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

TÓM TẮT LUẬN VĂN THẠC SĨ

TP. HỒ CHÍ MINH, tháng 01 năm 2022

MỤC LỤC

1. MỞ ĐẦU 1
1.1 Lý do chọn đề tài 1
1.2 Mục đích 1
1.3 Đối tượng và phạm vi nghiên cứu 1
1.3.1 Đối tượng nghiên cứu 1
1.3.2 Phạm vi nghiên cứu 1
1.4 Ý nghĩa khoa học và thực tiễn của đề tài nghiên cứu 1
1.4.1 Ý nghĩa khoa học 1
1.4.2 Ý nghĩa thực tiễn 1
2. TỔNG QUAN 2
2.1 Tình hình nghiên cứu trong và ngoài nước 2
2.1.1 Tình hình nghiên cứu ngoài nước 2
2.1.2 Tình hình nghiên cứu trong nước 2
2.2 Nhiệm vụ đề tài 2
3. NHỮNG NGHIÊN CỨU LÝ THUYẾT 3
3.1 Lý thuyết về mã hóa bất đối xứng 3
3.2 Lý thuyết về CRYSTALS-Kyber 4
3.3 Lý thuyết về Number Theoretic Transform (NTT) 6
3.3.1 NTT/INTT và độ phức tạp so với phép nhân đa thức 6
3.3.2 Phiên bản NTT/INTT tối ưu được sử dụng 7
3.4 Lý thuyết về phép toán rút gọn modulo Exact-KRED 9
3.5 Về bộ nhớ BRAM M10K trên FPGA Cyclone V 10
3.6 Xử lý tính toán lý thuyết trên phần mềm máy tính 10
4. TRÌNH BÀY, ĐÁNH GIÁ VÀ BÀN LUẬN KẾT QUẢ 11

4.1 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 11
4.1.1 Thiết kế butterfly unit xử lý CT/GS 11
4.1.1.1 Thiết kế bộ xử lý rút gọn modulo Exact-KRED 11
4.1.2 Bộ rút gọn modulo chia nửa 12
4.1.3 Thiết kế butterfly unit xử lý CT/GS 12
4.1.4 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 14
4.2 Kết quả tổng hợp và mô phỏng 23
4.2.1 Kết quả mô phỏng ModelSim 23
4.2.2 Kết quả tổng hợp Quartus 25
4.3 Đánh giá, bàn luận và so sánh kết quả 26
5. KẾT LUẬN VÀ KIẾN NGHỊ NHỮNG NGHIÊN CỨU TIẾP THEO 28
5.1 Các hướng tối ưu thiết kế phần cứng xử lý NTT và INTT 28
5.2 Thiết kế phần cứng xử lý CRYSTALS-Kyber 29
5.3 Kết luận 29
6. DANH MỤC CÔNG TRÌNH CÔNG BỐ CỦA TÁC GIẢ 29
7. DANH MỤC TÀI LIỆU THAM KHẢO 30

DANH SÁCH HÌNH MINH HỌA

Hình 1 Minh họa mã hóa bất đối xứng [23] 3

Hình 2 Ba giải thuật tao mã, mã hóa và giải mã của Kyber 4

Hình 3 NTT trong giải thuật tạo mã của Kyber 4

Hình 4 NTT trong giải thuật mã hóa của Kyber 5

Hình 5 NTT trong giải thuật giải mã của Kyber 5

Hình 6 Sơ đồ khối của bộ rút gọn Modulo Exact-KRED 12

Hình 7 Phần cứng butterfly unit xử lý CT/GS 13

Hình 8 Bộ Butterfly Unit kết hợp CT/GS cho thuật toán độ phức tạp thấp 14

Hình 9 Phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 15

Hình 10 Sơ đồ khối bộ xử lý NTT/INTT 17

Hình 11 Bộ SIPO Writeback vào nối tiếp ra song song với các ô nhớ 12-bit 17

Hình 12 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho NTT (n=128, 40 cycles đầu tiên) 18

Hình 13 Thứ tự địa chỉ truy xuất tại ROM cho NTT (n=128, 40 cycles đầu tiên) 19

Hình 14 Thứ tự địa chỉ truy xuất và thứ tự dữ liệu sắp xếp tại RAM cho INTT (n=128, 40 cycles đầu tiên) 20

