Title: ReMCA: A Reconfigurable Multi-Core Architecture for Full RNS Variant of BFV Homomorphic Evaluation

(TCAS I)

Preliminaries

The Textbook BFV Scheme

- BFV.KeyGen(λ, ω)
- BFV.Enc(m, pk)
- BFV.Dec(ct, sk)
- BFV.HomAdd(ct_0, ct_1)
- BFV.HomMult(ct_0, ct_1, rlk)

Preliminaries

Parameter Setup

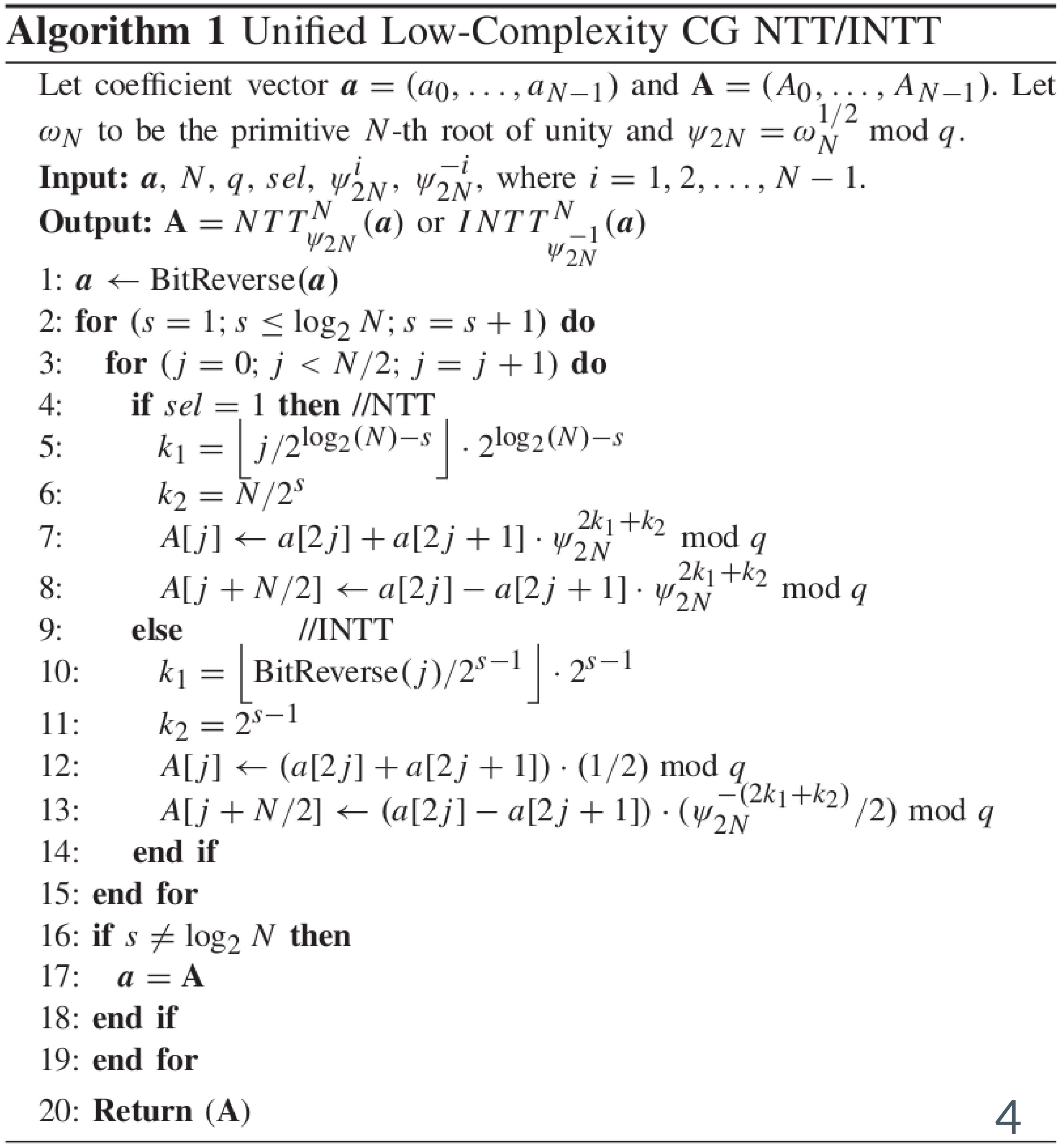
- polynomial degree: 4096
- the standard deviation of the Gaussian distribution to σ: 3.19
- the size of modulus q: 128 bit(product of four 32 bit primes)?
- the size of the larger modulus Q to at least 288-bit(product of nine 32 bit primes)
- 32-bit primes to construct the RNS for our implementation

Algorithm And Approach

Unified Low-Complexity NTT/INTT

It is a algorithm can control the butterfly unit to do NTT or INTT, decrease the complexity of memory access.

$(\frac{N}{2}log_2^N + N) + (\frac{N}{2}log_2^N + 2N) -> (Nlog_2^N + N)$



Algorithm 1 Unified Low-Complexity CG NTT/INTT

Let coefficient vector $\boldsymbol{a} = (a_0, \ldots, a_{N-1})$ and $\mathbf{A} = (A_0, \ldots, A_{N-1})$. Let ω_N to be the primitive N-th root of unity and $\psi_{2N} = \omega_N^{1/2} \bmod q$.

Input: $\boldsymbol{a}$, N, q, sel, ψ_{2N}^{i}, ψ_{2N}^{-i}, where $i = 1, 2, \ldots, N-1$.

Output: $\mathbf{A} = NTT_{\psi_{2N}}^{N}(\boldsymbol{a})$ or $INTT_{\psi_{2N}^{-1}}^{N}(\boldsymbol{a})$

```
1: $\boldsymbol{a} \leftarrow \text{BitReverse}(\boldsymbol{a})$
2: for ($s = 1; s \le \log_2 N; s = s + 1$) do
3:   for ($j = 0; j < N/2; j = j + 1$) do
4:     if $sel = 1$ then //NTT
5:       $k_1 = \lfloor j/2^{\log_2(N)-s} \rfloor \cdot 2^{\log_2(N)-s}$
6:       $k_2 = N/2^s$
7:       $A[j] \leftarrow a[2j] + a[2j+1] \cdot \psi_{2N}^{2k_1+k_2} \bmod q$
8:       $A[j+N/2] \leftarrow a[2j] - a[2j+1] \cdot \psi_{2N}^{2k_1+k_2} \bmod q$
9:     else //INTT
10:      $k_1 = \lfloor \text{BitReverse}(j)/2^{s-1} \rfloor \cdot 2^{s-1}$
11:      $k_2 = 2^{s-1}$
12:      $A[j] \leftarrow (a[2j] + a[2j+1]) \cdot (1/2) \bmod q$
13:      $A[j+N/2] \leftarrow (a[2j] - a[2j+1]) \cdot (\psi_{2N}^{-(2k_1+k_2)}/2) \bmod q$
14:    end if
15:  end for
16:  if $s \ne \log_2 N$ then
17:    $\boldsymbol{a} = \mathbf{A}$
18:  end if
19: end for
20: Return ($\mathbf{A}$)
```

Algorithm And Approach

RNS Basis Extension

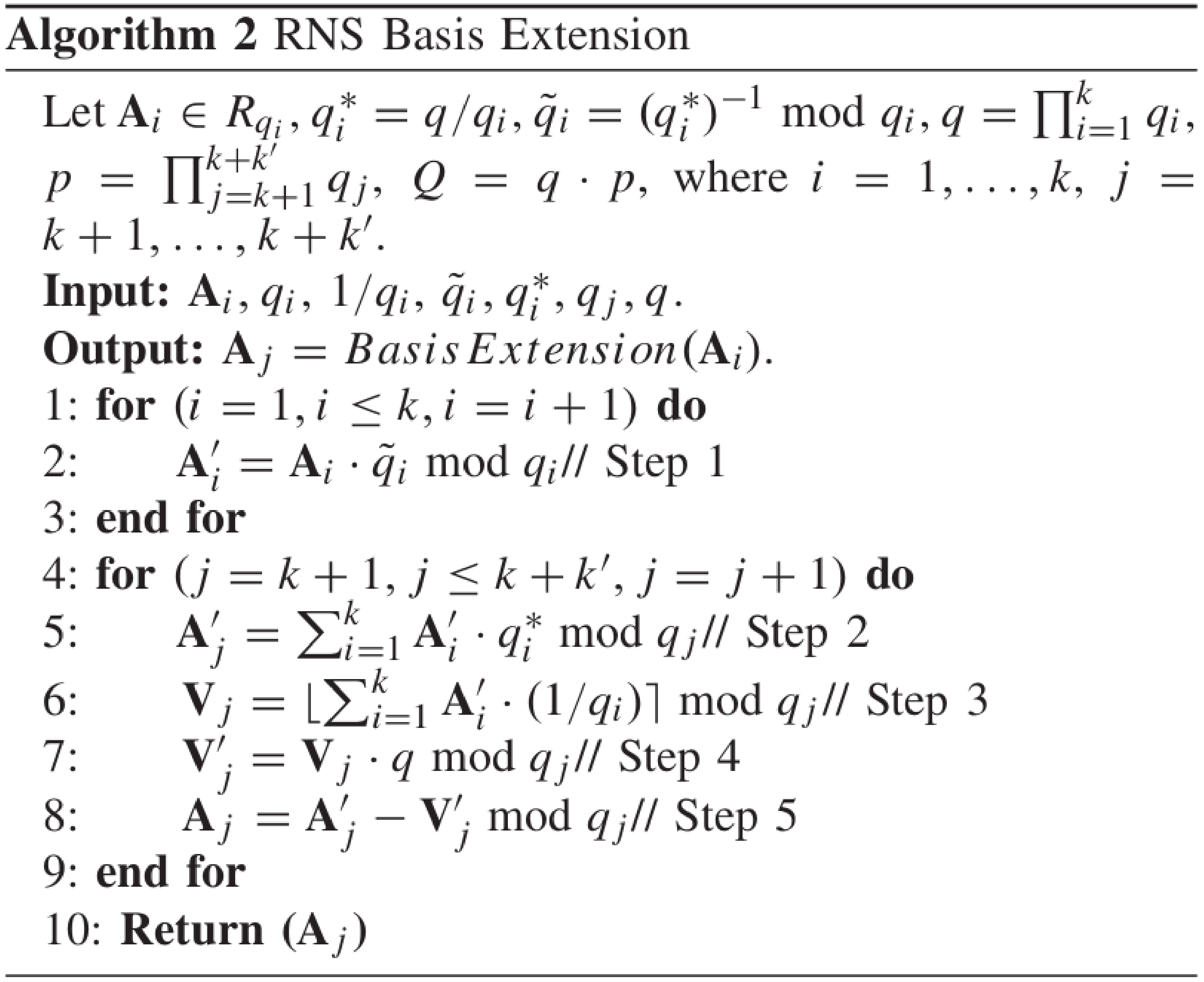
Chinese remainder theorem(https://zh.wikipedia.org/zh-hans/中国剩余定理)

