ĐẠI HỌC QUỐC GIA TP. HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

NGUYỄN TUẤN HÙNG

THIẾT KẾ PHẦN CỨNG XỬ LÝ NTT VÀ INTT CHO MÃ HOÁ LƯỢNG TỬ CRYSTALS-KYBER

Chuyên ngành : Điện Tử

Mã số : 8520203

LUẬN VĂN THẠC SĨ

TP. HỒ CHÍ MINH, tháng 01 năm 2022

ĐẠI HỌC QUỐC GIA TP.HCM **TRƯỜNG ĐẠI HỌC BÁCH KHOA**	**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM Độc lập - Tự do - Hạnh phúc**

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên: Nguyễn Tuấn Hùng ..... MSHV: 2070027

Ngày, tháng, năm sinh: 11/06/1997..... Nơi sinh: TP.HCM

Chuyên ngành: Điện Tử ..... Mã số : 8520203

I. TÊN ĐỀ TÀI:

THIẾT KẾ PHẦN CỨNG XỬ LÝ NTT VA INTT CHO MÃ HÓA LƯỢNG TỬ CRYSTALS-KYBER

II. NHIỆM VỤ VÀ NỘI DUNG:

Xây dựng phần cứng xử lý NTT và INTT trên nền tảng FPGA, bằng ngôn ngữ mô tả phần cứng Verilog, cho mã hóa lượng tử CRYSTALS-Kyber. Qua đó, đánh giá và bàn luận các kết quả cũng như bàn luận về các hướng nghiên cứu chuyên sâu để ứng dụng mã hóa lượng tử trên phần cứng.

III. NGÀY GIAO NHIỆM VỤ : (Ghi theo trong QĐ giao đề tài)

IV. NGÀY HOÀN THÀNH NHIỆM VỤ: (Ghi theo trong QĐ giao đề tài)

V. CÁN BỘ HƯỚNG DẪN (Ghi rõ học hàm, học vị, họ, tên):

Tp. HCM, ngày . . . . tháng .. . . năm 20....

CÁN BỘ HƯỚNG DẪN (Họ tên và chữ ký)	**CHỦ NHIỆM BỘ MÔN ĐÀO TẠO** (Họ tên và chữ ký)

TRƯỞNG KHOA……

(Họ tên và chữ ký)

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC BÁCH KHOA –ĐHQG -HCM

Cán bộ hướng dẫn khoa học : .....

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 1 : .....

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Cán bộ chấm nhận xét 2 : .....

(Ghi rõ họ, tên, học hàm, học vị và chữ ký)

Luận văn thạc sĩ được bảo vệ tại Trường Đại học Bách Khoa, ĐHQG Tp. HCM ngày . . . . . tháng . . . . năm . . . . .

Thành phần Hội đồng đánh giá luận văn thạc sĩ gồm:

(Ghi rõ họ, tên, học hàm, học vị của Hội đồng chấm bảo vệ luận văn thạc sĩ)

1. .....
2. .....
3. .....
4. .....
5. .....

Xác nhận của Chủ tịch Hội đồng đánh giá LV và Trưởng Khoa quản lý chuyên ngành sau khi luận văn đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG **TRƯỞNG KHOA**

LỜI CẢM ƠN

Học viên xin gửi lời cảm ơn sâu sắc đến quý thầy cô, cha mẹ và bạn bè đã giúp đỡ và ủng hộ trong suốt quá trình hoàn thành luận văn. Sự giúp đỡ đó có ý nghĩa tinh thần rất lớn đối với học viên.

Xin được gửi lời cảm ơn chân thành đến thầy TS. Trần Hoàng Linh. Người hết lòng giúp đỡ, dạy bảo, và tạo điều kiện thuận lợi cho học viên trong suốt quá trình thực hiện luận văn tốt nghiệp. Cảm ơn thầy đã định hướng, góp ý và giúp em có thể hoàn thành luận văn một cách tốt nhất.

Xin cảm ơn các quý thầy cô trong Khoa Điện – Điện tử, trường Đại học Bách Khoa Thành phố Hồ chí Minh. Kết quả em đạt được ngày hôm nay đều nhờ sự chỉ dạy của các thầy cô.

Tp. Hồ Chí Minh, ngày 05 tháng 12 năm 2021.

Học viên

Nguyễn Tuấn Hùng

TÓM TẮT LUẬN VĂN

Vòng thứ 3 của quá trình tiêu chuẩn hóa mật mã lượng tử NIST đã công bố CRYSTALS-Kyber là một trong những ứng viên lọt vào vòng chung kết. Là một dạng mã hóa dựa trên Lattice, CRYSTALS-Kyber đặt nặng yêu cầu xử lý vào NTT và INTT để nhân đa thức. Luận văn này trình bày một thiết kế phần cứng xử lý NTT và INTT cho mã hoá lượng tử CRYSTALS-Kyber. Trong đó tập trung vào việc thiết kế và tối ưu hóa kiến trúc bộ tăng tốc NTT với các thông số phù hợp cho CRYSTALS-Kyber. Nghiên cứu bao gồm việc sửa đổi các mô-đun tính toán và các Butterfly Unit nhằm giảm độ phức tạp tính toán. Kết quả là thiết kế đạt được 237 MHz *f*max khi tổng hợp trên Intel FPGA Cyclone V với Quartus. Việc cân bằng thanh ghi giữa các mạch tổ hợp cho phép thiết kế pipeline đạt được tốc độ mạch tối ưu.

ABSTRACT

NIST post-quantum cryptography standardization round 3 announced CRYSTALS-Kyber as one of the finalists. As a lattice-based cryptography scheme, CRYSTALS-Kyber relies heavily on polynomial multiplication efficiency. This paper presents a high-speed and pipelined hardware number theoretic transform (NTT) and INTT accelerator for CRYSTALS-Kyber. Our work centers around designing and optimizing the NTT accelerator architecture with suitable parameter for hardware implementations of CRYSTALS-Kyber. The work includes modifying the modular arithmetic modules and butterfly units with an efficient low complexity algorithm. As a result, our design achieved 237 MHz *f*max synthesized on Intel FPGA Cyclone V with Quartus. Resources utilization through combinational logic path rebalances allowed us to efficiently pipeline between hardware modules.

LỜI CAM ĐOAN

Học viên cam đoan rằng, ngoài trừ các kết quả tham khảo từ các công trình khác như đã ghi rõ và trích dẫn trong luận văn này, các công việc nghiên cứu và trình bày trong luận văn này là do chính học viên thực hiện

Học viên

Nguyễn Tuấn Hùng

MỤC LỤC

1. MỞ ĐẦU 1
1.1 Lý do chọn đề tài 1
1.2 Mục đích 2
1.3 Đối tượng và phạm vi nghiên cứu 2
1.3.1 Đối tượng nghiên cứu 2
1.3.2 Phạm vi nghiên cứu 2
1.4 Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu 2
1.4.1 Ý nghĩa khoa học 2
1.4.2 Ý nghĩa thực tiễn 3
2. TỔNG QUAN 3
2.1 Tình hình nghiên cứu trong và ngoài nước 3
2.1.1 Tình hình nghiên cứu ngoài nước 3
2.1.2 Tình hình nghiên cứu trong nước 4
2.2 Nhiệm vụ đề tài 4
3. NHỮNG NGHIÊN CỨU LÝ THUYẾT 5
3.1 Lý thuyết về mã hóa bất đối xứng 5
3.2 Lý thuyết về CRYSTALS-Kyber 6
3.3 Lý thuyết về Number Theoretic Transform (NTT) 10
3.3.1 NTT/INTT và độ phức tạp so với phép nhân đa thức 10
3.3.2 Phiên bản NTT/INTT tối ưu được sử dụng 11
3.4 Lý thuyết về phép toán rút gọn modulo Exact-KRED 15
3.5 Về bộ nhớ BRAM M10K trên FPGA Cyclone V 16
3.6 Xử lý tính toán lý thuyết trên phần mềm máy tính 17
4. TRÌNH BÀY, ĐÁNH GIÁ VÀ BÀN LUẬN KẾT QUẢ 18
4.1 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 18

4.1.1 Thiết kế butterfly unit xử lý CT/GS 19
4.1.1.1 Thiết kế bộ xử lý rút gọn modulo Exact-KRED 19
4.1.2 Bộ rút gọn modulo chia nửa 19
4.1.3 Thiết kế butterfly unit xử lý CT/GS 20
4.1.4 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 22
4.2 Kết quả tổng hợp và mô phỏng 32
4.2.1 Kết quả mô phỏng ModelSim 32
4.2.2 Kết quả tổng hợp Quartus 34
4.3 Đánh giá, bàn luận và so sánh kết quả 35
5. KẾT LUẬN VÀ KIẾN NGHỊ NHỮNG NGHIÊN CỨU TIẾP THEO 37
5.1 Các hướng tối ưu thiết kế phần cứng xử lý NTT và INTT 37
5.2 Thiết kế phần cứng xử lý CRYSTALS-Kyber 38
5.3 Kết luận 38
6. DANH MỤC CÔNG TRÌNH CÔNG BỐ CỦA TÁC GIẢ 39
7. DANH MỤC TÀI LIỆU THAM KHẢO 39
8. PHỤ LỤC 42
8.1 Các phần tối ưu cần lưu ý ở phần mềm Quartus 42
8.2 Sơ đồ thiết kế chính theo Quartus 43
8.3 Phần truy xuất RAM, ROM đầy đủ 43
8.4 Các thuật ngữ được sử dụng 55

DANH SÁCH HÌNH MINH HỌA

Hình 1 Minh họa mã hóa bất đối xứng [23] 6

Hình 2 Minh họa gốc toán học phức tạp của phép toán lưới mà Kyber dựa trên 7

Hình 3 Ring-learning with errors (RLWE) 7

Hình 4 Module-learning with errors (MLWE) 7

Hình 5 Quy trình tạo khóa, mã hóa và giải mã của dạng mã hóa lưới 8

Hình 6 Ba giải thuật tao mã, mã hóa và giải mã của Kyber 8

Hình 7 NTT trong giải thuật tạo mã của Kyber 9

Hình 8 NTT trong giải thuật mã hóa của Kyber 9

Hình 9 NTT trong giải thuật giải mã của Kyber 10

Hình 10 NWC NTT và INT với xử lý trước và xử lý sau 12

Hình 11 Mô hình của các Butterfly Unit (BU). (1) Cooley - Tukey Butterfly. (2) Gentleman - Sande Butterfly 13

Hình 12 Sơ đồ khối của bộ rút gọn Modulo Exact-KRED 19

Hình 13 Phần cứng butterfly unit xử lý CT/GS 21

Hình 14 Bộ Butterfly Unit kết hợp CT/GS cho thuật toán độ phức tạp thấp 22

Hình 15 Phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 23

Hình 16 Sơ đồ khối bộ xử lý NTT/INTT 25

Hình 17 Bộ SIPO Writeback vào nối tiếp ra song song với các ô nhớ 12-bit 26

Hình 18 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho NTT (n=128, 40 cycles đầu tiên) 27

Hình 19 Thứ tự địa chỉ truy xuất tại ROM cho NTT (n=128, 40 cycles đầu tiên) 28

Hình 20 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho INTT (n=128, 40 cycles đầu tiên) 29

Hình 21 Thứ tự địa chỉ truy xuất tại ROM cho INTT (n=128, 40 cycles đầu tiên) 30

Hình 22 Mô phỏng trên dạng sóng kết quả của thiết kế Exact-KRED 32

Hình 23 Mô phỏng trên dạng sóng kết quả của thiết kế BU. 33

Hình 24 Mô phỏng dạng sóng kết quả của phần cứng xử lý NTT và INTT 33

Hình 25 Kết quả tổng hợp tài nguyên trên Quartus 34

Hình 26 Kết quả tốc độ mạch trường hợp Slow 1100 mV 85C 34

Hình 27 Kết quả tốc độ mạch trường hợp Fast 1100 mV 0C 35

Hình 28 Kết quả tốc độ mạch trung bình 35

Hình 29 Chế độ tối ưu khi tổng hợp trên Quartus 42

Hình 30 Sơ đồ Netlist Viewer của thiết kế tổng trên Quartus 43

DANH SÁCH BẢNG SỐ LIỆU

Bảng 1 Thông số của từng phiên bản Kyber 10

Bảng 2 Tìm giá trị $\gamma 2n = \omega n$, ... 17

Bảng 3 Thực hiện tính $\gamma 2n$ ở các giá trị mũ khác nhau (bảng tới mức mũ 24) 17

Bảng 4 Bảng chân phần cứng butterfly unit xử lý CT/GS 21

Bảng 5 Bảng chân phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 23

Bảng 6 Thiết kế đề xuất so với các nghiên cứu NTT tương tự trước đây (n = 256) 36

Bảng 7 Phần thứ tự truy xuất và thứ tự hệ số phương trình gốc trong bộ nhớ RAM cho NTT 43

Bảng 8 Phần thứ tự truy xuất và thứ tự hệ số phương trình gốc trong bộ nhớ RAM cho INTT 46

Bảng 9 Thứ tự truy xuất hệ số Twiddle Factor từ ROM cho cấu hình 2 x 2 BU cho NTT 49

Bảng 10 Bảng giá trị Twiddle Factor γ 52

Bảng 11 Bảng thuật ngữ 55

1. MỞ ĐẦU

1.1 Lý do chọn đề tài

Tổ chức National Institute of Standards and Technology (NIST) đã tổ chức thực hiện quy trình chuẩn hoá mã hoá sau lượng tử (Post-Quantumn Cryptography hay PQC) hay mã hoá lượng tử từ 2016 [1]. Đây là dạng mã hóa mới nhằm thay thế các dạng mã hóa bất đối xứng hiện tại.

NIST mô tả lý do thực hiện chuẩn hoá của họ là để tìm ra một loại mã hoá mới an toàn trước sự phát triển tương lai của máy tính lượng tử. Điều này dựa trên các nghiên cứu gần đây cho thấy mã hoá bất đối xứng hiện đại (ECDSA, RSA, …) đang được sử dụng sẽ không còn an toàn trước máy tính lượng tử. Các thông tin đang được mã hoá sẽ không còn bí mật trong tương lai gần. Nhu cầu phát triển chuẩn mã hoá để sử dụng cho thời kỳ hậu lượng tử hoá là rất cấp thiết.

Cho đến vòng 3 của quy trình chuẩn hoá mã hoá lượng tử, có 5 ứng cử viên được lựa chọn cho vòng tiếp theo. Trong đó bao gồm 4 ứng cử viên thuộc dòng mã hoá lưới (lattice-based) và 1 ứng cử viên thuộc dòng mã hoá code-based. Dòng mã hoá lattice-based đang chứng tỏ mình là một trong những dòng mã hoá của tương lai với khả năng bảo mật trước máy tính lượng tử và hiệu quả tính toán rất khả thi.

Trong 4 ứng cử viên mã hoá lattice-based, CRYSTALS-Kyber hay Kyber là ứng cử viên đầu tiên và được đánh giá rất triển vọng để được chuẩn hoá trong tương lai [2]. Kyber đòi hỏi nỗ lực tính toán nghiêm túc, chủ yếu là phép nhân các đa thức trên một vành đa thức có kích thước không đổi. Dạng mã hoá trên mạng mô-đun (module lattice) này mang lại sự cân bằng tốt giữa hiệu quả và bảo mật. Tuy nhiên, quá trình tạo, mã hóa và giải mã khóa có thể chiếm một tỷ lệ lớn trong khả năng tính toán và chu kỳ đồng hồ của bộ vi xử lý. Kyber sử dụng một kỹ thuật hỗ trợ tốc độ phép tinh nhân có tên là Number-Theoretic Transform (NTT) và chọn các tham số để hỗ trợ kỹ thuật này. Để triển khai Kyber một cách hiệu quả, việc tối ưu hóa NTT và NTT nghịch đảo (INTT) là rất quan trọng.

Nhiều nhà nghiên cứu triển khai mã hoá lattice-based trên phần mềm để chứng minh khái niệm cho nghiên cứu của họ và cũng một phần thể hiện tốc độ tính toán của giải thuật được nghiên cứu. Tuy vậy, khi có một dòng mã hoá được chuẩn hoá, các ứng dụng thực tế thường được tối ưu khi triển khai diện rộng bằng phần cứng. Tiêu biểu có thể kể đến AES-NI

[3], RSA/ECC co-processor [4],… Nhiều nghiên cứu về mã hóa lượng tử gần đây cũng được thực hiện trên ASIC, FPGA dưới dạng độc lập hoặc làm bộ xử lý phụ cho các CPU RISC V, ARM, x86,… Trong đó, các chip FPGA là nền tảng tốt để phát triển các nghiên cứu lý thuyết và đánh giá sức mạnh giải thuật.

Để thiết kế phần cứng trên FPGA, ngôn ngữ mô tả phần cứng được sử dụng rất phổ biến. Đây là phương pháp thiết kế mạch kỹ thuật số chính xác, giúp người thiết kế có được phần lớn quyền quyết định các kết quả thiết kế với công cụ từ các nhà phát triển FPGA như Intel hay Xillinx.

Trên đây là tổng quan về các lý do để học viên thực hiện đề tài nghiên cứu thiết kế phần cứng xử lý NTT và INTT trên nền tảng FPGA, bằng ngôn ngữ Verilog, cho mã hóa lượng tử CRYSTALS-Kyber.

1.2 Mục đích

Mục đích nghiên cứu là xây dựng phần cứng xử lý NTT và INTT trên nền tảng FPGA, bằng ngôn ngữ Verilog, cho mã hóa lượng tử CRYSTALS-Kyber. Qua đó, đánh giá và bàn luận các kết quả cũng như bàn luận về các hướng nghiên cứu chuyên sâu để ứng dụng mã hóa lượng tử trên phần cứng.

1.3 Đối tượng và phạm vi nghiên cứu

1.3.1 Đối tượng nghiên cứu

Đối tượng nghiên cứu của luận văn là thuật toán NTT và INTT với các thông số phù hợp với mã hóa lượng tử CRYSTALS-Kyber, sử dụng ngôn ngữ mô tả phần cứng Verilog và đánh giá trên nền tảng FPGA.

1.3.2 Phạm vi nghiên cứu

Phạm vi nghiên cứu của luận văn là thuật toán NTT và INTT với các thông số phù hợp với mã hóa lượng tử CRYSTALS-Kyber, tập trung chủ yếu vào việc tối ưu cách áp dụng NTT và INTT trên phần cứng qua ngôn ngữ mô tả phần cứng. Chip FPGA sử dụng để đánh giá trong đề tài là Intel Cyclone V 5CSXFC6D6F31C6.

1.4 Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu

1.4.1 Ý nghĩa khoa học

Đề tài đóng góp về ý nghĩa khoa học trong việc nghiên cứu cách ứng dụng hiệu quả của dạng mã hóa lượng tử lattice-based, vốn đang còn rất mới và vẫn đang được chuẩn hóa.

1.4.2 Ý nghĩa thực tiễn

Đề tài đóng góp về ý nghĩa thực tiễn trong quá trình chuẩn hóa mã hóa lượng tử để bảo mật thông tin kỹ thuật số trong tương lai. Ngoài ra, đề tài còn giúp đánh giá mức độ khả thi trong việc ứng dụng của mã hóa CRYSTALS-Kyber trên các thiết bị điện tử sau này. Ứng dụng này có thể dưới dạng tích hợp như một phần của vi xử lý chính của các thiết bị đó hoặc dưới dạng một dạng vi xử lý phụ độc lập.

