CFNTT: Scalable Radix-2/4 NTT Multiplication Architecture with an Efficient Conflict-free Memory Mapping Scheme

Xiangren Chen, Bohan Yang, Shouyi Yin, Shaojun Wei and Leibo Liu*

School of Integrated Circuits, Tsinghua University, China.

CHES 2022, Issue 1

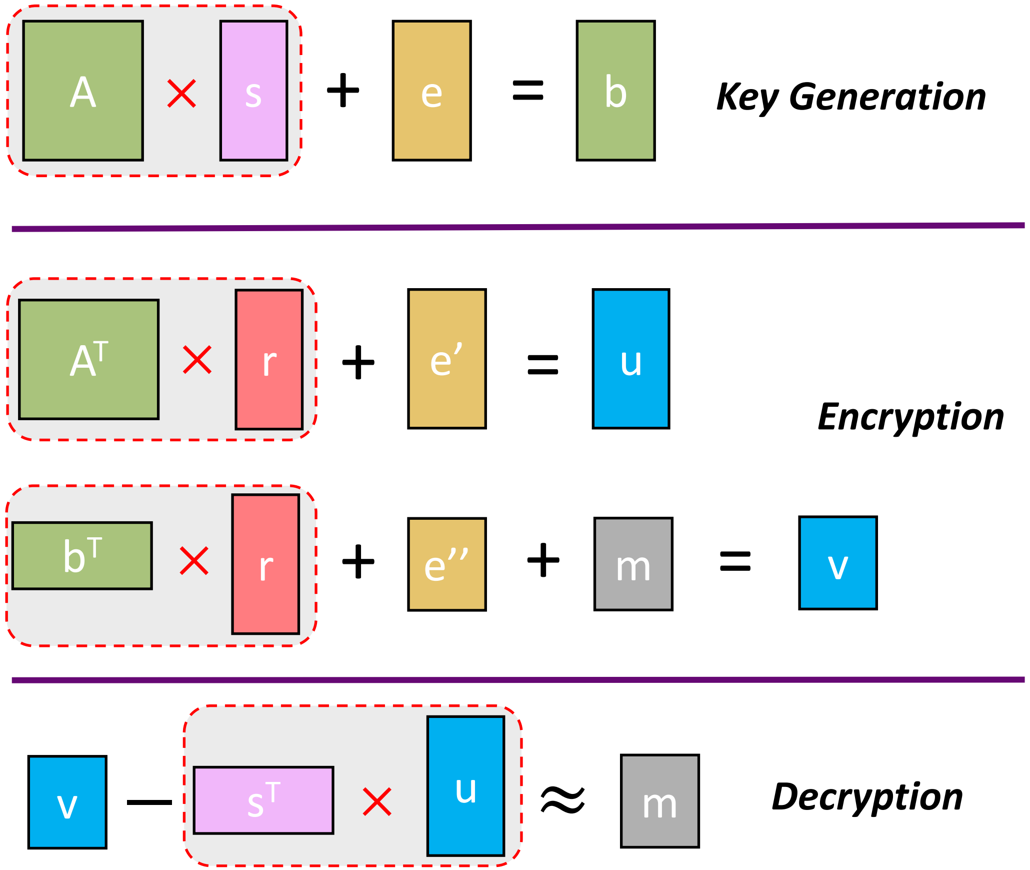
Outlines

- Introduction
- Optimized Radix-4 NTT/INTT Algorithm
- Conflict-free Memory Mapping Scheme
- Hardware Architecture of Radix-2/4 NTT
- Implementation Result and Comparison

Outlines

- Introduction
- Optimized Radix-4 NTT/INTT Algorithm
- Conflict-free Memory Mapping Scheme
- Hardware Architecture of Radix-2/4 NTT
- Implementation Result and Comparison

NTT Related Cryptographic Scheme

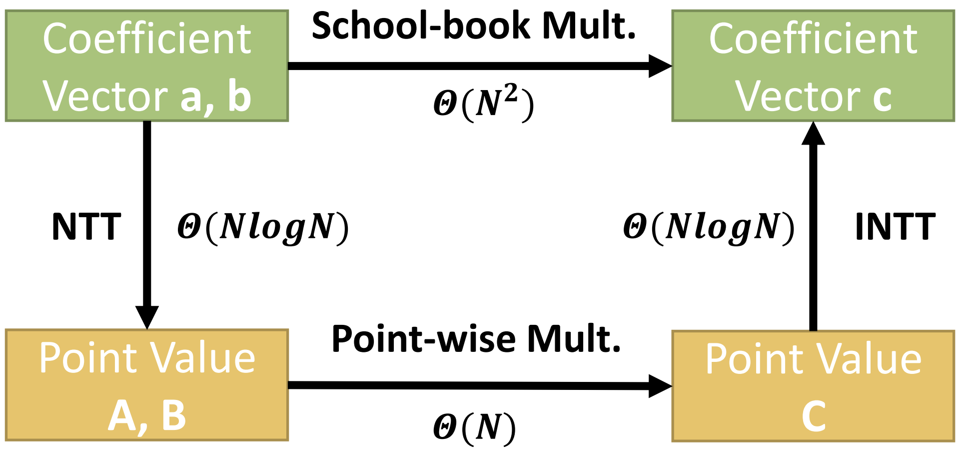


Polynomial computation in MLWE-based scheme e.g. Kyber

School-book Algorithm： $\Theta(N^2)$

Karatsuba Algorithm： $\Theta(N^{log2^3})$

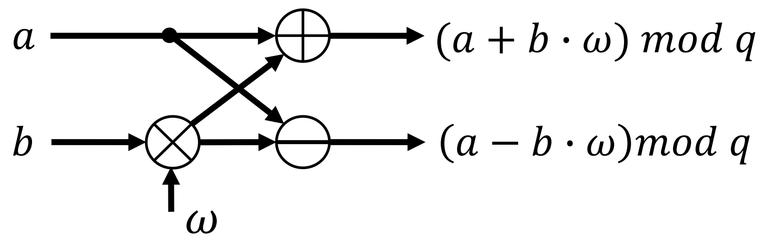
Number Theoretical Transformation： $\boldsymbol{\Theta(NlogN)}$



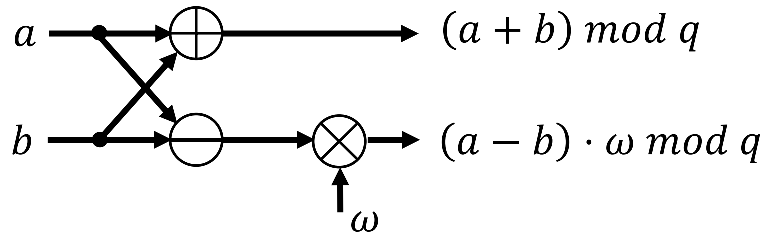
NTT Multiplication over the Ring

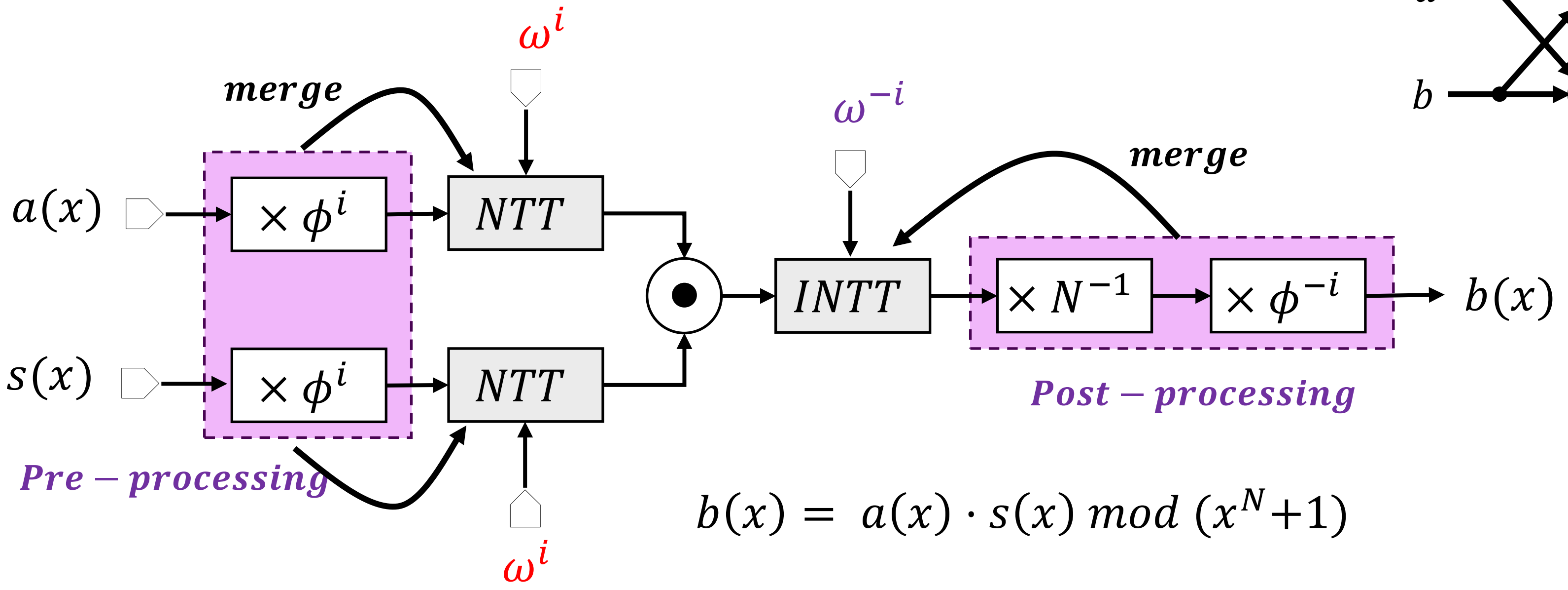
- $\mathbb{R}_q = \mathbb{Z}_q[x]/< f(x) >\quad q \equiv 1\ (mod\ 2N)$
- $f(x) = x^N + 1\ \Rightarrow$ Negative Wrapped Convolution (NWC)
- ω : $N\text{-}th$ roots of unity
- ϕ : $2N\text{-}th$ roots of unity