Hình 15 Thứ tự địa chỉ truy xuất tại ROM cho INTT (n=128, 40 cycles đầu tiên) 21

Hình 16 Mô phỏng trên dạng sóng kết quả của thiết kế Exact-KRED 23

Hình 17 Mô phỏng trên dạng sóng kết quả của thiết kế BU. 24

Hình 18 Mô phỏng dạng sóng kết quả của phần cứng xử lý NTT và INTT 24

Hình 19 Kết quả tổng hợp tài nguyên trên Quartus 25

Hình 20 Kết quả tốc độ mạch trường hợp Slow 1100 mV 85C 25

Hình 21 Kết quả tốc độ mạch trường hợp Fast 1100 mV 0C 26

Hình 22 Kết quả tốc độ mạch trung bình 26

DANH SÁCH BẢNG SỐ LIỆU

Bảng 1 Thông số của từng phiên bản Kyber 5

Bảng 2 Tìm giá trị $\gamma 2n = \omega n$, ... 10

Bảng 3 Thực hiện tính $\gamma 2n$ ở các giá trị mũ khác nhau (bảng tới mức mũ 24) 10

Bảng 4 Bảng chân phần cứng butterfly unit xử lý CT/GS 13

Bảng 5 Bảng chân phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber 15

Bảng 6 Thiết kế đề xuất so với các nghiên cứu NTT tương tự trước đây (n = 256) 27

1. MỞ ĐẦU

1.1 Lý do chọn đề tài

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.

2.1.2 Tình hình nghiên cứu trong nước

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.

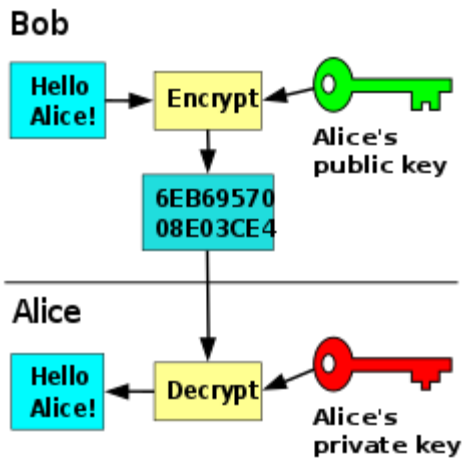
Nội dung 3: Đánh giá, bàn luận về thiết kế. So sánh kết quả đạt được với các nghiên cứu trước.

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

3.1 Lý thuyết về mã hóa bất đối xứng



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

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: $\mathbf{t} := \mathsf{Compress}_q(\mathbf{A}\mathbf{s} + \mathbf{e}, d_t)$
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 2 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 $pk \in \mathcal{B}^{12 \cdot k \cdot n/8 + 32}$

1: $d \leftarrow \mathcal{B}^{32}$
2: $(\rho, \sigma) := \mathrm{G}(d)$
3: $N := 0$
4: **for** i from 0 to $k-1$ **do** ▷ Generate matrix $\hat{\mathbf{A}} \in R_q^{k \times k}$ in NTT domain
5: **for** j from 0 to $k-1$ **do**
6: $\hat{\mathbf{A}}[i][j] := \mathsf{Parse}(\mathsf{XOF}(\rho, j, i))$
7: **end for**
8: **end for**
9: **for** i from 0 to $k-1$ **do** ▷ Sample $\mathbf{s} \in R_q^k$ from B_{η_1}
10: $\mathbf{s}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
11: $N := N + 1$
12: **end for**
13: **for** i from 0 to $k-1$ **do** ▷ Sample $\mathbf{e} \in R_q^k$ from B_{η_1}
14: $\mathbf{e}[i] := \mathsf{CBD}_{\eta_1}(\mathsf{PRF}(\sigma, N))$
15: $N := N + 1$
16: **end for**
17: $\hat{\mathbf{s}} := \mathsf{NTT}(\mathbf{s})$
18: $\hat{\mathbf{e}} := \mathsf{NTT}(\mathbf{e})$
19: $\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
20: $pk := (\mathsf{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q) \| \rho)$ ▷ $pk := \mathbf{A}\mathbf{s} + \mathbf{e}$
21: $sk := \mathsf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$ ▷ $sk := \mathbf{s}$
22: **return** (pk, sk)

Hình 3 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 4 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 $m \in \mathcal{B}^{32}$

```
u := Decompress_q(Decode_du(c), d_u)
v := Decompress_q(Decode_dv(c + d_u · k · n/8), d_v)
ŝ := Decode_12(sk)
m := Encode_1(Compress_q(v − NTT^{-1}(ŝ^T ∘ NTT(u)), 1))   ▷ m := Compress_q(v − s^T u, 1))
return m
```

Hình 5 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)

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]:

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

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

Với ω_n là primitive nth root of unity.