$$q = \prod_{i=1}^{k} q_i, p = \prod_{j=k+1}^{k+k'} q_j, Q = p * q$$

$$where \quad k = 4 \quad k' = 5$$

$$A_j \equiv (\sum(A_i * \tilde{q}_i mod q_i) * q_i^* - V' * q) mod q_j$$

$$V' = \lfloor \sum(A_i * \tilde{q}_i mod q_i^*)/q_i \rceil$$



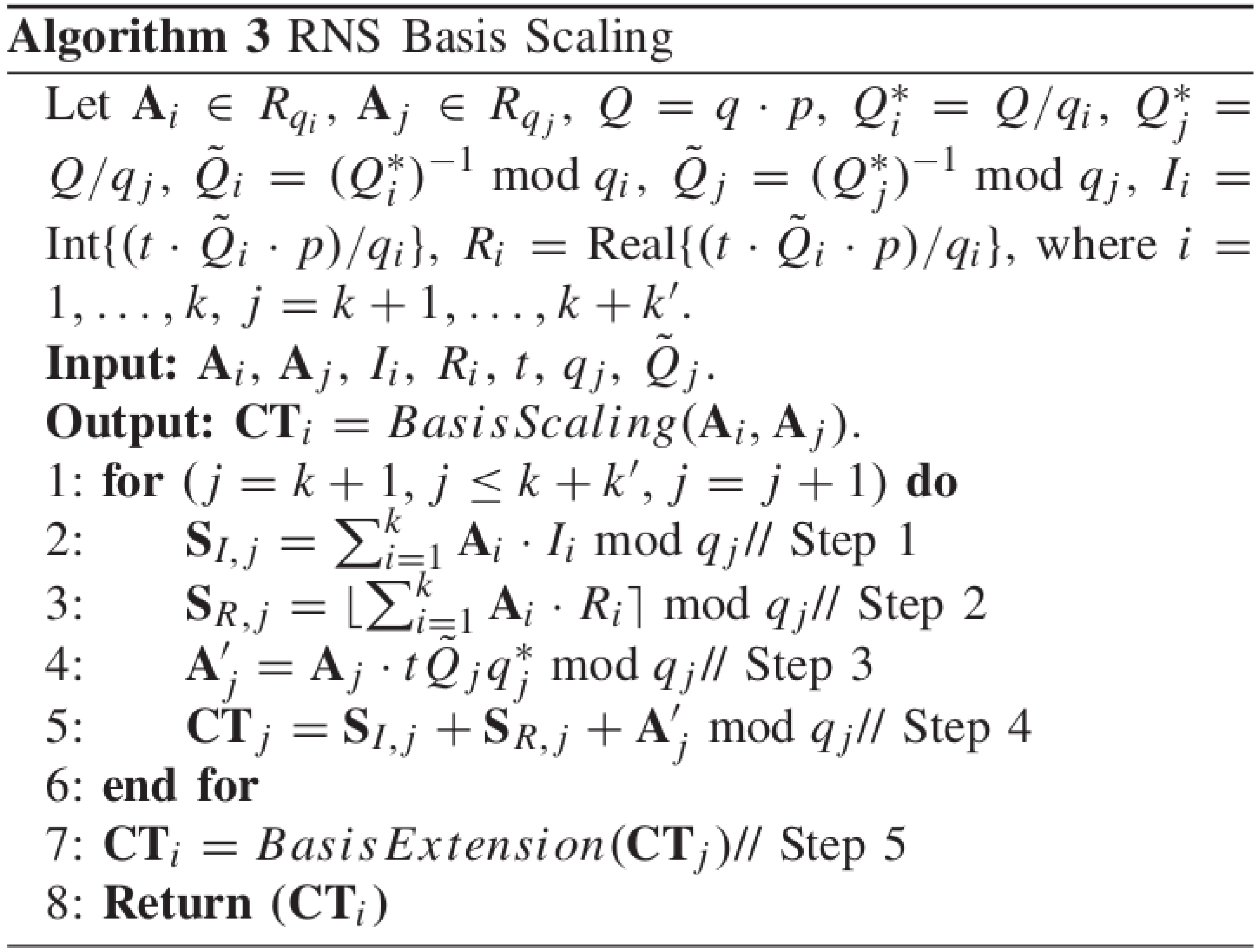
Algorithm 2 RNS Basis Extension

```
Let $\mathbf{A}_i \in R_{q_i}, q_i^* = q/q_i, \tilde{q}_i = (q_i^*)^{-1} \bmod q_i, q = \prod_{i=1}^{k} q_i,$
$p = \prod_{j=k+1}^{k+k'} q_j, Q = q \cdot p,$ where $i = 1, \ldots, k, j = k+1, \ldots, k+k'.$
**Input:** $\mathbf{A}_i, q_i, 1/q_i, \tilde{q}_i, q_i^*, q_j, q.$
**Output:** $\mathbf{A}_j = BasisExtension(\mathbf{A}_i).$
1: **for** $(i = 1, i \leq k, i = i + 1)$ **do**
2:     $\mathbf{A}_i' = \mathbf{A}_i \cdot \tilde{q}_i \bmod q_i$// Step 1
3: **end for**
4: **for** $(j = k + 1, j \leq k + k', j = j + 1)$ **do**
5:     $\mathbf{A}_j' = \sum_{i=1}^{k} \mathbf{A}_i' \cdot q_i^* \bmod q_j$// Step 2
6:     $\mathbf{V}_j = \lfloor \sum_{i=1}^{k} \mathbf{A}_i' \cdot (1/q_i) \rceil \bmod q_j$// Step 3
7:     $\mathbf{V}_j' = \mathbf{V}_j \cdot q \bmod q_j$// Step 4
8:     $\mathbf{A}_j = \mathbf{A}_j' - \mathbf{V}_j' \bmod q_j$// Step 5
9: **end for**
10: **Return** $(\mathbf{A}_j)$
```

Algorithm And Approach

RNS Basis

After complete the RNS basis extension, the ciphertext multiplication is performed in the RNS of R Q in parallel. Then,the extended tensored ciphertexts need to be scaled down



```
Algorithm 3 RNS Basis Scaling
Let $\mathbf{A}_i \in R_{q_i}$, $\mathbf{A}_j \in R_{q_j}$, $Q = q \cdot p$, $Q_i^* = Q/q_i$, $Q_j^* = Q/q_j$, $\tilde{Q}_i = (Q_i^*)^{-1} \bmod q_i$, $\tilde{Q}_j = (Q_j^*)^{-1} \bmod q_j$, $I_i = \mathrm{Int}\{(t \cdot \tilde{Q}_i \cdot p)/q_i\}$, $R_i = \mathrm{Real}\{(t \cdot \tilde{Q}_i \cdot p)/q_i\}$, where $i = 1, \ldots, k$, $j = k+1, \ldots, k+k'$.
Input: $\mathbf{A}_i, \mathbf{A}_j, I_i, R_i, t, q_j, \tilde{Q}_j$.
Output: $\mathbf{CT}_i = BasisScaling(\mathbf{A}_i, \mathbf{A}_j)$.
1: for $(j = k+1, j \le k+k', j = j+1)$ do
2:     $\mathbf{S}_{I,j} = \sum_{i=1}^{k} \mathbf{A}_i \cdot I_i \bmod q_j$// Step 1
3:     $\mathbf{S}_{R,j} = \lfloor \sum_{i=1}^{k} \mathbf{A}_i \cdot R_i \rceil \bmod q_j$// Step 2
4:     $\mathbf{A}'_j = \mathbf{A}_j \cdot t\tilde{Q}_j q_j^* \bmod q_j$// Step 3
5:     $\mathbf{CT}_j = \mathbf{S}_{I,j} + \mathbf{S}_{R,j} + \mathbf{A}'_j \bmod q_j$// Step 4
6: end for
7: $\mathbf{CT}_i = BasisExtension(\mathbf{CT}_j)$// Step 5
8: Return $(\mathbf{CT}_i)$
```

