Scientific Journal of Electronical & Cyber Defence
Vol. 10, No. 4, Winter 2023, Serial No. 40

Speeding up the execution-time of Crystals-Kyber PQC Algorithm on FPGA

M. Ghafari, Hatam Abdoli*, M. Abbasi
Assistant Professor, Bu - Ali Sina University, Hamadan, Iran*

(Received: 11/09/2022, Accepted: 09/10/2022)

ABSTRACT

Quantum computers have much more computing power than classical computers and this has created a challenge in the field of public-key cryptography algorithms, which is predicted quantum computers will reach the computational power to break existing public-key cryptography algorithms by 2030. To solve this problem, NIST published a call for post-quantum cryptography algorithms. Implementing these algorithms faces challenges such as execution time and resources. One of the algorithms that made it to the third round is the CRYSTALS-KYBER algorithm. In this algorithm, by optimizing the NTT module, the execution time is reduced. Usually, the implementation of NTT is created with radix-2, but in the proposed method, radix-4 is used, and this reduces the execution time. Changes to NTT are required to implement radix-4 NTT. DIT is used to implement NTT and DIF is used to implement INTT. In NTT and INTT formulas changes are made to the twiddle factors and the values of the twiddle factors stored to the ROM. In the following, we compared radix-4 butterfly unit with radix-2 butterfly unit. By reusing results in CT and GS butterfly units, we need four multiplications, additions, and subtractions, and the structure of radix-4 butterfly unit is mentioned. The memory unit uses eight RAMs to increase read and write speeds, four of which are for writing and the remaining four are for reading. It is also necessary to make corrections to the NTT parameters which are suitable for implementation on Kyber. Next, we implemented the proposed method on two FPGA Artix-7 and Virtex-7 using Vivado software. In the implementation on Artix-7 and Virtex-7 in exchange for a slight increase in the resources, the execution time in Artix-7 is reduced by 28.74% and 12.34% compared to similar implementations.

Keywords: Post-Quantum Cryptography, Crystals-Kyber, NTT, Radix-4, Polynomial Multiplication, Hardware Implementation.

Corresponding Author Email: Abdoli@basu.ac.ir

نشریه علمی "پدافند الکترونیکی و سایبری"
سال دهم، شماره ۴، زمستان ۱۴۰۱، ص ۱۱۰- ۱۰۱

علمی – پژوهشی

تسریع زمان اجرای الگوریتم رمزنگاری پساکوانتوم Crystals-Kyber روی FPGA

محمد غفاری[۱]، حاتم عبدلی[۲*]، مهدی عباسی[۲]

۱- دانشجوی کارشناسی ارشد، ۲- استادیار، ۳- دانشیار، دانشگاه بوعلی سینا، همدان، ایران

(دریافت: ۱۴۰۱/۰۶/۲۰، پذیرش: ۱۴۰۱/۰۷/۱۷)

چکیده

کامپیوترهای کوانتومی توان محاسباتی بسیار بیشتری نسبت به کامپیوترهای کلاسیک دارند و این مسئله باعث ایجاد چالش در حوزه رمزنگاری کلید عمومی شده است، به‌طوری‌که پیش‌بینی می‌شود در آینده کامپیوترهای کوانتومی به‌اندازه‌ای قدرتمند شوند که بتوانند الگوریتم‌های رمزنگاری کلید عمومی را بشکنند. به‌منظور حل این مشکل NIST یک فراخوانی را برای رمزنگاری پساکوانتوم منتشر کرد. یکی از الگوریتم‌های راه‌یافته به دور سوم، الگوریتم CRYSTALS-KYBER است. در این الگوریتم با بهینه‌سازی واحد NTT می‌توان زمان اجرا را کاهش داد. در حالت عادی پیاده‌سازی NTT، با پایه دو صورت گرفته ولی در روش پیشنهادی از پایه چهار استفاده شده است و این امر باعث کاهش زمان اجرا شده است. برای پیاده‌سازی NTT با پایه چهار و متناسب با الگوریتم Kyber، لازم است تغییراتی در NTT رخ دهد. در ادامه واحد پروانه پایه دو با واحد پروانه پایه چهار مقایسه شده است. در واحد حافظه به‌منظور افزایش سرعت خواندن و نوشتن از هشت RAM استفاده شده که چهار عدد از آن‌ها برای نوشتن و چهار عدد باقیمانده برای خواندن هم‌زمان است. در بخش تولید آدرس، پیش‌تر آدرس‌ها به‌صورت دوتایی تولید می‌شد ولی در روش پیشنهادی به‌صورت چهارتایی تولید می‌شود و همچنین لازم است در پارامترهای NTT اصلاحاتی انجام شود که برای پیاده‌سازی روی Kyber مناسب باشد. در ادامه، روش پیشنهادی روی دو تراشه FPGA، Artix-7 و Virtex-7 با استفاده از نرم‌افزار Vivado پیاده‌سازی شده است که در ازای افزایش جزئی منابع موردنیاز، زمان اجرا در Artix-7 در مقایسه با پیاده‌سازی‌های مشابه ۲۸/۷۴ درصد و ۱۲/۳۴ درصد کاهش یافته است.

کلیدواژه‌ها: رمزنگاری پساکوانتوم، Crystals-Kyber، NTT، پایه چهار، ضرب چندجمله‌ای، پیاده‌سازی سخت‌افزاری

۱- مقدمه

کامپیوترهای کوانتومی، کامپیوترهایی هستند که برای انجام محاسبات از خاصیت‌های فیزیک کوانتوم استفاده می‌کنند که این ویژگی باعث شده عملکرد خارق‌العاده‌ای نسبت به کامپیوترهای کلاسیک که در حال حاضر استفاده می‌شوند، داشته باشند و این عملکرد شگفت‌انگیز باعث جلب‌توجه بسیاری شده است. در کامپیوترهای کلاسیک از بیت استفاده می‌شود درحالی‌که 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 شده است که حدوداً تا سال ۲۰۳۰ کامپیوترهای کوانتومی به‌قدری قدرتمند می‌شوند که می‌توانند الگوریتم‌های رمزنگاری کلید عمومی را بشکنند [۱ و 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 CRYSTALS-KYBER به‌عنوان فینالیست پذیرفته شده‌اند و پنج الگوریتم Bike، FrodoKEM، HQC، NTRUprime و SIKE به‌عنوان الگوریتم‌های جایگزین در نظر گرفته شده‌اند. سه الگوریتم Saber، NTRU و CRYSTALS-KYBER مبتنی بر lattice و الگوریتم Classic McEliece مبتنی بر کد هستند؛ همچنین سه الگوریتم امضای دیجیتال CRYSTALS-DILITHIUM، Rainbow و Falcon به‌عنوان فینالیست پذیرفته شده‌اند و سه الگوریتم امضای دیجیتال GeMSS، Picnic و SPHINCS+ به‌عنوان الگوریتم امضای دیجیتال

* رایانامه نویسنده مسئول: Hatamabdoli@gmail.com

[1] Post Quantum Cryptography

ناشر: دانشگاه جامع امام حسین (ع)

جـایگزین انتخـاب شـده‌انـد. دو الگـوریتم امضـای دیجیتـال CRYSTALS-DILITHIUM و Falcon مبتنی بر lattice هستند و الگوریتم امضای دیجیتال Rainbow مبتنی بر چندمتغیره است.

