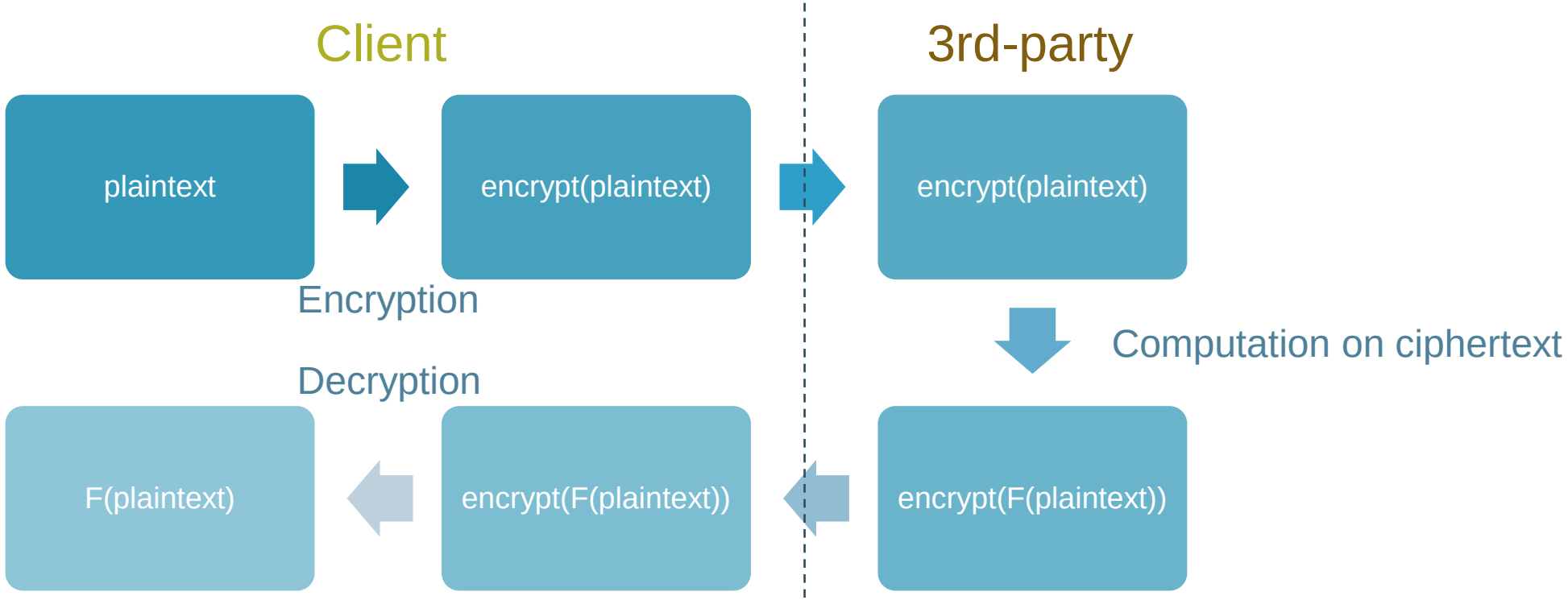
KU LEUVEN

Hardware Acceleration of the Prime-Factor and Rader NTT for BGV Fully Homomorphic Encryption

Presentation: David Du Pont

Fully Homomorphic Encryption (FHE)

- Allows performing computations on encrypted data



Fully Homomorphic Encryption (FHE)

- Based on ring learning with errors (RLWE)
- Data is encrypted into large polynomials
- Number of operations bounded by noise growth

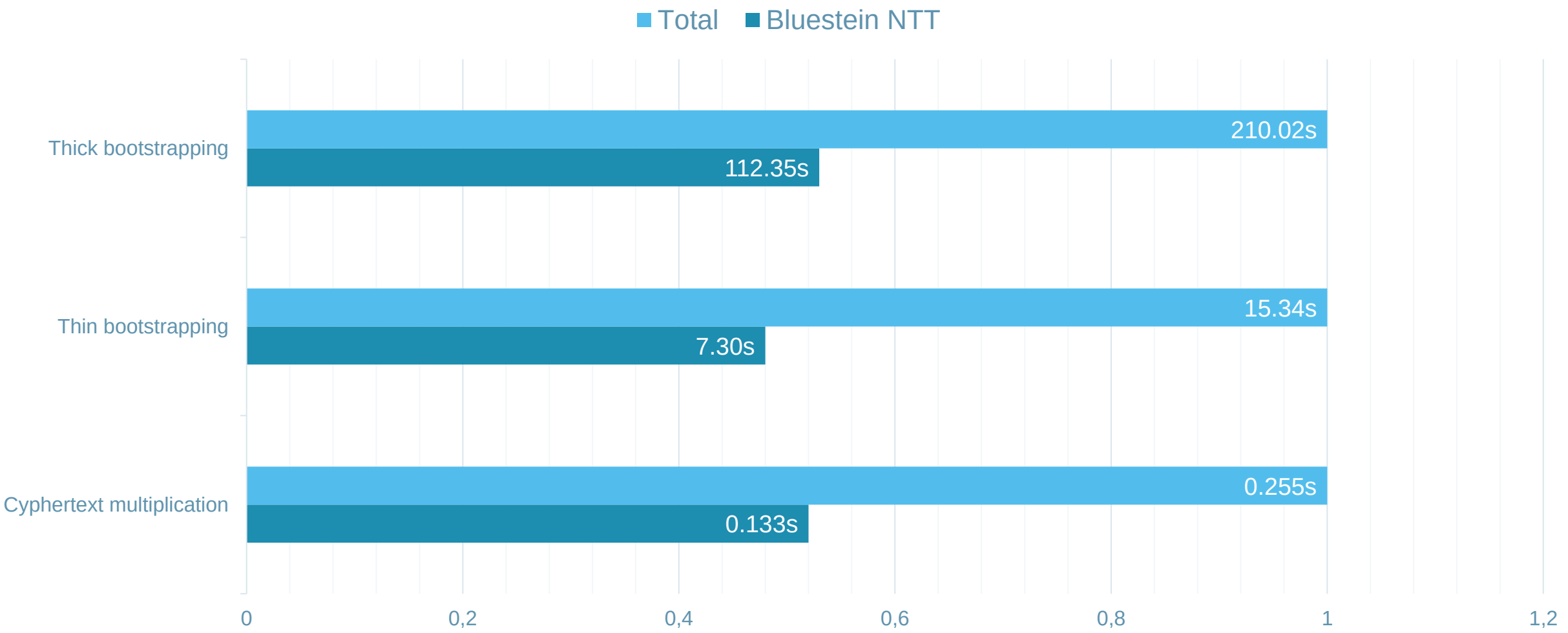
$\mathbb{Z}_{17}[x]/\langle x^n+1\rangle$

$$
\begin{array}{cll}
 & 7 + 4x + 16x^2 + \ldots + 14x^n & \\
\times & 1 + 2x + 11x^2 + \ldots + 3x^n & \text{secret} \\
+ & 0 - 1x + 1x^2 + \ldots + 1x^n & \text{small error} \\
\hline
= & 12 - 9x + 10x^2 + \ldots + 4x^n &
\end{array}
$$

Number theoretic transform (NTT)

- Number theoretic transform (NTT) is finite field equivalent of discrete Fourier transform (DFT)
- Efficient implementation of polynomial multiplication using NTT
- HElib's BGV requires non-power-of-two length NTTs

Timing of HElib operations



Non-power-of-two FFT algorithms

- Prime-factor FFT algorithm
- Bluestein's algorithm
- Rader's algorithm

Prime-factor FFT algorithm (PFA)

- N factorizes into coprime number N_1 and N_2
- Transform N-point DFT into N_1xN_2 two dimensional DFT
- Only permutation of data, no additional multiplications

+ Reduces number of twiddle factors

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 10 & 5 \\ 6 & 1 & 11 \\ 12 & 7 & 2 \\ 3 & 13 & 8 \\ 9 & 4 & 14 \end{bmatrix}$$