Radix-2 Cooley-Tukey BFU



Radix-2 Gentleman-Sande BFU

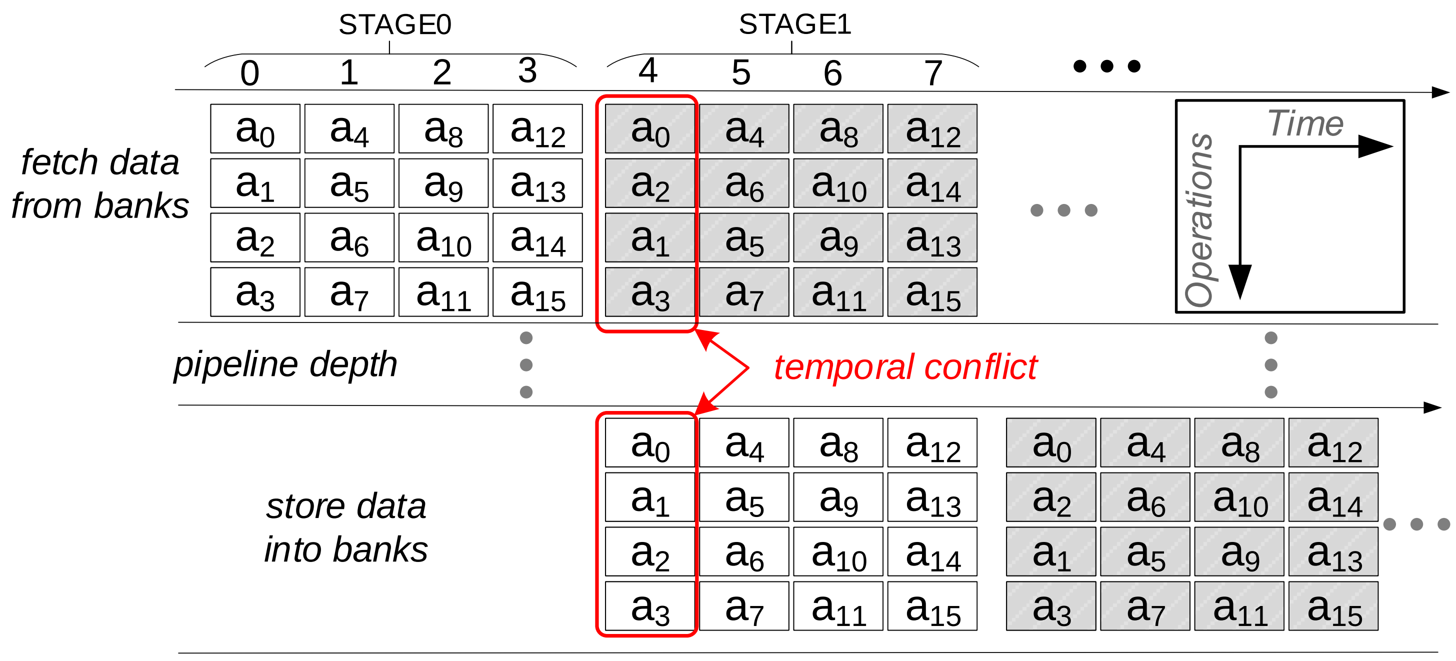




$$b(x) = a(x) \cdot s(x)\ mod\ (x^N+1)$$

Temporal Conflict

- Read After Write (RAW) pipelined hazard
  - $N/4d \geq pipeline\ depth$
  - More stringent when considering the higher radix NTT

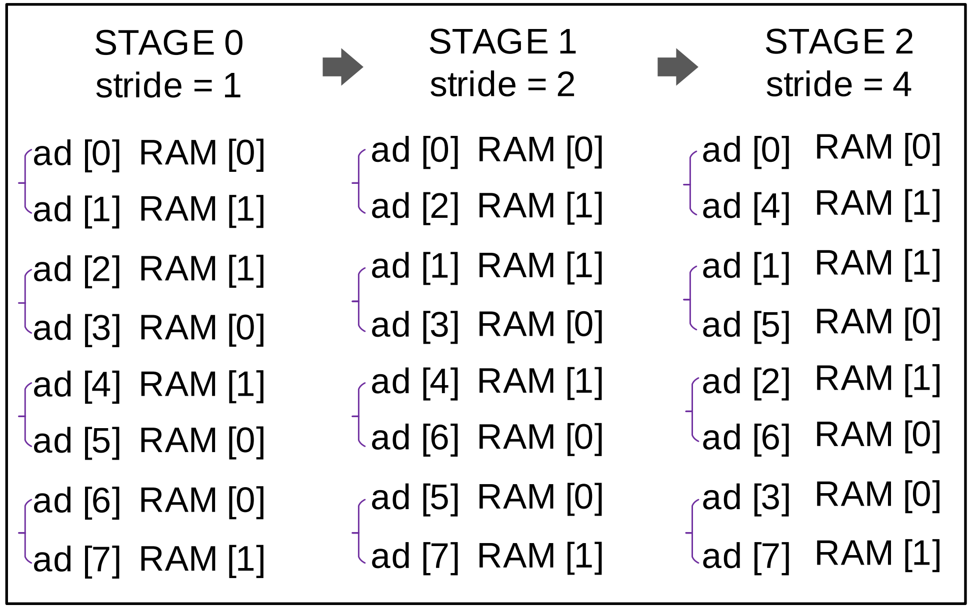


The dataflow of 16-point radix-2 in-place NTT with d = 2

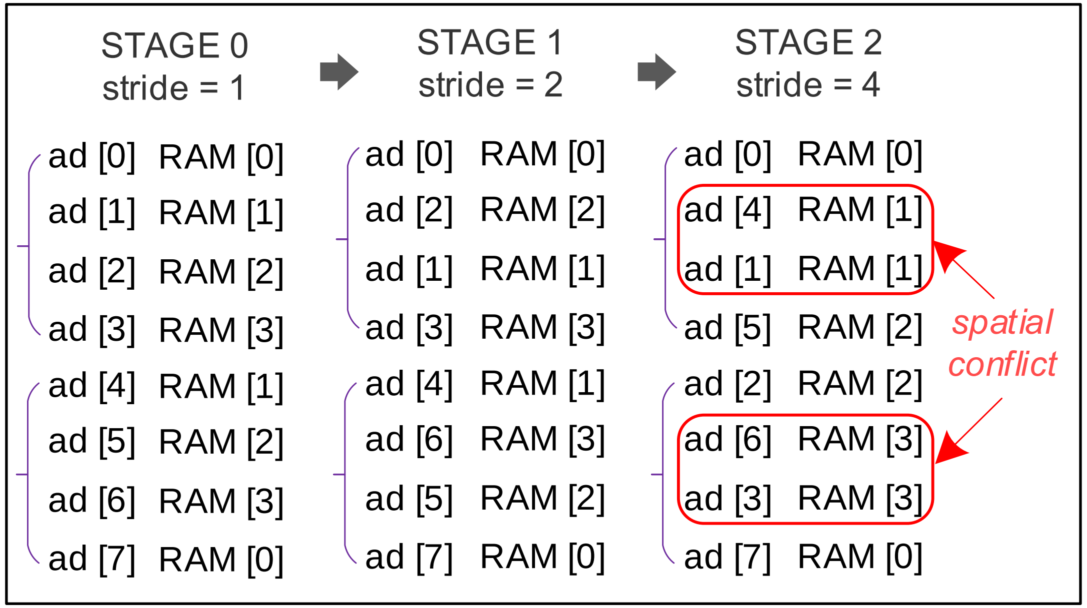
Spatial Conflict

Tsinghua University

[Joh92]
- ✓ *Memory mapping scheme for in-place FFT with arbitrary radix*
- ✗ *Placing multiple butterfly units leads to access conflict*



(a) Radix-2 in-place NTT with d = 1.



(b) Radix-2 in-place NTT with d = 2.

[Joh92] L. G. Johnson. Conflict free memory addressing for dedicated fft hardware. IEEE Trans. on Circuit and Systems-II, 39(5):312–316, 1992.

Motivations

Tsinghua University

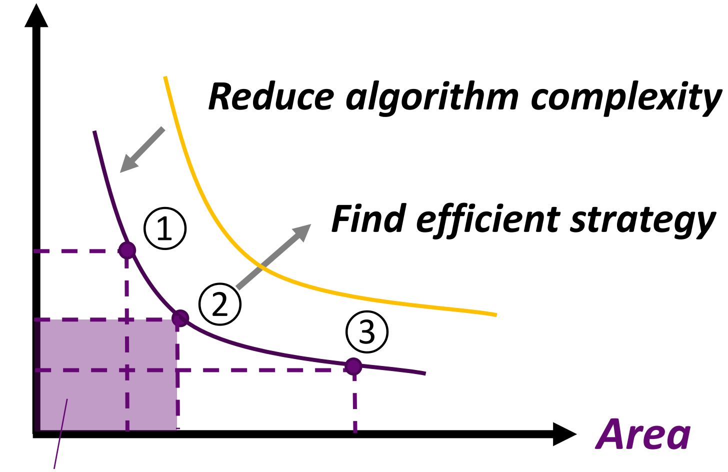
◻ 3Ds in lattice-based PQC:

- *<u>D</u>iverse security parameters of lattice-based scheme* — e.g. { *N = 1024 q = 12289 Falcon* ; *N = 256 q = 8380417 Dilithium* }
- *<u>D</u>ifferent resource constraints of computation platform* — *Embedded device vs. Server*
- *<u>D</u>ifferent throughput requirements of practical application* — *IoT vs. 5G*

Scalability Dimensions

T	*Types of NTT : in-place*	*Varying stride access*
N	*Length of vector is power-of-2*	*Design-time parameter*
q	*Different prime moduli 12-23bit*	
d	*Number of butterfly units*	*Seldomly considered at the same time*
R	*Radix-2 ⇒ radix-r^k*	

