اولین کنفرانس ملی ایده های نو در مهندسی برق
۱۶ و ۱۷ آذر ماه ۱۳۹۱- دانشگاه آزاد اسلامی واحد خوراسگان- اصفهان
NCNIEE NATIONAL CONFERENCE NEW IDEA IN ELECTRICAL ENGINEERING

طراحی رمزنگار AES-256 به صورت خط لوله و پیاده سازی آن بر روی FPGA

پرهام درّی[۱]، محمد رضا سلطان آقایی[۲]

[۱] دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد نجف آباد، parham.d32@gmail.com
[۲] استادیار مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد خوراسگان، msoltanaghaii@yahoo.com

چکیده – الگوریتم رمزنگاری *AES* یا رایندال (*Rijndael*) یکی از متداولترین الگوریتمهای رمزنگاری استاندارد است. یکی از مشکلات الگوریتم رایندال، متفاوت بودن الگوریتمهای رمزگذاری و رمزگشایی ، و نحوه پیاده سازی آن بر روی *FPGA* است. در این مطالعه، ابتدا انواع قالبهای الگوریتم رایندال مورد بررسی قرار گرفت، سپس مدلی برای پیاده سازی این الگوریتم (شامل قسمتهای رمزگذار و رمزگشا) برروی *FPGA* ارائه خواهد شد که از نظر حجم سختافزار مصرفی و نرخ گذردهی کارآمد باشد. برای رسیدن به این اهداف، پیادهسازی الگوریتم بصورت خط لوله (*pipeline*) برروی *FPGA* انجام گرفت. هر کدام از قسمت های رمزکننده و رمزگشا قادر است چهار قطعه داده متفاوت به همراه چهار قطعه کلید متعلق به هر داده را به طور همزمان دریافت نماید تا عملیات رمزگذاری یا رمزگشایی بر روی دادههای دریافتی به موازات یکدیگر به صورت خط لوله و با استفاده اشتراکی از منابع انجام گیرد. نتایج سنتز رمزکننده و رمزگشای پیشنهادی گویای صحت عملکرد و کارایی مناسب روش پیشنهادی است. همچنین منابع مصرفی کاهش یافته و الگوریتم پیاده سازی شده از سرعت بیشتری برخوردار است.

کلید واژه- امنیت اطلاعات، رمزنگاری، الگوریتم AES، رایندال، FPGA

۱- مقدمه

رمزنگاری (Cryptograghy) نقش مهمی در زمینه امنیت اطلاعات ایفا می‌کند. اطلاعات مهمی که به ناچار باید از در محیط‌های ناامن مبادله و یا ذخیره شوند. الگوریتم‌های رمزنگاری در دو دسته کلی الگوریتم‌های متقارن(کلید پنهان) و الگوریتم-های نامتقارن(کلید عمومی) دسته‌بندی می‌شوند.[۱]. الگوریتم-های رمزنگاری متقارن سریع‌تر بوده و معمولا برای رمزنگاری داده‌های پرحجم و مواقعی که قادریم کلید پنهان را میان کاربران به صورت امن به اشتراک بگذاریم، استفاده می‌شود. در حالی که الگوریتم‌های نامتقارن معمولا برای رمزکردن کلیدهای جلسه و یا امضا، مورد استفاده قرار می‌گیرند.

با شکسته شدن الگوریتم رمزنگاری DES (Data Encryption Standard) در سال ۱۹۹۸، در سال ۲۰۰۱ میلادی الگوریتم رایندال(Rijndeal) به دنبال فراخوانی که NIST (National Institute of Standards and Technology) انجام داد، به عنوان الگوریتم رمزنگاری استاندارد جایگزین تصویب شد[۲،۳]. پس از آن پیاده‌سازی این الگوریتم بر روی سکوهای مختلف آغاز شد. در حالت کلی پیاده‌سازی رایندال به دو دسته پیاده‌سازی نرم‌افزاری و پیاده‌سازی سخت‌افزاری قابل تقسیم است. محدودیت‌های از جمله وابستگی داده‌ها در طول پردازش، در اختیار نداشتن منابع مورد نیاز در هر لحظه و خاصیت تک-پردازنده‌ایی و اجرای پی‌درپی دستورها باعث می‌شوند که نتوان در پیاده‌سازی نرم‌افزاری الگوریتم رایندال به سرعت‌های بالا دست یافت.