در پیـاده‌سـازی سـخت‌افـزاری الگـوریتم‌هـای رمزنگـاری پساکوانتوم به دو موضوع توجه شـده اسـت: ۱- زمـان اجـرا و ۲- منابع موردنیاز. به‌طورکلی برای بهبود پیاده‌سـازی سـخت‌افـزاری الگوریتم‌های رمزنگاری پساکوانتوم از دو رویکـرد اصـلی اسـتفاده می‌شود:

- بهبود کارایی بخش‌های پیچیده الگوریتم مانند بهینـه‌سـازی بخش ضرب چندجملـه‌ای بـا اسـتفاده از روش‌هـای حسـاب کامپیوتری
- استفاده از تکنیک‌های مختلـف طراحـی در سـطح معمـاری کامپیوتر (مانند pipelining) با هدف افزایش کارایی سـخت-افزار الگوریتم

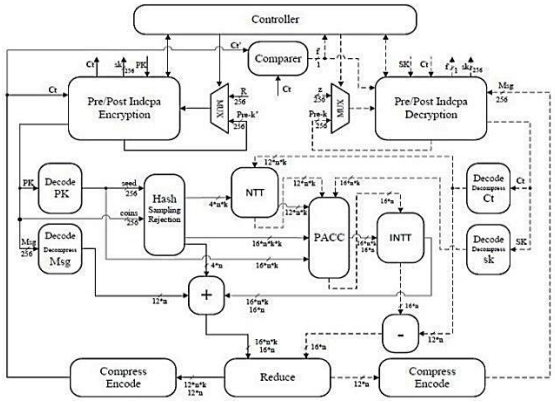
در کارهایی که تاکنون انجام شده است در بعضـی از مقـالات به چگونگی کـاهش زمـان اجـرا توجـه کـرده‌انـد، عـده‌ای دیگـر تمرکزشان بر کاهش منابع است و برخی به هردو رویکـرد توجـه داشته‌اند. در این مقاله با اعمال تغییراتی در واحد NTT، به‌عنوان یکی از بخش‌های پیچیده الگوریتم Kyber، بهبـود زمـان اجـرای الگوریتم Kyber حاصل شده است.

۱-۱- آشنایی با Kyber

Kyber یک سازوکار کپسوله‌سازی کلید (KEM) است که امنیـت آن بر اساس سختی حل کردن مسئله یادگیری با خطا در lattice است [۳]. ساختار Kyber از دو نوع پیروی می‌کند، یکـی CPA و دیگری CCA است. CPA نوعی حمله است که در آن فرد مهاجم هم به متن رمز شده و هم به متن آشکار دسترسـی دارد و سـعی می‌کند که متوجه شود که متن چگونه رمـز شـده اسـت و CCA نوعی حمله است که فرد مهاجم می‌تواند با متن رمز شده انتخابی به متن آشکار دست پیدا کند.

برای نمایش حلقه چندجمله‌ای Z[X]/ (Xn+1) از R و بـرای نمـایش حلقـه Zq[X]/ (Xn+1) از Rq اسـتفاده مـی‌شـود و به‌طوری‌که n=256، n'=9 و q=3329 است. از حروف بزرگ بـرای نشـان دادن مـاتریس‌هـا و حـروف کوچـک بـرای نشـان دادن بردارهایی با ضرایبی در R و Rq استفاده می‌شود.

شکل (۱) طرحی از الگوریتم Kyber را نشـان مـی‌دهـد کـه هردو عمل رمزگذاری و رمزگشـایی را مـی‌توانـد انجـام دهـد. در شکل (۱) قسمت‌هایی که با خط‌چین نشان داده شده‌اند، مربـوط به رمزگشایی و قسمت‌هایی که با خط ساده نشان داده شـده‌انـد، مربوط به رمزگذاری است. ماژول NTT در مرکز شـکل مشـخص شده است. در ادامه با اجزای Kyber آشنا می‌شویم.



شکل (۱). معماری کلی الگوریتم Kyber [۱]

۱-۲- آشنایی با NTT

NTT شـباهت‌هـایی بـا FFT دارد و باعـث مـی‌شـود پیچیـدگی محاسباتی ضرب چندجملـه‌ای از $O(n^2)$ بـه $O(n \log n)$ کـاهش یابد. در NTT از Nامین ریشه واحد ω استفاده می‌شود و N توانی از دو است. می‌توان گفت NTT روشی بهینه برای ضرب است.

$$T(a_i) = \sum a \qquad (۱)$$

برای به دست آوردن INTT کافی است، عبارت در N/۱ ضرب شود و به‌جای ω، از ω-1 استفاده شود [۲].

$$T(A_j) = -\sum \qquad (۲)$$

مـاژول NTT مهـم‌تـرین و پیچیـده‌تـرین بخـش محاسـباتی الگوریتم Kyber است که ضرب‌هـای چندجملـه‌ای و پیچیـده در این ماژول انجام می‌شود. بخش عمده پردازش‌ها و زمـان اجـرای الگـوریتم Kyber مربـوط بـه NTT اسـت [۱]. بـه همـین دلیـل بهینه‌سـازی و بهبـود کـارایی NTT بـرای تسـریع زمـان اجـرای الگوریتم Kyber در بیشتر پژوهش‌های پیشـین و همچنـین ایـن پژوهش در نظر گرفته شده است.

نمونه‌برداری از توزیع دوجمله‌ای

میـزان خطـا از توزیـع دوجملـه‌ای متمرکـز Bη (CBD) نمونه‌برداری می‌شود و f به‌عنوان خروجی طبق Bη نمونه‌بـرداری مـی‌شـود. در Kyber نمونـه‌بـرداری بـه ازای η=2 یـا η=3 انجـام می‌شود.

تابع درهم‌سازی

در الگوریتم Kyber تعدادی تابع درهم‌سازی وجود دارد مانند SHAKE-128, 256 و SHA3-256,512 کـه مبنـای ایـن توابـع

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 شده است.

جدول (۱). مجموعه پارامترها برای الگوریتم Kyber

Algorithm	Level	Parameters (n/k/q)	Public key/Secret key/Ciphertext size p/s/c (in Bytes)
Kyber۵۱۲	۱	۲۵۶/۲/۳۳۲۹	۸۰۰/۱۶۳۲/۷۳۶
Kyber۷۶۸	۳	۲۵۶/۳/۳۳۲۹	۱۱۸۴/۲۴۰۰/۱۰۸۸
Kyber۱۰۲۴	۵	۲۵۶/۴/۳۳۲۹	۱۵۶۸/۳۱۶۸/۱۵۶۸

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 مراجع [۴ و ۵] برای کاهش منابع موردنیاز از واحد پروانه یکپارچه‌شده[1] استفاده شده است که کار واحد پروانه CT و GS را انجام می‌دهد.