Bluestein’s algorithm

- Computes N-point DFT using N-point convolution
- Convolution can be padded to power of two length

\+ Works for any N

\+ No permutation of input data

\+ Efficient to compute power-of-two-length DFT

\- Number of points in DFT is at best doubled, at worst quadrupled

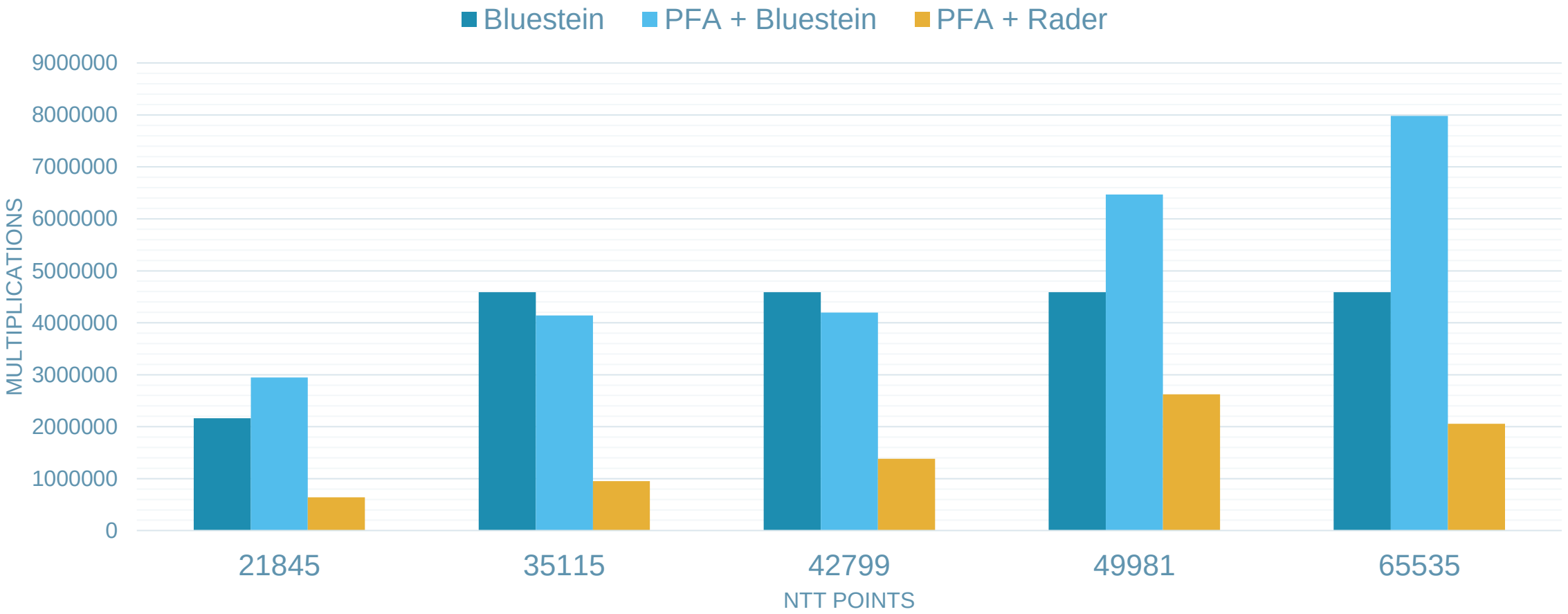
Rader's algorithm

- Computes N-point DFT using (N-1)-point convolution
- Can only be used when N is prime
- PFA can be used to compute (N-1)-point DFT

+ Number of points in DFT is not increased
+ Possible to choose N so (N-1)-point DFT is efficient to compute

- Permutation of input data

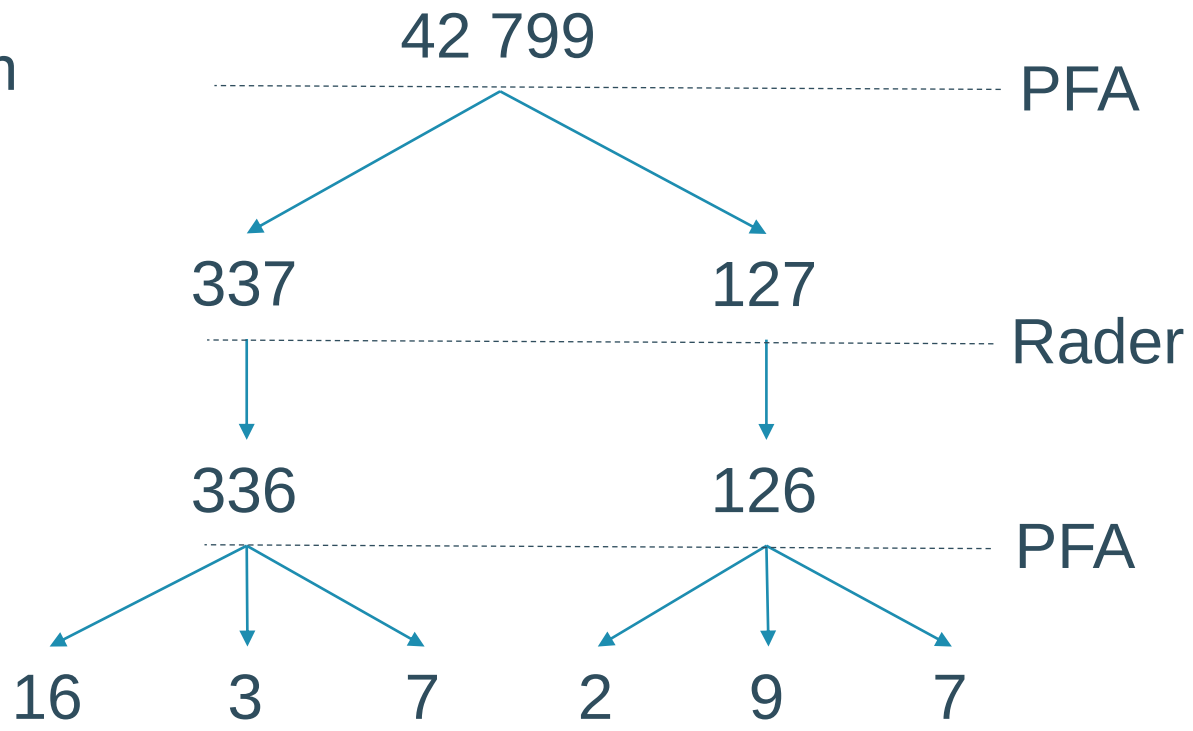
Comparison of FFT algorithms



Recursive use of PFA and Rader's algorithm

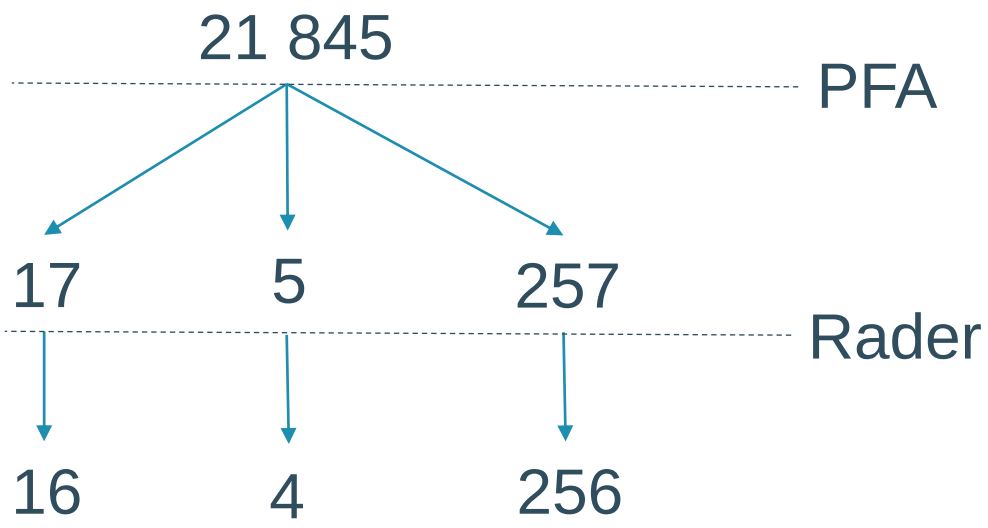
- If N is not prime use PFA
- If N is not prime use Rader's algorithm