3.3.2 Phiên bản NTT/INTT tối ưu được sử dụng

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: â = bit − reverse(a)
2: for (i = 1; i < l; i++) do
3:     m = 2^i
4:     for (j = 0; j < m/2 − 1; j++) do
5:         ω ← γ_2N^((2j+1)N/m)
6:         for (k = 0; k < N/m − 1; k++) do
7:             u ← â[k + j]
8:             t ← ω · â[k + j + m/2] mod q
9:             â[k + j] = u + t mod q
10:            â[k + j + m/2] = u − t mod q
11:        end for
12:    end for
13: end for
14: return â
```

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

Input: polynomial $\hat{a}(a_0, a_1, \ldots, a_{n-1})$ 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: $a = INTT(\hat{a})$

```
1: for (i = 1; i < l; i++) do
2:     m = 2^(l−i)
3:     for (j = 0; j < m/2 − 1; j++) do
4:         ω ← γ_2N^(−(2j+1)N/m)
5:         for (k = 0; k < N/m − 1; k++) do
```

6:	$u \leftarrow \hat{a}[k+j]$
7:	$t \leftarrow \hat{a}[k+j+m/2]\ mod\ q$
8:	$\hat{a}[k+j] = \frac{u+t}{2} mod\ q$
9:	$\hat{a}[k+j+m/2] = \frac{u-t}{2} \cdot\ \omega\ mod\ q$
10:	**end for**
11:	**end for**
12:	**end for**
13:	$a = bit - reverse(\hat{a})$
14:	**return** a

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

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

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.

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

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

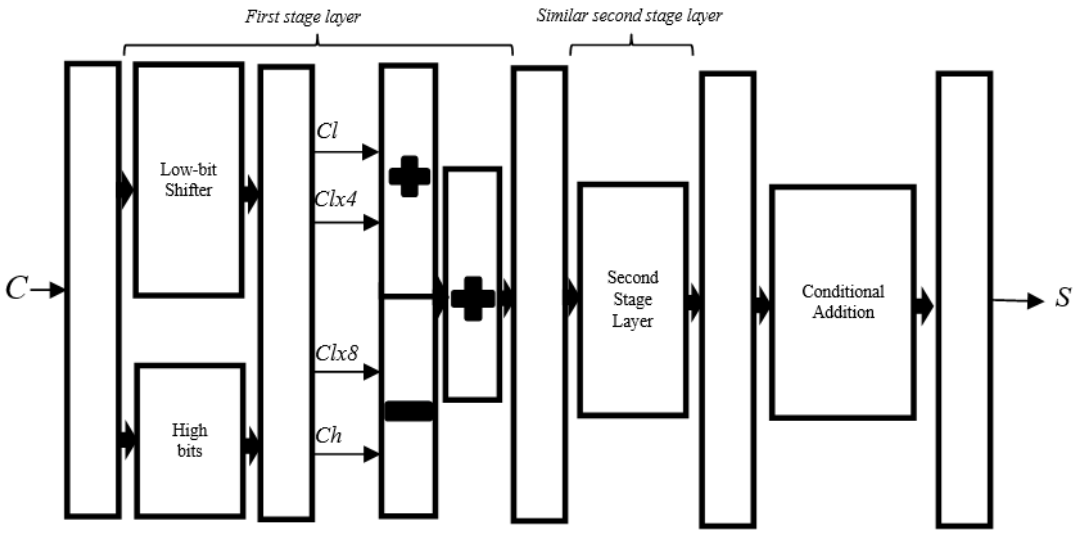
4.1 Thiết kế phần cứng xử lý NTT và INTT cho CRYSTALS-Kyber

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 6 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:

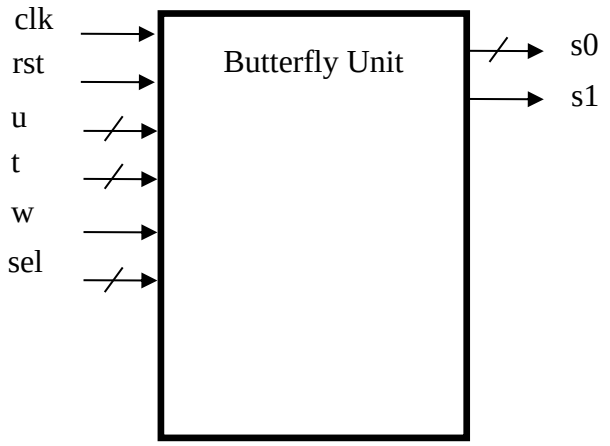