در مراجع [۵-۷] با اعمال تغییراتی در modular reduction توانستند طراحی خود را بهبود ببخشند. برای مثال در مرجع [۶] از Modified Barrett Reduction استفاده شده است، در مرجع [۷] با استفاده از عملیات منطقی AND، OR و XOR و تغییر جایگاه بیت‌ها توانستند واحد modular reduction را بهبود دهند و در مرجع [۵] از Montgomery Reduction استفاده شده است. در مراجع [۴ و ۵] با افزایش تعداد واحد پروانه توانستند زمان اجرا را کاهش دهند. در مرجع [۵] از ویژگی قابلیت پیکربندی در زمان اجرا[2] یا (RTC) استفاده شده است که موجب انعطاف‌پذیری طراحی شده است.

در مراجع [۷ و ۸] برای بهبود عملکرد، تغییراتی را در حافظه اعمال کردند. در مرجع [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 مرجع [۷] برای اطمینان از ذخیره داده و افزایش سرعت از شانزده بلوک SRAM استفاده کردند.

در مراجع [۶ و ۹] برای پیاده‌سازی NTT از DIT[3] و برای پیاده‌سازی INTT از DIF[4] استفاده شده است. در مرجع [۸] NTTای را طراحی کرده‌اند که از پایه چهار[5] استفاده می‌کند و حافظه آن بدون برخورد[6] 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 پایه چهار که برای Kyber مناسب‌سازی شده است را به‌گونه‌ای بهینه کنیم که زمان اجرای کمی داشته باشد و درعین‌حال به منابع خیلی زیادی احتیاج نداشته باشد.

۳- روش پیشنهادی

تاکنون در پیاده‌سازی‌هایی که برای الگوریتم‌های پساکوانتوم انجام شده است از NTT پایه دو استفاده شده است؛ یعنی دو ضریب واردشده، محاسبات آن انجام شده و سپس خارج می‌شوند. حال اگر بتوان در NTT از پایه چهار استفاده کرد، عملکرد NTT به‌طور محسوسی بهبود می‌یابد که برای تحقق این امر لازم است تغییرات زیادی 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[7] در حافظه روبه‌رو شویم، چون باید مطمئن شد که دیتا به‌موقع فراخوانی و در زمان و جای مناسب ذخیره می‌شود.

در طراحی‌های گذشته چون بر اساس پایه دو بودند واحد تولید آدرس[8]، دو آدرس را تولید می‌کرد و حافظه هم دو ضریب را می‌خواند یا می‌نوشت که برای پایه چهار مناسب نبودند، پس در کد تغییراتی انجام شد که این واحد با پایه چهار سازگار باشند.

با داشتن NTT پایه چهار می‌توان انتظار داشت که عملکرد و زمان اجرای الگوریتم Kyber بهبود یابد چون حداقل، زمان اجرای NTT کاهش می‌یابد.

۳-۱- تغییر در NTT

پیش‌تر FFT پایه چهار طراحی شده بود و این تغییر مبنا تأثیر زیادی بر عملکرد داشت [۱۰-۱۲]. FFT و NTT هم شباهت‌هایی نسبت به هم دارند، در مرجع [۸] نحوه‌ی

[1] Unified butterfly unit
[2] Run-time configurability
[3] Decimation in time
[4] Decimation in frequency
[5] Radix-4
[6] Conflict-free
[7] Conflict
[8] Address generator

محاسبات NTT پایه چهار توضیح داده شده است. با توجه به اینکه بیشترین زمان اجرای الگوریتم‌های پساکوانتوم به بخش ضرب‌های چندجمله‌ای اختصاص دارد، کاهش زمان اجرای ماجول NTT تأثیر بسزایی در کاهش زمان کل اجرای الگوریتم مربوطه خواهد داشت [۱۲ و ۱۳].

برای اجرای NTT از DIT و برای INTT از DIF استفاده می‌شود. در ادامه نحوه‌ی تبدیل DIT و DIF معمولی به DIT و DIF پایه چهار توضیح داده می‌شود.

DIT پایه چهار

فرمول NTT که با پیش‌پردازش[1] ترکیب شده است به‌صورت زیر است:

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

عبارت بالا را به چهار قسمت برابر تبدیل می‌کنیم:

$$\begin{aligned} A_i &= \sum_{j=0}^{\frac{N}{4-1}} a_{4j}\, \phi_{2N}^{4j}\, \omega_N^{i.(4j)} + \sum_{j=0}^{\frac{N}{4-1}} a_{4j+1}\, \phi_{2N}^{4j+1}\, \omega_N^{i.(4j+1)} \\ &+ \sum_{j=0}^{\frac{N}{4-1}} a_{4j+2}\, \phi_{2N}^{4j+2}\, \omega_N^{i.(4j+2)} \\ &+ \sum_{j=0}^{\frac{N}{4-1}} a_{4j+3}\, \phi_{2N}^{4j+3}\, \omega_N^{i.(4j+3)} \ \ mod\ q \qquad i = 0,1,2,\dots,N-1 \end{aligned} \tag{۴}$$

در ادامه twiddle factor های ω و ϕ را ساده می‌کنیم.

$$\begin{aligned} A_i &= \sum_{j=0}^{\frac{N}{4-1}} a_{4j}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij} \\ &+ \omega_N^{i}.\phi_{2N}^{1}.\sum_{j=0}^{\frac{N}{4-1}} a_{4j+1}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij} \\ &+ \ \omega_N^{2i}.\phi_{2N}^{2}.\sum_{j=0}^{\frac{N}{4-1}} a_{4j+2}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij} \\ &+ \omega_N^{3i}.\phi_{2N}^{3}.\sum_{j=0}^{\frac{N}{4-1}} a_{4j+3}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij} \ \ mod\ q \end{aligned} \tag{۵}$$

برای ساده‌تر شدن $\sum_{j=0}^{\frac{N}{4-1}} a_{4j+1}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij}$، $\sum_{j=0}^{\frac{N}{4-1}} a_{4j}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij}$، $\sum_{j=0}^{\frac{N}{4-1}} a_{4j+2}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij}$ و $\sum_{j=0}^{\frac{N}{4-1}} a_{4j+3}\, \phi_{\frac{N}{2}}^{j}\, \omega_{\frac{N}{4}}^{ij}$ را به ترتیب F_0، F_1، F_2 و F_3 نامیده و معادله (۶) به شکل زیر نوشته می‌شود:

$$A_i = F_0 + \omega_N^{i}.\phi_{2N}^{1}.F_1 + \omega_N^{2i}.\phi_{2N}^{2}.F_2 + \omega_N^{3i}.\phi_{2N}^{3}.F_3 \tag{۶}$$

