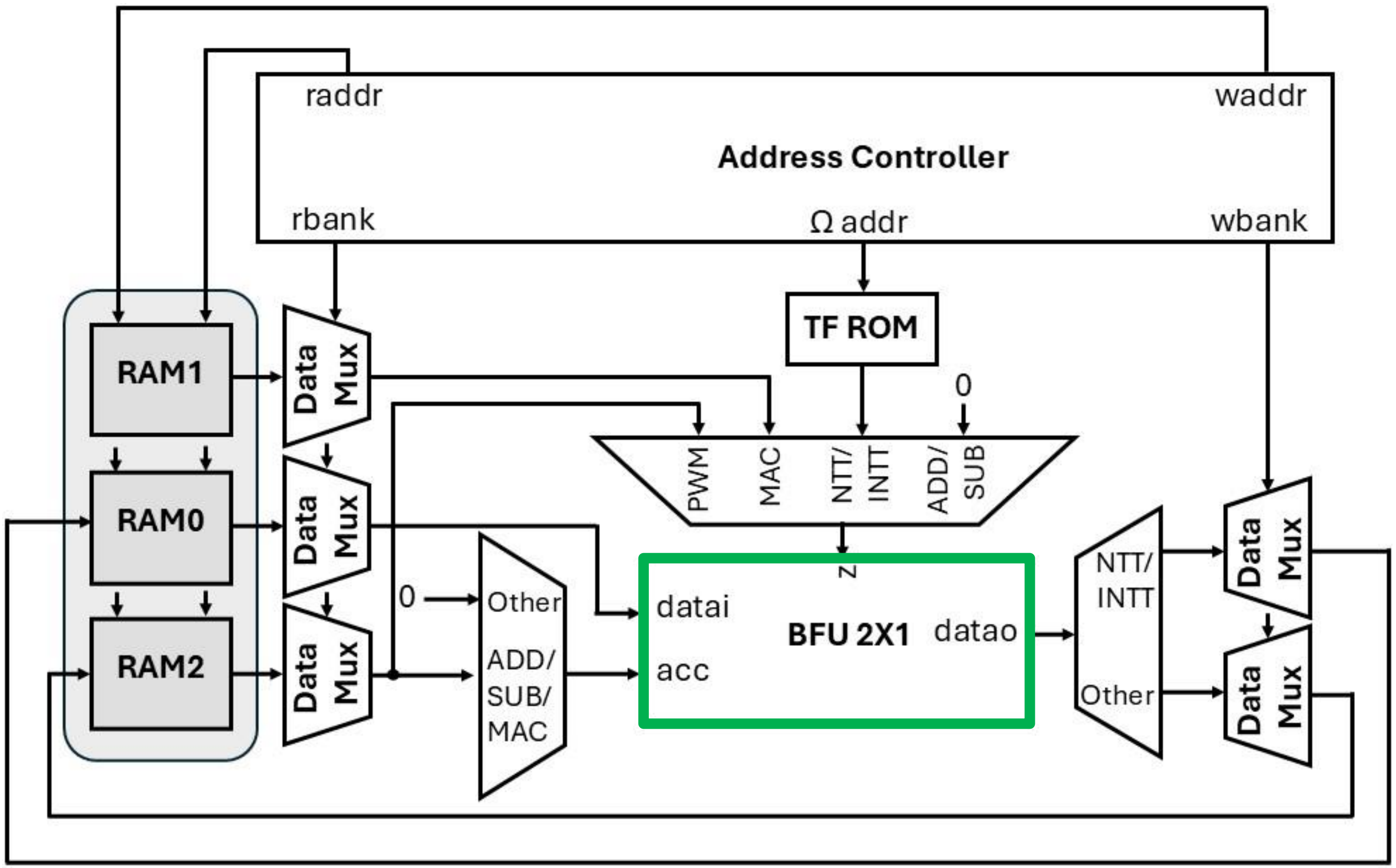


$\mathrm{NTT}_{b0\to no}^{-1\,GS}$

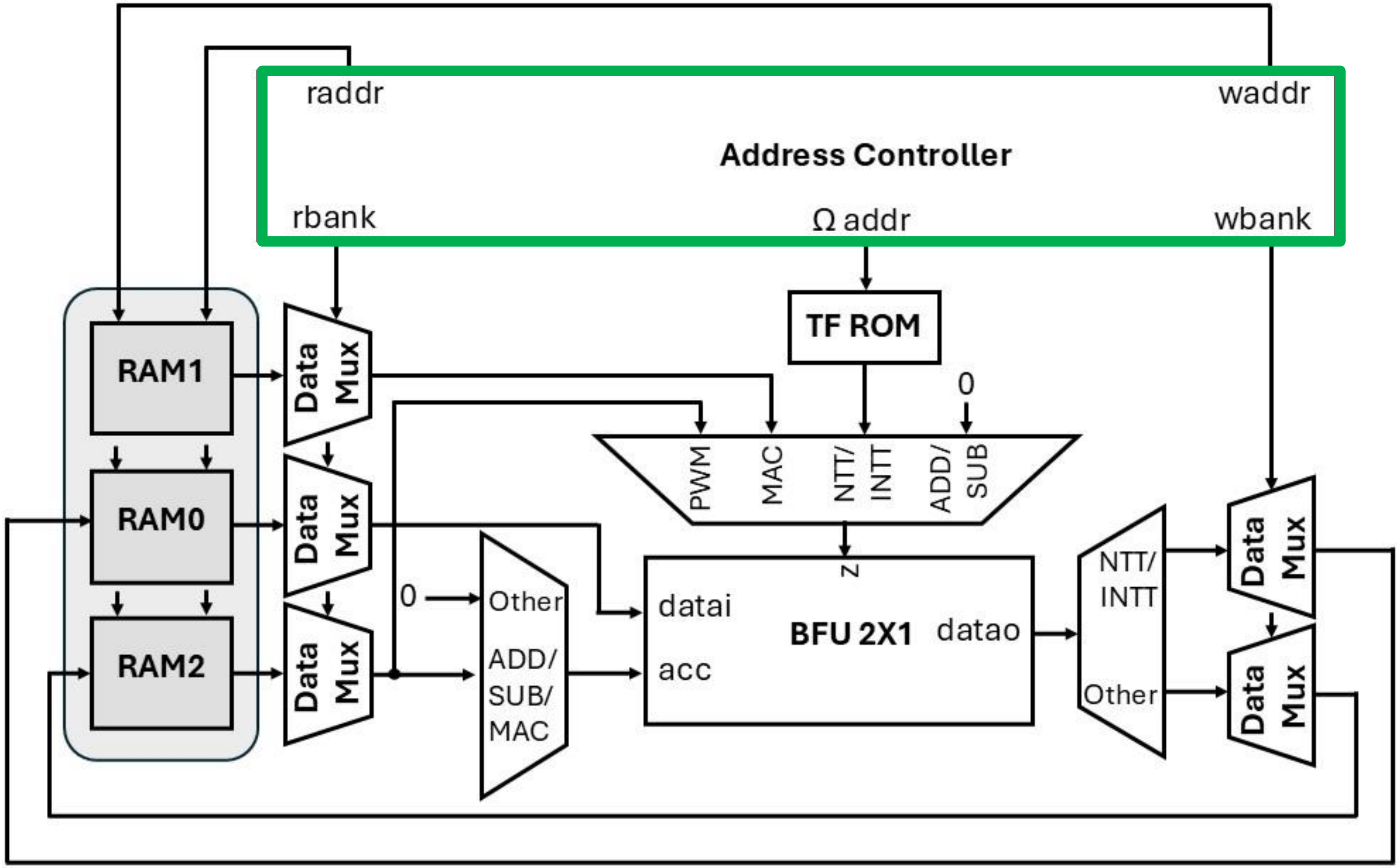
Polynomial Arithmetic Unit Architecture



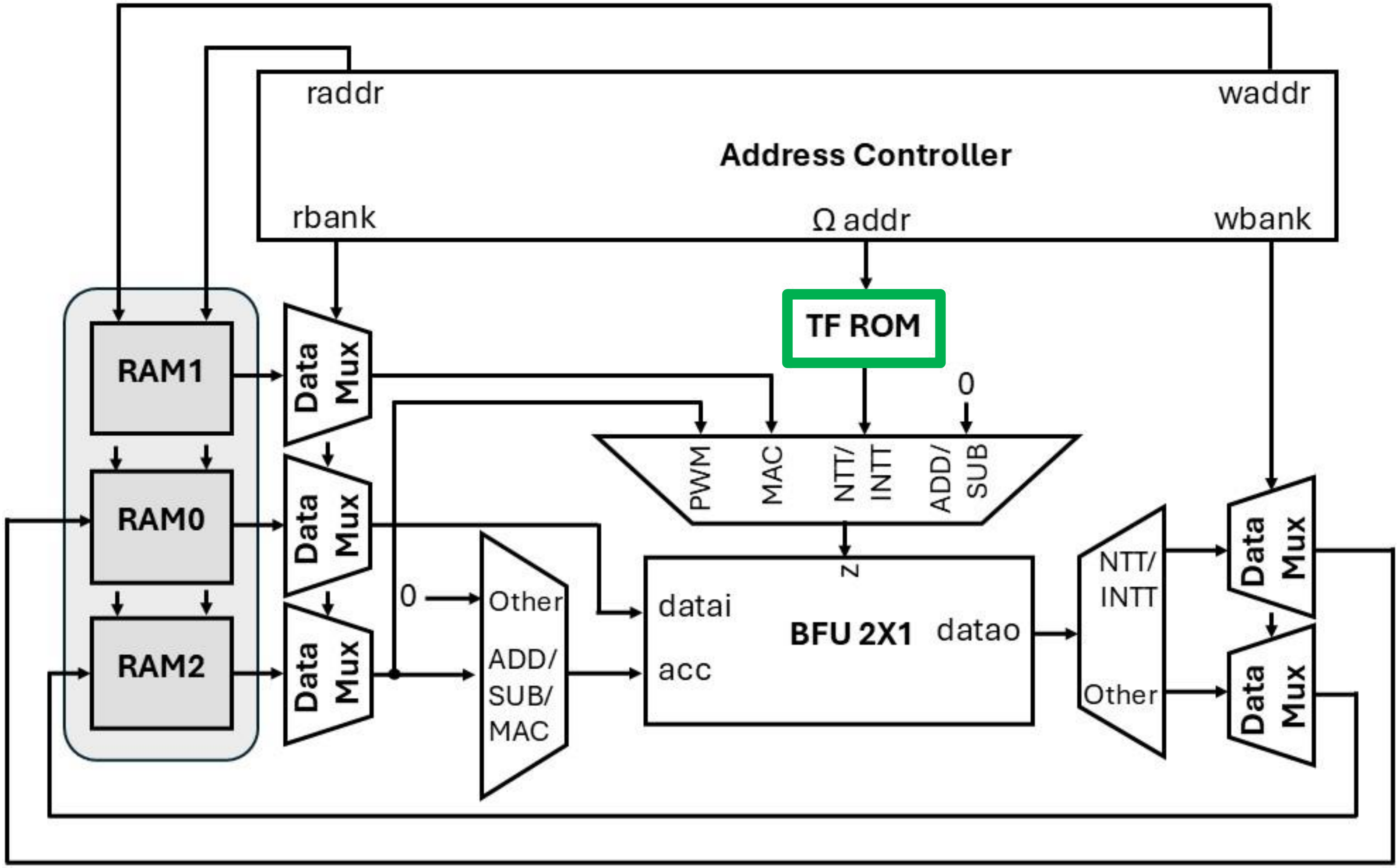
Polynomial Arithmetic Unit Architecture



Polynomial Arithmetic Unit Architecture



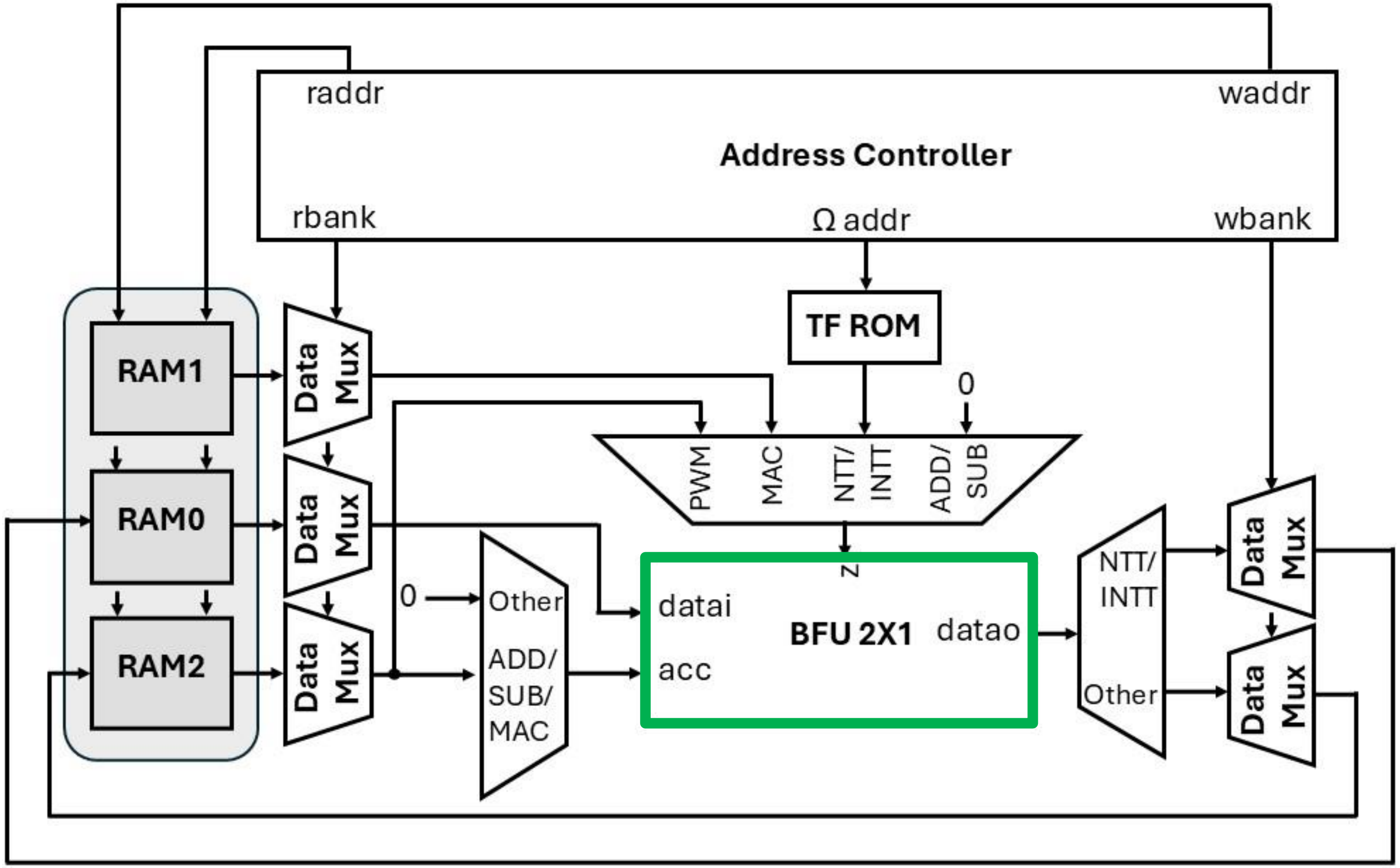
Polynomial Arithmetic Unit Architecture



Polynomial Arithmetic Unit Architecture

