پیاده سازی الگوریتم رمز بلوکی SPECK بر روی FPGA با استفاده از ساختار حلقه

ایرج هادی نژاد*[۱]، محمد امین امیری[۲]
۱- دانشجوی دکتری، دانشگاه صنعتی مالک اشتر.
۲- استادیار، دانشگاه صنعتی مالک اشتر.

خلاصه

الگوریتم رمز SPECK از دسته الگوریتم‌های رمز بلوکی سبک وزن می‌باشد که علاوه بر امنیت بالا و مقاومت در برابر روش‌های حمله شناخته شده، بطور همزمان عملکرد سخت‌افزاری و نرم‌افزاری بهینه‌ای داشته و برای پیاده‌سازی در تمام کاربردهای سبک قابل استفاده می‌باشد. در این مقاله روشی برای پیاده‌سازی الگوریتم رمز SPECK128/128 بر روی FPGA بر مبنای ساختار حلقه پیشنهاد شده است که در مقایسه با روش‌های موجود ضمن کاهش سطح مصرفی، نرخ بازدهی را به مقدار قابل توجهی افزایش داده است. روش پیشنهادی با ارائه یک توصیف ساختاری از ماژول رمز و تبدیل آن به چند زیر ماژول در قالب یک مدل ماشین حالت، الگوریتم رمز SPECK را اجرایی می‌کند. علاوه براین به منظور بهبود عملکرد روش پیشنهادی، یک ساختار موازی‌سازی نیز به معماری حلقه اضافه شده که سبب افزایش نرخ بازدهی شده است. نتایج شبیه سازی نشان دهنده کارایی الگوریتم پیشنهادی و عملکرد مناسب آن برای پیاده‌سازی بر روی تراشه FPGA است.

کلمات کلیدی: رمز بلوکی SPECK، پیاده‌سازی FPGA، معماری حلقه، موازی سازی.

۱. مقدمه

آژانس امنیت ملی آمریکا (NSA) در سال ۲۰۱۳ یک خانواده جدیدی از رمزنگاری بلوکی سبک وزن به نام SPECK را معرفی کرد[۱]. هدف از ارائه این الگوریتم رمزنگاری، ارائه امنیت، انعطاف‌پذیری و مقاومت مناسب در برابر حملات تحلیلی در کاربردهای کم هزینه برای پیاده‌سازی بهینه سخت‌افزاری و نرم افزاری بوده است. الگوریتم SPECK شامل ۱۰ نسخه متنوع با ابعاد کلید و بلوک داده متفاوت می باشد که امکان پیاده‌سازی دلخواه برای کاربر را با توجه به کاربردهای مشخص و امنیت مورد نیاز فراهم می‌کند. امروزه از الگوریتم رمز SPECK به دلیل سادگی در پیاده‌سازی و امکان اجرا با سرعت بالا در کاربردهایی متعددی نظیر اینترنت اشیا (IOT) ، RFID و ... استفاده می‌شود[۲].

FPGAها به دلیل انعطاف پذیر بودنشان در پیاده‌سازی ماژول‌های رمزنگاری با معماری‌های مختلف و همین‌طور قابلیت اجرا با سرعت و امنیت متفاوت، به طور گسترده برای رمزکردن اطلاعات در سیستم های مختلف مورد استفاده قرار می-گیرند[۳]. به همین دلیل در کنار پیاده‌سازی انجام شده برای الگوریتم SPECK به صورت نرم افزاری بر روی میکروکنترلهایی نظیر AVR [۴]، پیاده‌سازی سخت افزاری متنوعی نیز با FPGAها با معماری‌هایی نظیر حلقه، unrollred و بیتی-سریالی برای نسخه‌های مختلف انجام شده و عملکرد آنها مقایسه شده است [۵-۷].

* Email:hadinejad.iraj@gmail.com

معماری unrolled بیشتر در مواقعی استفاده می‌شود که صرف نظر از سطح مصرفی برای پیاده سازی الگوریتم رمز، بازدهی بالا مدنظر باشد. در عوض در معماری حلقه، کاهش سطح مصرفی برای پیاده سازی بیشتر از بالا بودن بازدهی مورد توجه می‌باشد. در معماری بیتی-سریالی هم عملیات رمزنگاری صرفا بر روی بیت‌های ورودی انجام می‌شود که علاوه بر کاهش چشمگیر سطح مصرفی، کاهش شدید بازدهی الگوریتم رمز را نیز به دنبال خواهد داشت[۸].

در ادامه این مقاله و در قسمت دوم الگوریتم رمز بلوکی SPECK معرفی شده است. در قسمت سوم روشی برای پیاده‌سازی الگوریتم رمز SPECK128/128 بر مبنای ساختار حلقه پیشنهاد شده که ضمن کاهش سطح مصرفی، نرخ بازدهی را نیز افزایش داده است. علاوه بر این به منظور بهبود عملکرد روش پیشنهادی در این قسمت یک ساختار موازی به معماری حلقه افزوده شده که ضمن کاهش تعداد حلقه‌های اجرای الگوریتم، نرخ بازدهی را نیز افزایش داده است. در قسمت چهارم هم برای ارزیابی عملکرد الگوریتم پیشنهادی، شبیه‌سازی و پیاده‌سازی بر روی تعدادی از خانواده‌های تراشه FPGA اجرا و با الگوریتم‌های دیگر که تا کنون پیاده‌سازی شده‌اند مقایسه شده است. در نهایت در قسمت پنجم نتیجه‌گیری ارائه شده است.