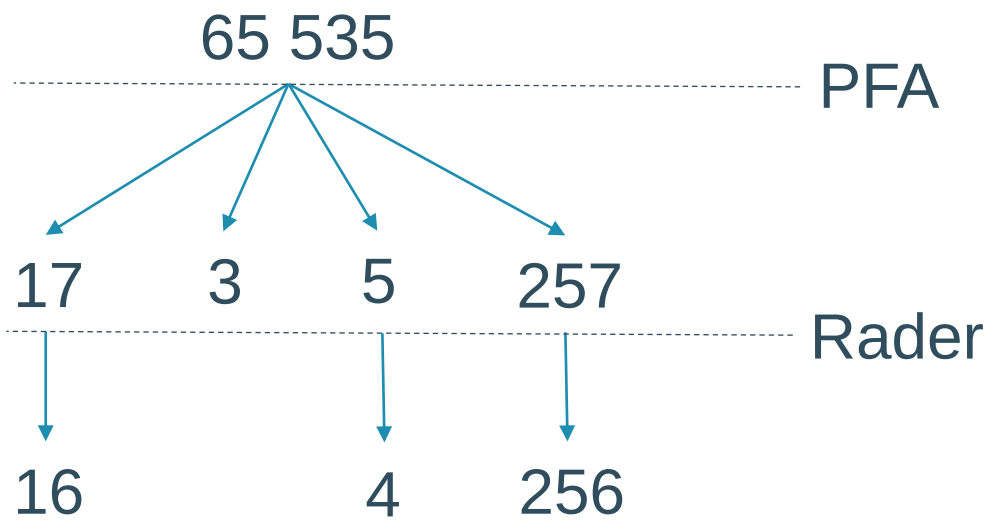
- Ends with small prime or prime power



Recursive use of PFA and Rader's algorithm

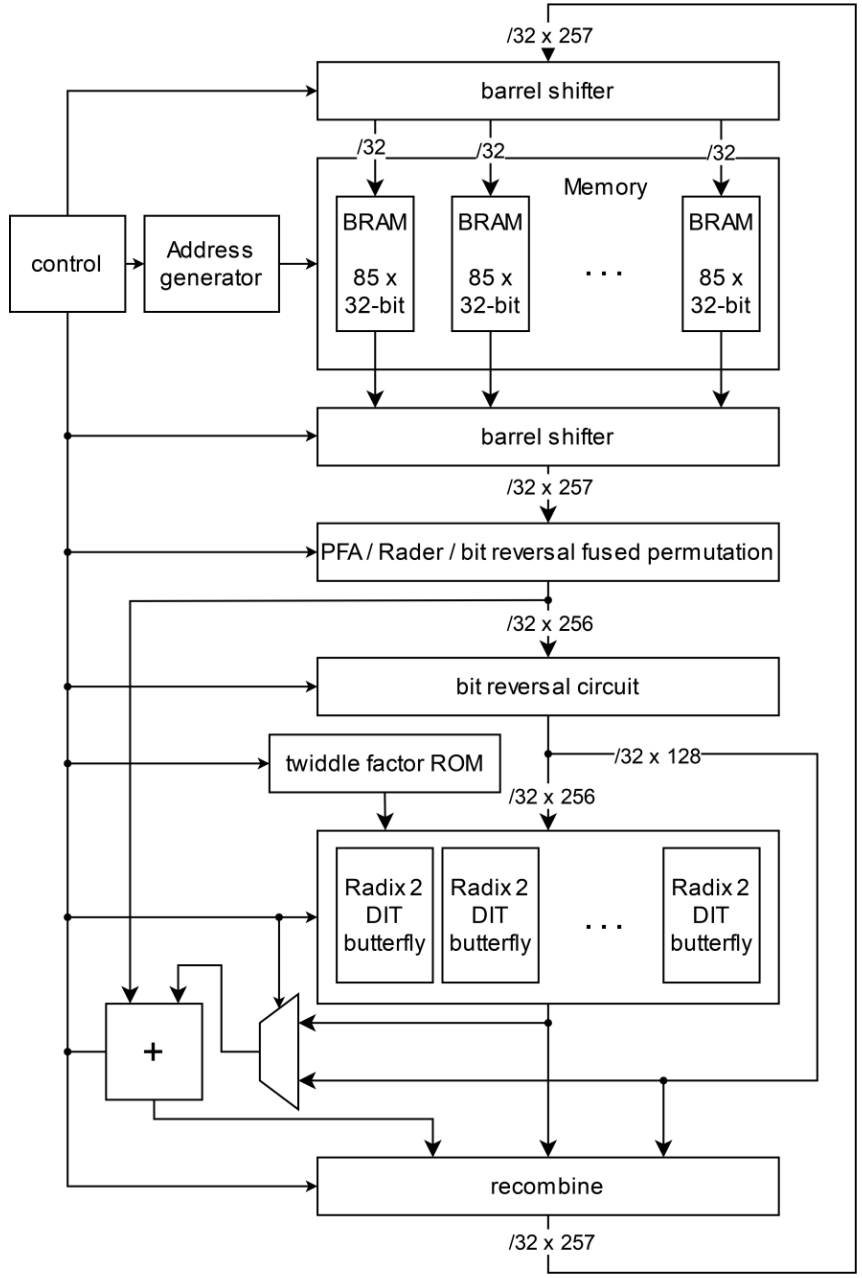
More efficient parameter choices





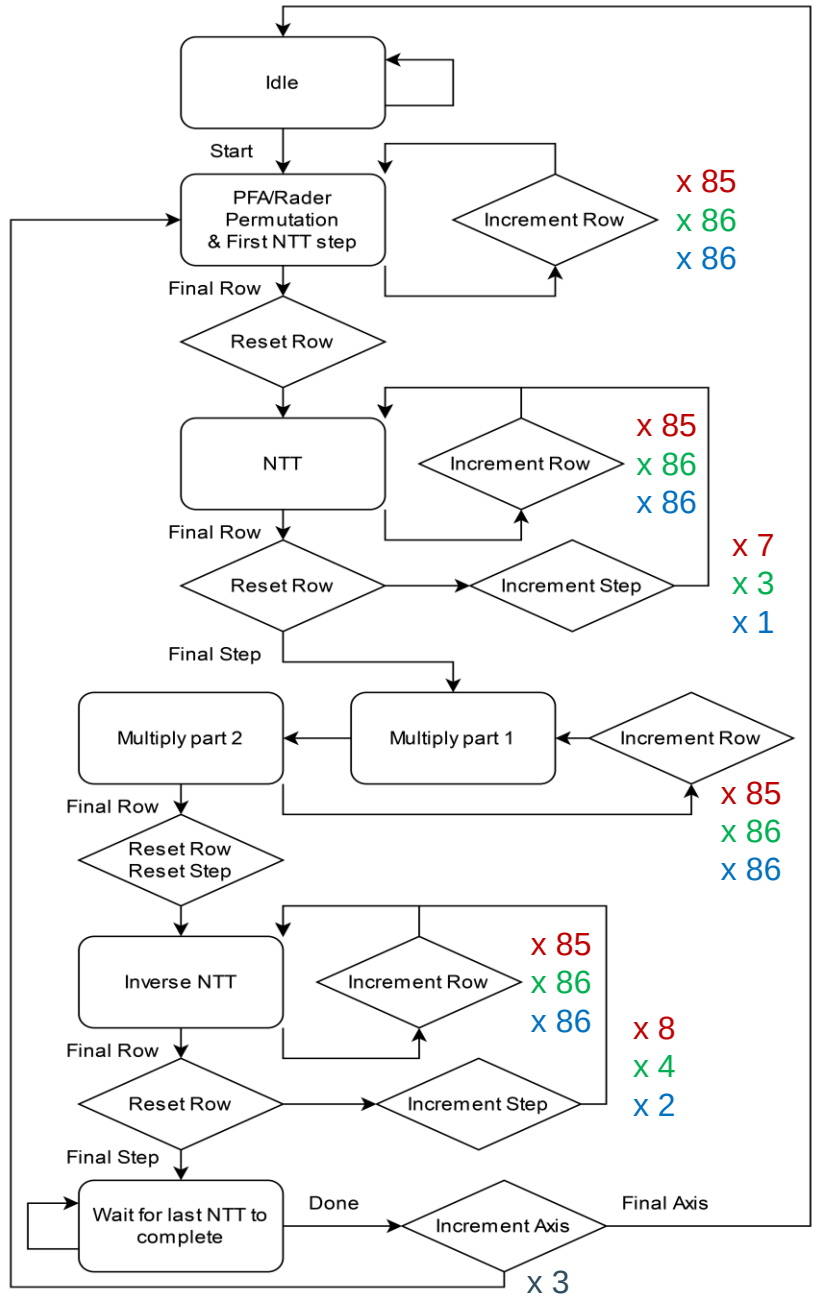
Architecture overview

- Maximize parallelism
- Fully Pipelined
- Functional Reuse



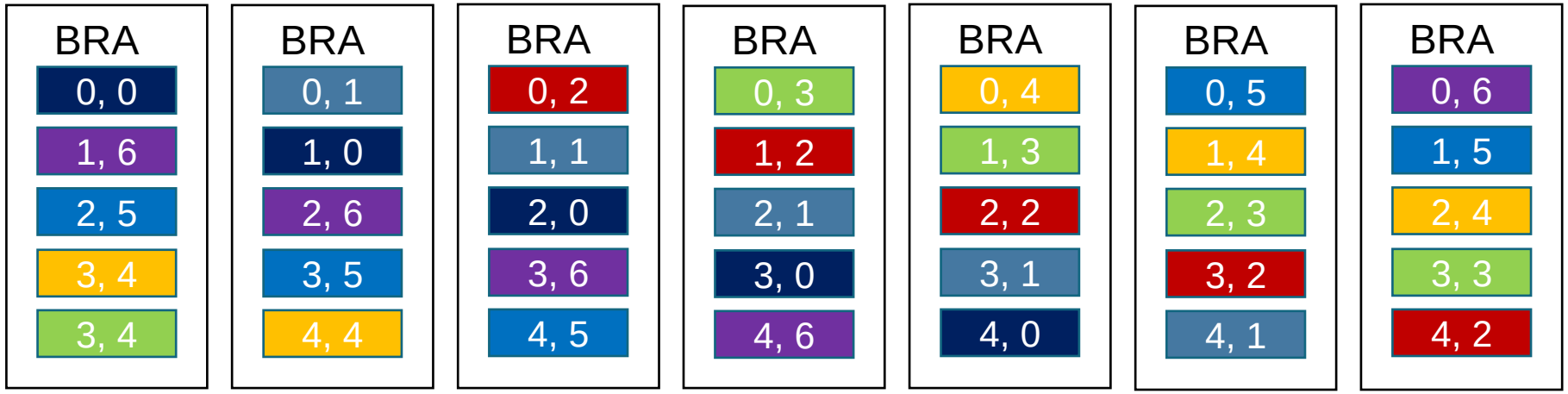