Design decisions

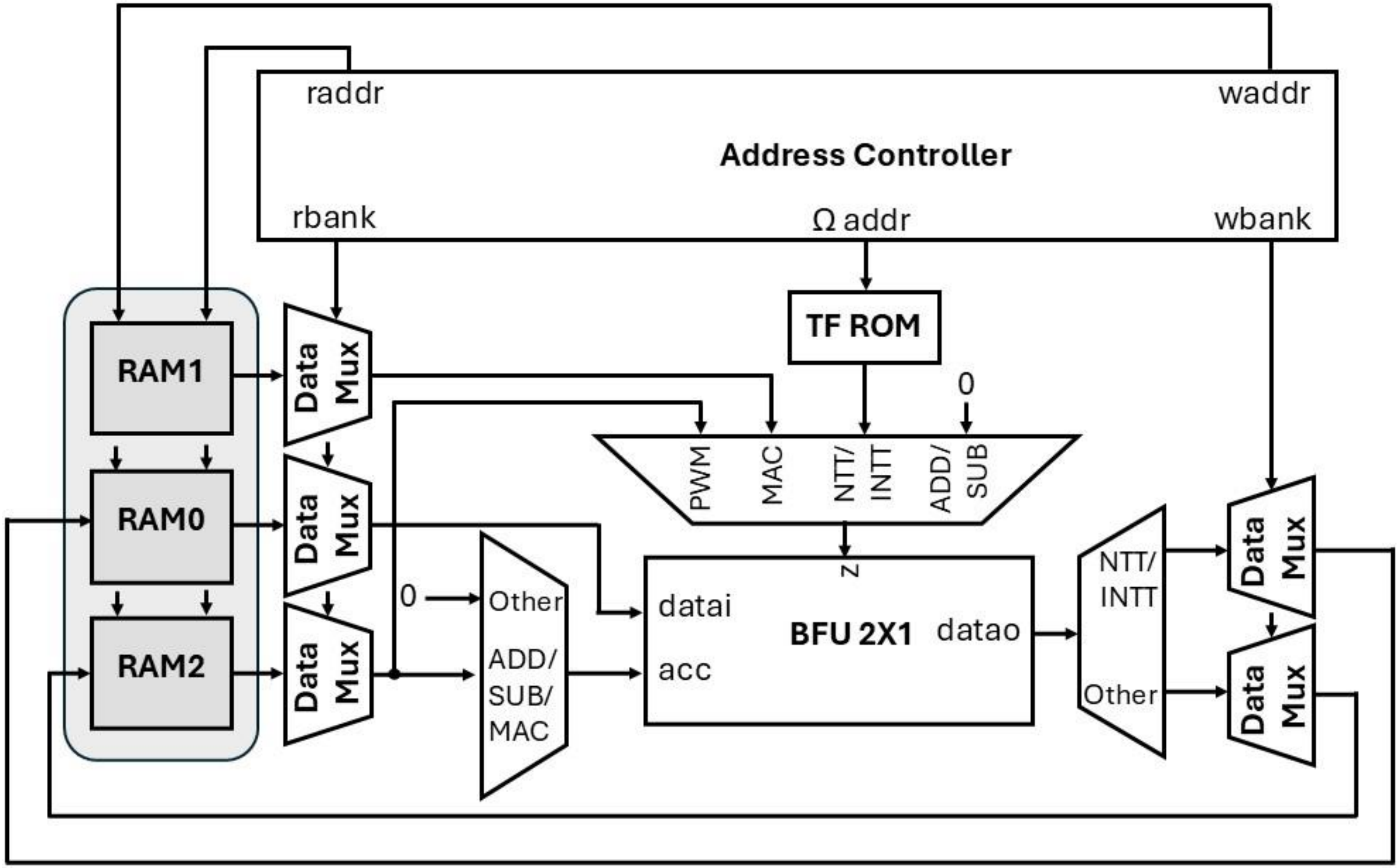
- 2 BFU in parallel



Polynomial Arithmetic Unit Architecture

Design decisions

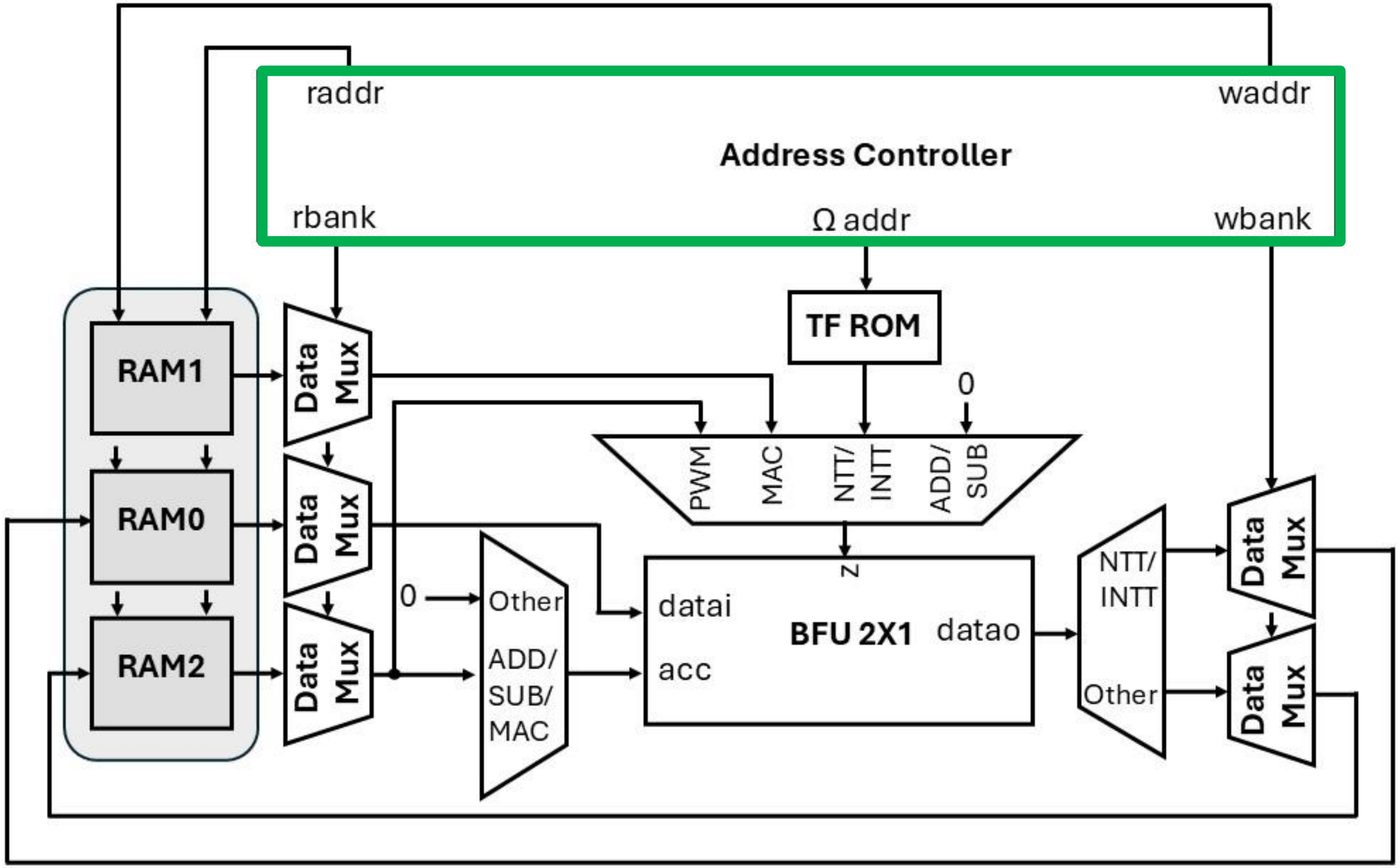
- 2 BFU in parallel
- In-place NTT



Polynomial Arithmetic Unit Architecture

Design decisions

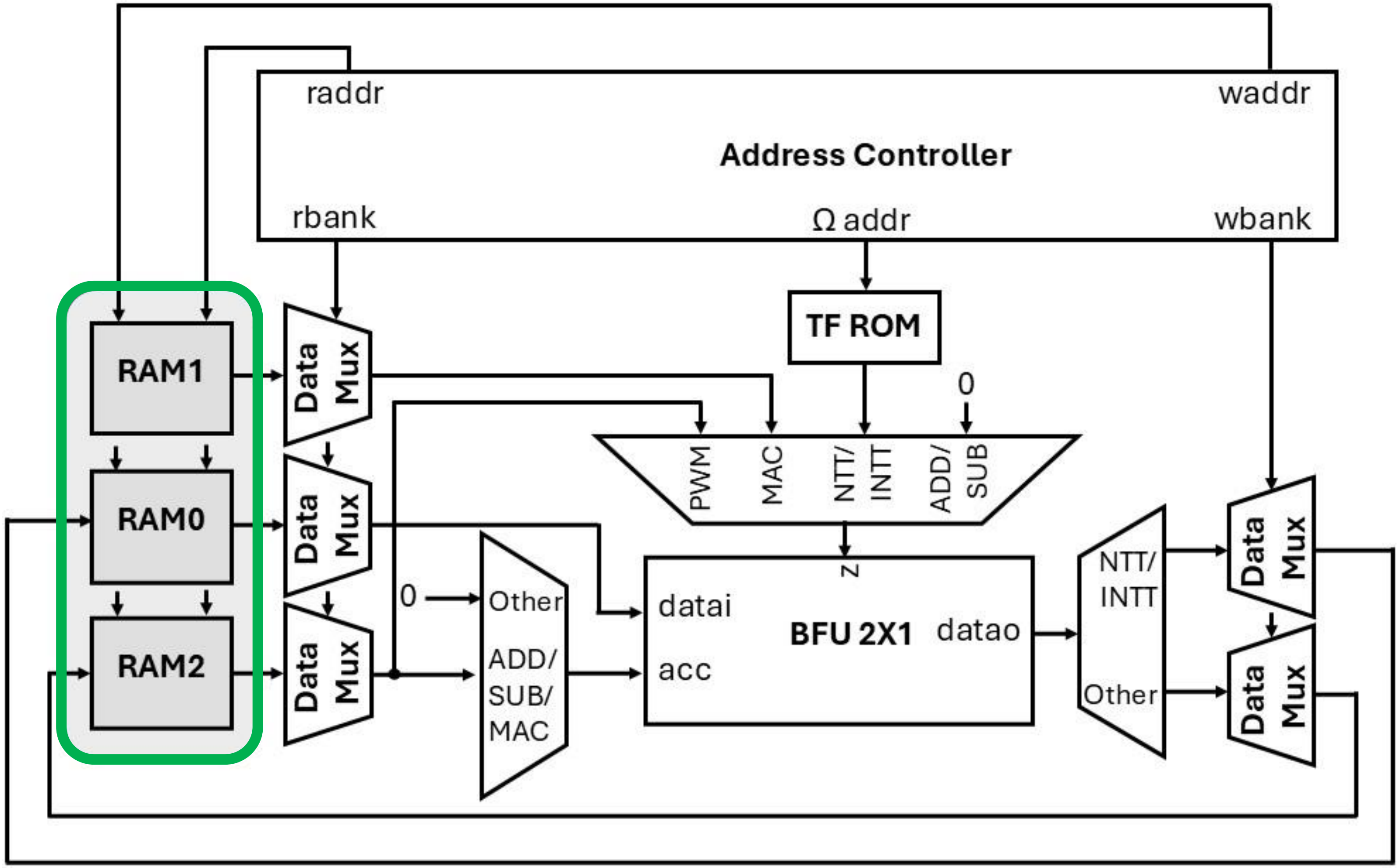
- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access