اما از آنجا که سرویس‌های محرمانگی از سطح برنامه‌های کاربردی فراتر رفته و در حوزه امنیت بستر نیز گسترش پیدا کرده‌اند، نمی‌توان تنها به پیاده‌سازی نرم‌افزاری بسنده نمود. از این‌رو استفاده از این الگوریتم‌ها در محیط‌های پر سرعتی ازجمله خطوط ارتباطی شبکه‌ها، سیستم‌های VPN و مسیریاب‌ها به یک چالش جدی پژوهشی تبدیل شده است[۴]. در این راستا در

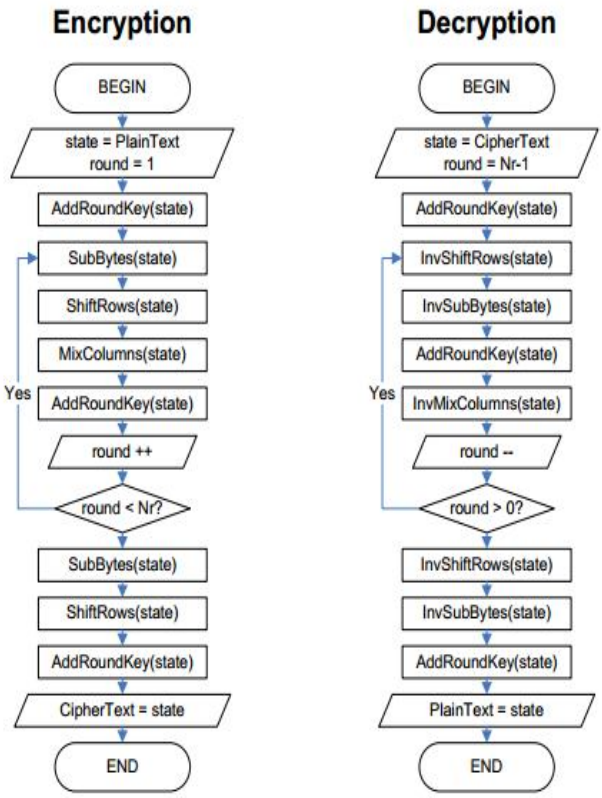
کاربردهای پرسرعت می‌توان از شتاب دهنده‌های سخت-افزارهای(Hardware Accelerator) استفاده کرد. در طراحی این شتاب دهنده‌ها معمولا از سخت‌افزارهای قابل برنامه-ریزی(Reconfiguration Hardware) استفاده می‌شود.

این سخت‌افزارها در خانواده‌های مختلفی قابل دسترسی هستند که معروف‌ترین آنها خانواده‌ی FPGA (Field Programmable Gate Array) ها است. برای پیاده‌سازی سخت‌افزار بر روی FPGAها، می‌توان عملکرد سخت-افزار را قبل از پیاده‌سازی فیزیکی، شبیه‌سازی(Simulation) نمود.

در این مقاله پس از مقدمه، ابتدا مروری بر ساختار و معماری الگوریتم رایندال آورده خواهد شد، پس از آن در بخش ۳ مدلی برای پیاده سازی قست رمزکننده و مدلی برای پیاده سازی قست رمزگشا پیشنهاد و مزیت این دو مدل ارائه می‌گردد و در انتها نیز به‌عنوان نتیجه‌گیری، مطالب مقاله مرور می‌گردند.

۲- ساختار الگوریتم رایندال

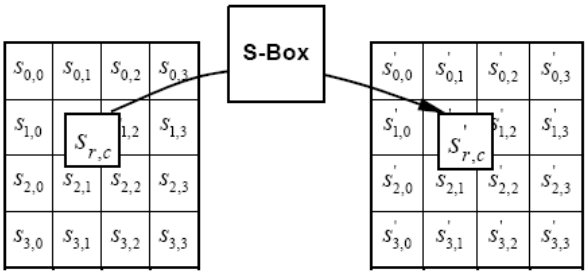
رایندال یک الگوریتم رمز قطعه‌ای متقارن با طول قالب داده ۱۲۸، ۱۹۲ و ۲۵۶ بیت است طول کلید نیز مستقل از طول قالب، ۱۲۸،۱۹۲یا ۲۵۶ بیت می‌باشد. الگوریتم بسته به طول قالب داده و طول کلید مشتمل بر ۱۰، ۱۲ یا ۱۴ دور خواهد بود.[۱۰] رایندال دارای ساختاری برای بسط کلید است که از روی کلید اصلی بسته به تعداد دورها، تعدادی زیر کلید تولید می‌کند که در هر دور به قالب داده اضافه می‌شوند. الگوریتم شامل سه تبدیل مهم MixColumn() و ShiftRow() و SubByte() است که اولی یک تابع جایگزینی غیر خطی و تامین کننده امنیت سیستم و دومی وسومی توابعی خطی برای افزایش گسترش و اختلاط الگوریتم‌اند. در این رمز قطعه‌ای ساختار سیستم رمزگشا دقیقا مشابه سیستم رمزگذار نیست. هم چنین چون با افزایش طول کلید تعداد دورهای الگوریتم افزایش می‌یابد، زمان اجرا و سرعت الگوریتم به طول کلید وابسته است. ساختار الگوریتم رایندال برای عملیات رمزگذاری و رمزگشایی در شکل1 نشان داده شده است. در ادامه واحدهای تشکیل دهنده این الگوریتم مرور می-شوند.