Control Flow

- Each NTT computation stage done on all rows along one axis before moving to the next stage
- Pipeline stalls only in final state to wait for the last NTTs along the current axis to complete



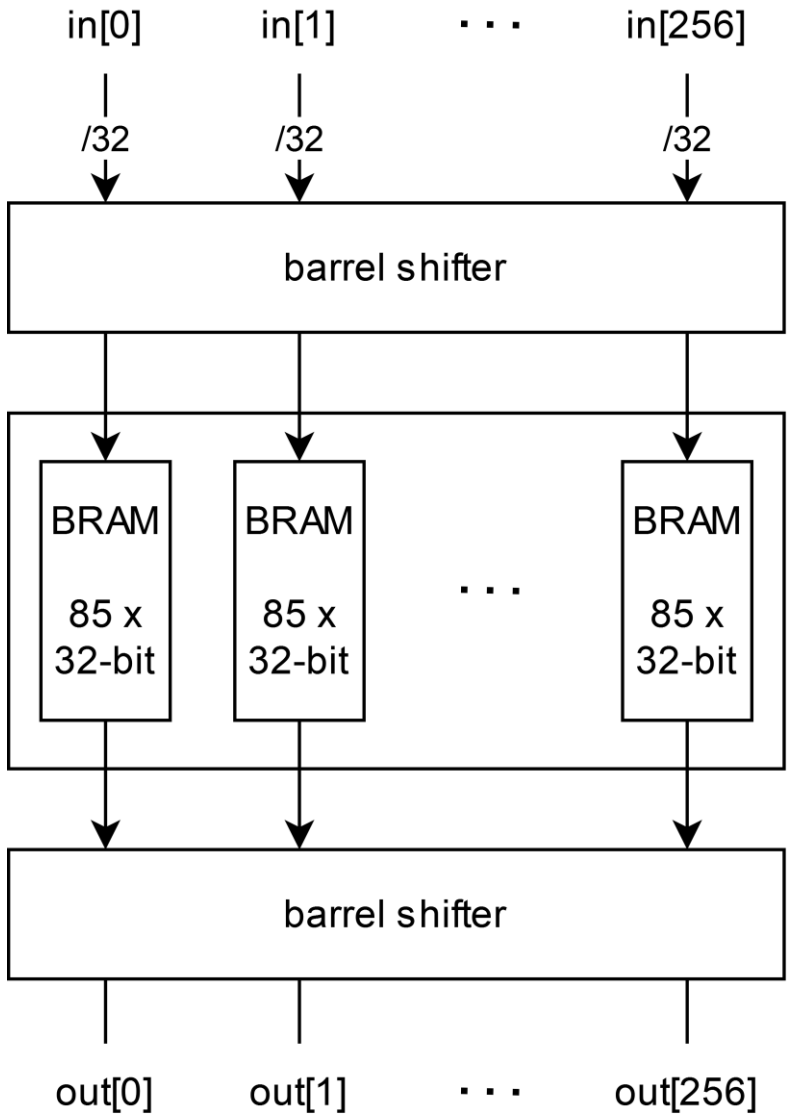
Memory

- Data stored in Block RAM on FPGA
- Access elements of both rows and columns in parallel for 2D-FFT



