بهینه‌سازی ماژول NTT در الگوریتم رمزنگاری پساکوانتوم CrystalsKyber

محمد غفاری[۱]، حاتم عبدلی[۲]

[۱]دانشجوی کارشناسی ارشد مهندسی معماری سیستم‌های کامپیوتری
mohammad.g7777@gmail.com
[۲]استادیار، گروه کامپیوتر، دانشکده مهندسی، دانشگاه بوعلی سینا، همدان
abdoli@basu.ac.ir

چکیده

با توجه به اینکه کامپیوترهای کوانتومی توان محاسباتی خیلی بیشتری نسبت به کامپیوترهای کلاسیک دارند، این مسئله باعث ایجاد چالش در حوزه رمزنگاری شده است، به‌طوریکه پیش‌بینی می‌شود در سال‌های آتی کامپیوترهای کوانتومی به اندازه‌ای قدرتمند شوند که بتوانند الگوریتم‌های رمزنگاری کلید عمومی را بشکنند. به منظور حل این مشکل الگوریتم‌های رمزنگاری پساکوانتوم مطرح شده و در حال تکامل هستند. یکی از الگوریتم‌های راه یافته به مراحل نهایی انتخاب الگوریتم‌های پساکوانتوم، الگوریتم CRYSTALSKYBER است. یکی از ماجول‌های پیچیده این الگوریتم واحد NTT است که می‌توان با بهینه سازی آن زمان اجرای الگوریتم را کاهش داد. در این پژوهش، برای پیاده‌سازی واحد NTT که قبلا با پایه دو صورت می‌گرفت ، از پایه چهار استفاده شده‌است و این امر باعث کاهش زمان اجرا شده‌است. برای پیاده‌سازی NTT پایه چهار متناسب با Kyber لازم است تغییراتی در NTT رخ دهد، بطوریکه در واحد حافظه روش پیشنهادی، تولید آدرس و همچنین تعداد ماجول‌های حافظه جهت خواندن و نوشتن همزمان تغییر کرده است. نتایج پیاده‌سازی روش پیشنهادی روی دو تراشه متفاوت FPGA با استفاده از نرم افزار Vivado نشان می‌دهد که در ازای افزایش جزیی منابع موردنیاز، زمان اجرا در Artix7 در مقایسه با پیاده‌سازی های مشابه بیش از ۲۸ درصد کاهش یافته‌است.

کلمات کلیدی: رمزنگاری پساکوانتوم، CrystalsKyber، NTT، پایه چهار، ضرب چند جمله ای، بهینه‌سازی سخت‌افزار.

۱ مقدمه

کامپیوترهای کوانتومی، کامپیوترهایی هستند که برای انجام محاسبات از خاصیت‌های فیزیک کوانتوم استفاده می‌کنند که این ویژگی باعث شده عملکرد خارق العاده‌ای نسبت کامپیوترهای کلاسیک که در حال حاضر استفاده می‌شوند، داشته باشند و این عملکرد شگفت‌انگیز باعث جلب توجه بسیاری شده‌است. در

کامپیوترهای کلاسیک از بیت استفاده می‌شود در حالی که کامپیوترهای کوانتومی مبتنی بر کیوبیت است [۱].

امروزه مؤسسات تحقیقاتی و شرکت‌های بزرگی مانند گوگل، IBM، NVIDIA و مایکروسافت روی کامپیوترهای کوانتومی سرمایه‌گذاری کرده‌اند و IBM، Q System One و گوگل، Sycamore و USTC، Jiuzhang 2 را معرفی کرده‌اند. پیش‌بینی شده‌است که حدوداً تا سال ۲۰۳۰ کامپیوترهای کوانتومی به‌قدری قدرتمند می‌شوند که می‌توانند الگوریتم‌های رمزنگاری کلید عمومی را بشکنند [۱]،[۲]. برای حل این مشکل می‌بایست از الگوریتم‌هایی استفاده شود که در مقابل حملات کوانتومی مقاوم باشند. انجمن ملی استاندارد و تکنولوژی آمریکا (NIST) یک فراخوانی در رابطه با این موضوع در اواخر سال ۲۰۱۶ منتشر کرد و به این صورت الگوریتم‌های رمزنگاری پساکوانتوم (PQC)[1] و امضای دیجیتال پساکوانتوم به وجود آمدند.