Time = Cycles / Frequency



ATP = Area x Time

Outlines

- Introduction
- **Optimized Radix-4 NTT/INTT Algorithm**
- Conflict-free Memory Mapping Scheme
- Hardware Architecture of Radix-2/4 NTT
- Implementation Result and Comparison

The Derivation of Radix-4 NTT without Preprocessing

Tsinghua University

$$A_i = \sum_{j=0}^{N-1} a_j \phi_{2N}^{j} \omega_N^{ij} \bmod q$$

Split →

$$A_i = \sum_{j=0}^{N/4-1} a_{4j}\phi_{2N}^{4j}\omega_N^{i\cdot(4j)} + \sum_{j=0}^{N/4-1} a_{4j+1}\phi_{2N}^{4j+1}\omega_N^{i\cdot(4j+1)} + \sum_{j=0}^{N/4-1} a_{4j+2}\phi_{2N}^{4j+2}\omega_N^{i\cdot(4j+2)} + \sum_{j=0}^{N/4-1} a_{4j+3}\phi_{2N}^{4j+3}\omega_N^{i\cdot(4j+3)} \bmod q$$

Elimination property

$$\omega_{dN}^{dk} = \omega_N^{k}$$

$$A_i = \sum_{j=0}^{N/4-1} \underbrace{a_{4j}\phi_{N/2}^{j}\omega_{N/4}^{ij}}_{F_0} + \omega_N^{i}\cdot\phi_{2N}^{1}\cdot \sum_{j=0}^{N/4-1} \underbrace{a_{4j+1}\phi_{N/2}^{j}\omega_{N/4}^{ij}}_{F_1} + \omega_N^{2i}\cdot\phi_{2N}^{2}\cdot \sum_{j=0}^{N/4-1} \underbrace{a_{4j+2}\phi_{N/2}^{j}\omega_{N/4}^{ij}}_{F_2} + \omega_N^{3i}\cdot\phi_{2N}^{3}\cdot \sum_{j=0}^{N/4-1} \underbrace{a_{4j+3}\phi_{N/2}^{j}\omega_{N/4}^{ij}}_{F_3} \bmod q$$

✓ $N\log_4 N + 2N \Rightarrow N\log_4 N$

Periodicity property

$$\omega_N^{k+N} = \omega_N^{k}$$

$i \in [0, N/4-1]$ ***Four N/4-point operation***

$$\begin{bmatrix} A_i \\ A_{i+N/4} \\ A_{i+2N/4} \\ A_{i+3N/4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^1 & -1 & -\omega_4^1 \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^1 & -1 & \omega_4^1 \end{bmatrix} \times \left(\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \phi_{2N}^{2i+1} \\ \phi_{2N}^{2(2i+1)} \\ \phi_{2N}^{3(2i+1)} \end{bmatrix} \right)$$

Recursion → N/4 4-points

The Derivation of Radix-4 INTT without Postprocessing

$$a_i = N^{-1} \cdot \phi_{2N}^{-i} \cdot \sum_{j=0}^{N-1} A_j \omega_N^{-ij} \bmod q$$

Split

$$a_i = N^{-1} \cdot (\phi_{2N}^{-i} \cdot \sum_{j=0}^{N/4-1} A_j \phi_{2N}^{-j} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \sum_{j=N/4}^{2N/4-1} A_j \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \sum_{j=2N/4}^{3N/4-1} A_j \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \sum_{j=3N/4}^{N-1} A_j \omega_N^{-ij}) \bmod q$$

Periodicity and Binary property

$$a_i = N^{-1} \cdot (\phi_{2N}^{-i} \cdot \sum_{j=0}^{N/4-1} A_j \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+N/4} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-2i} \cdot \sum_{j=0}^{N/4-1} A_{j+2N/4} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-3i} \cdot \sum_{j=0}^{N/4-1} A_{j+3N/4} \omega_N^{-ij}) \bmod q$$

Elimination property

$$a_i = N^{-1} \cdot (\phi_{2N}^{-i} \cdot \sum_{j=0}^{N/4-1} A_j \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+N/4} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-2i} \cdot \sum_{j=0}^{N/4-1} A_{j+2N/4} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-3i} \cdot \sum_{j=0}^{N/4-1} A_{j+3N/4} \omega_N^{-ij}) \bmod q$$

Next page

The Derivation of Radix-4 INTT without Postprocessing

Tsinghua University

$$a_i = N^{-1} \cdot (\phi_{2N}^{-i} \cdot \sum_{j=0}^{N/4-1} A_j \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+N/4} \omega_N^{-ij} +$$

$$\phi_{2N}^{-i} \cdot \omega_4^{-2i} \cdot \sum_{j=0}^{N/4-1} A_{j+2N/4} \omega_N^{-ij} + \phi_{2N}^{-i} \cdot \omega_4^{-3i} \cdot \sum_{j=0}^{N/4-1} A_{j+3N/4} \omega_N^{-ij}) \bmod q$$

$$\omega_4^{-i} = \begin{cases} 0 \\ \omega_4^{-1} \\ -1 \\ -\omega_4^{-1} \end{cases} \quad \omega_4^{-2i} = \begin{cases} 1 & i = 4r \\ -1 & i = 4r+1 \\ 1 & i = 4r+2 \\ -1 & i = 4r+3 \end{cases} \quad \omega_4^{-3i} = \begin{cases} 0 \\ \omega_4^{-1} \\ -1 \\ -\omega_4^{-1} \end{cases}$$

G_0, G_1, G_2, G_3:

$$a_{4i} = (\frac{N}{4})^{-1} \cdot (\underbrace{\frac{1}{4} \cdot \phi_{N/2}^{-i} \cdot \sum_{j=0}^{N/4-1} A_j \omega_{N/4}^{-ij}}_{G_0} + \underbrace{\frac{1}{4} \cdot \phi_{N/2}^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+N/4} \omega_{N/4}^{-ij}}_{G_1} +$$

$$\underbrace{\frac{1}{4} \cdot \phi_{N/2}^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+2N/4} \omega_{N/4}^{-ij}}_{G_2} + \underbrace{\frac{1}{4} \cdot \phi_{N/2}^{-i} \cdot \sum_{j=0}^{N/4-1} A_{j+3N/4} \omega_{N/4}^{-ij}}_{G_3}) \bmod q$$

✓ $Nlog_4N + N \Rightarrow Nlog_4N$

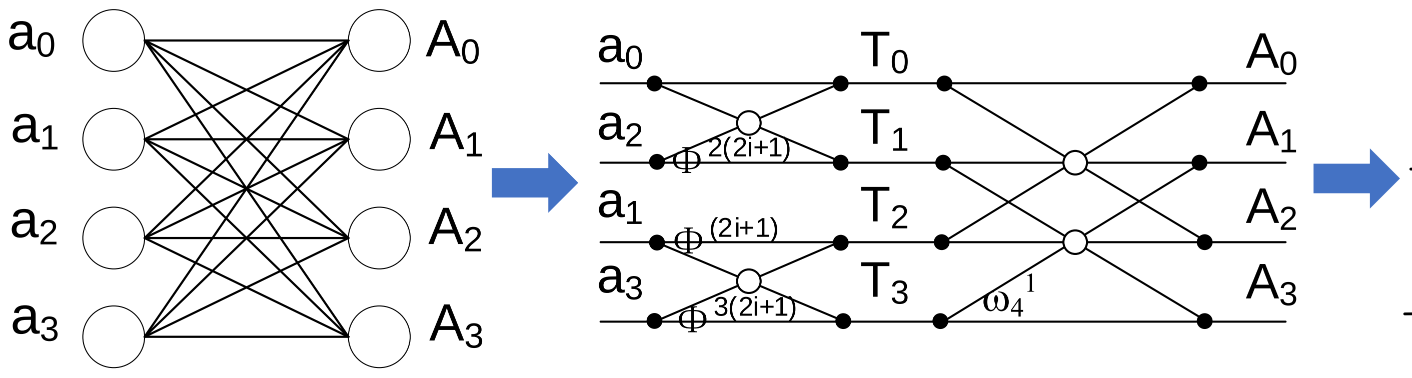
Sum and Binary property

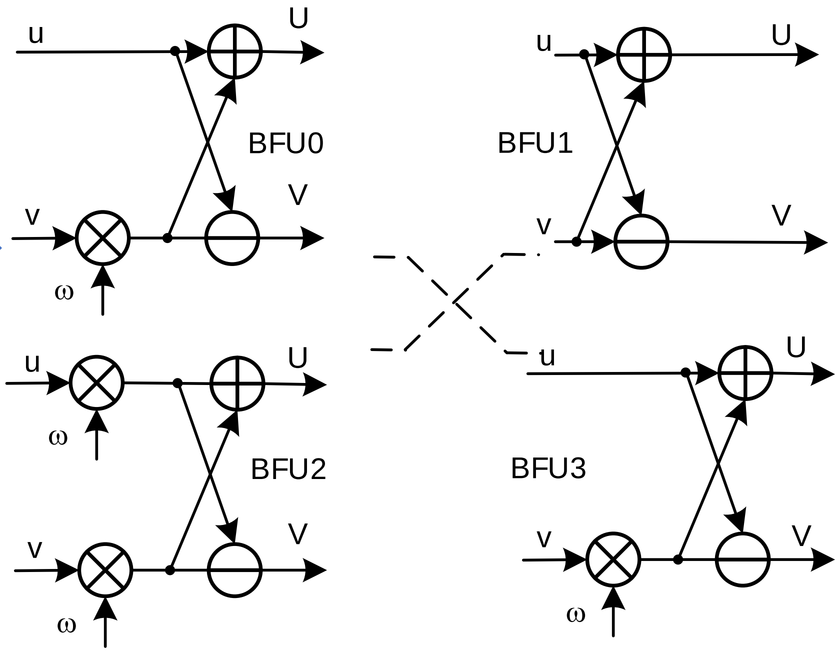
$i \in [0, N/4-1]$ ***Four N/4-point operation***

$$\begin{bmatrix} a_{4i} \\ a_{4i+1} \\ a_{4i+2} \\ a_{4i+3} \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^{-1} & -1 & -\omega_4^{-1} \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^{-1} & -1 & \omega_4^{-1} \end{bmatrix} \times \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} \right) \cdot \begin{bmatrix} 1 \\ \phi_{2N}^{-(2j+1)} \\ \phi_{2N}^{-2(2j+1)} \\ \phi_{2N}^{-3(2j+1)} \end{bmatrix}$$