شکل۱: فلوچارت الگوریتم رمزگذاری و رمزگشایی AES در قیاس با هم[۶]

۲-۱- واحدهای SubByte و InvSubByte

واحد SubByte با استفاده از جدول SBox (SubstitutionBox) مقادیر ورودی را با مقادیر جدیدی جانشین میکند. وظیفه این واحد انجام عملیات غیرخطی بر روی داده‌های جهت امنیت سیستم است. شکل۲ گویای این عملیات است. برای پیاده سازی واحد SBox یا معکوس آن سه روش پیش‌رو داریم که در ادامه به مقایسه آن‌ها جهت انتخاب مناسب-ترین روش خواهیم پرداخت.



شکل۲: عملکرد SubBytes[۶]

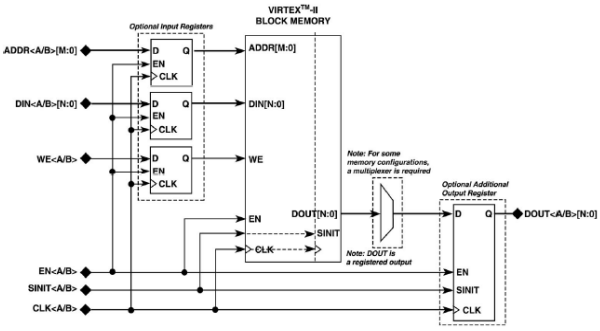
۲-۱-۱- پیاده‌سازی حافظه S-Box به صورت ترکیبی

در این طراحی تاخیر S-Box با تاخیر ۳ واحد دیگر در یک دور جمع شده و فرکانس پالس ساعت را کاهش می‌دهند ولی امکان انجام هر دور در یک پالس ساعت را فراهم می‌کند که در واقع تفاوت اصلی دو طرح عمده پیاده‌سازی AES نیز درهمین واحد است.

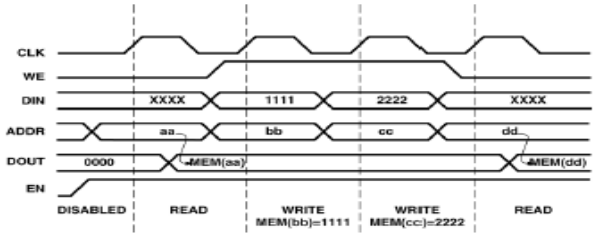
به این ترتیب یک S-Box ترکیبی دارای تاخیر حدود ۶ ns است و همین تاخیر باعث می‌شود تا فرکانس پالس ساعت در نهایت برای این طرح از ۱۰۷ MHz افزایش نیابد. حجم یک S-Box برابر با ۱۲۸ LUT(Look Up Table) است. توجه کنید که SubBytes نیاز به ۱۶ عدد S-Box دارد و در تولید کلید نیز ۴ عدد مصرف شده است که در مجموع AES نیاز به ۲۰ عدد S-Box دارد. در نتیجه این طرح باعث می‌شود که حجم کلی مدار نیز بسیار زیاد شود.

۲-۱-۲- پیاده‌سازی حافظه S-Box با استفاده از ROM

در این طرح از بلوک‌های داخلی حافظه در FPGA استفاده شده است. در شکل۳ ساختار یک بلوک داخلی حافظهٔ FPGA آورده شده است. بلوک‌های داخلی حافظهٔ در FPGA دارای خصوصیات زمانبندی ویژه‌ای هستندکه در شکل۴ آورده شده است.



شکل۳: ساختار یک بلوک داخلی حافظه FPGA



شکل۴: زمانبندی بلوک حافظه داخلی

یعنی یک پالس ساعت بین آدرس و داده خوانده، شده فاصله است، که اگر آدرس و داده خوانده شده را رجیستر کنیم در مجموع سه پالس ساعت مصرف می‌شود. در بخش نتایج پیاده‌سازی نیز به این مطلب پرداخته شده است.

با استفاده از حافظه‌های دارای دو ورودی و دو خروجی(Dual-Port Block RAM) هر بلوک حافظه داخلی شامل دو عدد S-Box است. به این ترتیب که ۸ بیت ورودی به S-Box به عنوان آدرس به این حافظه وارد می‌شود و محتوای این حافظه همان مقادیر جانشینی است. بنابراین مقدار خوانده شده از این حافظه همان خروجی مورد انتظار از S-Box است.