۲. رمز بلوکی SPECK

خانواده رمز بلوکی سبک-وزن Speck در قالب ۱۰ نسخه به گونه‌ای طراحی شده است که بتواند عملکرد مناسبی را در هر دو کاربرد سخت افزاری و نرم افزاری داشته باشد. هر نسخه از الگوریتم رمز خانواده Speck با طول کلمه n-بیتی و تعداد m-کلمه برای کلید، به صورت Speck2n/mn نشان داده می‌شود. در اینجا mn برابر با طول بیت‌های کلید و 2n هم برابر با طول بیت‌های هر بلوک ورودی می‌باشد. برای مثال Speck128/128 بیان کننده نسخه‌ای از رمز بلوکی Speck با طول بلوک ۱۲۸بیت و طول کلید ۱۲۸ بیت خواهد بود.

۱،۲.تابع دور[1]

الگوریتم رمز Speck2n در فرآیندهای رمزنگاری و رمزگشایی (با کلمات n-بیتی و طول بلوک 2n-بیتی) از عملگرهای زیر استفاده می‌کند:

عملگر XOR بیتی، $\oplus$،

عملگر جمع پیمانه‌ای 2^n ، $+$ و

چرخش بیتی به اندازه j بیت به چپ و راست که به ترتیب با S^j و S^{-j} نمایش داده می‌شوند.

برای $k \in \mathrm{GF}(2)^n$، تابع دور Speck2n وابسته به کلید k نگاشتی از $R_k: \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n \rightarrow \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n$ می‌باشد که با رابطه زیر تعریف می‌شود:

$$R_k(x,y) = \left((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k\right) \quad (۱)$$

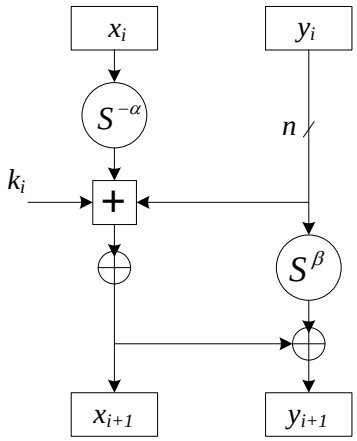
در رابطه بالا، x، کلمه سمت چپ از بلوک داده، y، کلمه سمت راست از بلوک داده و k هم کلید محاسبه شده برای دور متناظر است. تنها برای حالتی که $n = 16$ باشد مقادیر چرخش برابر با $\alpha = 7$ و $\beta = 2$ بوده و برای سایر حالت‌ها مقادیر چرخش برابر با $\alpha = 8$ و $\beta = 3$ در نظر گرفته شده‌اند.

فرآیند معکوس تابع دور که برای رمزگشایی به کار می‌رود از تفریق پیمانه‌ای به جای جمع پیمانه‌ای استفاده می‌کند و با رابطه زیر تعریف می‌شود:

$$R_k^{-1}(x,y) = \left(S^{\alpha}\left((x \oplus k) - S^{-\beta}(x \oplus y)\right), S^{-\beta}(x \oplus y)\right) \quad (2)$$

[1] Round function

برای تولید کلید در رمز SPECK یک کلید اصلی اولیه دریافت شده و از روی آن یک دنباله از T کلمه کلید $k_0, \dots, k_{T-1}$ بسط داده شده و برای رمز کردن در هر دور متناظر مورد استفاده قرار می‌گیرد. بلوک دیاگرام اجرای یک دور از الگوریتم رمز SPECK در شکل ۱ نشان داده شده است. فرآیند رمزنگاری از ترکیب دورهای متوالی $R_{k_{T-1}} \text{o} \dots \text{o}\ R_{k_1} \text{o} R_{k_0}$ انجام می‌شود.



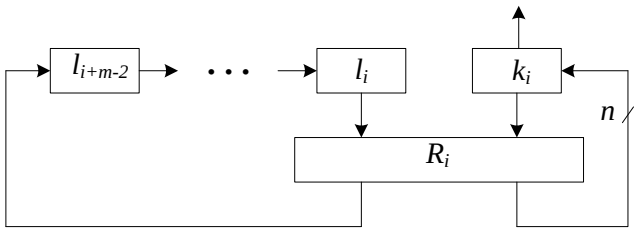
شکل ۱ –نمایش نمادین یک دور از الگوریتم Speck، که در آن (x_i, y_i) بیان کننده ورودی دور فعلی بعد از اجرای i دور از عملیات رمز می باشد.