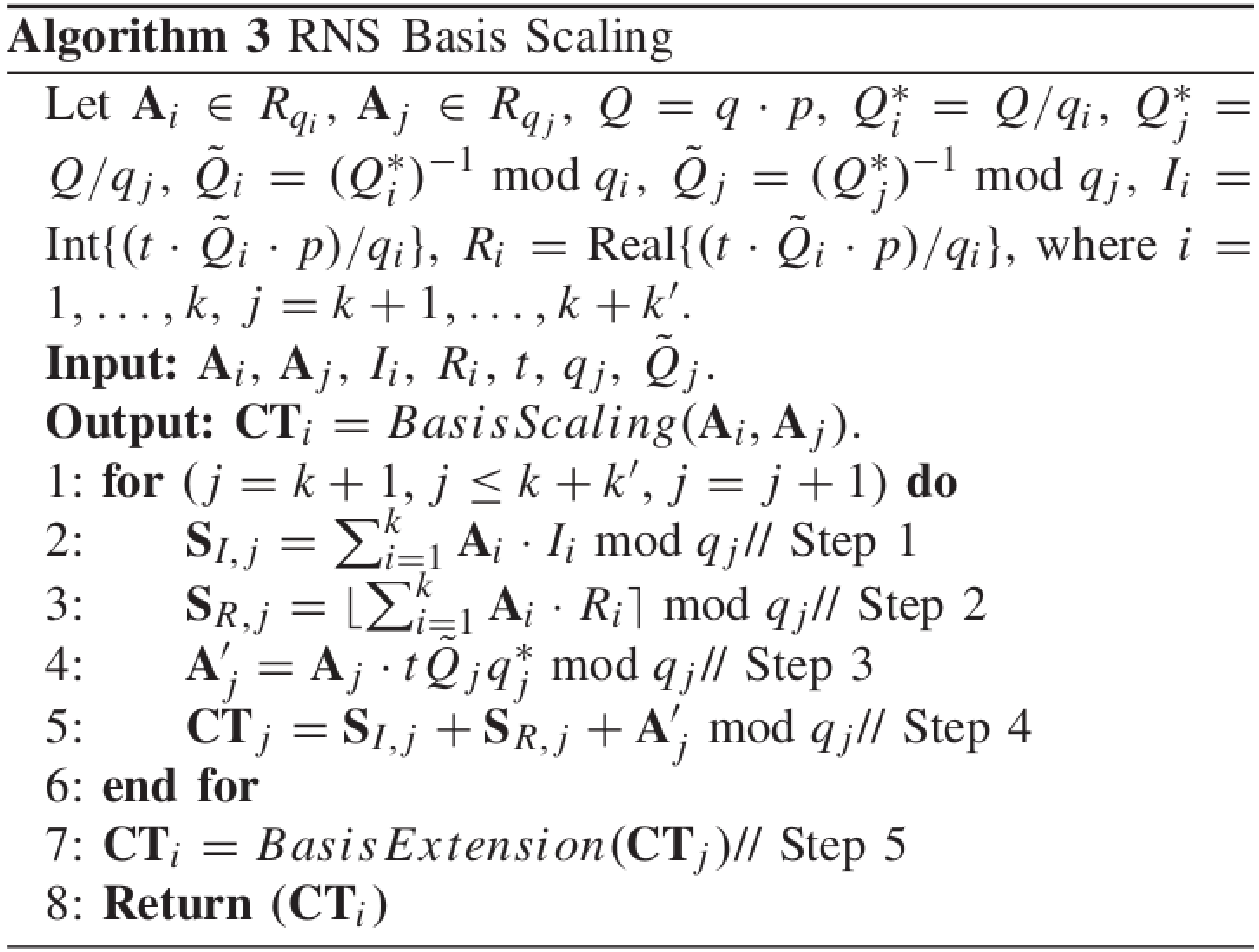
Algorithm And Approach

RNS Basis

$$\lfloor t/q * A_i \rceil mod q_j =$$

$$\lfloor \sum A_i * (t\tilde{Q}_i P/q_i)$$

$$+A_j * (t\tilde{Q}_i q_j^*) mod q_j \rceil mod q_j$$



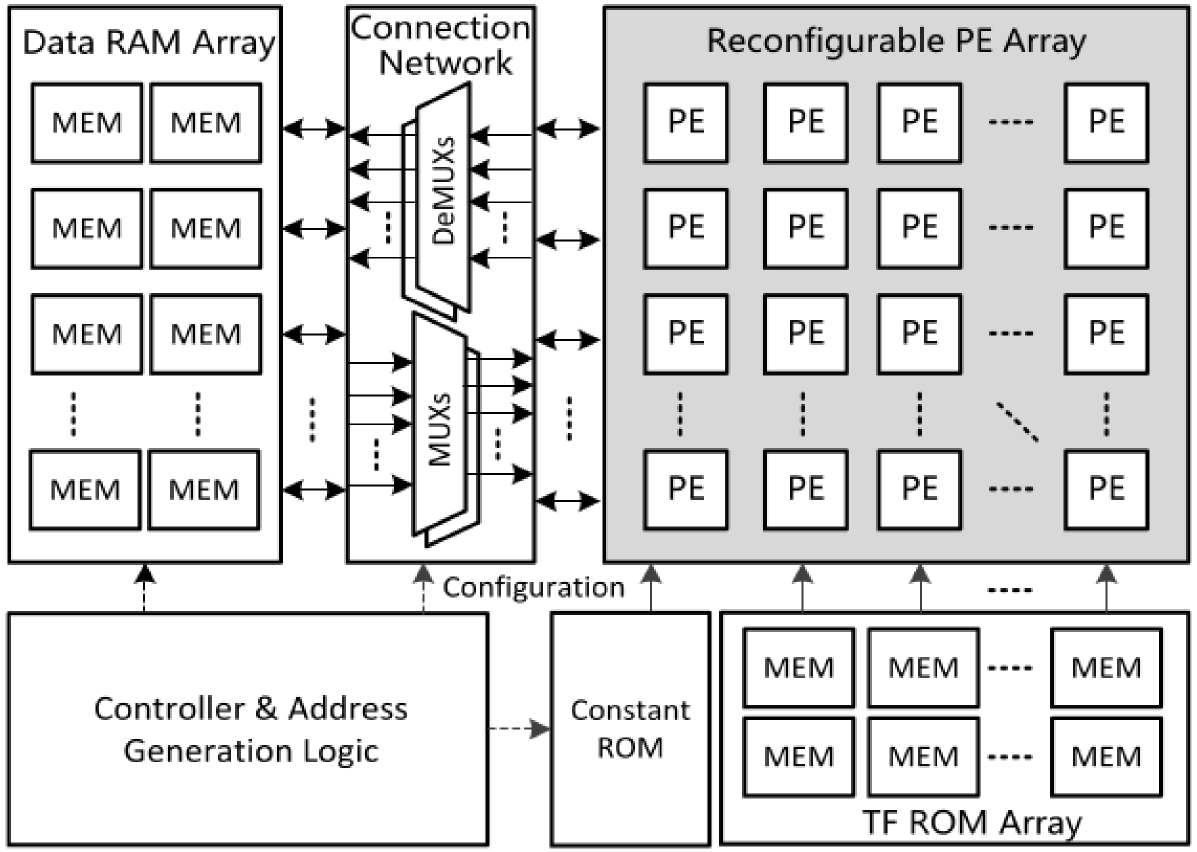
Algorithm 3 RNS Basis Scaling

```
Let $\mathbf{A}_i \in R_{q_i}$, $\mathbf{A}_j \in R_{q_j}$, $Q = q \cdot p$, $Q_i^* = Q/q_i$, $Q_j^* = Q/q_j$, $\tilde{Q}_i = (Q_i^*)^{-1} \bmod q_i$, $\tilde{Q}_j = (Q_j^*)^{-1} \bmod q_j$, $I_i = \mathrm{Int}\{(t \cdot \tilde{Q}_i \cdot p)/q_i\}$, $R_i = \mathrm{Real}\{(t \cdot \tilde{Q}_i \cdot p)/q_i\}$, where $i = 1, \ldots, k$, $j = k+1, \ldots, k+k'$.
Input: $\mathbf{A}_i, \mathbf{A}_j, I_i, R_i, t, q_j, \tilde{Q}_j$.
Output: $\mathbf{CT}_i = BasisScaling(\mathbf{A}_i, \mathbf{A}_j)$.
1: for ($j = k+1, j \leq k+k', j = j+1$) do
2:     $\mathbf{S}_{I,j} = \sum_{i=1}^{k} \mathbf{A}_i \cdot I_i \bmod q_j$// Step 1
3:     $\mathbf{S}_{R,j} = \lfloor \sum_{i=1}^{k} \mathbf{A}_i \cdot R_i \rceil \bmod q_j$// Step 2
4:     $\mathbf{A}'_j = \mathbf{A}_j \cdot t\tilde{Q}_j q_j^* \bmod q_j$// Step 3
5:     $\mathbf{CT}_j = \mathbf{S}_{I,j} + \mathbf{S}_{R,j} + \mathbf{A}'_j \bmod q_j$// Step 4
6: end for
7: $\mathbf{CT}_i = BasisExtension(\mathbf{CT}_j)$// Step 5
8: Return ($\mathbf{CT}_i$)
```

Architecture

Overall Architecture

Reconfigurable PE Array: perform NTT/INTT, the modular multplication
TF ROM Array: store the twiddle factor array
Data RAM Array: store the input polynomials, intermediate results and final results



One row or multiple rows of PE can be configured as a channel to perform the polynomial arithmetic operations on one RNS base.

Total of 40 PEs, in which each row corresponds to one channel and each channel contains two slices and 8 PEs (4 for each slice). To maximize the parallelism of the processing path and match the number of extended RNS bases, set five channels in PE array.

Architecture

Reconfigurable PE Unit

Reconfigurable PE: INTT,NTT,MULT
Barrett modular multiplier: modular multiplier, modular reduction

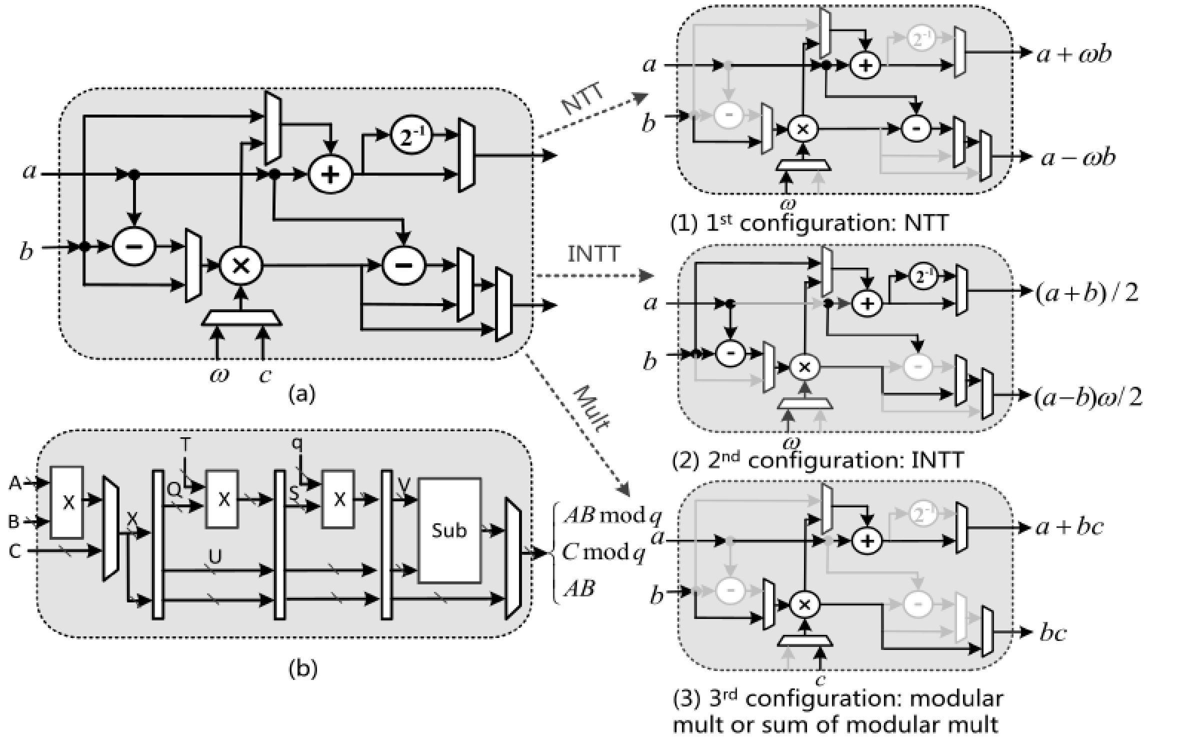


Fig. 2. Architecture of reconfigurable PE. (a) Reconfigurable PE, (b) Barrett modular multiplier.

1.PE not only supports the functions with the variable modulus, but also supports the summation of modular multiplication
2.By merging the multiplicative factor 1/2 into the twiddle factors, the reconfigurable PE eliminates the multiplication of 1/2 in the subtraction path and improves the performance of PE unit.
$\frac{x}{2} = (2\lfloor\frac{x}{2}\rfloor + 1)\frac{q+1}{2} = \lfloor\frac{x}{2}\rfloor(q+1) + \frac{q+1}{2} = \lfloor\frac{x}{2}\rfloor + \frac{q+1}{2}(modq)$?
3.The Barrett modular multiplier we presented employs a reconfigurable architecture and avoids the needs of other computing units for ReMCA.

Architecture

Confilct-Free Memory Access for NTT/INTT

Bank is dual-port pattern(could select the bank read port based on PEs)
For the bit-reversal operation, could change the address mapping pattern to avoid timing-consuming or memory-consuming.