و طبق خاصیت تناوب[2] twiddle factor سه معادله زیر به دست می‌آید:

$$\begin{aligned} A_{i+\frac{N}{4}} &= F_0 + \omega_N^{i}.\omega_4^{1}.\phi_{2N}^{1}.F_1 + \omega_N^{2i}.\omega_4^{2}.\phi_{2N}^{2}.F_2 \\ &\quad + \omega_N^{3i}.\omega_4^{3}.\phi_{2N}^{3}.F_3 \\ A_{i+2N/4} &= F_0 + \omega_N^{i}.\omega_4^{2}.\phi_{2N}^{1}.F_1 \\ &\quad + \omega_N^{2i}.(\omega_4^{2})^{2}.\phi_{2N}^{2}.F_2 \\ &\quad + \omega_N^{3i}.(\omega_4^{3})^{2}.\phi_{2N}^{3}.F_3 \\ A_{i+3N/4} &= F_0 + \omega_N^{i}.\omega_4^{3}.\phi_{2N}^{1}.F_1 \\ &\quad + \omega_N^{2i}.(\omega_4^{2})^{3}.\phi_{2N}^{2}.F_2 \\ &\quad + \omega_N^{3i}.(\omega_4^{3})^{3}.\phi_{2N}^{3}.F_3 \end{aligned} \tag{۷}$$

$$i = 0,1,2,\dots,\frac{N}{4-1}$$

همچنین می‌توان نتیجه گرفت که $\omega_4^3 = -\omega_4^1$ و $\omega_4^2 = -\omega_4^0 = -1$ که می‌توان معادلات بالا را به شکل زیر نشان داد:

$$\begin{bmatrix} A_i \\ A_{i+N/4} \\ A_{i+2N/4} \\ A_{i+3N/4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^1 & -1 & -\omega_4^1 \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^1 & -1 & \omega_4^1 \end{bmatrix} \times \left(\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \omega_N^{i}.\phi_{2N}^{1} \\ \omega_N^{2i}.\phi_{2N}^{2} \\ \omega_N^{3i}.\phi_{2N}^{3} \end{bmatrix} \right) \tag{۸}$$

با ادغام n امین ریشه واحد ω_N و دو n امین ریشه واحد ϕ_{2N} معادله زیر به دست می‌آید:

$$\begin{bmatrix} A_i \\ A_{i+N/4} \\ A_{i+2N/4} \\ A_{i+3N/4} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^1 & -1 & -\omega_4^1 \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^1 & -1 & \omega_4^1 \end{bmatrix} \times \left(\begin{bmatrix} F_0 \\ F_1 \\ F_2 \\ F_3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \phi_{2N}^{2i+1} \\ \phi_{2N}^{2(2i+1)} \\ \phi_{2N}^{3(2i+1)} \end{bmatrix} \right) \tag{۹}$$

$$i = 0,1,2,\dots,\frac{N}{4-1}$$

DIF پایه چهار

فرمول INTT که با پس پردازش[3] ترکیب شده است به‌صورت زیر است:

$$a_i = N^{-1}.\ \phi_{2N}^{-i}.\sum_{j=0}^{N-1} A_j\, \omega_N^{-ij} \ \ mod\ q \qquad i = .0,1,2,\dots,N-1 \tag{۱۰}$$

عبارت بالا را به چهار قسمت برابر تبدیل می‌کنیم:

$$\begin{aligned} a_i = N^{-1}.\ (&\phi_{2N}^{-i}.\sum_{j=0}^{N/4-1} A_j\, \phi_{2N}^{-j}\, \omega_N^{-ij} \\ &+ \ \phi_{2N}^{-i}.\sum_{j=N/4}^{2N/4-1} A_j\, \omega_N^{-ij} \\ &+ \ \phi_{2N}^{-i}.\sum_{j=2N/4}^{3N/4-1} A_j\, \omega_N^{-ij} \\ &+ \ \phi_{2N}^{-i}.\sum_{j=3N/4}^{N-1} A_j\, \omega_N^{-ij})\ mod\ q \end{aligned} \tag{۱۱}$$

[1] Pre-processing

[2] Periodicity

[3] Post-processing

طبق خاصیت‌های twiddle factor می‌توان جمع‌های بالا را در بازه‌ی [0, N/4-1] نشان داد.

$$\begin{aligned} a_i &= N^{-1}.\ (\phi_{2N}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_j\,\omega_N^{-ij} + \ \phi_{2N}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{N}{4}}\,\omega_N^{-i\left(j+\frac{N}{4}\right)} \\ &+ \ \phi_{2N}^{-i}\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{2N}{4}}\,\omega_N^{-i\left(j+\frac{2N}{4}\right)} \\ &+ \ \phi_{2N}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{3N}{4}}\,\omega_N^{-i\left(j+\frac{3N}{4}\right)})\ mod\ q \end{aligned} \quad (۱۲)$$

معادله‌ی بالا را می‌توان ساده کرد که به‌صورت زیر می‌شود:

$$\begin{aligned} a_i &= N^{-1}.\ (\phi_{2N}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_j\,\omega_N^{-ij} + \ \phi_{2N}^{-i}.\omega_4^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{N}{4}}\,\omega_N^{-ij} \\ &+ \ \phi_{2N}^{-i}.\omega_4^{-2i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{2N}{4}}\,\omega_N^{-ij} \\ &+ \ \phi_{2N}^{-i}.\omega_4^{-3i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+\frac{3N}{4}+}\,\omega_N^{-ij})\ mod\ q \end{aligned} \quad (۱۳)$$

طبق خاصیت‌های twiddle factor می‌توان ساده‌سازی را انجام داد:

$$\begin{aligned} a_{4i} &= (\frac{N}{4})^{-1}.\left(\frac{1}{4}.\phi_{\frac{N}{2}}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_j\,\omega_{\frac{N}{4}}^{-ij}\right. \\ &+ \frac{1}{4}.\phi_{\frac{N}{2}}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_{j+N/4}\,\omega_{\frac{N}{4}}^{-ij} + \frac{1}{4}.\phi_{\frac{N}{2}}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_j\,\omega_{\frac{2N}{4}}^{-ij} \\ &\left.+ \frac{1}{4}.\phi_{\frac{N}{2}}^{-i}.\sum_{j=0}^{\frac{N}{4-1}} A_j\,\omega_{\frac{3N}{4}}^{-ij}\right)\ mod\ q \\ i &= 0,1,2,\dots,N/4-1 \end{aligned} \quad (۱۴)$$

برای ساده‌تر شدن، چهار جمع بالا را به ترتیب، با G_0، G_1، G_2 و G_3 نشان می‌دهیم.

$$\begin{aligned} a_{4i} &= G_0 + G_1 + G_2 + G_3 \\ a_{4i+1} &= \phi_{2N}^{-1}.\omega_N^{-j}.(G_0 + \omega_4^{-1}\,G_1 + \omega_4^{-2}\,G_2 \\ &\quad + \omega_4^{-3}\,G_3 \\ a_{4i+2} &= \phi_{2N}^{-2}.\omega_N^{-2j}.[G_0 + \omega_4^{-2}\,G_1 + (\omega_4^{-2})^2\,G_2 \\ &\quad + (\omega_4^{-3})^2\,G_3\,] \\ a_{4i+3} &= \phi_{2N}^{-3}.\omega_N^{-3j}.[G_0 + \omega_4^{-3}\,G_1 + (\omega_4^{-2})^3\,G_2 \\ &\quad + (\omega_4^{-3})^3\,G_3\,] \end{aligned} \quad (۱۵)$$