$$\frac{x}{2} mod\ q \equiv \left(2 * \left\lfloor\frac{x}{2}\right\rfloor + 1\right)\frac{q+1}{2} \equiv \left\lfloor\frac{x}{2}\right\rfloor (q+1) + \frac{q+1}{2} \equiv \left\lfloor\frac{x}{2}\right\rfloor + \frac{q+1}{2} mod\ q \quad (6)$$

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 7 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

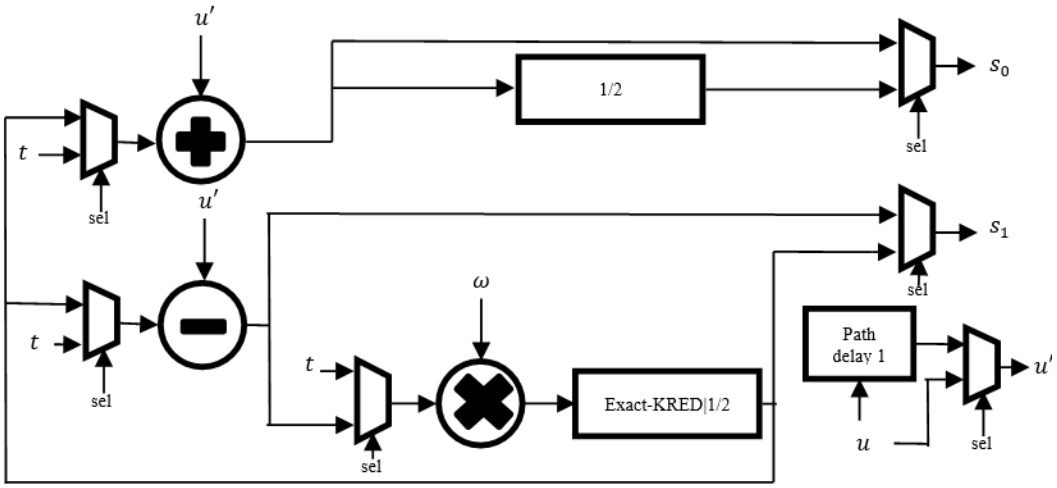
Từ hình 11, chúng tôi thiết kế bộ bướm phần cứng phù hợp cho cả giải thuật CT và GS. Tài nguyên của bộ BU này bao gồm bộ cộng số học modulo (một cộng, một trừ), một bộ nhân DSP và một bộ rút gọn modulo Exact-KRED, và hai bộ rút gọn modulo chia nửa. Bộ rút gọn modulo chia nửa cho mức tiêu thụ tài nguyên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

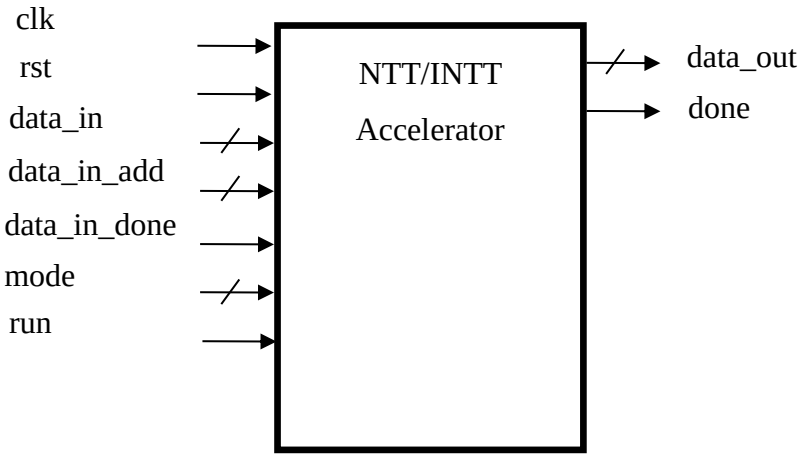
Đối với giải thuật CT, tín hiệu theo đường dẫn trên của mux được thực hiện, trong đó bộ cộng mong đợi kết quả nhân mô-đun sẽ đến sau 11 clock kể từ thời điểm nhập dữ liệu. Độ trễ 2 clock được thêm vào giải thuật CT để phù hợp với độ trễ thực hiện giải thuật GS.