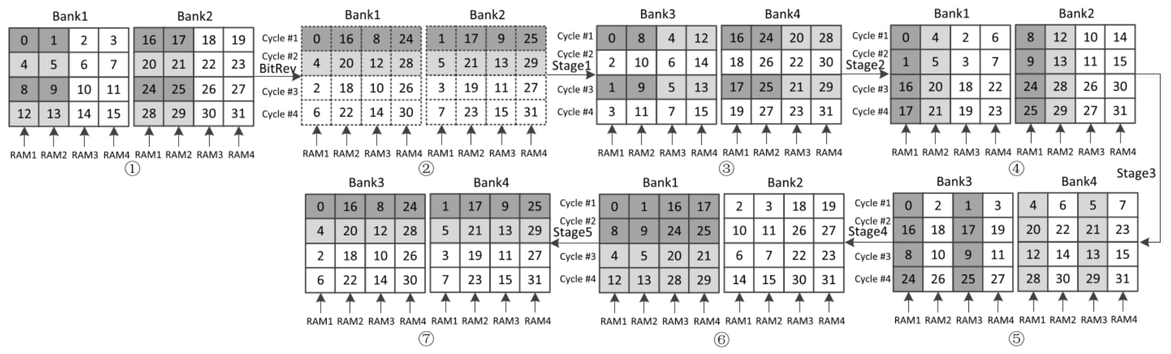


Fig. 3. Data memory access pattern of NTT for $N = 32$, $P = 4$.

Architecture

Unified Computing Model

Unified Hardware Architecture Mapping Model:

- model 1: compute 32 bits modular mult of four contiguous integers in vector A_i nad vector B_i or four constants in parallel.

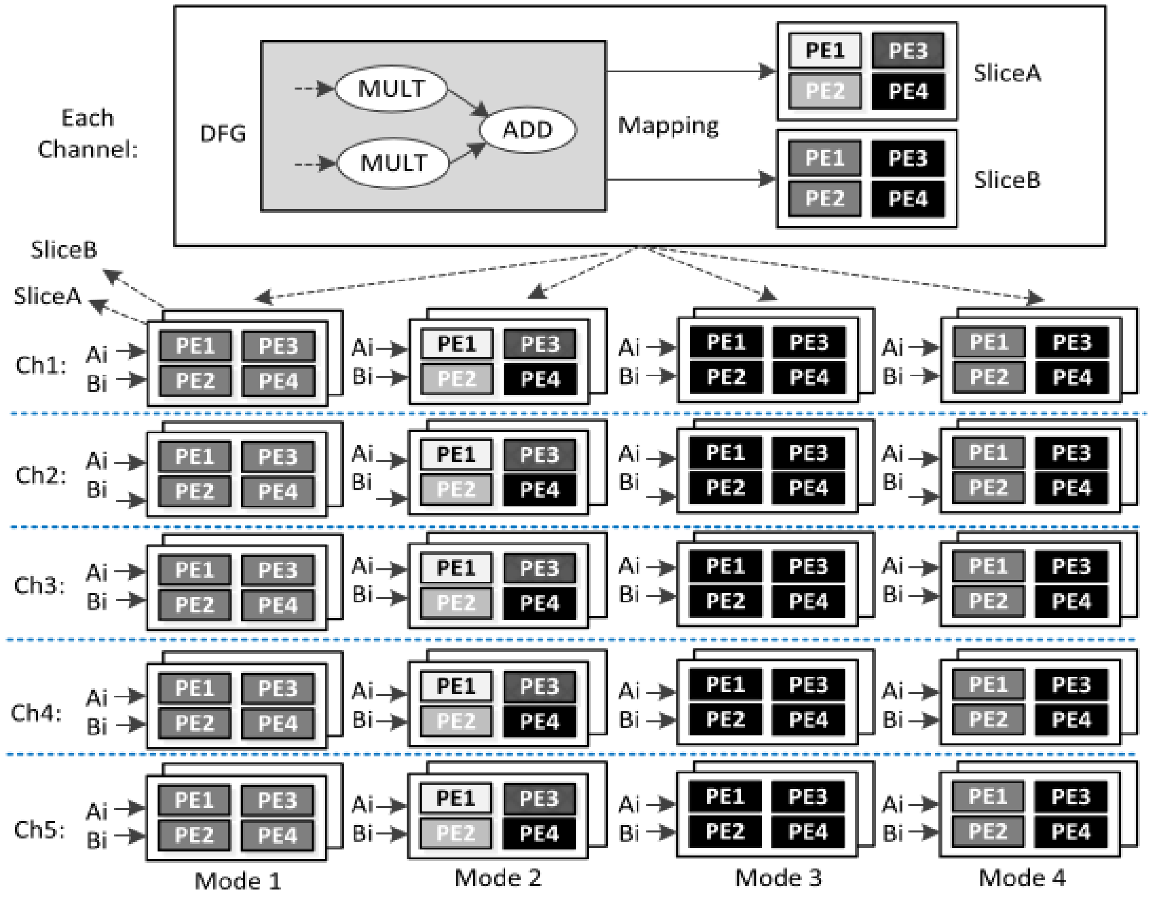


Fig. 4. Unified hardware architecture mapping model of ReMCA.

- model 2: compute the summation of four products, while the inputs of four products are from four different vectors and four constants respectively(A2 S2;A3 S1)

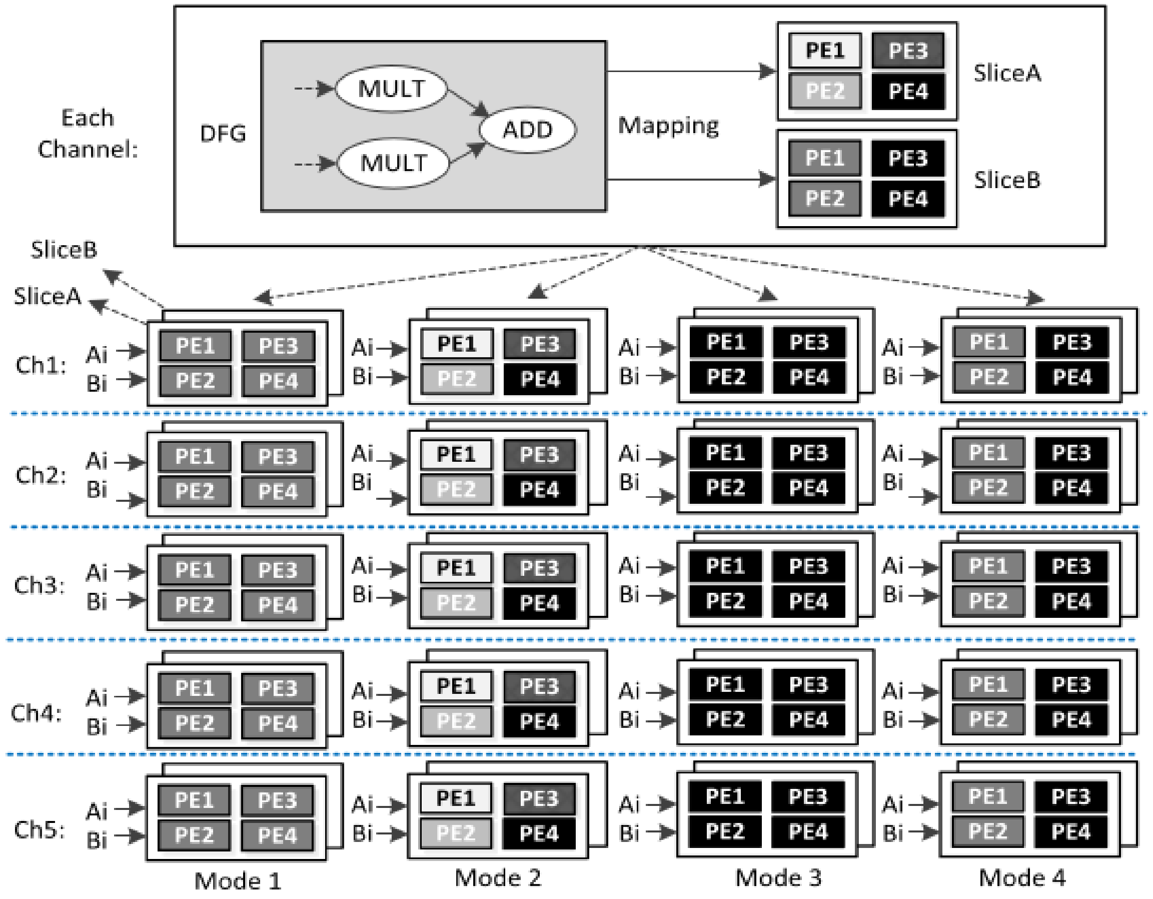


Fig. 4. Unified hardware architecture mapping model of ReMCA.

- model 3: The NTT/INTT transforms are computed using Mode 3
- model 4: requires the cooperation of first four channels.Compute the summation of four products followed by a rounding operation
  (A2 S3)

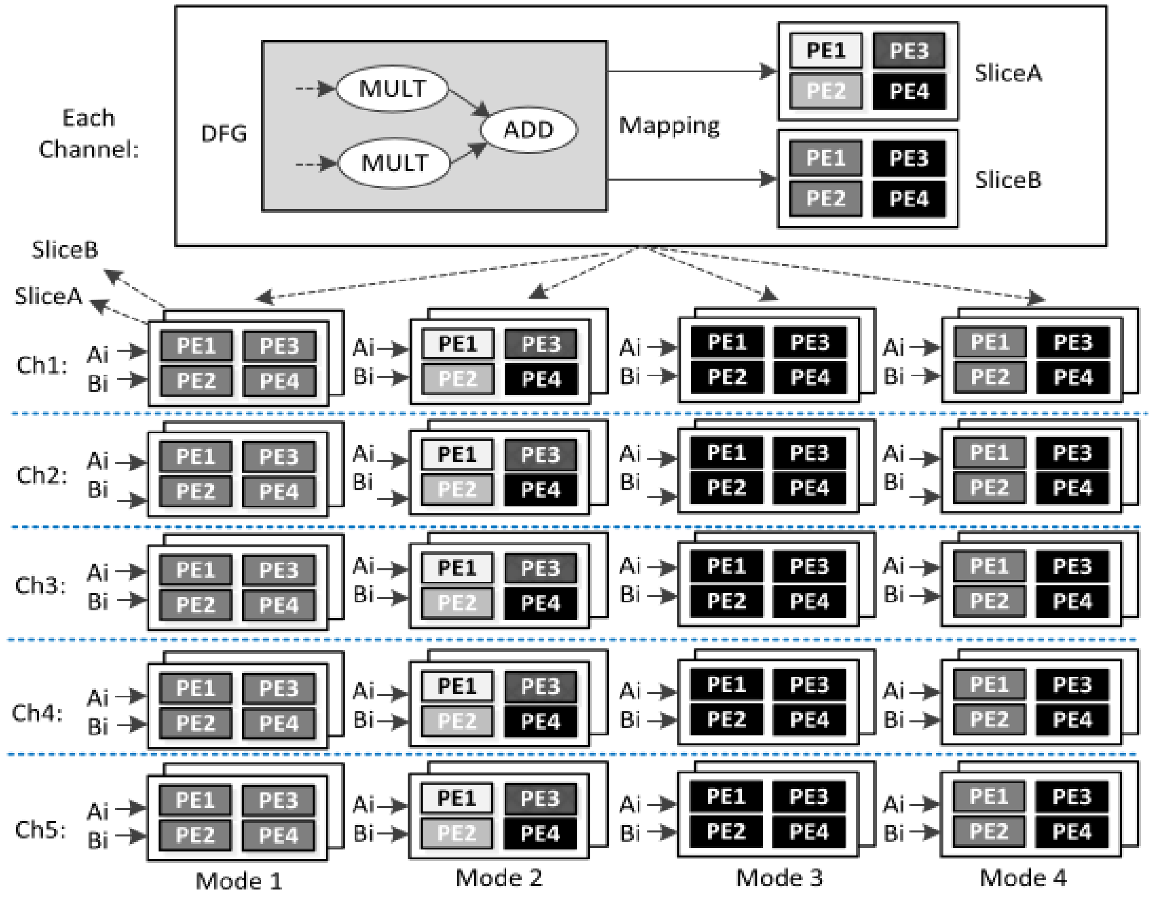


Fig. 4. Unified hardware architecture mapping model of ReMCA.