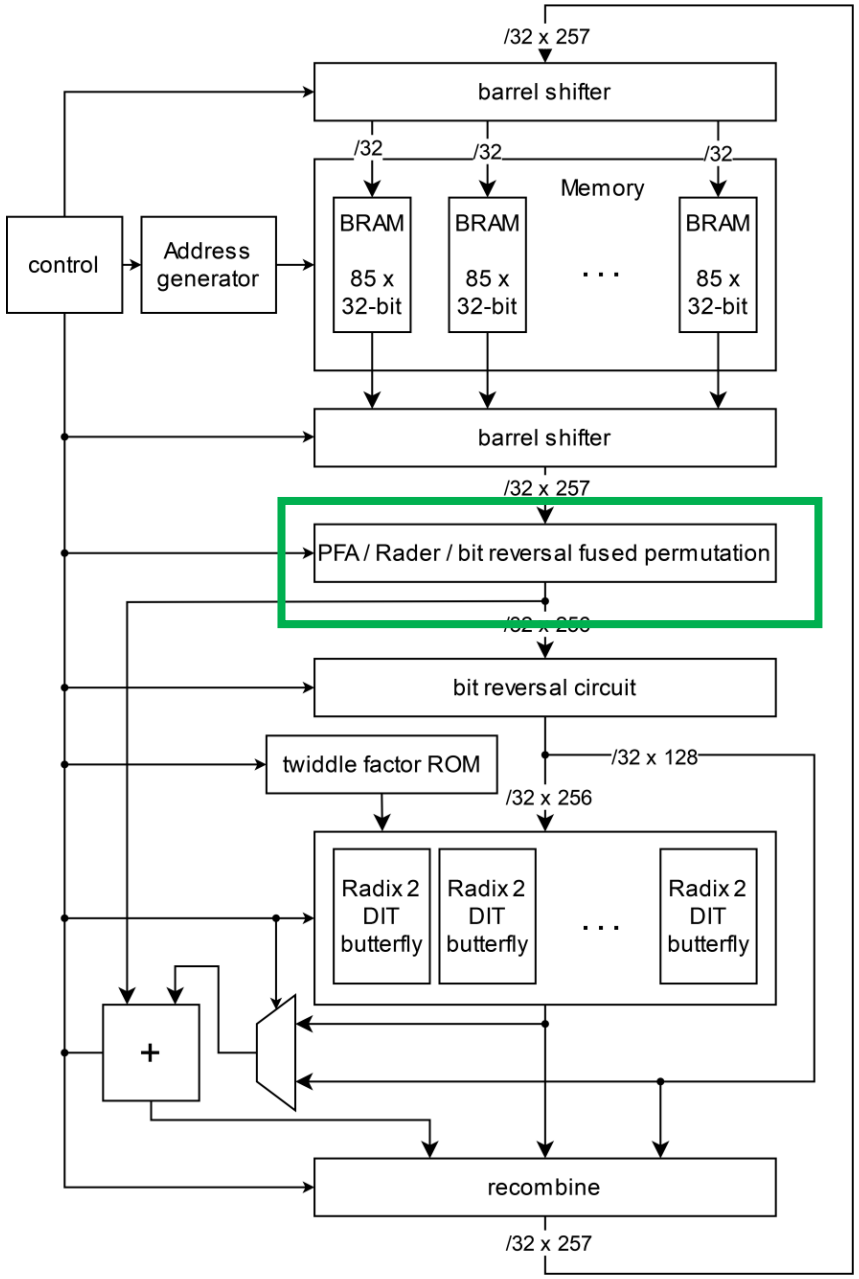
Memory

- Data stored in Block RAM on FPGA
- Access elements of both rows and columns in parallel for 2D-FFT
- Barrel shifter at input and output to remove and reapply offset



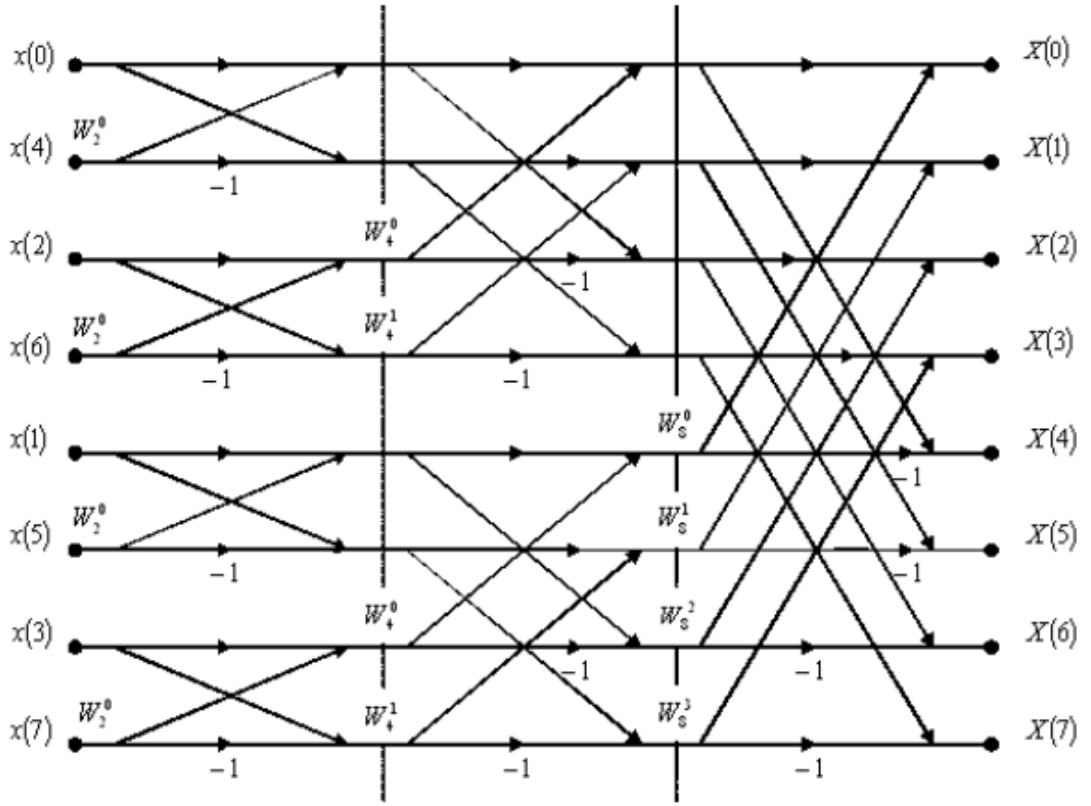
Combining Permutations

- Re-indexing for PFA and Rader's algorithm can be combined into a single permutation
- Six total permutations
- Select permutation with 6-to-1 multiplexers



Cooley-Tukey FFT

- Breaks larger NTT of size 2n into two n-size NTTs
- Uses “butterflies” for computations
- Design focuses on 256-point FFT with 128 butterflies; hardware also used for 16-point and 4-point FFTs