Polynomial Arithmetic Unit Architecture

Design decisions

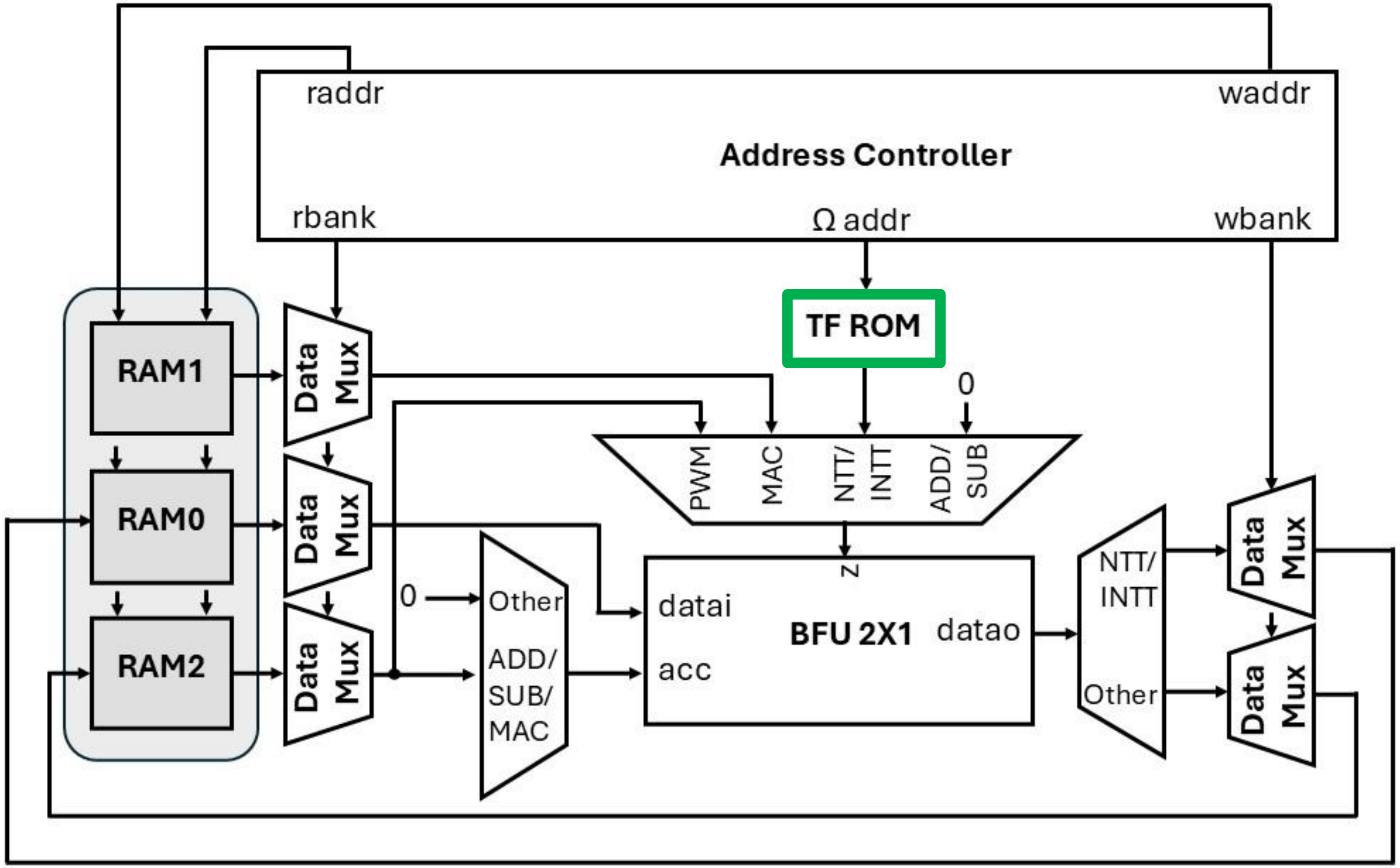
- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access
- Storing a polynomial in 4 BRAMs



Polynomial Arithmetic Unit Architecture

Design decisions

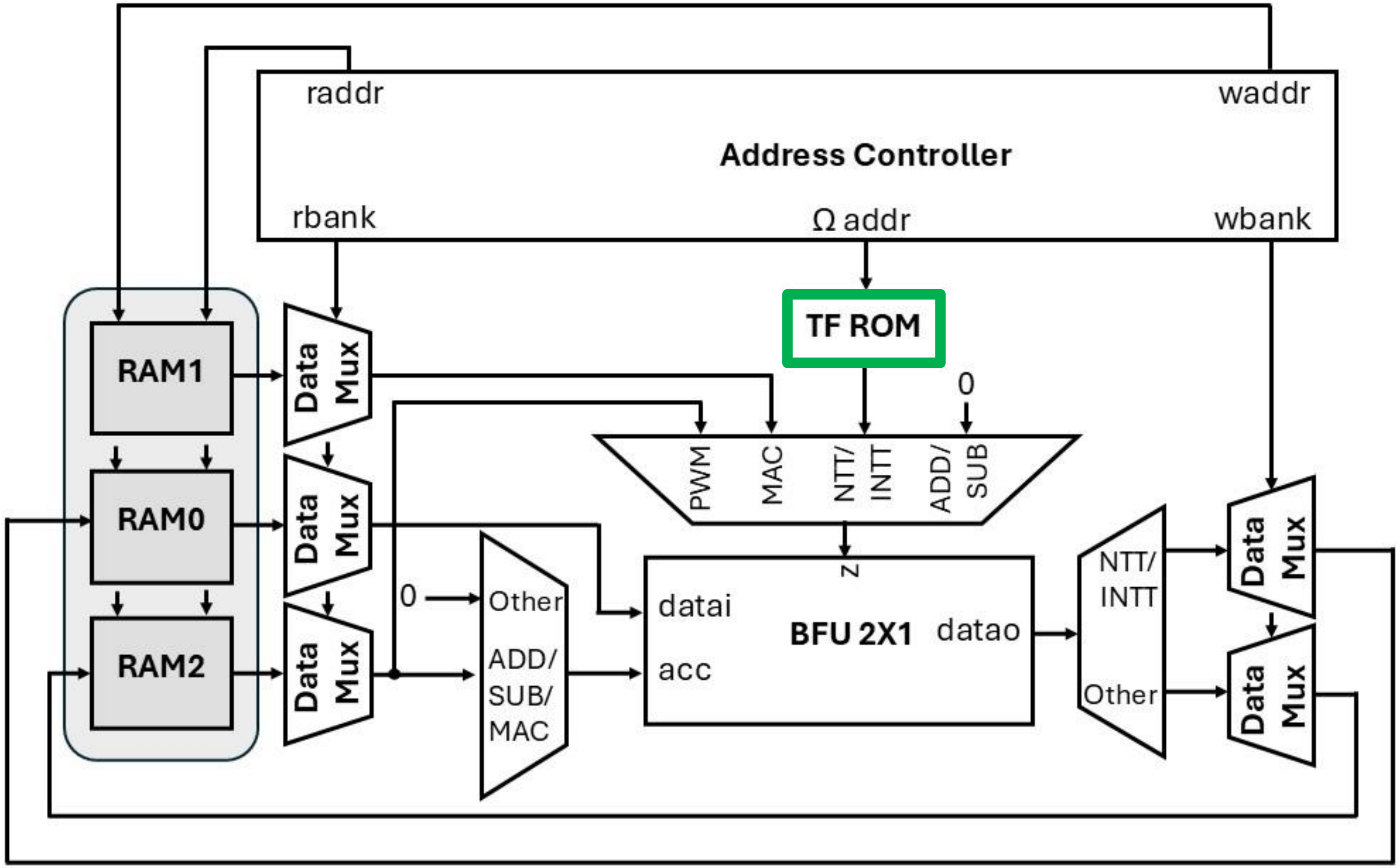
- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access
- Storing a polynomial in 4 BRAMs
- Storing the NTT twiddle factors (TF) in LUTs instead of BRAM



Polynomial Arithmetic Unit Architecture

Design decisions

- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access
- Storing a polynomial in 4 BRAMs
- Storing the NTT twiddle factors (TF) in LUTs instead of BRAM
- Reusing the TFs during INTT



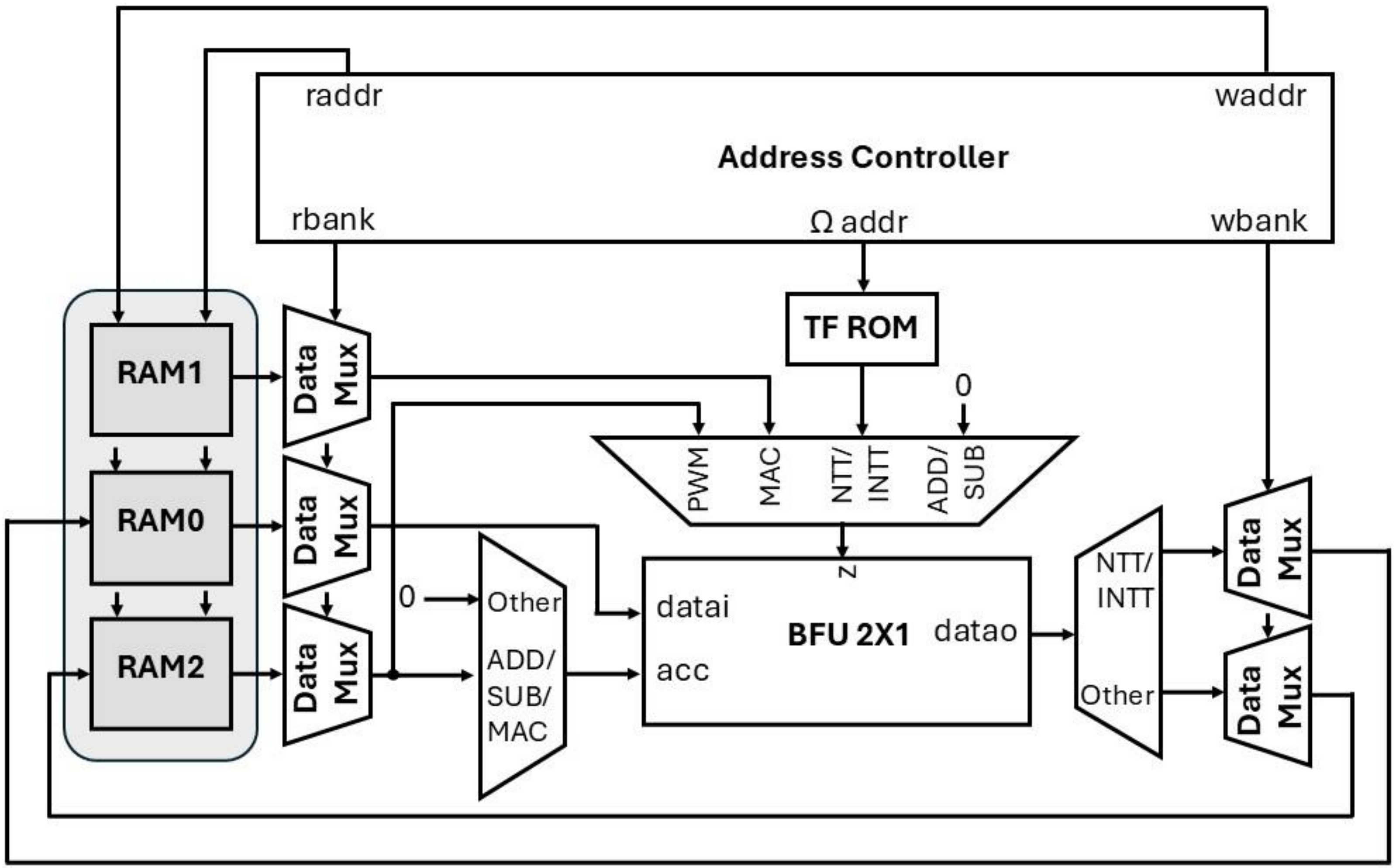
Polynomial Arithmetic Unit Architecture

Design decisions

- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access
- Storing a polynomial in 4 BRAMs
- Storing the NTT twiddle factors (TF) in LUTs instead of BRAM
- Reusing the TFs during INTT