۲,۲.بسط کلید[1]

در فرآیند بسط کلید مطابق شکل۲، یک کلید اصلی $K = (l_{m-2}, \dots, l_0, k_0)$ با فرض $l_i, k_0 \in GF(2)^n$ و انتخاب m از مقادیر [2,3,4]، به عنوان ورودی اولیه در نظر گرفته شده و در ادامه با انجام عملیات‌های متوالی، زیر کلیدهای k_i متناظر با هر دور i $(0 \leq i < T)$ به کمک روابط (۵) و (۶) تولید می‌شوند.

$$l_{i+m-1} = (k_i + S^{-\alpha} l_i) \oplus i \qquad (۵)$$

$$k_{i+1} = S^{\beta} k_i \oplus l_{i+m-1} \qquad (۶)$$



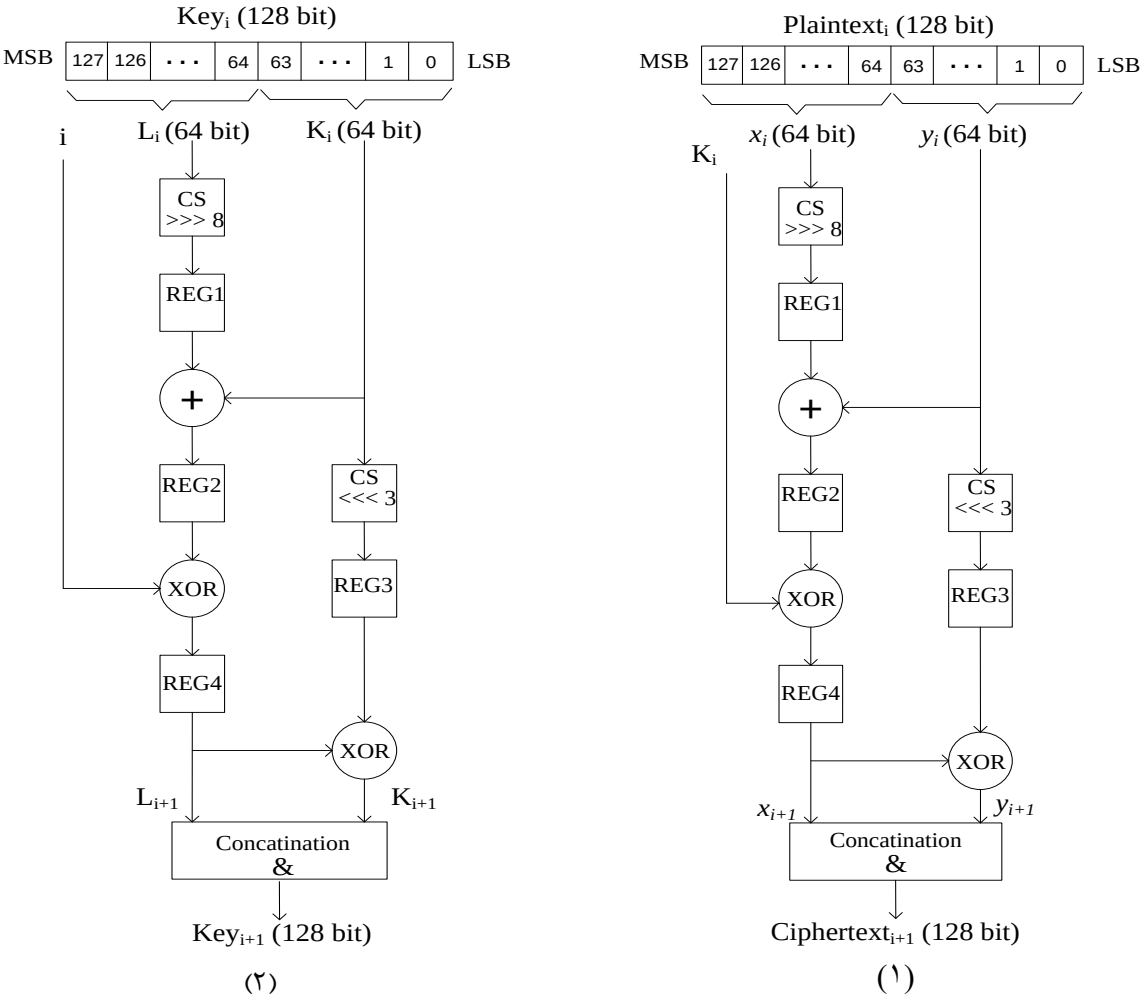
شکل ۲–نمایش بسط کلید SPECK، درآن R_i تابع دور Speck است که با i به عنوان شماره دور کلید عمل می‌کند.

۳. پیاده سازی بر روی FPGA

در این مقاله الگوریتم رمز بلوکی SPECK128/128 با تعداد دور ۳۲ (T=32) و مقادیر $\alpha = 8$ و $\beta = 3$، برای پیاده‌سازی با معماری حلقه بر روی تراشه FPGA با زبان توصیف سخت افزار VHDL انتخاب شده است. برای اجرای الگوریتم رمز در ابتدا می‌بایست مدلی برای پیاده‌سازی تابع دور و فرآیند تولید کلید ارائه شود تا در ادامه با ترکیب آنها

[1] Key expansion

امکان اجرای یک دور از الگوریتم رمز برای تبدیل متن کشف[1] به متن رمز شده[2] بر مبنای کلید ورودی[3] فراهم شود. به همین منظور مدلی برای پیاده‌سازی تابع دور و فرآیند تولید کلید مطابق با شکل ۳ پیشنهاد شده است که در ادامه مراحل اجرای آن بیان می‌شود.



شکل ۳–نمودار بلوکی پیاده سازی روش پیشنهادی برای اجرای یک دور از عملیات رمزنگاری.(۱)پیاده سازی تابع دور.(۲) پیاده فرآیند بسط کلید.