2. TỔNG QUAN

2.1 Tình hình nghiên cứu trong và ngoài nước

2.1.1 Tình hình nghiên cứu ngoài nước

Trong các nghiên cứu trước đây, một trong những triển khai phần cứng hoàn toàn sớm nhất của mã hóa lượng tử [5] là dành cho một dạng mã hóa lượng tử lattice-based có tên Round5. Nghiên cứu này bao gồm thiết kế phần cứng của hàm băm SHA-3 Keccak, AES-GCM và thuật toán Round5. Sự đơn giản từ lợi thế của mô-đun Round5 được sử dụng hiệu quả trên phần cứng, cho thấy tương lai của việc ứng dụng mã hóa lượng tử trên phần cứng. Một phần để tiếp nối việc xử lý mã hóa trên phần cứng như trước đây với mã hóa hiện đại, một phần để giảm tải yêu cầu về tài nguyên xử lý cho các vi xử lý tác vụ chính.

Nghiên cứu [6] trình bày một thiết kế phần cứng đầy đủ tương tự của CRYSTALS-Kyber, giúp tăng tốc hiệu suất 129 lần so với việc triển khai bộ xử lý Cortex-M4 [7]. Trong [8], Jati và cộng sự trình bày thiết kế chống tấn công kênh ngoại (side-channel attack) đầu tiên của CRYSTALS-Kyber trên phần cứng với hiệu suất ấn tượng.

Zhao và cộng sự [9] phân tích mã phần mềm để tối ưu hóa thiết kế của chúng, mang lại kết quả tích cực. Các cách triển khai khác của Kyber khác nhau, từ bộ xử lý lai giữa phần cứng và phần mềm [10] [11] [12] đến phần cứng thuần túy [7] [8] [13] [14]. Các nghiên cứu gần đây về triển khai phần cứng CRYSTALS-Kyber hầu hết tối ưu hóa quá trình xử lý giải thuật NTT và INTT, do các phần còn lại của Kyber đều dựa trên mã hóa hiện đại thuần túy có thể tái sử dụng từ các nghiên cứu trước.

Trong các nghiên cứu đó, nhiều hướng tối ưu khác nhau được trình bày. Nghiên cứu [15] trình bày một kỹ thuật tính số dư modulo mới được gọi là K2-RED cho cấu trúc NTT của họ, giúp cải thiện hiệu suất đáng kể so với cách tính số dư bằng kỹ thuật Barret hay

Montgomery. Một kỹ thuật tính số dư modulo khác được sử dụng trong [16], [17] được sửa đổi với các hằng số được tính toán trước.

Zhang và cộng sự [18] trình bày sơ đồ truy cập bộ nhớ kiểu ping-pong để truy cập RAM hiệu quả. Đây cũng là một hướng tối ưu ứng dụng phần cứng của mã hóa lượng tử, giải tắc nghẽn ở phần truy cập bộ nhớ. Bên cạnh đó, Poppelmann và cộng sự [19] giới thiệu cấu trúc chủ-tớ cho nhiều bộ nhân đa thức kết hợp xử lý tính toán cho các phần cần sức mạnh xử lý tính toán lớn.

2.1.2 Tình hình nghiên cứu trong nước

Nghiên cứu về mã hóa lượng tử trong nước chưa có nhiều trong thời gian này. Tuy vậy, các nghiên cứu về phần cứng ứng dụng trên công nghệ FPGA và các nghiên cứu về thuật toán Fast Fourier Tranform đã phát triển từ nhiều năm trở lại đây.

Nghiên cứu [20] thiết kế hệ thống tính toán Fast Fourier Transform (FFT) 2048 điểm xây dựng trên nền tảng FPGA. Nghiên cứu phù hợp để tham khảo về FFT và cách thực hiện phép nhân hiệu quả trên FPGA. Nghiên cứu [21] cũng sử dụng bộ nhân CORDIC tương tự [20] để tính toán FFT, cấu trúc bộ nhân sử dụng thanh ghi dịch (bộ nhân xoay góc thích nghi) cũng là một hướng tối ưu để tham khảo. Nghiên cứu [22] thể hiện một ứng dụng thực tiễn của việc sử dụng biến đổi Fourier nhanh (FFT). Biến đổi Fourier là một giải thuật quan trọng sử dụng trong nhiều mặt của đời sống, trong xử lý tín hiệu, hình ảnh, … và giờ đây được ứng dụng trong mã hóa lượng tử.

2.2 Nhiệm vụ đề tài

Nhiệm vụ đề tài bao gồm các mục được sau đây, nhằm để thực hiện mục đích đề tài đã đề ra, kết quả cần đạt và giới hạn đề tài. Qua đó trình bày các kết quả nghiên cứu đã được thực hiện.

Nội dung 1: Tìm hiểu lý thuyết mã hóa bất đối xứng, mã hóa lượng tử, NTT, các thành phần toán học cần thiết để thiết kế phần cứng.

Nội dung 2: Xây dựng thiết kế phần cứng xử lý NTT và INTT cho mã hóa lượng tử CRYSTALS-Kyber, tổng hợp thiết kế và mô phỏng kiểm tra.

Nội dung 3: Đánh giá, bàn luận về thiết kế. So sánh kết quả đạt được với các nghiên cứu trước.

Nội dung 4: Bàn luận về hướng phát triển tương lai và kết luận cho nghiên cứu.

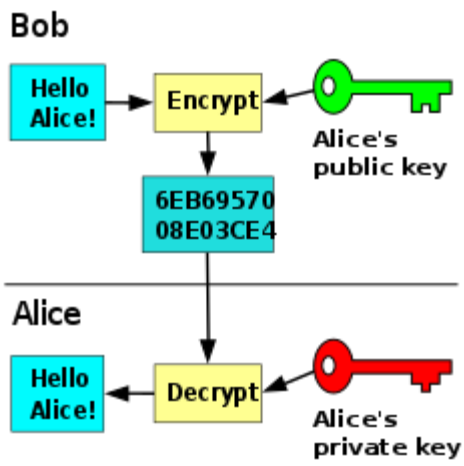
Trong đó, phần tiếp theo của luận văn sẽ thể hiện nội dung 1. Phần 4 của luận văn trình bày các phần được nêu trong nội dung 2. Ở phần 4 của luận văn, học viên trình bày nội dung 3. Học viên thực hiện kết luận và bàn luận hướng phát triển theo nội dung 4 ở phần 5 của luận văn. Phần 6 là danh mục các công trình nghiên cứu của học viên. Phần 7 là danh mục tài liệu tham khảo. Phần 8 là phụ lục, gồm các ghi chú về từ ngữ sử dụng trong luận văn và các thông tin thêm.

3. NHỮNG NGHIÊN CỨU LÝ THUYẾT

Trong phần này, học viên trình bày các nghiên cứu lý thuyết có liên quan đến đề tài. Qua đó lựa chọn các nền tảng lý thuyết cần để xây dựng phần cứng. Các nghiên cứu lý thuyết đi từ mã hóa bất đối xứng, mã hóa lượng tử CRYSTALS-Kyber, Number Theoretic Transform (NTT) đến các giải thuật chuyên sâu để tối ưu trên phần cứng tốt hơn.

3.1 Lý thuyết về mã hóa bất đối xứng

Mã hóa bất đối xứng là dạng mã hóa gồm 2 chìa khóa mã: chìa khóa công khai mà chìa khóa bí mật. Chìa khóa bí mật có thể dùng để mã hóa một văn bản mà chỉ chìa khóa công khai mới xác nhận và giải mã được hoặc một cơ chế tương tự ngược lại. Mã hóa bất đối xứng thường dựa trên một vấn đề toán học phức tạp chỉ có thể giải được một chiều. Có thể lấy ví dụ từ mã hóa RSA dựa trên độ phức tạp của việc phân tích kết quả mã hóa công khai (vốn là một hằng số mũ với chìa khóa bí mật) ra các thừa số.



Hình 1 Minh họa mã hóa bất đối xứng [23]

Hình 1 minh họa cấu trúc của một mô hình mã hóa bất đối xứng. Bao gồm một chìa khóa công khai và một chìa khóa bí mật, được giữ bởi 2 đầu của cuộc đối thoại, Bob và Alice, tương xứng.

3.2 Lý thuyết về CRYSTALS-Kyber

Lý thuyết về CRYSTALS-Kyber chủ yếu được trình bày bởi tác giả Avanzi và cộng sự tại [2] và tại trang web của CRYSTALS-Kyber cho những phiên bản mới nhất. Bởi vì CRYSTALS-Kyber là kiểu mã hóa lượng tử chưa chuẩn hóa và vẫn đang cập nhật liên tục. Phiên bản CRYSTALS-Kyber được lựa chọn trong nghiên cứu là phiên bản 3 [2].

CRYSTALS-Kyber là mô hình mật mã dựa trên bài toán Module Learning With Errors (MLWE [24]). Chi tiết về Kyber có thể được tìm thấy trong nghiên cứu ban đầu của Avanzi và cộng sự [25]. Kyber có hai phần: mã hóa công khai chống được phương pháp tấn công chọn văn bản hoặc CPAPKE, cơ chế đóng gói khóa bảo mật hoặc CCAKEM. CPAPKE được bao gồm trong CCAKEM như một bước bắt buộc để tạo mã khóa và mã hóa cũng như giải mã. Trong khi CPAPKE có ba bước khác nhau (tạo khóa, mã hóa và giải mã), cả ba bước đều yêu cầu một bước nhân đa thức lớn có độ phức tạp $O(n^2)$. Về bản chất, CRYSTALS-Kyber hiểu đơn giản là mã hóa bất đối xứng, bao gồm một phép toán có độ phức tạp lớn (MLWE) và giao thức trao đổi kết quả từ phép toán đó. Hình 2 trình bày độ phức tạp của phép toán trên lưới mà Kyber dựa trên, đơn giản hóa. Bài toán cho ma trận A thuộc miền giới hạn Z_q, nhiễu e thuộc phân phối nhiễu X và t, tìm s. Các phương pháp được trình bày sau nhằm để tinh chỉnh độ phức tạp và hiệu quả của giải thuật.

Hình 2 Minh họa gốc toán học phức tạp của phép toán lưới mà Kyber dựa trên

Hình 3 thể hiện Ring-learning with errors (RLWE), một cách để giảm lưu trữ n^2 phần tử ma trận xuống còn n phần tử ma trận. Đây là tiền đề của MLWE của Kyber.

$$\begin{pmatrix} a_{0,0} & -a_{n,0} & -a_{n-1,0} & \cdots & -a_{1,n} \\ a_{1,0} & a_{0,0} & -a_{n,0} & \cdots & \\ a_{2,0} & a_{1,0} & a_{0,0} & \cdots & \\ a_{3,0} & a_{2,0} & a_{1,0} & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & & & & \\ a_{n,0} & & & \cdots & a_{0,0} \end{pmatrix} \times \begin{pmatrix} s_0 \\ s_1 \\ \\ \vdots \\ \\ s_n \end{pmatrix} + \begin{pmatrix} e_0 \\ e_1 \\ \\ \vdots \\ \\ e_n \end{pmatrix} = \begin{pmatrix} t_0 \\ t_1 \\ \\ \vdots \\ \\ t_n \end{pmatrix}$$

Hình 3 Ring-learning with errors (RLWE)

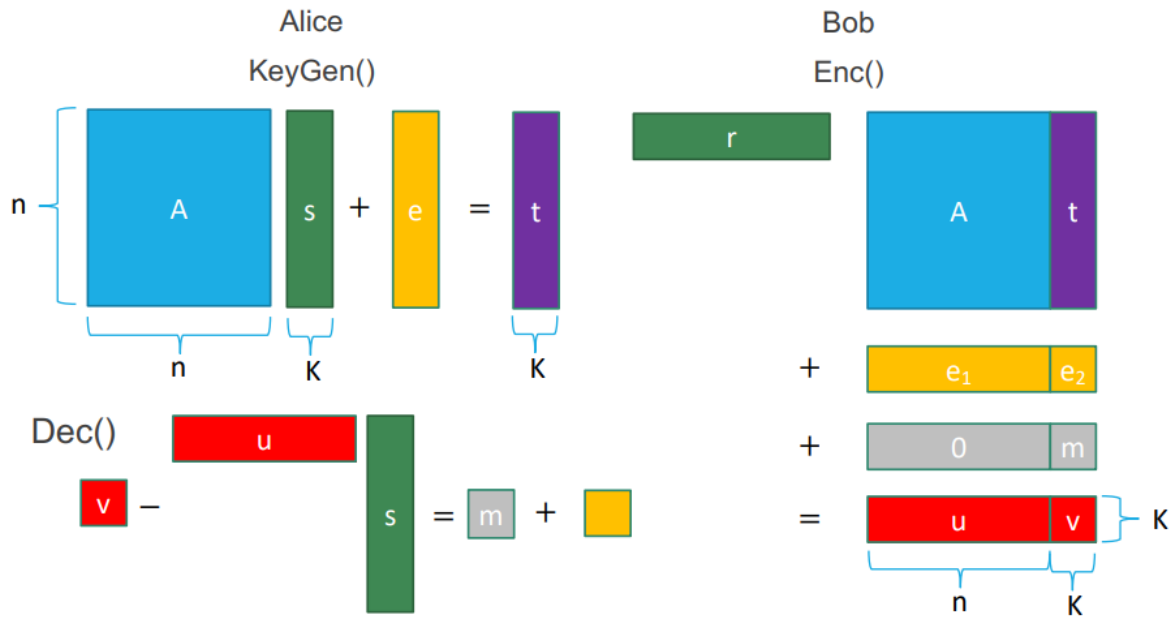
Hình 4 là dạng Module-learning with errors (MLWE) mà Kyber sử dụng. Kyber dùng thông số k để nâng độ phức tạp của ma trận A, tăng tính linh hoạt cho giải thuật.

$$\begin{pmatrix} A_{0,0}(X) & \cdots & A_{0,k}(X) \\ \vdots & \ddots & \vdots \\ A_{k,0}(X) & \cdots & A_{k,k}(X) \end{pmatrix} \times \begin{pmatrix} s_0(X) \\ \vdots \\ s_k(X) \end{pmatrix} + \begin{pmatrix} e_0(X) \\ \vdots \\ e_k(X) \end{pmatrix} = \begin{pmatrix} t_0(X) \\ \vdots \\ t_k(X) \end{pmatrix}$$

Hình 4 Module-learning with errors (MLWE)

Cơ chế giao tiếp, đóng gói và chuyển đổi từ văn bản gốc m sang văn bản đã được mã hóa c cũng được thể hiện trong hình 5. Với Alice và Bob là hai đầu thực hiện giao tiếp mã hóa Kyber. Alice tạo chìa khóa mã (bao gồm chìa khóa công khai A và chìa khóa bí mật s). Bob sử

dụng chìa khóa công khai để mã hóa văn bản gốc m. Alice giải mã từ văn bản đã mã hóa u, v về lại m từ chìa khóa riêng tư s.



Hình 5 Quy trình tạo khóa, mã hóa và giải mã của dạng mã hóa lưới

Kyber dựa trên rất nhiều phép nhân đa thức. Trong ngôn ngữ lập trình thông thường, các đa thức của Kyber được thể hiện dưới dạng các hệ số của đa thức, sắp xếp theo bậc. Ba giải thuật tao mã, mã hóa và giải mã của Kyber thể hiện ở hình 6.

Algorithm 1 Kyber.CPA.KeyGen(): key generation

1: $\rho, \sigma \leftarrow \{0,1\}^{256}$
2: $\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
3: $(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \mathsf{Sam}(\sigma)$
4: $\mathbf{t} := \mathsf{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$
5: **return** $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algorithm 2 Kyber.CPA.Enc($pk = (\mathbf{t}, \rho), m \in \mathcal{M}$): encryption

1: $r \leftarrow \{0,1\}^{256}$
2: $\mathbf{t} := \mathsf{Decompress}_q(\mathbf{t}, d_t)$
3: $\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
4: $(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \mathsf{Sam}(r)$
5: $\mathbf{u} := \mathsf{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
6: $v := \mathsf{Compress}_q\left(\mathbf{t}^T\mathbf{r} + e_2 + \lceil \frac{q}{2} \rfloor \cdot m, d_v\right)$
7: **return** $c := (\mathbf{u}, v)$

Algorithm 3 Kyber.CPA.Dec($sk = \mathbf{s}, c = (\mathbf{u}, v)$): decryption

1: $\mathbf{u} := \mathsf{Decompress}_q(\mathbf{u}, d_u)$
2: $v := \mathsf{Decompress}_q(v, d_v)$
3: **return** $\mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

Hình 6 Ba giải thuật tao mã, mã hóa và giải mã của Kyber

Một cách rất hiệu quả để xử lý các phép nhân đa thức là Number Theoretic Transform (NTT). Vì lý do đó, NTT được bao gồm trong định nghĩa của Kyber, và các thông số của các phiên bản Kyber được tinh chỉnh để được tính toán hiệu quả với NTT. Hình 7,8,9 thể hiện vai trò của NTT xuất hiện trong giải thuật tạo khóa, mã hóa và giải mã chi tiết của Kyber.

Algorithm 4 KYBER.CPAPKE.KeyGen(): key generation

Output: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Output: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$

```
d ← B^32
(ρ, σ) := G(d)
N := 0
for i from 0 to k − 1 do                    ▷ Generate matrix Â ∈ R_q^{k×k} in NTT domain
    for j from 0 to k − 1 do
        Â[i][j] := Parse(XOF(ρ, j, i))
    end for
end for
for i from 0 to k − 1 do                    ▷ Sample s ∈ R_q^k from B_η1
    s[i] := CBD_η1(PRF(σ, N))
    N := N + 1
end for
for i from 0 to k − 1 do                    ▷ Sample e ∈ R_q^k from B_η1
    e[i] := CBD_η1(PRF(σ, N))
    N := N + 1
end for
ŝ := NTT(s)
ê := NTT(e)
t̂ := Â ∘ ŝ + ê
pk := (Encode_12(t̂ mod^+ q)‖ρ)              ▷ pk := As + e
sk := Encode_12(ŝ mod^+ q)                  ▷ sk := s
return (pk, sk)
```

Hình 7 NTT trong giải thuật tạo mã của Kyber

Algorithm 5 KYBER.CPAPKE.Enc(pk, m, r): encryption

Input: Public key $pk \in \mathcal{B}^{12 \cdot k \cdot n/8+32}$
Input: Message $m \in \mathcal{B}^{32}$
Input: Random coins $r \in \mathcal{B}^{32}$
Output: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$

```
N := 0
t̂ := Decode_12(pk)
ρ := pk + 12 · k · n/8
for i from 0 to k − 1 do                    ▷ Generate matrix Â ∈ R_q^{k×k} in NTT domain
    for j from 0 to k − 1 do
        Â^T[i][j] := Parse(XOF(ρ, i, j))
    end for
end for
for i from 0 to k − 1 do                    ▷ Sample r ∈ R_q^k from B_η1
    r[i] := CBD_η1(PRF(r, N))
    N := N + 1
end for
for i from 0 to k − 1 do                    ▷ Sample e1 ∈ R_q^k from B_η2
    e1[i] := CBD_η2(PRF(r, N))
    N := N + 1
end for
e2 := CBD_η2(PRF(r, N))                     ▷ Sample e2 ∈ R_q from B_η2
r̂ := NTT(r)
u := NTT^−1(Â^T ∘ r̂) + e1                   ▷ u := A^T r + e1
v := NTT^−1(t̂^T ∘ r̂) + e2 + Decompress_q(Decode_1(m), 1)   ▷ v := t^T r + e2 + Decompress_q(m, 1)
c1 := Encode_du(Compress_q(u, d_u))
c2 := Encode_dv(Compress_q(v, d_v))
return c = (c1‖c2)                          ▷ c := (Compress_q(u, d_u), Compress_q(v, d_v))
```

Hình 8 NTT trong giải thuật mã hóa của Kyber

Algorithm 6 KYBER.CPAPKE.Dec(sk, c): decryption

Input: Secret key $sk \in \mathcal{B}^{12 \cdot k \cdot n/8}$
Input: Ciphertext $c \in \mathcal{B}^{d_u \cdot k \cdot n/8 + d_v \cdot n/8}$
Output: Message $m \in \mathcal{B}^{32}$

1: $\mathbf{u} := \mathsf{Decompress}_q(\mathsf{Decode}_{d_u}(c), d_u)$
2: $v := \mathsf{Decompress}_q(\mathsf{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
3: $\hat{\mathbf{s}} := \mathsf{Decode}_{12}(sk)$
4: $m := \mathsf{Encode}_1(\mathsf{Compress}_q(v - \mathsf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ \mathsf{NTT}(\mathbf{u})), 1))$ ▷ $m := \mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1))$
5: **return** m

Hình 9 NTT trong giải thuật giải mã của Kyber

Bảng 1 thể hiện thông số của từng phiên bản Kyber, các thông số này đã được tinh chỉnh để phù hợp với việc sử dụng NTT. Phiên bản được chú trọng nghiên cứu tập trung cho thông số n = 256 của Kyber.