Genericity

- Design-time generic architecture



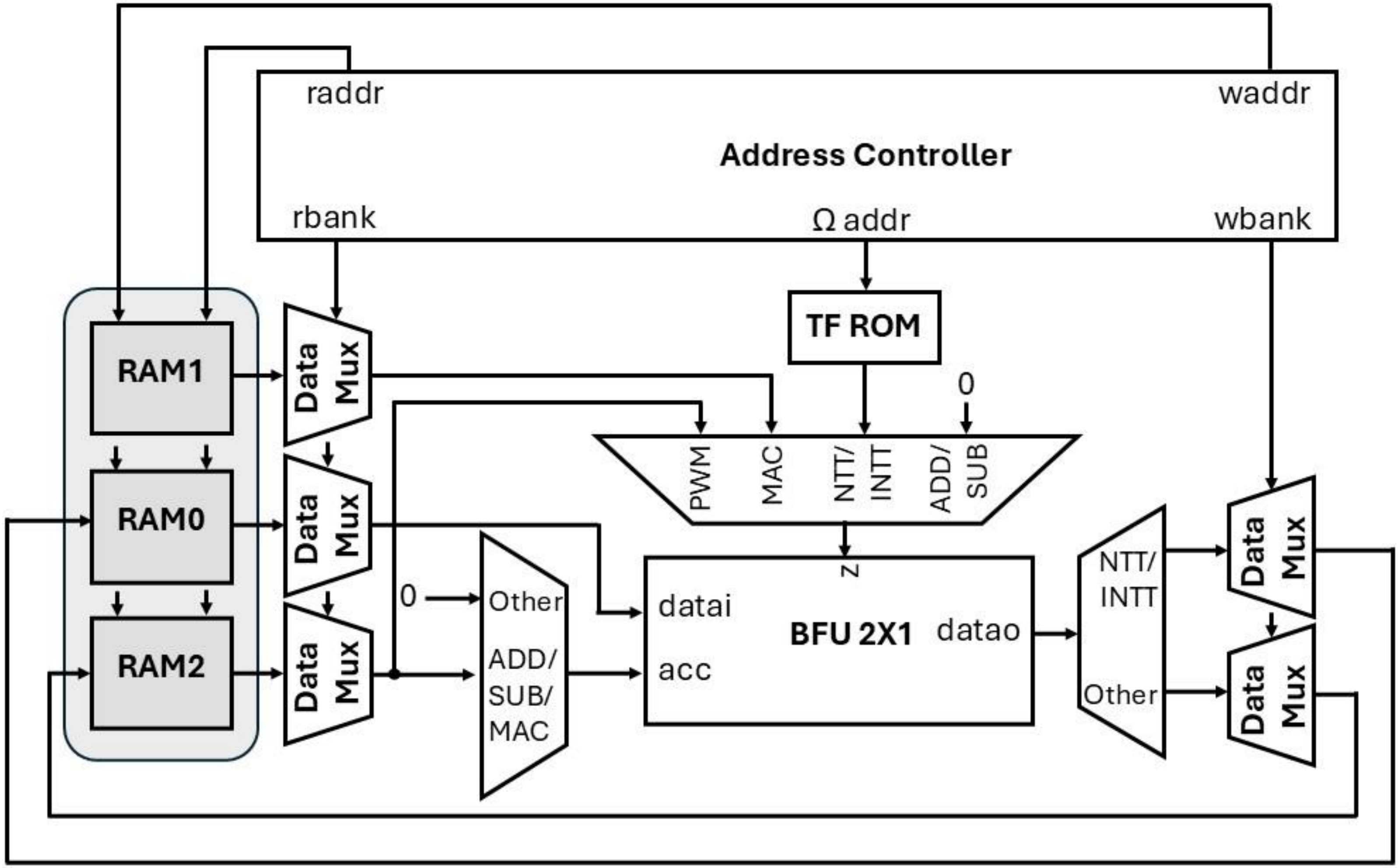
Polynomial Arithmetic Unit Architecture

Design decisions

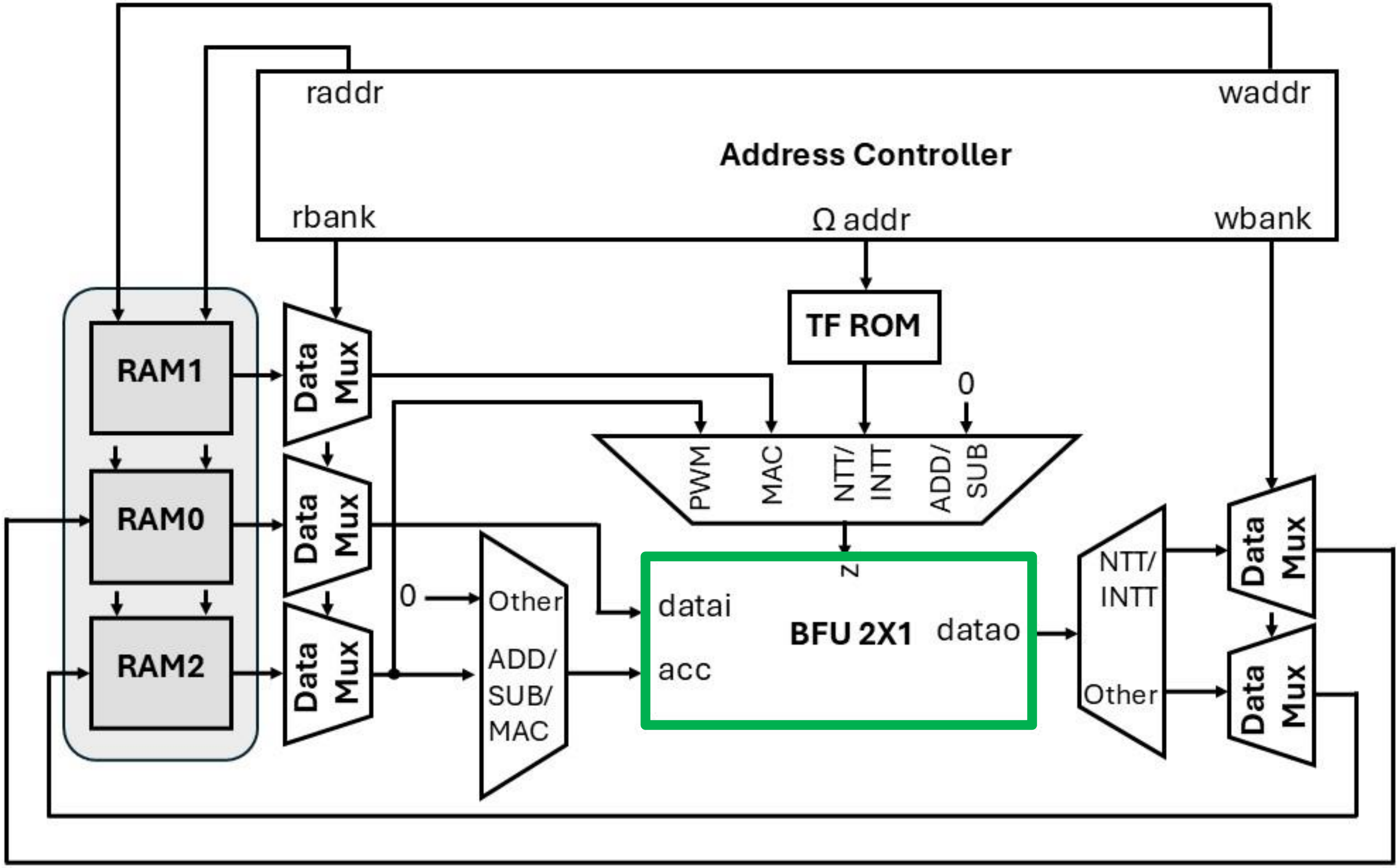
- 2 BFU in parallel
- In-place NTT
- Conflict-free memory access
- Storing a polynomial in 4 BRAMs
- Storing the NTT twiddle factors (TF) in LUTs instead of BRAM
- Reusing the TFs during INTT

Genericity

- Design-time generic architecture
- Semi-generic
  - Modular reduction is fully customized per scheme
  - Address controller adapts to # of NTT layers (even or odd)

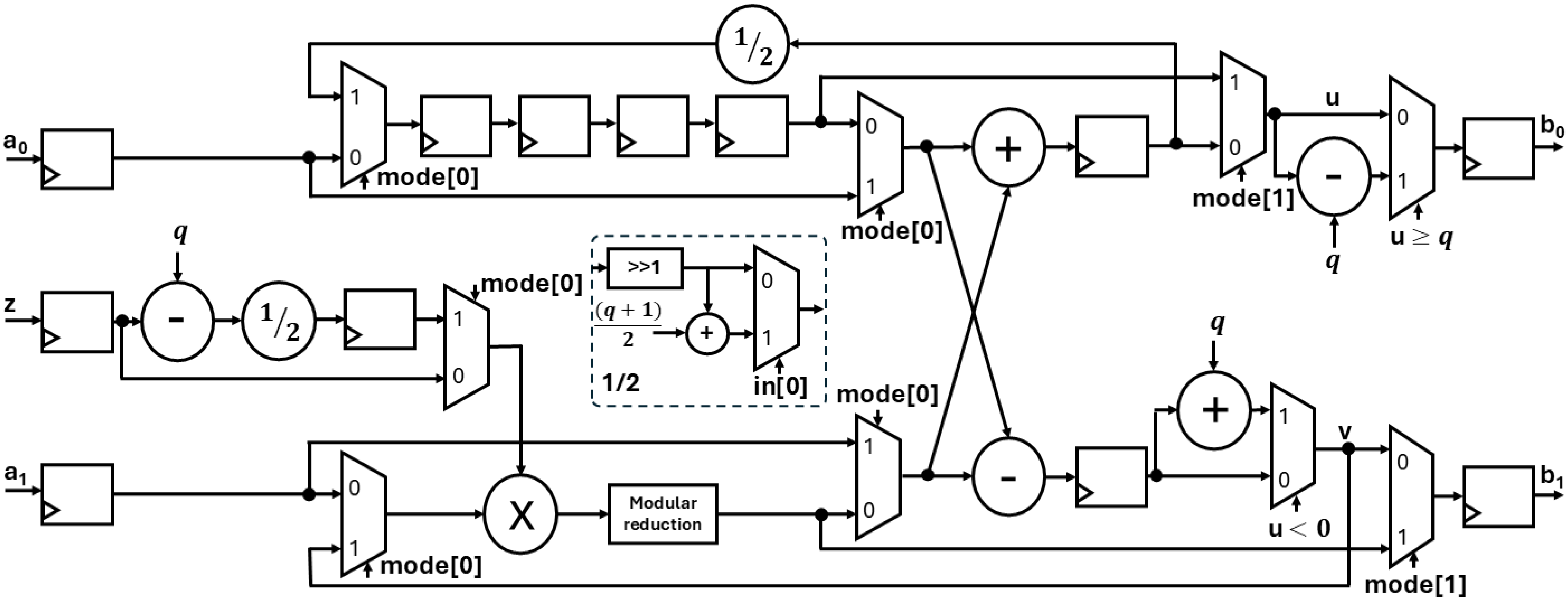


Polynomial Arithmetic Unit Architecture



Butterfly Architecture

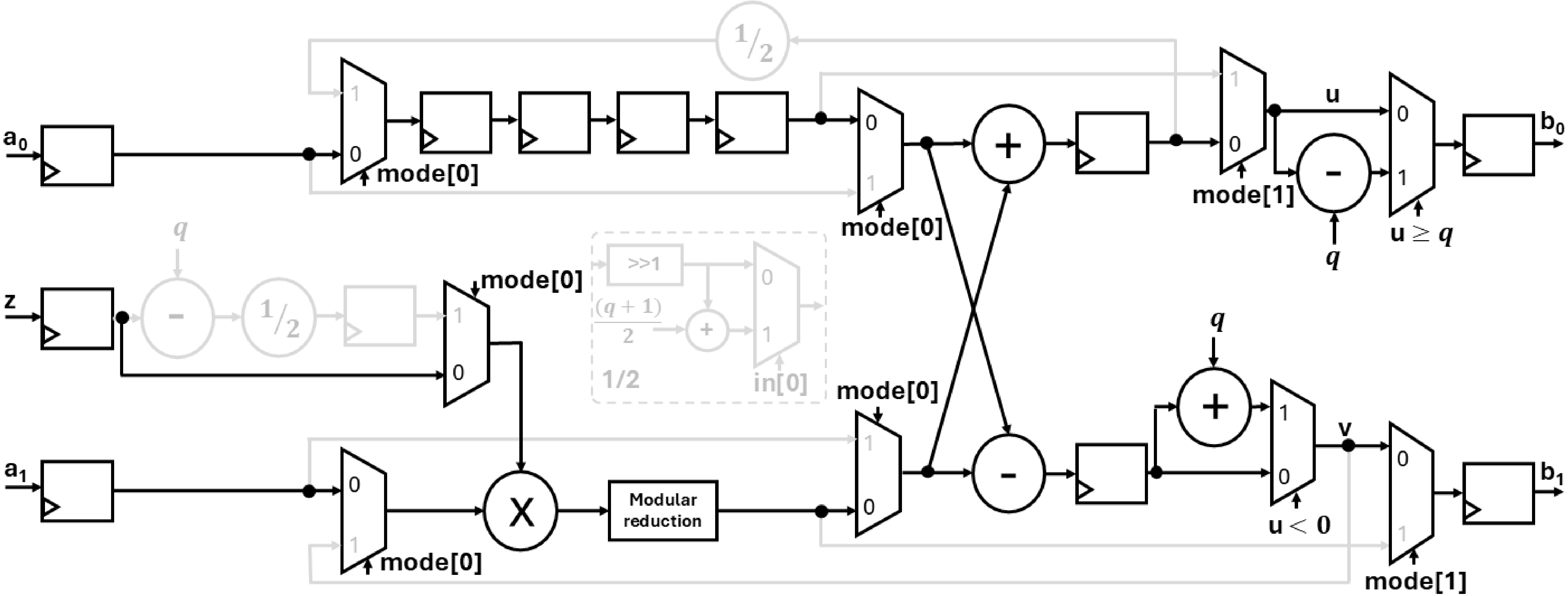
mode[1:0]	[00]	[01]	[10]	[11]
	NTT/MAC	ADD/SUB	PWM	INTT



Butterfly Architecture

mode[1:0]	[00]	[01]	[10]	[11]
	NTT/MAC	ADD/SUB	PWM	INTT

$$b_0 = a_0 + (a_1 * z) \mod q$$
$$b_1 = a_0 - (a_1 * z) \mod q$$

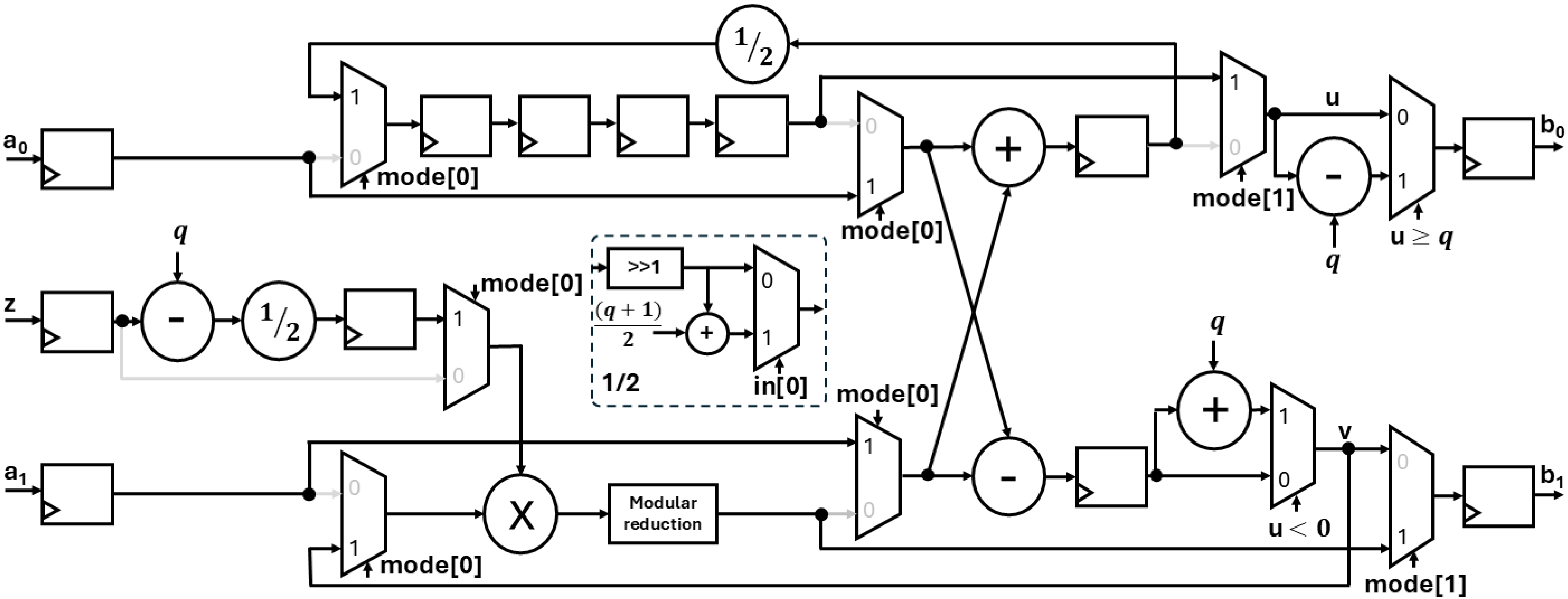


Butterfly Architecture

mode[1:0]	[00]	[01]	[10]	[11]
	NTT/MAC	ADD/SUB	PWM	INTT

$b_0 = (a_0 + a_1)/2 \bmod q$

$b_1 = (a_0 - a_1) * z^{-1}/2 \bmod q$



Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right)$; s.t $e \leq {}^{1}/_{(q-1)^2}$

$$\underset{\boxed{t}}{n} = \underset{\boxed{u * m}}{2n * n} \gg \underset{\boxed{k}}{2n} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$out \cong u - t * q$$