Architecture

Unified Computing Model

Unified Data Memory Organization Model:

MEM consists of four memory banks, where each memory bank further contains four 1024-depth and 32-bit-width dual-port RAMs.

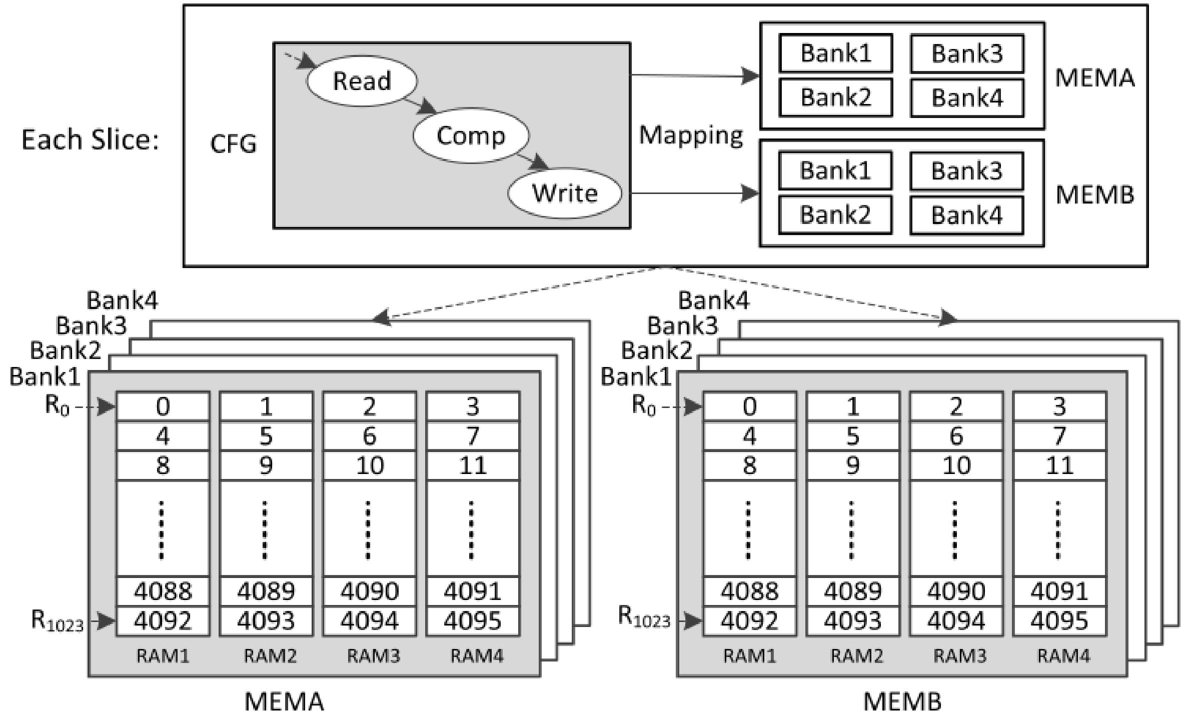


Fig. 5. Unified data memory organization model of ReMCA.

- MEMA is used to store the inputs/outputs and intermediates results of almost all functional units in homomorphic evaluation of RNS-BFV except for the NTT and INTT.
- MEMB is mainly used to store the inputs/outputs and intermediate results of NTT and INTT(MEMB is enough?4096*2?)

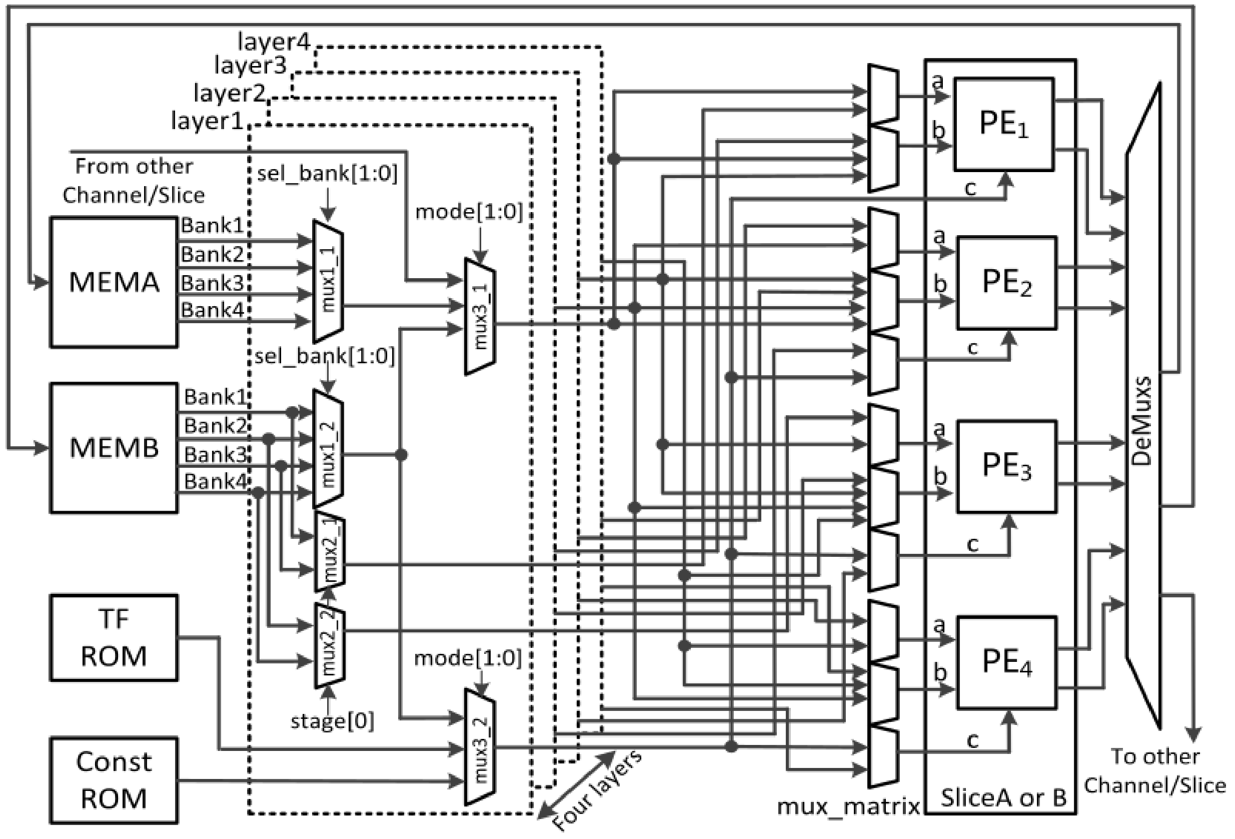


Fig. 6. Detailed structure of interconnection network of each slice.

Mapping Method And Execution Flow

The homomorphic multiplication of RNS-BFV includes four computing units: basis extension, ciphertext multiplication, basis scaling and relinearization

Mapping Method And Execution Flow

Computing Units Mapping

Basis Extension Unit

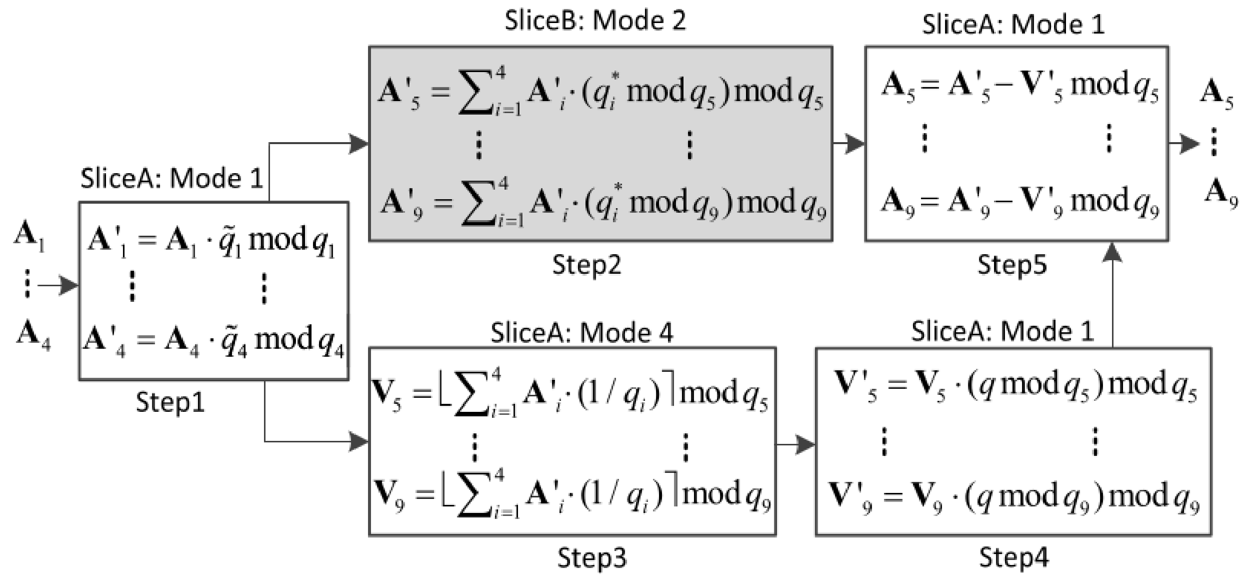

SliceA: Mode 1
$\mathbf{A}_1 \vdots \mathbf{A}_4$
$\mathbf{A'}_1 = \mathbf{A}_1 \cdot \tilde{q}_1 \bmod q_1$ ⋮ $\mathbf{A'}_4 = \mathbf{A}_4 \cdot \tilde{q}_4 \bmod q_4$
Step1
SliceB: Mode 2
$\mathbf{A'}_5 = \sum_{i=1}^{4} \mathbf{A'}_i \cdot (q_i^* \bmod q_5) \bmod q_5$ ⋮ $\mathbf{A'}_9 = \sum_{i=1}^{4} \mathbf{A'}_i \cdot (q_i^* \bmod q_9) \bmod q_9$
Step2
SliceA: Mode 4
$\mathbf{V}_5 = \lfloor \sum_{i=1}^{4} \mathbf{A'}_i \cdot (1/q_i) \rceil \bmod q_5$ ⋮ $\mathbf{V}_9 = \lfloor \sum_{i=1}^{4} \mathbf{A'}_i \cdot (1/q_i) \rceil \bmod q_9$
Step3
SliceA: Mode 1
$\mathbf{V'}_5 = \mathbf{V}_5 \cdot (q \bmod q_5) \bmod q_5$ ⋮ $\mathbf{V'}_9 = \mathbf{V}_9 \cdot (q \bmod q_9) \bmod q_9$
Step4
SliceA: Mode 1
$\mathbf{A}_5 = \mathbf{A'}_5 - \mathbf{V'}_5 \bmod q_5$ ⋮ $\mathbf{A}_9 = \mathbf{A'}_9 - \mathbf{V'}_9 \bmod q_9$
Step5
$\mathbf{A}_5 \vdots \mathbf{A}_9$

Fig. 7. Mapping method of basis extension unit.

