طراحی و پیاده‌سازی رمزنگار AES در بستر FPGA برای خطوط پرسرعت

پرهام درّی[۱]، دانشجوی کارشناسی ارشد؛ علی قیاسیان[۲]، استادیار؛ حسین سعیدی[۳]، استاد

۱- دانشکده مهندسی کامپیوتر - دانشگاه آزاد اسلامی واحد نجف‌آباد - نجف‌آباد - ایران - parhamdorri@sarv.net
۲- گروه برق - دانشکده فنی و مهندسی - دانشگاه شهرکرد - شهرکرد - ایران - ghiasian.ali@eng.sku.ac.ir
۳- دانشکده مهندسی برق و کامپیوتر - دانشگاه صنعتی اصفهان - اصفهان - ایران - hsaidi@cc.iut.ac.ir

چکیده: الگوریتم رمزنگاری AES یکی از متداول‌ترین الگوریتم‌های رمزنگاری متقارن است. به‌علت قابلیت‌های این الگوریتم، آن را می‌توان بر روی بسترهای مختلفی ازجمله بـر روی بسـترهای سخـت‌افزاری نظیر FPGA پیاده‌سازی کرد. همچنین به‌علت ساختار الگوریتم می‌توان مسیر داده را به‌صورت چرخشی و یا غیر چرخشی پیاده‌سازی نمود. ازآنجاکه بسته به کاربرد، استفاده از هریک از این دو معماری تأثیر فراوانی بر میزان گذردهی و میزان منابع مصرفی دارد، می‌بایست در طراحی توازنی میان این دو عامل همواره متناقض بـرقرار شود. همچنین ازآنجاکه در این الگوریتم قسمت S-Box بخش بحرانی جهت دستیابی بـه این اهـداف است، ایـن مقاله به ارائه یک مدار ترکیبی به‌منظور پیاده‌سازی S-Box استفاده‌شده در تبدل جای‌گشت بایت در الگوریتم AES و همچنین طراحی مسیر داده در این الگوریتم به‌صورت غیر چرخشی و با استفاده از تکنیک خط‌لوله می‌پردازد. نتایج حاصل در مرحله Place & Route نشان می‌دهد که معماری ارائه‌شده در این مقاله به‌میزان ۳۶۶۹ slices مصرف کرده و با بیشترین فرکانس پالس ساعت ۵۷۰/۷۷۶ MHz قادر است عمل کند بنابراین به گذردهی ۷۱/۳۵ Gbps دست می‌یابد. این نتایج بر روی Virtex 7 FPGA (xc7v585t -3ff1157) و با استفاده از نرم‌افزار Xilinx ISE 14.2 به‌دست آمده است.

واژه‌های کلیدی: رمزنگاری، الگوریتم AES، بسترهای سخت‌افزاری، FPGA.

Design and Implementation of AES Encryption Engine on FPGA for High-speed Links

Parham Dorri[1], MSc Student; Ali Ghiasian[2], Assistant Professor; Hossein Saidi[3], Professor

1- Faculty of Computer Engineering, Islamic Azad University of Najafabad Branch, Najafabad, Iran, Email: parhamdorri@sarv.net
2- Electrical Department, Faculty of Engineering, Shahrekord University, Shahrekord, Iran, Email: ghiasian.ali@eng.sku.ac.ir
3- Faculty of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan, Iran, Email: hsaidi@cc.iut.ac.ir

Abstract: Advanced Encryption Standard (AES) is one of the most common standard encryption algorithms. Inspired by its characteristics, AES algorithm can be implemented on various hardware platforms such as FPGA. Also, the data path can be implemented in either loop-unrolling or rolling architecture. These two architectures have direct impact on the amount of area consumption on the chip as well as system throughput. Then, a smart design should be able to consider the trade-off between area and throughput and provide a good balance between these two conflicting factors. In this paper, we propose such a design to represent the area-throughput trade-off for FPGA implementation of the AES algorithm. With loop unrolling and pipelining techniques, throughput of 71.35 Gbps is achievable in Virtex 7 FPGA (xc7v585t-3ff1157). This design has just used 3669 Slices on the chip. The extracted results from the Place & Route report of Xilinx ISE 14.2 indicates that the maximum attainable clock frequency is 570.776 MHz.

Keywords: Encryption, AES algorithm, Hardware platforms, FPGA.

تاریخ ارسال مقاله: ۱۳۹۳/۰۶/۲۶
تاریخ اصلاح مقاله: ۱۳۹۳/۰۸/۰۳ و ۱۳۹۳/۰۹/۱۶
تاریخ پذیرش مقاله: ۱۳۹۳/۱۲/۱۳
نام نویسنده مسئول: علی قیاسیان
نشانی نویسنده مسئول: ایران - شهرکرد - بلوار رهبر - دانشگاه شهرکرد - دانشکده فنی و مهندسی - گروه برق.

۱- مقدمه

امروزه به‌واسطه نیاز جهت ایجاد امنیت برای اطلاعات مهم و سری، الگوریتم‌های رمزنگاری[۱] جایگاه ویژه‌ای پیدا کرده‌اند. با استفاده از این الگوریتم‌ها می‌توان امنیت اطلاعات مهمی را که به‌ناچار در محیط‌های ناامن مبادله و یا ذخیره شوند تضمین کرد. "رمزنگاری عبارت است از یک «نظام» یا الگوی «ریاضی/منطقی» که بر اساس آن اطلاعات و مفاهیم آشکار و قابل فهم برای همگان، طبق روالی برگشت‌پذیر به اطلاعاتی نامفهوم و گنگ تبدیل می‌شود [۱]." انتخاب الگوریتم مناسب جهت رمزنگاری داده‌ها، مهم‌ترین عامل در ایجاد امنیت یک سیستم است. در استفاده از الگوریتم‌های رمزنگاری، همواره این نکته باید در نظر گرفته شود که محرمانگی یک الگوریتم مزیتی برای آن الگوریتم ناست. صرف‌نظر از تئوری ریاضی که یک الگوریتم بر مبنای آن تولید و به‌واسطه آن قدرتمند شده است، آن الگوریتم‌هایی بهتر هستند که مشهور بوده و به‌خوبی مستندسازی شده‌اند، زیرا که به‌خوبی مطالعه و آزمایش شده و حتی تحت حملات گوناگون قرار گرفته‌اند. بنابراین در این پژوهش به الگوریتم AES[۲] (Rijndael) پرداخته شده است، زیرا این الگوریتم از زمان ابداع تابه‌حال توسط مراجع بزرگی به‌خوبی مورد آزمودن، بررسی و پیاده‌سازی قرارگرفته است.

پس از شکسته شدن الگوریتم رمزنگاری DES[۳] در سال ۱۹۹۸، در سال ۲۰۰۱ میلادی الگوریتم رمزنگاری رایندال[۴] به دنبال فراخوانی NIST[۵] در سال ۲۰۰۱ میلادی به‌منظور یافتن الگوریتم استاندارد، به‌عنوان الگوریتم رمزنگاری جایگزین انتخاب شد. پس‌ازآن طراحان به پیاده‌سازی این الگوریتم بر روی بسترهای مختلف سخت‌افزاری و نرم‌افزاری پرداختند. در پیاده‌سازی‌های نرم‌افزاری همواره عوامل محدودکننده‌ای وجود دارند که مانع از دستیابی به سرعت‌های بالا در این نوع از پیاده‌سازی‌ها می‌شوند. وابستگی داده‌ها در طول پردازش، زمان‌بندی و اشتراک منابع میان پردازش‌ها و نخ‌ها، خاصیت تک‌پردازنده‌ای در اغلب سیستم‌ها و اجرای پی‌درپی دستورها ازجمله این محدودیت‌ها می‌باشند. به همین علت، بیش از ده سال است که اکثر مطالعات در زمینه مهندسی رمزنگاری، بر روی بهینه‌سازی هسته‌های رمزنگار سخت‌افزاری جهت دستیابی به بیشترین بازدهی متمرکز شده‌اند. از طرفی مقدار منابع سخت‌افزاری مصرفی نیز عامل بسیار مهمی جهت دست‌یابی به بهره‌وری در هزینه‌ها و کاهش توان مصرفی است.

جهت دست‌یابی به این اهداف، راهکارهای زیادی برای طراحی منابع رمزنگاری ارائه شد. این راهکارها در بالاترین سطح انتزاع به تصمیم‌گیری بر سر بستر پیاده‌سازی و فضای طراحی جهت پیاده‌سازی منابع رمزنگاری می‌پردازند. به‌منظور پیاده‌سازی الگوریتم‌ها در بستر سخت‌افزاری، طراحان انتخاب‌هایی ازجمله: استفاده از پردازنده‌های همه‌منظوره (GPP[۶])، طراحی بر روی سخت‌افزارهای قابل بازپیکربندی مجدد FPGA[۷] و یا طراحی با استفاده از مدارهای مجتمع ASIC[۸] را پیش رو دارند. استفاده از پردازنده‌های همه‌منظوره به‌علت اجرای ترتیبی دستورالعمل‌ها و نیز ثابت و محدود بودن پهنای باند داده، برای کاربردهایی با کارایی بالا مناسب نیست، درحالی‌که سخت‌افزارهایی چون ASIC و FPGA چنین محدودیت‌هایی ندارند.

به‌منظور افزایش کارایی، دست‌یابی به حداکثر میزان گذردهی و همچنین کاهش میزان مصرف توان و کاهش منابع سخت‌افزاری مصرفی، طراحان می‌توانند از تکنولوژی ASIC جهت پیاده‌سازی طرح نهایی بر روی سخت‌افزار استفاده کنند. لازم به ذکر است که با استفاده از این تکنولوژی می‌توان به بهترین عملکرد دست یافت، اما این روش برای کاربردهای محدود دارای هزینه بالایی بوده و توجیه اقتصادی ندارد. همچنین این روش نیاز به مدیریت دقیق منابع سخت‌افزاری مانند انتخاب پهنای باند داده، ثبات‌ها، اندازه حافظه و مواردی از این قبیل دارد، زیرا که در این روش طراحی به‌صورت hardwired structure بوده و قابل تغییر و به‌روزرسانی مجدد ناست. بنابراین بهترین گزینه پیش رو استفاده از سخت‌افزارهای قابل پیکربندی مجدد FPGA است. اگرچه سرعت سیستم نهائی پیاده‌سازی شده بر روی FPGA نسبت به ASIC معمولاً کمتر است، ولی نسبت به سیستم پیاده‌سازی شده به‌صورت نرم‌افزاری و یا پیاده‌سازی‌ها بر روی پردازنده‌های همه‌منظوره از سرعت بیشتری برخوردار است.