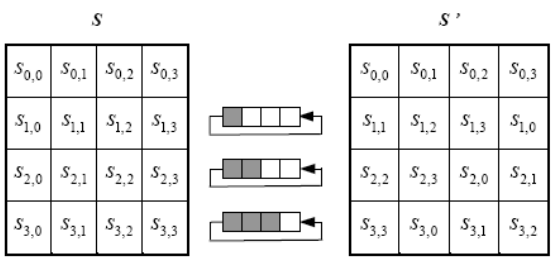
۲-۱-۳- پیاده‌سازی حافظه S-Box با استفاده از عبارت ریاضی

امکان پیاده‌سازی S-Box با انجام عمل معکوس‌گیری و اعمال تبدیل مستوی به صورت سخت‌افزاری وجود دارد. مشکل این روش انجام عملیات معکوس‌گیری است. در این طرح عملیات معکوس‌گیری در $GF(2^8)$ به معکوس‌گیری در $GF(2^4)$ به همراه تعدادی عملیات ضرب و جمع در $GF(2^4)$ کاهش می‌یابد.

مدار لازم برای انجام این طرح، پیاده‌سازی شد ولی حجم و نرخ دادهٔ آن از دو طرح فوق بر روی FPGA بدتر شد. ظاهراً علت آن است که بلوک‌های پایه‌ای FPGA برای عملیات دروازه‌های منطقی مورد استفاده در این طرح بزرگ هستند.[۵]

۲-۲- واحدهای ShiftRow و InvShiftRow

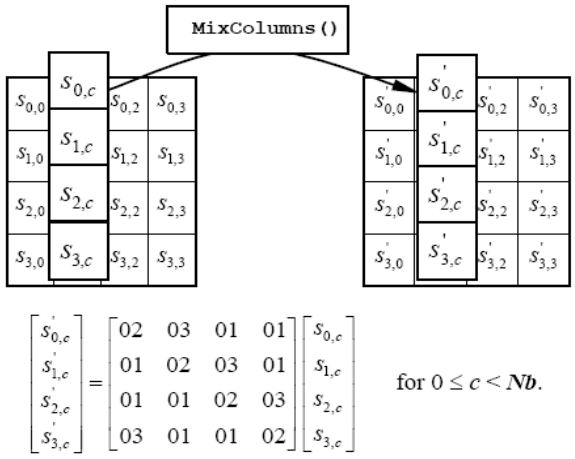
عملکرد این واحد بسیار ساده بوده و تنها سطرهای قالب را مطابق با شکل۵ شیفت گردشی می‌دهد. برای این واحد یک برنامه مجزا نوشته شده است که عمل ساده فوق را به آسانی و با حجم کم و نرخ دادهٔ زیاد انجام می‌دهد.



شکل۵: عملکرد ShiftRows[۶]

۲-۳- واحدهای MixColumns و InvMixColumns

عملکرد این واحد بر اساس ضرب یک ۸ بیتی از قالب در حال پردازش در اعداد ثابت ۲ یا ۴ بیتی در میدان $GF(2^8)$ بنا شده است که در شکل۶ ماتریس انتقال عملیات رمزگذاری آمده است.

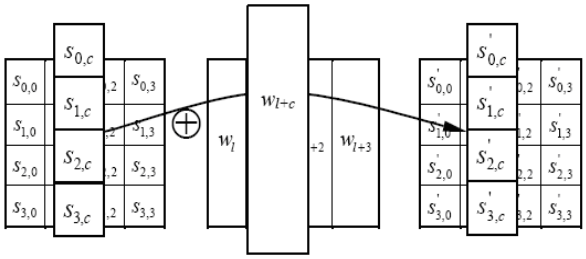


شکل ۶ : عملکرد MixColumns[۶]

برای پیاده‌سازی، این واحد به چهار ضرب ستونی موازی شکسته شده است که در هر ضرب ستونی لازم است هر بایت در اعداد ۲ و ۳ ضرب شود یعنی ۸ ضرب در هر ضرب ستونی و چهار ضرب ستونی برای MixColumns (۳۲ ضرب بایت در ۲ بیتی در $GF(2^8)$).[۶،۵،۹]

۲-۴- واحد AddRoundKey

این نیز یکی از واحدهای ساده در یک دور است زیرا تنها زیر کلید هر دور را با قالب آن دور XOR (جمع در پیمانه ۲) می‌کند. شکل ۷ چگونگی انجام این عمل را نشان می‌دهد.