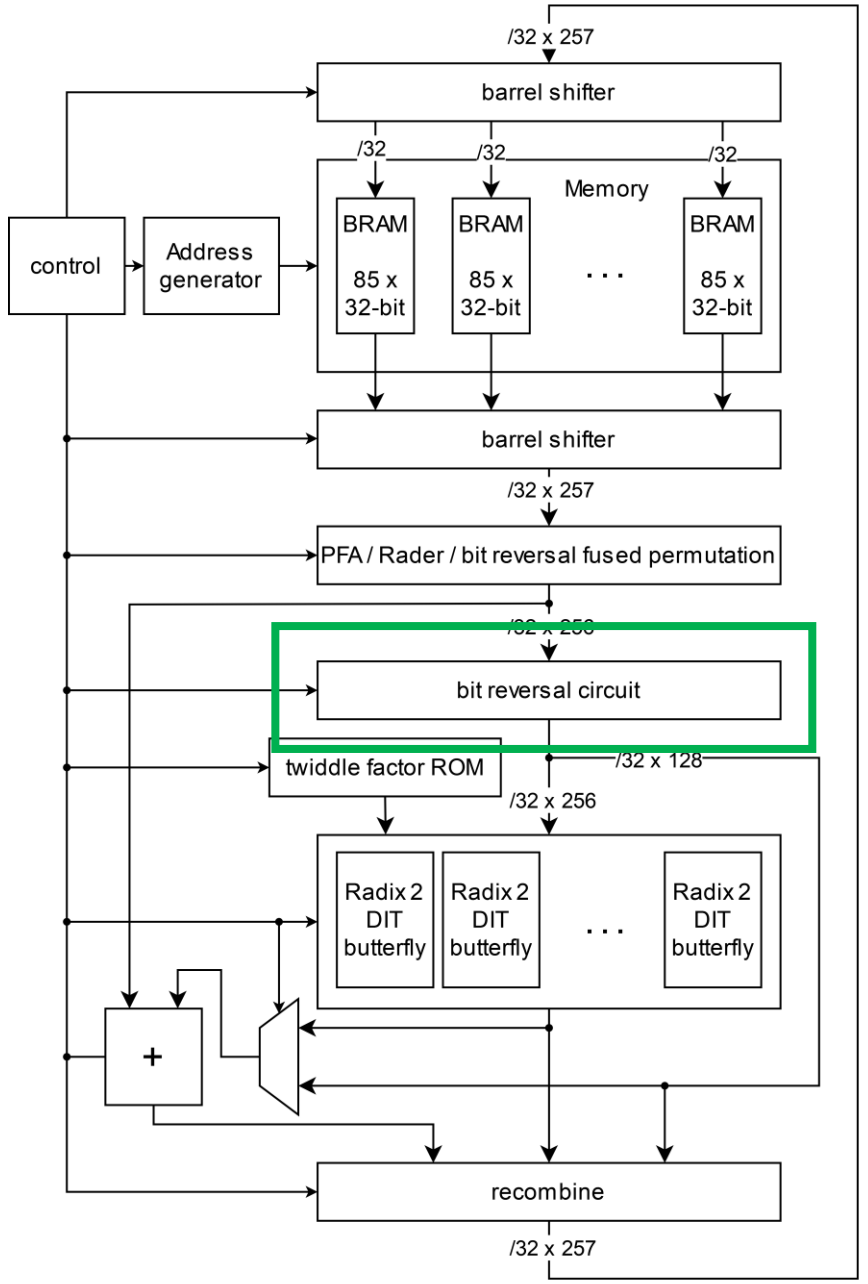


Butterfly diagram for radix-2 DIT FFT algorithm from Pace et al. [15]

Bit-reversal permutations

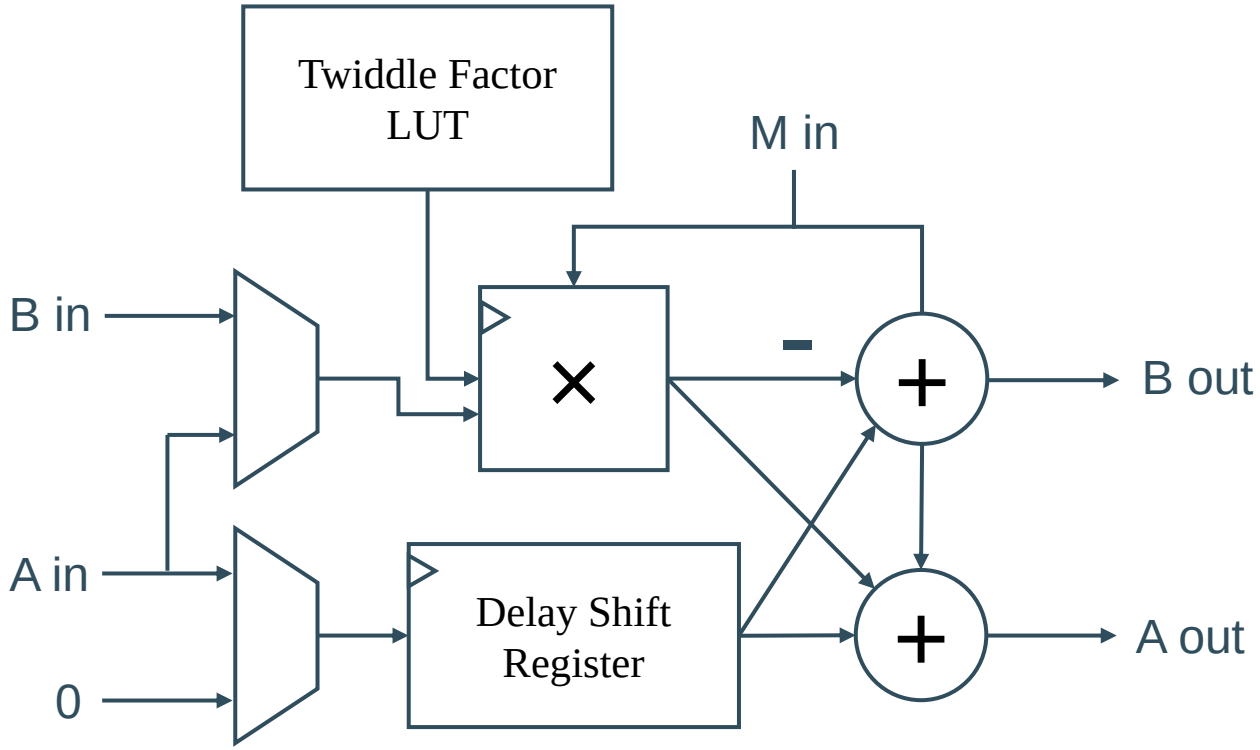
0 1 2 3 4 5 6 7 → 0 4 2 6 1 5 3 7

- Cooley-Tukey FFT requires "bit-reversal" reordering after each butterfly stage
- Implemented as a series of eight bit-reversal permutations of increasing length 2, 4, 8, 16, 32, 64, 128, 256



Butterfly Units

- Uses word-level Montgomery modular multiplier from Mert et al. [14]
- Twiddle factors pre-stored in lookup table
- Butterfly unit multipliers repurposed for Rader's algorithm convolution
  - Multiplexers added to allow multiplication without final addition



Additions for Rader's algorithm

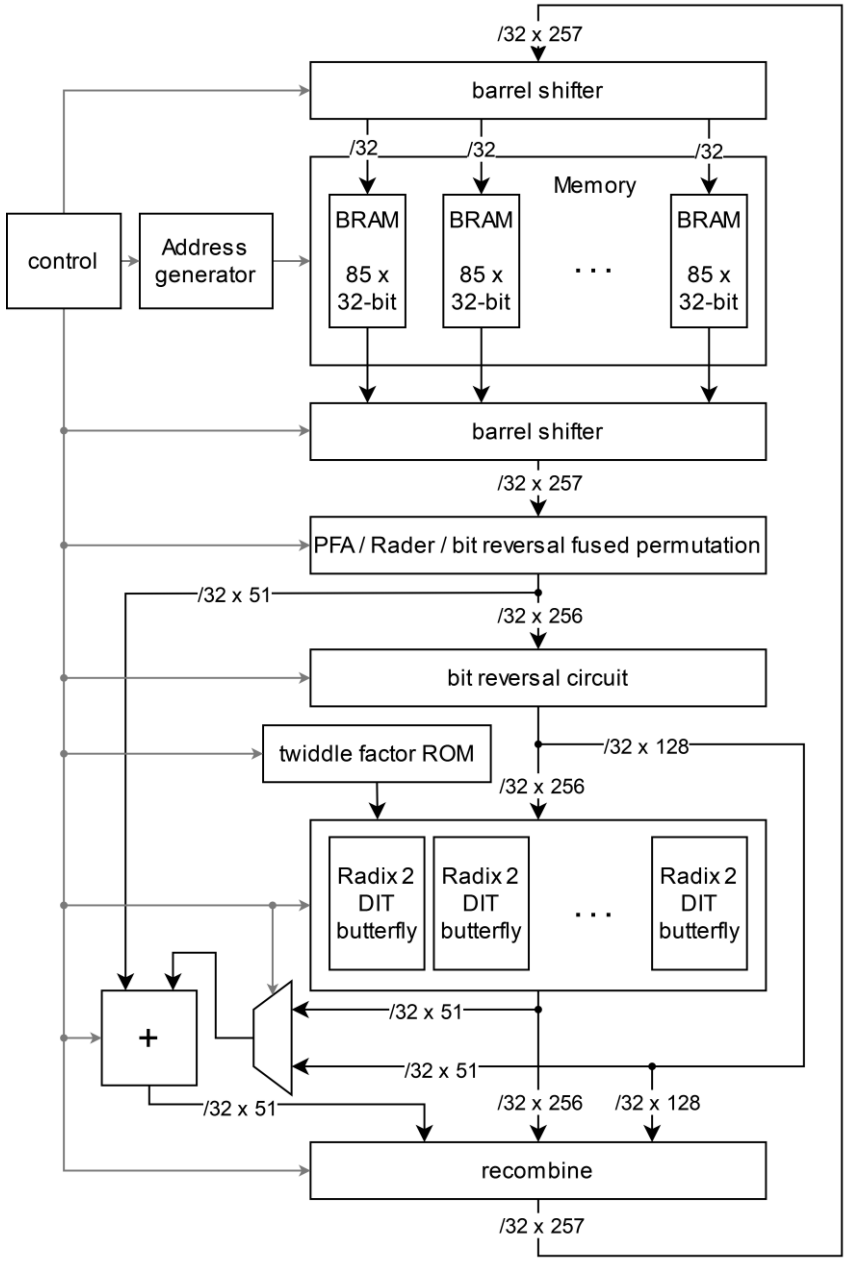
- Pointwise multiplication done separately for even and uneven indices
- Two steps
  1. $X_0=x_0+A_0$ and even-indexed multiplication
  2. C_0+x_0 and uneven-indexed multiplication
- Perform steps in subsequent cycles -> no memory needed to store x_0