Recursion → N/4 4-points

Divide and Conquer for Butterfly Operation





Two-layer radix-4 Cooley-Tukey butterfly unit in <u>forward NTT</u>

$$T_0 = (F_0 + F_2 \cdot \phi_{2N}^{2(2i+1)}) \qquad A_i = T_0 + T_2$$

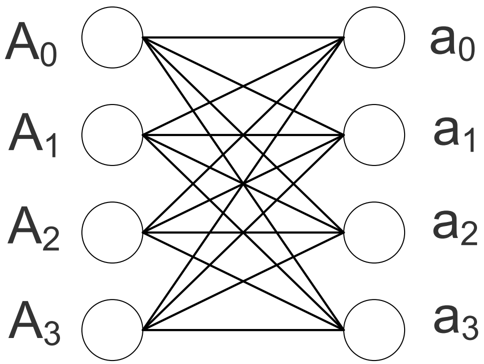
$$T_1 = (F_0 - F_2 \cdot \phi_{2N}^{2(2i+1)}) \qquad A_{i+N/4} = (T_1 + T_3 \cdot \omega_4^1)$$

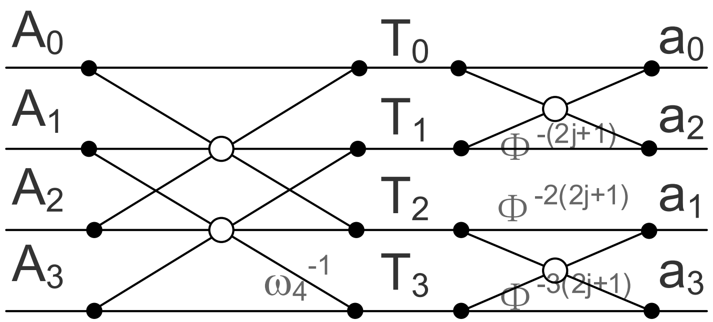
$$T_2 = (F_1 \cdot \phi_{2N}^{2i+1} + F_3 \cdot \phi_{2N}^{3(2i+1)}) \qquad A_{i+2N/4} = (T_0 - T_2)$$

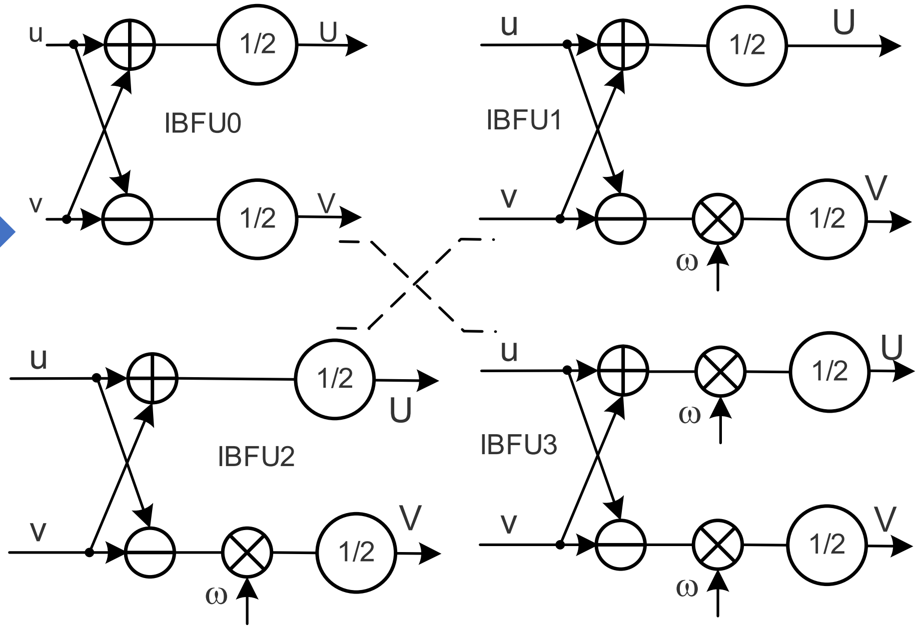
$$T_3 = (F_1 \cdot \phi_{2N}^{2i+1} - F_3 \cdot \phi_{2N}^{3(2i+1)}) \qquad A_{i+3N/4} = (T_1 - T_3 \cdot \omega_4^1)$$

Num. of Op.	MM.	MA.	MS.
Direct	#5	#6	#6
Two-layer	#4	#4	#4
Variation	↓20%	↓33%	↓33%

Divide and Conquer for Butterfly Operation







Two-layer radix-4 Gentleman-Sande butterfly unit in <u>inverse NTT</u>

$$T_0 = F_0 + F_2 \qquad a_{4i} = T_0 + T_2$$

$$T_1 = F_0 - F_2 \qquad a_{4i+1} = (T_1 + T_3) \cdot \phi_{2N}^{-(2j+1)}$$

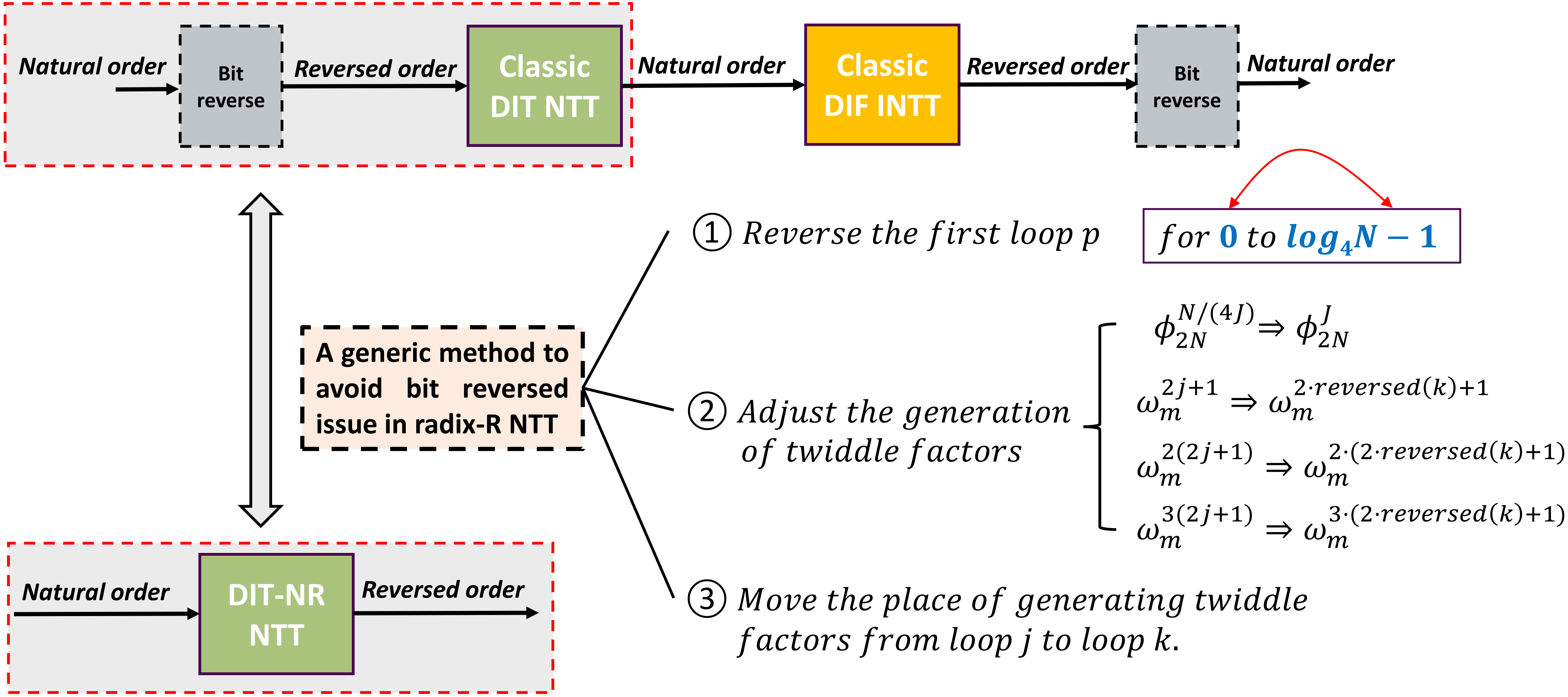
$$T_2 = F_1 + F_3 \qquad a_{4i+2} = (T_0 - T_2) \cdot \phi_{2N}^{-2(2j+1)}$$

$$T_3 = (F_1 - F_3) \cdot \omega_4^{-1} \qquad a_{4i+3} = (T_1 - T_3) \cdot \phi_{2N}^{-3(2j+1)}$$

Num. of Op.	MM.	MA.	MS.
Direct	#5	#6	#6
Two-layer	#4	#4	#4
Variation	↓20%	↓33%	↓33%