در دور سوم این رقابت، در سال ۲۰۲۰ چهار الگوریتم رمزنگاری کلید عمومی Saber، Classic McEliece، NTRU و CRYSTALSKYBER به عنوان فینالیست پذیرفته شده‌اند و پنج الگوریتم Bike، FrodoKEM، HQC، NTRUprime و SIKE به‌عنوان الگوریتم‌های جایگزین در نظر گرفته شده‌اند. سه الگوریتم Saber، NTRU و CRYSTALSKYBER مبتنی بر lattice و الگوریتم Classic McEliece مبتنی بر کد هستند. همچنین سه الگوریتم امضای دیجیتال CRYSTALSDILITHIUM، Rainbow و Falcon به عنوان فینالیست پذیرفته شده‌اند و سه الگوریتم امضای دیجیتال GeMSS، Picnic و SPHINCS+ به‌عنوان الگوریتم امضای دیجیتال جایگزین انتخاب شده‌اند. دو الگوریتم امضای دیجیتال CRYSTALSDILITHIUM و Falcon مبتنی بر lattice هستند و الگوریتم امضای دیجیتال Rainbow مبتنی بر چندمتغیره است.

در پیاده‌سازی سخت افزاری الگوریتم‌های رمزنگاری پساکوانتوم به دو موضوع توجه شده است: ۱ زمان اجرا و ۲ منابع موردنیاز. به طور کلی برای بهبود پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری پساکوانتوم از دو رویکرد اصلی استفاده می‌شود:

- بهبود کارایی بخش‌های پیچیده الگوریتم مانند بهینه‌سازی بخش ضرب چندجمله‌ای با استفاده از روش‌های حساب کامپیوتری
- استفاده از تکنیک‌های مختلف طراحی در سطح معماری کامپیوتر (مانند pipelining) با هدف افزایش کارایی سخت‌افزار الگوریتم

در کارهایی که تاکنون انجام شده‌است در بعضی از مقالات به چگونگی کاهش زمان اجرا توجه کرده‌اند، عده‌ای دیگر تمرکزشان بر کاهش منابع است و برخی به هردو رویکرد توجه داشته‌اند.

در این مقاله با اعمال تغییراتی در واحد NTT، به‌عنوان یکی از بخش‌های پیچیده الگوریتم Kyber، بهبود زمان اجرای الگوریتم Kyber حاصل شده است که جزییات این تغییرات در ادامه مقاله توضیح داده شده‌است.

[1] Post Quantum Cryptography

۱.۱ آشنایی با Kyber

Kyber یک سازوکار کپسوله‌سازی کلید (KEM) است که امنیت آن براساس سختی حل کردن مسئله یادگیری با خطا در lattice است [۳]. ساختار Kyber از دو نوع پیروی می‌کند، یکی CPA و دیگری CCA است. CPA نوعی حمله است که در آن فرد مهاجم هم به متن رمزشده و هم به متن آشکار دسترسی دارد و سعی می‌کند که متوجه شود که متن چگونه رمز شده‌است و CCA نوعی حمله است که فرد مهاجم می‌تواند با متن رمزشده انتخابی به متن آشکار دست پیدا کند. برای نمایش حلقه چندجمله‌ای $Z[X]/(Xn+1)$ از R و برای نمایش حلقه $Zq[X]/(Xn+1)$ از Rq استفاده می‌شود و به‌طوری که $n=256$، $n'=9$ و $q=3329$ است. از حروف بزرگ برای نشان دادن ماتریس‌ها و حروف کوچک برای نشان دادن بردارهایی با ضرایبی در R و Rq استفاده می‌شود.

در ادامه با اجزای Kyber آشنا می‌شویم.

۲.۱ آشنایی با NTT

NTT شباهت‌هایی با FFT دارد؛ پیچیدگی محاسباتی NTT برابر با $O(nlogn)$ است در حالی که پیچیدگی محاسباتی FFT برابر $O(n^2)$ است و از Nامین ریشه واحد ω استفاده می‌شود و N توانی از دو است. می‌توان گفت NTT روشی بهینه برای ضرب است.

$$A_i = NTT\,(a_i) = \int_{j=0}^{N-1} a_j\,\omega_N^{ij} \mod q \qquad i = 0, 1, 2, \ldots, N-1 \qquad (1)$$

برای به دست آوردن INTT کافی است، عبارت در $1/N$ ضرب شود و به جای ω، از $\omega-1$ استفاده شود.