Algorithm 1: Optimized Rader's Algorithm

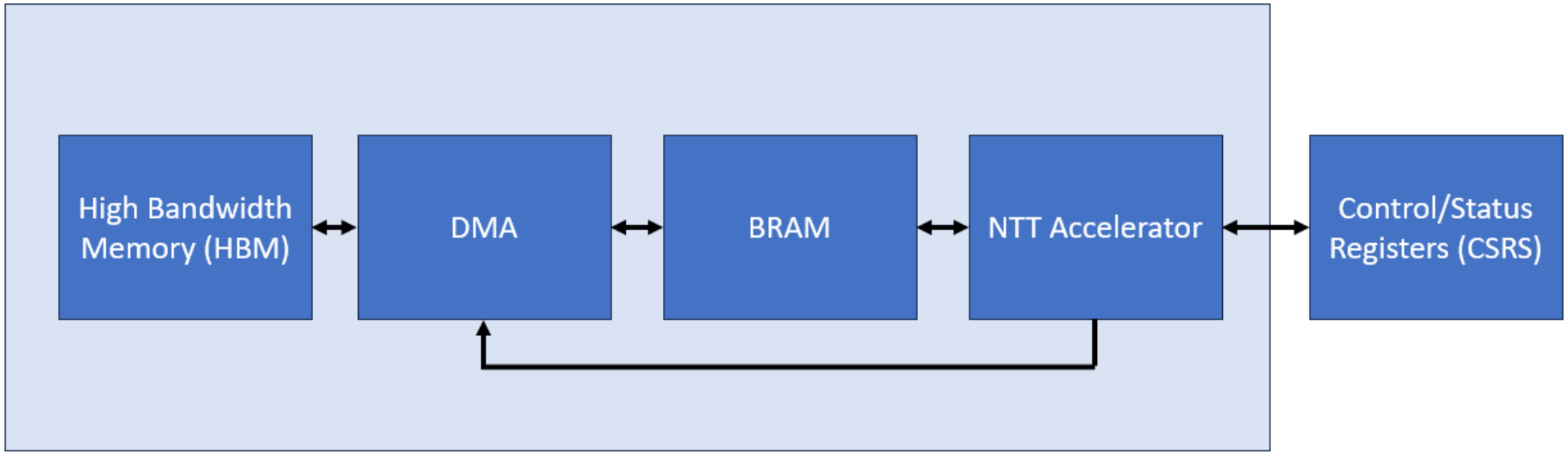
Data: $\mathbf{x}$, sequence of N integers
Result: $\mathbf{X} = \mathcal{N}(\mathbf{x})$

1 $\mathbf{x}' \leftarrow \text{RaderPermutation}([x_1, x_2, \ldots, x_{N-1}])$;
2 $\mathbf{A} \leftarrow \mathcal{N}(\mathbf{x}')$;
3 $X_0 \leftarrow A_0 + x_0$;
4 $\mathbf{C} \leftarrow \mathbf{B} \odot \mathbf{A}$;
5 $C_0 \leftarrow C_0 + x_0$;
6 $[X_1, X_2, \ldots, X_{N-1}] \leftarrow \mathcal{N}^{-1}(\mathbf{C})$;

Architecture overview



Hardware-Software Interface



- 85 cycles fill BRAMs
- Duplicate BRAM and transfer memory during NTT computation

Implementation Results

Frequency (MHz)	Cycles	CLB LUT	CLB Register	BRAM	DSP
250	2987	224074	179900	300	1024

- AMD Alveo U250 FPGA

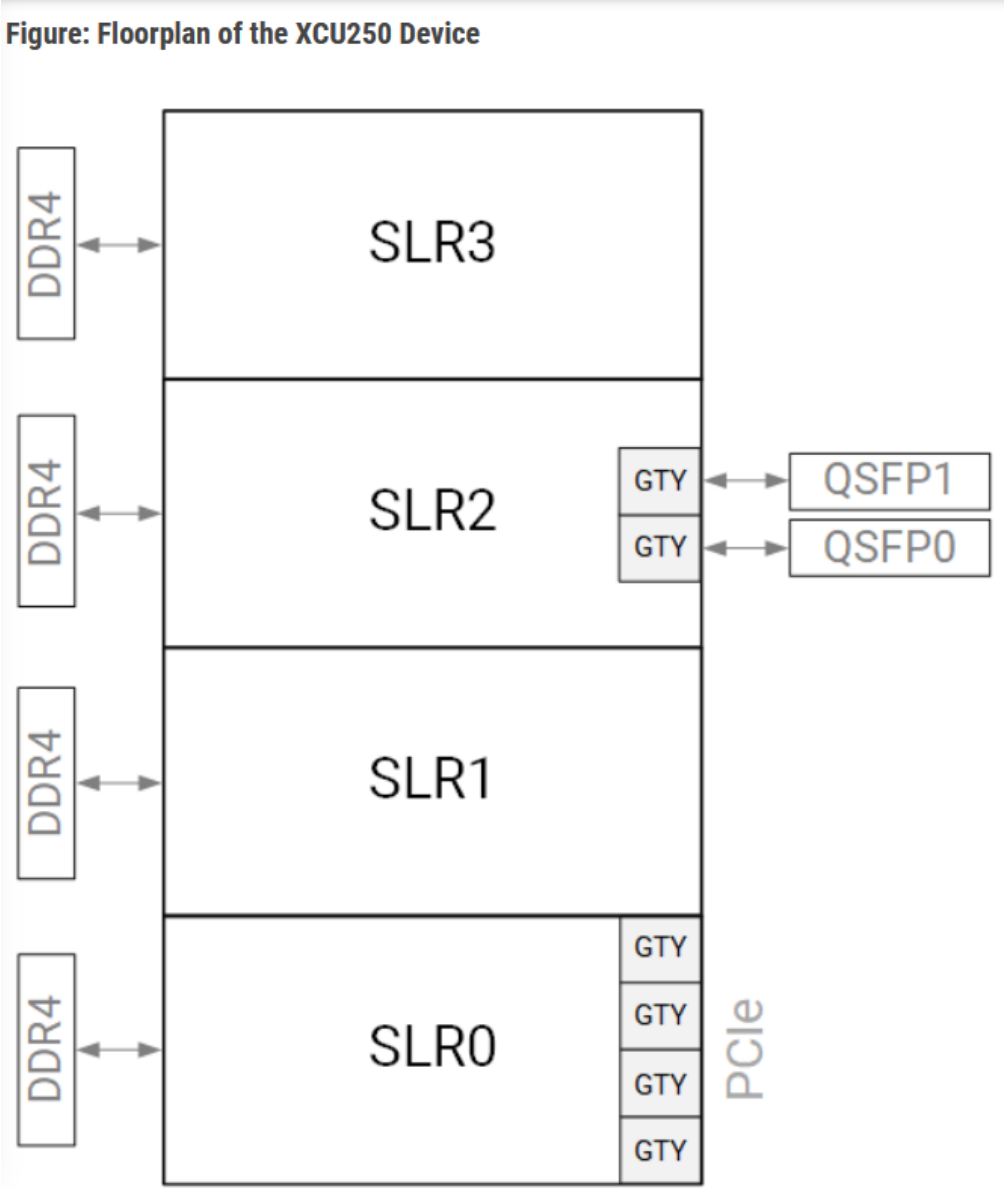
Implementation Results

Frequency (MHz)	Cycles	CLB LUT	CLB Register	BRAM	DSP
250	2987	224074	179900	300	1024