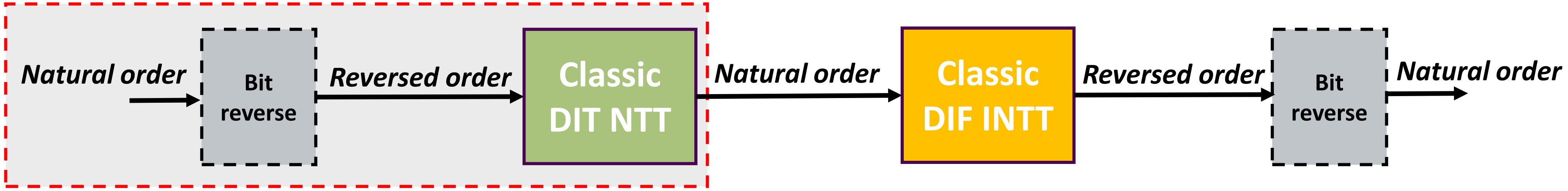
Proposed DIT-NR Radix-4 NTT

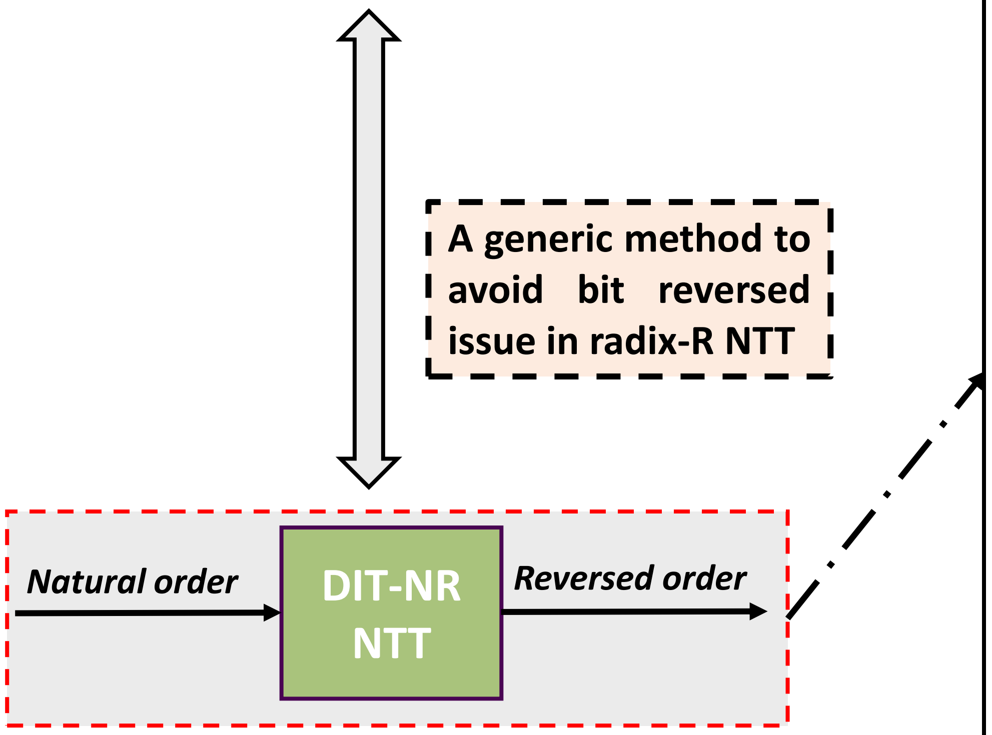
- The bit-reversed operation is needed in classic NTT and INTT operation.



Proposed DIT-NR Radix-4 NTT

- The bit-reversed operation is needed in classic NTT and INTT operation.

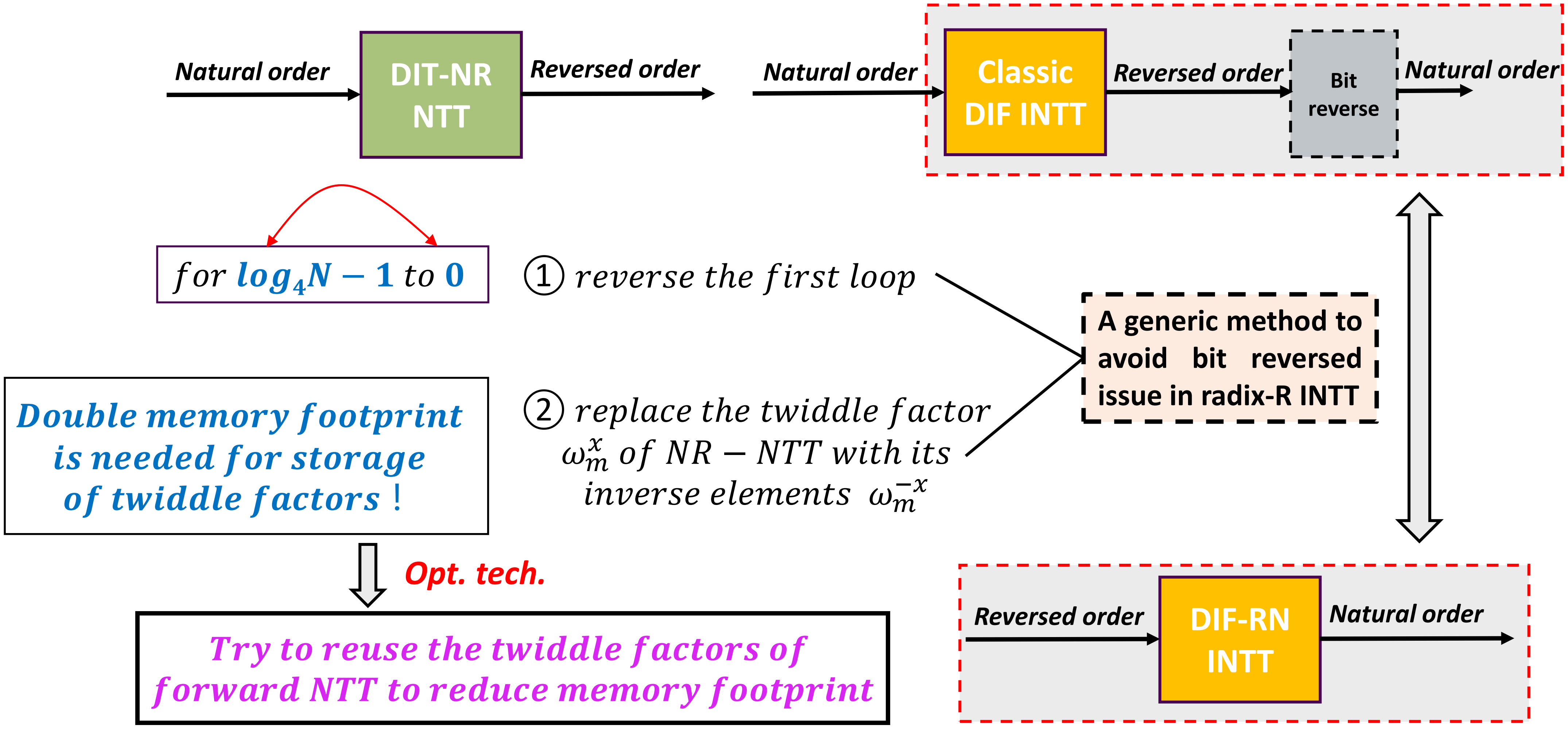




for $p = \log_4 N - 1$ to 0 **do**
  $J \leftarrow 4^p$
  $r \leftarrow 0$
  for $k = 0$ to $N/(4J) - 1$ **do**
    $\omega_1 \leftarrow \omega a1\_\mathrm{ROM}[r]$
    $\omega_2 \leftarrow \omega a2\_\mathrm{ROM}[r]$
    $\omega_3 \leftarrow \omega a3\_\mathrm{ROM}[r]$
    $r \leftarrow r + 1$
    for $j = 0$ to $J - 1$ **do**
      $T_0 \leftarrow (a_{4kJ+j} + a_{4kJ+j+2J} \cdot \omega_2) \bmod q$
      $T_1 \leftarrow (a_{4kJ+j} - a_{4kJ+j+2J} \cdot \omega_2) \bmod q$
      $T_2 \leftarrow (a_{4kJ+j+J} \cdot \omega_1 + a_{4kJ+j+3J} \cdot \omega_3) \bmod q$
      $T_3 \leftarrow (a_{4kJ+j+J} \cdot \omega_1 - a_{4kJ+j+3J} \cdot \omega_3) \bmod q$
      $A_{4kJ+j} \leftarrow (T_0 + T_2) \bmod q$
      $A_{4kJ+j+J} \leftarrow (T_1 + T_3 \cdot \omega_4^1) \bmod q$
      $A_{4kJ+j+2J} \leftarrow (T_0 - T_2) \bmod q$
      $A_{4kJ+j+3J} \leftarrow (T_1 - T_3 \cdot \omega_4^1) \bmod q$
    end for
  end for
end for

Proposed DIF-RN Radix-4 INTT

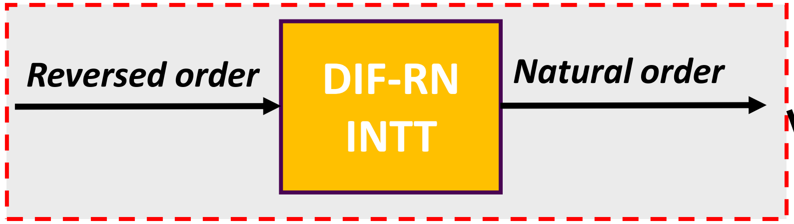
- The bit-reversed operation is needed in classic NTT and INTT operation.