با توجه به نکات ذکرشده و نیز اهمیت پیاده‌سازی الگوریتم‌های رمزنگاری در بسترهای سخت‌افزاری، هدف اصلی این مقاله ارائه راهکاری جهت دستیابی به بیشترین مقدار سرعت و حداقل مصرف منابع سخت‌افزاری، در پیاده‌سازی سخت‌افزاری الگوریتم رمزنگاری AES در بستر FPGA است. برای دست‌یابی به این اهداف، از روش خط‌لوله در قسمت‌های مختلف عملیات رمزنگاری داده‌ها به‌منظور کاهش طول مسیرهای بحرانی و نیز کاهش میزان تأخیر مسیرها از ورودی به خروجی و درنتیجه دستیابی به بیشترین سرعت، استفاده خواهد شد. همچنین به بهینه کردن مراحل مختلف رمزنگاری بخصوص مرحله جای‌گشت بایت[۹]، با پیاده‌سازی این بخش به‌صورت مدار ترکیبی به‌جای استفاده از lookup table جهت دست‌یابی به حداقل منابع سخت‌افزاری مصرفی، پرداخته می‌شود. همچنین در طرح پیشنهادی از حافظه‌های BRAM موجود در FPGA استفاده نخواهد شد. ازآنجاکه در طرح پیشنهادی مرحله جای‌گشت بایت به‌صورت مدار ترکیبی پیاده‌سازی می‌شود، می‌توان با افزودن رجیسترهای خط‌لوله در مکان مناسب مانع از ایجاد مسیرهایی با تأخیر بالا و درنتیجه افت فرکانس مدار شد، اما همواره باید دقت شود که استفاده از این رجیسترها موجب افزایش منابع سخت‌افزاری مصرفی می‌شود. ازاین‌رو در کاربردهای مختلف، طراح سیستم همواره باید بسته به نوع کاربرد تعادلی میان دو عامل گذردهی[۱۰] و مصرف منابع سخت‌افزاری ایجاد کند.

در پایان این بخش لازم به ذکر است که نمونه سخت‌افزارهای رمزنگاری و نیز هسته‌های IPCore رمزنگاری مناسب جهت قرار گرفتن بر روی FPGA توسط کمپانی‌های خارجی تولید شده و موجود است،

اما ازآن‌جهت که بحث بر سر امنیت اطلاعت بوده و همواره داده‌های پراهمیت رمز می‌شوند، نمی‌توان به نمونه محصولات خارجی در این زمینه اعتماد نمود. همچنین قیمت محصولات خارجی بسیار بالا است. به‌این‌علت پیاده‌سازی‌های داخلی در زمینه الگوریتم‌های رمزنگاری چه در بسترهای سخت‌افزاری و چه در بسترهای نرم‌افزاری حائز اهمیت بسیار می‌باشند.

در این مقاله پس از مقدمه، در بخش ۲ برخی از بهترین کارهای انجام‌شده به‌منظور پیاده‌سازی الگوریتم‌های رمزنگاری AES بر بسترهای سخت‌افزاری ارائه شده است. سپس در بخش ۳ به‌صورت کاملاً مختصر معماری این الگوریتم آورده شده است. در بخش ۴ نیز طرح پیشنهادی این مقاله جهت دستیابی به اهداف بیان‌شده و طراحی رمزنگار AES در بستر FPGA ارائه می‌شود. در بخش ۵ نیز نتایج حاصل از پیاده‌سازی به همراه مقایسه با دیگر کارهای انجام‌شده ارائه می‌گردد. در نهایت در بخش ۶ به‌عنوان نتیجه‌گیری مطالب این مقاله مرور می‌شود. همچنین در ضمایم روابط و مدارهای ترکیبی موردنیاز جهت طراحی S-Box این الگوریتم آورده شده است.

۲- بررسی کارهای انجام‌شده

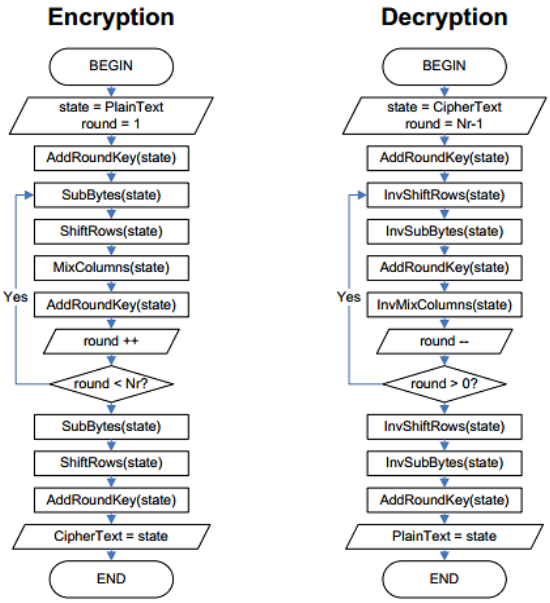
ازآنجاکه امروزه نیاز به افزایش سرعت در شبکه‌های کامپیوتری و از طرفی نیاز به انتقال اطلاعات به‌صورت امن روزبه‌روز اهمیت بیشتری پیدا کرده است، طراحان زیادی بر پیاده‌سازی سخت‌افزاری و نرم‌افزاری الگوریتم‌های رمزنگاری باقابلیت گذردهی بالا تمرکز کرده‌اند. همچنین ازآنجاکه ماهیت پیاده‌سازی‌های نرم‌افزاری بسیار کندتر از پیاده‌سازی‌های سخت‌افزاری است و پیاده‌سازی‌های نرم‌افزاری قابلیت بهینه‌سازی زیادی ندارند، به‌منظور پیاده‌سازی‌های پرسرعت الگوریتم‌های رمز، بیشترین تمرکز بر روی راهکارهای سخت‌افزاری است. راهکارهای سخت‌افزاری را می‌توان در بالاترین سطح به پیاده‌سازی بر روی پردازنده‌های همه‌منظوره، FPGA و ASIC تقسیم‌بندی کرد. پیاده‌سازی بر روی هریک از این بسترها مزایا و معایب خود را داشته و بسته به کاربردهای مختلف و میزان گذردهی مورد انتظار، استفاده از هریک از این بسترها می‌تواند مفید باشد.

در این راستا مقاله [۲] به طراحی و پیاده‌سازی الگوریتم رمزگذار و رمزگشای AES با طول کلید ۱۲۸ بیت در بستر FPGA و با طرح خط‌لوله فیدبک‌دار پرداخته و در نهایت به‌میزان بازدهی ۲/۵ Gb/s در فرکانس پالس ساعت ۱۹۵ MHz دست یافته است. نویسنده این مقاله جهت مقایسه طراحی خود با دیگر طراحی‌های ارائه‌شده، بیان می‌کند ازآنجاکه این طراحی میزان منابع پایینی مصرف کرده است در صورت لزوم جهت دستیابی به نرخ گذردهی بالاتر می‌توان با قبول افزایش هزینه سخت‌افزار مصرف‌شده، به تعداد دلخواه پردازنده رمزنگار را با یکدیگر موازی کرد و به نرخ داده بالاتر دست یافت. در رابطه با این ایده، ذکر این نکته ضروری است که موازی کردن ۱۰ پردازنده رمزنگار بر روی یک FPGA الزاماً به معنای ۱۰ برابر شدن میزان منابع مصرفی و نیز ۱۰ برابر شدن میزان گذردهی نیست. بلکه در عمل میزان منابع مصرفی بیشتر از ۱۰ برابر شده و نیز احتمال کمتر از ۱۰ برابر شدن گذردهی وجود دارد. چراکه چینش Sliceها نسبت به یک پردازنده رمزنگار متفاوت شده و احتمال کاهش فرکانس کل مدار نیز به‌دلیل پیچیده شدن طراحی وجود دارد. مقاله [۳] رمزنگار AES را به‌صورت خط‌لوله بر روی Virtex3200E FPGA پیاده‌سازی کرده و در بهترین حالت، در معماری غیر چرخشی و پیاده‌سازی اجزاء مدار به‌صورت مدار ترکیبی به بازدهی ۱۸/۱۲ Gbps در فرکانس ۱۴۵ MHz دست یافته است. این مقاله موتور رمزنگار را با دو معماری چرخشی و غیر چرخشی پیاده‌سازی نموده و همچنین مانند [۲] بخش S-Box الگوریتم بر اساس سه ایده استفاده از جداول LUT، استفاده از Block RAM و پیاده‌سازی این بخش به‌صورت مدار ترکیبی بر روی میدان محدود گالوا پیاده‌سازی و این ایده‌ها را با یکدیگر مقایسه کرده است. با توجه به نتایج ارائه‌شده در این مقاله می‌توان تأثیر معماری‌های مختلف را بر میزان بازدهی و منابع مصرفی به‌منظور استفاده در کاربردهای متفاوت، دریافت. مقاله [۴] نیز به پیاده‌سازی رمزنگار AES با معماری غیر چرخشی بر روی Virtex II Pro، با طراحی اجزا با استفاده از BRAM و یا به‌صورت مدار ترکیبی پرداخته است. همچنین هر دور الگوریتم را به‌صورت چهار طبقه خط‌لوله و یا هفت طبقه خط‌لوله پیاده‌سازی کرده است. در طراحی هر دور به‌صورت هفت طبقه خط‌لوله و بدون استفاده از BRAM در بهترین حالت به حداکثر بازدهی ۲۱/۶۴ Gbps در فرکانس ۱۶۹/۱ MHz با تأخیر ۷۱ پالس ساعت و میزان منابع مصرفی ۹۴۴۶ Slices دست یافته است. همچنین در طراحی هر دور الگوریتم به‌صورت چهار طبقه خط‌لوله و بدون استفاده از BRAM نیز به بازدهی ۲۱/۵۴ Gbps در فرکانس ۱۶۸/۳ MHz با تأخیر ۴۱ پالس ساعت و میزان منابع مصرفی ۱۲۴۵۰ Slices دست یافته است. همین نویسنده دو سال بعد در مقاله [۵] رمزنگاری با معماری مشابه (چهار طبقه خط‌لوله در هر دور) بر روی ASIC پیاده‌سازی کرده و به‌میزان بازدهی ۳۰ Gb/s تا ۷۰ Gb/s با حداکثر فرکانس (۱۱۸/۳ b/cc) ۶۰۶ MHz در تکنولوژی ۰/۱۸ um CMOS دست یافته است. همچنین نویسنده این مقاله تأثیر پیاده‌سازی‌های متفاوت از طراحی S-Box به‌صورت LUT و یا مدار ترکیبی و نیز تعداد طبقات خط‌لوله در هر دور را با یکدیگر مقایسه کرده است (یک تا چهار طبقه خط‌لوله در هر دور). مقاله [۶] از تکنیک خط‌لوله‌های جزئی جهت تسریع عملیات رمزنگاری AES استفاده کرده و این الگوریتم را با استفاده از خانواده Altera Stratix FPGA پیاده‌سازی کرده است. درواقع تکنیک استفاده‌شده در این روش بین دو معماری چرخشی و غیر چرخشی برای پیاده‌سازی الگوریتم AES بوده و برای FPGAهای ارزان‌قیمت مناسب است. در نهایت در بهترین حالت به‌میزان گذردهی ۲۰/۴۸ Gb/s در فرکانس ۱۶۰/۰۵ MHz و میزان منابع مصرفی ۱۲۵۶۰ Slices برای معماری غیر چرخشی رسیده است. مقاله [۷] به طراحی یک کمک پردازنده سخت‌افزاری برای الگوریتم رمزنگاری AES بر روی تراشه‌های FPGA