Bảng 1 Thông số của từng phiên bản Kyber

Version	n	k	q	η_1	η_2	(d_u,d_v)	δ
Kyber512	256	2	3329	3	2	(10,4)	2^{-139}
Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Kyber1024	256	4	3329	2	2	(11,5)	2^{-174}

Như vậy, NTT đóng vai trò rất quan trọng trong thuật toán mã hóa CRYSTALS-Kyber. Phần tiếp theo trình lý thuyết NTT, INTT và giải thuật NTT, INTT được sử dụng để nghiên cứu.

3.3 Lý thuyết về Number Theoretic Transform (NTT)

Như đã đề cập ở trên, yếu tố thiết yếu của mã hóa CRYSTALS-Kyber là NTT và nghiên cứu [25] mô tả chi tiết về phép nhân đa thức trong Kyber và vị trí cần sử dụng các thuật toán NTT và INTT. Trong đó NTT là phép biến đổi Number Theoretic Transform và INTT hay NTT^{-1} là phép biến đổi ngược Inverse Number Theoretic Transform.

3.3.1 NTT/INTT và độ phức tạp so với phép nhân đa thức

Để biểu thị một đa thức trên máy tính, dạng biểu diễn coefficient representation hay biểu diễn hệ số được sử dụng. Dạng hiển thị này lưu các hệ số của phương trình dưới dạng dãy hệ số trên máy tính như sau:

$$C = 2 + 3x + 4x^2 \rightarrow C = [2,3,4]$$

Một dạng biểu diễn khác của đa thức là dạng biểu diễn giá trị (value representation), với đa thức bậc *d*, ta cần *d+1* điểm để biểu diễn đa thức dưới dạng này. Điểm đặc biệt của dạng biểu diễn này việc nhân đa thức trở nên dễ dàng hơn. Để chuyển một đa thức từ dạng hệ số sang dạng giá trị và ngược lại, kỹ thuật Fast Fourier Transform (FFT) và Inverse FFT (IFFT) được sử dụng. [26]

Kỹ thuật FFT và IFFT có độ phức tạp từ O($nlog(n)$). Đây là một cải thiện rất lớn so với độ phức tạp O(n^2) của Discrete Fourier Transform. [25,29]

Đối với trường hợp một trường giới hạn trong một vành như Kyber trong trường Z_q, việc thay đổi dạng biểu diễn của đa thức sử dụng kỹ thuật Number Theoretic Tranform (NTT) và Inverse NTT (NTT^{-1} hoặc INTT) [25].

Kết quả nhân đa thức $h = f \cdot g$ có thể được tính với NTT và INTT như phương trình bên dưới:

$$h = NTT^{-1}(NTT(f) \cdot NTT(g)) \quad (1)$$

Phiên bản cổ điển của NTT là phương trình như sau [32]:

$$\hat{a}_i = NTT_N(a)_i = \sum_{j=0}^{N-1} a_j \omega_N^{ij} \bmod q$$

$$i = 0, 1, \ldots, N - 1$$

Phiên bản cổ điển của INTT là phương trình như sau [32]:

$$a_i = INTT_N(\hat{a})_i = N^{-1} \sum_{j=0}^{N-1} \hat{a}_j \omega_N^{-ij} \bmod q$$

$$i = 0, 1, \ldots, N - 1$$

Với ω_n là primitive nth root of unity.

3.3.2 Phiên bản NTT/INTT tối ưu được sử dụng

Kyber có $n = 256$. Với các kết quả ma trận A hay phương trình h có 256 phần tử hệ số như (1), yêu cầu giải thuật xử lý NTT cho f và g từ 128 phần tử hệ số thành 256 phần tử hệ số với 128 phần tử hệ số đệm là 0. Để tránh phải đệm thêm các phần tử hệ số 0 vào f và g, cần sử dụng phiên bản thuật toán NTT có Negative Wrapped Convolution (NWC) [19].

Được mô tả bởi Poppelman và cộng sự trong [19] từ định lý toán học 1 và 2 như sau.

Định lý 1: Gọi ω là $2n$ −th primitive root of unity thuộc tập hữu hạn Zp. Cho $a = (a_0, \dots, a_{(n-1)})$và $b = (b_0, \dots, b_{(n-1)})$ là các vectơ có độ dài n với các phần tử thuộc tập hữu hạn Zp và $\breve{a} = (a_0, \dots, a_{(n-1)}, 0, \dots, 0), \breve{b} = (b_0, \dots, b_{(n-1)}, 0, \dots, 0)$ các vectơ tương ứng có độ dài $2n$, trong đó các thành phần theo sau là chứa đầy các số không. Với $\circ$ có nghĩa là phép nhân thành phần hệ số thì $a \cdot b = NTT_{\omega}^{-1}(NTT_{\omega}(\breve{a}) \circ NTT_{\omega}(\breve{b}))$.

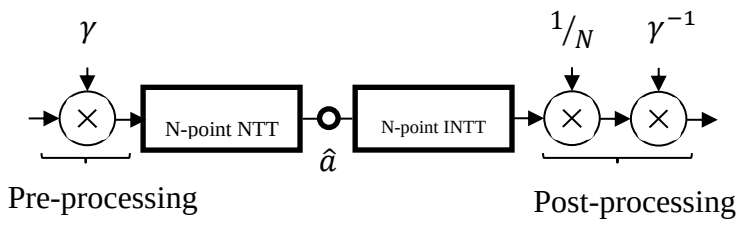
Định lý 2: Gọi ω là n-th primitive rooth of unity thuộc tập hữu hạn Zp và $\gamma^2 = \omega$. Gọi $a = (a_0, \dots, a_{(n-1)})$và $b = (b_0, \dots, b_{(n-1)})$ là các vectơ có độ dài n với các phần tử thuộc Zp.

1. Tích từng thành phần hệ số bọc dương của a và b là $NTT_{\omega}^{-1}(NTT_{\omega}(a) \circ NTT_{\omega}(b))$.

2. Gọi $d = (d_0, \dots, d_{(n-1)})$ là tích hệ số bọc âm của a và b. Cho $\bar{a}, \bar{b}$ và $\bar{d}$ được xác định là $(a_0, \gamma a_1, \dots, \gamma^{(n-1)} a_{(n-1)})$, $(b_0, \gamma b_1, \dots, \gamma^{(n-1)} b_{(n-1)})$ và $(d_0, \gamma d_1, \dots, \gamma^{(n-1)} d_{(n-1)})$. Khi đó $\bar{d} = NTT_{\omega}^{-1}(NTT_{\omega}(\bar{a}) * NTT_{\omega}(\bar{b}))$.

Ở bài nghiên cứu này, ta định nghĩa với ω_n là primitive nth root of unity và $\gamma = \sqrt{\omega_n}$. Vì Kyber số nguyên tố q thỏa mãn $q \equiv 1 \; mod \; 2n$ và với Kyber thuộc vành đa thức R_q là vành $Z_q[X]/(X^n + 1)$, kết quả NTT/INTT của Kyber có thể được tính bằng tích hệ số bọc âm tức phần 2 của định lý 2 (NWC).

Chúng ta cần áp dụng quy trình xử lý trước và xử lý sau cho các hệ số đa thức đầu vào và đầu ra, nhân chúng với γ và γ^{-1}tương ứng như được mô tả trong hình 10. Qua đó sẽ loại bỏ được các thành phần γ, đưa bài toán trở về lại kết quả $d = NTT_{\omega}^{-1}(NTT_{\omega}(a) \circ NTT_{\omega}(b))$.



Hình 10 NWC NTT và INT với xử lý trước và xử lý sau

Phiên bản NWC của NTT là phương trình như sau :

$$\hat{a}_i = NTT_N(a)_i = \sum_{j=0}^{N-1} a_j \omega_N^{ij} \gamma_{2N}^{j} \; mod \; q$$

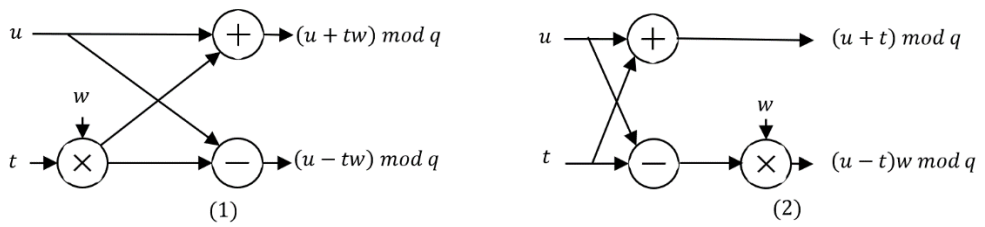
$$i = 0, 1, \ldots, N - 1$$

Phiên bản NWC của INTT là phương trình như sau :

$$a_i = INTT_N(\hat{a})_i = N^{-1} \gamma_{2N}^{-i} \sum_{j=0}^{N-1} \hat{a}_j \omega_N^{-ij} \; mod \; q$$

$$i = 0, 1, \ldots, N - 1$$

Để tính toán chi tiết NTT, các butterfly unit (BU) của Fast Fourier Transform (FFT) [25] [27] được sử dụng, trong đó NTT được thực hiện với BU Cooley-Tuckey (CT) và INTT được thực hiện với BU Gentleman-Sande (GS). Các đơn vị BU CT và GS là các phương pháp hiệu quả để thiết kế NTT cho phần cứng có thể lập trình được. Hình 11 mô tả sơ đồ chung cho các BU này.



Hình 11 Mô hình của các Butterfly Unit (BU). (1) Cooley - Tukey Butterfly. (2) Gentleman - Sande Butterfly

Cách tiếp theo để tối ưu hóa cho thuật toán NTT là thuật toán NTT / INTT độ phức tạp thấp từ nghiên cứu [32]. Độ phức tạp của thuận toán NTT cổ điển với NWC là $O\left({}^{N}/_{2} \log N + N\right)$, ${}^{N}/_{2} \log N$ là độ phức tạp của NTT và N bước nhân ban đầu với $\gamma_{2n} = \sqrt{\omega_n}$. Sử dụng kết quả của phương trình (3) từ [32] như sau:

$$\omega_m^j \gamma_{2m} \equiv \gamma_{2m}^{2j+1} \equiv \gamma_{2N}^{(2j+1)N/m} (3)$$

$$m = 2^1, 2^2, \ldots, N; \; j = 0, 1, \ldots, m/2 - 1$$

Sử dụng kết quả của (3) thay cho ω_n sẽ giảm bớt bước tiền xử lý xử lý và độ phức tạp của giải thuật NTT xuống còn $O\left({}^{N}/_{2}\log N\right)$.

Với INTT, để loại bỏ bước xử lý hậu kỳ nhân 1/N, [32] cũng đề xuất phương pháp chia nửa kết quả từ BU GS, qua đó bỏ qua được phần xử lý hậu kỳ 1/N, giảm độ phức tạp của INTT còn $O\left({}^{N}/_{2}\log N + N\right)$ như sau cho cấu hình xử lý với BU của Gentleman-Sande

$$a_{2i} = (N/2)^{-1}\gamma_{2N}^{-i}\sum_{j=0}^{N/2-1}\hat{b}_j^{(0)}\omega_{N/2}^{-ij}\ mod\ q$$

$$a_{2i+1} = (N/2)^{-1}\gamma_{2N}^{-i}\sum_{j=0}^{N/2-1}\hat{b}_j^{(1)}\omega_{N/2}^{-ij}\ mod\ q\ \ (4)$$

Với $\hat{b}_j^{(0)} = \frac{\hat{a}_j+\hat{a}_{(j+N/2)}}{2} mod\ q, \hat{b}_j^{(1)} = \frac{\hat{a}_j-\hat{a}_{(j+N/2)}}{2}\omega_N^{-j}\gamma_{2N}^{-1}$. Phương trình (4) cho thấy đây là phương trình tương tự NWC INTT ở trên, nhưng đã giảm xuống N/2 điểm thay vì N điểm.

Tương tự như NTT (3), độ phức tạp hiện tại của INTT là $O\left({}^{N}/_{2}\log N + N\right)$. Ta cũng có kết quả phương trình INTT gốc.

$$\omega_m^j\gamma_{2m}^{-1} \equiv \gamma_{2m}^{-(2j+1)} \equiv \gamma_{2N}^{-(2j+1)N/m}(5)$$

$$m = 2^1, 2^2, \ldots, N;\ j = 0, 1, \ldots, m/2\ -\ 1$$

Kết quả từ (5) cũng giúp giảm bước xử lý hậu kỳ nhân γ^{-1}. Qua đó giảm độ phức tạp của INTT còn $O\left({}^{N}/_{2}\log N\right)$.

Thuật toán 1 và 2 trình bày các phép toán NTT và INTT tương ứng sau khi đã tối ưu như trên. Đây là phiên bản cơ sở sử dụng để xây dựng thiết kế phần cứng. Bao gồm những sửa đổi nhỏ để phù hợp với thiết kế phần cứng, theo [5] [19] [32]. Thuật toán 1 trình bày giải thuật NTT độ phức tạp thấp bằng cách sử dụng BU Cooley - Tukey. Hàm đảo ngược bit bit-reverse là hàm đảo thứ tệ hệ số theo trình tự đảo bit cho các hệ số của đa thức. Thuật toán 2 trình bày giải thuật INTT sử dụng BU Gentleman – Sande.

Thuật toán 1. Low complexity NTT operation with Cooley – Tuckey butterfly
Input: polynomial $a(a_0, a_1, \ldots, a_{n-1})$ as $a(x) \in \mathbb{Z}_q[X]/(X_n + 1)$, $\omega_n \in \mathbb{Z}_q$ is the N-th primitive root of unity, $n = 2^l$ and $\gamma_{2n} = \sqrt{\omega_n}$ **Output:** $\hat{a} = NTT(a)$ 1: $\hat{a} = bit - reverse(a)$

```
for (i = 1; i < l; i++) do
    m = 2^i
    for (j = 0; j < m/2 − 1; j++) do
        ω ← γ_2N^((2j+1)N/m)
        for (k = 0; k < N/m − 1; k++) do
            u ← â[k + j]
            t ← ω · â[k + j + m/2] mod q
            â[k + j] = u + t mod q
            â[k + j + m/2] = u − t mod q
        end for
    end for
end for
return â
```

Thuật toán 2. Low complexity INTT operation with Gentleman – Sande butterfly

Input: polynomial $\hat{a}(a_0, a_1, \dots, a_{n-1})$ as $\hat{a}(x) \in \mathbb{Z}_q[X]/(X_n + 1)$, $\omega_n \in \mathbb{Z}_q$ is the n-th primitive root of unity, $n = 2^l$ and $\gamma_{2n} = \sqrt{\omega_n}$

Output: $a = INTT(\hat{a})$

```
for (i = 1; i < l; i++) do
    m = 2^(l−i)
    for (j = 0; j < m/2 − 1; j++) do
        ω ← γ_2N^(−(2j+1)N/m)
        for (k = 0; k < N/m − 1; k++) do
            u ← â[k + j]
            t ← â[k + j + m/2] mod q
            â[k + j] = (u+t)/2 mod q
            â[k + j + m/2] = (u−t)/2 · ω mod q
        end for
    end for
end for
a = bit − reverse(â)
return a
```

3.4 Lý thuyết về phép toán rút gọn modulo Exact-KRED

Phiên bản rút gọn modulo được sử dụng là một phiên bản cải tiến từ thiết kế KRED, K-RED-2x [23] và K^2-RED [13], được gọi là Exact-KRED. KRED là một tùy biến của bộ rút gọn modulo Montgomery nhưng được thiết kế để hoạt động cho một dạng modulus đặc biệt được mô tả trong phương trình sau:

$$modulus\ q = k * 2^m + 1 \quad (5)$$

Với Kyber có modulus là số nguyên tố q = 3329 thỏa (5), k = 13 và m = 8. Sử dụng KRED, K-RED-2x trên một số đầu vào C cho kết quả là $k \cdot C \ mod \ q$. Sử dụng K^2-RED tạo ra kết quả $k^2 \cdot C \ mod \ q$. Nghiên cứu [13] cho thấy K^2-RED tiết kiệm bộ nhớ hơn và phù hợp với mô-đun 12-bit q hơn so với K-RED-2x [23]. Đơn vị mô-đun được thực hiện sau bước nhân trong BU. Đề bù phần dư k^2 trong kết quả C'', cần xử lý trước ω thành $\omega' = (k^{-2} \cdot \omega) \ mod \ q$.

Thuật toán 3 trình bày thuật toán Exact-KRED cho Kyber. Trong đó, *zero-extend* và *signed-extend* là chức năng mở rộng theo độ dài bit với phần đệm tương ứng là 0 và theo dấu của số nhị phân. Chiều dài bit của phần mở rộng được ghi chú sau hàm. Trong [20], giải thuật KRED nói rằng có những trường hợp kết quả bị tràn với một số giá trị nhất định. Trong quá trình triển khai K^2-RED, kết quả tràn khi C'' lớn hơn modulus q. Tình trạng này xảy ra do C'' lúc đó là số âm và hiển thị dưới dạng một số không dấu công $2^{12} > q$. Để khắc phục tình trạng này, một phép cộng có điều kiện cho C'' được thêm vào sau dòng 6 của thuật toán 3.