Proposed DIF-RN Radix-4 INTT

- **Opt. tech. 1**: ***Derive three new tricks as following***

$$\begin{cases} \phi_{2N}^{-i} \bmod q = \phi_{2N}^{-N/2} \cdot \phi_{2N}^{N/2-i} = \omega_4^{-1} \cdot \phi_{2N}^{N/2-i} = -\omega_4^{1} \cdot \phi_{2N}^{N/2-i} \bmod q \\ \phi_{2N}^{-i} \bmod q = -\phi_{2N}^{N} \cdot \phi_{2N}^{N-i} = -\phi_{2N}^{N-i} \bmod q \\ \phi_{2N}^{-i} \bmod q = \phi_{2N}^{-3N/2} \cdot \phi_{2N}^{3N/2-i} = \omega_4^{1} \cdot \phi_{2N}^{3N/2-i} \bmod q \end{cases}$$



- **Opt. tech. 2**: ***Apply the derived tricks to obtain the ω_m^{-x} by resuing ω_m^{x}***

$$\omega_m^{-(2j+1)} \Rightarrow -\omega_4^{1} \cdot \omega_m^{N/2-[2\cdot reversed(k)+1]}$$
$$\omega_m^{-2(2j+1)} \Rightarrow -\omega_m^{N-[2\cdot(2\cdot reversed(k)+1)]}$$
$$\omega_m^{-3(2j+1)} \Rightarrow \omega_4^{1} \cdot \omega_m^{3N/2-[3\cdot(2\cdot reversed(k)+1)]}$$

Reuse the twiddle factor of forward NTT

Modify the two-layer radix-4 GS-BFU

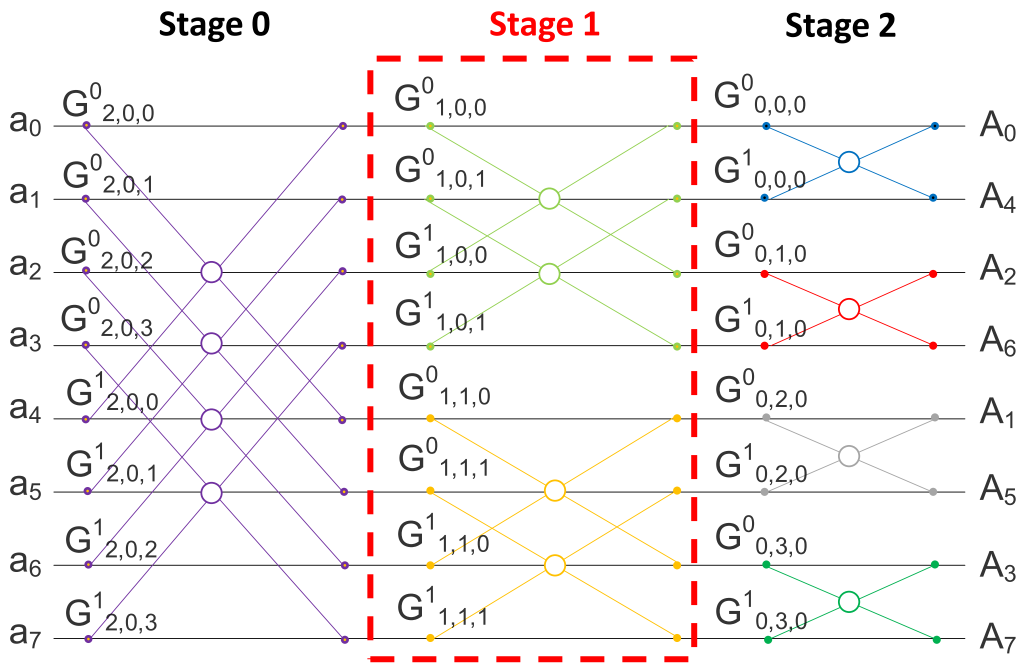
```
for p = log_4 N − 1 to 0 do
  J ← 4^p
  r ← 0
  for k = 0 to N/(4J) − 1 do
    ω1 ← ωa1_ROM [r]
    ω2 ← ωa2_ROM [r]
    ω3 ← ωa3_ROM [r]
    r ← r + 1
    for j = 0 to J − 1 do
      T0 ← (a_{4kJ+j} + a_{4kJ+j+2J} · ω2) mod q
      T1 ← (a_{4kJ+j} − a_{4kJ+j+2J} · ω2) mod q
      T2 ← (a_{4kJ+j+J} · ω1 + a_{4kJ+j+3J} · ω3) mod q
      T3 ← (a_{4kJ+j+J} · ω1 − a_{4kJ+j+3J} · ω3) mod q
      A_{4kJ+j} ← (T0 + T2) mod q
      A_{4kJ+j+J} ← (T1 + T3 · ω4^1) mod q
      A_{4kJ+j+2J} ← (T0 − T2) mod q
      A_{4kJ+j+3J} ← (T1 − T3 · ω4^1) mod q
    end for
  end for
end for
```

↓ 50% memory footprint of twiddle factors

Outlines

- Introduction
- Optimized Radix-4 NTT/INTT Algorithm
- **Conflict-free Memory Mapping Scheme**
- Hardware Architecture of Radix-2/4 NTT
- Implementation Result and Comparison

Determine the Point-access Order



The data flow of 8 points radix-2 DIT-NR NTT.

- $G^{i}_{p,k,j}$: The i–th point in stage p, group k and round j where i = 0,1,…,R−1 and R denotes the radix
- The four data points at stage 1 can be parallelly accessed in two types of order as below:

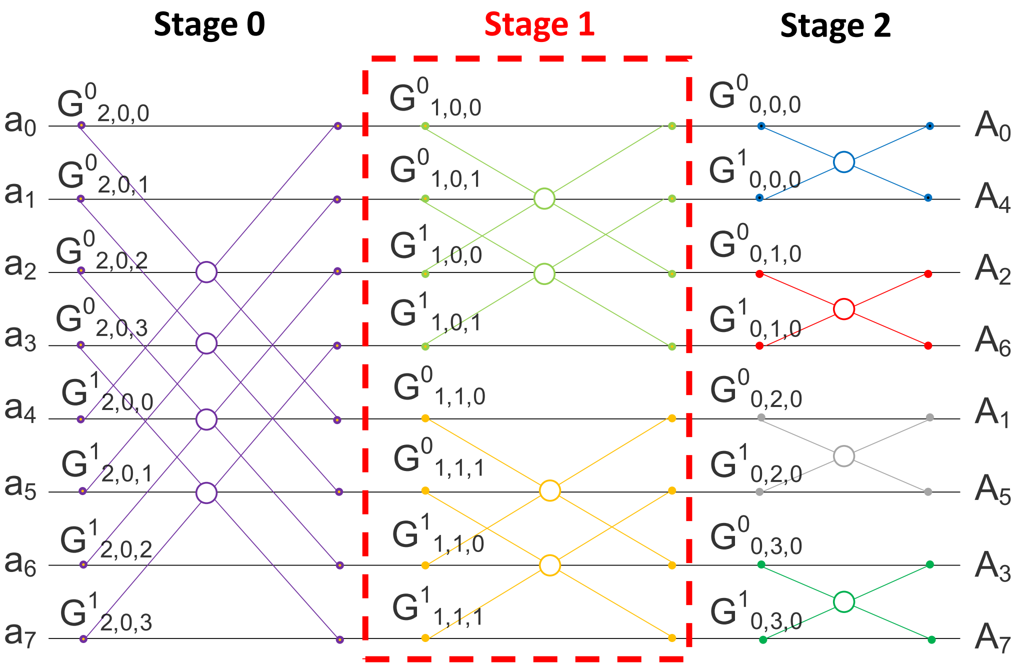
① *Four data points with the same group index k*

$$\{G^0_{1,0,0}, G^1_{1,0,0}, G^0_{1,0,1}, G^1_{1,0,1}\} \rightarrow \{G^0_{1,1,0}, G^1_{1,1,0}, G^0_{1,1,1}, G^1_{1,1,1}\}$$

② *Four data points with the same round index j*

$$\{G^0_{1,0,0}, G^1_{1,0,0}, G^0_{1,1,0}, G^1_{1,1,0}\} \rightarrow \{G^0_{1,0,1}, G^1_{1,0,1}, G^0_{1,1,1}, G^1_{1,1,1}\}$$

Determine the Point-access Order



The data flow of 8 points radix-2 DIT-NR NTT.

- $G^i_{p,k,j}$: The i–th point in stage p, group k and round j

  where i = 0,1,…,R−1 and R denotes the radix

- The four data points at stage 1 can be parallelly accessed in two types of order as below:

① *Four data points with the same group index k*

$$\{G^0_{1,0,0}, G^1_{1,0,0}, G^0_{1,0,1}, G^1_{1,0,1}\} \rightarrow \{G^0_{1,1,0}, G^1_{1,1,0}, G^0_{1,1,1}, G^1_{1,1,1}\}$$

② *Four data points with the same round index j*

$$\{G^0_{1,0,0}, G^1_{1,0,0}, G^0_{1,1,0}, G^1_{1,1,0}\} \rightarrow \{G^0_{1,0,1}, G^1_{1,0,1}, G^0_{1,1,1}, G^1_{1,1,1}\}$$

Proposed Parallel NTT Algorithm with Arbitrary Radix

Tsinghua University

Algorithm 6 Scalable Iterative NTT Algorithm

Input: $\boldsymbol{a}$, N, R. Here $\boldsymbol{a}$ denotes a vector of length N. d is the number of butterfly units. R denotes the radix of NTT.