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 8 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 9 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.

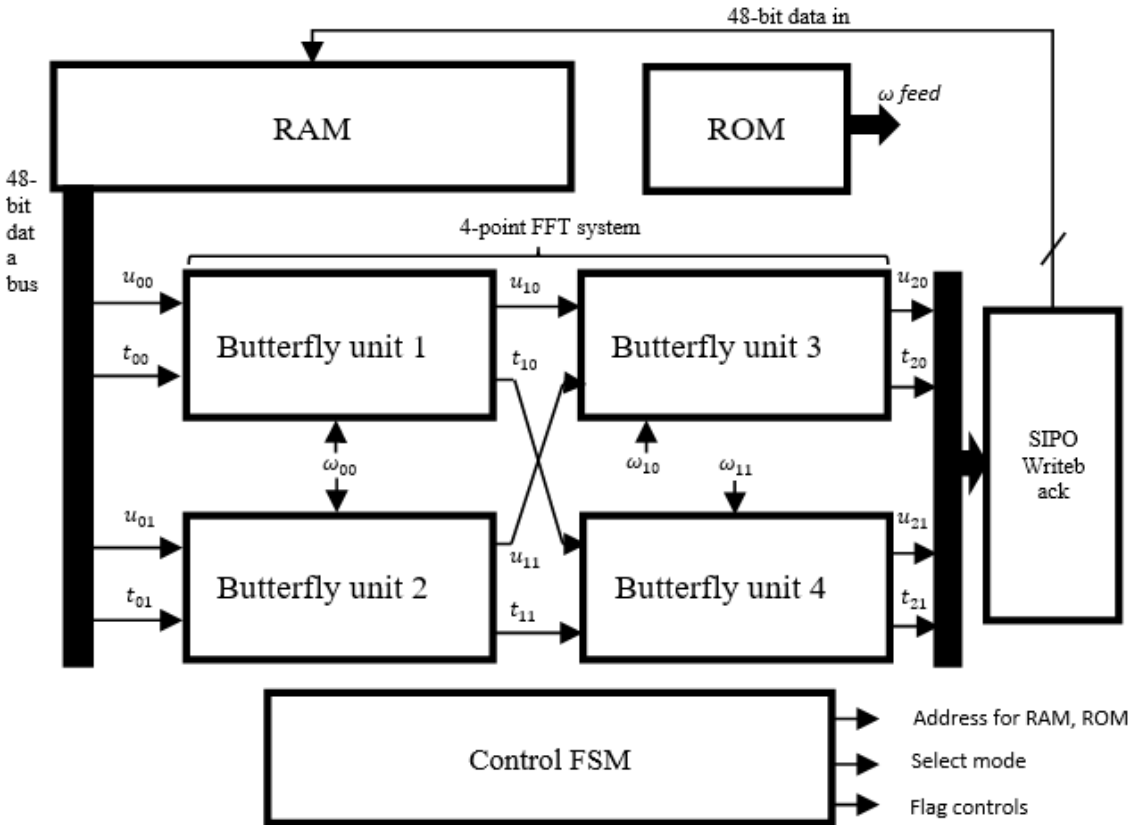
Dữ liệu tính toán trên mỗi ngõ ra của 2 BU hàng sau được đặt trên một thanh ghi dịch nối tiếp 12-bit sang song song 48-bit (SIPO). Sau 4 chu kỳ đầu tiên có kết quả là một chuỗi 48-bit ở mỗi đầu u_{20}, t_{20}, u_{21}, t_{21}, kết quả sẽ được ghi lần lượt trở lại vào RAM từ thanh ghi đầu tiên. Mỗi chu 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.

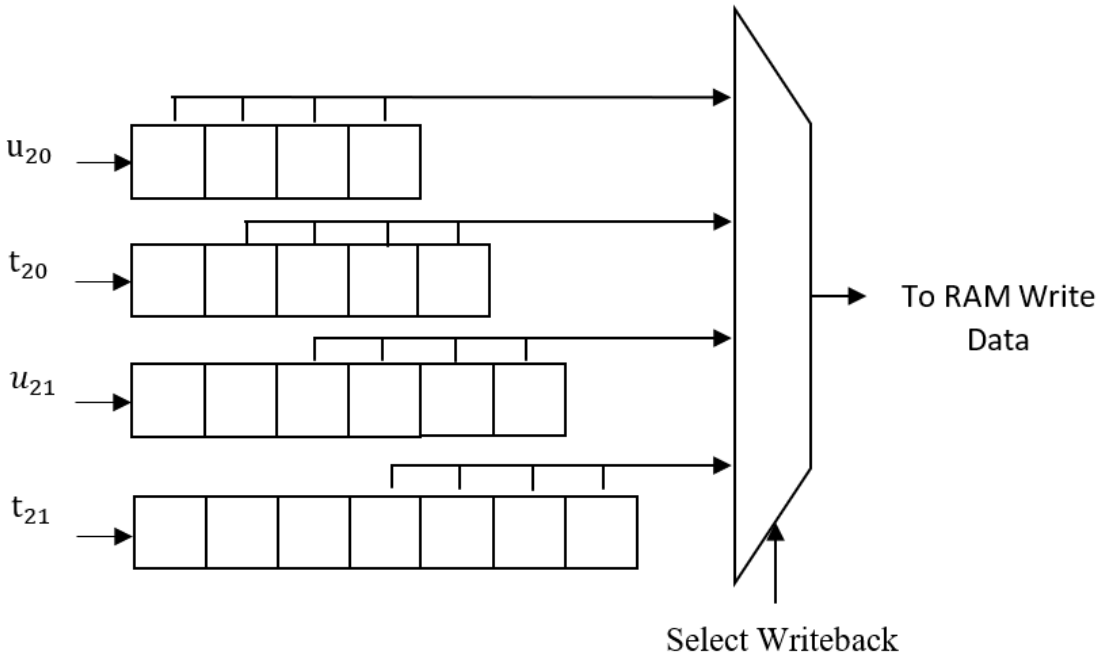
Mỗi phép tính NTT / INTT tốn 30 xung nhịp cho đến khi dữ liệu được ghi trở lại RAM. Độ trễ này là phù hợp để tránh xung đột đọc và ghi dữ liệu vào địa chỉ chưa được truy xuất để tính toán trước đó. Trong đó độ trễ của 2 BU là 26 clock, 2 clock từ truy xuất RAM và 2 clock độ trễ điều khiển.