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^{k}}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right); \text{ s.t } e \leq {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n * n}{\boxed{u * m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$out \cong u - t * q$$

- Additional multiplications that are expensive in time and hardware

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

[Kim-19], [Pham-23]

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right);\ \text{s.t } e \leq {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n * n}{\boxed{u * m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$out \cong u - t * q$$

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1\ ,\ k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

- Additional multiplications that are expensive in time and hardware

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right);\ \text{s.t } e \leq {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n*n}{\boxed{u * m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

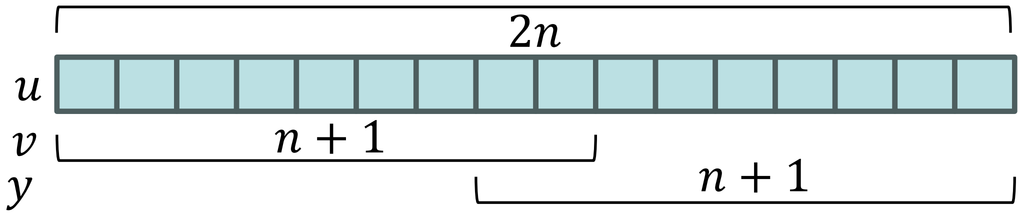
$$out \cong u - t * q$$

[Kim-19], [Pham-23]

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1\ ,\ k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$



▶ Additional multiplications that are expensive in time and hardware

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right);\ \text{s.t } e \leq {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n * n}{\boxed{u * m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

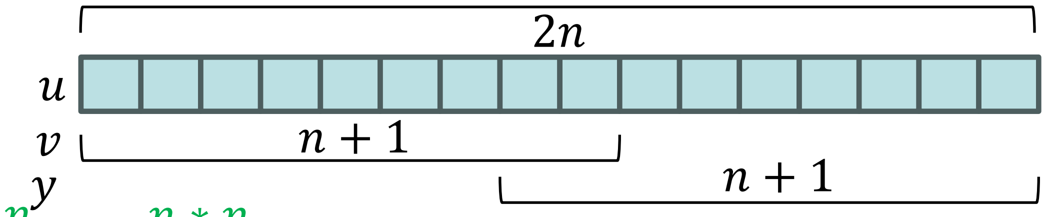
$$out \cong u - t * q$$

[Kim-19], [Pham-23]

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1\ ,\ k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$



$$\overset{n}{\boxed{w}} = \overset{n * n}{\boxed{v * m}} \gg (n + 1) \cong \text{quotient}\left({}^{u}/_{q}\right)$$

- Additional multiplications that are expensive in time and hardware

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right);\ \text{s.t}\ e \le {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n*n}{\boxed{u*m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

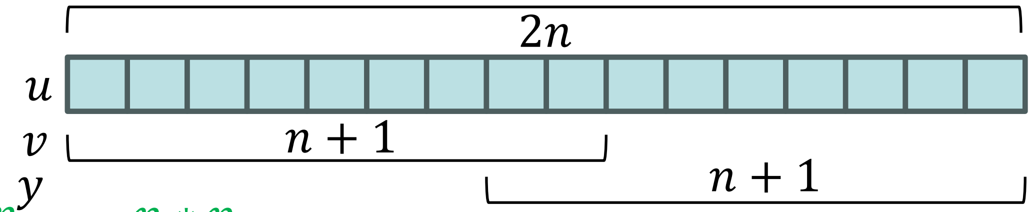
$$out \cong u - t * q$$

[Kim-19], [Pham-23]

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1\ ,\ k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$



$$\overset{n}{\boxed{w}} = \overset{n*n}{\boxed{v*m}} \gg (n+1) \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$x = w * q \bmod 2^{n+1}$$

- Additional multiplications that are expensive in time and hardware

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$$e = \left({}^{1}/_{q}\right) - \left({}^{m}/_{2^k}\right);\ \text{s.t}\ e \le {}^{1}/_{(q-1)^2}$$

$$\overset{n}{\boxed{t}} = \overset{2n*n}{\boxed{u*m}} \gg \overset{2n}{\boxed{k}} \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$out \cong u - t * q$$

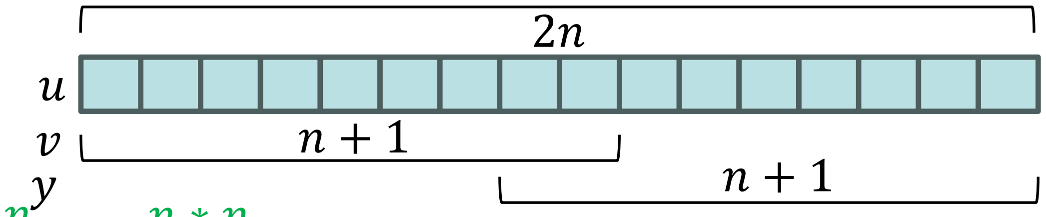
▶ Additional multiplications that are expensive in time and hardware

[Kim-19], [Pham-23]

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1\ ,\ k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$



$$\overset{n}{\boxed{w}} = \overset{n*n}{\boxed{v*m}} \gg (n+1) \cong \text{quotient}\left({}^{u}/_{q}\right)$$

$$x = w * q \bmod 2^{n+1}$$

$$out \cong y - x$$

Optimized Barrett Reduction

$$a * b \bmod q$$

$$u = a * b$$

Original Barrett

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$

$e = \left(1/q\right) - \left(m/2^k\right)$; s.t $e \leq 1/(q-1)^2$

$$\underbrace{\boxed{t}}_{n} = \underbrace{\boxed{u * m}}_{2n * n} \gg \underbrace{\boxed{k}}_{2n} \cong \text{quotient}\left(u/q\right)$$

$$out \cong u - t * q$$

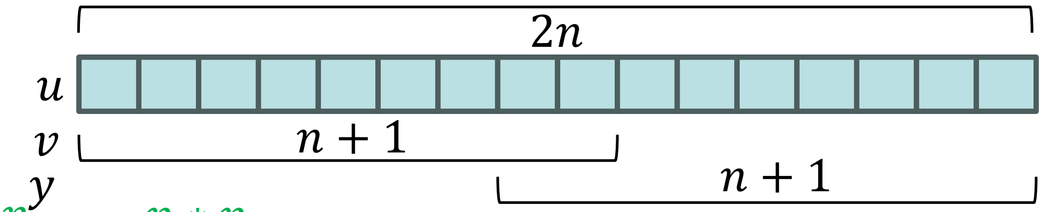
▶ Additional multiplications that are expensive in time and hardware

[Kim-19], [Pham-23]

Optimized Barrett

$$n = \lfloor log_2(q) \rfloor + 1 \text{ , } k = 2 * n$$

$$m = \left\lfloor \frac{2^k}{q} \right\rfloor$$



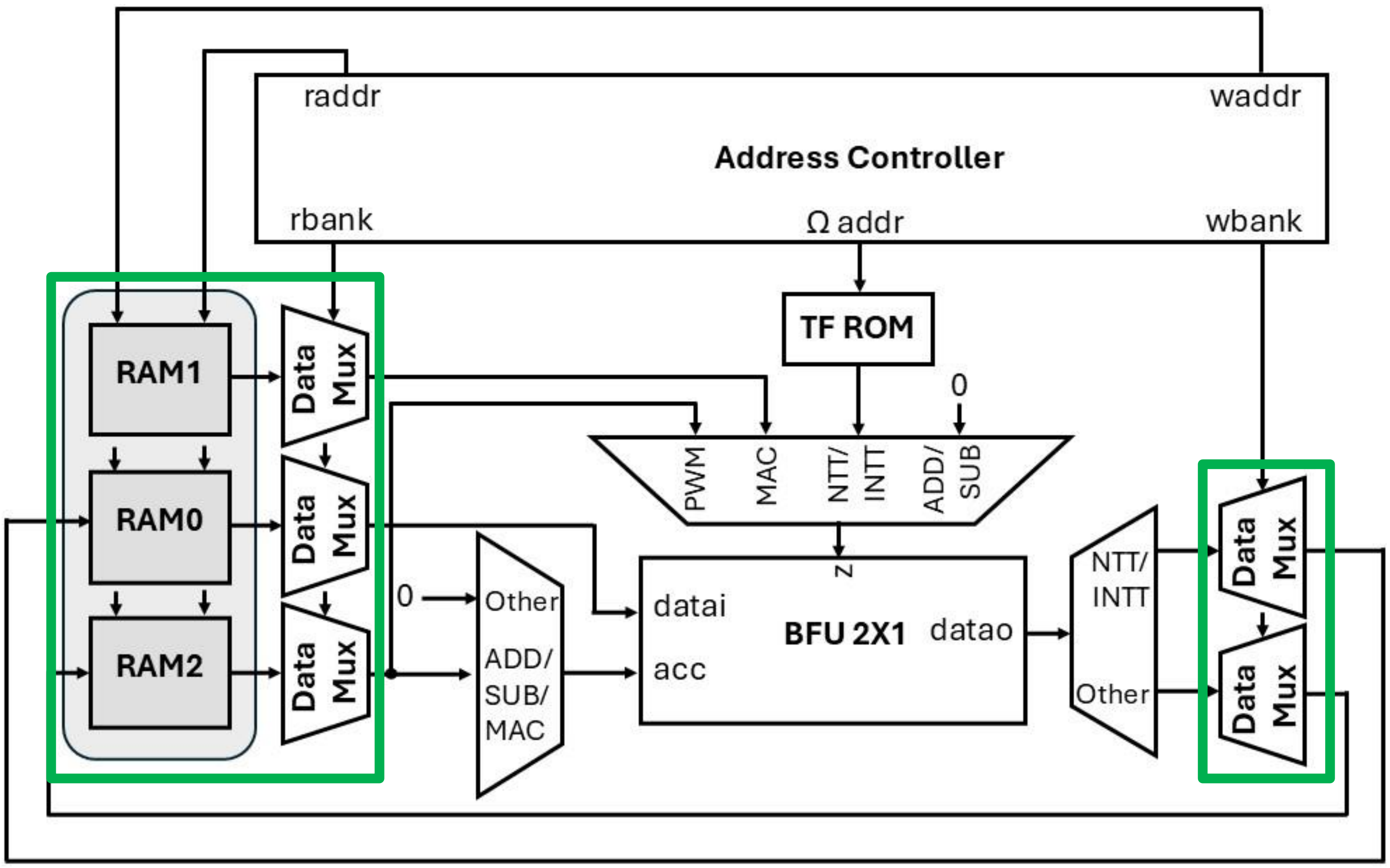
$$\underbrace{\boxed{w}}_{n} = \underbrace{\boxed{v * m}}_{n * n} \gg (n+1) \cong \text{quotient}\left(u/q\right)$$

$$x = w * q \bmod 2^{n+1}$$

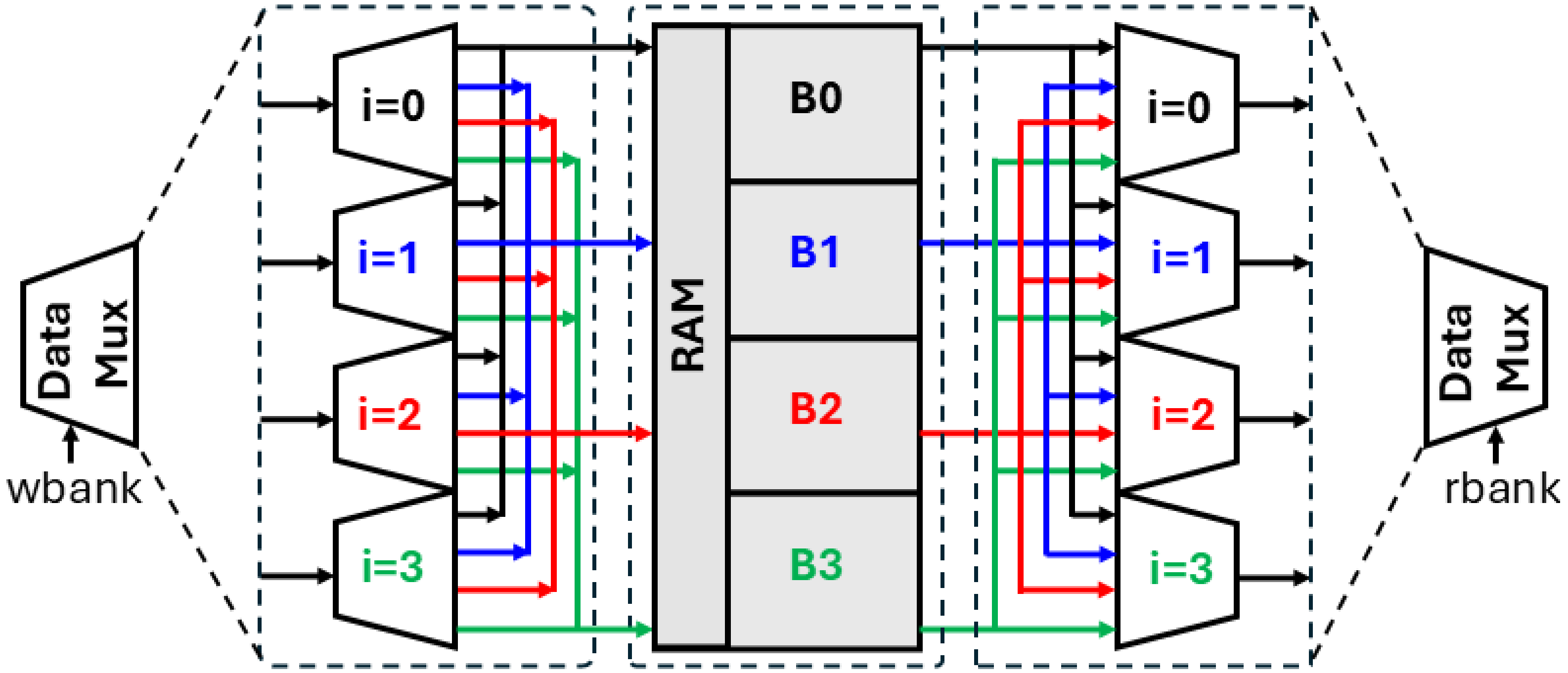
$$out \cong y - x$$

▶ These multiplications can be done using simple addition, subtraction and shift operations