Algorithm 3. Exact-KRED for Kyber NTT/INTT accelerator

Input: q modulus = 3329, binary number C = $(C_{23}, \dots, C_1, C_0)$, k = 13, m = 8
Output: $S = (k^2 * C) \ mod \ q$
1: Cl ← zero-extend$(C_7, \dots, C_1, C_0)$ to 16-bit
2: Ch← $(C_{23}, \dots, C_9, C_8)$
3: C’ = ((Cl<<3) – Ch) + (Cl<<2+Cl)
4: Cl’← zero-extend$(C'_7, \dots, C'_1, C'_0)$ to 12-bit
5: Ch’←signed-extend$(C'_{23}, \dots, C'_9, C'_8)$ to 12-bit
6: C’’ = ((Cl’<<3) – Ch’) + (Cl’<<2+Cl’)
7: **If** C’’ ≥ q
8: S ← C’’ + q
9: **Else**
10: S ← C’’
11: **return** S

3.5 Về bộ nhớ BRAM M10K trên FPGA Cyclone V

Bộ nhớ BRAM M10K trên FPGA Cyclone V [28] cho nhiều chế độ sử dụng từ Simple Dual Port tới True Dual Port. Với việc nhắm đến thực hiện cấu hình BU 2x2 với 2 đầu vào 24-bit cho 2 BU, chế độ Simple Dual Port là đủ để phục vụ nhu cầu.

Một điều lưu ý như nghiên cứu [29] nói về việc sử dụng BRAM M10K, quá trình tạo BRAM này bằng ngôn ngữ mô tả phần cứng hay trình tạo IP từ Quartus đều phải thực hiện đặt thanh ghi đầu ra cho BRAM.

Số lượng lớp thanh ghi đầu ra tối ưu cho thiết kế là từ 2 đến 3 lớp thanh ghi [29]. Số lượng lớp thanh ghi này đủ để phần mềm Quartus sắp xếp được đường đi cho dữ liệu từ BRAM đến mạch tổ hợp tiếp theo với tốc độ tốt nhất. Số lượng lớp thanh ghi nhiều hơn cho thấy không cần thiết do quá giới hạn của 3 tầng logic của chip, trừ một số trường hợp đặt biệt.

3.6 Xử lý tính toán lý thuyết trên phần mềm máy tính

Phần giải thuật NTT/INTT và các Twiddle Factor được xử lý và tính toán trước trên máy tính bằng phần mềm Microsoft Excel. Trong đó có các bước:

- Tìm giá trị ω_n (với Wolfram) và $\gamma_{2n} = \sqrt{\omega_n}$, $\omega' = (k^{-2} \cdot \omega)\ mod\ q$.
- Chuyển đổi từ công thức cấu hình 1 BU sang 2 x 2 BU
- Sắp xếp vị trí đầu vào và thứ tự truy xuất bộ nhớ.

  Với phiên bản $n = 128$ sẽ thiết kế và đánh giá, tính được giá trị

$$\omega_n = 33^{128} mod\ 3329 = 1$$

  Dựa trên điều kiện NWC hay công thức Euler, γ_{2n} có tồn tại . Để tìm $\gamma_{2n} = \sqrt{\omega_n}$, thực hiện bảng tính Excel cho ra giá trị như sau:

Bảng 2 Tìm giá trị $\gamma_{2n} = \sqrt{\omega_n}$,

STT	Cal	Result	
	…	…	NO
m	892	33	YES
33	893	1818	NO
N	894	276	NO
128	…	…	NO
Q	2436	1818	NO
3329	2437	33	YES
Gamma	2438	1579	NO
892	2439	3127	NO

Với 2 giá trị $\gamma_{2n} = \sqrt{\omega_n}$ tìm được là 892 và 2437. Ở các phần tính toán tiếp theo, thiết kế chỉ sử dụng giá trị nhỏ hơn là 892. Bảng 3 trình bày tới số thứ tự mũ 24 trên 127 (từ 0 đến 127)

Bảng 3 Thực hiện tính γ_{2n} ở các giá trị mũ khác nhau (bảng tới mức mũ 24)

STT	γ_{2n}	γ_{2n}^{-1}	$k^{-2}\gamma_{2n}$	$k^{-2}\gamma_{2n}^{-1}$
0	1	1	2285	2285
1	892	2549	872	2044
2	1848	2304	1508	1491

3	1455	723	2333	871
4	1432	630	3042	1422
5	1041	1212	1779	3021
6	1052	2617	282	961
7	1239	2308	1465	644
8	1089	2094	1602	1017
9	1868	319	602	3193
10	1795	1977	247	3321
11	554	2662	870	587
12	2760	1919	1474	622
13	314	1092	1755	1799
14	2009	1435	3203	3239
15	863	1651	1187	778
16	1355	2447	205	2004
17	3061	2298	156	1097
18	2102	331	2652	652
19	2444	2761	1807	430
20	1600	1729	758	2571
21	568	885	2899	1522
22	2998	1227	2677	677
23	1031	268	2232	3173
24	882	1974	1325	3124

Ngoài ra, để sắp xếp vị trí Twiddle Factor và thứ tự truy xuất bộ nhớ, các thứ tự này cũng được sắp xếp sẵn trên Excel. Phần phụ lục 8.3 trình bày các bản số liệu được tính toán đầy đủ.

4. TRÌNH BÀY, ĐÁNH GIÁ VÀ BÀN LUẬN KẾT QUẢ

4.1 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber

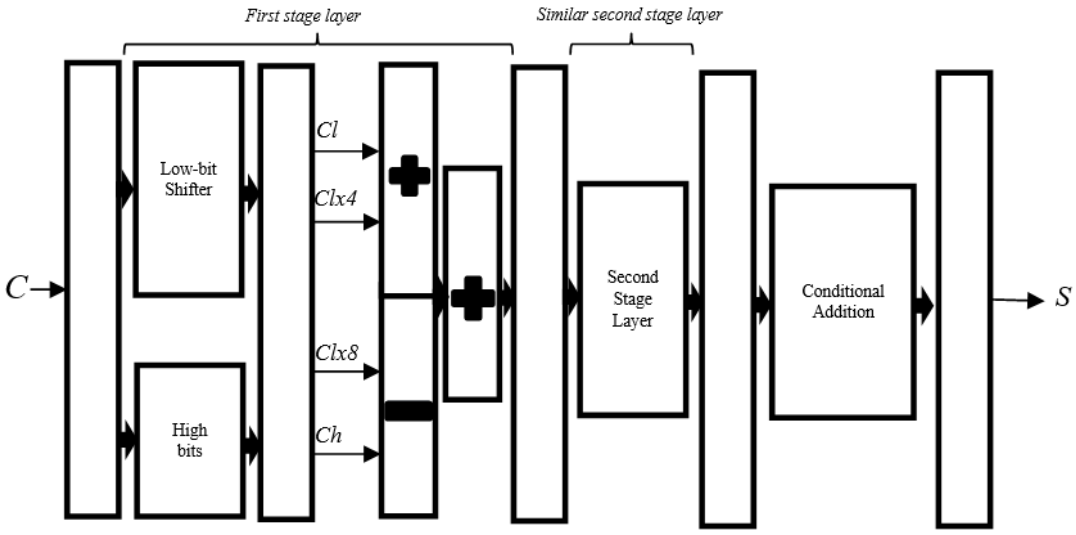
Chúng tôi mô tả thiết kế phần cứng của bộ gia tốc NTT và INTT đa thức cho Kyber trong hai phần. Đầu tiên, chúng tôi trình bày cấu trúc phần cứng của đơn vị cánh bướm đã sửa đổi được điều chỉnh theo thuật toán 1 và 2. Sau đó, chúng tôi trình bày kiến trúc tổng thể của bộ gia tốc với cấu hình đơn vị cánh bướm 2x2 và phần cứng giảm mô-đun được cải tiến.

4.1.1 Thiết kế butterfly unit xử lý CT/GS

4.1.1.1 Thiết kế bộ xử lý rút gọn modulo Exact-KRED

Bộ rút gọn modulo Exact-KRED được thiết kế dành cho $q = 3329$ của Kyber. Bao gồm 5 tầng pipeline xử lý rút gọn modulo liên tục. Chia ra làm 3 quy trình như đã trình bày ở thuật toán 3. Quy trình đầu tiên bao gồm xử lý đầu vào 24-bit từ bộ nhân DSP sang ngõ ra 16-bit. Quy trình thứ hai lặp lại xử lý ngõ vào 16-bit thành ngõ ra 12-bit. Quy trình cuối cũng là bộ cộng có điều kiện để xử lý các trường hợp kết quả bị tràn. Bộ cộng có điều kiện này với q có độ dài 12-bit cho mức tiêu thụ tài nguyên thấp, chỉ gồm một bộ cộng, bộ so sánh và một bộ mux lựa chọn điều kiện.

Hình 12 là sơ đồ khối bộ rút gọn modulo Exact-KRED. Bộ rút gọn modulo qua mỗi quy trình phân nhánh ngõ vào thành các dải bit cao và bit thấp. Dải bit thấp kèo theo bộ dịch để thực hiện *zero-extend* và *signed-extend.* Quy trình đầu sử dụng 2 clock, quy trình sau tương tự sử dụng tiếp tục 2 clock và bộ cộng có điều kiện kiểm tra kết quả trước khi pipeline sang ngõ ra.



Hình 12 Sơ đồ khối của bộ rút gọn Modulo Exact-KRED

4.1.2 Bộ rút gọn modulo chia nửa

Theo thuật toán 2, kết quả của bộ GS cần chia nửa (modulus q) [22]. Thực hiện chia nửa một số theo modulo q có hai trường hợp. Kết quả ngõ vào của bộ chia nửa này đã là một

số modulo q. Với số chẵn, chỉ cần dịch phải để ra được kết quả. Với số lẻ, kết quả chia nửa modulo q được tính như sau:

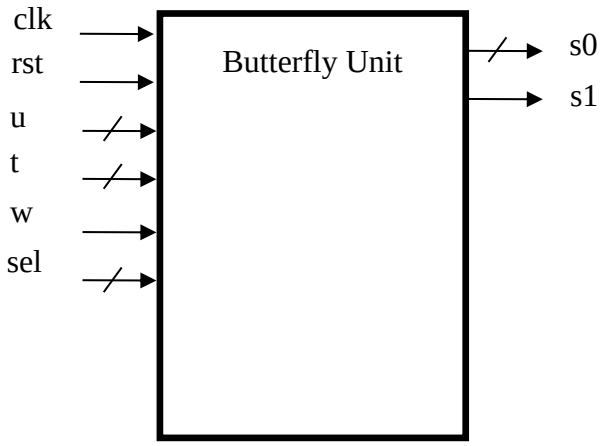
$$\frac{x}{2} mod\ q \equiv \left(2 * \left\lfloor\frac{x}{2}\right\rfloor + 1\right)\frac{q+1}{2} \equiv \left\lfloor\frac{x}{2}\right\rfloor (q+1) + \frac{q+1}{2} \equiv \left\lfloor\frac{x}{2}\right\rfloor + \frac{q+1}{2} mod\ q\ (6)$$

Vì nếu rút gọn tiếp tục ½ $mod\ q$ không phải là số tiện lợi để xử lý. Với q là số nguyên tố, $\frac{q+1}{2}$ với $q = 3329$ của Kyber cho kết quả là hằng số 1665. Phương trình sau trình cho kết quả:

$$\frac{x}{2} mod\ q = \begin{cases} \left\lfloor\frac{x}{2}\right\rfloor\ if\ x\ is\ even \\ \left\lfloor\frac{x}{2}\right\rfloor + \frac{q+1}{2} mod\ q\ if\ x\ is\ odd \end{cases}$$

Bộ rút gọn modulo chia nửa chỉ bao gồm một bộ mux, một bộ cộng và một bộ dịch phải 1 vì $\left\lfloor\frac{x}{2}\right\rfloor = (x \gg 1)$. Bộ này được pipeline 2 tầng để tăng tốc độ của mạch.

4.1.3 Thiết kế butterfly unit xử lý CT/GS



Hình 13 Phần cứng butterfly unit xử lý CT/GS

Bảng 4 Bảng chân phần cứng butterfly unit xử lý CT/GS

Tên chân	**Loại ngõ**	**Độ dài (bit)**	**Công dụng**
clk	Vào	1	Clock hệ thống
rst	Vào	1	Reset
u	Vào	12	Ngõ vào dữ liệu u
t	Vào	12	Ngõ vào dữ liệu t
w	Vào	12	Ngõ vào dữ liệu Twiddle Factor
sel	Vào	2	Chọn chế độ sử dụng BU: - 01 NTT - 00 INTT - 10,11 Bypass (Bỏ qua và tiếp nối dữ liệu sang ngõ ra)
s0	Ra	12	Ngõ ra dữ liệu kết quả 1 s0
s1	Ra	12	Ngõ ra dữ liệu kết quả 2 s1

Ngoài hai thành phần là bộ rút gọn modulo nêu trên, các thành phần còn lại của bộ BU gồm hai bộ cộng trừ modulo. Bộ cộng trừ modulo là những bộ cộng trừ đơn giản sắp xếp bù *q* cho kết quả cộng trừ 12-bit. Mỗi bộ được pipeline 3 tầng để tăng tốc độ của mạch. Bộ nhân 12-bit sang 24-bit sử dụng 1 đơn vị DSP của FPGA. Các thành phần nhỏ gồm mux và các thanh ghi.

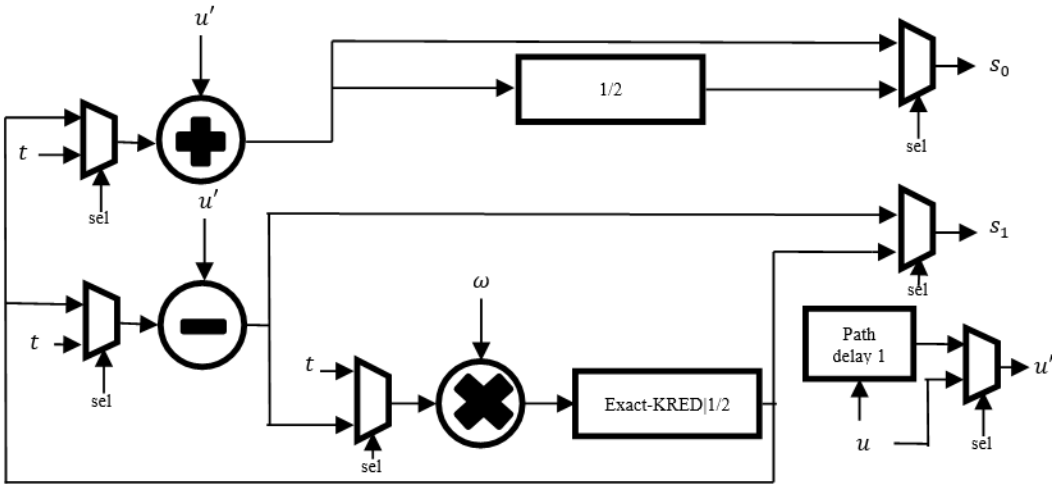
Từ hình 11, chúng tôi thiết kế bộ bướm phần cứng phù hợp cho cả giải thuật CT và GS. Tài nguyên của bộ BU này bao gồm bộ cộng số học modulo (một cộng, một trừ), một bộ nhân DSP và một bộ rút gọn modulo Exact-KRED, và hai bộ rút gọn modulo chia nửa. Bộ rút gọn

modulo chia nửa cho mức tiêu thụ tài nguyên không đáng kể. Ngõ ra của bộ nhân nối trực tiếp vào ngõ vào của bộ Exact-KRED.

Hình 14 cung cấp cấu trúc chi tiết của thiết bị. Độ trễ của bộ nhân DSP và đơn vị Exact-KRED có 7 chu kỳ trễ. Độ trễ đường dẫn chặn đường dẫn dữ liệu và được điều chỉnh theo tổng độ trễ. Tổng độ trễ cho mỗi quy trình thực hiện giải thuật CT / GS trên thiết bị cánh bướm là 13 chu kỳ, với đầu vào và đầu ra đã được pipeline. Tín hiệu Sel được sử dụng để chọn chế độ hoạt động. Phép cộng và phép trừ mô-đun với hai bộ cộng mô-đun không ảnh hưởng đáng kể đến hiệu suất tốc độ do số nguyên tố q 12-bit nhỏ.

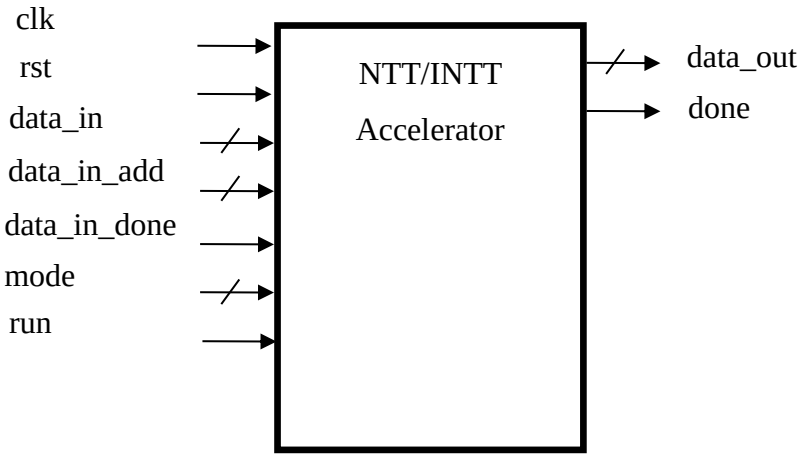
Đối với giải thuật CT, tín hiệu theo đường dẫn trên của mux được thực hiện, trong đó bộ cộng mong đợi kết quả nhân mô-đun sẽ đến sau 11 clock kể từ thời điểm nhập dữ liệu. Độ trễ 2 clock được thêm vào giải thuật CT để phù hợp với độ trễ thực hiện giải thuật GS.

Đối với giải thuật GS, đường dẫn bên dưới của mux được thực hiện. Bộ rút gọn modulo chia nửa trong giải thuật GS tốn 2 clock. Tổng độ trễ khi bộ BU được sử dụng trong hoạt động GS là 13 clock.



Hình 14 Bộ Butterfly Unit kết hợp CT/GS cho thuật toán độ phức tạp thấp

4.1.4 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber



Hình 15 Phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber

Bảng 5 Bảng chân phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber

Tên chân	**Loại ngõ**	**Độ dài (bit)**	**Công dụng**
clk	Vào	1	Clock hệ thống
rst	Vào	1	Reset
data_in	Vào	48	Ngõ vào dữ liệu bus 48
data_in_add	Vào	5	Địa chỉ ghi dữ liệu
data_in_done	Vào	1	Chân báo hiệu đã truyền dữ liệu hoàn tất, tích cực cao.
data_out	Ra	48	Ngõ ra dữ liệu cho chế độ xuất dữ liệu
mode	Vào	2	Chế độ sử dụng: - 01 NTT - 10 INTT - 00 Nhập dữ liệu - 11 Xuất dữ liệu
run	Vào	1	Chân báo hiệu thi hành quá trình xử lý, tích cực cao.
done	Ra	1	Chân báo hiệu hoàn thành quá trình xử lý, tích cực cao.

Hình 16 cho thấy cấu trúc chung của bộ gia tốc NTT / INTT cho Kyber. Đối với kiến trúc tổng thể, bộ gia tốc NTT/INTT phần cứng của sử dụng hai loại bộ nhớ. Một RAM để lưu trữ các phần tử hệ số của phương trình gốc cần thực hiện NTT / INTT. Ba ROM dùng để lưu. RAM sử dụng Intel Cyclone V M10K ở chế độ simple dual-port, cho phép truy cập đọc ghi đồng thời vào hai địa chỉ bộ nhớ khác nhau. Bộ gia tốc sử dụng thiết kế 4 BU sắp xếp ở cấu

hình 2 x 2, cấu hình này có đặc điểm tính toán với $log(n)$ chẵn rất tốt. Với $log(n)$ lẻ, 2 BU ở tầng đầu được bypass và giá trị hệ số đi thẳng đến 2 BU (BU 3 và BU4) sau.