و ASIC پرداخته است. همچنین برای چهار هسته پردازنده در فرکانس ۹۵ MHz به بازدهی ۶۱۵ Mb/s بر روی Virtex 4 FPGA دست یافته است. مقاله [۸] راهکار جدیدی برای طراحی موتورهای رمزنگار با کلید متقارن با استفاده از کمک پردازنده‌ها ارائه داده است. در نهایت طرح پیشنهادی روی تراشه Xilinx Virtex 5 FPGA-XC5VLX110T پیاده‌سازی شده و به‌میزان بازدهی ۲۵۶ Mb/s در فرکانس ۱۶۰ MHz دست یافته است. لازم به ذکر است که کمک‌پردازنده استفاده‌شده در این خانواده می‌تواند با استفاده از تکنولوژی PowerPC و یا MicroBlaze طراحی و استفاده شود. توجه شود که استفاده از هسته‌های PowerPC و MicroBlaze ازآن‌جهت که این هسته‌ها خاصیت پردازنده‌ها را دارند و دستورالعمل‌ها را به‌صورت ترتیبی اجرا می‌کنند برای کاربردهای پرسرعت و نیازمند به مصرف پایین منابع سخت‌افزاری مناسب نمی‌باشند. مقاله [۹] ادعا می‌کند که به‌میزان بازدهی ۵۰۰ Gb/s برای رمزنگار و رمزگشای AES دست یافته است. درواقع نویسنده مقاله یک هسته رمزنگار با بازدهی ۴/۹ Gb/s با میزان سطح مصرفی ۱۲۶۴ Slices و ۶۳۱۸ LUTs طراحی کرده و بیان می‌کند که می‌توان ۱۰۰ تکرار از این هسته رمزنگار را در یک Virtex-6 FPGA قرار داد و درنتیجه به سرعت ۵۰۰ Gb/s دست یافت. همچنین تلاش کرده تا میزان سطح مصرفی یک هسته رمزنگار را تا حد امکان به حداقل رسانده و عملیات را به‌صورت پیاده‌سازی با استفاده از مداری ترکیبی انجام دهد. پیش‌ازاین نقدی بر معایب استفاده از چنین معماری برای مقاله [۲] بیان شد. مقاله [۱۰] نیز به پیاده‌سازی رمزنگار و رمزگشای AES بر روی شتاب‌دهنده سخت‌افزاری با تکنولوژی CMOS ۴۵ nm پرداخته است. این طرح محاسبات دورهای رمزنگاری را در حوزه میدان گالوای $GF(2^4)^2$ محاسبه کرده است و به کارایی ۶۶ Gb/s با ولتاژ کاری ۱/۳۵ V برای الگوریتم AES-128 دست یافته است. مقاله [۱۱] با استفاده از زبان برنامه‌نویسی VHDL و ابزار سنتز Xilinx ISE 12.1 یک پردازنده رمزنگار چندهسته‌ای را بر روی تراشه Xilinx Virtex 4 FPGA پیاده‌سازی کرده است. نتایج این مقاله نشان می‌دهد که با افزایش تعداد هسته‌ها، منابع مصرفی به‌صورت خطی افزایش یافته و فرکانس کاری مدار به‌صورت ثابت به‌میزان ۱۹۲ MHz باقی می‌ماند. در واقع نویسنده چهار موتور رمزنگار متفاوت با استفاده از ۲، ۴، ۶ و ۸ هسته طراحی و آن‌ها را با یکدیگر مقایسه کرده است. در نهایت به بیشترین بازدهی ۳۴۶۰ Mbps برای پردازنده ۸ هسته‌ای دست یافته است. لازم به ذکر است که طراح با استفاده از تکنولوژی PicoBlaze این هسته‌های کمک پردازنده را در FPGA ایجاد کرده و الگوریتم رمز را بر روی آن‌ها اجرا می‌کند. مقاله [۱۲] راهکار متفاوتی را ارائه می‌دهد، در واقع یک پردازنده همه‌منظوره را با یک کمک پردازنده رمزنگار AES برای انجام عملیات رمزگذاری و رمزگشایی داده‌ها با گذردهی بالا، بر اساس معماری decoupled architecture توسعه می‌دهد و آن را FastCrypto نام‌گذاری می‌کند. این کمک‌پردازنده رمزنگار AES شامل چهار موتور رمز موازی است که هریک به‌صورت خط‌لوله طراحی شده‌اند. در نهایت FastCrypto با استفاده از زبان برنامه‌نویسی VHDL بر روی خانواده Xilinx Virtex 5 FPGA پیاده‌سازی شده است. نویسنده ادعا می‌کند میزان بازدهی موتور رمز برابر با ۲۲۲ Gb/s در فرکانس ۴۴۴ MHz برای چهار موتور رمزنگار خط‌لوله AES که به‌صورت موازی عملیات رمزنگاری را انجام می‌دهند، به‌دست آمده است. در نهایت مقاله مروری [۱۳] نیز اطلاعات و راهکارهای کلی در این مبحث را بیان کرده و همچنین برخی از مقالات مرتبط در این زمینه شامل پیاده‌سازی‌های متفاوت در بسترهای سخت‌افزاری را مورد بررسی و مقایسه قرار داده است. همچنین مراجع [۱۴، ۱۵] نمونه طراحی‌های تجاری کمپانی Hellion هستند. این محصولات بر روی تراشه Xilinx Virtex 7 FPGA به‌میزان بازدهی ۴۰ Gb/s با حداقل میزان منابع مصرفی ۳۴۰۰ Slices دست یافته‌اند. نتایج نشان می‌دهد که طراحی ارائه‌شده در این پژوهش با محصولات این کمپانی قابل رقابت است.

۳- آشنایی با الگوریتم رمزنگاری رایندال

رایندال یک الگوریتم رمزنگاری متقارن است. طول بلوک داده موردپذیرش توسط این الگوریتم تنها می‌تواند به‌اندازه یکی از سه مقدار ۱۲۸، ۱۹۲ و ۲۵۶ بیت باشد. طول کلید نیز مستقل از طول داده می‌تواند، ۱۲۸، ۱۹۲ یا ۲۵۶ بیت باشد. الگوریتم بسته به طول داده و طول کلید شامل ۱۰، ۱۲ یا ۱۴ دور خواهد بود [۱۶]. تفاوت الگوریتم AES با رایندال در این است که الگوریتم AES تنها از طول قالب داده ۱۲۸ بیتی به‌عنوان ورودی پشتیبانی می‌کند. فلوچارت الگوریتم رایندال برای عملیات رمزگذاری و رمزگشایی داده‌ها در شکل ۱ نشان داده شده است.



شکل ۱: فلوچارت رمزگذار و رمزگشایی AES [۱۷]

این الگوریتم دارای ساختاری برای بسط کلید است که از روی کلید اصلی بسته به تعداد دورها، تعدادی زیرکلید تولید می‌کند که در هر دور با داده جمع (XOR) می‌شوند. الگوریتم شامل سه تبدیل مهم ()MixColumn و ()ShiftRow و ()SubByte است. در این الگوریتم ساختار سیستم رمزگشا دقیقاً مشابه سیستم رمزنگار نیست. همچنین چون با افزایش طول کلید تعداد دورهای الگوریتم افزایش می‌یابد، زمان اجرا و سرعت الگوریتم به طول کلید وابسته است [۱].

در این الگوریتم داده‌ها در متغیرهای ماتریسی (ماتریس state) با چهار سطر و چهار ستون ذخیره و در تبدیل‌ها و دورهای متفاوت الگوریتم مورد پردازش قرار می‌گیرند. اندازه هر درایه از این ماتریس نیز برابر با یک بایت است. در ادامه هریک از تبدیل‌های به‌کاررفته در الگوریتم معرفی و ازلحاظ تکنیک‌های پیش رو جهت پیاده‌سازی به‌صورت سخت‌افزاری بررسی می‌شنود.

۳-۱- بررسی تبدیل SubBytes

واحد SubByte با استفاده از جدول S-Box (جدول جانشینی بایت)، یکایک بایت‌های ماتریس state را با مقادیر جدیدی جایگزین می‌کند. وظیفه این واحد انجام عملیات غیر خطی بر روی داده‌ها جهت افزایش امنیت سیستم است. لازم به ذکر است که جهت پیاده‌سازی سخت‌افزاری این الگوریتم یکی از مهم‌ترین بخش‌ها در تعیین میزان گذردهی و مصرف منابع، همین بخش جایگشت بایت است. به همین علت اکثر تلاش‌های صورت گرفته باهدف کاهش مصرف منابع و افزایش گذردهی موتور رمزنگاری AES بر روی این بخش بوده است.

درواقع مقادیر جدول S-Box با استفاده از دو تبدیل مهم به‌دست می‌آید. ابتدا با انجام معکوس حاصل‌ضرب در میدان محدود گالوا $GF(2^8)$ بر روی داده ورودی و سپس با انجام تبدیل Affine بر روی نتایج حاصل، مقادیر این جدول به‌دست می‌آید. ازآنجاکه در این بخش عملیات بر روی داده‌های ۸ بیتی صورت می‌گیرد و به‌علت محدود بودن دامنه اعداد ۸ بیتی در مبنای ۲ (۲۵۶ مقدار متفاوت برای یک بایت)، می‌توان نتایج حاصل از عملیات این بخش را از پیش محاسبه کرده و در جداولی ذخیره و استفاده نمود.