Memory


raddr
waddr
Address Controller
rbank
Ω addr
wbank
TF ROM
0
RAM1
RAM0
RAM2
Data Mux
Data Mux
Data Mux
PWM
MAC
NTT/ INTT
ADD/ SUB
z
datai
acc
BFU 2X1
datao
0
Other
ADD/ SUB/ MAC
NTT/ INTT
Other
Data Mux
Data Mux

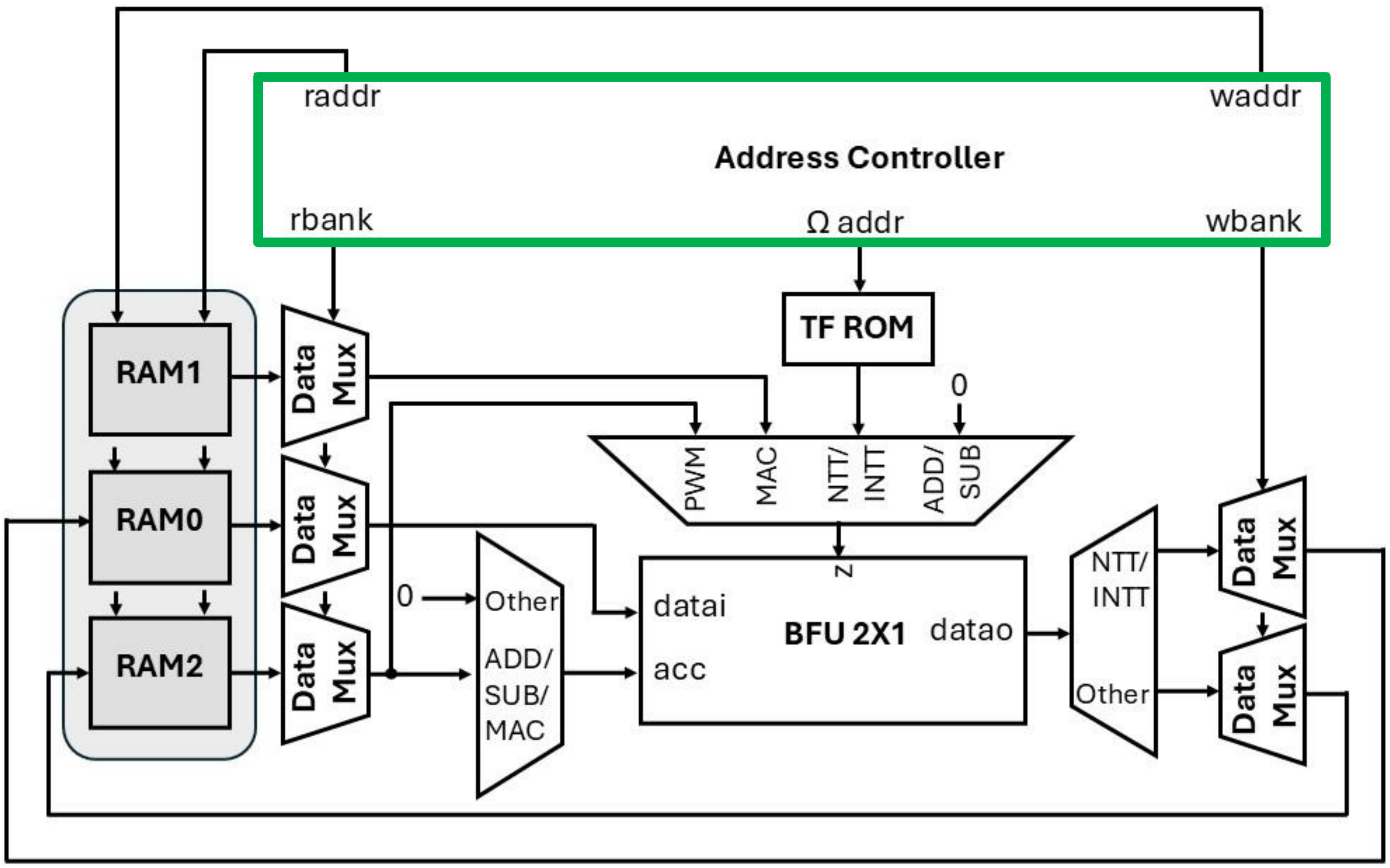
Memory



- dual-port 36-kbit BRAMs
- 1024 x 36-bits

n	# of polys
256	16
512	8
1024	4

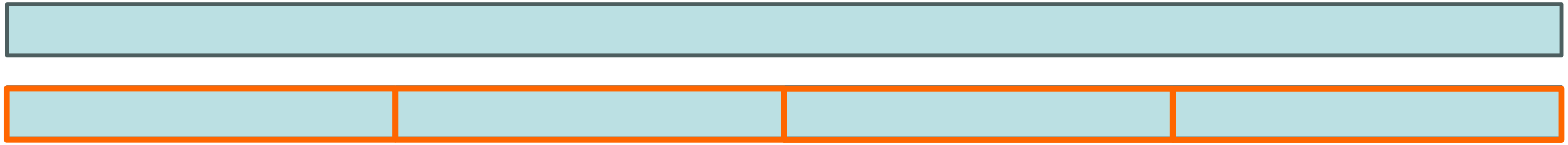
Address controller



Address controller

- How generic is the address controller?
  - The number of NTT levels ($\log(n)$) is even or odd.
  - The NTT fully splits the high-order poly or not (as in Kyber).

Address controller ($n = 256$)

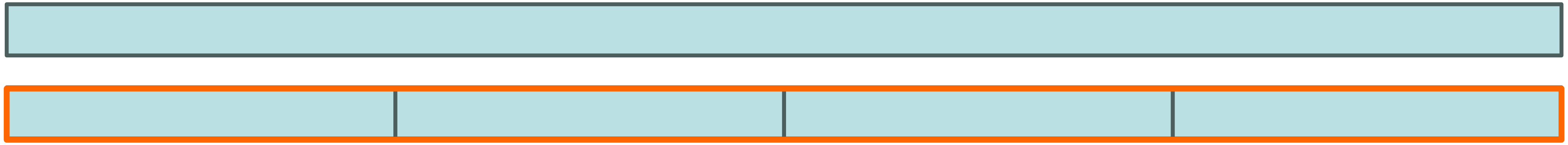


Layer: 0,1

layer	Ch	L	C	S
0,1	4	64	1	8

Address controller ($n = 256$)

Layer: 0,1



layer	Ch	L	C	S
0,1	4	64	1	8

$r_{a0} = cnt_0 + (cnt_1 \ll S)$

$r_{a1} = cnt_0 + (cnt_1 \ll S) + L$

$r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$

$r_{a3} = cnt_0 + (cnt_1 \ll S) + (L \ll 1) + L$

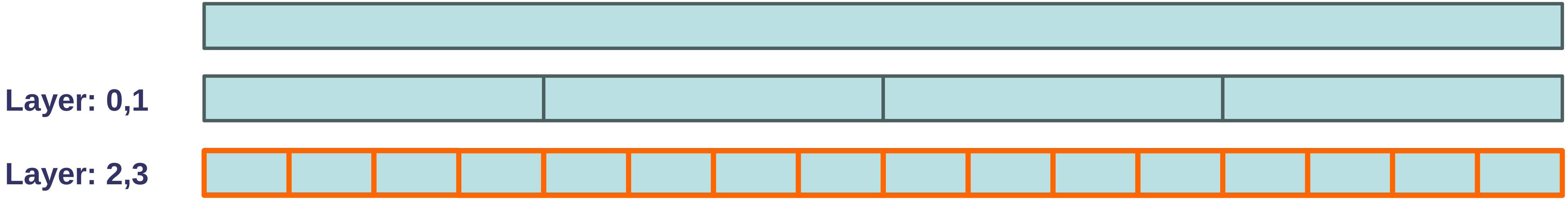
NTT/INTT

Layer 0		Layer 1	
[0,128]	[64,192]	[0,64]	[128,192]
[1,129]	[65,193]	[1,65]	[129,193]
…		…	
[63,191]	[127,255]	[63,127]	[191,255]

Twiddle factor

Layer 0		Layer 1	
[1]	[1]	[2]	[3]

Address controller ($n = 256$)

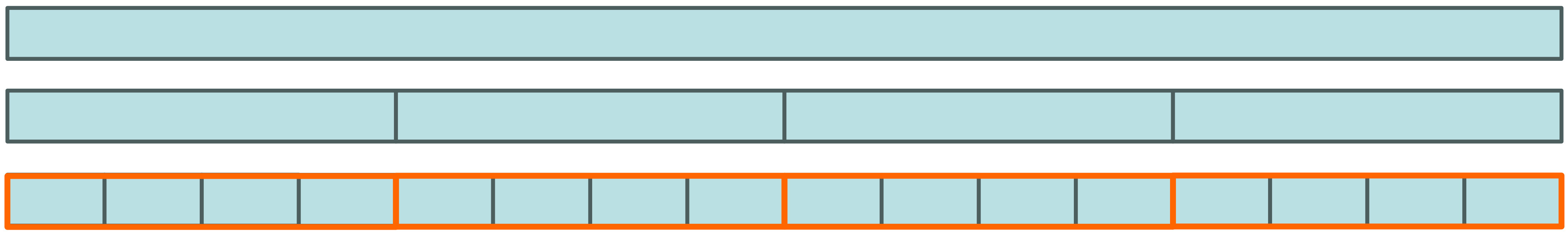


layer	Ch	L	C	S
0,1	4	64	1	8
2,3	16	16	4	6

Address controller ($n = 256$)

Layer: 0,1

Layer: 2,3



layer	Ch	L	C	S
0,1	4	64	1	8
2,3	16	16	4	6

$$r_{a0} = cnt_0 + (cnt_1 \ll S) \qquad r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$$

$$r_{a1} = cnt_0 + (cnt_1 \ll S) + L \qquad r_{a3} = cnt_0 + (cnt_1 \ll S) + (L \ll 1) + L$$

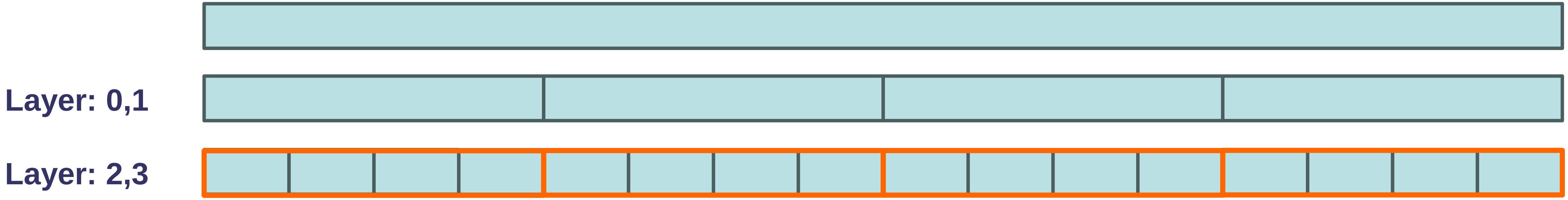
NTT/INTT

Layer 2	Layer 3
[0,32] ... [16,48] [15,47] [31,63]	[0,16] ... [32,48] [15,31] [47,63]
[64,96] ... [80,12] [79,111] [95,127]	[64,80] ... [96,12] [79,95] [111,127]
[128,160] ... [144,176] [143,175] [159,191]	[128,144] ... [160,176] [143,159] [175,191]
[192,224] ... [208,240] [207,239] [223,255]	[192,208] ... [224,240] [207,223] [239,255]

Twiddle factor

Layer 2	Layer 3
[4] [4] [5] [5] [6] [6] [7] [7]	[8] [9] ... [14] [15]

Address controller ($n = 256$)



layer	Ch	L	C	S
0,1	4	64	1	8
2,3	16	16	4	6

NTT/INTT

	Layer 2	Layer 3
0	[0,32] … [16,48] [15,47] [31,63]	[0,16] … [32,48] [15,31] [47,63]
	[64,96] … [80,12] [79,111] [95,127]	[64,80] … [96,12] [79,95] [111,127]
	[128,160] [144,176] … [143,175] [159,191]	[128,144] [160,176] … [143,159] [175,191]
	[192,224] [208,240] … [207,239] [223,255]	[192,208] [224,240] … [207,223] [239,255]

Twiddle factor

Layer 2	Layer 3
[4] [4] [5] [5] [6] [6] [7] [7]	[8] [9] … [14] [15]