۱،۳.پیاده‌سازی تابع دور

برای پیاده‌سازی تابع دور از نمودار بلوکی مطابق با شکل ۳-(۱) استفاده شده است. همانطوریکه مشخص است عملیات رمز بر روی پیام دریافتی x_i, y_i به همراه کلید K_i به عنوان ورودی در دور iام اجرا می‌شود. در حین اجرای الگوریتم رمز از عملگرهای جمع پیمانه‌ای$(+)$ ، چرخش بیتی (CS) به اندازه j به چپ $(\lll j)$ یا راست $(\ggg j)$ و XOR بیتی استفاده شده است. بعد از اجرای هر عملیات، محتوای داده در یک سیگنال حافظه (REG) ذخیره شده و در نهایت با محاسبه مقادیر متن رمز شده x_{i+1}, y_{i+1}، و الحاق آنها با یکدیگر، ورودی برای دور بعدی یعنی $i+1$ تولید می‌شود.

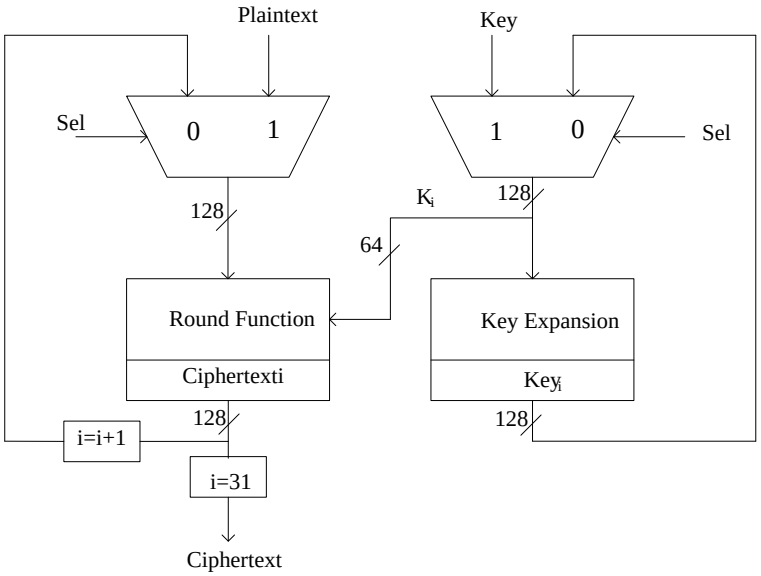
[1] Plaintext
[2] Ciphertext
[3] Key

۲،۳. پیاده سازی فرآیند بسط کلید

پیاده‌سازی قسمت بسط کلید بر اساس نمودار بلوکی مطابق با شکل ۳-(۲) صورت گرفته است. همانطوریکه مشخص است در ابتدا کلید K_i و L_i به همراه عدد دور i به عنوان ورودی دریافت شده و در ادامه با استفاده عملگرهای جمع پیمانه‌ای(+)، چرخش بیتی (CS) و XOR بیتی و ذخیره‌سازی محتوای داده در چند حافظه (REG)، کلید K_{i+1} و L_{i+1} مطابق با الگوریتم رمز تولید شده و به دور بعد ارسال می‌شود.

۳،۳.پیاده سازی ماژول رمز بر مبنای معماری حلقه

این مدل از پیاده‌سازی که مانند یک حلقه متوالی عمل می‌کند، یک دور از الگوریتم Speck را به ترتیب با یک یا چند کلاک در هر حلقه اجرا کرده و در ادامه خروجی هر دور را به عنوان ورودی برای حلقه تکرار بعدی استفاده می‌کند. این نوع از معماری معمولا در مواردی استفاده می‌شود که مساحت مصرفی کمتر برای پیاده‌سازی مد نظر باشد. نمودار بلوکی روش پیشنهادی برای اجرای الگوریتم Speck128/128 با ساختار حلقه در شکل ۴نشان داده شده است.

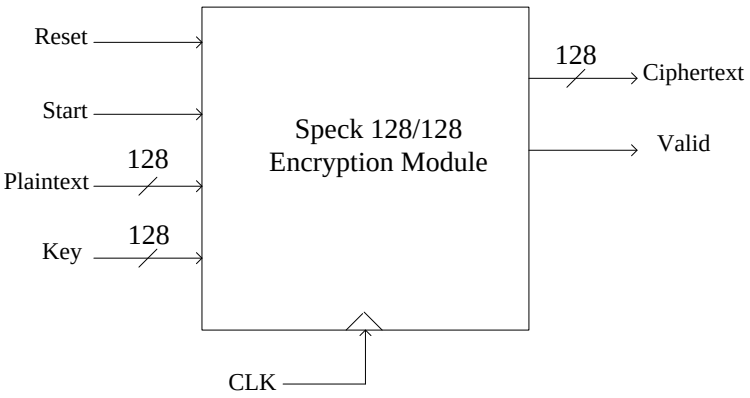


شکل ۴–بلوک دیاگرام پیشنهادی پیاده سازی الگوریتم SPECK128/128 با ساختار حلقه.

همانطوریکه از روی شکل۴ مشخص است در اولین حلقه و در دور اول (i=0)، با یک شدن مقدار سیگنال کنترل Sel، متن کشف به همراه کلید اولیه به عنوان ورودی به تابع دور داده می‌شود و بعد از انجام محاسبات، داده رمز شده در یک حافظه ذخیره می‌شود. علاوه بر این، همزمان با اجرای تابع دور، عملیات بسط کلید با ورودی کلید اولیه نیز انجام شده و مقدار آن در حافظه دیگری ذخیره می‌شود. در حلقه‌های تکرار بعدی (0 < i < T-1) مقدار سیگنال کنترلی Sel صفر شده و مقادیر ذخیره شده در حافظه‌ها به عنوان ورودی به تابع دور و مرحله بسط کلید داده می‌شوند. در حلقه آخر (T-1) هم، مقدار خروجی محاسبه شده به عنوان متن رمز شده نهایی در نظر گرفته می‌شود.