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.

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 10 Sơ đồ khối bộ xử lý NTT/INTT



Hình 11 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 12 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 13 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 14 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 15 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**

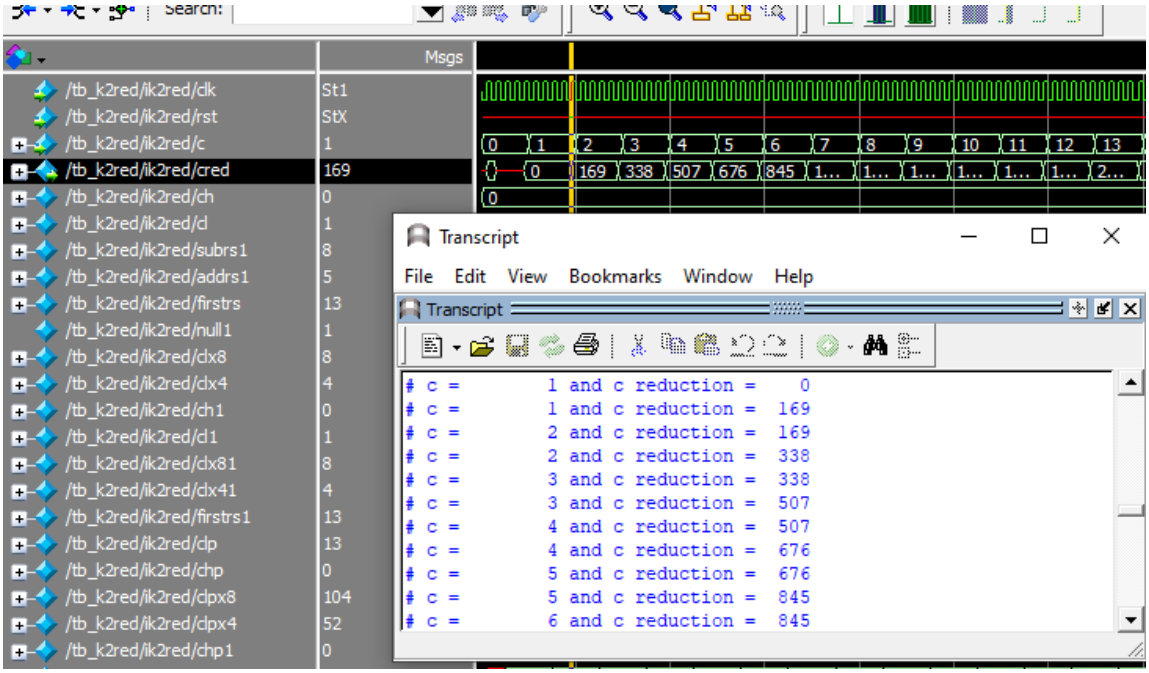
```
18:     end for
19:     c = c >> 2
19: end for
20: return a(x)
```

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.

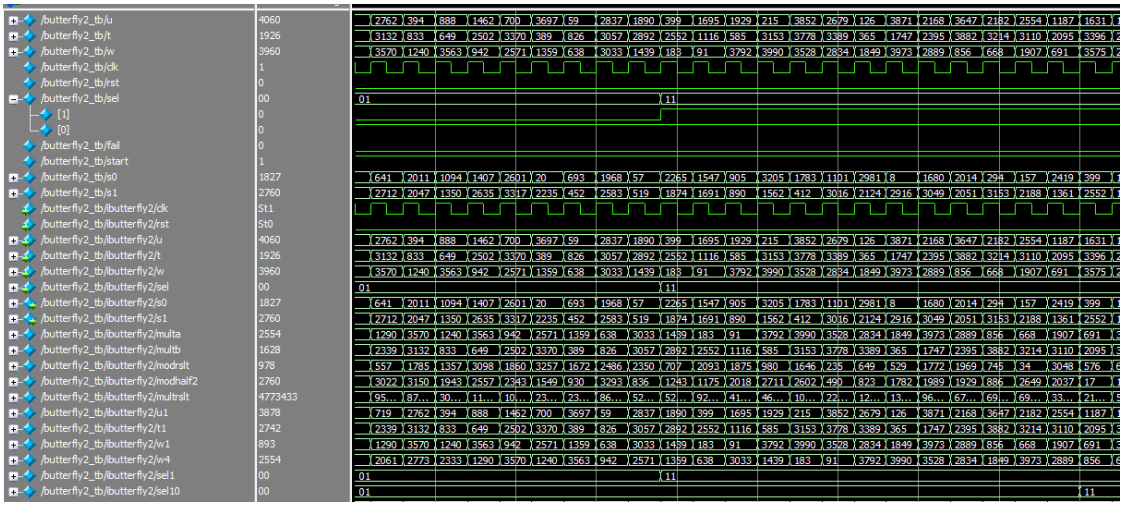
4.2.1 Kết quả mô phỏng ModelSim

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.



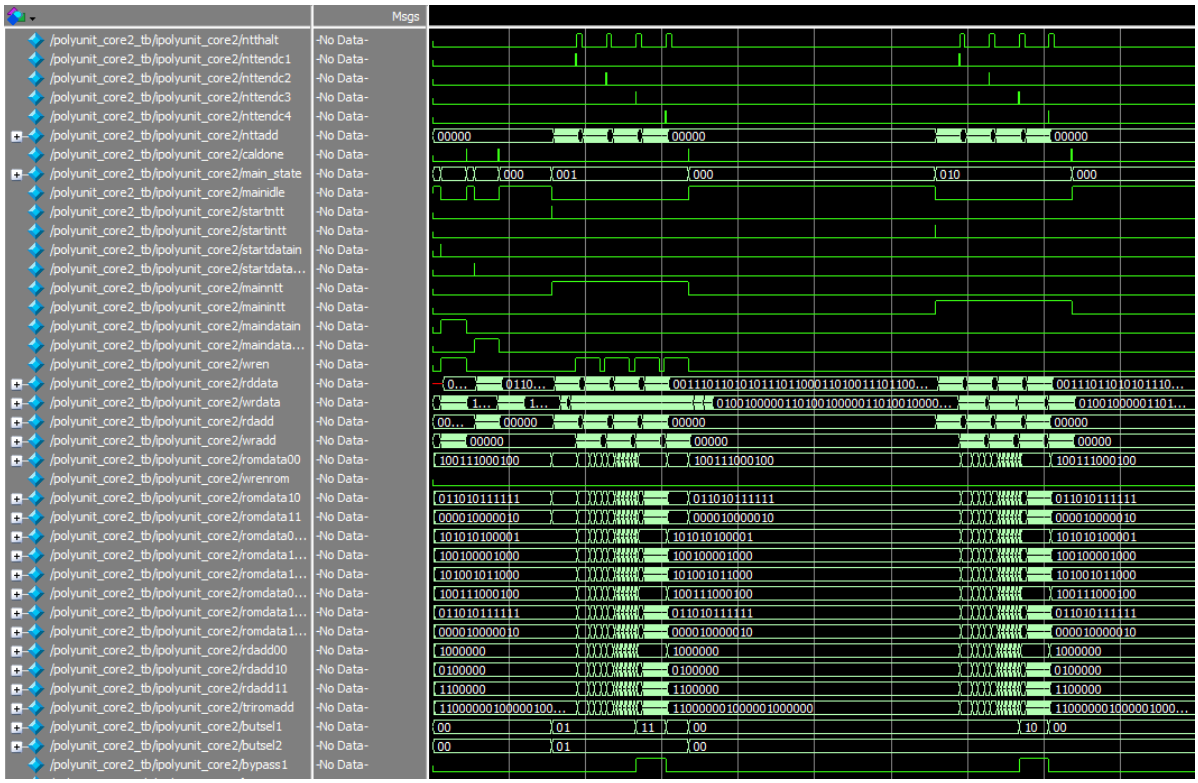