$r_{a0} = cnt_0 + (cnt_1 \ll S)$

$r_{a1} = cnt_0 + (cnt_1 \ll S) + L$

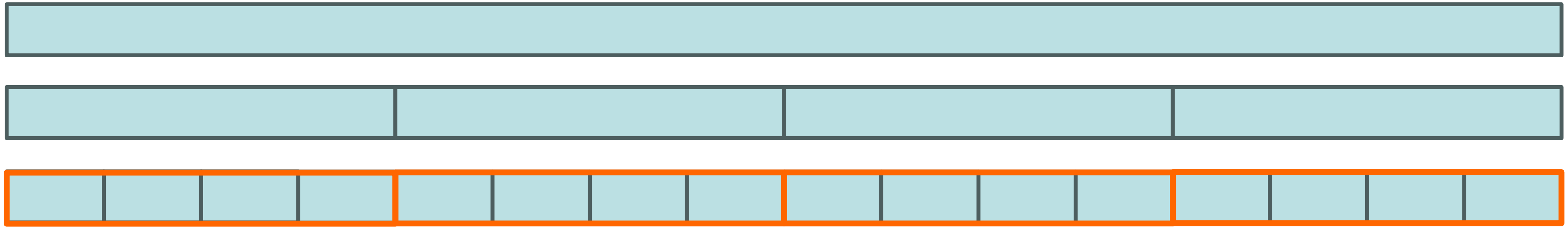
$r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$

$r_{a3} = cnt_0 + (cnt_1 \ll S) + (L \ll 1) + L$

Address controller ($n = 256$)

Layer: 0,1

Layer: 2,3



layer	Ch	L	C	S
0,1	4	64	1	8
2,3	16	16	4	6

$$r_{a0} = cnt_0 + (cnt_1 \ll S) \qquad r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$$

$$r_{a1} = cnt_0 + (cnt_1 \ll S) + L \qquad r_{a3} = cnt_0 + (cnt_1 \ll S) + (L \ll 1) + L$$

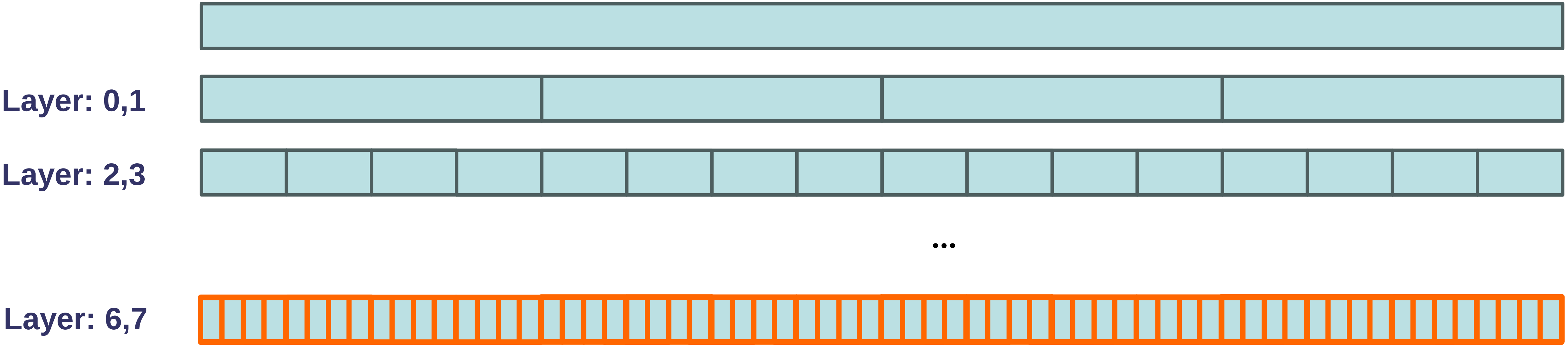
NTT/INTT

	Layer 2	Layer 3
	[0,32] … [16,48] [15,47] [31,63]	[0,16] … [32,48] [15,31] [47,63]
1	[64,96] … [80,12] [79,111] [95,127]	[64,80] … [96,12] [79,95] [111,127]
	[128,160] … [144,176] [143,175] [159,191]	[128,144] … [160,176] [143,159] [175,191]
	[192,224] … [208,240] [207,239] [223,255]	[192,208] … [224,240] [207,223] [239,255]

Twiddle factor

Layer 2	Layer 3
[4] [4] [5] [5] [6] [6] [7] [7]	[8] [9] … [14] [15]

Address controller ($n = 256$)



layer	Ch	L	C	S
0,1	4	64	1	8
2,3	16	16	4	6
4,5	64	4	16	4
6,7	256	1	64	2

$r_{a0} = cnt_0 + (cnt_1 \ll S)$

$r_{a1} = cnt_0 + (cnt_1 \ll S) + L$

$r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$

$r_{a3} = cnt_0 + (cnt_1 \ll S) + (L \ll 1) + L$

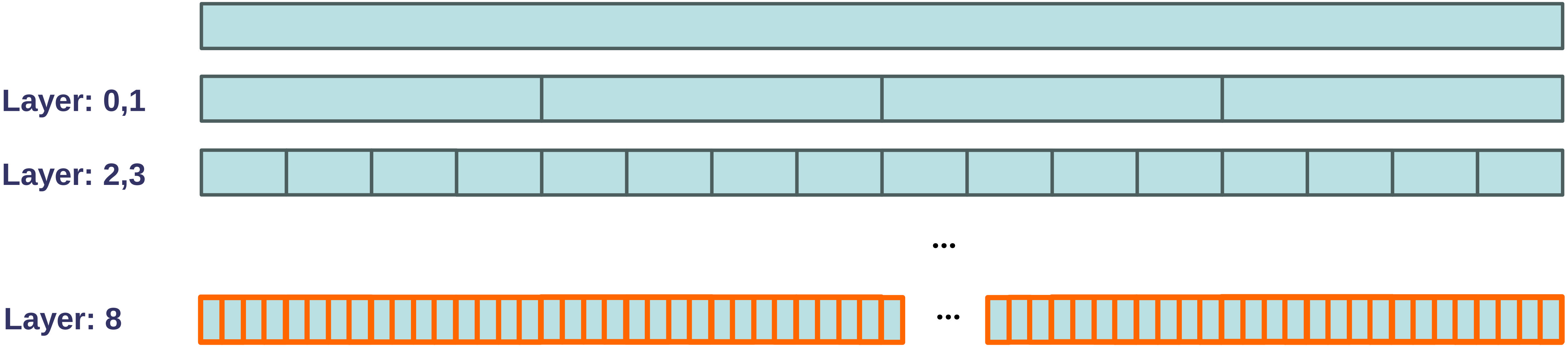
NTT/INTT

Layer 6		Layer 7	
[0,2]	[1,3]	[0,1]	[2,3]
...		...	
[252,254]	[253,255]	[252,253]	[254,255]

Twiddle factor

Layer 6		Layer 7	
[64]	[64]	[128]	[129]
...		...	
[127]	[127]	[254]	[255]

Address controller ($n = 512$)



layer	Ch	L	C	S
0,1	4	128	1	9
2,3	16	32	4	7
4,5	64	16	16	5
6,7	256	2	64	3
8	512	1	128	2

$r_{a0} = cnt_0 + (cnt_1 \ll S)$ $r_{a2} = cnt_0 + (cnt_1 \ll S) + (L \ll 1)$

NTT/INTT

Layer 7	Layer 8
[0,2] [4,6] [1,3] [5,7]	[0,1] [2,3]
[8,10] [12,14] [9,11] [13,15]	...
...	
[504,506] [508,510] [505,507] [509,511]	[508,509] [510,511]

Twiddle factor

Layer 7	Layer 8
[128] [129] ... [254] [255]	[256] [257] ... [510] [511]

Address controller: conflict-free memory access

[Zhang-20], [Land-21]

$$Bank = \sum_{i=0}^{\left\lceil \frac{1}{2}\log(n) \right\rceil - 1} RawAddr[2i+1:2i] \mod 4$$

$$RAMAddr = RawAddr \gg 2$$

$RAMAddr$	Bank 0	Bank 1	Bank 2	Bank 3
0	0	1	2	3
1	7	4	5	6
2	10	11	8	9
3	13	14	15	12
4	19	16	17	18
5	22	23	20	21
6	25	26	27	24
7	28	29	30	31
8	34	35	32	33
9	37	38	39	36
10	40	41	42	43
11	47	44	45	46
12	49	50	51	48
13	52	53	54	55
14	59	56	57	58
15	62	63	60	61

Performances – Digital Signatures

Dilithium									
Work	**Platform**	**Resources**				**Freq.**	**Latency (CCs)**		
		LUT	**FF**	**DSP**	**BRAM**	**[MHz]**	**NTT**	**INTT**	**PWM**
Nguyen-OA24	A7	7451	5275	0	0	180	319	319	
Land-Cardis21	A7	5676	1218	41	1	311	533	536	
Bekwith-ICFPT21	VUS+	4509	3146	8	0				
Zhao-TCHES22	Z7000	2812	1748	10	2		296	296	
Gupta-TCAS-I23	ZUS+	2759	2037	4	7		606	614	147
Pham-TCAS-I23	ZUS+	2637	1071	8	1	385	268	268	
Wang-TVLSI22	Z7000	2386	932	8	1	217	264		
Derya-eprint21	A7	2119	1058	8	3	117	1052	1318	3688
This work	**A7**	**2604**	**770**	**4**	**0**	**100**	**519**	**519**	**134**

Performances – Digital Signatures cont.

Hawk									
Work	Platform	Resources				Freq.	Latency (CCs)		
		LUT	FF	DSP	BRAM	[MHz]	NTT	INTT	PWM
Hawk512, p1	A7	3801	1135	8	0	83	1159	1159	262
Hawk512, p2	A7	3968	1135	8	0	83	1159	1159	262
Hawk1024, p1	A7	4287	1139	8	0	83	2567	2567	518
Hawk1024, p2	A7	4451	1139	8	0	83	2567	2567	518
Raccoon									
Raccoon, q1	A7	3194	912	4	0	83	1159	1159	262
Raccoon, q2	A7	3458	998	4	0	83	1159	1159	262

Performances – KEMs/Encryption schemes

Kyber									
Work	**Platform**	**Resources**				**Freq.**	**Latency (CCs)**		
		LUT	**FF**	**DSP**	**BRAM**	**[MHz]**	**NTT**	**INTT**	**PWM**
Nguyen,OA24	A7	4834	4683	0	1	250	247	247	
Derya,eprint21	V7	2128	1144	8	3	174	922	1184	3812
Xing-TCHES21	A7	1579	1058	2	3		512	448	256
Nguyen-TCAS-I24	A7	1416	1074	2	1.5	227	448	448	256
Ni-ISCAS23	A7	1154	1031	2	0	300	456	456	265
Yaman-DATE21	A7	948	352	1	2.5	190	904	904	3359
Zhang-ISCAS21	A7	609	640	2	4	257	490	490	
This work	**A7**	**1583**	**458**	**2**	**0**	**100**	**455**	**455**	**134**
Polka									
This work	**A7**	**2512**	**593**	**2**	**0**	**100**	**2567**	**2567**	**518**

Thank you

Lattice Isomorphism Problem (LIP)

- Two lattices are isomorphic if there exists an orthonormal transformation $\mathcal{O} \in \mathcal{O}_n(\mathbb{R})$ sending one to the other. Finding this transformation, if it exists, is known as the Lattice Isomorphism Problem (LIP).
- For any two lattices $\mathcal{L} = \mathcal{L}(\mathrm{B}), \acute{\mathcal{L}} = \mathcal{L}(\acute{\mathrm{B}})$ where $\mathcal{L}$ has a basis $\mathrm{B_Q}$ such that $\mathrm{B_Q^T} \cdot \mathrm{B_Q} = \mathrm{Q}$
  - Two quadratic forms $\mathrm{Q}, \acute{\mathrm{Q}}$ (also called the Gram matrices) are equivalent if there exists a unimodular U such that $\mathrm{U^T} \cdot \mathrm{Q} \cdot \mathrm{U} = \acute{\mathrm{Q}}$.
  - We have that two lattices are isomorphic if and only if their Gram matrices are equivalent.

Convolution-based NTT

- Cyclic Convolution-based NTT over the ring $\mathcal{R}_q = \mathbb{Z}_q/(x^n - 1)$
  - $c = a \cdot b \in \mathcal{R}_q$, then $c = \sum_{k=0}^{n-1} c_k x^k$,
    where $c_k = \sum_{i=0}^{k} a_i b_{k-1} + \sum_{i=k+1}^{n-1} a_i b_{k+n-i} \mod q$
- Negative Wrapped Convolution-based NTT over the ring

$$\mathcal{R}_q = \mathbb{Z}_q/(x^n + 1)$$

  - $c = a \cdot b \in \mathcal{R}_q$, then $c = \sum_{k=0}^{n-1} c_k x^k$,
    where $c_k = \sum_{i=0}^{k} a_i b_{k-1} - \sum_{i=k+1}^{n-1} a_i b_{k+n-i} \mod q$

How to find the roots of unity?