Dữ liệu được đọc từ RAM dưới dạng một data bus 48-bit, gồm 4 phần tử hệ số cho hai bộ BU song song. Dữ liệu từ data bus bắt chéo với u_{00} là 12 bit đầu, u_{01} là 12 bit tiếp theo, t_{00} là 12 bit kế tiếp và t_{01}, là 12 bit cuối cùng.

Dữ liệu tính toán trên mỗi ngõ ra của 2 BU hàng sau được đặt trên một thanh ghi dịch nối tiếp 12-bit sang song song 48-bit (SIPO). Sau 4 chu kỳ đầu tiên có kết quả là một chuỗi 48-bit ở mỗi đầu u_{20}, t_{20}, u_{21}, t_{21}, kết quả sẽ được ghi lần lượt trở lại vào RAM từ thanh ghi đầu tiên. Mỗi chu kỳ ghi hết 4 clock, sau đó 4 kết quả mới đã vào vị trí để ghi tiếp theo. Các thanh ghi sipo ở vị trí t_{20}, u_{21}, t_{21} được dời thêm 1, 2, 3 clock theo độ trễ tương ứng. Hình 17 trình bày quá trình dữ liệu dịch vào SIPO Writeback.

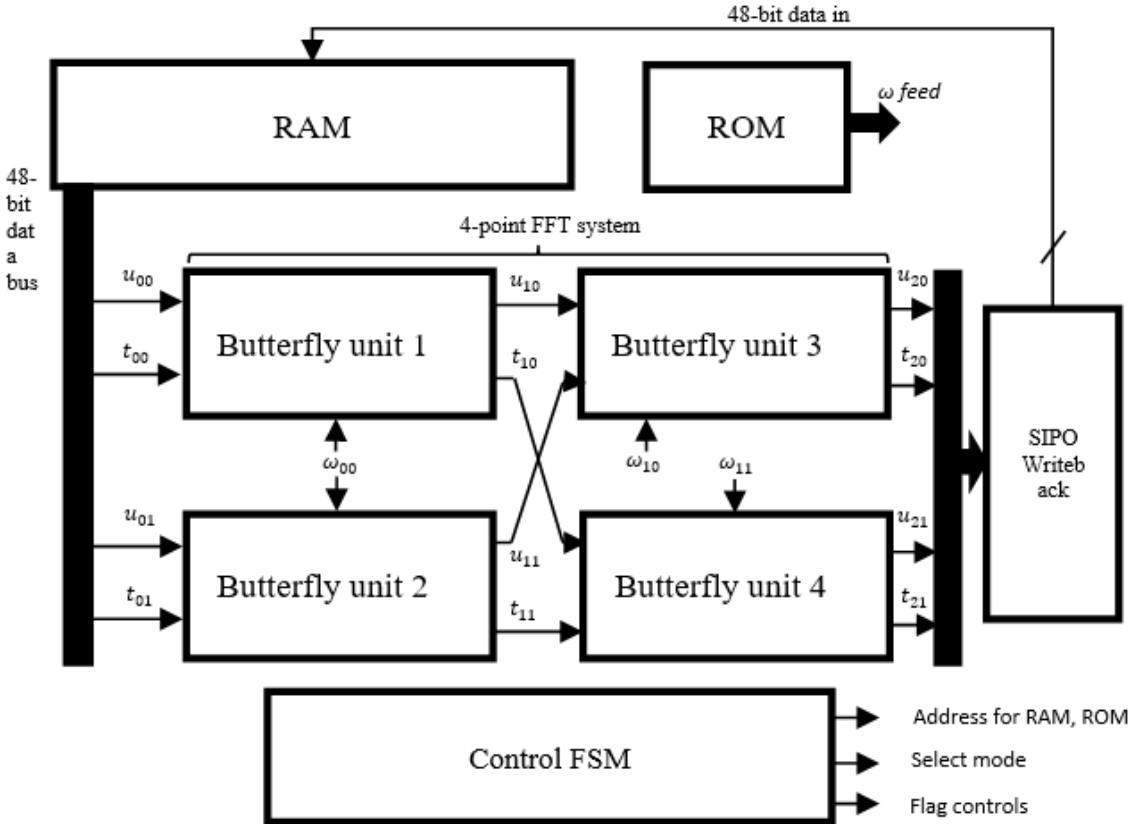
Các ROM cũng cung cấp cho các đơn vị cánh bướm với hệ số twiddle factor tương ứng cho mỗi hoạt động. Giá trị ω được tính toán và lưu trữ trước. Để phù hợp với thứ tự twiddle factor được sử dụng ở cấu hình BU 2x2, twiddle factor được đọc theo thứ tự cycle, từ cycle chỉ đến địa chỉ chứa twiddle factor tương ứng. Thứ tự twiddle factor được sắp xếp trước vào ROM. Một ROM phụ chuyển đổi thứ tự cycle thành thứ tự twiddle factor $\omega_{00}, \omega_{10}, \omega_{11}$. Các thứ tự này đi đến địa chỉ của 3 ROM chứa giá trị twiddle factor tương ứng.

Mỗi phép tính NTT / INTT tốn 30 xung nhịp cho đến khi dữ liệu được ghi trở lại RAM. Độ trễ này là phù hợp để tránh xung đột đọc và ghi dữ liệu vào địa chỉ chưa được truy xuất để tính toán trước đó. Trong đó độ trễ của 2 BU là 26 clock, 2 clock từ truy xuất RAM và 2 clock độ trễ điều khiển.

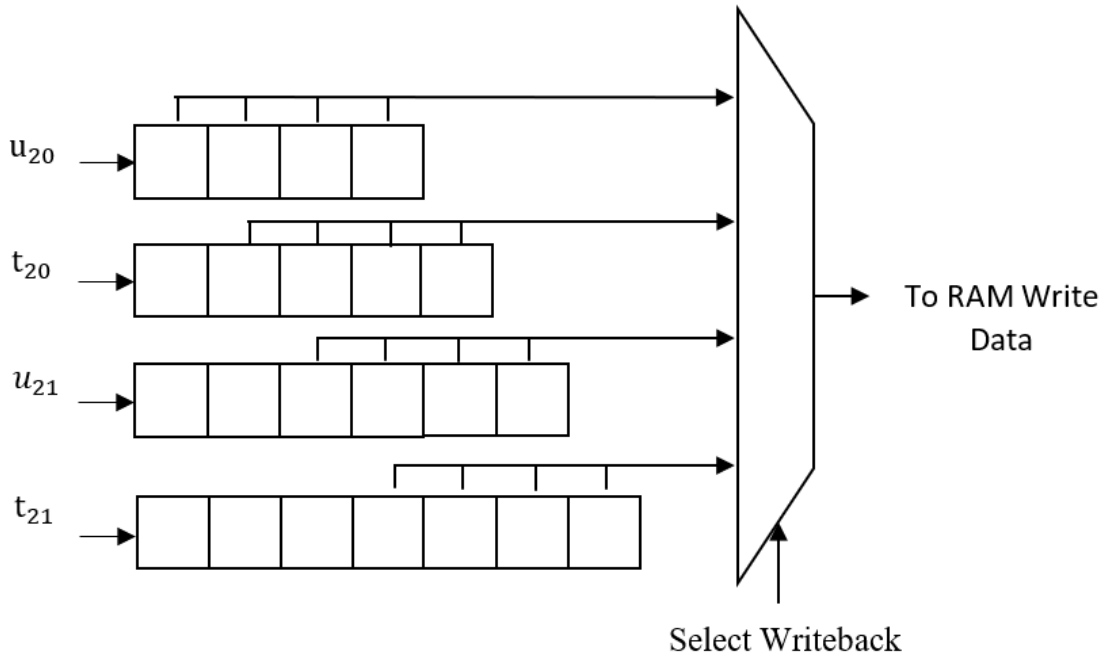
Thuật toán 4 và thuận toán 5 thể hiện giải thuật NTT và INTT được sử dụng cho thiết kế phần cứng xử lý NTT và INTT này. Các bộ BU cũng được sử dụng trong chế độ Cooley - Tuckey (CT_mode) hoặc Gentleman – Sande (GS_mode) tương ứng. Các giá trị mũ của γ_{2n} và γ_{2n}^{-1} cùng với việc nhân với k^{-2} để bù phần dư trong giải thuật Exact-KRED được tính trước và lưu trên hệ thống ROM.

Thuật toán 4 và thuận toán 5 thực hiện đầy đủ với số lớp $log(n)$ NTT/INTT chẵn. Với trường hợp NTT/INTT trên số lớp lẻ ($n = 128$), phần 2 BU đầu tiên của cấu hình 2 x 2 BU được bỏ qua, dữ liệu đi thẳng đến 2 BU sau.

Truy xuất RAM và thứ tự truy xuất ROM của thuật toán 4 được thể hiện ở hình 18 và hình 19. Truy xuất RAM và thứ tự truy xuất ROM của thuật toán 5 được thể hiện ở hình 20 và hình 21. Với hình 18 và hình 19, giá trị trong ô của hệ số phương trình chỉ thứ tự ban đầu của giá trị hệ số đó tại phương trình *a* ban đầu.



Hình 16 Sơ đồ khối bộ xử lý NTT/INTT



Hình 17 Bộ SIPO Writeback vào nối tiếp ra song song với các ô nhớ 12-bit

Cycle #	Seq	Address	A4	A3	A2	A1
1	0	0	96	64	32	0
2	8	1	97	65	33	1
3	16	2	98	66	34	2
4	24	3	99	67	35	3
5	1	4	100	68	36	4
6	9	5	101	69	37	5
7	17	6	102	70	38	6
8	25	7	103	71	39	7
9	2	8	104	72	40	8
10	10	9	105	73	41	9
11	18	10	106	74	42	10
12	26	11	107	75	43	11
13	3	12	108	76	44	12
14	11	13	109	77	45	13
15	19	14	110	78	46	14
16	27	15	111	79	47	15
17	4	16	112	80	48	16
18	12	17	113	81	49	17
19	20	18	114	82	50	18
20	28	19	115	83	51	19
21	5	20	116	84	52	20
22	13	21	117	85	53	21
23	21	22	118	86	54	22
24	29	23	119	87	55	23
25	6	24	120	88	56	24
26	14	25	121	89	57	25
27	22	26	122	90	58	26
28	30	27	123	91	59	27
29	7	28	124	92	60	28
30	15	29	125	93	61	29
31	23	30	126	94	62	30
32	31	31	127	95	63	31
33	0	0	24	16	8	0
34	2	1	25	17	9	1
35	4	2	26	18	10	2
36	6	3	27	19	11	3
37	1	4	28	20	12	4
38	3	5	29	21	13	5
39	5	6	30	22	14	6
40	7	7	31	23	15	7
..	...	...	...	...	...	...

Hình 18 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho NTT (n=128, 40 cycles đầu tiên)

Hình 18 thể hiện sắp xếp bộ nhớ RAM với $n = 128$, và thứ tự địa chỉ truy xuất cho đến cycle 40. Cột đầu tiên là thứ tự cycle khi bắt đầu thực hiện tính toán NTT, Cột Seq là địa chỉ cần truy xuất tại Cycle đó. Bên phải từ Address đến A1, A2, A3, A4 là vị trí các hệ số của phương trình trong bộ nhớ RAM.

Cycle	w00	w10	w11
1	64	32	96
2	64	32	96
3	64	32	96
4	64	32	96
5	64	32	96
6	64	32	96
7	64	32	96
8	64	32	96
9	64	32	96
10	64	32	96
11	64	32	96
12	64	32	96
13	64	32	96
14	64	32	96
15	64	32	96
16	64	32	96
17	64	32	96
18	64	32	96
19	64	32	96
20	64	32	96
21	64	32	96
22	64	32	96
23	64	32	96
24	64	32	96
25	64	32	96
26	64	32	96
27	64	32	96
28	64	32	96
29	64	32	96
30	64	32	96
31	64	32	96
32	64	32	96
33	16	8	72
34	16	8	72
35	16	8	72
36	16	8	72
37	16	8	72
38	16	8	72
39	16	8	72
40	16	8	72

Hình 19 Thứ tự địa chỉ truy xuất tại ROM cho NTT (n=128, 40 cycles đầu tiên)

Hình 19 thể hiện thứ tự địa chỉ truy xuất ROM cho NTT với n = 128 tại 40 cycle đầu tiên. Các giá trị truy xuất theo thứ tự $\omega_{00}, \omega_{10}, \omega_{11}$.

Cycle #	Seq	Address	A4	A3	A2	A1
1	0	0	9	8	1	0
2	1	1	11	10	3	2
3	2	2	13	12	5	4
4	3	3	15	14	7	6
5	4	4	25	24	17	16
6	5	5	27	26	19	18
7	6	6	29	28	21	20
8	7	7	31	30	23	22
9	8	8	41	40	33	32
10	9	9	43	42	35	34
11	10	10	45	44	37	36
12	11	11	47	46	39	38
13	12	12	57	56	49	48
14	13	13	59	58	51	50
15	14	14	61	60	53	52
16	15	15	63	62	55	54
17	16	16	73	72	65	64
18	17	17	75	74	67	66
19	18	18	77	76	69	68
20	19	19	79	78	71	70
21	20	20	89	88	81	80
22	21	21	91	90	83	82
23	22	22	93	92	85	84
24	23	23	95	94	87	86
25	24	24	105	104	97	96
26	25	25	107	106	99	98
27	26	26	109	108	101	100
28	27	27	111	110	103	102
29	28	28	121	120	113	112
30	29	29	123	122	115	114
31	30	30	125	124	117	116
32	31	31	127	126	119	118
33	0	0	6	4	2	0
34	2	1	7	5	3	1
35	4	2	14	12	10	8
36	6	3	15	13	11	9
37	1	4	22	20	18	16
38	3	5	23	21	19	17
39	5	6	30	28	26	24
40	7	7	31	29	27	25
..	...	...	...	...	...	...

Hình 20 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho INTT (n=128, 40 cycles đầu tiên)

Hình 20 thể hiện sắp xếp bộ nhớ RAM với $n = 128$, và thứ tự địa chỉ truy xuất cho đến cycle 40. Cột đầu tiên là thứ tự cycle khi bắt đầu thực hiện tính toán INTT, Cột Seq là địa chỉ cần truy xuất tại Cycle đó. Bên phải từ Address đến A1, A2, A3, A4 là vị trí các hệ số của phương trình trong bộ nhớ RAM.

Cycle	w00	w10	w11
1		9	1
2		11	3
3		13	5
4		15	7
5		25	17
6		27	19
7		29	21
8		31	23
9		41	33
10		43	35
11		45	37
12		47	39
13		57	49
14		59	51
15		61	53
16		63	55
17		73	65
18		75	67
19		77	69
20		79	71
21		89	81
22		91	83
23		93	85
24		95	87
25		105	97
26		107	99
27		109	101
28		111	103
29		121	113
30		123	115
31		125	117
32		127	119
33	4	2	66
34	4	2	66
35	68	34	98
36	68	34	98
37	36	18	82
38	36	18	82
39	100	50	114
40	100	50	114

Hình 21 Thứ tự địa chỉ truy xuất tại ROM cho INTT (n=128, 40 cycles đầu tiên)

Hình 21 thể hiện thứ tự địa chỉ truy xuất ROM cho INTT với n = 128 tại 40 cycle đầu tiên. Các giá trị truy xuất theo thứ tự $\omega_{00}, \omega_{10}, \omega_{11}$.

Thuật toán 4 Proposed NTT operation with Cooley – Tuckey butterfly for Kyber

Input: polynomial $a(a_0, a_1, \dots, a_{n-1})$ as $a(x) \in \mathbb{Z}_q[X]/(X_n + 1)$, $\omega_n \in \mathbb{Z}_q$ is the n-th primitive root of unity, $n = 2^l$ and $\gamma_{2n} = \sqrt{\omega_n}$, $c = n$

Output: $a(x) = NTT(a)$

1: **for** $(i = 1;\ i < l; i = i + 2)$ **do**
2: $m = 2^i$
3: $c = c/4$
4: **for** $(j = 0; j < {}^{m}/_{2} - 1; j++)$ **do**
5: **for** $(k = 4i \cdot c; k < 4i \cdot c + c; k++)$ **do**
6: $u_{00} \leftarrow a[k], t_{00} \leftarrow a[k + c]$
7: $u_{01} \leftarrow a[k + 2c], t_{01} \leftarrow a[k + 3c]$
8: $\omega_{00} \leftarrow k^{-2} \cdot \gamma[(2j + 1)N/m]$ (*pre-computed*)
9: $(u_{10}, t_{10}) \leftarrow BU1(u_{00}, t_{00}, \omega_{00}, CT_{mode})$
10: $(u_{11}, t_{11}) \leftarrow BU2(u_{01}, t_{01}, \omega_{00}, CT_{mode})$
11: $\omega_{10} \leftarrow k^{-2} \cdot \gamma[(2j + 1)N/2m]$ (*pre-computed*)
12: $\omega_{11} \leftarrow k^{-2} \cdot \gamma[2j + 3)N/2m]$ (*pre-computed*)
13: $(u_{20}, t_{20}) \leftarrow BU3(u_{10}, u_{11}, \omega_{10}, CT_{mode})$
14: $(u_{21}, t_{21}) \leftarrow BU4(t_{10}, t_{11}, \omega_{11}, CT_{mode})$
15: $a[k] \leftarrow u_{20}, a[k + c] \leftarrow t_{20}$
16: $a[k + 2c] \leftarrow u_{21}, a[k + 3c] \leftarrow t_{21}$
17: **end for**
18: **end for**
19: **end for**
20: **return** a(x)

Thuật toán 5 Proposed INTT operation with Gentleman – Sande butterfly for Kyber

Input: polynomial $a(a_0, a_1, \dots, a_{n-1})$ as $a(x) \in \mathbb{Z}_q[X]/(X_n + 1)$, $\omega_n \in \mathbb{Z}_q$ is the n-th primitive root of unity, $n = 2^l$ and $\gamma_{2n} = \sqrt{\omega_n}$

Output: $a(x) = INTT(a)$

1: **for** $(i = 1;\ i < l; i = i - 2)$ **do**
2: $m = 2^{l-i}$
3: c = 1
4: **for** $(i = 1;\ i < l; i = i - 2)$ **do**
5: **for** $(k = 4i \cdot c; k < 4i \cdot c + c; k++)$ **do**
6: $u_{00} \leftarrow a[k], t_{00} \leftarrow a[k + c]$
7: $u_{01} \leftarrow a[k + 2c], t_{01} \leftarrow a[k + 3c]$
8: $\omega_{00} \leftarrow k^{-2} \cdot \gamma^{-1}[(2j + 1)N/m]$ (*pre-computed*)
9: $(u_{10}, t_{10}) \leftarrow BU1(u_{00}, t_{00}, \omega_{00}, GS_{mode})$
10: $(u_{11}, t_{11}) \leftarrow BU2(u_{01}, t_{01}, \omega_{00}, GS_{mode})$
11: $\omega_{10} \leftarrow k^{-2} \cdot \gamma^{-1}[(2j + 1)N/2m]$ (*pre-computed*)
12: $\omega_{11} \leftarrow k^{-2} \cdot \gamma^{-1}[2j + 3)N/2m]$ (*pre-computed*)
13: $(u_{20}, t_{20}) \leftarrow BU3(u_{10}, u_{11}, \omega_{10}, GS_{mode})$
14: $(u_{21}, t_{21}) \leftarrow BU4(t_{10}, t_{11}, \omega_{11}, GS_{mode})$
15: $a[k] \leftarrow u_{20}, a[k + c] \leftarrow t_{20}$
16: $a[k + 2c] \leftarrow u_{21}, a[k + 3c] \leftarrow t_{21}$
17: **end for**