Output: $\boldsymbol{A} = Scalable\_NTT(\boldsymbol{a})$

for $p = \log_R N - 1$ to 0 **do**
  $J \leftarrow R^p$
  if $J < d$ **then**
    for $k = 0$ to $N/(Rd) - 1$ **do**
      for $i = 0$ to $d/J - 1$ **do**
        for $j = 0$ to $J - 1$ **do**

$$\begin{bmatrix} A_0 \\ A_1 \\ \cdots \\ A_{R-1} \end{bmatrix} = BF\left(\begin{bmatrix} a_{kRd+iRJ+j} \\ a_{kRd+iRJ+j+J} \\ \cdots \\ a_{kRd+iRJ+j+(R-1)J} \end{bmatrix}\right)$$

        end for
      end for
    end for
  else
    for $k = 0$ to $N/(RJ) - 1$ **do**
      for $i = 0$ to $J/d - 1$ **do**
        for $j = 0$ to $d - 1$ **do**

$$\begin{bmatrix} A_0 \\ A_1 \\ \cdots \\ A_{R-1} \end{bmatrix} = BF\left(\begin{bmatrix} a_{kRJ+id+j} \\ a_{kRJ+id+j+J} \\ \cdots \\ a_{kRJ+id+j+(R-1)J} \end{bmatrix}\right)$$

        end for
      end for
    end for
  end if
end for
return $\boldsymbol{A}$

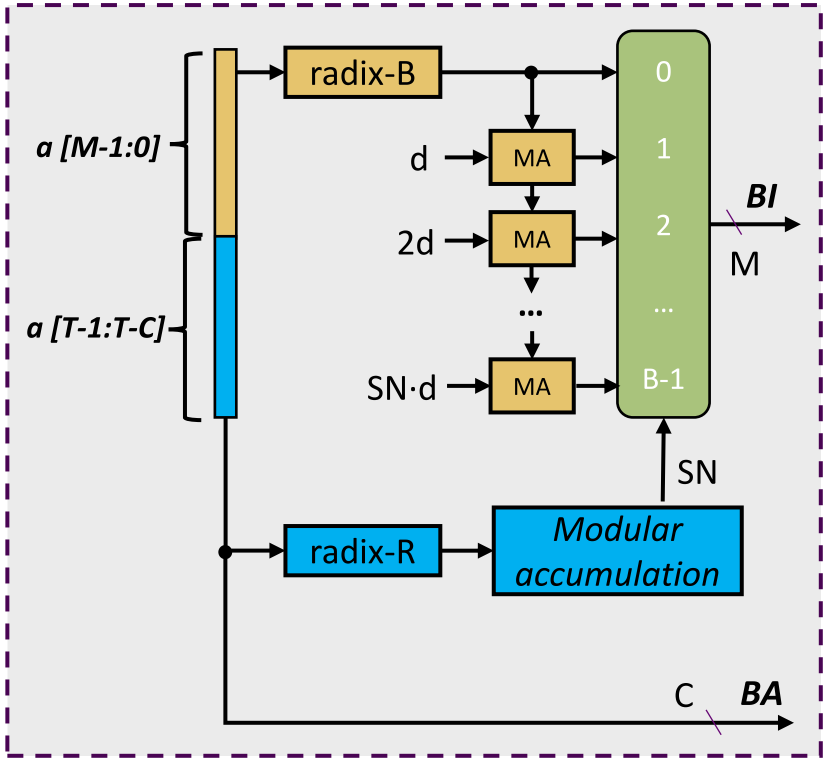
Loop Tile 1

Loop Tile 2

- The relative size of d and J will influence the number of iterations.
  - $J < d$, the parallel data sets will contain several iterations of inner loop. (Loop Tile 1)
  - $J \geq d$, the inner loop will cover several parallel data sets. (Loop Tile 2)

Conflict-free Memory Mapping for Arbitrary Radix NTT

Tsinghua University



Memory mapper for arbitrary radix r^k

- d: *Number of BFUs*;
- B: *Number of banks*, $B = R \times d, N \mid B$;
- $T = log_r N$; $M = log_r B$; $C = T - M$;

- ✓ Automated configuration
- ✓ Initialized for different radix $R = r^k$
- ✓ Facilitate other stride access
- ✓ We further extend this method in our paper at DAC 2022*

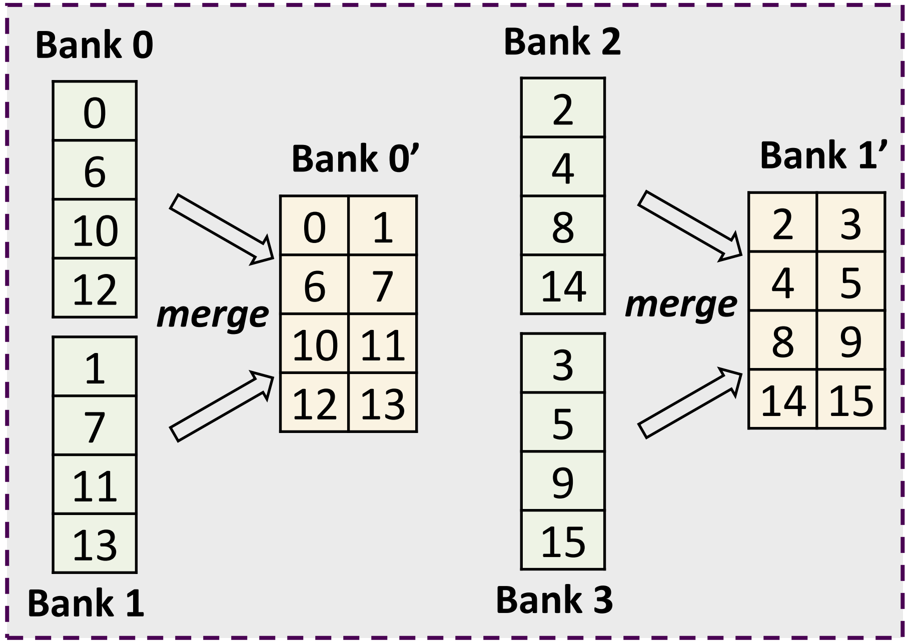
*Xiangren Chen, Bohan Yang, …, Leibo Liu. 2022. Efficient access scheme for multi-bank based NTT architecture through conflict graph. (DAC '22).

Case Study of Conflict-free Mapping

Parameter	N = 16	R = 2	d = 2	B = 4

old address	mix radix R-B	step number	slide distance	bank index		bank address
0	000	(0+0) mod 2 = 0	(0x2) mod 4 = 0	(0+0) mod 4 = 0	$G_{2,0,0}$ √	00
1	001			(0+1) mod 4 = 1		
2	002			(0+2) mod 4 = 2		
3	003			(0+3) mod 4 = 3		
4	010	(0+1) mod 2 = 1	(1x2) mod 4 = 2	(2+0) mod 4 = 2	$G_{3,0,0}$ √	01
5	011			(2+1) mod 4 = 3		
6	012			(2+2) mod 4 = 0		
7	013			(2+3) mod 4 = 1		
8	100	(1+0) mod 2 = 1	(1x2) mod 4 = 2	(2+0) mod 4 = 2	$G_{2,1,0}$ √	10
9	101			(2+1) mod 4 = 3		
10	102			(2+2) mod 4 = 0		
11	103			(2+3) mod 4 = 1		
12	110	(1+0) mod 2 = 0	(0x2) mod 4 = 0	(0+0) mod 4 = 0		11
13	111			(0+1) mod 4 = 1		
14	112			(0+2) mod 4 = 2		
15	113			(0+3) mod 4 = 3		