Mapping Method And Execution Flow

Computing Units Mapping

Ciphertext Multiplication Unit

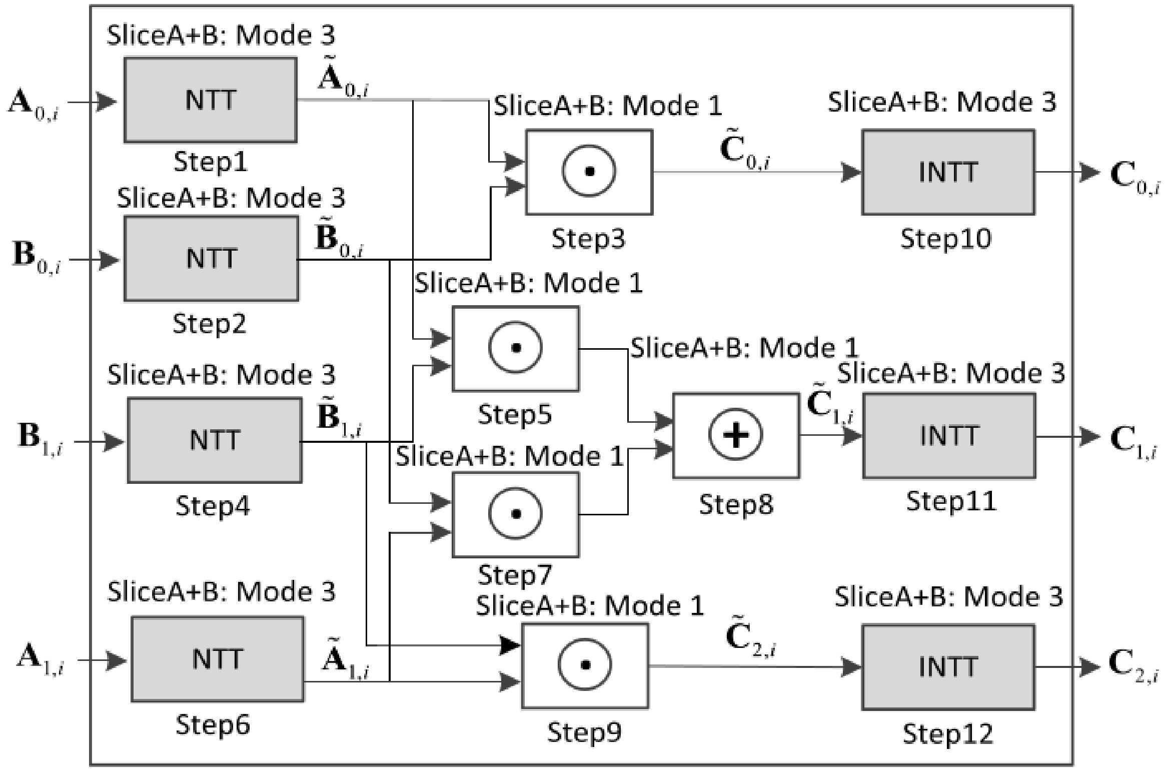


Fig. 8. Mapping method of ciphertext multiplication unit.

Mapping Method And Execution Flow

Computing Units Mapping

Basis Scaling Unit:

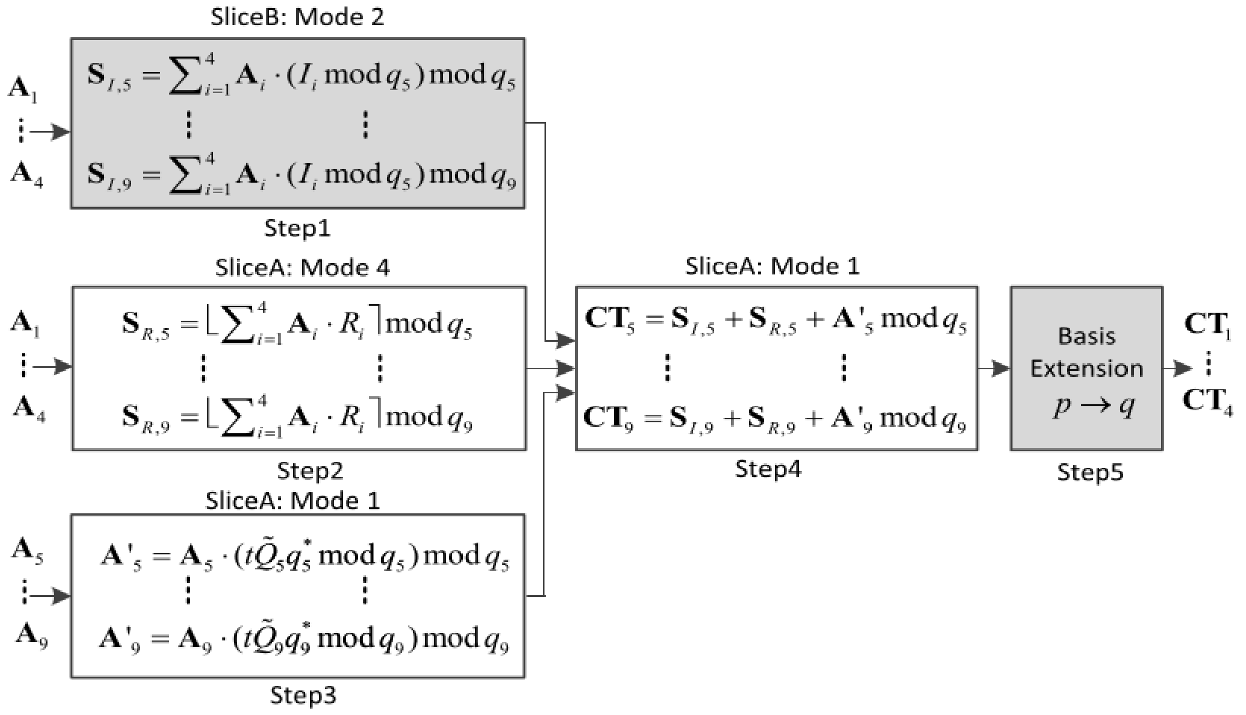

SliceB: Mode 2
$S_{I,5} = \sum_{i=1}^{4} A_i \cdot (I_i \bmod q_5) \bmod q_5$
$S_{I,9} = \sum_{i=1}^{4} A_i \cdot (I_i \bmod q_5) \bmod q_9$
Step1
SliceA: Mode 4
$S_{R,5} = \lfloor \sum_{i=1}^{4} A_i \cdot R_i \rceil \bmod q_5$
$S_{R,9} = \lfloor \sum_{i=1}^{4} A_i \cdot R_i \rceil \bmod q_9$
Step2
SliceA: Mode 1
$A'_5 = A_5 \cdot (t\tilde{Q}_5 q_5^* \bmod q_5) \bmod q_5$
$A'_9 = A_9 \cdot (t\tilde{Q}_9 q_9^* \bmod q_9) \bmod q_9$
Step3
SliceA: Mode 1
$CT_5 = S_{I,5} + S_{R,5} + A'_5 \bmod q_5$
$CT_9 = S_{I,9} + S_{R,9} + A'_9 \bmod q_9$
Step4
Basis Extension $p \to q$
Step5
A_1
A_4
A_5
A_9
CT_1
CT_4

Fig. 9. Mapping method of basis scaling unit.

Mapping Method And Execution Flow

Computing Units Mapping

relinearization unit:

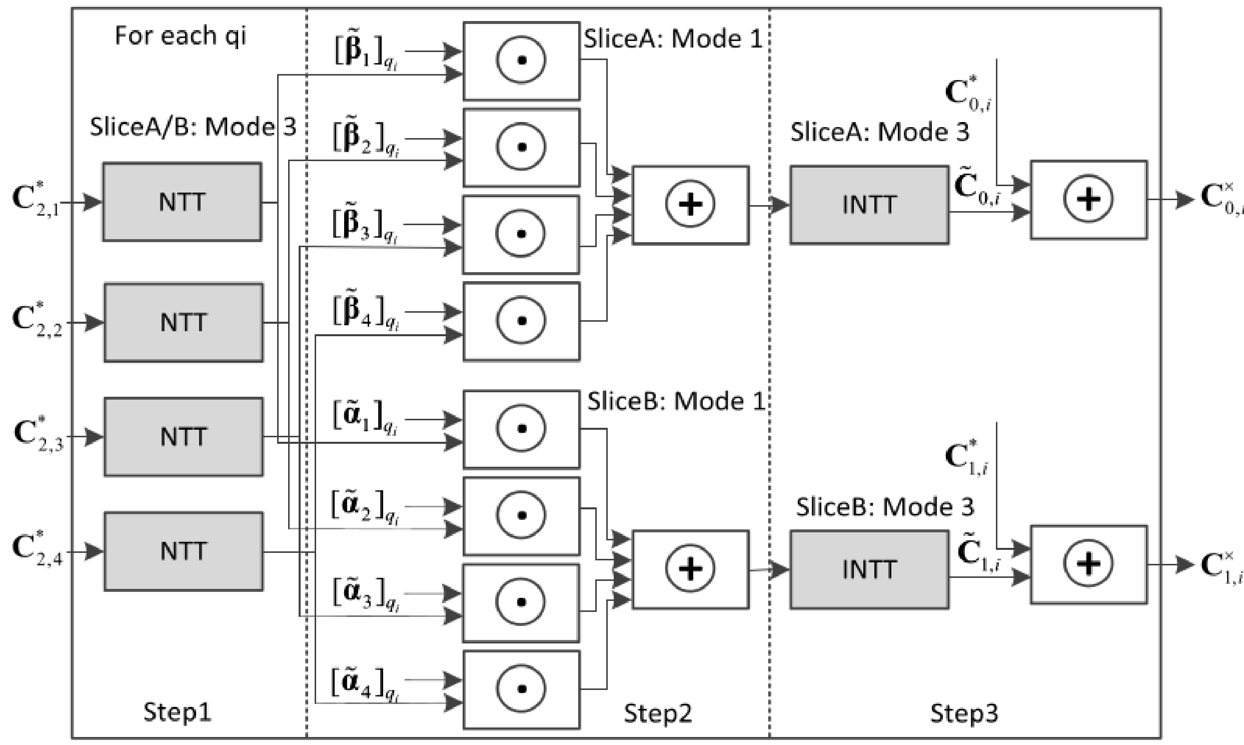


Fig. 10. Mapping method of RNS based relinearization unit.