می‌توان نتیجه گرفت که $\omega_4^{-2} = -\omega_4^0 = -1$ و $\omega_4^{-3} = -\omega_4^1$ که می‌توان معادلات بالا را به شکل رابطه (۱۶) نشان داد:

$$\begin{bmatrix} a_{4i} \\ a_{4i+1} \\ a_{4i+2} \\ a_{4i+3} \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^{-1} & -1 & -\omega_4^{-1} \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^{-1} & -1 & \omega_4^{-1} \end{bmatrix} \times \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix}\right) \cdot \begin{bmatrix} 1 \\ \omega_N^{-j}.\phi_{2N}^{-1} \\ \omega_N^{-2j}.\phi_{2N}^{-2} \\ \omega_N^{-3j}.\phi_{2N}^{-3} \end{bmatrix} \quad (16)$$

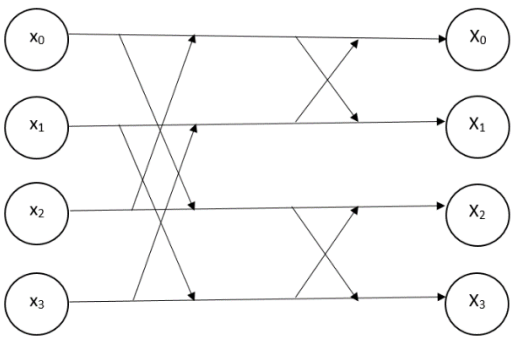
با ادغام twiddle factor ها معادله زیر به دست می‌آید.

$$\begin{bmatrix} a_{4i} \\ a_{4i+1} \\ a_{4i+2} \\ a_{4i+3} \end{bmatrix} = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4^{-1} & -1 & -\omega_4^{-1} \\ 1 & -1 & 1 & -1 \\ 1 & -\omega_4^{-1} & -1 & \omega_4^{-1} \end{bmatrix} \times \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix}\right) \cdot \begin{bmatrix} 1 \\ \phi_{2N}^{-(2j+1)} \\ \phi_{2N}^{-2(2j+1)} \\ \phi_{2N}^{-3(2j+1)} \end{bmatrix} \quad (17)$$

با اعمال این تغییرات در DIT NTT و DIF INTT لازم است که مقادیر twiddle factor ها در ROM ذخیره شود.

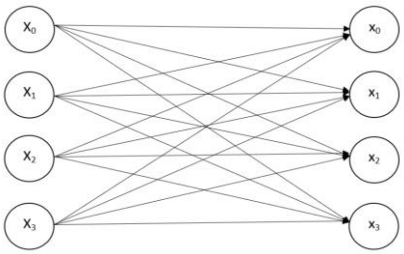
۲-۳- تغییرات در واحد پروانه

برای پیاده‌سازی واحد پروانه به‌طور معمول از پایه دو استفاده می‌شود شکل (۲) نحوه محاسبات در پایه دو را نشان می‌دهد.



شکل (۲). نحوه محاسبات در پایه دو

با استفاده از روش‌هایی می‌توان تعداد ضرب، تفریق و جمع را کاهش داد شکل (۳) نمایی از واحد پروانه چهار نقطه‌ای در پایه چهار را نشان می‌دهد.

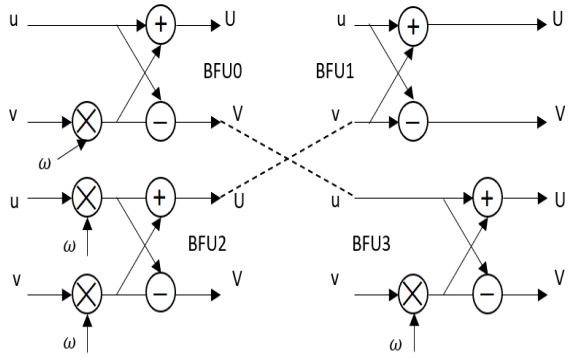


شکل (۳). نحوه محاسبات در پایه چهار

همان‌طور که مشخص است محاسبات در مثال چهار نقطه‌ای، در پایه چهار در یک مرحله انجام می‌شود ولی در پایه دو، در دو مرحله انجام می‌شود که این موضوع نشان می‌دهد که استفاده از پایه چهار می‌تواند باعث کاهش زمان شود.

استفاده از تقسیم و غلبه در واحد پروانه CT

اگر به‌صورت مستقیم معادله (9) 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 چهارضرب، چهار تفریق و چهار جمع نیاز داریم. شکل (۴) DIT NTT را در پایه چهار نشان می‌دهد.

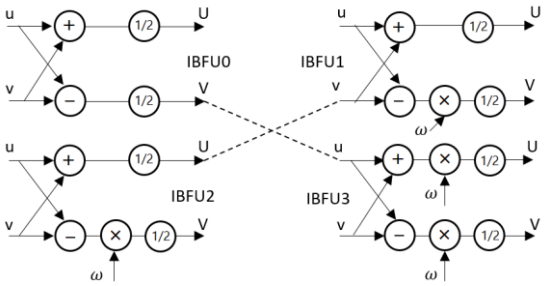


شکل (۴). DIT NTT در پایه ۴

استفاده از تقسیم و غلبه در واحد پروانه GS

اگر به‌صورت مستقیم معادله (17) را محاسبه کنیم، به منابع بیشتری نیاز داریم ولی می‌توان مانند DIT NTT مقادیر به‌دست‌آمده را دوباره استفاده کرد و سپس به چهار ضرب، چهار تفریق و چهار جمع نیاز داریم که به‌صرفه‌تر می‌باشد.

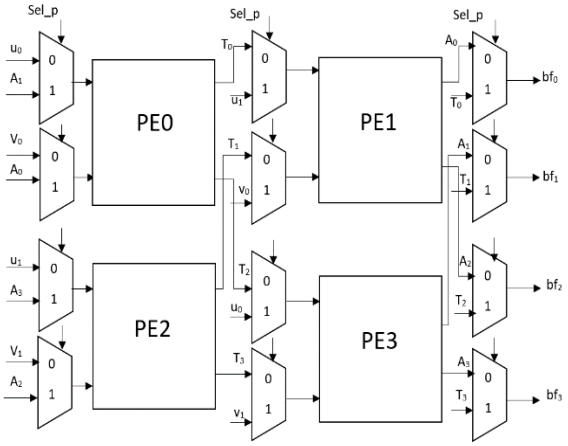
شکل (۵) جزییات DIF INTT را نشان می‌دهد.



شکل (۵). DIF INTT در پایه ۴

واحد پروانه پایه چهار

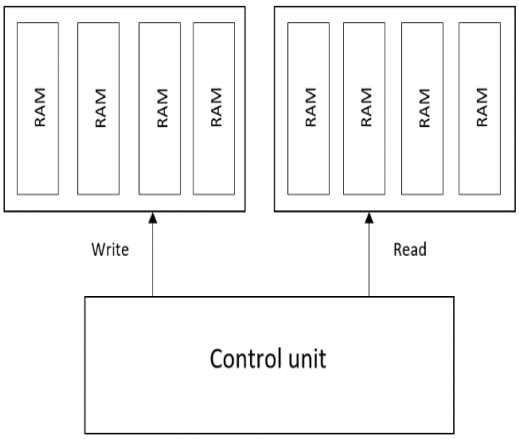
در شکل (۶) از چهار PE[1] یا عنصر پردازشی استفاده شده است. در عناصر پردازشی PE محاسبات مربوط به عملیات واحد پروانه مبنای ۴ انجام می‌شود تا در نهایت خروجی ضرب چندجمله‌ای درجه n تولید شود. هنگامی‌که sel_p یک باشد، دیتای مرتبط به A_i را فراخوانی می‌کند و محاسبات آن در ستون اول PE ها انجام می‌شود و مقادیر 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.