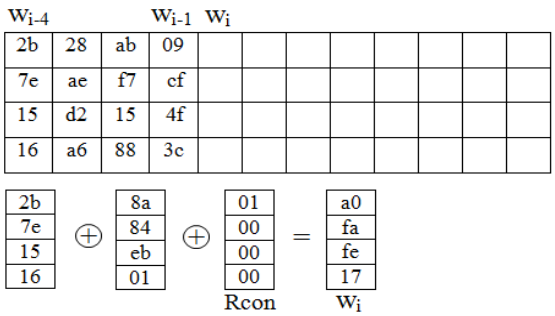
شکل۷: عملکرد AddRoundKey[۶]

مدار لازم برای آن ۱۲۸ بیت قالب هر دور را با بیت متناظر زیر کلید ۱۲۸ بیتی، XOR می‌کند.

۲-۵- واحد KeyExpansion

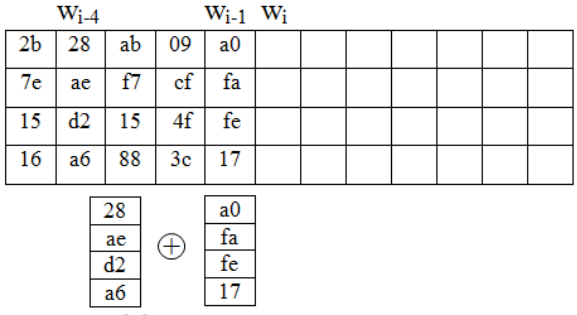
در این زیر واحد عملیات پردازش کلید و تولید زیرکلیدها انجام می‌شود. همانطور که در شکل ۱۰ آورده شده است مراحل تولید زیر کلید دور فعلی از زیر کلید دور قبل به صورت زیر است:

- ۳۲ بیت آخر زیر کلید قبل (w_{i-1}) یک بایت شیفت گردشی به راست داده می‌شود.
- ۳۲ بیت حاصل از S-Box عبور می‌کند.
- سپس ۳۲ بیت حاصل با w_{i-4} و Rcon همان دور XOR می‌شود تا w_i حاصل شود. Rcon عدد هشت بیتی مشخص برای هر دور است.



شکل۸: تولید کلمه اول زیر کلید[۸]

برای تولید سه کلمه دیگر زیر کلید یک دور، w_{i-1} با w_{i-4} بسادگی XOR شده و w_i جدید حاصل می‌شود. که w_i یک ۳۲ بیتی از زیر کلیدها بطور متوالی است یعنی کلید ورودی w_0-w_3 را تولید می‌کند و زیر کلیدها بترتیب سایر w_iها را تولید می‌کنند.



شکل۹ : تولید سایر کلمه‌های زیر کلید[۸]

در طراحی پردازشگر زیر کلید نیز بعلت وجود S-Box دو طرح انجام هر دور در یک پالس ساعت و سه پالس ساعت پیاده‌سازی شده است. زیرا S-Box را می‌توان به صورت ترکیبی و در یک پالس ساعت پیاده‌سازی کرد و یا با استفاده از بلوک‌های حافظهٔ داخلی و در سه پالس ساعت پیاده نمود.

۳- ارائه معماری پیشنهادی

یک ایده‌ی مناسب برای دستیابی به سرعت‌های بالا استفاده از معماری خطلوله در پیاده‌سازی الگوریتم است. در این روش واحد سخت‌افزاری پیاده‌سازی شده برای هر دور را به تعداد دورها تکرار می‌کنند تا خروجی تولید شده در هر دور را به دور بعد انتقال دهند. در این‌گونه پیاده سازی، حجم سخت‌افزار مصرفی به‌طور چشم‌گیری افزایش پیدا می‌کند.[۷]

در این بخش معماری پیشنهادی که مبتنی بر معماری خط لوله است، ارائه خواهد شد. در این طرح هر کدام از قسمت های رمزکننده و رمزگشا قادر است چهار قطعه داده متفاوت ۱۲۸ بیتی به همراه چهار قطعه کلید ۲۵۶ بیتی متعلق به هر داده را به طور همزمان دریافت نماید تا عملیات رمزگذاری یا رمزگشایی بر روی داده‌های دریافتی به موازات یکدیگر به صورت خط لوله و با استفاده اشتراکی از منابع انجام گیرد.

۳-۱- پیشنهاد مدل رمزکننده مبتنی بر معماری خط-لوله

جهت چهار برابر کردن توانایی موتور رمزکننده، یک روش پیشنهادی می‌تواند اینگونه باشد که، یک موتور رمز معمولی را به صورت متوالی چهار بار استفاده شود، اما این روش ساده‌ترین و در عین حال بدترین شیوه ممکن است، چرا که انجام چهار عمل به صورت متوالی نیاز به زمان اجرای چهار برابر دارد، روش پیشنهادی دیگر می‌تواند اینگونه بیان شود که، از چهار موتور رمزکننده به صورت همزمان استفاده شود، این روش نیز غیر منطقی است، چرا که در این صورت از منابع به صورت چهار برابر استفاده می‌کنیم، ترجیح می‌دهیم با استفاده از روش خطلوله و استفاده بهینه Sliceها به سرعتی بهتر دست یابیم. حالتی بهینه در استفاده از منابع و سرعت.