از نگاه کلی، ورودی‌ها و خروجی‌های ماژول رمز پیشنهادی در شکل ۵ نشان داده شده اند. در ساختار پیشنهادی، سیگنال Start به عنوان یک سیگنال ورودی برای اجرای فرآیند رمزنگاری در نظر گرفته شده است. در واقع با یک شدن مقدار سیگنال Start، فرآیند رمزنگاری بر روی متن کشف بر اساس کلید اولیه ورودی آغاز شده و با هر لبه بالارونده از سیگنال clk، یک حلقه از الگوریتم رمز (معادل با یک تابع دور و فرآیند بسط کلید) اجرا می‌شود. در ادامه با اجرای عملیات رمزنگاری در قالب ۳۲ حلقه متوالی، متن رمز شده نهایی به خروجی تحویل داده شده و همزمان سیگنال Valid هم که

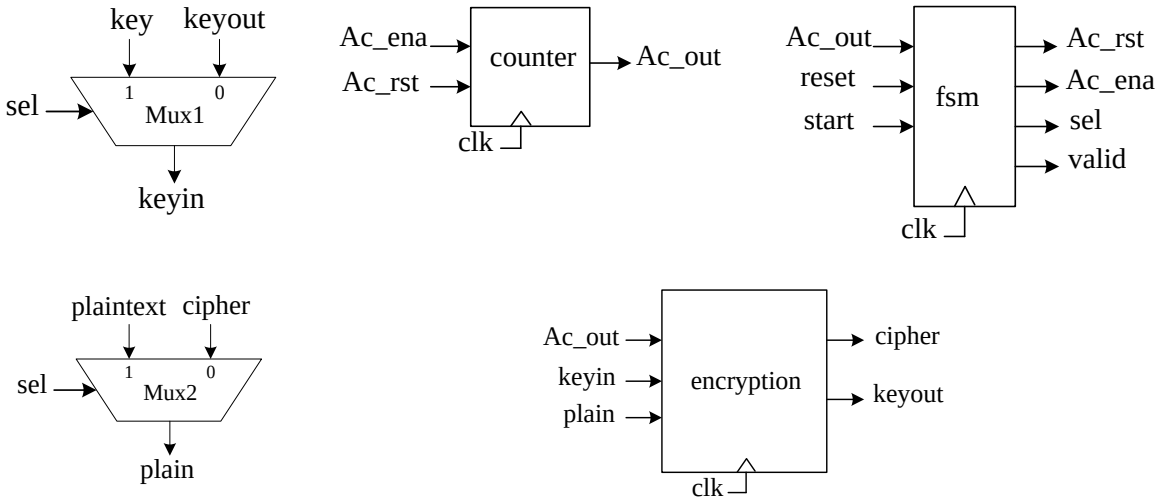
برای تایید درستی متن رمز شده در خروجی در نظر گرفته شده است، مقدار آن برابر یک خواهد شد. سیگنال Reset هم در ورودی برای متوقف کردن عملکرد ماژول رمز در نظر گرفته شده که در صورت یک شدن آن، عملیات رمزنگاری متوقف و سیستم منتظر دریافت سیگنال Start برای اجرای مجدد عملیات رمزنگاری می‌شود.



شکل ۵–نمایش ورودی ها و خروجی های ماژول رمز پیشنهادی با معماری حلقه.

۴,۳.پیاده‌سازی ماژول رمز به صورت ساختاری

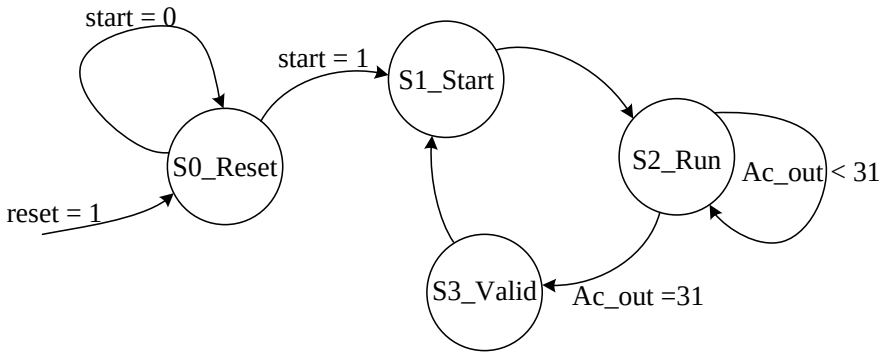
هدف از پیاده‌سازی به صورت ساختاری، تبدیل ماژول اصلی رمز به چند زیر ماژول مجزا و ایجاد ارتباط آنها با یکدیگر برای بهبود عملکرد الگوریتم رمز می‌باشد. در طراحی پیشنهادی با معماری حلقه، از پنج زیر ماژول که در شکل ۶ نشان داده شده، استفاده شده است.



شکل ۶–زیر ماژول های تعریف شده برای پیاده سازی ماژول رمز به همراه سیگنال های میانی ارتباطی و سیگنال‌های ورودی–خروجی.