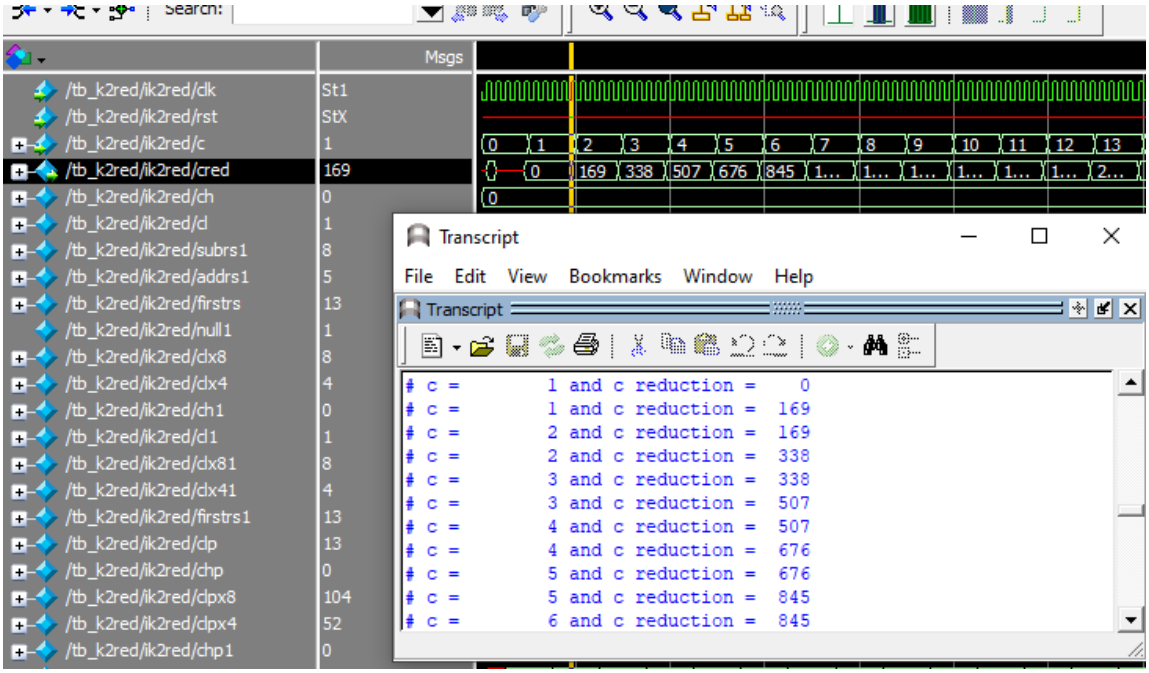
```
18:     end for
19:     c = c >> 2
19: end for
20: return a(x)
```

4.2 Kết quả tổng hợp và mô phỏng

Kết quả mô phỏng mạch và kiểm tra với Testbench được thực hiện trên phần mềm ModelSim cho từng thiết kế nhỏ đến thiết kế tổng. Kết quả tổng hợp mạch sử dụng hai phần mềm là Quartus Prime Standard Edition 21.1 (Hỗ trợ Cyclone V). Kết quả nghiên cứu mô phỏng và tổng hợp sử dụng chung cấu trúc cho n = 128 và n = 256. Qua đó có thể kiểm tra kết quả tính toán với số lớp NTT/INTT lẻ, và đồng thời có thể so sánh với các nghiên cứu tương tự.

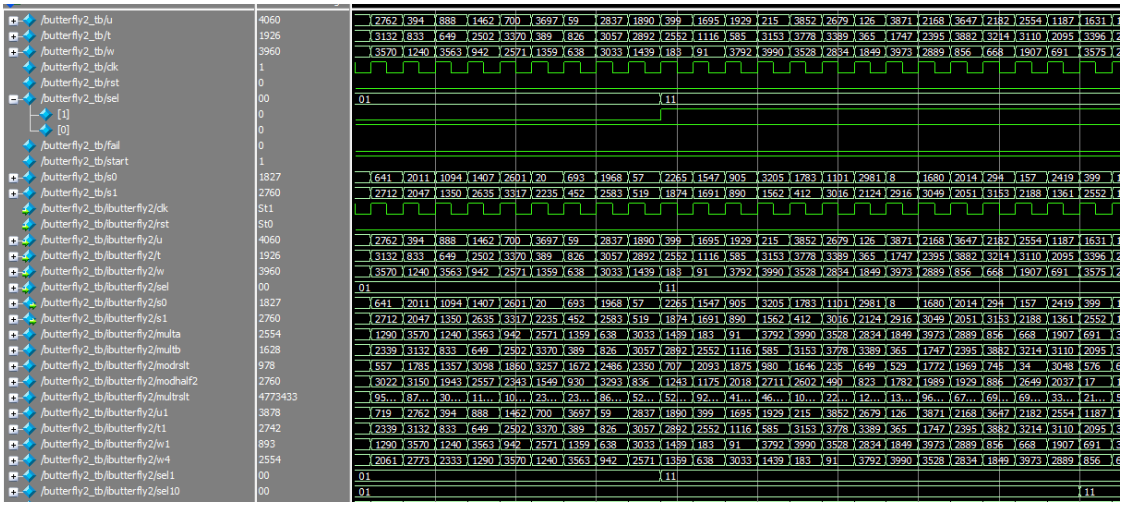
4.2.1 Kết quả mô phỏng ModelSim

Nghiên cứu thiết kế Testbench cho các khối nhỏ để kiểm thử kết quả. Hình 21 trình bày mô phỏng kết quả của Exact-KRED.



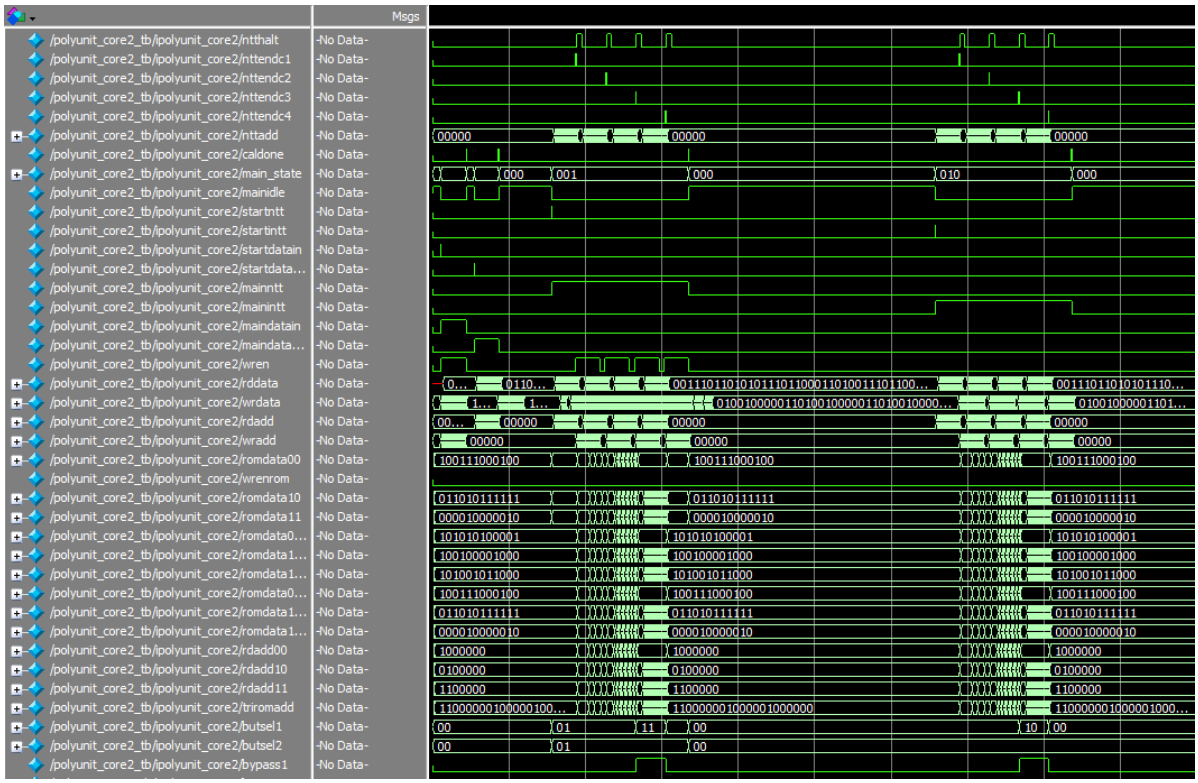
Hình 22 Mô phỏng trên dạng sóng kết quả của thiết kế Exact-KRED

Hình 23 trình bày mô phỏng kết quả trên dạng sóng kết quả của thiết kế BU với các chế độ CT, GS và bypass.



Hình 23 Mô phỏng trên dạng sóng kết quả của thiết kế BU.

Với khối thiết kế tổng phần cứng xử lý NTT và INTT cho mã hóa lượng tử CRYSTALS-Kyber thực hiện đầy đủ các chức năng nạp dữ liệu, xuất dữ liệu, NTT và INTT. Hình 24 trình bày mô phỏng dạng sóng và kết quả đi qua bốn chu trình của khối thiết kế tổng.



Hình 24 Mô phỏng dạng sóng kết quả của phần cứng xử lý NTT và INTT

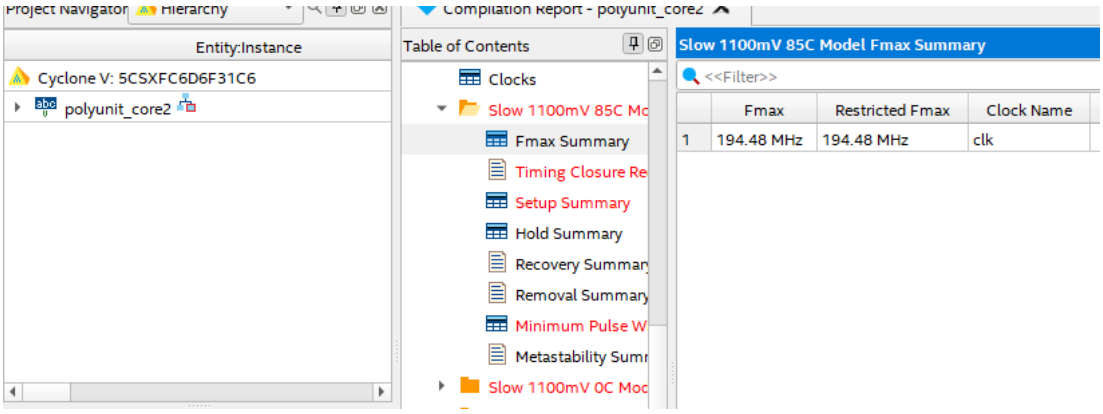
4.2.2 Kết quả tổng hợp Quartus

Kết quả tổng hợp Quartus được dùng để kiểm tra tốc độ và tài nguyên tiêu thụ của thiết kế để so sánh và kiểm tra.

Flow Status	Successful - Thu Jan 6 12:28:51 2022
Quartus Prime Version	21.1.0 Build 842 10/21/2021 SJ Standard Edition
Revision Name	polyunit_core2
Top-level Entity Name	polyunit_core2
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	1,322 / 41,910 (3 %)
Total registers	2929
Total pins	108 / 499 (22 %)
Total virtual pins	0
Total block memory bits	2,268 / 5,662,720 (< 1 %)
Total DSP Blocks	4 / 112 (4 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

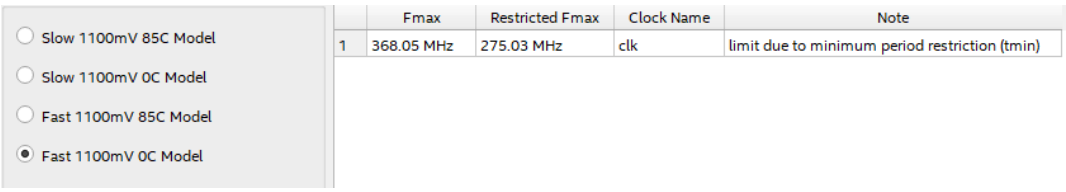
Hình 25 Kết quả tổng hợp tài nguyên trên Quartus

Hình 25 cho thấy kết quả tổng hợp tài nguyên tiêu thụ của thiết kế phần cứng xử lý NTT và INTT cho mã hóa lượng tử CRYSTALS-Kyber. Thiết kế sử dụng 1322 ALMs, 2929 thanh ghi, 4 khối DSP, 2268 block memory bit đến từ 1 khối BRAM M10K.



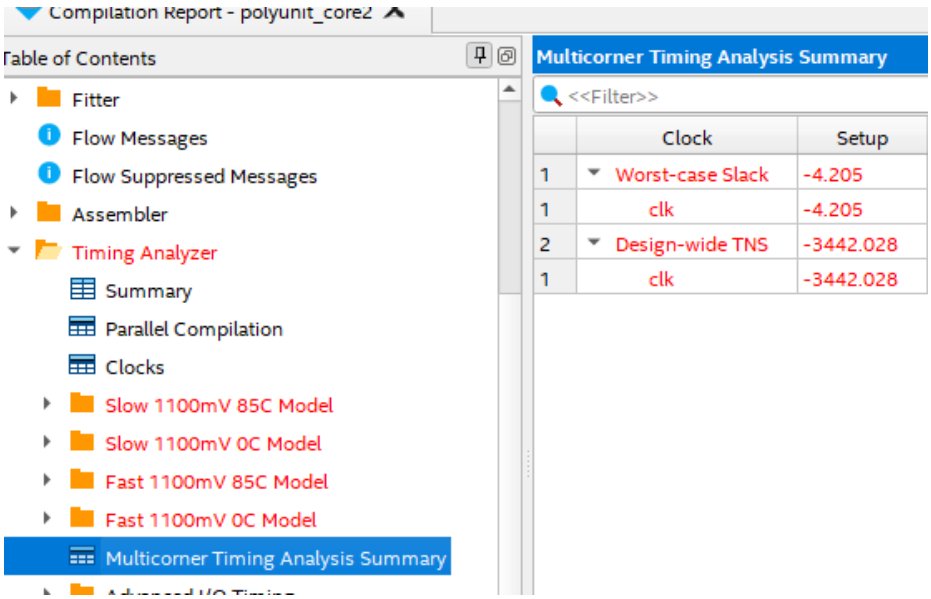
Hình 26 Kết quả tốc độ mạch trường hợp Slow 1100 mV 85C

Hình 26 thể hiện kết quả tốc độ mạch ở trường hợp góc mô phỏng môi trường khó nhất đạt 194.48 MHz.



Hình 27 Kết quả tốc độ mạch trường hợp Fast 1100 mV 0C

Hình 27 thể hiện kết quả tốc độ mạch ở trường hợp góc mô phỏng môi trường tốt nhất đạt 368.05 MHz giới hạn ở 275 MHz.



Hình 28 Kết quả tốc độ mạch trung bình

Hình 28 thể hiện kết luận về tốc độ mạch trung bình đa môi trường ở mức 4.2 $ns \sim 237\ MHz\ fmax$

4.3 Đánh giá, bàn luận và so sánh kết quả

Bảng 6 Thiết kế đề xuất so với các nghiên cứu NTT tương tự trước đây (n = 256)

Thiết kế	**Cấu hình BU**	**Area**				**Speed [MHz]**	**NTT/INTT Cycles**	**NTT/INTT [ns]**	**NTT Speed Ratio**	**Area x Speed Ratio**
		LUTs	**FFs**	**DSPs**	**BRAM**					
[31][2]	2	1737	1167	2	3	161	512/576	3174/3571	2.636	3.41
Low-Comp [32][2]	2	741	330	2	5	245	644/644[1]	2642/2642	2.2	1.23
[33]	2	2908	170	9	0	79.66	1935/1930	25155/25090	20	43
HS-NTT [15][2]	2x2	801	717	4	2	222	324/324	1458/1458	1.21	0.73
Nghiên cứu này	**2x2**	**1322**	**2929**	**4**	**1**	**237**	**286/286**	**1204/1204**	**1**	**1**

[1]Chuyển đổi từ chia hai kết quả cycles với n = 512

[2]Sử dụng Xilinx Artix-7

[3]Sử dụng Xilinx Virtex-7

[4]Kết quả trên ALMs từ báo cáo tổng hợp của Quartus

Bảng 6 so sánh thiết kế đề xuất từ nghiên cứu này với các nghiên cứu NTT tương tự. Tuy vậy, có một lưu ý về mức độ tài nguyên tiêu thụ. Với các nghiên cứu thực hiện trên FPGA Intel, Quartus chỉ trả về kết quả tổng hợp ALM và ALUTs. Nghiên cứu [30] cho thấy kết quả tổng hợp từ FPGA sử dụng công nghệ ALM cho số lượng ALUTs lớn hơn nhiều so với số lượng LUTs tổng hợp từ công nghệ FPGA từ Xillinx do 2 ALUT từ chung 1 ALM khó có thể sử dụng được cùng lúc mà chỉ sử dụng được 1 ALUT trên 1 ALM. Ngoài ra ALM còn chứa nhiều thanh ghi, và các thành phần khác khiến kết quả tài nguyên tổng hợp từ Quartus để so sánh với kết quả tài nguyên từ FPGA Xillinx phổ biến hơn tương đối khó so sánh. Ở bảng 6, kết quả LUTs sử dụng số ALM từ báo cáo tổng hợp. NTT Speed Ratio so sánh thời gian tính toán trên *ns* so với các nghiên cứu khác. Area x Speed ratio so sánh thời gian tính toán cùng với LUT tiêu thụ.

Nghiên cứu [31] sử dụng cấu hình 3 khối RAM và hệ thống nhân Karatsuba cũng như phiên bản cải tiến của rút gọn modulo Barret Reduction. Số lượng NTT Cycles [31] cần để tính toán là phù hợp với cấu hình 2 BU. Việc tốc độ của nghiên cứu [31] thấp hơn trong khi tốn nhiều tài nguyên hơn đến từ việc BU từ [31] tích hợp rất nhiều tính năng. So với [31], thiết kế

của nghiên cứu này nhanh hơn gấp 2.6 lần với tỉ lệ tài nguyên tiêu thụ trên tốc độ tốt ở mức 3.41.

Nghiên cứu [32] sử dụng cho giải thuật mã hóa NewHope cũ, là nghiên cứu tiên phong cho thay đổi NTT/INTT, bỏ các bước xử lý hậu kỳ của INTT. Nghiên cứu [32] cho kết quả tốc độ với cấu hình 2 BU cao. So với [32], nghiên cứu này đạt tốc độ xử lý NTT/INTT nhanh hơn và với tỉ lệ tài nguyên tiêu thụ trên tốc độ ở mức tối ưu hơn 1.23.

Nghiên cứu [33] áp dụng cấu trúc thêm cho RISC và xây dựng trên ASIC. Nghiên cứu có tốc độ chưa cao nhưng đạt mức tích hợp tốt do không tối ưu pipeline mạnh cho ALU, dẫn đến đường tới hạn của thiết kế chưa đạt được tốc độ cao. Các thành phần module từ nghiên cứu của luận văn này được pipeline và tối ưu về tốc độ, tránh số lượng mức logic cao.