$$a_j = INTT\,(A_j) = \frac{1}{N}\int_{i=0}^{N-1} A_i\,\omega_N^{-ij} \mod q \qquad j = 0, 1, 2, \ldots, N-1 \qquad (2)$$

میزان خطا از توزیع دوجمله‌ای متمرکز $B\eta$ (CBD) نمونه‌برداری می‌شود و f به‌عنوان خروجی طبق $B\eta$ نمونه‌برداری می‌شود. در Kyber نمونه‌برداری به ازای $\eta=2$ یا $\eta=3$ انجام می‌شود. در الگوریتم Kyber تعدادی تابع درهم‌سازی وجود دارد مانند SHAKE128, 256 و SHA3256, 512 که مبنای این توابع الگوریتم Keccak می‌باشد که وظیفه آن درهم‌سازی دیتا است و این باعث می‌شود که دیتا غیرقابل تشخیص باشد. در جدول ۱ مجموعه پارامترهای الگوریتم Kyber نشان داده شده‌است.

ساختار این مقاله به این صورت است که در بخش بعدی پیشینه پژوهش بررسی می‌شود؛ در بخش ۳ روش پیشنهادی بیان می‌شود و سپس در بخش ۴ نتایج حاصل از پیاده‌سازی روش پیشنهادی مورد مقایسه و ارزیابی قرار می‌گیرد. در نهایت بخش ۵ شامل جمع‌بندی و نتیجه‌گیری است.

۲ پیشینه پژوهش

تاکنون بهینه‌سازی‌های بسیاری برای Kyber ارائه شده است که در ادامه برخی از موارد که با NTT مرتبط هستند را ذکر می‌کنیم. در [۴] و [۵] برای کاهش منابع موردنیاز از واحد پروانه یک‌پارچه شده استفاده شده

جدول ۱: مجموعه پارامترها برای الگوریتم Kyber

Algorithm	Level	Parameters $(n/k/q)$	Public key/ Secret key/ Cipher-text size $p/s/c$ (in Bytes)
Kyber512	۱	۲۵۶/۲/۳۳۲۹	۸۰۰/۱۶۳۲/۷۳۶
Kyber768	۳	۲۵۶/۳/۳۳۲۹	۱۱۸۴/۲۴۰۰/۱۰۸۸
Kyber1024	۵	۲۵۶/۴/۳۳۲۹	۱۵۶۸/۳۱۶۸/۱۵۶۸

است که کار واحد پروانه CT و GS را انجام می‌دهد.

در [۶]، [۷] و [۵] با اعمال تغییراتی در modular reduction توانستند طراحی خود را بهبود ببخشند. برای مثال در [۶] از Modified Barrett Reduction استفاده شده است، در [۷] با استفاده از عملیات منطقی AND، OR و XOR و تغییر جایگاه بیت‌ها توانستند واحد modular reduction را بهبود دهند و در [۵] از Montgomery Reduction استفاده شده است. در [۴] و [۵] با افزایش تعداد واحد پروانه توانستند زمان اجرا را کاهش دهند. در [۵] از ویژگی قابلیت پیکربندی در زمان اجرا یا (RTC) استفاده شده است که موجب انعطاف‌پذیری طراحی شده است.

در [۷] و [۸] برای بهبود عملکرد، تغییراتی را در حافظه اعمال کردند. در [۸] پهنای باند حافظه را دو برابر کردند تا مشکل دسترسی به RAM حل شود و تغییر در الگوی دسترسی به حافظه باعث افزایش سرعت و بهره‌وری شده است و در [۷] برای اطمینان از ذخیره داده و افزایش سرعت از شانزده بلوک SRAM استفاده کردند.

در [۶] و [۹] برای پیاده‌سازی NTT از DIT و برای پیاده‌سازی INTT از DIF استفاده شده است. در [۸] NTTای را طراحی کرده‌اند که از مبنای چهار استفاده می‌کند و حافظه‌ی آن بدون برخورد[۲] است و از روش‌هایی استفاده کردند که تعداد حافظه موردنیاز آن کم بود.

در مقالاتی که اولویت کاهش منابع بوده زمان اجرای آنها نسبتاً طولانی بوده و در مقالاتی که اولویت کاهش زمان اجرا بوده و برای کاهش زمان اجرا تعداد زیادی واحد پروانه اضافه کرده‌اند، به منابع خیلی بیشتری احتیاج داشته‌اند. در روش پیشنهادی می‌خواهیم یک NTT مبنای چهار که برای Kyber مناسب‌سازی شده است را به گونه‌ای بهینه کنیم که زمان اجرای کمی داشته باشد و در عین حال به منابع خیلی زیادی احتیاج نداشته باشد.