Case study of 16-point radix-2 in-place NTT with 2 BFUs

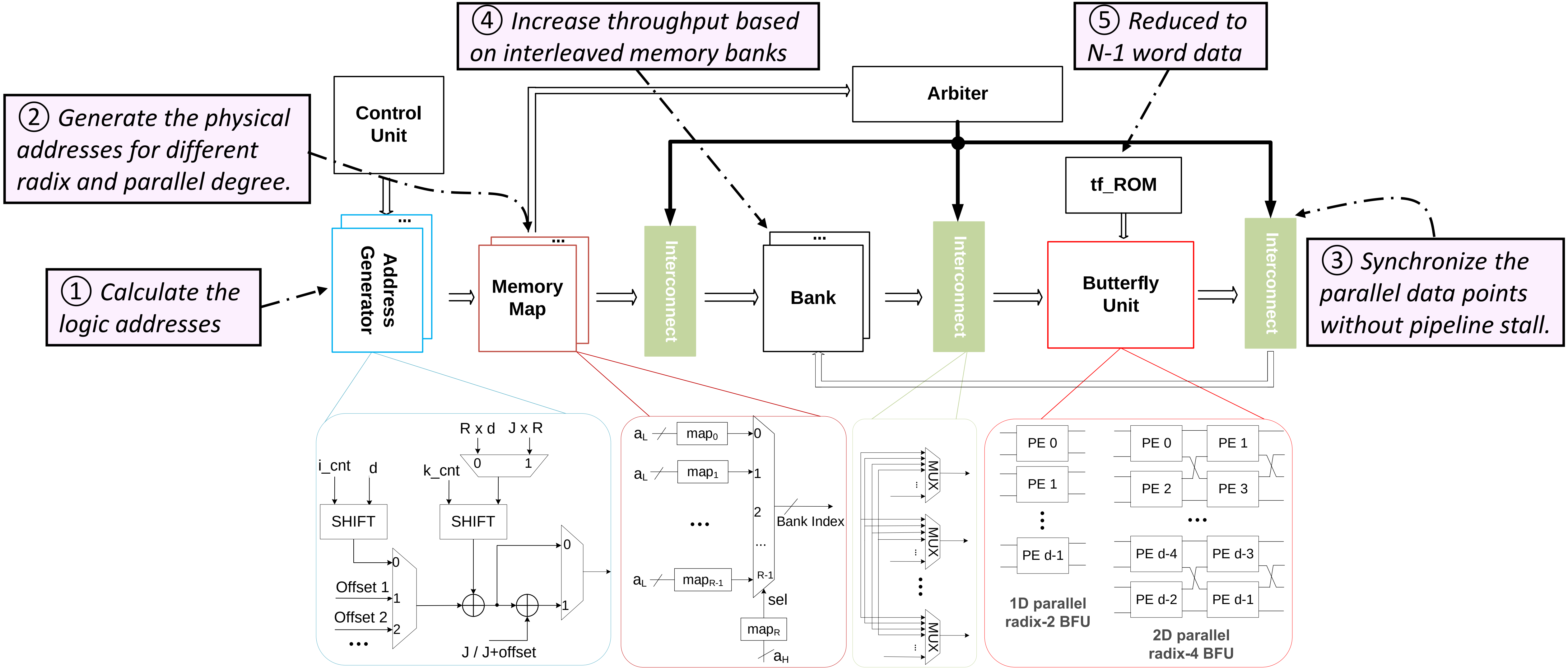


Reduce 4 banks to 2 banks

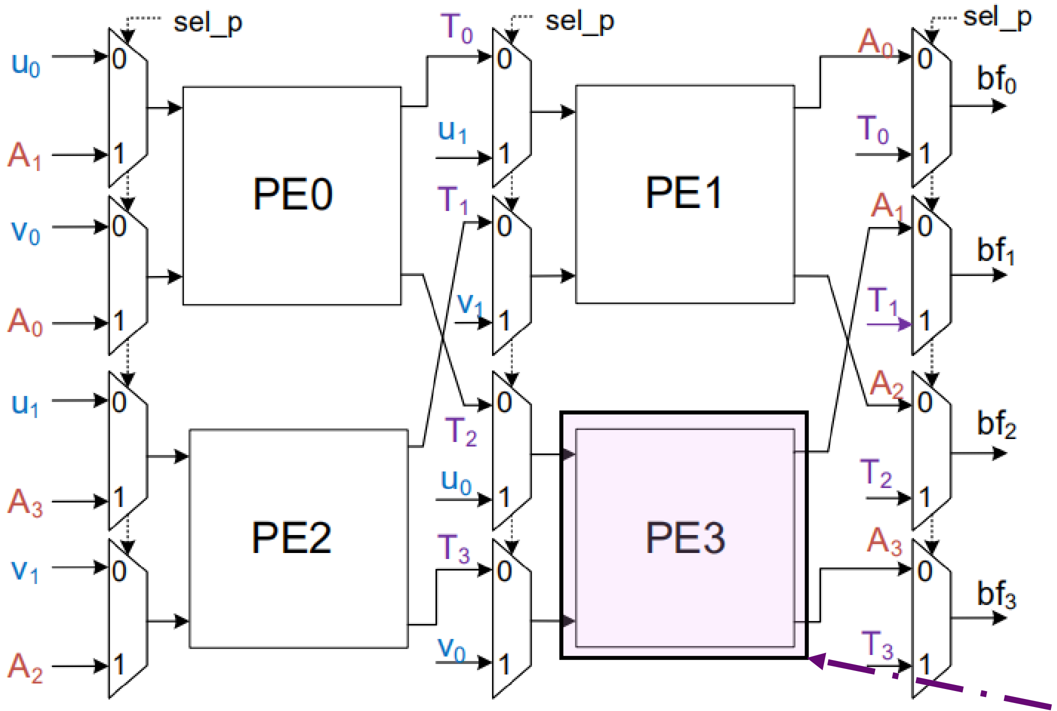
Outlines

- Introduction
- Optimized Radix-4 NTT/INTT Algorithm
- Conflict-free Memory Mapping Scheme
- **Hardware Architecture of Radix-2/4 NTT**
- Implementation Result and Comparison

Hardware Architecture of Radix-2/4 NTT



Unified Design of Radix-4 Butterfly Unit



The routing structure of radix-4 PE array.

PE types	#MM	#MA	#MS	#Half
PE0	1	1/2	1/2	2
PE1	0	1/2	1/2	2
PE2	2	1/2	1/2	2
PE3	1	1/2	1/2	2
Total	**4**	**4/8**	**4/8**	**8**

- ✓ sel_p = 0/1 perform NTT/INTT
- ✓ Barrett reduction based modular multiplier
- ✓ *The multiplication by ω_4^1 is a constant modular multiplier within PE3.*

Outlines

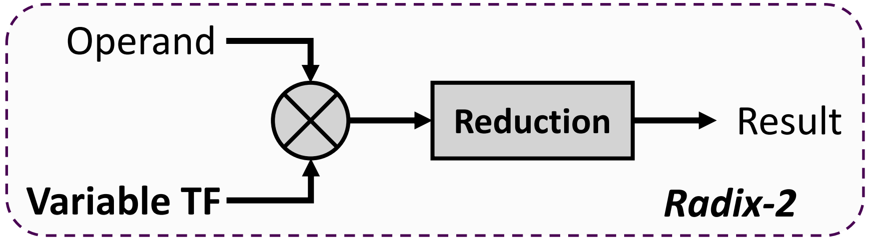
- Introduction
- Optimized Radix-4 NTT/INTT Algorithm
- Conflict-free Memory Mapping Scheme
- Hardware Architecture of Radix-2/4 NTT
- **Implementation Result and Comparison**

Implementation Result and Comparison

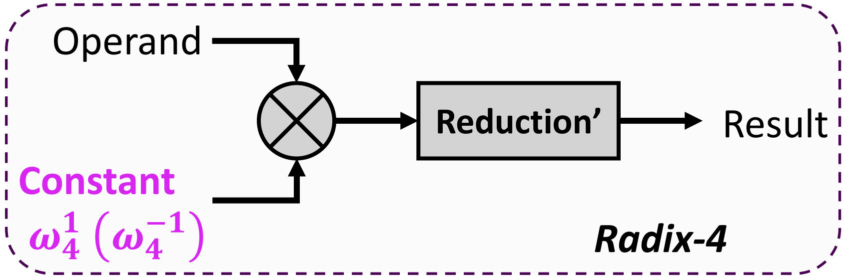
☐ Theoretical Evaluation Between Radix-4 and Radix-2 **Multi-bank** based NTT

Type	***AGU*** *Address Generator*	***MMU*** *Memory Map*	***AB*** *Arbiter*	***INN*** *Interconnect Network*	***Banks***	***BFU*** *Butterfly Unit*	***TF ROM*** *Twiddle Factor*	***NTT/INTT cycles***
Radix − 2	$2n$	$2n \times$ $2-to-1$ $MUXs$	$2n \times$ $2-to-1$ $MUXs$	$2n \times$ $2-to-1$ $MUXs$	$2n$	$n \times$ $MA/MS/MM$	$N-1$	$(N/2d)\log_2 N$
Radix − 4	n	$n \times$ $4-to-1$ $MUXs$	$n \times$ $4-to-1$ $MUXs$	$n \times$ $4-to-1$ $MUXs$	n	$n \times$ $MA/MS/MM^*$	$N-1$	$(N/d)\log_4 N$
Variation	↓ 50%	—	↓ 75%	↓ 75%	↓ 50%	—	—	—

n: *number of processing element*



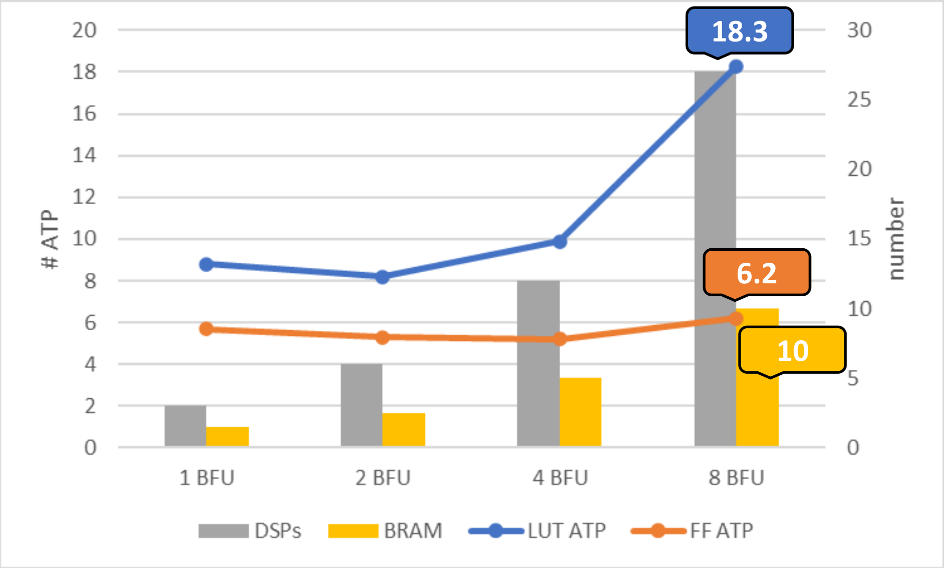
↓ Mult. Complexity



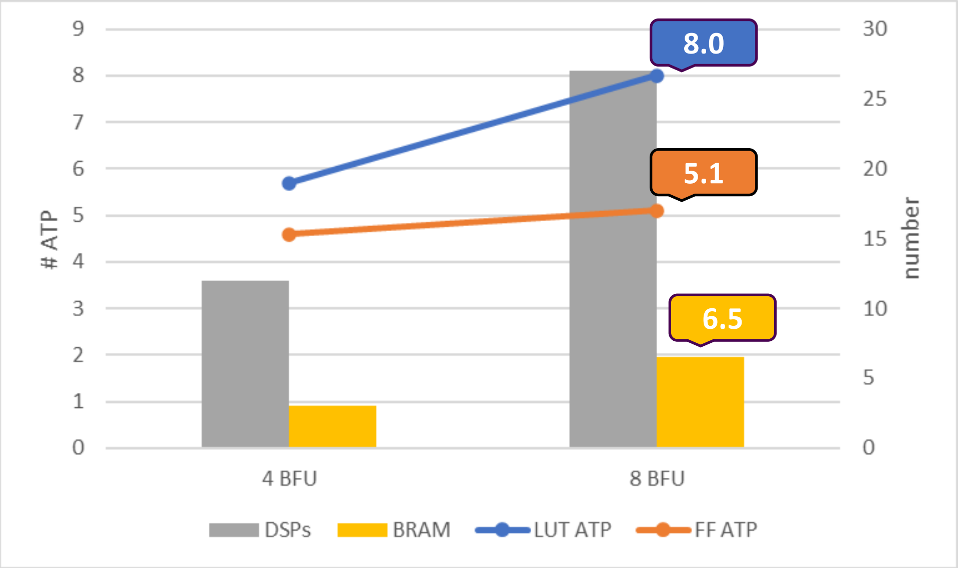
Implementation Result and Comparison

- **Implementation platform**: Vivado 2020.2 + Virtex-7 FPGA (xc7vx690tffffg1761-3)

- **Benefits of radix-4** : e.g. #BFU = 8 ⇒ **LUT ATP ↓ 2.2 × FF ATP ↓ 1.2 × #BRAMs ↓ 1.5**



(a) radix-2 NTT



(b) radix-4 NTT

The variation of ATP in radix-2 and radix-4 NTT (N = 1024, 14-bit q)

Implementation Result and Comparison

The comparison between radix-4 NTT core and other works.

Thank You！

Q & A

Reference code: https://github.com/xiang-rc/cfntt_ref