بنابراین به‌منظور پیاده‌سازی سخت‌افزاری بخش S-Box الگوریتم و یافتن یک بایت جایگزین سه راهکار اولیه پیش رو است. راه‌حل اول محاسبه تمامی مقادیر ممکن برای ۸ بیت داده ورودی و ذخیره نتایج در یک جدول مراجعه LUT[۱۱] و پیاده‌سازی این جدول با استفاده از گیت‌های داخلی FPGA است. راه‌حل دوم استفاده از حافظه‌های BRAM در داخل FPGA و قرار دادن این جدول در این حافظه‌ها است. راه‌حل سوم نیز پیاده‌سازی این بخش به‌صورت مدار ترکیبی است. استفاده از هریک از این راهکارها بسته به سیاست‌های طراحی و کاربرد می‌تواند مفید باشد اما لازم به ذکر است با توجه به مباحث و نتایج ارائه‌شده در مقالات [۵-۳] به‌منظور دست‌یابی به بیشترین میزان گذردهی، بهترین روش پیاده‌سازی این بخش به‌صورت مدار ترکیبی است.

در اینجا نیز ذکر این نکته خالی از لطف نیست که هرچند استفاده از Block RAMها موجب کاهش منابع مصرفی می‌شوند اما با این اوصاف استفاده از مدارات ترکیبی به‌جای استفاده از Block RAMها در کاربردهای پرسرعت ترجیح داده می‌شود. چراکه در پیاده‌سازی‌های بر مبنای مدار ترکیبی همواره می‌توان با استفاده از تکنیک خط‌لوله موجب کاهش طول مسیرهای بحرانی و درنتیجه افزایش فرکانس مدار شد. درحالی‌که فرکانس دسترسی به حافظه محدود بوده و بر اساس تکنولوژی ساخت حافظه است. همچنین در هنگام نوشتن و یا خواند از حافظه، خصوصیات زمان‌بندی آن باید رعایت شود. واضح است که سیکل‌های موردنیاز برای خواندن و نوشتن از حافظه موجب کاهش میزان گذردهی سیستم نهایی می‌شود.

۳-۲- بررسی تبدیل ShiftRows

با استفاده از این توابع، هر چهار سطر از آرایه state در عملیات رمزگذاری به‌سمت چپ (در تبدیل ShiftRows) و در عملیات رمزگشایی به‌سمت راست (در تبدیل invShiftRows) شیفت داده می‌شود. لازم به ذکر است که در این چرخش سطری نکته ظریفی نهفته است. ازآنجاکه محاسبه تلفیقی و درهم‌سازی داده‌ها در طول الگوریتم به‌صورت ستونی انجام می‌گیرد، بنابراین درصورتی‌که چنین چرخش‌های سطری در الگوریتم وجود نداشت، عملیات رمزنگاری داده‌ها در کل الگوریتم بر روی ستون‌های ۳۲ بیتی متمرکز می‌شد. چنین چیزی معادل است با رمزنگاری بلوک‌های ۳۲ بیتی اطلاعات به‌جای بلوک‌های ۱۲۸ بیتی[۱]. واضح است که چنین اتفاقی چقدر موجب افت امنیت یک الگوریتم است، با مقایسه فضای حالت دو مقدار 2^{128} با 2^{32} این موضوع قابل درک است.

جهت پیاده‌سازی سخت‌افزاری این واحد نیازی به استفاده از عناصر منطقی نبوده و تنها کافی است مسیر سیمی مناسبی با توجه به موقعیت هر بایت پس از انجام شیفت، از ورودی به خروجی واحد کشیده شود.

۳-۳- بررسی تبدیل MixColumns

با استفاده از این تابع، هر ستون از ماتریس state به‌طور مستقیم و مجزا دیگر ستون‌ها تلفیق و درهم‌سازی می‌شود. این فرآیند بدین‌صورت است که هر ستون از ماتریس state در یک ماتریس ثابت (ماتریس انتقال) ضرب شده و نتیجه به خروجی این بخش ارسال می‌شود. این عمل ضرب ماتریسی نیز بر روی میدان محدود گالوا $GF(2^8)$ و در پیمانه چندجمله‌ای مولد $m(x)$ صورت می‌گیرد. رابطه (۱) این چندجمله‌ای را نشان می‌دهد.

$$m(x) = x^8 + x^4 + x^3 + x + 1 \qquad (۱)$$

ازآنجاکه اعداد یک بایتی در میدان $GF(2^8)$ محدود می‌باشند، یک راهکار برای پیاده‌سازی این بخش می‌تواند به این‌صورت باشد که نتایج

حاصل از عملیات تلفیق ستونی با انجام مجموعه‌ای از پیش محاسبات، درون جدول‌هایی ذخیره و استفاده شود (این جدول‌ها نیز می‌تواند با استفاده از LUT و یا BRAM پیاده‌سازی شوند). در این‌صورت عمل ضرب در این میدان با جستجو در جدول و یک عمل XOR قابل محاسبه است. راهکار دیگر می‌تواند استفاده از مدار ترکیبی ضرب در میدان $GF(2^8)$ باشد.

۳-٤- بررسی تبدیل AddRoundKey

در الگوریتم AES مانند بسیاری از الگوریتم‌های متقارن دیگر، تنها یک شاه‌کلید[۱۲] وجود دارد. در هر دور عملیات رمزنگاری نیز یک کلید متفاوت از کلید دورهای دیگر مورد استفاده قرار می‌گیرد. در ابتدای انجام عملیات رمزگذاری و یا رمزگشایی داده‌ها کلیه کلیدهای دور[۱۳] طبق یک الگوریتم پیچیده وارون‌ناپذیر از روی شاه‌کلید ساخته می‌شود.

تبدیل AddRoundKey یکی از واحدهای ساده در یک دور رمزنگاری است زیرا تنها کلید هر دور را با داده آن دور XOR (جمع در پیمانه ۲) می‌کند. پیاده‌سازی سخت‌افزاری این بخش نیز با استفاده از عملگر XOR به‌راحتی قابل پیاده‌سازی است.

۴- طراحی رمزنگار AES بر بستر FPGA

در بسترهای سخت‌افزاری از راهکارهای متفاوتی بر اساس سیاست‌های کاربردی می‌توان بهره برد. در طراحی سخت‌افزار سطوح مختلف انتزاع وجود دارد. راهکارهای پیاده‌سازی در بالاترین سطح انتزاع، به پیاده‌سازی الگوریتم بر روی سه بستر متفاوت (با معماری کاملاً متفاوت) پردازنده‌های همه‌منظوره (GPP)، سخت‌افزارهای قابل بازپیکربندی مجدد FPGA و مدارهای مجتمع ASIC قابل تقسیم است.

در سطح طراحی و پیاده‌سازی الگوریتم نیز راهکارهای متفاوتی پیش رو است. همان‌گونه که در شکل ۱ دیده می‌شود، ساختار الگوریتم رمزنگاری AES به‌گونه‌ای است که می‌تواند به‌صورت حلقه‌ای پیاده‌سازی شود. به این معنا که یک پیاده‌سازی از دور میانی را می‌توان به‌صورت چرخشی و به‌صورت متوالی مورد استفاده قرار داد تا مراحل رمزنگاری داده‌ها تکمیل شود. راهکار دیگر در این سطح از طراحی، غیر چرخشی کردن مسیر داده در الگوریتم است. در این حالت هر دور الگوریتم به‌صورت مجزا از دورهای قبلی فراخوانی و مورد استفاده قرار گیرد. درواقع در این حالت هر دور الگوریتم بر روی Sliceهای مجزایی نسبت به دیگر دورها پیاده‌سازی می‌شود و داده خروجی از هر دور الگوریتم به دور بعد داده می‌شود.

ازآنجاکه در پیاده‌سازی‌های سخت‌افزاری می‌توان به‌منظور تسریع در انجام عملیات از راهکارهای موازی‌سازی بهره برد، در سطح بعدی طراحی می‌توان بر سر تعداد فراخوانی‌ها در قسمت‌های مختلف الگوریتم تصمیم‌گیری کرد. همان‌گونه که در بخش ۲-۱- بیان شد، بخش S-Box الگوریتم را می‌توان در بالاترین سطح طراحی به سه شیوه استفاده از جداول LUT، استفاده از حافظه‌های BRAM و یا به‌صورت مدار ترکیبی پیاده‌سازی نمود. به‌این‌علت که بخش S-Box بر روی هر بایت به‌صورت مجزا عمل می‌کند (یک بایت ورودی را مستقل از دیگر بایت‌های ماتریس state به بایت متناظر تبدیل می‌کند) و در کل ماتریس state شامل ۱۶ بایت است، تبدیل SubByte را می‌توان صرف‌نظر از نحوه پیاده‌سازی (سه انتخاب بیان‌شده)، حداقل با یک S-box که به‌صورت چرخشی مورد استفاده قرار می‌گیرد و یا حداکثر با ۱۶ واحد S-box که به‌صورت موازی عمل می‌کنند، پیاده‌سازی نمود. با این اوصاف و بدون در نظر گرفتن S-Boxهای استفاده‌شده در واحد KeyExpansion، در طراحی موتور رمزنگار AES-128 بر بستر سخت‌افزاری می‌توان حداقل یک فراخوانی از S-Box و یا حداکثر ۱۶۰ فراخوانی از آن داشت.

در سطح بعدی از طراحی می‌توان بر چگونگی پیاده‌سازی اجزا و تبدیل‌ها در الگوریتم تمرکز کرد. به‌منظور دستیابی به حداقل منابع سخت‌افزاری مصرفی، تا آنجا که ممکن است می‌بایست عملیات رمزنگاری و مسیر داده با استفاده از حلقه‌های چرخشی پیاده‌سازی شود. حال آن‌که جهت دست‌یابی به بیشترین گذردهی، تا آنجا که ممکن است می‌بایست عملیات رمزنگاری و مسیر داده به‌صورت غیر چرخشی پیاده‌سازی شود تا در اولین گام بتوان با هر پالس ساعت، ۱۲۸ بیت خروجی رمز شده تولید کرد. بنابراین در طراحی با بیشترین گذردهی نیاز است از جدول S-Box، ۱۶۰ فراخوانی مستقل داشت. ازآنجاکه در چنین طرحی میزان منابع مصرفی بسیار بالا می‌رود، باید هر قسمت از الگوریتم به‌صورت بهینه پیاده‌سازی شود.

همچنین به‌منظور دستیابی به بیشترین فرکانس می‌بایست قسمت‌های مختلف طراحی به‌گونه‌ای باشند که موجب افت فرکانس موتور رمزنگار نشوند. برای تحقق چنین اهدافی، طراحی S-Box با استفاده از LUT پیشنهاد نمی‌شود چراکه موجب افزایش بسیار زیاد منابع مصرفی و کاهش فرکانس مدار نسبت به دیگر روش‌ها می‌شود. همچنین استفاده از حافظه‌های BRAM به‌علت افزایش تأخیر و همچنین کاهش فرکانس مدار، توصیه نمی‌شود. از طرفی در پیاده‌سازی S-Box به روش مدار ترکیبی، همواره می‌توان با قرار دادن رجیسترهای خط‌لوله موجب افزایش فرکانس مدار شد، هرچند که با افزایش این رجیسترها منابع مصرفی نیز افزایش می‌یابد.