برای نیل به این هدف، دور اول عملیات رمزگذاری برای هر یک از چهار داده رمزنشده ورودی و کلیدهای متناظرشان اجرا شده و نتایج حاصل داخل خطلوله می‌شوند، سپس پس از اتمام دور اول رمزگذاری هر چهارداده ورودی، با استفاده از خروجی-های خطلوله دور دوم رمزگذاری داده ها شروع شده و مجددا نتایج حاصل وارد خطلوله می‌شوند، و این روند ادامه می‌یابد تا عملیات رمزگذاری چهار داده به صورت همزمان (با تفاوت زمانی یک کلاک) به اتمام برسد. همین روند دقیقا باید در موتور ساخت زیرکلید از روی کلید اصلی اعمال شود تا داده‌ها با کلیدهای متناظر خود رمز شوند.

جدول۱: نتایج مدل پیشنهادی و مدل اولیه

رمزکننده	xc4vlx25			
	موتور اولیه		موتور جدید	
	Used	Utilization	Used	Utilization
Number of Slice Flip Flops	۱۱۷۴	۵٪	۲۲۸۱	۱۰٪
Number of LUTs	۳۷۳۹	۱۷٪	۱۱۶۶۰	۵۴٪
Number of Occupied Slices	۲۴۸۹	۲۳٪	۶۳۸۶	۵۹٪
Number of FIFO/RAMB	۸	۱۱٪	۴	۵٪
تعداد کلاک مصرفی جهت کامل شدن عملیات رمزگذاری	۱۷	-	۶۷	-

نتایج جدول۱ تا مرحله Place&Route می‌باشد. با مقایسه نتایج این جدول به این نتیجه می رسیم که طراحی که انجام داده‌ایم بهینه بوده، هرچند که تعداد کلاک مصرفی بهینه نشده و تقریبا چهاربرابر شده‌است. (موتور جدید ۴ برابر موتور اولیه است و همزمان ۴ داده متفاوت را رمز می کند نه یکی) تعداد slice flip flop های مصرفی ۲ برابر شده نه ۴ برابر، و تعداد slice های مصرف شده از ٪۲۳ به ٪۵۹ رسیده که این میزان کمتر ۳ برابر است.

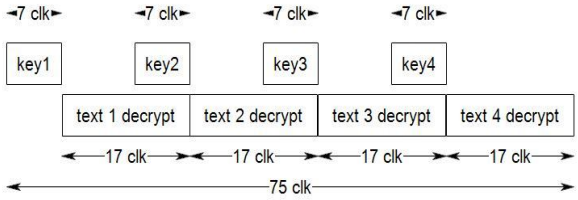
۳-۲- پیشنهاد مدل رمزگشا

جهت طراحی رمزگشا به صورت چهار برابر، از روش‌های متفاوتی میتوان بهره برد، مثلا استفاده از خط لوله، مانند آنچه که در طراحی رمزکننده استفاده شد. اما با کمی بررسی بیشتر در ماهیت روش رمزگشایی، به این نتیجه می‌رسیم که روش بهینه-تری جهت نیل به این مقصود وجود دارد، روشی که شاید در آغاز ابتدایی به نظر برسد، اما منجر به پیاده‌سازی بهینه تری نسبت به خطلوله می‌شود. لازم به ذکر است که این روش را برای رمزکننده نمی‌توان استفاده کرد.

علت این امر آن است که در موتور رمزکننده با تولید کلید هر دور، می‌توانیم عملیات رمزگذاری را انجام دهیم و لازم نیست

ابتدا تمامی کلید دورها را تولید کرده و سپس موتور شروع به رمزگذاری کند، اما در الگوریتم رمزگشایی به این علت که معکوس عملیات رمزگذاری صورت می‌گیرد، ابتدا باید تمامی کلید دورها از روی کلید اصلی تولید شوند، و به صورت معکوس تولیدشان به موتور اعمال شوند تا موتور رمزگشا بتواند عملیات رمزگشایی را آغاز کند. یعنی در عملیات رمزگشایی، آخرین کلید تولید شده برای اولین مرحله رمزگشایی و اولین کلید تولید شده برای آخرین مرحله رمزگشایی مصرف می‌شود. پس واضح است که استفاده از روش خطلوله جهت پیاده‌سازی رمزگشا منجر به کارایی بهتر نمی‌شود.