۳ روش پیشنهادی

تاکنون در پیاده‌سازی‌هایی که برای الگوریتم‌های پساکوانتوم انجام شده‌است از NTT مبنای دو استفاده شده است، یعنی دو ضریب وارد شده، محاسبات آن انجام شده و سپس خارج می‌شوند. حال اگر بتوان در NTT از مبنای چهار استفاده کرد، عملکرد NTT به‌طور محسوسی بهبود می‌یابد، که برای تحقق آن لازم است

[2] Conflict

تغییراتی در NTT رخ دهد.

در این معماری، باید به طول دیتا و تعداد واحدهای پروانه توجه کنیم و همین‌طور باید بررسی شود که آیا در حافظه ممکن است با برخوردی در حافظه روبه‌رو شویم، چون باید مطمئن شد که دیتا به موقع فراخوانی و در زمان و جای مناسب ذخیره می‌شود.

در طراحی‌های گذشته چون براساس مبنای دو بودند واحد تولید آدرس[3]، دو آدرس را تولید می‌کرد و حافظه هم دو ضریب را می‌خواند یا می‌نوشت که برای مبنای چهار مناسب نبودند، پس در کد تغییراتی انجام شد، که این واحد با مبنای چهار سازگار باشند. با داشتن NTT مبنای چهار می‌توان انتظار داشت که عملکرد و زمان اجرای الگوریتم Kyber بهبود یابد چون حداقل، زمان اجرای NTT کاهش می‌یابد. پیش‌تر FFT مبنای چهار طراحی شده بود و این تغییر مبنا تأثیر زیادی بر عملکرد داشت [۱۰]، [۱۱]. FFT و NTT هم شباهت‌هایی نسبت به هم دارند، در [۸] نحوه‌ی محاسبات NTT مبنای چهار توضیح داده شده است. برای اجرای NTT از DIT و برای INTT از DIF استفاده می‌شود. در ادامه نحوه‌ی تبدیل DIT و DIF معمولی به DIT و DIF مبنای چهار توضیح داده می‌شود. فرمول NTT که با پیش‌پردازش[4] ترکیب شده‌است به‌صورت زیر است:

$$A_i = \int_{j=0}^{N-1} a_j \, \phi_{2N}^{j} \, \omega_N^{ij} \mod q \qquad i = 0, 1, 2, \ldots, N-1 \tag{۳}$$

عبارت بالا را به چهار قسمت برابر تبدیل می‌کنیم:

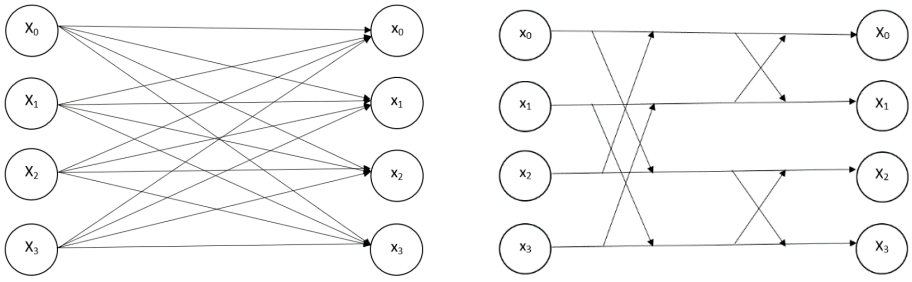
$$A_i = \sum_{j=0}^{\frac{N}{4}-1} a_{4j} \, \phi_{2N}^{4j} \, \omega_N^{i.(4j)} + \sum_{j=0}^{\frac{N}{4}-1} a_{4j+1} \, \phi_{2N}^{4j+1} \, \omega_N^{i.(4j+1)} + \tag{۴}$$

$$\sum_{j=0}^{\frac{N}{4}-1} a_{4j+2} \, \phi_{2N}^{4j+2} \, \omega_N^{i.(4j+2)} + \sum_{j=0}^{\frac{N}{4}-1} a_{4j+3} \, \phi_{2N}^{4j+3} \, \omega_N^{i.(4j+3)} \mod q \qquad i = 0, 1, 2, \ldots, N-1$$