۴-۱- طراحی بخش S-Box الگوریتم AES

طراحی یک S-Box فشرده اساسی‌ترین مشکل در طراحی سخت‌افزاری این الگوریتم به‌منظور دست‌یابی به حداقل منابع مصرفی و حتی دست‌یابی به فرکانس پالس ساعت بالا است. همان‌گونه که بیان شد به‌منظور طراحی این الگوریتم برای خطوط پرسرعت، بهترین راهکار پیاده‌سازی جدول S-Box به‌صورت مدار ترکیبی است. همچنین گفته شد که این جدول با انجام معکوس حاصل‌ضرب در میدان محدود گالوا $GF(2^8)$ بر روی داده ورودی و سپس با انجام تبدیل Affine بر روی نتایج حاصل قابل محاسبه است.

ازآنجاکه انجام چنین ضربی در این میدان بسیار پرهزینه و زمان‌گیر است مخترع این الگوریتم راهکار دیگری برای انجام این ضرب ارائه داده است. بر طبق [۱۸] می‌توان به‌جای محاسبه معکوس حاصل‌ضرب در میدان $GF(2^8)$، معکوس حاصل‌ضرب در $GF(2^4)$ را به همراه چند عمل ضرب، مربع و جمع در میدان $GF(2^4)$ محاسبه کرد. مقاله [۱۸] بیان می‌کند که با استفاده از چندجمله‌ای تحویل‌ناپذیر x^2+Ax+B، مقدار معکوس حاصل‌ضرب برای هر چندجمله‌ای دلخواه $bx+c$ را می‌توان با توجه به رابطه (۲) محاسبه کرد.

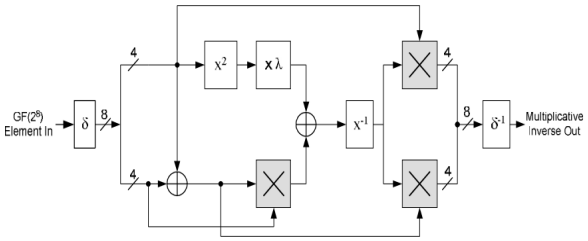
$$(bx+c)^{-1} = b(b^2B+bcA+c^2)^{-1}x + (c+bA)(b^2B+bcA+c^2)^{-1} \quad (۲)$$

بنابراین عناصر در میدان $GF(2^8)$ را می‌توان به‌صورت $bx+c$ نمایش داد به‌گونه‌ای که b نیمه پرارزش‌تر و c نیمه کم‌ارزش‌تر آن باشد. بر طبق [۱۹] محاسبه معکوس حاصل‌ضرب با تجزیه عناصر در میدان محدود $GF(2^8)$ (که بسیار پیچیده است)، به میدان‌هایی از مرتبه پایین‌تر $GF(2^1)$، $GF(2^2)$ و $GF((2^2)^2)$ صورت می‌گیرد. به‌منظور انجام این تبدیل‌ها، چندجمله‌ای‌های تحویل‌ناپذیر نشان داده‌شده در روابط (۳) مورد استفاده قرار می‌گیرند.

$$\begin{aligned} &GF(2^2) \to GF(2) && : x^2+x+1 \\ &GF((2^2)^2) \to GF(2^2) && : x^2+x+\varphi \\ &GF(((2^2)^2)^2) \to GF((2^2)^2) && : x^2+x+\lambda \end{aligned} \quad (۳)$$

در این روابط $\varphi=\{10\}_2$ و $\lambda=\{1100\}_2$ است. ازآنجاکه تبدیل $GF(((2^2)^2)^2) \to GF((2^2)^2)$ مدنظر است، چندجمله‌ای تحویل‌ناپذیر $x^2+x+\lambda$ انتخاب می‌شود. پس در رابطه (۲) مقدار $A=1$ و $B=\lambda$ خواهد بود. بنابراین رابطه (۲) را می‌توان به‌صورت رابطه (۴) بازنویسی کرد. شکل ۲ مدار ترکیبی رابطه (۴) را نشان می‌دهد.

$$(bx+c)^{-1} = b(b^2\lambda+c(b+c))^{-1}x + (c+b)(b^2\lambda+c(b+c))^{-1} \quad (۴)$$



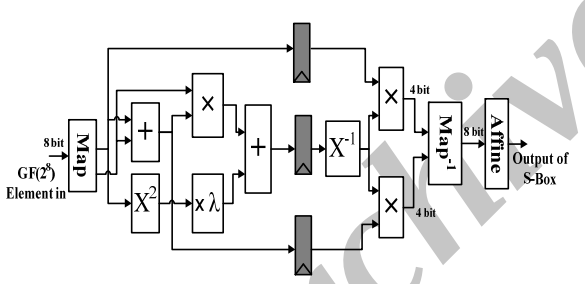
شکل ۲: مدار محاسبه معکوس حاصل‌ضرب برای S-Box [۱۹]

در این شکل بلوک δ یک جای‌گشت از $GF(2^8)$ به $GF(2^4)$ را انجام می‌دهد و بلوک δ^{-1} عکس این عمل را بر عهده دارد. بلوک x^2 مربع داده ورودی را در حوزه $GF(2^4)$ محاسبه می‌کند. بلوک x^{-1} مقدار معکوس حاصل‌ضرب را در حوزه $GF(2^4)$ محاسبه می‌کند. بلوک‌های + و × به‌ترتیب عملگرهای جمع و ضرب در این میدان هستند و بلوک $\lambda\times$ نیز عملگر ضرب با مقدار ثابت در میدان $GF(2^4)$ است. مدار ترکیبی هریک از این بلوک‌ها به همراه اثبات ریاضی آن‌ها در ضمایم آورده شده است.

۴-۱-۱- معماری پیشنهادی برای طراحی S-Box

ازآنجاکه مدار نشان داده‌شده در شکل ۲ به همراه مدارهای ارائه‌شده در ضمیمه به‌صورت مدار ترکیبی پیاده‌سازی می‌شود، درنتیجه این بخش موجب ایجاد مسیر بحرانی در کل مسیر داده از ورودی به خروجی در موتور رمزنگار می‌شود. به همین علت لازم است تا با استفاده از تکنیک خط‌لوله، طول مسیرهای بحرانی کاهش داده شوند، درنتیجه به‌دلیل کوتاه‌تر شدن طول مسیرهای بحرانی، موتور رمزنگار می‌تواند با فرکانس بالاتری کار کند.

با انجام بررسی‌ها انجام‌شده و با توجه به گزارش‌های نرم‌افزار Xilinx ISE، مناسب‌ترین قسمت جهت افزودن این رجیسترها و تبدیل مدار به دو طبقه خط‌لوله، مطابق طرح نشان داده‌شده در شکل ۳ است. همان‌گونه که در این شکل دیده می‌شود، این رجیسترها تقریباً در میانه مدار قرار داده شده‌اند.

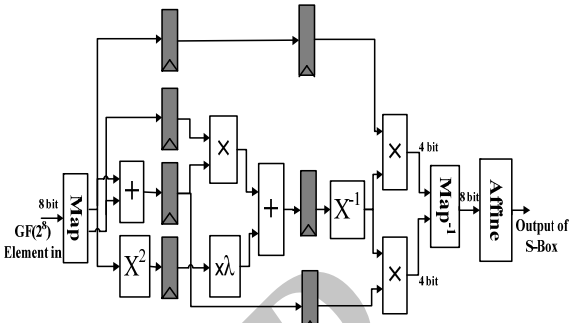


شکل ۳: مدار پیشنهادی ۱ برای S-Box به‌صورت دو طبقه خط‌لوله با استفاده از ۳ رجیستر

همچنین به‌منظور دستیابی به فرکانس بالاتر می‌توان یک طبقه رجیستر خط‌لوله دیگر به مدار پیشنهادی اضافه شود. البته همواره باید به این نکته توجه شود که افزودن رجیسترهای خط‌لوله همواره موجب افزایش فرکانس مدار نمی‌شود. بلکه مکان قرارگیری این رجیسترها بسیار مهم است. به‌عنوان‌مثال درصورتی‌که رجیستر دوم در شکل ۳ بعد از تابع X^{-1} و یا قبل از تابع + قرار منتقل شود، میزان فرکانس مدار کاهش می‌یابد. مکان قرارگیری این رجیسترها باید با توجه به طول مسیرهای متفاوت و همچنین با توجه به گزارش‌های حاصل از سنتز و پیاده‌سازی توسط نرم‌افزار ISE، انتخاب شود.

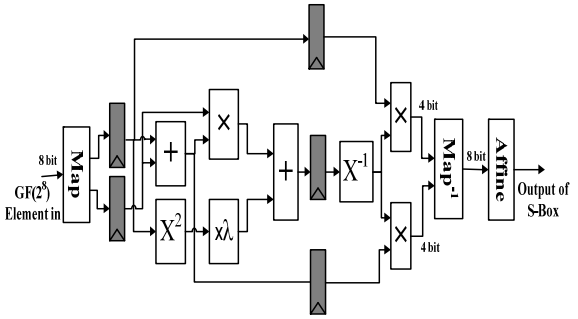
از طرف دیگر نیز همواره می‌بایست بین فرکانس مدار و میزان منابع مصرفی توازنی برقرار شود، زیرا همان‌گونه که دیده می‌شود، یکی

از عواملی که باعث می‌شود این دو پارامتر بر ضد یکدیگر باشند، تعداد رجیسترهای خط‌لوله استفاده‌شده در طراحی است. شکل ۴ مدار پیشنهادی S-Box را به‌صورت سه طبقه خط‌لوله نشان می‌دهد.



شکل ۴: مدار پیشنهادی ۲ برای S-Box به‌صورت سه طبقه خط‌لوله با استفاده از ۷ رجیستر

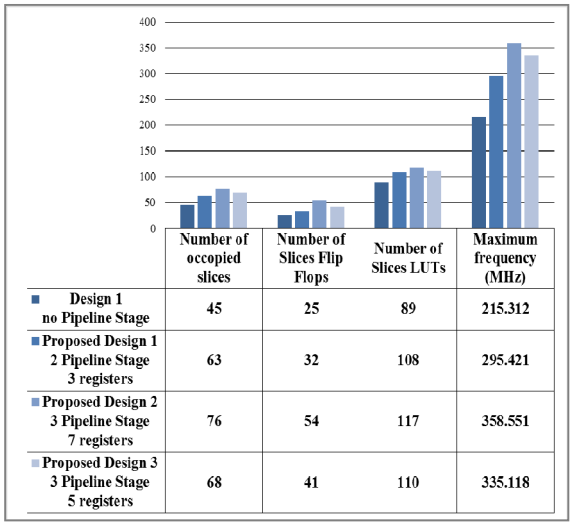
نتایج حاصل از پیاده‌سازی و همچنین گزارش‌های سنتز نرم‌افزار ISE، نشان می‌دهد که در حالت سه طبقه خط‌لوله مدار S-Box، بهترین حالت قرارگیری رجیسترها جهت دستیابی به بیشترین فرکانس، مطابق معماری ارائه‌شده در شکل ۴ است، اما به‌منظور کاهش مصرف منابع سخت‌افزاری می‌توان از مدار ارائه‌شده در شکل ۵ نیز استفاده کرد.