Hình 16 Mô phỏng trên dạng sóng kết quả của thiết kế Exact-KRED

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 17 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 18 Mô phỏng dạng sóng kết quả của phần cứng xử lý NTT và INTT

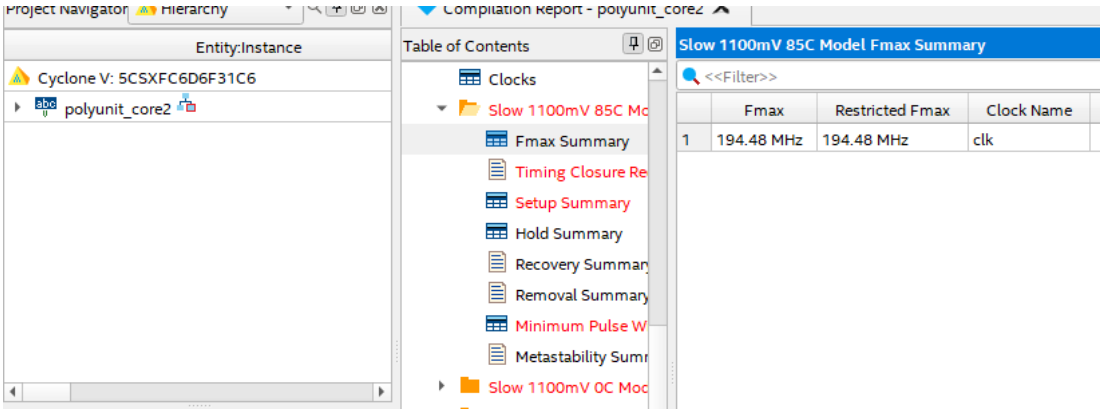
4.2.2 Kết quả tổng hợp Quartus

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 %)

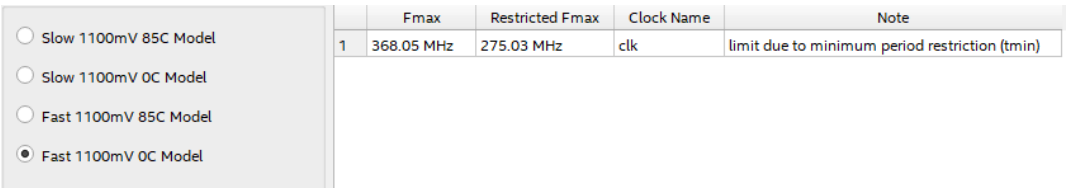
Hình 19 Kết quả tổng hợp tài nguyên trên Quartus

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.



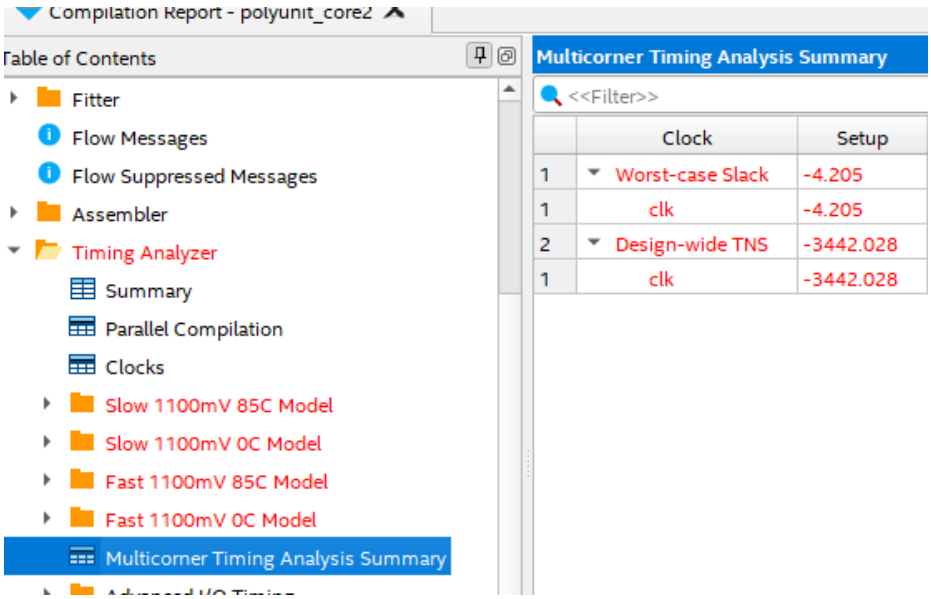
Hình 20 Kết quả tốc độ mạch trường hợp Slow 1100 mV 85C

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



Hình 21 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 22 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.

5.2 Thiết kế phần cứng xử lý CRYSTALS-Kyber

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.