در ادامه twiddle factor های ω و ϕ را ساده می‌کنیم. برای پیاده‌سازی واحد پروانه به‌طور معمول از مبنای ۲ استفاده می‌شود شکل ۱ (الف) نحوه محاسبات در مبنای ۲ را نشان می‌دهد. با استفاده از روش‌هایی می‌توان تعداد ضرب، تفریق و جمع را کاهش داد. شکل ۱ (ب) نمایی از واحد پروانه چهار نقطه‌ای در مبنای ۴ را نشان می‌دهد. همان‌طور که مشخص است محاسبات در مثال چهار نقطه‌ای، در مبنای ۴ در یک مرحله انجام می‌شود ولی در مبنای ۲، در دو مرحله انجام می‌شود که این موضوع نشان می‌دهد که استفاده از مبنای ۴ می‌تواند باعث کاهش زمان شود.

اگر به صورت مستقیم معادلات NTT مبنای ۴ را محاسبه کنیم، به پنج ضرب، شش تفریق و شش جمع نیاز داریم ولی می‌توان مقادیر را دوباره استفاده کرد که با استفاده از این روش به چهار ضرب، چهار تفریق و چهار جمع نیاز داریم.

[3] Address generator
[4] Preprocessing



(ب) (الف)

شکل ۱: نحوه محاسبات ماجول NTT: (الف) مبنای ۲ و (ب) مبنای ۴

در شکل ۲ از چهار PE[5] یا عنصر پردازشی استفاده شده‌است، هنگامی که sel_p یک باشد، دیتای مرتبط به A_i را فراخوانی می‌کند و محاسبات آن در ستون اول PEها انجام می‌شود و مقادیر میانی تولید می‌شود و در ادامه، محاسبات در PE های ستون دوم انجام می‌شود و در نهایت مقدار خروجی مشخص می‌شود. زمانی که sel_p صفر باشد، از دیتای جایگزین استفاده می‌شود.

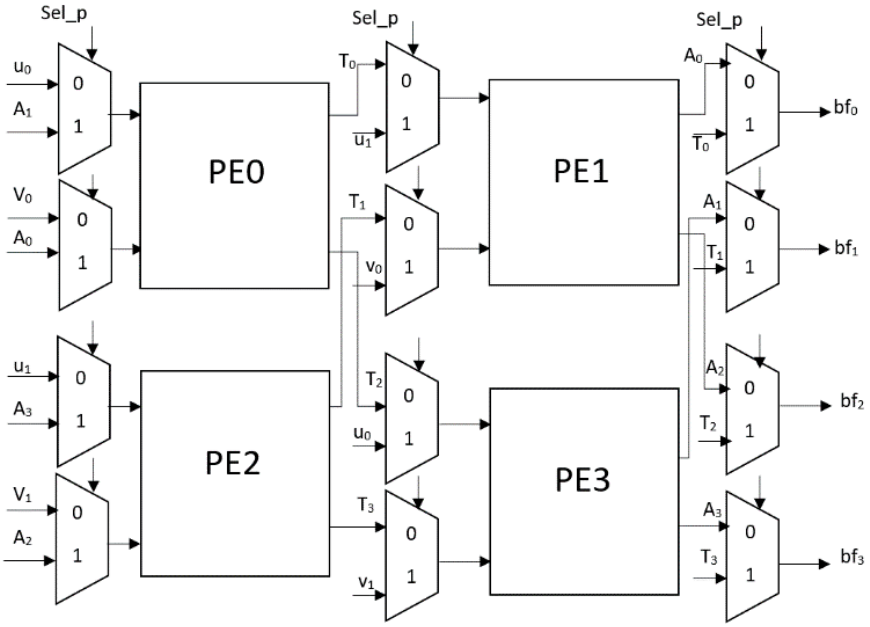
به‌منظور جلوگیری از هرگونه مشکل احتمالی و بالا بردن سرعت در خواندن و نوشتن از هشت حافظه RAM استفاده شده که چهارتا از آنها برای خواندن و چهارتای دیگر برای نوشتن به کار گرفته شده‌اند که این موضوع باعث تغییراتی در بخش آدرس‌دهی شده است. شکل ۳ ساختار حافظه را نشان می‌دهد. در پیاده‌سازی Kyber به‌صورت مستقیم نمی‌توان از NTT مبنای چهار استفاده کرد و NTT مبنای چهار نیاز به تغییراتی دارد تا به‌گونه‌ای شود که برای پیاده‌سازی الگوریتم Kyber مناسب باشد. به طور مثال مقدار n و q باید تغییر کنند، مقدار n برابر با ۲۵۶ باشد و مقدار q برابر با ۳۳۲۹ باشد.