شکل ۵: مدار پیشنهادی ۳ برای S-Box به‌صورت سه طبقه خط‌لوله با استفاده از ۵ رجیستر

جدول و نمودار نشان داده‌شده در شکل ۶ طراحی‌های ارائه‌شده در شکل‌های ۲ تا ۵ را ازلحاظ مصرف منابع و فرکانس مدار، بر روی تراشه Virtex 4 FPGA(xc4vlx25) با یکدیگر مقایسه می‌کند.

مدار پیشنهادی نشان داده‌شده در شکل ۵، ازآنجاکه از تعداد رجیسترهای کمتری نسبت به مدار شکل ۴ استفاده می‌کند، تعداد Sliceهای کمتری نیز مصرف می‌کند. ازآنجاکه مکان قرارگیری رجیسترهای مدار شکل ۴ مناسب‌تر از مدار شکل ۵ است، درنتیجه با فرکانس پالس ساعت بالاتری عمل می‌کند. این موضوع در نتایج ارائه‌شده در شکل ۶ قابل مشاهده است.



	Number of occopied slices	Number of Slices Flip Flops	Number of Slices LUTs	Maximum frequency (MHz)
Design 1 no Pipeline Stage	45	25	89	215.312
Proposed Design 1 2 Pipeline Stage 3 registers	63	32	108	295.421
Proposed Design 2 3 Pipeline Stage 7 registers	76	54	117	358.551
Proposed Design 3 3 Pipeline Stage 5 registers	68	41	110	335.118

شکل ۶: نمودار مقایسه مصرف منابع و فرکانس با اضافه کردن طبقات خط‌لوله در طراحی S-Box

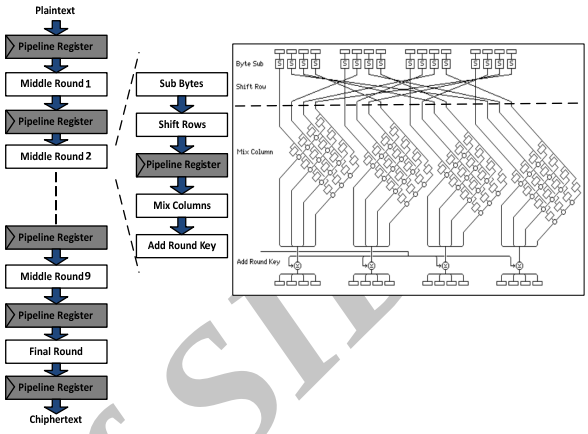
همان‌گونه که در شکل ۶ دیده می‌شود، با افزایش تعداد طبقات خط‌لوله میزان منابع مصرفی و همچنین فرکانس پالس ساعت مدار افزایش می‌یابد. ازآنجاکه هدف ما ایجاد توازن میان فرکانس مدار و منابع مصرفی است و در طرح نهایی برای موتور رمزنگار به تعداد ۱۶۰ مدار S-Box به‌صورت موازی استفاده می‌شود، بنابراین در طراحی موتور رمز از مدار پیشنهادی ارائه‌شده در شکل ۵ استفاده خواهیم کرد.

۴-۲- معماری پیشنهادی برای موتور رمزنگار AES

همان‌طور که پیش‌ازاین بیان شد، به‌منظور دستیابی بیشترین میزان گذردهی، در اولین سطح طراحی می‌بایست به دنبال این هدف بود تا با هر پالس ساعت یک بلوک داده (۱۲۸ بیت) رمزگذاری و یا رمزگشایی شود. به این منظور تمامی چرخش‌هایی که در مسیر داده وجود دارد باید باز شده و به‌صورت غیر چرخشی درآیند. واضح است که انجام چنین عملی منجر به افزایش منابع سخت‌افزاری مصرفی می‌شود. به همین منظور برای کنترل میزان منابع مصرفی سعی می‌شود تا هریک از اجزای الگوریتم به‌نحوی پیاده‌سازی شوند که کمترین میزان منابع را به خود اختصاص داده و از طرفی قادر باشند با فرکانس بالایی عمل نمایند و یا به‌عبارتی موجب ایجاد و یا افزایش تأخیر در مسیرهای بحرانی نشوند. به همین علت در بخش قبل مدارهای مختلفی برای پیاده‌سازی بخش S-Box این الگوریتم ارائه شد. همچنین این مدارها ازلحاظ منابع مصرفی و بیشترین فرکانس کاری با یکدیگر مقایسه شدند.

شکل ۷ معماری پیشنهادی برای رمزنگار پرسرعت AES را نشان می‌دهد. بر طبق این ایده بین هر دور الگوریتم یک رجیستر خط‌لوله قرار دارد. همچنین در داخل هر دور نیز یک رجیستر خط‌لوله قرار داده شده است. ازآنجاکه در این پژوهش هر دو بخش SubByte و MixColumns به‌صورت مدار ترکیبی پیاده‌سازی می‌شود، این دو

رجیستر خط‌لوله (رجیسترهای بین هر دو دور متوالی و داخل هر دور)، مانع از ایجاد مسیرهای بحرانی با تأخیر بالا می‌شوند. همچنین در این معماری از S-Box طراحی‌شده در شکل ۵ استفاده می‌شود. بنابراین هر دور از الگوریتم به‌صورت ۴ طبقه خط‌لوله طراحی می‌شود.



شکل ۷: معماری پیشنهادی برای طراحی رمزنگار پرسرعت AES

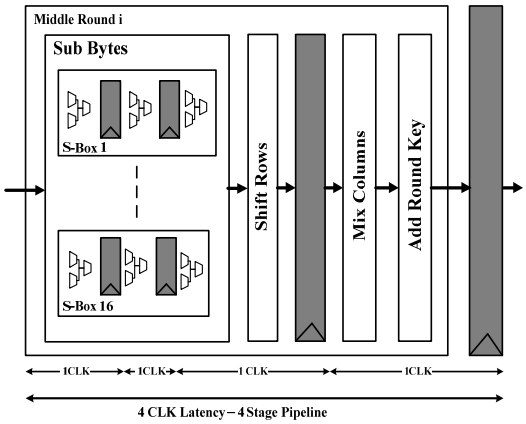
جدول ۱ میزان تأخیر موتور رمزنگاری را بسته به تعداد رجیسترهای خط‌لوله استفاده‌شده در قسمت‌های مختلف الگوریتم، نشان می‌دهد.

جدول ۱: تأخیر موتور رمزنگار در طراحی‌های مختلف

تعداد طبقات خط‌لوله در طراحی موتور رمزنگار	تأخیر موتور رمزنگار
طراحی هر دور بدون استفاده از تکنیک خط‌لوله	۱۰ پالس ساعت
طراحی هر دور به‌صورت دو طبقه خط‌لوله (افزودن رجیستر خط‌لوله میان دو تبدیل ShiftRows و MixColumns)	۲۰ پالس ساعت
طراحی هر دور به‌صورت سه طبقه خط‌لوله (طراحی S-Box به‌صورت دو طبقه خط‌لوله)	۳۰ پالس ساعت
طراحی هر دور به‌صورت چهار طبقه خط‌لوله (طراحی S-Box به‌صورت سه‌طبقه خط‌لوله)	۴۰ پالس ساعت

ازآنجاکه الگوریتم در ۱۰ دور اجرا می‌شود، درصورتی‌که در داخل هر دور الگوریتم از تکنیک خط‌لوله استفاده نشود، اجرای هر دور به‌میزان یک پالس ساعت زمان می‌گیرد و درنتیجه خروجی بعد از ۱۰ پالس ساعت تولید می‌شود.

شکل ۸ مسیر داده در یک دور از الگوریتم رمزنگاری را نشان می‌دهد. در این شکل مکان قرارگیری رجیسترهای خط‌لوله دورهای داخلی و خارجی مشخص شده است. همچنین در زیر هر قسمت از مسیر داده، تعداد پالس ساعت موردنیاز جهت تکمیل عملیات آن بخش و ارسال داده به بخش بعدی آورده شده است.



شکل ۸: مسیر داده در یک دور الگوریتم به همراه تأخیر هر زیر بخش در حالت ۴ طبقه خط‌لوله برای هر دور

رابطه (۵) فرمول محاسبه گذردهی را نشان می‌دهد.

$$\text{throughput} = \frac{\text{number of encryption bit} \times \text{clock frequency}}{\text{number of effective clock to complete encryption}} \quad (۵)$$

ازآنجاکه که موتور رمزنگار طراحی‌شده در این پژوهش قادر است با هر پالس ساعت (پس از طی شدن زمان راه‌اندازی به‌میزان ۷۰ پالس ساعت)، ۱۲۸ بیت داده رمز شده تولید کند، بنابراین افزایش مدت‌زمان راه‌اندازی از ۱۰ پالس ساعت به ۷۰ پالس ساعت در رمزنگاری داده‌های پرحجم موجب کاهش میزان گذردهی موتور رمزنگار نخواهد شد. بنابراین رابطه (۵) به رابطه (۶) تبدیل می‌شود.

$$\text{throughput} = \text{number of encryption bit} \times \text{clock frequency} \quad (۶)$$

نتایج حاصل از پیاده‌سازی معماری پیشنهادی در این مقاله نشان می‌دهد که این طرح بر روی تراشه Virtex 7 FPGA, xc7v585t-3ffg1157 قادر است با بیشترین پالس ساعت ۵۷۰/۷۷۶ MHz عمل کرده و به بیشترین میزان گذردهی ۷۱/۳۵ Gbps بر روی این تراشه دست یابد. همچنین این طراحی به‌میزان ۳۶۶۹ Slices مصرف کرده و از حافظه‌های BRAM استفاده نمی‌کند.