Execution Flow of RNS-BFV

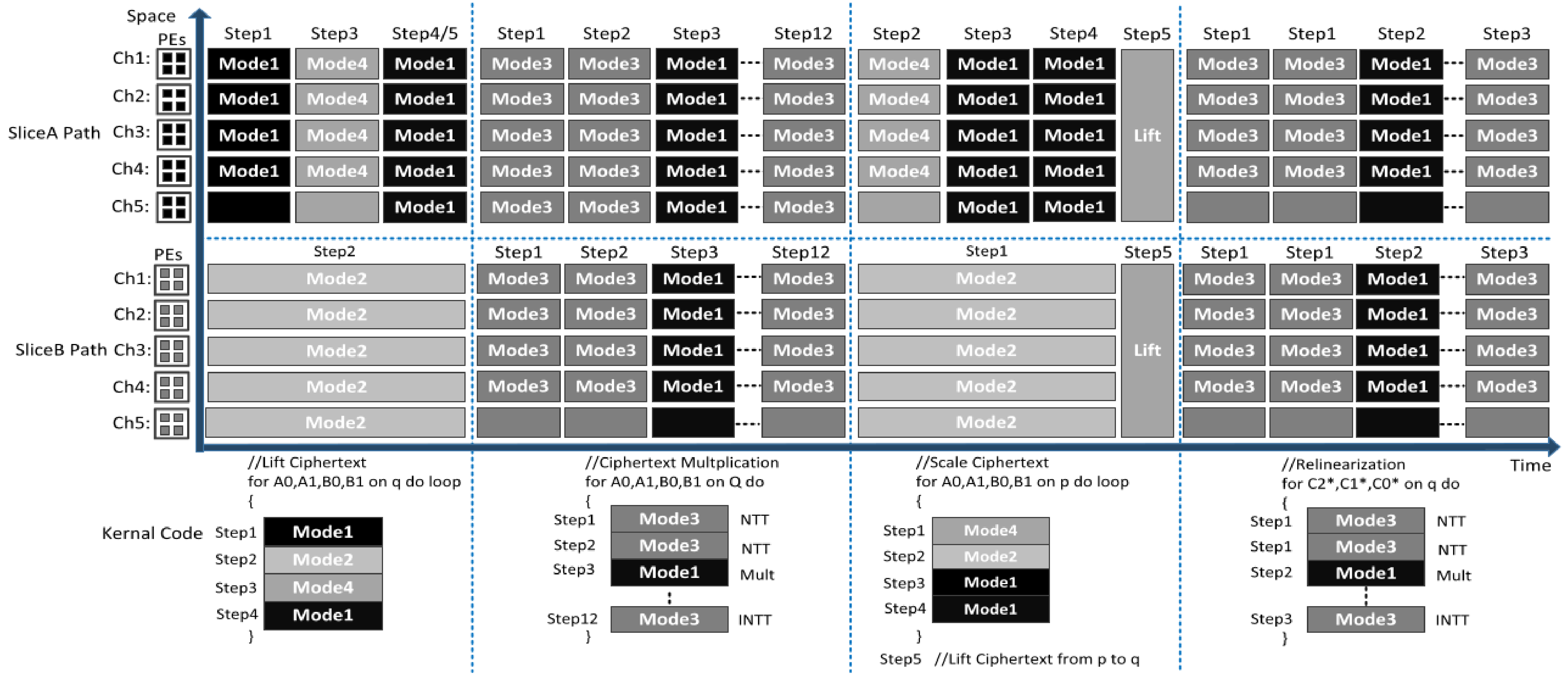


Fig. 11. Overall execution flow of RNS-BFV on ReMCA.

IMPLEMENTATION RESULTS AND COMPARISONS

FPGA Implementation Result

Xilinx Vivado tool
Virtex-7 XC7VX1140T
synthesized the design and achieved 250MHz
frequency under the parameter set ($N = 4096$, $\log(q) = 128$-bit, $\log(q_i) = 32$-bit)

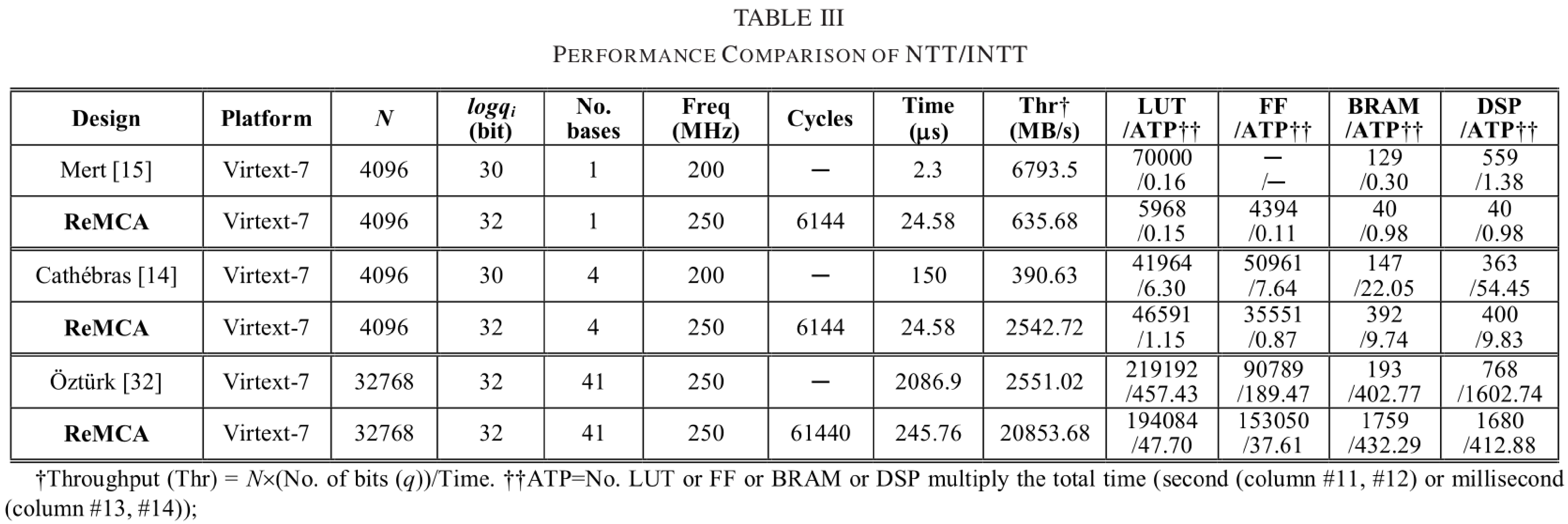


TABLE III

PERFORMANCE COMPARISON OF NTT/INTT

Design	**Platform**	***N***	***log*q_i (bit)**	**No. bases**	**Freq (MHz)**	**Cycles**	**Time (µs)**	**Thr† (MB/s)**	**LUT /ATP††**	**FF /ATP††**	**BRAM /ATP††**	**DSP /ATP††**
Mert [15]	Virtext-7	4096	30	1	200	—	2.3	6793.5	70000 /0.16	— /—	129 /0.30	559 /1.38
ReMCA	Virtext-7	4096	32	1	250	6144	24.58	635.68	5968 /0.15	4394 /0.11	40 /0.98	40 /0.98
Cathébras [14]	Virtext-7	4096	30	4	200	—	150	390.63	41964 /6.30	50961 /7.64	147 /22.05	363 /54.45
ReMCA	Virtext-7	4096	32	4	250	6144	24.58	2542.72	46591 /1.15	35551 /0.87	392 /9.74	400 /9.83
Öztürk [32]	Virtext-7	32768	32	41	250	—	2086.9	2551.02	219192 /457.43	90789 /189.47	193 /402.77	768 /1602.74
ReMCA	Virtext-7	32768	32	41	250	61440	245.76	20853.68	194084 /47.70	153050 /37.61	1759 /432.29	1680 /412.88

†Throughput (Thr) = $N\times$(No. of bits (q))/Time. ††ATP=No. LUT or FF or BRAM or DSP multiply the total time (second (column #11, #12) or millisecond (column #13, #14));

TABLE I

FPGA IMPLEMENTATION RESULT OF REMCA ON XC7VX1140T

***N*=4096**	Slice LUTs	Slice Registers	RAMB36E1	DSP48E1
Total available	712000	1424000	1880	3360
ReMCA	46591 (6.54%)	35551 (2.50%)	392 (20.85%)	400 (11.90%)
Each PE	1043 (0.15%)	811 (0.06%)	– –	10 (0.30%)
Each Barrett	560 (0.08%)	553 (0.04%)	– –	10 (0.30%)

TABLE II

TIMING RESULTS OF PRIMITIVE OPERATIONS

Operation	**Speed**	
	(cycles)	**(μs)**
Basis Extension	16384	65.54
Ciphertext Mult	47104	188.42
Basis Scaling	24576	98.30
Relinearization	22528	90.11
Total	110592	442.37
Each NTT	6144	24.58
Each INTT	6144	24.58
Each Point-wise Mult	1024	4.10