۴ ارزیابی نتایج پیاده‌سازی

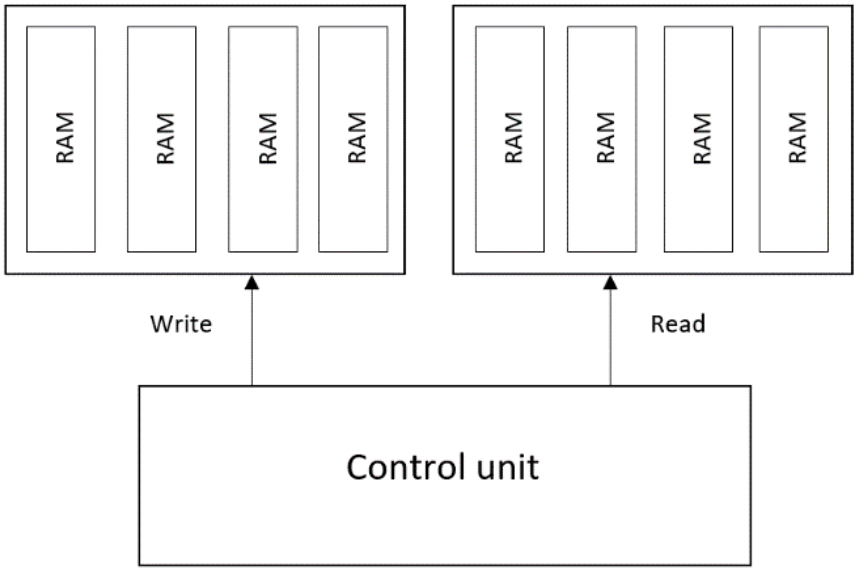
با استفاده از روش پیشنهادی، NTT مبنای چهار موردنظر را روی FPGAهای Artix-7 (xc7a200tffg11563) و Virtex-7 (xc7vx690tffg17612) توسط Vivado 2019.1 پیاده‌سازی شده است. نتایج به‌دست آمده طبق گزارشهای Vivado می‌باشد و کد روی FPGAها پروگرام نشده است.

واحد پروانه را به صورت جداگانه روی Artix-7 و Virtex7 پیاده‌سازی کرده‌ایم که طبق گزارش‌های به‌دست آمده روی Artix-7 به ۶۷۶ فلیپ فلاپ، ۱۱۳۲ LUT و ۴ DSP روی فرکانس ۱۸۵٫۱۸ مگاهرتز نیاز است و توان مصرفی آن ۰٫۲۷۱ وات می‌باشد و روی Virtex-7 به ۶۷۶ فلیپ فلاپ، ۱۱۵۶ LUT و ۴ DSP روی فرکانس ۲۳۴٫۷۴ مگاهرتز نیاز است و توان مصرفی آن ۰٫۵۰۲ وات می‌باشد. در شکل ۴ زمان اجرای روش پیشنهادی با سایر پیاده‌سازی‌ها روی Artix-7 مشابه، مقایسه شده‌است و همان‌طور که دیده می‌شود پیاده‌سازی روش پیشنهادی ما توانسته زمان خوبی را ثبت کند و زمان اجرای ۱٫۱۳۴ میکرو ثانیه