Nghiên cứu [15] sử dụng K^2-RED và cấu hình 2x2 BU cho kết quả tốc độ và tài nguyên tốt. Số lượng thanh ghi cũng tiết kiệm hơn rất nhiều. Tuy K^2-RED có một số trường hợp bị tràn, đây là việc có thể khắc phục được với Exact-KRED.

Sau khi so sánh với các nghiên cứu tương tự gần đây, nghiên cứu này cho kết quả ở mức khá. Nghiên cứu đã sử dụng phương pháp rút gọn modulo và giải thuật NTT/INTT tối ưu nhất hiện tại. Tuy vậy, nghiên cứu có mức tiêu thụ thanh ghi lớn đến từ việc tính toán trước vị trí của các hệ số Twiddle Factor, đây là vấn đề có thể khắc phục được bằng cách thay đổi quy trình áp dụng vòng lặp máy lên phần cứng, mang nhiều yếu tố tổ hợp hơn thanh ghi. Ngoài ra, các hướng áp dụng trên FPGA Xillinx có thể được nghiên cứu thêm trong các đề tài tiếp theo để có thể dễ so sánh hơn với các đề tài khác. Một số hướng tối ưu khác được trình bày trong phần kết luận.

5. KẾT LUẬN VÀ KIẾN NGHỊ NHỮNG NGHIÊN CỨU TIẾP THEO

5.1 Các hướng tối ưu thiết kế phần cứng xử lý NTT và INTT

Thiết kế còn nhiều hướng tối ưu cho tác vụ xử lý NTT và INTT. Không chỉ với các dòng mã hóa Lattice mà các dòng mã hóa lượng tử sau này cũng mang nhiều kỹ thuật cần sử dụng phép nhân tối ưu cho hệ số của các phương trình. Một số hướng tối ưu tiêu biểu là:

- Tận dụng tính tinh chỉnh thông số linh hoạt của ngôn ngữ mô tả phần cứng Verilog. Đưa ra giải pháp chung khi thay đổi thiết kế gốc dễ dàng hơn. Trong đó có thể bao gồm tinh chỉnh thông số để thay đổi *n*, thay đổi số BU, số RAM.
- Phát triển hệ thống câu lệnh điều khiển (instruction set). Hướng thay đổi này phù hợp để tích hợp phần xử lý vào các cấu trúc vi xử lý, dưới dạng một vi xử lý mã hóa phụ.
- Thử các phương pháp tấn công phân tích dữ liệu phụ. Bản thân thiết kế mang giải thuật không thay đổi thời gian tính toán, chống phân tích thời gian xử lý. Tuy vậy, các vấn đề về phân tích năng lượng xử lý, cần được nghiên cứu thêm.

5.2 Thiết kế phần cứng xử lý CRYSTALS-Kyber

Thiết kế phần cứng xử lý toàn bộ mã hóa lượng tử CRYSTALS-Kyber là một trong những mục đích chính để ứng dụng thiết kế phần cứng xử lý NTT/INTT. Kyber là một ứng cử viên có khả năng cao được chuẩn hóa thành mã hóa bất đối xứng tiêu chuẩn trong tương lai. So với các dòng mã hóa bất đối xứng RSA, ECDSA, dòng mã hóa lattice-based của Kyber tương đối mới. Trong trường hợp của Kyber, nhóm nghiên cứu cũng đang đưa ra nhiều cập nhật thông số để tìm hiểu hiệu suất xử lý và bảo mật của dạng mã hóa lượng tử này. Một trong những điều lưu ý trong ứng dụng thiết kế phần cứng xử lý NTT/INTT là:

- Phiên bản NTT/INTT của Kyber sử dụng thông số n = 256 không thực sự phân tích ra phương trình bậc 0.
- Thiết kế của Kyber hiện tại có hai phiên bản sử dụng Keccak và 90s (SHA-2). Keccak được đánh giá là hàm băm chuẩn hóa và có thể tối ưu cho phần cứng.

5.3 Kết luận

Mã hóa sau lượng tử hay mã hóa lượng tử là một hướng phát triển quan trọng của khoa học điện tử. Vai trò của hướng nghiên cứu này rất quan trọng cho sự phát triển vượt bậc của máy tính điện tử, và sau này là máy tính lượng tử. Các chuẩn mã hóa không ngừng được cải tiến để phục vụ nhiều mục đích bảo mật và truyền thông. Trong đó, mã hóa bằng phần cứng là một nhánh nghiên cứu thiết yếu để tránh các cuộc tấn công phân tích tài nguyên xử lý, gia tăng mức độ bảo mật và hiệu quả của các giải thuật mã hóa.

Học viên đề xuất thiết kế phần cứng của bộ xử lý NTT và INTT cho mã hóa lượng tử CRYSTALS-Kyber. Thiết kế bao gồm phần cứng rút gọn modulo được cải tiến, thiết kế BU

CT/GS tiết kiệm tài nguyên có độ phức tạp thấp và kiến trúc bộ xử lý NTT/INTT. Nghiên cứu này cũng cung cấp thêm thông tin trong quá trình tìm hiểu bản chất của các dạng mã hóa như Kyber và các thông số của dạng mã hóa này, góp phần vào công cuộc chuẩn hóa mã hóa lượng tử.

Trong các nghiên cứu trong tương lai, học viên sẽ tối ưu hóa hơn nữa trình tự truy cập bộ nhớ cần thiết để thực hiện hoạt động NTT và INTT. Giảm thiểu trường hợp phải sắp xếp lại bộ nhớ để thực hiện tính toán. Các hướng nghiên cứu chuyên sâu hơn cũng đã được phân tích ở các phần trên.

6. DANH MỤC CÔNG TRÌNH CÔNG BỐ CỦA TÁC GIẢ

1. Nguyen Trinh, Anh Le Thi Kim, Hung Nguyen, Linh Tran, *"Algorithmic TCAM on FPGA with data collision approach"*, Indonesian Journal of Electrical Engineering and Computer Science, Vol. 22, No. 1, 89-96, 2021
2. Nguyễn Tuấn Hùng, Vương Đình Hưng, GVHD: TS. Trần Hoàng Linh, *"Bộ Mã Hóa Đường Cong Nghệ 256 Bit Sử Dụng FPGA"*, Luận Văn Tốt Nghiệp Đại Học, Tháng 06 Năm 2019, DTU-182-17
3. Hung Nguyen, Linh Tran, "Design of polynomial NTT and INTT accelerator for Post-Quantum Cryptography CRYSTALS-Kyber"

7. DANH MỤC TÀI LIỆU THAM KHẢO

[1] NIST, "Post-Quantum Cryptography Standardization", csrc.nist.gov, https://csrc.nist.gov/Projects/post-quantum-cryptography.

[2] Bos, Joppe, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé, "CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM." In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pp. 353-367. IEEE, 2018, doi: 10.1109/EuroSP.2018.00032.

[3] Intel, "Intel® Data Protection Technology with AES-NI and Secure Key", https://www.intel.com/content/www/us/en/architecture-and-technology/advanced-encryption-standard-aes/data-protection-aes-general-technology.html

[4] Eberle, Hans, Nils Gura, Sheueling Chang Shantz, Vipul Gupta, Leonard Rarick, and Shreyas Sundaram. "A public-key cryptographic processor for RSA and ECC." In Proceedings. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2004., pp. 98-110. IEEE, 2004, doi: 10.1109/ASAP.2004.1342462

[5] Andrzejczak, Michal, Farnoud Farahmand, and Kris Gaj, "Full Hardware Implementation of the Post-Quantum Public-Key Cryptography Scheme Round5." In 2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1-2. IEEE, 2019, doi: 10.1109/ReConFig48160.2019.8994765.

[6] Huang, Yiming, Miaoqing Huang, Zhongkui Lei, and Jiaxuan Wu, "A pure hardware implementation of crystals-kyber PQC algorithm through resource reuse." IEICE Electronics Express (2020) ID: 17-20200234, doi: 10.1587/elex.17.20200234.

[7] Botros, Leon, Matthias J. Kannwischer, and Peter Schwabe, "Memory-efficient high-speed implementation of Kyber on Cortex-M4." In International Conference on Cryptology in Africa, pp. 209-228. Springer, Cham, 2019, doi: 10.1007/978-3-030-23696-0_11.

[8] Jati, Arpan, Naina Gupta, Anupam Chattopadhyay, and Somitra Kumar Sanadhya, "A Configurable Crystals-Kyber Hardware Implementation with Side-Channel Protection." Cryptology ePrint Archive (2021).

[9] Zhao, Yixuan, Zhiteng Chao, Jing Ye, Wen Wang, Yuan Cao, Shuai Chen, Xiaowei Li, and Huawei Li, "Optimization Space Exploration of Hardware Design for CRYSTALS-KYBER." In 2020 IEEE 29th Asian Test Symposium (ATS), pp. 1-6. IEEE, 2020. doi: 10.1109/ATS49688.2020.9301498.

[10] Albrecht, Martin R., Christian Hanser, Andrea Hoeller, Thomas Pöppelmann, Fernando Virdia, and Andreas Wallner, "Implementing RLWE-based schemes using an RSA co-processor." Cryptology ePrint Archive (2018).

[11] Sanal, Pakize, Emrah Karagoz, Hwajeong Seo, Reza Azarderakhsh, and Mehran Mozaffari-Kermani. "Kyber on ARM64: Compact Implementations of Kyber on 64-bit ARM Cortex-A Processors." Cryptology ePrint Archive (2021).

[12] Seo, Hwa-jeong, Hyeok-dong Kwon, Kyoung-bae Jang, and Hyunjun Kim, "Optimized implementation of scalable multi-precision multiplication method on RISC-V processor for high-speed computation of post-quantum cryptography." Journal of the Korea Institute of Information Security & Cryptology 31, no. 3 (2021): 473-480, doi: 10.13089/JKIISC.2021.31.3.473.

[13] Xing, Yufei, and Shuguo Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 328-356, doi: 10.46586/tches.v2021.i2.328-356.

[14] Guo, Wenbo, Shuguo Li, and Liang Kong, "An Efficient Implementation of KYBER." IEEE Transactions on Circuits and Systems II: Express Briefs (2021), doi: 10.1109/TCSII.2021.3103184.

[15] Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani, "High-Speed NTT-based Polynomial Multiplication Accelerator for CRYSTALS-Kyber Post-Quantum Cryptography." Cryptol. ePrint Arch., Tech. Rep 563 (2021): 2021.

[16] Yarman, Ferhat, Ahmet Can Mert, Erdinç Öztürk, and Erkay Savaş, "A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme." In 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1020-1025. IEEE, 2021, doi: 10.23919/DATE51398.2021.9474139

[17] Chen, Zhaohui, Yuan Ma, Tianyu Chen, Jingqiang Lin, and Jiwu Jing, "Towards efficient Kyber on FPGAs: A processor for vector of polynomials." In 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 247-252. IEEE, 2020, doi: 10.1109/ASP-DAC47756.2020.9045459.

[18] Zhang, Cong, Dongsheng Liu, Xingjie Liu, Xuecheng Zou, Guangda Niu, Bo Liu, and Quming Jiang, "Towards Efficient Hardware Implementation of NTT for Kyber on FPGAs." In 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5. IEEE, 2021, doi: 10.1109/ISCAS51556.2021.9401170.

[19] Pöppelmann, Thomas, and Tim Güneysu, "Towards efficient arithmetic for lattice-based cryptography on reconfigurable hardware." In International conference on cryptology and information security in Latin America, pp. 139-158. Springer, Berlin, Heidelberg, 2012, doi: 10.1007/978-3-642-33481-8_8

[20] Linh, Đàm & Hoang, Trong-Thuc & Tú, Bùi & Vũ, Đinh. (2014). "Hiện thực và so sánh các thiết kế FFT 2048 điểm xây dựng trên nền tảng FPGA."

[21] Trương Thị Như Quỳnh, Võ Thị Phương Thảo, Hoàng Trọng Thức & Lê Đức Hùng. (2017). "Thiết kế FFT 2048-điểm trên FPGA dựa trên thuật toán CORDIC xoay góc thích nghi với độ chính xác dấu chấm động đơn"

[22] Phạm Ngọc Bách, Nguyễn Văn Hiệp. (2018). "Sử dụng biến đổi nhanh fourier (FFT) nghiên cứu cấu trúc bão và sự phát triển xoáy bão trong sơ đồ ban đầu hóa xoáy động lực "

[23] Wikimedia Foundation. (2021, December 9). Public-key cryptography. Wikipedia., from https://en.wikipedia.org/wiki/Public-key_cryptography

[24] Langlois, Adeline, and Damien Stehlé, "Worst-case to average-case reductions for module lattices." Designs, Codes and Cryptography 75, no. 3 (2015): 565-599. Avanzi, Roberto, et al. "CRYSTALS-Kyber algorithm specifications and supporting documentation." *NIST PQC Round* 2.4 (2017), doi: 10.1007/s10623-014-9938-4

[25] Avanzi, Roberto, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation." NIST PQC Round 2, no. 4 (2017).

[26] The Fast Fourier Transform (FFT): Most Ingenious Algorithm Ever?, Reducible, 14 Nov. 2020, https://www.youtube.com/watch?v=h7apO7q16V0. Accessed 2021.

[27] Bisheh-Niasar, Mojtaba, Reza Azarderakhsh, and Mehran Mozaffari-Kermani, "A Monolithic Hardware Implementation of Kyber: Comparing Apples to Apples in PQC Candidates." In International Conference on Cryptology and Information Security in Latin America, pp. 108-126. Springer, Cham, 2021, doi: 10.1007/978-3-030-88238-9_6.

[28] Intel, "Cyclone V Device Datasheet," CV-51002 datasheet, Nov. 2019.

[29] Nguyen Trinh, Anh Le Thi Kim, Hung Nguyen, Linh Tran, "Algorithmic TCAM on FPGA with data collision approach", Indonesian Journal of Electrical Engineering and Computer Science, Vol. 22, No. 1, 89-96, 2021

[30] Frederic Rivoallon, Xillinx, "Measuring Device Performance and Utilization: A Competitive Overview", August 8, 2017, WPA496 (v1.0.1)

[31] Xing, Yufei, and Shuguo Li. "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA." IACR Transactions on Cryptographic Hardware and Embedded Systems (2021): 328-356.

[32] Zhang, Neng, Bohan Yang, Chen Chen, Shouyi Yin, Shaojun Wei, and Leibo Liu, "Highly efficient architecture of NewHope-NIST on FPGA using low-complexity NTT/INTT." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 49-72, doi: 10.13154/tches.v2020.i2.49-72.

[33] Fritzmann, Tim, Georg Sigl, and Johanna Sepúlveda. "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography." IACR Transactions on Cryptographic Hardware and Embedded Systems (2020): 239-280.

[34] Longa, Patrick, and Michael Naehrig, "Speeding up the number theoretic transform for faster ideal lattice-based cryptography." In International Conference on Cryptology and Network Security, pp. 124-139. Springer, Cham, 2016, doi: 10.1007/978-3-319-48965-0_8.

[35] Nguyen, Duc Tri, Viet B. Dang, and Kris Gaj. "High-Level Synthesis in Implementing and Benchmarking Number Theoretic Transform in Lattice-Based Post-Quantum Cryptography Using Software/Hardware Codesign." In ARC, pp. 247-257. 2020. doi: 10.1007/978-3-030-44534-8_19.

8. PHỤ LỤC

8.1 Các phần tối ưu cần lưu ý ở phần mềm Quartus

Phiên bản chính dùng để thực hiện nghiên cứu trong bài sử dụng các tối ưu về Speed của Quartus Prime Standard Edition 21.1. Ở chế độ này Quartus thực hiện sắp xếp vị trí của các thành phần trên FPGA gần nhau hơn nhằm tối ưu tốc độ. Hình 29 thể hiện chế độ tối ưu tổng hợp mạch được sử dụng.

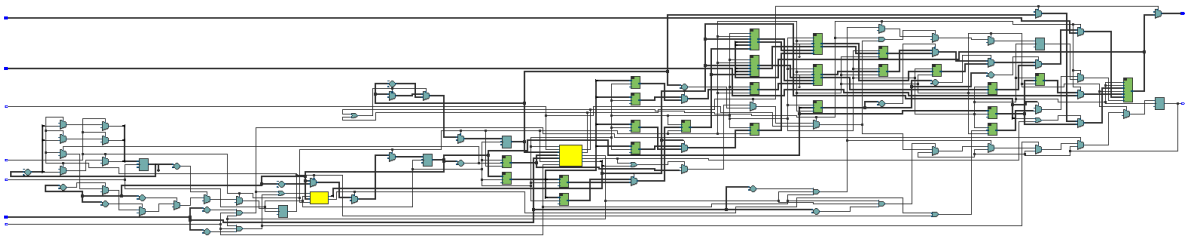
	Assignment Name	Value	Default Value	Entity Name	Section Id
1	COMPILER_SIGNATURE_ID	104252179705287.164144675607200	--	--	--
2	FITTER_EFFORT	Standard Fit	Auto Fit	--	--
3	MAX_CORE_JUNCTION_TEMP	85	--	--	--
4	MIN_CORE_JUNCTION_TEMP	0	--	--	--
5	MUX_RESTRUCTURE	Off	Auto	--	--
6	OPTIMIZATION_MODE	Aggressive Performance	Balanced	--	--
7	OPTIMIZATION_TECHNIQUE	Speed	Balanced	--	--
8	PARTITION_COLOR	-- (Not supported for targeted family)	--	--	Top
9	PARTITION_FITTER_PRESERVATION_LEVEL	-- (Not supported for targeted family)	--	--	Top
10	PARTITION_NETLIST_TYPE	-- (Not supported for targeted family)	--	--	Top
11	PHYSICAL_SYNTHESIS_COMBO_LOGIC	On	Off	--	--
12	PHYSICAL_SYNTHESIS_REGISTER_RETIMING	On	Off	--	--
13	PLACEMENT_EFFORT_MULTIPLIER	4.0	1.0	--	--
14	POWER_BOARD_THERMAL_MODEL	None (CONSERVATIVE)	--	--	--
15	POWER_PRESET_COOLING_SOLUTION	23 MM HEAT SINK WITH 200 LFPM AIRFLOW	--	--	--
16	PROJECT_OUTPUT_DIRECTORY	output_files	--	--	--
17	ROUTER_TIMING_OPTIMIZATION_LEVEL	MAXIMUM	Normal	--	--

Hình 29 Chế độ tối ưu khi tổng hợp trên Quartus

Ngoài ra, có rất nhiều tính năng tối ưu khác như Mux Restructure, Register Re-timing. Đây là các tính năng cho phép Quartus can thiệp sâu vào quá trình tổng hợp mạch để cho ra kết quả tốt nhất có thể. Trong đó, Quartus có thể thay đổi vị trí đặt các thanh ghi mà không gây ảnh hưởng đến kết quả đã mô phỏng.

8.2 Sơ đồ thiết kế chính theo Quartus

Sơ đồ thiết kế Netlist Viewer cho cái nhìn tổng quan về thiết kế được tổng hợp trên Quartus.



Hình 30 Sơ đồ Netlist Viewer của thiết kế tổng trên Quartus

Sơ đồ này có nhiều vai trò quan trọng trong quá trình tối ưu tốc độ của mạch hơn là để xem xét hình khối. Trên sơ đồ có thể đánh dấu các điểm Critical Path và xem các mạch tổ hợp ở giữa để tối ưu hóa và sắp xếp lại.