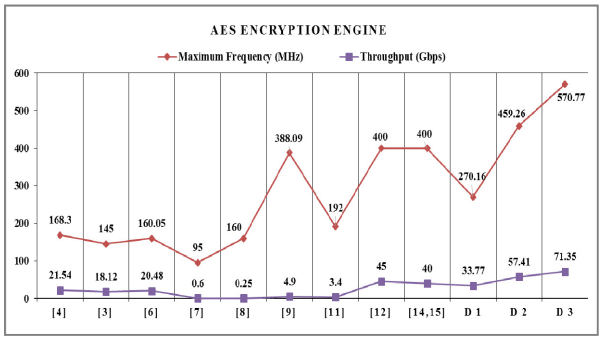
۵- مقایسه و بررسی نتایج

جدول ۲ نتایج طراحی این مقاله را با کارهای آکادمیک و تجاری صورت گرفته در دو بستر سخت‌افزاری FPGA و ASIC در این زمینه مقایسه می‌کند. همچنین به‌منظور مقایسه بهتر نتایج، طرح این مقاله بر روی تراشه‌های Virtex II Pro و Virtex 6 FPGA نیز قرار داده شده است.

جدول ۲: مقایسه کارهای آکادمیک و تجاری انجام‌شده با طراحی این مقاله

Project architecture	Name [ref]	Silicon target	Throughput (Gbps)	Max Frequency (MHz)	FPGA Slice	ASIC gates area	Main application
Crypto coprocessor	AES processor [۴]	VirtexII-Pro FPGA, XC2VP20	۲۱/۵۴	۱۶۸/۳۰	۱۲۴۵۰	-	IPSEC VPN
			۲۱/۶۴	۱۶۹/۱۰	۹۴۴۶	-	
	AES processor [۵]	ASIC 0.18µm	۷۰	۶۰۶	-	-	IPSEC VPN
	[۳]	Virtex-E FPGA, XCV3200E	۱۸/۱۲	۱۴۵	۱۵۱۱۲	-	-
	[۶]	Altera Stratix FPGA	۲۰/۴۸	۱۶۰/۰۵	-	-	VPN
	[۲۰]	ASIC 0.25µm	۰/۸۳	۶۶	-	-	-
Crypto processor one-core and multi-core	CCProc 4 cores [۷]	FPGA XCV4LX200	۰/۶۰	۹۵	۱۸۰۴۵	-	Symmetric encryption accelerator
	CCProc 1 core [۷]	ASIC 0.13µm	۰/۳۹	۲۵۰	-	۹۳KG ۵/۳mm^2	Symmetric encryption accelerator
	Hcrypt [۸]	Virtex 5, FPGA	۰/۲۵	۱۶۰	۷۹۶۰	-	network security VPN
	[۹]	FPGA, Xc6vlx75	۴/۹۰	۳۸۸/۰۹	۱۲۶۴	-	-
	MCCP [۱۱]	FPGA, XCV4SX35	۳/۴۰	۱۹۲	۸۱۱۰	-	Software, radio security
Custom GPP	[۱۲]	Virtex 5 FPGA	۴۵	۴۰۰	-	-	-
ASIC	[۱۰]	ASIC, 45nm	۶۶	۲۱۵۰	-	۱/۱mm^2	-
Helion Technology	AES-GCM [۱۴]	Virtex 6 FPGA (-3)	-	۳۱۲	۵۲۰۰	-	MACsec (802.1AE) (FC-SP)
	AES-GCM [۱۴]	Virtex 7 FPGA (-3)	-	۴۰۰	۵۵۰۰	-	MACsec (802.1AE) (FC-SP)
	AES-CTR [۱۵]	Virtex 7 FPGA (-2)	۴۰	-	۳۴۰۰	-	Ethernet
Proposed Work	Design1, This paper	Virtex II Pro, xc2vp20 -fg676	۳۳/۷۷	۲۷۰/۱۶۸	۹۰۳۱	-	IPSEC, Ethernet
	Design2, This paper	Virtex 6, xc6vsx475t -2ff1759	۵۷/۴۱	۴۵۹/۲۶۳	۳۸۷۹	-	IPSEC, Ethernet
	Design3, This paper	Virtex 7, xc7v585t -3ff1157	۷۱/۳۵	۵۷۰/۷۷۶	۳۶۶۹	-	IPSEC, Ethernet

همچنین در نمودار شکل ۹ برخی از بهترین کارهای انجام‌شده در بستر FPGA با نتایج این مقاله مقایسه شده است.



شکل ۹: مقایسه کارهای انجام‌شده با طراحی این پژوهش در بستر FPGA

لازم به ذکر است که نتایج سه طرح نشان داده‌شده در انتهای جدول ۲ مربوط به یک معماری است با این تفاوت که طرح ارائه‌شده در این مقاله بر روی تراشه‌های مختلف پیاده‌سازی شده است (با مقایسه این سه طرح، تأثیر تکنولوژی‌های متفاوت بر نتایج طراحی نهایی نیز دیده می‌شود). با توجه به نتایج ارائه‌شده در این جدول ۲ و نمودار شکل ۹، دیده می‌شود که معماری ارائه‌شده در این پژوهش بر روی تراشه Virtex II Pro FPGA از طراحی ارائه‌شده در مقاله [۴] بسیار بهتر است (توجه شود که هر دو طرح بر روی یک تراشه پیاده‌سازی شده‌اند). ازجمله تفاوت‌های طرح ارائه‌شده در این مقاله نسبت به طراحی‌های ارائه‌شده در مقاله‌های [۴] و [۵] در تعداد طبقات خط‌لوله و مکان و تعداد رجیسترهای خط‌لوله است. همچنین جهت پیاده‌سازی S-Boxهای طراحی‌شده در این مقاله‌ها از مقدار c در رابطه (۴) فاکتورگیری نشده است. همان‌گونه که قبلاً نیز ذکر شد و نمونه‌ای آورده شد (شکل‌های ۴ و ۵)، تعداد طبقات خط‌لوله و تعداد و مکان قرارگیری رجیسترهای آن، تأثیر فراوانی بر میزان فرکانس مدار و مصرف منابع سخت‌افزاری دارد. از طرفی همواره با افزایش پیچیدگی و افزایش حجم مدار امکان کاهش فرکانس مدار وجود داشته و طراح می‌بایست میان تمامی این مؤلفه‌ها توازنی برقرار کند تا به بهترین نتیجه دست یابد. همچنین لازم به ذکر است که علاوه بر دلایل بیان‌شده، بهبود نتایج ارائه‌شده در پژوهش حاضر نسبت به طراحی‌های مشابه ناشی از تفاوت در نحوه طراحی جزئیات (در سطوح پایین طراحی)، در پیاده‌سازی عملی طرح اصلی و نیز ناشی از اختلاف در نحوه طراحی و پیاده‌سازی باقی اجزا موتور رمزنگار است.

همان‌گونه که در بخش ۲ بیان شد، طراحی ارائه‌شده در [۸] از کمک پردازنده‌های داخلی FPGA استفاده کرده است (هسته‌های MicroBlaze و یا PowerPC) و همچنین معایب چنین طرحی در کاربردهای پرسرعت بیان شد. به‌عنوان یک مقایسه، دیده شد که طراحی این مقاله در فرکانس ۱۶۰ MHz به‌میزان گذردهی ۲۵۶ Mb/s می‌رسد حال آن‌که طرح ارائه‌شده در پژوهش حاضر که کاملاً

به‌صورت سخت‌افزاری پیاده‌سازی شده است در این فرکانس به‌میزان گذردهی Gbps ۲۰ می‌رسد. البته امروزه در نسل جدید FPGAها یعنی خانواده ZYNQ 7000 دو هسته ARM Cortex A9 وجود دارد. هرچند که این هسته‌ها با فرکانس پالس ساعت GHz ۱ قادرند عمل کنند اما باز ماهیت اجرای ترتیبی دستورالعمل‌ها را دارا بوده و استفاده از این ایده برای کاربردهای پرسرعت مناسب ناست. از طرف دیگر استفاده از هسته‌های پردازنده، همواره محدودیت‌هایی را به دنبال دارند. ازجمله این‌که دیگر نمی‌توان طراح نهایی را بر روی هر FPGA دلخواه و درنتیجه بر روی هر بردی قرار داده و استفاده کرد.

گفته شد که نویسنده مقاله [۹] ادعا می‌کند که با موازی کردن صد هسته رمزنگار می‌تواند به گذردهی Gbps ۵۰۰ بر روی تراشه Virtex 6 دست یابد. به‌عنوان یک مقایسه بین طرح ارائه‌شده در پژوهش حاضر با این کار دیده می‌شود که این طرح بر روی Virtex 6 به‌میزان Slice ۱۲۶۴ مصرف کرده است و به‌میزان گذردهی Gbps ۴/۹ دست یافته است. طبق گفته نویسنده این مقاله می‌توان به تعداد ۱۰۰ موتور رمز از طراحی فوق را در یک Virtex 6 قرار داده و به گذردهی Gbps ۵۰۰ دست یافت. سیستم نهایی از چنین طراحی به‌میزان تقریبی Slices ۱۲۶۴۰۰ = ۱۰۰ × ۱۲۶۴ مصرف می‌کند. درحالی‌که یک پیاده‌سازی از طراحی پژوهش حاضر بر روی Virtex 6 به‌میزان Slice ۳۸۷۹ مصرف کرده و به‌میزان گذردهی Gbps ۵۷/۴۱ دست یافته است. تنها با موازی کردن ۱۰ موتور رمز از طراحی این پژوهش بر روی Virtex 6 می‌توان به‌میزان گذردهی Gbps ۵۷۴ با میزان تقریبی مصرف منابع سخت‌افزاری Slices ۳۸۷۹۰ = ۱۰ × ۳۸۷۹ دست یافت. با مقایسه نتایج دیده می‌شود که طرح ارائه‌شده در پژوهش حاضر جهت طراحی سیستمی با Gbps ۵۰۰ گذردهی، به‌میزان ٪ ۳۰/۶ منابع سخت‌افزاری کمتری نسبت به طرح ارائه‌شده در مقاله [۹] مصرف می‌کند.

نویسنده مقاله [۱۲] بیان می‌کند در حالتی که FastCrypto به بیشترین گذردهی Gbps ۴۵ در فرکانس MHz ۴۰۰ می‌رسد، پردازنده رمزنگار شامل ۴ موتور رمز AES است که به‌صورت موازی با یکدیگر کار کرده و هرکدام دارای ۲ طبقه خط‌لوله در هر دور می‌باشند. همچنین این موتورها با فرکانس MHz ۱۰۰ عمل می‌کنند. این طرح با موازی کردن چهار موتور رمزنگار AES توانسته میزان بازدهی را به‌میزان قابل توجهی بالا ببرد، اما در مقابل این عمل موجب شده که منابع سخت‌افزاری زیادی مصرف کرده و نیز میزان توان مصرفی افزایش یابد که موجب کاهش کارایی نسبت به [۵] شود. لازم به ذکر است که در این سیستم تنها فرکانس کمک پردازنده‌ها برابر با MHz ۴۰۰ است. به‌عنوان یک مقایسه با طرح ارائه‌شده در این پژوهش، درصورتی‌که چهار موتور رمزنگار طراحی‌شده در این پژوهش را با یکدیگر موازی کنیم و هر چهار موتور با فرکانس MHz ۱۰۰ عمل کنند سیستم نهایی به‌میزان گذردهی Gbps ۵۰ دست می‌یابد (در اینجا تنها قسمت موتورهای رمز که هر دو به‌صورت سخت‌افزاری و بدون استفاده از پردازنده‌های همه‌منظوره طراحی شده‌اند، مقایسه شد). در این مقاله نویسنده میزان منابع مصرفی توسط موتورهای رمزنگار را بیان نکرده است و از جهت میزان منابع مصرفی نمی‌توان مقایسه‌ای انجام داد.