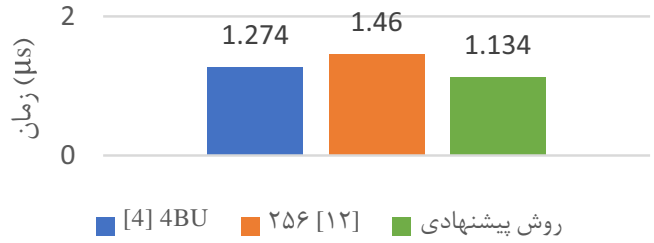
[5]Processing element



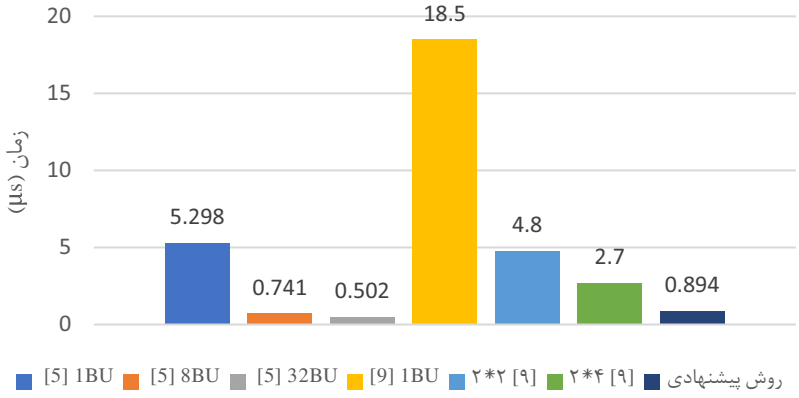
شکل ۲: ساختار مسیردهی در مبنای چهار



شکل ۳: ساختار حافظه در روش پیشنهادی



شکل ۴: مقایسه پیاده‌سازی‌ها روی Artix-7



شکل ۵: مقایسه پیاده‌سازی‌ها روی Vrtex-7

است (منظور از BU واحد پروانه می‌باشد).

زمان اجرای روش پیشنهادی روی Artix7 نسبت به ۲۵۶ [۱۲] ۲۸٫۷۴ درصد سریع‌تر و نسبت به [۴] 4 BU ۱۲٫۳۴ درصد سریع‌تر است. در شکل ۵ زمان اجرای روش پیشنهادی با سایر پیاده‌سازی‌ها روی Virtex-7 مقایسه شده‌است که توانسته سریع‌تر از اکثر پیاده‌سازی‌ها باشد. با توجه به اینکه پیاده‌سازی مشابهی برای Virtex-7 وجود نداشت، بناچار روش پیشنهادی با همه پیاده‌سازی‌های روی Virtex-7 مقایسه شده‌است.

زمان اجرای روش پیشنهادی روی Virtex-7 نسبت به [۹] 2*4 ۳٫۰۲ برابر سریع‌تر، نسبت به [۹] 2*2 ۵٫۳۶ برابر سریع‌تر، نسبت به [۹] 1 BU ۲۰٫۶۹ برابر سریع‌تر، نسبت به [۵] 32 BU ۱٫۷۸ برابر کندتر، نسبت به [۵] 8 BU ۱٫۲ برابر کندتر و نسبت به [۵] 1 BU ۵٫۹۲ برابر سریع‌تر است.

پیاده‌سازی های گوناگونی برای NTT وجود دارد برخی از آنها از طراحی همزمان سخت‌افزار/ نرم‌افزار استفاده کرده‌اند و برخی از طراحی تمام سخت‌افزاری یا برخی از پیاده‌سازی ها از تعداد واحد پروانه بیشتری

جدول ۲: مقایسه نتایج پیاده سازی روی Artix-7

Design	n	BU	LUTs	FFs	DSPs	BRAMs	Freq. (MHz)	NTT (CCs)	INTT (CCs)	Time (μs)
[۴]	۲۵۶	۴	۲۵۴۳	۷۹۲	۴	۹	۱۸۲	۲۳۲	۲۳۳	۲۷۴.۱
[۱۲]	۲۵۶	۲*۲	۸۰۱	۷۱۷	۴	۲	۲۲۲	۳۲۴	۳۲۴	۴۶.۱
روش پیشنهادی	۲۵۶	۲*۲	۲۳۰۱	۹۲۵	۴	۹	۱۸.۱۸۵	۲۱۰	۲۱۰	۱۳۴.۱

استفاده کرده‌اند یا در مواردی مقدار n متفاوت بوده است (معمولاً مقدار $n = ۲۵۶$ است) که همه‌ی موارد بالا روی سرعت و منابع موردنیاز اثرگذار می‌باشد ولی در جدول ۲ پیاده‌سازی‌های مشابه با روش پیشنهادی روی Artix-7 مقایسه شده‌اند. با مقایسه نتایج، می‌توان متوجه شد که روش پیشنهادی توانسته بهترین زمان اجرا را ثبت کند ولی در مقایسه با [۱۲]، روش پیشنهادی به منابع بیشتری نیاز دارد به‌طوری که ۲٫۸۷ برابر LUT بیشتر، ۱٫۲۹ برابر فلیپ فلاپ بیشتر، ۴٫۵ برابر BRAM بیشتر نیاز دارد و روش پیشنهادی در مقایسه با [۴] به LUT کمتر و فلیپ فلاپ بیشتر نیاز دارد.