مطابق با شکل ۶، زیر ماژول‌های شمارنده (counter) برای شمارش تعداد دورها، رمزکننده (encryption) برای اجرای عملیات رمزنگاری تابع دور و فرآیند بسط کلید، دو مالتی پلکسر MUX1 و MUX2 به ترتیب برای کنترل مقداردهی کلید اولیه در فرآیند بسط کلید و متن کشف اولیه درتابع دور به همراه یک ماشین حالت (fsm) برای کنترل سیگنال‌های ارتباطی بین زیر ماژول‌ها در نظر گرفته شده‌اند.

در این طراحی برای پیاده‌سازی ماشین حالت fsm از نمودار حالت ترسیم شده در شکل۷ استفاده شده است. همانطوریکه مشخص است ماشین حالت طراحی شده دارای ۴ حالت S0_Reset برای متوقف کردن عملیات رمزنگاری و انتظار برای یک شدن سیگنال start، S1_start برای انجام تنظیمات اولیه و آمادگی اجرای الگوریتم رمزنگاری، S2_Run برای اجرای ۳۲ حلقه تکرار متوالی به منظور پیاده‌سازی ۳۲ تابع دور و فرآیند بسط کلید و S3_Valid هم برای تایید خروجی رمز شده می‌باشد. برای بیان بهتر عملکرد ماشین حالت طراحی شده، مقادیر سیگنال‌های کنترلی در حالت‌های مختلف و تغییرات آنها در جدول ۱ آورده شده است. به این ترتیب در ساختار طراحی شده می‌توان با دریافت متن کشف و کلید بعد از ۳۲ کلاک، متن رمز شده نهایی را در خروجی دریافت کرد.



شکل ۷–نمایش نمودار حالت مورد استفاده در پیاده سازی زیر ماژول fsm.

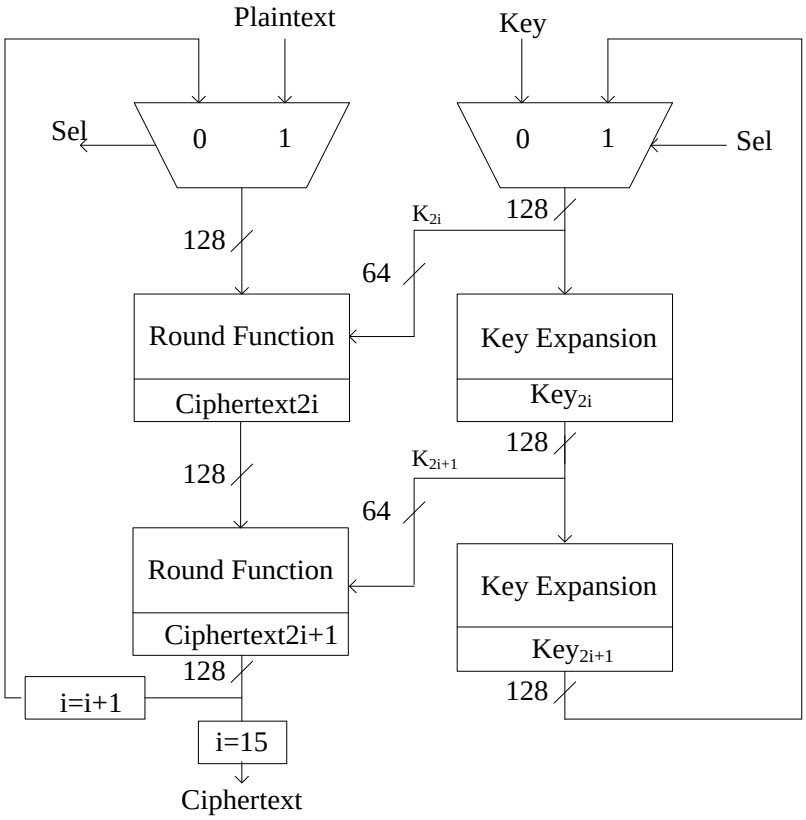
جدول۱- مقادیر سیگنال های کنترلی مربوط به زیر ماژول fsm در ۴ حالت متفاوت.

Signals / States	AC_rst	AC_ena	AC_out	sel	valid
S0_reset	×	×	×	×	×
S1_start	1	0	0	1	0
S2_Run	0	1	0-31	0	0
S3_valid	0	0	31	0	1

۵،۳. بهبود عملکرد معماری حلقه با موازی سازی

برای پیاده‌سازی الگوریتم رمز در FPGA، داشتن بازدهی بالاتر و سطح مصرفی کمتر مطلوب می‌باشد. استفاده از ساختار حلقه با وجود کاهش سطح مصرفی، ممکن است نیازمندی مربوط به افزایش بازدهی را برطرف نکند. به همین دلیل در این مقاله یک ساختار موازی به منظور کاهش تعداد حلقه‌ها و در نتیجه کاهش تعداد کلاک‌های لازم برای پیاده‌سازی الگوریتم رمز با بازدهی بیشتر ارائه شده است. مدل موازی سازی در ساختار پیشنهادی بر اساس کاهش تعداد حلقه‌های تکرار از ۳۲ به ۱۶ مطابق با نمودار بلوکی شکل ۸ طراحی شده است. در واقع در هر حلقه تکرار در این مدل، دو بار فرآیند بسط کلید و تابع دور به صورت متوالی به اجرا در آمده و در نتیجه آن با تعداد کلاک کمتر، خروجی با بازدهی بیشتر حاصل شده است.