$X^4 + 1$

m_1	m_2
$X^2 - \alpha$ $X^2 - 4$	$X^2 + \alpha$ $X^2 + 4$
$X^2 - \zeta^2$	$X^2 + \zeta^2$

$\alpha^2 = -1,\ \alpha = 4$
$\alpha^4 = 1,\ \alpha = \zeta_4$

$\beta^2 = \alpha$
$\beta^2 = 4$
$\beta = 2$

$\beta = \sqrt{\alpha}$
And $\alpha = \zeta_4$
So, $\beta = \zeta_8$

m_{11}	m_{12}	m_{21}	m_{22}
$X - \beta$ $X - 2$	$X + \beta$ $X + 2$	$X - \gamma$ $X - 8$	$X + \gamma$ $X + 8$
$X - \zeta$	$X + \zeta$	$X - \zeta^3$	$X + \zeta^3$

$-\gamma^2 = \alpha$
$\gamma^2 = -\alpha$
$\gamma = \sqrt{-\alpha} = \sqrt{-1 \times \alpha}$
$\gamma = \sqrt{\alpha^2\alpha} = \sqrt{\alpha}^3$
$\gamma = \sqrt{4}^3 = 8$
$\gamma = \zeta_8^3$

Degree n 4			
Degree n/2 2		Degree n/2 2	
Degree n/4 1	Degree n/4 1	Degree n/4 1	Degree n/4 1

$\zeta_4 \rightarrow \zeta^2$

$\zeta_8 \rightarrow \zeta$

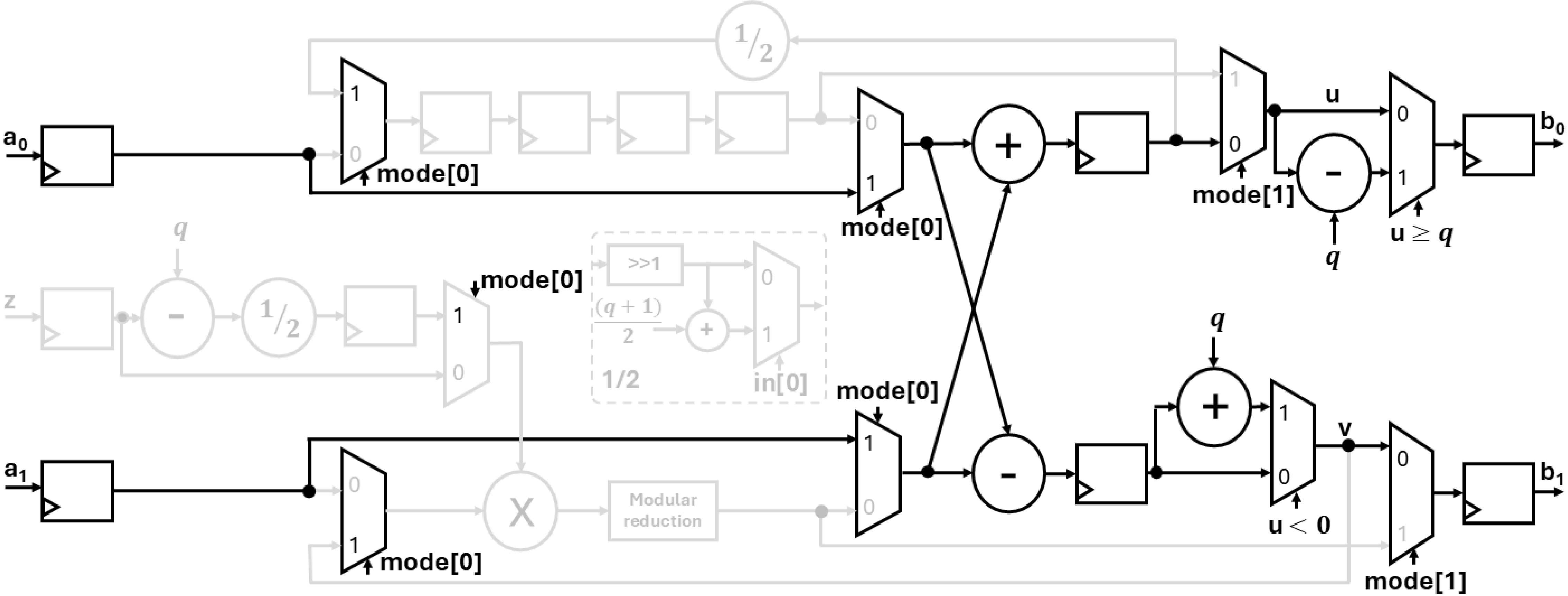
$\zeta_8^3 \rightarrow \zeta^3$

degree = 1

Butterfly Architecture

mode[1:0]	[00]	[01]	[10]	[11]
	NTT/MAC	ADD/SUB	PWM	INTT

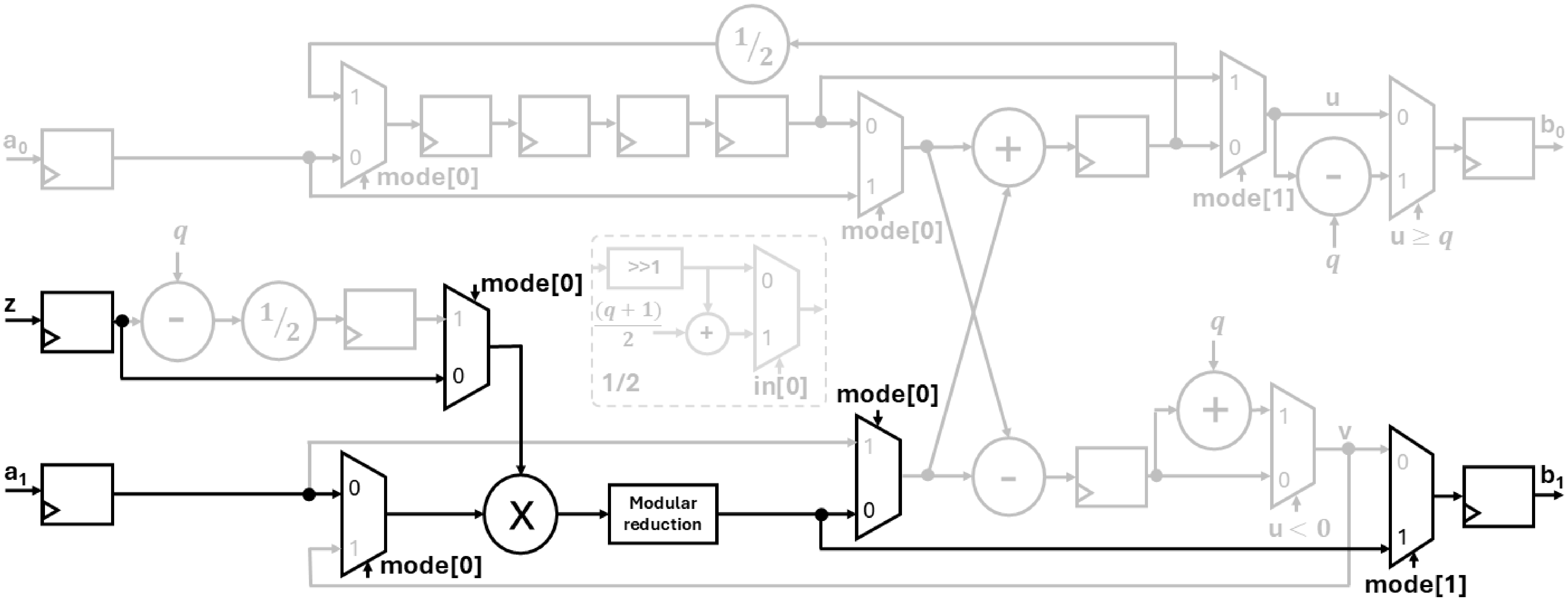
$$b_0 = a_0 + a_1 \mod q$$
$$b_1 = a_0 - a_1 \mod q$$



Butterfly Architecture

mode[1:0]	[00]	[01]	[10]	[11]
	NTT/MAC	ADD/SUB	PWM	INTT

$$b_1 = (a_1 * z) \bmod q$$



Why zeta = q - zetas[--k]?

e.g. $n = 8$ and $q = 113$

35^{-1} mod 113 = 1/35 mod 113 = (1+113*13)/35 mod 113 = 42
40^{-1} mod 113 = 1/40 mod 113 = (1+113*23)/40 mod 113 = 65
48^{-1} mod 113 = 1/48 mod 113 = (1+113*31)/48 mod 113 = 73
71^{-1} mod 113 = 1/71 mod 113 = (1+113*49)/71 mod 113 = 78

-71 mod 113 = 42
-48 mod 113 = 65
-40 mod 113 = 73
-35 mod 113 = 78

Why zeta = q - zetas[--k]?

e.g. $n = 8$ and $q = 113$

95^{-1} mod 113 = 1/95 mod 113 = (1+113*58)/95 mod 113 = 69 -44 mod 113 = 69
44^{-1} mod 113 = 1/44 mod 113 = (1+113*7) / 44 mod 113 = 18 -95 mod 113 = 18

35^{-1} mod 113 = 1/35 mod 113 = (1+113*13)/35 mod 113 = 42 -71 mod 113 = 42
40^{-1} mod 113 = 1/40 mod 113 = (1+113*23)/40 mod 113 = 65 -48 mod 113 = 65
48^{-1} mod 113 = 1/48 mod 113 = (1+113*31)/48 mod 113 = 73 -40 mod 113 = 73
71^{-1} mod 113 = 1/71 mod 113 = (1+113*49)/71 mod 113 = 78 -35 mod 113 = 78

Why zeta = q - zetas[--k]?

e.g. $n = 8$ and $q = 113$

98^{-1} mod 113 = 1/98 mod 113 = (1+113*13)/98 mod 113 = 15 → -98 mod 113 = 15

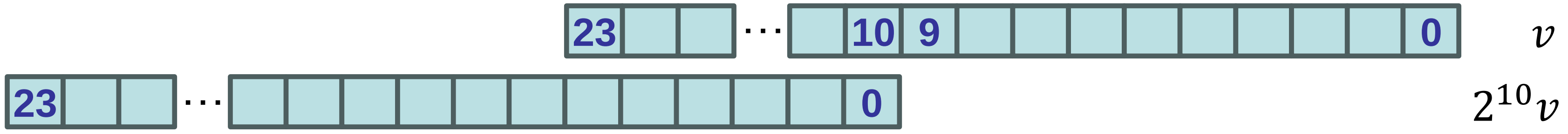
95^{-1} mod 113 = 1/95 mod 113 = (1+113*58)/95 mod 113 = 69 -44 mod 113 = 69
44^{-1} mod 113 = 1/44 mod 113 = (1+113*7) / 44 mod 113 = 18 -95 mod 113 = 18

35^{-1} mod 113 = 1/35 mod 113 = (1+113*13)/35 mod 113 = 42 -71 mod 113 = 42
40^{-1} mod 113 = 1/40 mod 113 = (1+113*23)/40 mod 113 = 65 -48 mod 113 = 65
48^{-1} mod 113 = 1/48 mod 113 = (1+113*31)/48 mod 113 = 73 -40 mod 113 = 73
71^{-1} mod 113 = 1/71 mod 113 = (1+113*49)/71 mod 113 = 78 -35 mod 113 = 78

Optimized Barrett Reduction (e.g. Dilithium, n = 23)

$$
\begin{aligned}
v * m &= v * \left(\underline{2^{23} + 2^{13}} + 2^2 + 2 + 1\right) \\
&= 2^{13} * \underbrace{\left(2^{10} v + v\right)}_{v_1} + 2 * (2v + v) + v
\end{aligned}
$$

$$v_1 = 2^{10} v + v = concat\{v[23:0] + v[23:10], v[9:0]\}$$



$$2 * (2v + v) + v = 2 * \underbrace{(2v + v + v[23:1])}_{v_2} + v[0]$$

$$v_2 = (v \ll 1) + v + v[23:1]$$

$$
\begin{aligned}
v * m &= 2^{13} v_1 + 2 v_2 + v[0] \\
&= 2 * (2^{12} v_1 + v_2) + v[0] \\
&= concat\{\underbrace{v_1 + v_2[25:12]}_{v_{12}}, v_2[11:0], v[0]\}
\end{aligned}
$$

$$w = v * m \gg 24 = v_{12}[34:11]$$

SOA - Dilithium

Work	FPGA	BFU	# BFU	Reduction	Config n	Config q	Description
Nguyen-OA24	A7	Radix-2 and Radix-4 multipath delay commutator (MDC)	8/16	K-RED	No	Yes, 23b, 12b	Reordering coefs after each BFU computation
Land-Cardis21	A7	Radix-2 CT/GS	2	Congruency: $2^{23} \equiv 2^{13} + 1 \bmod q$	No	No	Heavy use of DSPs with different operation modes 4 BRAMs to store coefs
Bekwith-ICFPT21	VUS+	Radix-4 CT/GS	4	Barrett	No	No	3 BRAMs to store coefs
Zhao-TCHES22	Z7000	Radix-2 MDC	4		No	No	Folding transformation: The first BFU computes the 1st layer in odd cycles and the 2nd layer in even cycles.
Gupta-TCAS-*I*23	ZUS+	Radix-2 CT/GS	2	Congruency: $2^{23} \equiv 2^{13} + 1 \bmod q$	No	No	Ping-pong memory access.
Pham-TCAS-*I*23	ZUS+	Radix-4 CT/GS	4	Modified Barrett	No	No	4 BRAMs to store coefs Conflict-free memory access
Wang-TVLSI22	Z7000	Radix-2 and Radix-4 CT/GS	2/4	Congruency: $2^{23} \equiv 2^{13} + 1 \bmod q$	No	No	4 BRAMs to store coefs Conflict-free memory access
Derya-eprint21	A7	Radix-2 CT/GS	1/8/32	Montgomery	Yes	Yes	# of BFU chosen at compile time while the n,q are configured at run time 2 BRAMs for each BFU and 1 BFRAM for TF

SOA - Kyber

Work	FPGA	BFU	# BFU	Reduction	Config n	Config q	Description
Nguyen,OA24	A7	Radix-2 and Radix-4 multipath delay commutator (MDC)	7/14	K-RED	No	Yes, 23b, 12b	Reordering coefs after each BFU computation
Derya,eprint21	V7	Radix-2 CT/GS	1/8/32	Montgomery	Yes	Yes	# of BFU chosen at compile time while the n,q are configured at run time 2 BRAMs for each BFU and 1 BFRAM for TF 4 DSPs in multiplication and 4 DSPs in modular reduction
Xing-TCHES21	A7	Radix-2 CT/GS	2	Adapted Barrett	No	No	
Nguyen-TCAS-*I*24	A7	Radix-2 CT/GS	2	LUT	No	No	Reordering of poly coef instead of changing the addr.
Ni-ISCAS23	A7	Radix-2 CT/GS	2	K-RED + LUT	No	No	Reordering of poly coef instead of changing the addr.
Yaman-DATE21	A7	Radix-2 CT/GS	1/4/16	Congruency: $2^{23} \equiv 2^{13} + 1 \bmod q$	No	No	A changing memory read pattern for efficient memory management.
Zhang-ISCAS21	A7	Radix-2 and Radix-4 CT/GS	2/4	Congruency: $2^{23} \equiv 2^{13} + 1 \bmod q$	No	No	Ping-pong memory access.

Digital signature output sizes

	Dilithium			Hawk		Raccoon		
NIST security level	2	3	5	1	5	1	3	5
Public key size (B)	1312	1952	2592	1024	2440	2256	3160	4064
Signature size (B)	2420	3293	4595	555	1221	11524	14544	20330
Private key size (B)	2528	4128	4864	184	360	14800*	18840*	26016*

* The values given are for unprotected Raccoon and they increase slightly as the masking order increases.