- Time for one NTT: 11.9µs
- Compared to software: 1.10ms
  - HElib on Intel Core i7-9750h @4.3 GHz
- 92× improvement

Implementation Results

- Problem: SLL (Super Long Line) Congestion
- Solution: All logic confined to single SLR → no SLLs used
  - New problem: local congestion within SLR



Comparison to Other Implementation

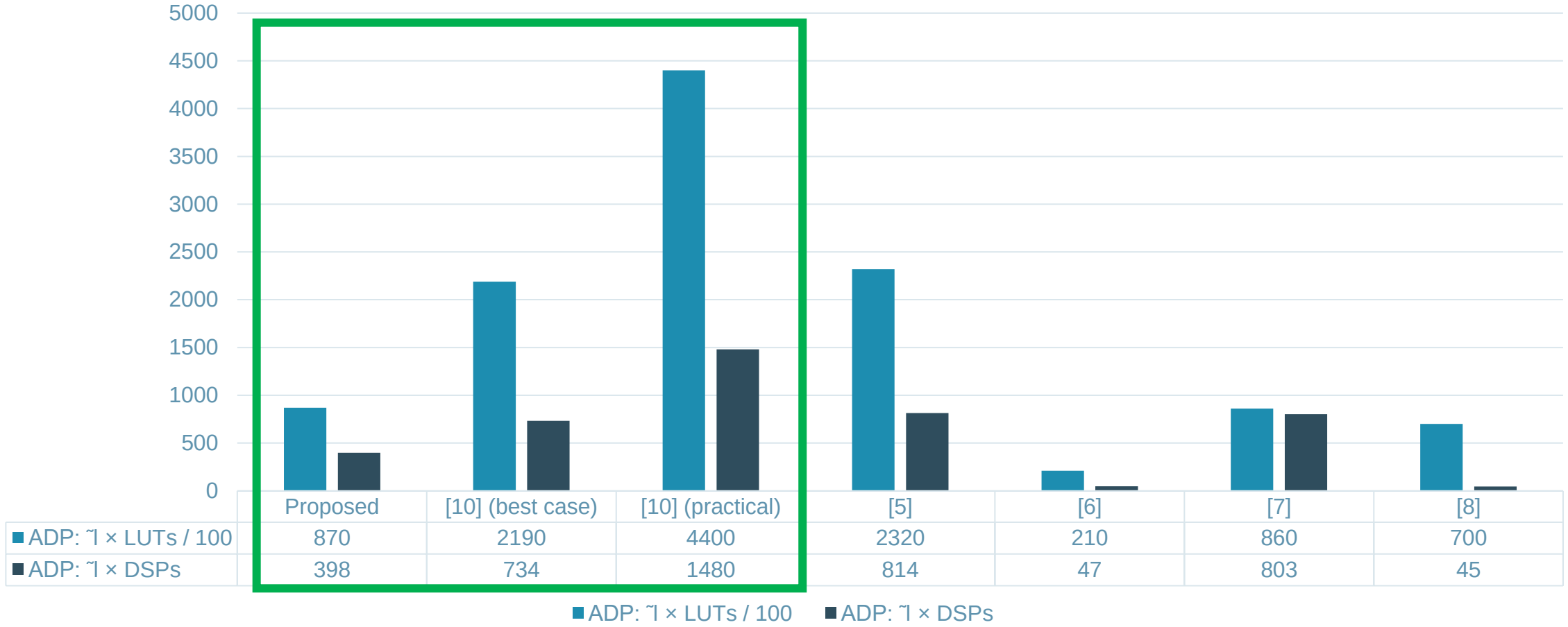
TABLE IV

COMPARISON TO OTHER NTT IMPLEMENTATIONS. BOTH NON-POWER-OF-TWO (LEFT) AND POWER-OF-TWO NTTS (RIGHT) ARE INCLUDED.

Design	**This work**	[10]	[5]	[6]	[7]	[8]	[9]
m	**21845**	8193/4369	32768	8192	4096	512	1024
Coefficient size (bits)	**32**	25	32	54	60	13	16
Platform	**Alveo U250**	Virtex-7	Virtex-7	Stratix 10 GX 2800	Virtex-7	Virtex 6	Artix-7
Frequency (MHz)	**250**	250	250	300	125	278	45.47
Cycles	**2987**	5825	12725	768	972	2304	18537
CLB LUTs	**224k**	76.2k	219k	142k	99.3k	1536	2908
CLB Registers	**170k**	-	90.7k	387k	-	953	170
BRAM	**429**	62	193	725	176	3	0
DSPs	**1024**	256	768	320	929	1	9
$\tilde{l}$ (µs)	**0.44**	2.9/5.8	1.0	0.14	0.86	45	815
ADP: $\tilde{l} \times$ LUTs	**87k**	219k/440k (**2.5×/5.1×**)	232k	21k	86k	70k	2.4M
ADP: $\tilde{l} \times$ DSPs	**398**	734/1480 (**1.8×/3.7×**)	814	47	803	45	7338

Comparison to Other Implementation

Performance scaled to 1024 NTT points and 32-bit words (lower is better)



	Proposed	[10] (best case)	[10] (practical)	[5]	[6]	[7]	[8]
ADP: ˜l × LUTs / 100	870	2190	4400	2320	210	860	700
ADP: ˜l × DSPs	398	734	1480	814	47	803	45

Conclusion

- Hardware architecture for efficient non-power-of-two NTT, targeting fully homomorphic encryption via the BGV scheme
- Combination of Prime-Factor FFT and Rader's algorithms shown to be superior for bootstrappable parameters in HElib's BGV
- Design focuses on 21845-th cyclotomic polynomial, employs efficient arithmetic, parallel processing, pipelining, and functional reuse
- Competitive performance demonstrated

Questions?

References

- [1] G. Pace and C. Vella. Describing and verifying fft circuits using sharphdl. 06 2023
- [5] E. ¨ Ozt¨ urk, Y. Dor¨ oz, E. Savas, and B. Sunar, “A custom accelerator for homomorphic encryption applications,” IEEE Trans. Computers, vol. 66, no. 1, pp. 3–16, 2017. [Online]. Available: https://doi.org/10.1109/TC.2016.2574340
- [6] M.S. Riazi, K. Laine, B. Pelton, and W. Dai, “HEAX: High-performance architecture for computation on homomorphically encrypted data in the cloud,” Cryptology ePrint Archive, Report 2019/1066, 2019, https://eprint.iacr.org/2019/1066.
- [7] A. C. Mert, E. Karabulut, E. ¨ Ozt¨ urk, E. Savas, and A. Aysu, “An extensive study of flexible design methods for the number theoretic transform,” IEEE Trans. Computers, vol. 71, no. 11, pp. 2829–2843, 2022. [Online]. Available: https://doi.org/10.1109/TC.2020.3017930
- [8] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, “Compact ring-LWE based cryptoprocessor,” Cryptology ePrint Archive, Report 2013/866, 2013, https://eprint.iacr.org/2013/866.
- [9] T. Fritzmann, G. Sigl, and J. Sep´ ulveda, “RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography,” Cryptology ePrint Archive, Report 2020/446, 2020, https://eprint.iacr.org/2020/446.
- [10] S.-Y. Wu, K.-Y. Chen, and M.-D. Shieh, “Efficient vlsi architecture of bluestein’s fft for fully homomorphic encryption,” in 2022 IEEE International Symposium on Circuits and Systems (ISCAS), 2022, pp. 2242–2245.