کاهش بیشتر تعداد حلقه‌ها با وجود افزایش بازدهی، میزان سطح مصرفی را بر روی تراشه FPGA افزایش داده و بنابراین ضروری بودن وجود یک مصالحه بین انتخاب بازدهی بیشتر و سطح مصرفی کمتر را نشان می‌دهد. به همین منظور در روش پیشنهادی، تعیین تعداد مناسب حلقه تکرار بر اساس معیار بیشینه نسبت بازدهی به سطح مصرفی (Throughput/Area)، در پیاده‌سازی الگوریتم رمز با عملکرد بهتر توصیه می‌شود.



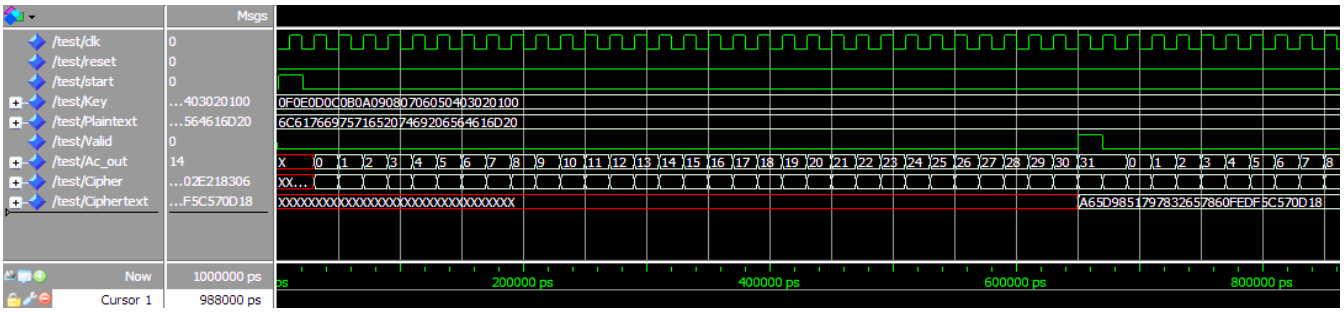
شکل ۸–نمایش مدل پیشنهادی موازی‌سازی با کاهش تعداد دور در معماری حلقه.

۴. شبیه سازی و پیاده سازی

برای ارزیابی عملکرد الگوریتم پیشنهادی و صحت پیاده‌سازی آن مقادیر کلید، متن کشف و متن رمز شده مطابق جدول۲ و از نوع هگزا دسی مال در نظر گرفته شده اند. علاوه بر این، پیاده سازی به زبان VHDL با نرم افزار ISE و بر روی تراشه FPGA از خانواده Spartan6 با شماره XC6SLX150 انجام شده و نتایج آن بعد از شبیه سازی با فرکانس کلاک ورودی ۵۰MHz توسط نرم افزار Modelsim ، در شکل ۹ نشان داده شده است. همانطوریکه مشخص است بلافاصله بعد از دریافت سیگنال Start، سیگنال Sel فعال شده و متن رمز شده در ۳۲ کلاک بعد در خروجی تولید شده است. علاوه بر این در این پیاده‌سازی، مقدار Slice مصرفی برابر با ۱۳۴ و حداکثر فرکانس کاری آن هم ۲۵۹,۴۴ Mhz به دست آمده است. بنابراین با توجه به این که ۳۲ کلاک طول می‌کشد تا بعد از دریافت ورودی، متن رمز شده با طول ۱۲۸ بیت در خروجی تولید شود، مقدار بازدهی در این پیاده‌سازی برابر با ۱۰۳۷,۷۶ Mb/s خواهد بود.

جدول ۲– مقادیر کلید و متن کشف مورد استفاده برای پیاده سازی الگوریتم SPECK 128/128

Key	0f0e0d0c0b0a0908 0706050403020100
Plaintext	6c61766975716520 7469206564616d20
Ciphertext	a65d985179783265 7860fedf5c570d18



شکل ۹–نتایج پیاده‌سازی الگوریتم رمز بلوکی Speck 128/128 با معماری حلقه با نرم‌افزار Modelsim.

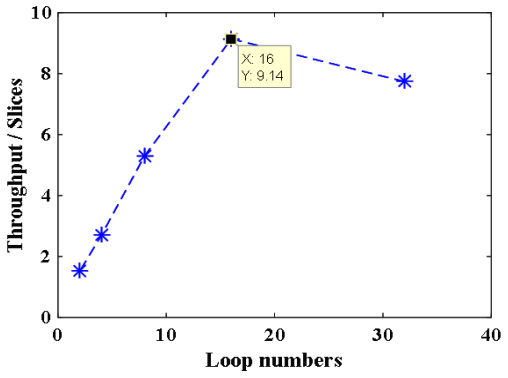
در ادامه برای ارزیابی بهتر، الگوریتم پیشنهادی بر روی تراشه‌های FPGA از خانواده‌های Spartan3 با شماره XC3S50 و Spartan3E با شماره XC3S100E و Virtex7 با شماره XC7VX330T پیاده‌سازی شده و نتایج آن به همراه نتایج پیاده‌سازی چند الگوریتم اخیر که تا کنون ارائه شده‌اند در جدول ۳ برای مقایسه آورده شده است. همانطوریکه مشخص است الگوریتم پیشنهادی دارای بازدهی بالاتر و همین طور سطح مصرفی پایین تر و در نتیجه معیار بازدهی به مساحت بیشتر نسبت روش‌های پیاده‌سازی موجود بر روی تراشه‌های مشترک می‌باشد. علاوه بر این، فرآیند موازی سازی با کاهش تعداد حلقه‌ها از ۳۲ به ۲ حلقه بر روی الگوریتم پیشنهادی به منظور بهبود عملکرد آن و با هدف انتخاب تعداد حلقه تکرار بهینه در نرم افزار ISE انجام شده و نتایج آن در شکل ۱۰ و جدول ۴ آورده شده است.