8.3 Phần truy xuất RAM, ROM đầy đủ

Bảng 7 Phần thứ tự truy xuất và thứ tự hệ số phương trình gốc trong bộ nhớ RAM cho NTT

Cycle #	Seq	Address	A4	A3	A2	A1
1	0	0	96	64	32	0
2	8	1	97	65	33	1
3	16	2	98	66	34	2
4	24	3	99	67	35	3
5	1	4	100	68	36	4
6	9	5	101	69	37	5
7	17	6	102	70	38	6
8	25	7	103	71	39	7
9	2	8	104	72	40	8
10	10	9	105	73	41	9
11	18	10	106	74	42	10
12	26	11	107	75	43	11
13	3	12	108	76	44	12
14	11	13	109	77	45	13
15	19	14	110	78	46	14
16	27	15	111	79	47	15
17	4	16	112	80	48	16
18	12	17	113	81	49	17
19	20	18	114	82	50	18
20	28	19	115	83	51	19
21	5	20	116	84	52	20
22	13	21	117	85	53	21

23	21	22	118	86	54	22
24	29	23	119	87	55	23
25	6	24	120	88	56	24
26	14	25	121	89	57	25
27	22	26	122	90	58	26
28	30	27	123	91	59	27
29	7	28	124	92	60	28
30	15	29	125	93	61	29
31	23	30	126	94	62	30
32	31	31	127	95	63	31
33	0	0	24	16	8	0
34	2	1	25	17	9	1
35	4	2	26	18	10	2
36	6	3	27	19	11	3
37	1	4	28	20	12	4
38	3	5	29	21	13	5
39	5	6	30	22	14	6
40	7	7	31	23	15	7
41	8	8	56	48	40	32
42	10	9	57	49	41	33
43	12	10	58	50	42	34
44	14	11	59	51	43	35
45	9	12	60	52	44	36
46	11	13	61	53	45	37
47	13	14	62	54	46	38
48	15	15	63	55	47	39
49	16	16	88	80	72	64
50	18	17	89	81	73	65
51	20	18	90	82	74	66
52	22	19	91	83	75	67
53	17	20	92	84	76	68
54	19	21	93	85	77	69
55	21	22	94	86	78	70
56	23	23	95	87	79	71
57	24	24	120	112	104	96
58	26	25	121	113	105	97
59	28	26	122	114	106	98
60	30	27	123	115	107	99
61	25	28	124	116	108	100
62	27	29	125	117	109	101
63	29	30	126	118	110	102
64	31	31	127	119	111	103
65	0	0	6	4	2	0
66	1	1	7	5	3	1
67	2	2	14	12	10	8
68	3	3	15	13	11	9

69	4	4	22	20	18	16
70	5	5	23	21	19	17
71	6	6	30	28	26	24
72	7	7	31	29	27	25
73	8	8	38	36	34	32
74	9	9	39	37	35	33
75	10	10	46	44	42	40
76	11	11	47	45	43	41
77	12	12	54	52	50	48
78	13	13	55	53	51	49
79	14	14	62	60	58	56
80	15	15	63	61	59	57
81	16	16	70	68	66	64
82	17	17	71	69	67	65
83	18	18	78	76	74	72
84	19	19	79	77	75	73
85	20	20	86	84	82	80
86	21	21	87	85	83	81
87	22	22	94	92	90	88
88	23	23	95	93	91	89
89	24	24	102	100	98	96
90	25	25	103	101	99	97
91	26	26	110	108	106	104
92	27	27	111	109	107	105
93	28	28	118	116	114	112
94	29	29	119	117	115	113
95	30	30	126	124	122	120
96	31	31	127	125	123	121
97	0	0	9	8	1	0
98	1	1	11	10	3	2
99	2	2	13	12	5	4
100	3	3	15	14	7	6
101	4	4	25	24	17	16
102	5	5	27	26	19	18
103	6	6	29	28	21	20
104	7	7	31	30	23	22
105	8	8	41	40	33	32
106	9	9	43	42	35	34
107	10	10	45	44	37	36
108	11	11	47	46	39	38
109	12	12	57	56	49	48
110	13	13	59	58	51	50
111	14	14	61	60	53	52
112	15	15	63	62	55	54
113	16	16	73	72	65	64
114	17	17	75	74	67	66

115	18	18	77	76	69	68
116	19	19	79	78	71	70
117	20	20	89	88	81	80
118	21	21	91	90	83	82
119	22	22	93	92	85	84
120	23	23	95	94	87	86
121	24	24	105	104	97	96
122	25	25	107	106	99	98
123	26	26	109	108	101	100
124	27	27	111	110	103	102
125	28	28	121	120	113	112
126	29	29	123	122	115	114
127	30	30	125	124	117	116
128	31	31	127	126	119	118

Bảng 8 Phần thứ tự truy xuất và thứ tự hệ số phương trình gốc trong bộ nhớ RAM cho INTT

Cycle #	Seq	Address	A4	A3	A2	A1
1	0	0	9	8	1	0
2	1	1	11	10	3	2
3	2	2	13	12	5	4
4	3	3	15	14	7	6
5	4	4	25	24	17	16
6	5	5	27	26	19	18
7	6	6	29	28	21	20
8	7	7	31	30	23	22
9	8	8	41	40	33	32
10	9	9	43	42	35	34
11	10	10	45	44	37	36
12	11	11	47	46	39	38
13	12	12	57	56	49	48
14	13	13	59	58	51	50
15	14	14	61	60	53	52
16	15	15	63	62	55	54
17	16	16	73	72	65	64
18	17	17	75	74	67	66
19	18	18	77	76	69	68
20	19	19	79	78	71	70
21	20	20	89	88	81	80
22	21	21	91	90	83	82
23	22	22	93	92	85	84
24	23	23	95	94	87	86
25	24	24	105	104	97	96
26	25	25	107	106	99	98
27	26	26	109	108	101	100

28	27	27	111	110	103	102
29	28	28	121	120	113	112
30	29	29	123	122	115	114
31	30	30	125	124	117	116
32	31	31	127	126	119	118
33	0	0	6	4	2	0
34	2	1	7	5	3	1
35	4	2	14	12	10	8
36	6	3	15	13	11	9
37	1	4	22	20	18	16
38	3	5	23	21	19	17
39	5	6	30	28	26	24
40	7	7	31	29	27	25
41	8	8	38	36	34	32
42	10	9	39	37	35	33
43	12	10	46	44	42	40
44	14	11	47	45	43	41
45	9	12	54	52	50	48
46	11	13	55	53	51	49
47	13	14	62	60	58	56
48	15	15	63	61	59	57
49	16	16	70	68	66	64
50	18	17	71	69	67	65
51	20	18	78	76	74	72
52	22	19	79	77	75	73
53	17	20	86	84	82	80
54	19	21	87	85	83	81
55	21	22	94	92	90	88
56	23	23	95	93	91	89
57	24	24	102	100	98	96
58	26	25	103	101	99	97
59	28	26	110	108	106	104
60	30	27	111	109	107	105
61	25	28	118	116	114	112
62	27	29	119	117	115	113
63	29	30	126	124	122	120
64	31	31	127	125	123	121
65	0	0	24	16	8	0
66	8	1	25	17	9	1
67	16	2	26	18	10	2
68	24	3	27	19	11	3
69	1	4	28	20	12	4
70	9	5	29	21	13	5
71	17	6	30	22	14	6
72	25	7	31	23	15	7
73	2	8	56	48	40	32

74	10	9	57	49	41	33
75	18	10	58	50	42	34
76	26	11	59	51	43	35
77	3	12	60	52	44	36
78	11	13	61	53	45	37
79	19	14	62	54	46	38
80	27	15	63	55	47	39
81	4	16	88	80	72	64
82	12	17	89	81	73	65
83	20	18	90	82	74	66
84	28	19	91	83	75	67
85	5	20	92	84	76	68
86	13	21	93	85	77	69
87	21	22	94	86	78	70
88	29	23	95	87	79	71
89	6	24	120	112	104	96
90	14	25	121	113	105	97
91	22	26	122	114	106	98
92	30	27	123	115	107	99
93	7	28	124	116	108	100
94	15	29	125	117	109	101
95	23	30	126	118	110	102
96	31	31	127	119	111	103
97	0	0	96	64	32	0
98	1	1	97	65	33	1
99	2	2	98	66	34	2
100	3	3	99	67	35	3
101	4	4	100	68	36	4
102	5	5	101	69	37	5
103	6	6	102	70	38	6
104	7	7	103	71	39	7
105	8	8	104	72	40	8
106	9	9	105	73	41	9
107	10	10	106	74	42	10
108	11	11	107	75	43	11
109	12	12	108	76	44	12
110	13	13	109	77	45	13
111	14	14	110	78	46	14
112	15	15	111	79	47	15
113	16	16	112	80	48	16
114	17	17	113	81	49	17
115	18	18	114	82	50	18
116	19	19	115	83	51	19
117	20	20	116	84	52	20
118	21	21	117	85	53	21
119	22	22	118	86	54	22

120	23	23	119	87	55	23
121	24	24	120	88	56	24
122	25	25	121	89	57	25
123	26	26	122	90	58	26
124	27	27	123	91	59	27
125	28	28	124	92	60	28
126	29	29	125	93	61	29
127	30	30	126	94	62	30
128	31	31	127	95	63	31

Bảng 9 Thứ tự truy xuất hệ số Twiddle Factor từ ROM cho cấu hình 2 x 2 BU cho NTT

Cycle	w00	w10	w11
1	64	32	96
2	64	32	96
3	64	32	96
4	64	32	96
5	64	32	96
6	64	32	96
7	64	32	96
8	64	32	96
9	64	32	96
10	64	32	96
11	64	32	96
12	64	32	96
13	64	32	96
14	64	32	96
15	64	32	96
16	64	32	96
17	64	32	96
18	64	32	96
19	64	32	96
20	64	32	96
21	64	32	96
22	64	32	96
23	64	32	96
24	64	32	96
25	64	32	96
26	64	32	96
27	64	32	96
28	64	32	96
29	64	32	96
30	64	32	96
31	64	32	96
32	64	32	96

33	16	8	72
34	16	8	72
35	16	8	72
36	16	8	72
37	16	8	72
38	16	8	72
39	16	8	72
40	16	8	72
41	80	40	104
42	80	40	104
43	80	40	104
44	80	40	104
45	80	40	104
46	80	40	104
47	80	40	104
48	80	40	104
49	48	24	88
50	48	24	88
51	48	24	88
52	48	24	88
53	48	24	88
54	48	24	88
55	48	24	88
56	48	24	88
57	112	56	120
58	112	56	120
59	112	56	120
60	112	56	120
61	112	56	120
62	112	56	120
63	112	56	120
64	112	56	120
65	4	2	66
66	4	2	66
67	68	34	98
68	68	34	98
69	36	18	82
70	36	18	82
71	100	50	114
72	100	50	114
73	20	10	74
74	20	10	74
75	84	42	106
76	84	42	106
77	52	26	90
78	52	26	90

79	116	58	122
80	116	58	122
81	12	6	70
82	12	6	70
83	76	38	102
84	76	38	102
85	44	22	86
86	44	22	86
87	108	54	118
88	108	54	118
89	28	14	78
90	28	14	78
91	92	46	110
92	92	46	110
93	60	30	94
94	60	30	94
95	124	62	126
96	124	62	126
97		9	1
98		11	3
99		13	5
100		15	7
101		25	17
102		27	19
103		29	21
104		31	23
105		41	33
106		43	35
107		45	37
108		47	39
109		57	49
110		59	51
111		61	53
112		63	55
113		73	65
114		75	67
115		77	69
116		79	71
117		89	81
118		91	83
119		93	85
120		95	87
121		105	97
122		107	99
123		109	101
124		111	103

125	121	113
126	123	115
127	125	117
128	127	119

Bảng 10 Bảng giá trị Twiddle Factor γ

STT	γ	γ^{-1}	$\gamma * k^{-2}$	$\gamma^{-1} * k^{-2}$
0	1	1	2285	2285
1	892	2549	872	2044
2	1848	2304	1508	1491
3	1455	723	2333	871
4	1432	630	3042	1422
5	1041	1212	1779	3021
6	1052	2617	282	961
7	1239	2308	1465	644
8	1089	2094	1602	1017
9	1868	319	602	3193
10	1795	1977	247	3321
11	554	2662	870	587
12	2760	1919	1474	622
13	314	1092	1755	1799
14	2009	1435	3203	3239
15	863	1651	1187	778
16	1355	2447	205	2004
17	3061	2298	156	1097
18	2102	331	2652	652
19	2444	2761	1807	430
20	1600	1729	758	2571
21	568	885	2899	1522
22	2998	1227	2677	677
23	1031	268	2232	3173
24	882	1974	1325	3124
25	1678	2466	2551	2142
26	1894	1320	90	126
27	2237	3015	1530	1574
28	1410	569	2707	1855
29	667	2775	2742	2459
30	1352	1534	8	3082
31	3010	1461	136	2727
32	1235	2240	2312	1727
33	1021	2090	2685	1864
34	712	2277	2368	3047
35	2117	2288	308	1550
36	2699	1897	1907	287
37	2606	1874	2458	996

38	1025	1481	1838	1821
39	780	2437	1285	2457
40	3273	535	1871	732
41	2377	2773	1846	1218
42	461	2513	1421	3009
43	1179	3281	854	177
44	69	193	1202	1577
45	1173	403	460	2051
46	3296	807	1162	3058
47	2768	2789	3109	1159
48	450	1339	2918	264
49	992	2037	3000	603
50	219	3253	1065	2777
51	394	583	1460	555
52	40	2580	1517	2970
53	680	2110	2486	958
54	1573	2474	2314	448
55	109	733	2719	418
56	1853	2393	2946	1787
57	1540	2099	147	2455
58	2877	2865	2499	1711
59	2303	756	2535	3038
60	2532	2786	3147	962
61	3096	1143	235	1819
62	2697	1438	666	107
63	2572	1847	1335	2552
64	447	3046	2721	2500
65	941	375	2980	1322
66	2681	1197	725	2036
67	2300	1637	2338	2078
68	2481	2642	3127	1493
69	2229	1722	3224	3221
70	1274	2647	1544	2931
71	1684	939	2945	1739
72	1996	1426	130	2648
73	642	1063	2210	2114
74	927	650	951	516
75	2443	1409	2851	422
76	1583	1062	1861	3158
77	279	2804	1676	2144
78	1414	2319	1860	2476
79	735	1703	1659	3083
80	2508	296	1571	573
81	2688	1584	75	817
82	2419	289	1275	1223
83	1175	17	1701	2226

84	1	1	2285	2285
85	17	1175	2226	1701
86	289	2419	1223	1275
87	1584	2688	817	75
88	296	2508	573	1571
89	1703	735	3083	1659
90	2319	1414	2476	1860
91	2804	279	2144	1676
92	1062	1583	3158	1861
93	1409	2443	422	2851
94	650	927	516	951
95	1063	642	2114	2210
96	1426	1996	2648	130
97	939	1684	1739	2945
98	2647	1274	2931	1544
99	1722	2229	3221	3224
100	2642	2481	1493	3127
101	1637	2300	2078	2338
102	1197	2681	2036	725
103	375	941	1322	2980
104	3046	447	2500	2721
105	1847	2572	2552	1335
106	1438	2697	107	666
107	1143	3096	1819	235
108	2786	2532	962	3147
109	756	2303	3038	2535
110	2865	2877	1711	2499
111	2099	1540	2455	147
112	2393	1853	1787	2946
113	733	109	418	2719
114	2474	1573	448	2314
115	2110	680	958	2486
116	2580	40	2970	1517
117	583	394	555	1460
118	3253	219	2777	1065
119	2037	992	603	3000
120	1339	450	264	2918
121	2789	2768	1159	3109
122	807	3296	3058	1162
123	403	1173	2051	460
124	193	69	1577	1202
125	3281	1179	177	854
126	2513	461	3009	1421
127	2773	2377	1218	1846

8.4 Các thuật ngữ được sử dụng

Bảng 11 Bảng thuật ngữ

Thuật ngữ	Định nghĩa
NTT	Number Theoretic Transform
INTT	Inverse Number Theoretic Transform hay còn được viết là NTT^{-1}
FFT/IFFT	Fast Fourier Transform
NTT/FFT layer/lớp	Số lớp chỉ tất cả các phép toán trên cũng một tầng trên cây tính toán NTT/FFT. Khi nói đến gộp lớp, ý chỉ là thực hiện nhiều tính toán trên nhiều tầng khác nhau dùng nhiều BU
Rút gọn Modulo	Trong mã hóa điện tử, số học luôn luôn thực hiện trên các số nguyên. Mặc dù tất cả các giá trị này phải được biểu diễn theo modulo q, tại một số thời điểm các số nguyên có thể trở nên lớn đến mức chúng sẽ làm tràn thanh ghi. Một bước giảm làm cho các số nhỏ trở lại, nhưng đảm bảo duy trì rằng chúng vẫn giữ nguyên giá trị modulo q gọi là phép rút gọn Modulo. Thông thường, hai thuật toán khác nhau được sử dụng: Barett và Montgomery.
Butterfly Unit (BU)	Là khối tính toán của FFT/IFFT, hay còn có định nghĩa là tính toán DIT và DIF
Miền thời gian (Time Domain)	
q	Số Modulo giới hạn vành. Trong trường hợp của Kyber, $q = 3329$.
n	Độ dài đa thức, trường hợp Kyber, $n = 256$
n-th primitive root of unity	Gốc hội tụ nguyên thủy thứ n hay n-th primitive root of unity trong miền Z_q là một số ω sao cho $\omega^n = 1 \; mod \; q$.
Twiddle Factor	Một thuật ngữ chỉ một số là kết quả của phép mũ của primitive root of unity với một số nguyên dương.
Miền NTT/ Miền tần số	Miền NTT là miền mà các kết quả thuộc về sau khi qua phép biến đổi NTT. Miền tần số, miền thời gian là một cách gọi quen thuộc của phép toán FFT/IFFT phổ biến.
Miền thời gian	Miền gốc ban đầu, chưa qua biến đổi NTT

- **Trang 3:** Tờ nhiệm vụ luận văn thạc sĩ

- **Trang 4:** Lời cám ơn

- **Trang 5:** Tóm tắt luận văn thạc sĩ (Tiếng Việt và Tiếng Anh).

- **Trang 6**: Lời cam đoan của tác giả LV

- **Mục lục**

- **Toàn bộ nội dung luận văn (thực hiện theo đề cương đã bảo vệ)**

- **Tài liệu tham khảo**

- **Phụ lục** (nếu có)

PHẦN LÝ LỊCH TRÍCH NGANG

Họ và tên: Nguyễn Tuấn Hùng

Ngày, tháng, năm sinh: 11/06/1997 Nơi sinh: TP. Hồ Chí Minh

Địa chỉ liên lạc: 110/44/22 Tô Hiệu, P. Hiệp Tân, Q. Tân Phú, TP. HCM

QUÁ TRÌNH ĐÀO TẠO

Từ	Đến	Chương trình đào tạo	Nơi đào tạo	Thành tích
2015	2019	Kỹ Sư	Trường Đại học Bách Khoa – Đại học Quốc Gia TP.HCM	XL: Giỏi

QUÁ TRÌNH CÔNG TÁC

Từ	Đến	Chức vụ	Nơi công tác	Thành tích
2018	2019	Kỹ Sư	Công ty Arrive Technologies Việt Nam	