۵ نتیجه‌گیری

به علت قدرت بالای محاسباتی کامپیوترهای کوانتومی، در آینده این کامپیوترها می‌توانند الگوریتم‌های رمزنگاری کلید عمومی فعلی را بشکنند؛ برای مقابله با این موضوع الگوریتم‌های رمزنگاری پساکوانتوم به وجود آمدند. پیاده‌سازی الگوریتم‌های رمزنگاری پساکوانتوم با چالش‌های مختلفی، همچون منابع مورد نیاز و زمان اجرا روبه‌رو بودند و هستند، برای حل این چالش‌ها روش‌های مختلفی پیشنهاد شده‌است. یکی از پیچیده‌ترین و پرهزینه‌ترین ماجول‌های محاسباتی الگوریتم Kyber ماجول NTT است و در روش پیشنهادی از مبنای محاسبه NTT از ۲ به مبنای ۴ تغییر کرده که باعث تغییرات در سایر قسمت های NTT شده است و توانسته زمان اجرا کوتاهی در مقایسه با سایر پیاده‌سازی‌ها داشته‌باشد درحالیکه به منابع موردنیاز متعادلی نیاز دارد. روش پیشنهادی با تغییر مبنا در NTT توانست زمان اجرا را بهبود دهد و در مقایسه با پیاده‌سازی‌های مشابه توانست ۲۸٫۷۴ درصد زمان اجرای کوتاه‌تری را ثبت کند.

مراجع

[1] F. Farahmand, D. T. Nguyen, V. B. Dang, A. Ferozpuri and K. Gaj, “Software/Hardware Codesign of the Post Quantum Cryptography Algorithm NTRUEncrypt Using High-Level Synthesis and Register-Transfer Level Design Methodologies,” *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 225-231, doi: 10.1109/FPL.2019.00042.

[2] J. Xie, K. Basu, K. Gaj and U. Guin, “Special Session: The Recent Advance in Hardware Implementation of Post-Quantum Cryptography,” *2020 IEEE 38th VLSI Test Symposium (VTS)*, 2020, pp. 1-10, doi: 10.1109/VTS48691.2020.9107585.

[3] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation," 2017, NIST PQC Round 2, vol. 9, pp. 11.

[4] F. Yaman, A. C. Mert, E. Öztürk and E. Savaş, "A Hardware Accelerator for Polynomial Multiplication Operation of CRYSTALS-KYBER PQC Scheme," *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2021, pp. 1020-1025, doi: 10.23919/DATE51398.2021.9474139.

[5] K. Derya, A.C. Mert, E. Öztürk and E. Savaş, "CoHA-NTT: A Configurable Hardware Accelerator for NTT-based Polynomial Multiplication," Microprocessors and Microsystems, 2022, vol. 89, p.104451.

[6] Y. Xing and S. Li, "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA," *TCHES*, vol. 2021, no. 2, pp. 328–356, Feb. 2021.

[7] W. Guo, S. Li and L. Kong, "An Efficient Implementation of KYBER," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 3, pp. 1562-1566, March 2022, doi: 10.1109/TCSII.2021.3103184.

[8] C. Zhang *et al*., "Towards Efficient Hardware Implementation of NTT for Kyber on FPGAs," *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1-5, doi: 10.1109/ISCAS51556.2021.9401170.

[9] X. Chen, B. Yang, S. Yin, S. Wei, and L. Liu, "CFNTT: Scalable Radix-2/4 NTT Multiplication Architecture with an Efficient Conflict-free Memory Mapping Scheme", *TCHES*, vol. 2022, no. 1, pp. 94–126, Nov. 2021.

[10] M. Garrido, J. Grajal, M. A. Sanchez and O. Gustafsson, "Pipelined Radix-2k Feedforward FFT Architectures," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 23-32, Jan. 2013.

[11] E. E. Swartzlander, W. K. W. Young and S. J. Joseph, "A radix 4 delay commutator for fast Fourier transform processor implementation," in *IEEE Journal of Solid-State Circuits*, vol. 19, no. 5, pp. 702-709, Oct. 1984, doi: 10.1109/JSSC.1984.1052211.

[12] M. Bisheh-Niasar, R. Azarderakhsh and M. Mozaffari-Kermani, "High-Speed NTT-based Polynomial Multiplication Accelerator for Post-Quantum Cryptography," *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, 2021, pp. 94-101.