جدول۳- مقایسه نتایج پیاده سازی الگوریتم پیشنهادی با الگوریتم های دیگر بر روی FPGA.

Method	device	Area (Slices)	Max. Freq. (MHz)	Throughput (Mb/s)	Throughput /Area
Proposed	Spartan3	294	120	480	1.63
	Virtex-7	167	450	1800	10.77
	Spartan3E	295	137	548	1.86
	Spartan6	134	259.4	1037.7	7.74
SPECK 96/96[7]	Virtex-7	130	363	1245	3.24
	Spartan3E	462	111	380	0.61
SPECK 128/128[8]	Spartan3	427	142	263.2	0.616

جدول۴- نتایج پیاده سازی الگوریتم رمز SPECK با اجرای موازی سازی بر روی معماری حلقه به ازای کاهش تعداد حلقه ها در تراشه Spartan6.

Loop numbers	Area (Slices)	Max. Freq. (MHz)	Throughput (Mb/s)	Throughput / Area
32	134	259.44	1037.76	7.74
16	163	186.14	1489.12	9.14
8	349	115.65	1850.4	5.3
4	717	60.41	1933.12	2.69
2	1357	31.84	2037.76	1.51



شکل ۱۰–نمودار تغییر بازدهی به مساحت مصرفی نسبت به تعداد حلقه‌های پیاده سازی شده با روش پیشنهادی.

همانطوریکه مشخص است به ازای حالتی که در آن تعداد حلقه ۱۶ است (اجرای ۲تابع دور و فرآیند بسط کلید متوالی در هر حلقه)، معیار نسبت بازدهی به مساحت بیشترین مقدار را دارد که نشان دهنده عملکرد بهتر الگوریتم رمز نگاری در این حالت است.

۵.نتیجه‌گیری

در این مقاله بعد از معرفی الگوریتم رمز SPECK، روشی برای پیاده‌سازی الگوریتم رمز با نسخه SPECK128/128 بر مبنای ساختار حلقه بر روی FPGA پیشنهاد شده است که در مقایسه با دیگر روش‌های موجود ضمن کاهش سطح مصرفی، نرخ بازدهی را به مقدار مناسبی افزایش داده است. در روش پیشنهادی، ماژول رمز به صورت ساختاری و با استفاده از چند زیر ماژول طراحی و در ادامه با تولید سیگنال‌های کنترلی در قالب یک مدل ماشین حالت، الگوریتم رمز پیاده سازی شده است. علاوه بر این به منظور بهبود عملکرد روش پیشنهادی، یک ساختار موازی نیز به معماری حلقه اضافه شده که در نتیجه آن با کاهش تعداد حلقه‌های اجرای الگوریتم رمز، نرخ بازدهی نیز افزایش یافته است. نتایج پیاده‌سازی الگوریتم پیشنهادی در مقایسه با سایر الگوریتم‌ها نشان دهنده کارایی و عملکرد مناسب آن برای پیاده‌سازی بر روی تراشه FPGA است.

۶. مراجع

1.Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks and L. Wingers, B., (2013) “The simon and speck of lightweight block ciphers,” National Security Agency 9800 Savage Road, Fort Meade, MD 20755, USA.

2.Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B. and Wingers, L., (2015). “SIMON and SPECK: Block Ciphers for the Internet of Things”. IACR Cryptology ePrint Archive, p.585.

3.Aysu, A., Gulcan, E. and Schaumont, P., (2014) “SIMON says: Break area records of block ciphers on FPGAs,” Embedded Systems Letters, IEEE 6(2), pp.37-40.

4.R. Beaulieu, D. Shors, J. Smith , S. Treatman-Clark, B. Weeks, and L. Wingers, (2014)“The SIMON and SPECK block ciphers on AVR 8-bit microcontrollers”. Springer, International Workshop on Lightweight Cryptography for Security and Privacy, pp. 3-20.

5.Maene, P., and Verbauwhede, I.,(2015) "Single-cycle implementations of block ciphers." In International Workshop on Lightweight Cryptography for Security and Privacy, Springer, Cham, pp. 131-147.

6.Diehl, W., Farahmand, F., Yalla, P., Kaps, J.P. and Gaj, K. (2017) “Comparison of hardware and software implementations of selected lightweight block ciphers”. IEEE, 27th International Conference on In Field Programmable Logic and Applications (FPL), pp. 1-4.

7.Park T., Seo, H. and Kim, H., (2016) "Parallel implementations of simon and speck", IEEE International Conference on Platform Technology and Service, pp. 1-6.

8.Nemati A., Feizi, S., Ahmadi, A. and Makki, V., (2015) “A low-cost and flexible FPGA implementation for SPECK block Cipher”. 12th IEEE International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), pp. 42-47.