شکل (۶). ساختار مسیردهی در پایه چهار

۳-۳- تغییر در ساختار حافظه

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 (۷) ساختار حافظه را نشان می‌دهد.



شکل (۷). ساختار حافظه در روش پیشنهادی

۴-۳- مناسب‌سازی NTT پایه چهار برای Kyber

در پیاده‌سازی Kyber به‌صورت مستقیم نمی‌توان از NTT پایه

[1] Processing Element

چهار استفاده کرد و NTT پایه چهـار نیـاز بـه تغییراتـی دارد تـا به‌گونه‌ای شود که برای پیاده‌سازی الگوریتم Kyber مناسب باشد.

به‌طور مثال مقدار n و q باید تغییر کننـد، مقـدار n برابـر بـا ۲۵۶ باشد و مقدار q برابر با ۳۳۲۹ باشد و اندازه دیتا مـی‌بایسـت به‌گونه‌ای باشد که با Kyber سازگار باشد.

۴- ارزیابی نتایج پیاده‌سازی

بـا اسـتفاده از روش پیشـنهادی، NTT پایـه چهـار مـورد نظر را روی FPGA های Artix-7 (xc7a200tffg1156-3) و Virtex-7 (xc7vx690tffg1761-2) توسـط Vivado 2019.1 پیـاده‌سـازی شـده است. نتایج به‌دست‌آمده طبق گزارش‌های Vivado می‌باشد و کد روی FPGA ها پروگرام نشده است.

واحد پروانه را بـه‌صـورت جداگانـه روی Artix-7 و Virtex-7 پیاده‌سازی کرده‌ایم که طبـق گـزارش‌هـای بـه‌دسـت‌آمـده روی Artix-7 بـــه ۶۷۶ فلیـــپ فـــلاپ، ۱۱۳۲ LUT و ۴ DSP روی فرکانس ۱۸۵/۱۸ مگاهرتز نیاز اسـت و تـوان مصـرفی 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 مصـرفی آن ۰/۵۰۲ وات می‌باشد.

توان مصرفی کل روش پیشنهادی روی Artix-7 ۰/ ۲۷۹ وات و توان مصرفی روی Virtex-7 ۰/ ۵۲۷ وات می‌باشد.

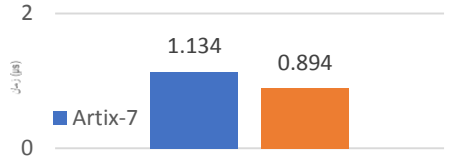
جدول (۲) نتایج به‌دست‌آمده را که حاصل گزارش‌های Vivado می‌باشد، نشان می‌دهد که در ازای افزایش تعـداد BRAMهـا، زمـان اجرا نسبت به سایر پیاده‌سازی‌ها کاهش یافته است.

جدول (۲). نتایج پیاده‌سازی روش پیشنهادی

	Platform	Parameter	LUTs	FFs	DSPs	BRAMs	Freq. (MHz)	NTT (CCs)	INTT (CCs)	**Time (µs)**
روش پیشنهادی	Artix-۷	n=۲۵۶	۲۳۰۱	۹۲۵	۴	۹	۱۸۵/۱۸	۲۱۰	۲۱۰	**۱/۱۳۴**
روش پیشنهادی	Virtex-۷	n=۲۵۶	۲۳۷۵	۹۰۱	۴	۹	۲۳۴/۷۴	۲۱۰	۲۱۰	**۰/۸۹۴**

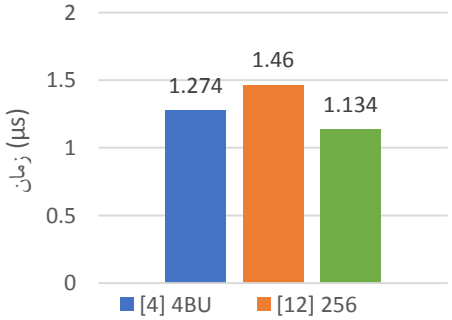
۴-۱- مقایسـه زمـان اجـرا روش پیشـنهادی در پیاده‌سازی‌های مختلف

در شـکل (۸) زمـان اجـرای روش پیشـنهادی روی دو FPGA Artix-7 و Virtex-7 با یکدیگر مقایسه شده‌اند.



شکل (۸). مقایسه زمان اجرا در Artix-7 و Virtex-7

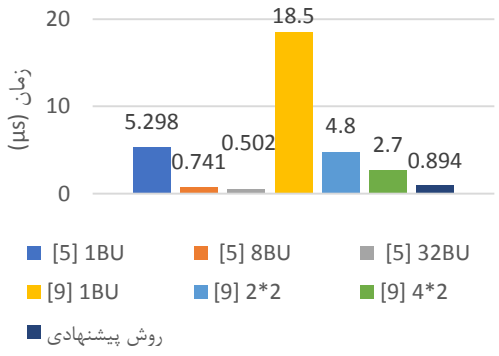
در شـکل (۹) زمـان اجـرای روش پیشـنهادی بـا سـایر پیاده‌سازی‌ها روی Artix-7 مشابه، مقایسه شده است؛ و همان‌طور که دیده می‌شود پیاده‌سازی روش پیشنهادی مـا توانسـته زمـان خوبی را ثبت کند و زمان اجرا ۰/۱۳۴ میکروثانیه است. (منظور از BU واحد پروانه می‌باشد)



شکل (۹). مقایسه پیاده‌سازی‌ها روی Artix-7

زمان اجرای روش پیشـنهادی روی Artix-7 نسـبت بـه ۲۵۶ [۱۲] ۲۸/۷۴ درصـد سـریع‌تـر و نسـبت بـه [۴] 4BU ۱۲/۳۴ درصد سریع‌تر است.

در شـکل (۱۰) زمـان اجـرای روش پیشـنهادی بـا سـایر پیاده‌سازی‌ها روی Virtex-7 مقایسه شده است که مطابق شـکل، روش پیشنهادی توانسته سریع‌تر از اکثر پیاده‌سازی‌هـا باشـد. بـا توجه به اینکه پیاده‌سازی مشابهی برای Virtex-7 وجود نداشـت، به‌ناچار روش پیشنهادی با همه پیـاده‌سـازی‌هـای روی Virtex-7 مقایسه شده است.



شکل (۱۰). مقایسه پیاده‌سازی‌ها روی Virtex-7

زمان اجرای روش پیشنهادی روی Virtex-7 نسـبت بـه [۹] ۴×۲ ۳/۰۲ برابر سریع‌تر، نسبت به [۹] ۲×۲ ۵/۳۶ برابر سریع‌تـر، نسبت به [۹] 1 BU ۲۰/۶۹ برابر سریع‌تر، نسبت بـه [۵] 32BU ۱/۷۸ برابر کندتر، نسبت به [۵] 8BU ۱/۲ برابر کنـدتر و نسـبت به [۵] 1BU ۵/۹۲ برابر سریع‌تر است.