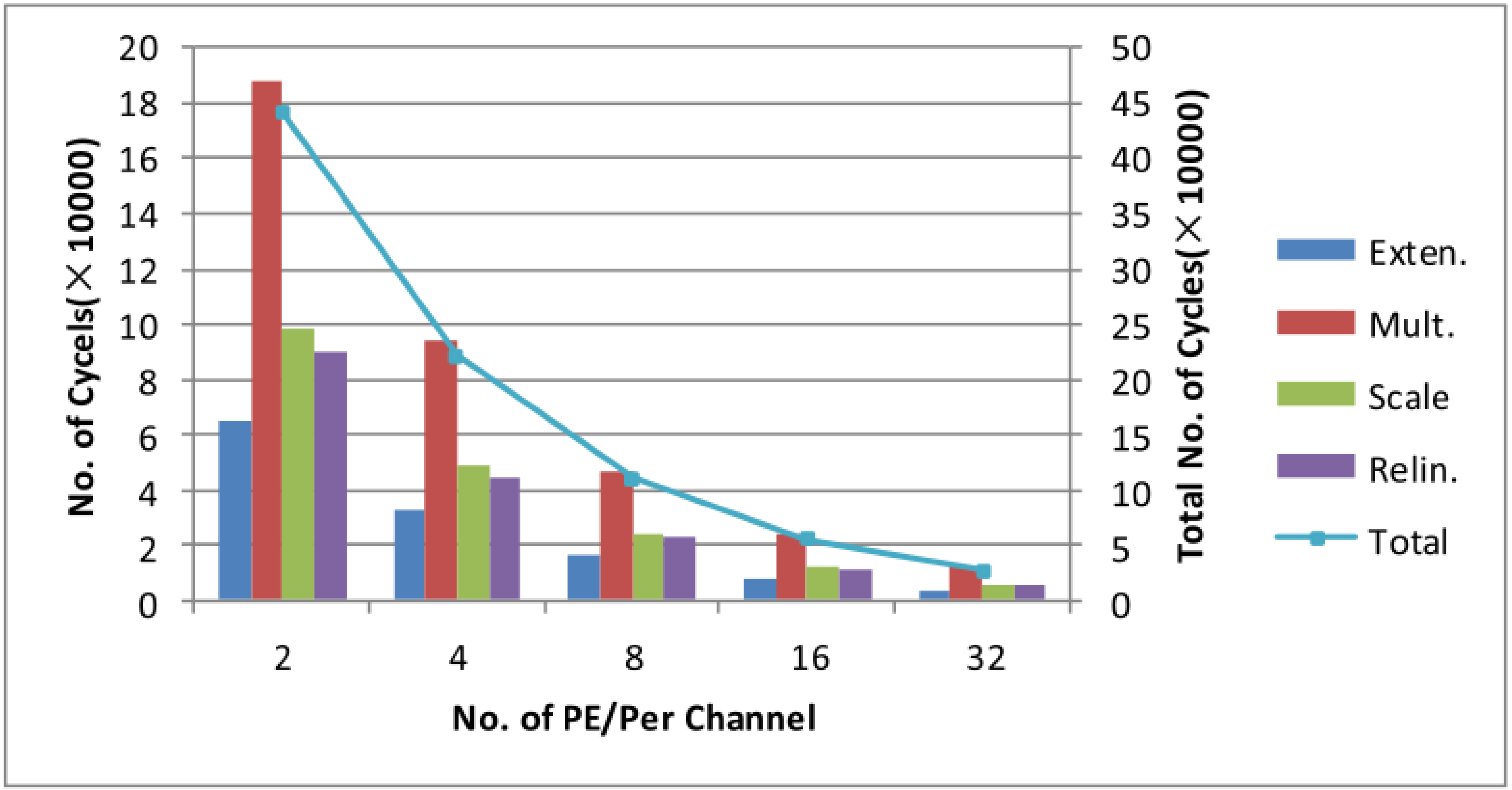


Fig. 12. The relationship between the number of PEs and performance.

Comparison of NTT/INTT Acceleration

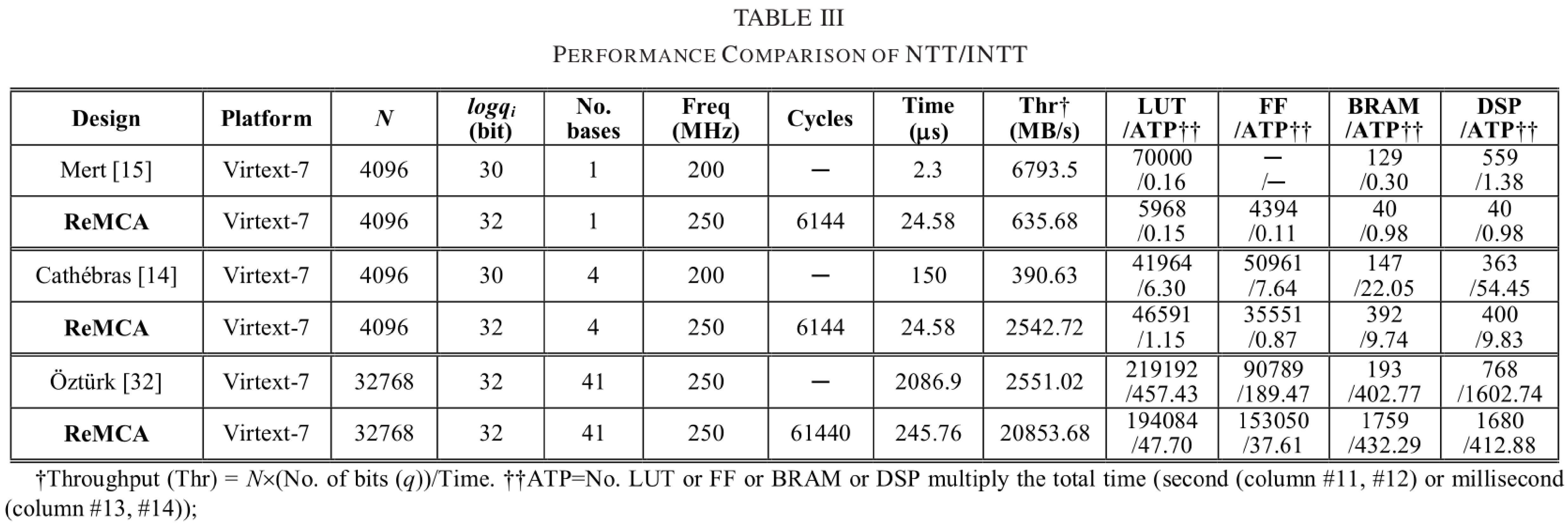


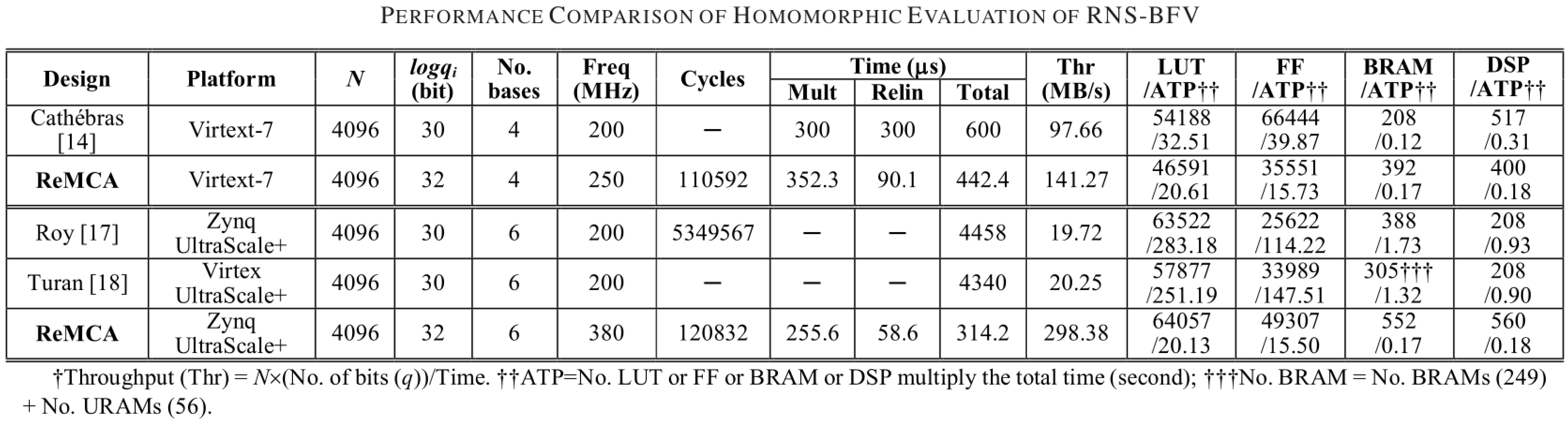
TABLE III

PERFORMANCE COMPARISON OF NTT/INTT

Design	**Platform**	***N***	***logq_i* (bit)**	**No. bases**	**Freq (MHz)**	**Cycles**	**Time (µs)**	**Thr† (MB/s)**	**LUT /ATP††**	**FF /ATP††**	**BRAM /ATP††**	**DSP /ATP††**
Mert [15]	Virtext-7	4096	30	1	200	—	2.3	6793.5	70000 /0.16	— /—	129 /0.30	559 /1.38
ReMCA	Virtext-7	4096	32	1	250	6144	24.58	635.68	5968 /0.15	4394 /0.11	40 /0.98	40 /0.98
Cathébras [14]	Virtext-7	4096	30	4	200	—	150	390.63	41964 /6.30	50961 /7.64	147 /22.05	363 /54.45
ReMCA	Virtext-7	4096	32	4	250	6144	24.58	2542.72	46591 /1.15	35551 /0.87	392 /9.74	400 /9.83
Öztürk [32]	Virtext-7	32768	32	41	250	—	2086.9	2551.02	219192 /457.43	90789 /189.47	193 /402.77	768 /1602.74
ReMCA	Virtext-7	32768	32	41	250	61440	245.76	20853.68	194084 /47.70	153050 /37.61	1759 /432.29	1680 /412.88

†Throughput (Thr) = N×(No. of bits (q))/Time. ††ATP=No. LUT or FF or BRAM or DSP multiply the total time (second (column #11, #12) or millisecond (column #13, #14));

Comparison of BFV Acceleration



PERFORMANCE COMPARISON OF HOMOMORPHIC EVALUATION OF RNS-BFV

Design	**Platform**	***N***	***logq_i* (bit)**	**No. bases**	**Freq (MHz)**	**Cycles**	**Time (μs)**			**Thr (MB/s)**	**LUT /ATP††**	**FF /ATP††**	**BRAM /ATP††**	**DSP /ATP††**
							Mult	**Relin**	**Total**					
Cathébras [14]	Virtext-7	4096	30	4	200	—	300	300	600	97.66	54188 /32.51	66444 /39.87	208 /0.12	517 /0.31
ReMCA	Virtext-7	4096	32	4	250	110592	352.3	90.1	442.4	141.27	46591 /20.61	35551 /15.73	392 /0.17	400 /0.18
Roy [17]	Zynq UltraScale+	4096	30	6	200	5349567	—	—	4458	19.72	63522 /283.18	25622 /114.22	388 /1.73	208 /0.93
Turan [18]	Virtex UltraScale+	4096	30	6	200	—	—	—	4340	20.25	57877 /251.19	33989 /147.51	305††† /1.32	208 /0.90
ReMCA	Zynq UltraScale+	4096	32	6	380	120832	255.6	58.6	314.2	298.38	64057 /20.13	49307 /15.50	552 /0.17	560 /0.18

†Throughput (Thr) = N×(No. of bits (q))/Time. ††ATP=No. LUT or FF or BRAM or DSP multiply the total time (second); †††No. BRAM = No. BRAMs (249) + No. URAMs (56).