برای رسیدن به نتیجه مطلوب کافی است کنترلری طراحی شود، تا به صورت بهینه از موتور رمزگشا استفاده کرده و هیچگاه موتور رمزگشا بیکار نماند. شکل۱۲ این مطلب را به خوبی بیان می‌کند.



شکل ۱۰: روند عملیات انجام شده در ماژول رمزگشا

همانگونه که در شکل۱۰ نشان داده شده است در این روش ابتدا کلید های مربوط به ورودی اول پس از 7clk تولید شده پس از کامل شدن مرحله ساخت کلیدهای دور برای اولین کلید اصلی، سریعا داده ی اول (text1) شروع به رمزگشایی توسط موتور رمزگشا می‌شود، سپس در ۷ کلاک باقی مانده تا اتمام عملیات رمزگشایی اولین داده کلیدهای دور مربوط به داده دوم از روی کلید اصلی دوم ساخته می‌شوند، و پس از اتمام همزمان رمزگشایی داده اول و ساخت کلید دوم، عملیات رمزگشایی برای دومین داده شروع می‌شود، و به همین منوال عملیات ادامه می-یابد. در نهایت طی چهار کلاک سیگنال done فعال شده و مقادیر رمز گشایی شده هر کلید به مدت یک پالس ساعت روی باس خروجی قرار داده می‌شوند.

ماژول ساخت کلید از روی کلید اصلی، در طول اجرا ۸ کلید ۲۵۶ بیتی تولید می‌کند که برای افزایش سرعت کلیدها دوتا دوتا تولید خواهدشد، پس این ۸ کلید ۲۵۶ بیتی در واقع ۱۶ کلید ۱۲۸ بیتی‌اند که تنها ۱۵ کلید ابتدایی معتبر و آخرین کلید نامعتبر است(الگوریتم aes-256 ۱۴ کلید دور مازاد بر کلید اصلی تولید می‌کند).

همانگونه که گفته شد لازم است تا این کلیدها ذخیره شوند و سپس به صورت معکوس به موتور رمزگشا اعمال شوند تا عملیات رمزگشایی به درستی انجام شود. برای همین منظور سه حافظه t0 و t1 و t2 را تعریف می‌کنیم.

جهت استفاده بهینه از منابع سیستم و کاهش Slicهای مصرفی این حافظه ها را به صورت ۲۵۶×۴ تعریف شده که نحوه پر و خالی شدن آنها در جدول۲ نشان داده شده است.

همانگونه که در جدول۲ نشان داده شده، با تولید کلید های دور از روی کلید اصلی، ابتدا نتایج در t0 و سپس در t1 نوشته می شوند (شماره های ۱ و ۲ بیان کننده تقدم و تاخر در نوشتن داده ها است، و علامت + نشان دهنده نوشتن داده روی حافظه و علامت - نشان دهنده خواندن داده از حافظه است)، سپس هنگامی که تمامی کلیدهای دور کلید اصلی اول ساخته شدند جهت استفاده در موتور رمزگشا ابتدا مقادیر t1 به صورت معکوس و سپس مقادیر t0 به صورت معکوس خوانده می‌شوند، همانطور که با کمی دقت مشاهده می‌شود هنگامی که کلیدهای دور مربوط به کلید اصلی دوم در حال نوشته شدن بر روی حافظه t1 است، قبلا مقادیر معتبر از این حافظه خوانده شده و مصرف شده‌اند، و در همین حین مقادیر معتبری از t0 خوانده مشود که شامل اولین کلید های فرعی کلید اصلی اول است. و این عملیات برای باقی کلیدها و داده‌ها به همین صورت تکرار می‌شود.

جدول ۲: ترتیب ذخیره و مصرف زیرکلیدها

Text in4 dec		Text in3 dec		Text in2 dec		Text in1 dec			256*4bit memory
		Key4 generate		Key3 generate		Key2 generate		Key1 generate	
-		+1	-	+2		-		+1	T0
	-	+2		-		+1	-	+2	T1
		-		+1	-	+2			T2

همانگونه که در شکل۱۲ و توضیحات جدول ۲ بیان شد این روش در ایده‌آل‌ترین حالت خود قرار دارد چرا که موتور رمزکننده هیچگاه بی‌کار نمی‌ماند. و از طرفی برای ذخیره‌سازی کلیدهای دور از کمترین حافظه ممکن استفاده شده است.

در جدول۳ نتایج حاصل از پیاده‌سازی موتور رمزگشای ارائه شده، به همراه موتور اولیه که تنها قادر به رمزگشایی یک داده ۱۲۸بیتی به همراه کلید۲۵۶ است، آورده شده. با مقاسیه این داده ها بهینه بودن مدل پیشنهادی آشکار است.