۴-۲- مقایسه کلی روش پیشنهادی با سایر روش‌ها

پیاده‌سازی‌های گوناگونی برای NTT وجود دارد برخی از آن‌ها از طراحی هم‌زمان سخت‌افزار/ نرم‌افزار استفاده کرده‌اند و برخی از طراحی تمام سخت‌افزاری یا برخی از پیاده‌سازی‌ها از تعداد واحد پروانه بیشتری 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 (۳) پیاده‌سازی‌های مشابه با روش پیشنهادی روی Artix-7 مقایسه شده‌اند.

جدول (۳). مقایسه نتایج پیاده‌سازی روی Artix-7

Design	n	BU	LUTs	FFs	DSPs	BRAMs	Freq. (MHz)	NTT (CCs)	INTT (CCs)	Time (µs)
[4]	۲۵۶	۴	۲۵۴۳	۷۹۲	۴	۹	۱۸۲	۲۳۲	۲۳۳	۱/۲۷۴
[12]	۲۵۶	۲*۲	۸۰۱	۷۱۷	۴	۲	۲۲۲	۳۲۴	۳۲۴	۱/۴۶
روش پیشنهادی	۲۵۶	۲*۲	۲۳۰۱	۹۲۵	۴	۹	۱۸۵/۱۸	۲۱۰	۲۱۰	۱/۱۳۴

با مقایسه نتایج، می‌توان متوجه شد که روش پیشنهادی توانسته بهترین زمان اجرا را ثبت کند ولی در مقایسه با مرجع [۱۲]، روش پیشنهادی به منابع بیشتری نیاز دارد به‌طوری‌که ۲/۸۷ برابر LUT بیشتر، ۱/۲۹ برابر فلیپ فلاپ بیشتر، ۴/۵ برابر BRAM بیشتر نیاز دارد و روش پیشنهادی در مقایسه با مرجع [۴] به LUT کمتر، فلیپ فلاپ بیشتر نیاز دارد. در کل نتایج به‌دست‌آمده روی Artix-7 خوب و راضی‌کننده بودند.

در جدول (۴) نتایج پیاده‌سازی‌های مختلف روی Virtex-7 نشان داده‌شده است. در این جدول پژوهش‌های مختلف، به ازای مقادیر مختلف برای n و نوع پروانه مورد مقایسه و ارزیابی قرار گرفته‌اند.

جدول (۴). نتایج پیاده‌سازی روی Virtex -7

Design	n	BU	LUTs	FFs	DSPs	BRAMs	Freq. (MHz)	NTT (CCs)	INTT (CCs)	Time (µs)
[5]	۲۵۶	۱	۲۱۲۸	۱۱۴۴	۸	۳	۱۷۴	۹۲۲	۱۱۸۴	۵/۲۹۸
[5]	۲۵۶	۸	۱۱K	۵۴۲۲	۶۴	۱۲	۱۸۶	۱۳۸	۱۷۶	۰/۷۴۱
[5]	۲۵۶	۳۲	۶۱K	۱۷K	۲۵۶	۴۸	۱۶۷	۸۴	۱۰۱	۰/۵۰۲
[9]	۱۰۲۴	۱	۴۷۵	۳۰۷	۳	۱/۵	۲۷۸	-	۵۱۳۴	۱۸/۵
[9]	۱۰۲۴	۲*۲	۱۱۹۶	۹۶۹	۱۲	۳	۲۷۰	-	۱۳۰۸	۴/۸
[9]	۱۰۲۴	۲*۴	۲۹۵۳	۱۸۷۵	۲۷	۵/۵	۲۵۰	-	۶۶۸	۲/۷
روش پیشنهادی	۲۵۶	۲*۲	۲۳۷۵	۹۰۱	۴	۹	۲۳۴/۷۴	۲۱۰	۲۱۰	۰/۸۹۴

پیاده‌سازی روش پیشنهادی در مقایسه با مرجع [۵] که یک واحد پروانه دارد هم‌زمان اجرا و هم منابع موردنیاز کمتری (در بیشتر اوقات) دارد، به‌طوری‌که زمان اجرا در مرجع [۵] با یک واحد پروانه، ۵/۹۲ برابر، تعداد BRAMها ۰/۳۳ برابر، تعداد DSPها دو برابر، تعداد فلیپ فلاپ‌ها ۱/۲۶ برابر و تعداد LUT ها ۰/۸۹۶ برابر بیشتر می‌باشد.

در پیاده‌سازی [۵] که چهار واحد پروانه بیشتر نسبت به روش پیشنهادی دارد و ۰/۱۵۳ میکروثانیه سریع‌تر است ولی تعداد BRAM هایش ۱/۳۳ برابر، تعداد DSPهایش ۱۶ برابر، تعداد فلیپ فلاپ‌هایش ۶/۰۱ برابر و تعداد LUT هایش ۴/۶۳ برابر بیشتر می‌باشد.

اگرچه نتایج به‌دست‌آمده روش پیشنهادی روی Virtex-7 به‌خوبی نتایج حاصل از پیاده‌سازی بر روی تراشه Artix-7 نیست ولی نتایج باز هم قابل‌قبول هستند.

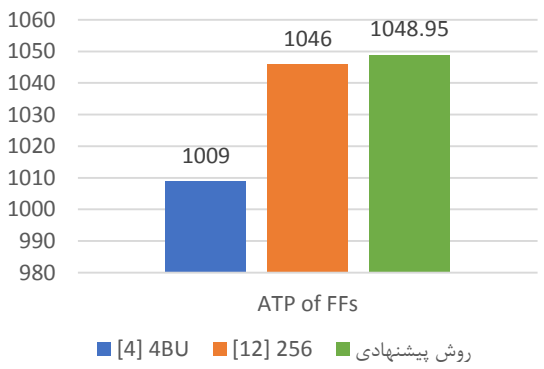
۴-۳- شاخص بهره‌وری ATP

این شاخص از حاصل‌ضرب زمان در مساحت به دست می‌آید و مقدار مساحت بر اساس فلیپ فلاپ‌ها یا LUT ها یا DSP ها یا BRAM ها به‌دست می‌آید و عدد این شاخص هرچقدر کوچک‌تر باشد، بهتر است.

مقدار شاخص ATP در Artix-7

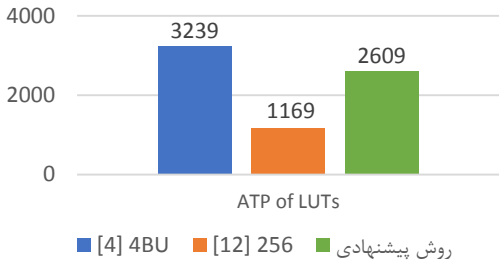
- ATP در فلیپ فلاپ‌ها

شکل ۱۱ مقدار شاخص ATP را در فلیپ فلاپ‌ها نشان می‌دهد و طبق شکل، پیاده‌سازی [۴] BU۴ بهترین مقدار را به دست آورده است و روش پیشنهادی با مقدار ۱۰۴۸/۵ در جایگاه سوم قرار دارد.



شکل (۱۱). شاخص ATP در فلیپ فلاپ‌ها روی Artix-7

- ATP در LUTها

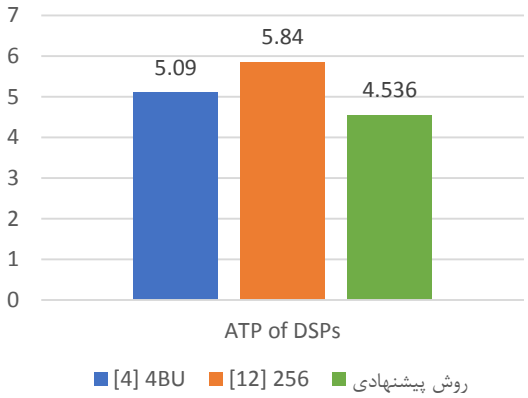


شکل (۱۲). شاخص ATP در LUTها روی Artix-7

در شکل (۱۲) که مقدار شاخص ATP در LUT ها را نشان می‌دهد طبق نتایج به‌دست‌آمده پیاده‌سازی 256 [۱۲] بهترین نتیجه را به دست آورده و روش پیشنهادی با مقدار ۲۶۰۹ در رتبه دوم قرار دارد.

- ATP در DSP

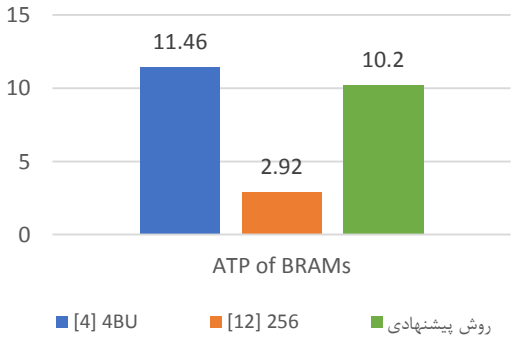
در شکل (۱۳) که مقدار شاخص ATP در DSP ها را نمایش می‌دهد، پیاده‌سازی روش پیشنهادی با مقدار ۴/۵۳۶، در مقایسه با سایر روش‌ها، کمترین مقدار را دارد و در جایگاه نخست قرار گرفته است.



شکل (۱۳). شاخص ATP در DSP روی Artix-7

- ATP در BRAM ها

در شکل (۱۴) که مقدار شاخص ATP در BRAM ها را نشان می‌دهد و پیاده‌سازی 256 [۱۲] کمترین مقدار را دارد و روش پیشنهادی مقام دوم را کسب کرد.



شکل (۱۴). شاخص ATP در BRAM روی Artix-7

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 موردنیاز و زمان اجرا روبه‌رو بودند و هستند. برای حل این چالش‌ها روش‌های مختلفی پیشنهاد شده است. از تغییر در بخش‌های محاسباتی و استفاده از تکنیک‌هایی در سطح معماری کامپیوتر، برای بهبود پیاده‌سازی و کاهش زمان این الگوریتم‌ها استفاده شده است. در بهینه‌سازی‌های CRYSTALS-Kyber اکثراً به NTT توجه کرده‌اند و علاوه بر NTT به نحوه سازمان‌دهی حافظه هم توجه شده است و بیشتر پیاده‌سازی‌هایی که زمان کمی داشتند، به این موضوعات توجه کرده‌اند. در این پژوهش، از NTT پایه چهار استفاده شده که باعث تغییرات در سایر قسمت‌های NTT شده است و توانسته زمان اجرای کمتری در مقایسه با سایر پیاده‌سازی‌ها داشته باشد و به منابع موردنیاز متعادلی نیاز دارد. تاکنون پیاده‌سازی‌های بسیاری برای الگوریتم‌های رمزنگاری پساکوانتوم ارائه شده است و هرکدام ویژگی‌های خاص خود را داشته‌اند. روش پیشنهادی با تغییر مبنا در NTT توانست زمان اجرا را بهبود دهد و در مقایسه با پیاده‌سازی‌های مشابه توانست ۲۸/۷۴ درصد و ۱۲/۳۴ درصد زمان اجرای کوتاه‌تری را ثبت کند.

جهت توسعه کار در آینده، استفاده از مبناهای بالاتر می‌تواند در زمان اجرا تأثیرگذار باشد مثلاً اگر n=256 باشد، می‌توان از پایه شانزده استفاده کرد ولی محاسباتش پیچیده می‌شود و استفاده از سایر روش‌ها برای ضرب می‌تواند باعث تغییر در زمان اجرا شود.

ی همگرایی و

۶- مراجع

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

[2] Xie, J., Basu, K., Gaj, K., & Guin, U., "Special Session: The Recent Advance in Hardware Implementation of Post-Quantum Cryptography,"In IEEE 38th VLSI Test Symposium (VTS), pp. 1-10, 2020.

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

[4] Yarman, F., Mert, A.C., Öztürk, E. & Savaş, E., "A hardware accelerator for polynomial multiplication operation of CRYSTALS-KYBER PQC scheme," In 2021 Design, Automation & Test in Europe Conference & Exhibition pp. 1020-1025, 2021.

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

[6] Xing, Y. & Li, S., "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp.328-356, 2021.

[7] Guo, W., Li, S. & Kong, L., "An Efficient Implementation of KYBER," IEEE Transactions on Circuits and Systems II: Express Briefs, 2021.

[8] Zhang, C., Liu, D., Liu, X., Zou, X., Niu, G., Liu, B. & Jiang, Q., "Towards efficient hardware implementation of NTT for kyber on FPGAs," In 2021 IEEE International Symposium on Circuits and Systems (ISCAS) pp. 1-5, 2021.

[9] Chen, X., Yang, B., Yin, S., Wei, S. & Liu, L., "CFNTT: Scalable Radix-2/4 NTT Multiplication Architecture with an Efficient Conflict-free Memory Mapping Scheme," IACR Transactions on Cryptographic Hardware and Embedded Systems, pp.94-126, 2022.

[10] Garrido, M., Grajal, J., Sanchez, M.A. & Gustafsson, O., "Pipelined radix-$2^{k} $ feedforward FFT architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 23-32, 2011.

[11] Swartzlander, E.E., Young, W.K. & Joseph, S.J., "A radix 4 delay commutator for fast Fourier transform processor implementation," IEEE Journal of solid-state circuits, pp. 702-709, 1984.

[12] Bisheh-Niasar, M., Azarderakhsh, R. & Mozaffari-Kermani, M., "High-speed NTT-based polynomial multiplication accelerator for CRYSTALS-Kyber post-quantum cryptography," Cryptology ePrint Archive, 2021.

[13] Doustimotlagh, S. N., "A New Mechanism for Enhancing the Security of Military Internet of Things by Using Quantum and Classic Cryptography," Electronic and Cyber Defense, vol. 9, no. 6, 2021, pp. 29-49, 2021. (In Persian)