جدول۳: نتایج مدل پیشنهادی و مدل اولیه

رمزگشا	xc4vlx25			
	موتور اولیه		موتور جدید	
	Used	Utilization	Used	Utilization
Number of Slice Flip Flops	۶۷۶	۳٪	۵۵۶۵	۲۵٪
Number of LUTs	۲۶۴۸	۱۲٪	۹۰۵۱	۴۲٪
Number of Occupied Slices	۱۴۸۳	۱۳٪	۵۹۴۲	۵۵٪
Number of FIFO/RAMB	۱۰	۱۳٪	۱۰	۱۳٪
تعداد کلاک مصرفی جهت کامل شدن عملیات رمزگشایی	۲۵	-	۷۹	-

نتایج جدول۳ تا مرحله Place&Route می‌باشد. با مقایسه نتایج این جدول مشاهده می‌شود که میزان Sliceهای مصرفی تقریبا چهار برابر شده و این امر به واسطه ماهیت این الگوریتم است، اما از طرفی توانستیم سرعت انجام عملیات را به میزان ۲۱٪ بهبود داده و کل داده‌ها تنها با طی ۷۹ کلاک رمزگشایی می‌شوند.

این طرح بر روی تراشه‌ی xc4vlx25 از خانواده Xilinx در نرم‌افزار ISE 13.4 پیاده‌سازی شده است. شبیه‌سازی مدار پیشنهادی در نرم‌افزار ModelSim و همچنین سنتز آن نیز در نرم‌افزار ISE 13.4 انجام گرفته است.

تراشه‌ی XC4VLX25 دارای ۷۲ واحد حافظه بلوکی، 168 Kb حافظه توزیع شده، ۱۰۷۵۲ واحد Slice، ولتاژ 1.20 DC و فرکانس 400MHz و درجه سرعت (Speed Grade) ۱۰ است.

۴- نتیجه‌گیری

هدف اصلی این مطالعه، بررسی پیاده‌سازی الگوریتم رایندال بر روی FPGA بود. برای این منظور پس از مرور این الگوریتم، معماری جدیدی جهت پیاده‌سازی آن ارائه شد و در نهایت مقایسه‌ایی بر اساس نتایج حاصل از پیاده‌سازی این معماری و مدل اولیه صورت گرفت. در معماری پیشنهادی، مشاهده گردید که در موتور رمزکننده موجب کاهش منابع مصرفی و در موتور رمزگشا موجب بهبود در سرعت عملیات شد. بر طبق نتایج حاصل، بیشترین فرکانس ساعت رمزکننده برابر با 85.12MHz و برای رمزگشا برابر با 65.53MHz بدست آمد. در ادامه تحقیق، پیشنهاد میشود برای دستیابی به بالاترین امنیت رمزنگاری، علاوه بر استفاده از این الگوریتم، از الگوریتم توزیع کلید کوانتومی QKD(Quantum Key Distribution) بصورت ترکیبی استفاده شود.

سپاسگزاری

اینجانب پرهام درّی بر خود لازم می‌دانم از جناب آقای دکتر سلطان آقایی که صبورانه راهنمای من بودند تشکر کنم.

مراجع

[1] Schneier, B., “Applied Cryptography: Protocols, Algorithms, and Source Code in C,” John Wiley & Sons, 1996.
[2] Nechavatal, J., “Report on the Development of Advanced Encryption Standard (AES),” NIST, October 2000.
[3] Daemen, J. and Rijmen, V., “The Rijndeal Block Cipher: AES Proposal,” First AES Candidate Conference (AES1), August 1998, pp.20-22.
[4] Hodjat, A., “Area-Throughput Trade-Offs for Fully Pipelined 30 to 70 Gbit/sec AES Processors,” IEEE Transactions on Computers, VOL. 55, NO.4, April, 2006.
[5] Daemen, J. and Rijmen, V., “AES Proposal: The Rijndael Block Cipher. Version 2,” September, 1999, available at *http://www.nist.gov/aes/*
[6] M. Khalil, and M. Hani, “Verilog Design of A 256-Bit AES Crypto Processor Core,” pp. 12-31, 2007.
[7] McLoone, M. and McCanny, J.V., “Rijndael FPGA Implementations Utilizing Look-Up Tables,” Journal of VLSI Signal Processing Systems, pp.261-275, 2003.
[8] Forouzan, B. A., “Cryptography & Network Security,” McGraw-Hill, Inc ,part 1-7, 2008.
[9] Stallings, W., “ Cryptography and Network Security: Principles and Practice,” Prentice Hall Press, 2010.
[۱۰] مشارکت‌کنندگان ویکی‌پدیا. «استاندارد رمزنگاری داده‌ها» ویکی‌پدیا، دانشنامهٔ آزاد، ۲۷ اکتبر ۲۰۱۲، وب. ۲۰ نوامبر ۲۰۱۲.