دیده شد که طراحی این پژوهش بر روی دو تراشه Virtex 6 FPGA و Virtex 7 FPGA از طراحی‌های آکادمیک و برخی از طراحی‌های تجاری نظیرشان چه ازنظر فرکانس و چه ازنظر مصرف منابع سخت‌افزاری بهتر است. همچنین طرحی این پژوهش بر روی Virtex 7 از طرح تجاری ارائه‌شده در [۱۵] تنها ازنظر بیشترین فرکانس کاری مدار بهتر است. چراکه برای کاربردهای تجاری همواره تلاش‌هایی جهت بهینه‌سازی در سطح پیاده‌سازی و کاهش Sliceهای مصرفی صورت می‌گیرد (تلاشی که از عهده این مقاله خارج بود).

۶- نتیجه‌گیری

ازآنجاکه امروزه سرعت نقش مهمی در ارائه سرویس‌های شبکه ایفا می‌کند و در کنار آن امنیت اطلاعات امری ضروری است و همچنین زیرساخت‌های شبکه از سرعت‌های بالا پشتیبانی می‌کنند (مثل خطوط فیبر نوری)، در این مقاله به طراحی و پیاده‌سازی یک موتور رمزنگار، مبتنی بر پروتکل رمز AES، در بستر FPGA پرداخته شد. همچنین این طرح علاوه بر دستیابی به بیشترین فرکانس ممکن میزان منابع کمی مصرف می‌کند. به‌گونه‌ای که طرح ارائه‌شده در این مقاله قادر است بر روی تراشه Virtex 7 FPGA تنها به‌میزان Slices ۳۶۶۹ مصرف کرده و به بیشترین گذردهی Gbps ۷۱/۳۵ در فرکانس MHz ۵۷۰/۷۷۶ دست یابد.

لازم به ذکر است که می‌توان از این طرح در طراحی مسیریاب‌های شبکه، بخصوص در طراحی مسیریاب‌های شبکه‌های فیبر نوری و شبکه‌های اترنت پرسرعت که نیاز به محرمانگی اطلاعات دارند، استفاده کرد. درواقع ازآنجاکه پروتکلIPSec برای ایجاد امنیت در لایه IP وضع شده است و می‌تواند به‌طور شفاف همه ترافیک را در سطح لایه IP رمز کرده و به آن اصالت ببخشد، از این طرح می‌توان جهت طراحی هسته IPSec به‌صورت سخت‌افزاری و باقابلیت پشتیبانی از سرعت‌های بالا استفاده کرد.

در پایان به‌منظور ادامه کار و جهت دستیابی به بیشترین گذردهی، کاهش منابع مصرفی و نیز کاهش توان مصرفی، پیشنهاد می‌شود این طراحی در سطح پیاده‌سازی بر روی سخت‌افزار بهینه شده و همچنین بر روی تراشه‌های ASIC پیاده‌سازی شود.

پیوست‌ها

از آنجا که این پژوهش به پیاده‌سازی سخت‌افزاری رمزنگار AES می‌پردازد و بخش S-Box الگوریتم را به‌صورت مدار ترکیبی پیاده‌سازی می‌کند، در این بخش مدارهای منطقی استفاده‌شده جهت پیاده‌سازی سخت‌افزاری بلوک‌های به‌کاررفته در شکل ۲ به همراه مبانی ریاضی آن‌ها مورد بررسی قرار می‌گیرد.

الف) تبدیل عناصر $GF(2^8)$ به دو عنصر متناظر در $GF(2^4)$

همان‌گونه که پیش‌ازاین بیان شد، ازآنجاکه انجام عملیات معکوس حاصل‌ضرب در میدان $GF(2^8)$ بسیار پیچیده است، توصیه می‌شود تا این میدان به میدان $GF(2^4)$ تبدیل شده و عملیات مورد نیاز بر روی این میدان کوچک‌تر صورت گیرد. به‌این‌علت اولین قدم در انجام محاسبات تبدیل این دو میدان به یکدیگر است. برای انجام این تبدیل و معکوس آن از ماتریس‌های نشان داده‌شده در شکل ۱۰ استفاده می‌شود.

$$\delta\times q=\begin{pmatrix}1&0&1&0&0&0&0&0\\1&1&0&1&1&1&1&0\\1&0&1&0&1&1&0&0\\1&0&1&0&1&1&1&0\\1&1&0&0&0&1&1&0\\1&0&0&1&1&1&1&0\\0&1&0&1&0&0&1&0\\0&1&0&0&0&0&1&1\end{pmatrix}\times\begin{pmatrix}q_7\\q_6\\q_5\\q_4\\q_3\\q_2\\q_1\\q_0\end{pmatrix}\quad \delta^{-1}\times q=\begin{pmatrix}1&1&1&0&0&0&1&0\\0&1&0&0&0&1&0&0\\0&1&1&0&0&0&1&0\\0&1&1&1&0&1&1&0\\0&0&1&1&1&1&1&0\\1&0&0&1&1&1&1&0\\0&0&1&1&0&0&0&0\\0&1&1&1&0&1&0&1\end{pmatrix}\times\begin{pmatrix}q_7\\q_6\\q_5\\q_4\\q_3\\q_2\\q_1\\q_0\end{pmatrix}$$

شکل ۱۰: ماتریس‌های موردنیاز برای تبدیل عناصر از $GF(2^8)$ به $GF(2^4)$ و برعکس [۱۹]

همان‌گونه که قبلاً ذکر شد، کوچک‌ترین واحدی که در این الگوریتم بر روی آن پردازش صورت می‌گیرد برابر با یک بایت است. در ماتریس‌های نشان داده‌شده در این شکل بیت $q7$ مشخص‌کننده پرارزش‌ترین بیت و $q0$ مشخص‌کننده کم‌ارزش‌ترین بیت داده ورودی است. ماتریس نشان داده‌شده در شکل ۱۰ را می‌توان در سطح گیت‌های منطقی مطابق مقادیر نشان داده‌شده در شکل ۱۱ پیاده‌سازی نمود.

$$\delta\times q=\begin{pmatrix}q_7\oplus q_5\\q_7\oplus q_6\oplus q_4\oplus q_3\oplus q_2\oplus q_1\\q_7\oplus q_5\oplus q_3\oplus q_2\\q_7\oplus q_5\oplus q_3\oplus q_2\oplus q_1\\q_7\oplus q_6\oplus q_2\oplus q_1\\q_7\oplus q_4\oplus q_3\oplus q_2\oplus q_1\\q_6\oplus q_4\oplus q_1\\q_6\oplus q_1\oplus q_0\end{pmatrix}\quad \delta^{-1}\times q=\begin{pmatrix}q_7\oplus q_6\oplus q_5\oplus q_1\\q_6\oplus q_2\\q_6\oplus q_5\oplus q_1\\q_6\oplus q_5\oplus q_4\oplus q_2\oplus q_1\\q_5\oplus q_4\oplus q_3\oplus q_2\oplus q_1\\q_7\oplus q_4\oplus q_3\oplus q_2\oplus q_1\\q_5\oplus q_4\\q_6\oplus q_5\oplus q_4\oplus q_2\oplus q_0\end{pmatrix}$$

شکل ۱۱: نحوه پیاده‌سازی تبدیل عناصر از $GF(2^8)$ به $GF(2^4)$ به‌صورت مدار ترکیبی [۱۹]

ب) انجام عملیات مربع در میدان $GF(2^4)$

با فرض این‌که $k=q^2$ باشد و k و q هر دو المان‌هایی از $GF(2^4)$ باشند، هریک را می‌توان به فرمت باینری به‌صورت $k=\{k_3\,k_2\,k_1\,k_0\}_2$ و $q=\{q_3\,q_2\,q_1\,q_0\}_2$ بیان کرد، بنابراین:

$$k=\left(\underbrace{k_3k_2}_{k_H}\underbrace{k_1k_0}_{k_L}\right)=k_Hx+k_L=\left(\underbrace{q_3q_2}_{q_H}\underbrace{q_1q_0}_{q_L}\right)^2=(q_Hx+q_L)^2 \quad (۷)$$

$$k=q_H{}^2x^2+q_Hq_Lx+q_Hq_Lx+q_L{}^2=q_H{}^2x^2+q_L{}^2$$

مقدار x^2 را با استفاده از چندجمله‌ای تحویل‌ناپذیر نشان داده‌شده در رابطه (۳) ($x^2+x+\varphi$) می‌توان به‌صورت $x^2=x+\varphi$ بازنویسی کرد. بنابراین معادلات (۸) حاصل می‌شود.

$$k=q_H{}^2(x+\varphi)+q_L{}^2$$
$$k=\underbrace{q_H{}^2}_{k_H}x+\left(\underbrace{q_H{}^2\varphi+q_L{}^2}_{k_L}\right)\in GF(2)^2 \quad (۸)$$

بنابراین مقدار K_H برابر است با:

$$k_H=q_H{}^2=(q_3q_2)^2=(q_3x+q_2)^2 \quad (۹)$$
$$k_H=q_3{}^2x^2+q_3q_2x+q_3q_2x+q_2{}^2=q_3x^2+q_2$$

مقدار x^2 را با استفاده از چندجمله‌ای تحویل‌ناپذیر نشان داده‌شده در رابطه (۳) (x^2+x+1) می‌توان به‌صورت $x^2=x+1$ بازنویسی کرد. بنابراین معادلات (۱۰) حاصل می‌شود.

$$k_H=q_3(x+1)+q_2 \quad (۱۰)$$
$$k_3x+k_2=q_3x+(q_2+q_3)\in GF(2)$$

همچنین مقدار K_L با استفاده از روابط (۱۱) قابل محاسبه است. همان‌گونه که قبلاً ذکر شده است $\varphi=\{10\}_2$.

$$k_L=q_H{}^2\varphi+b_L{}^2=(q_3q_2)^2\{10\}_2+(q_1q_0)^2$$
$$k_L=(q_3x+q_2)^2(\{1\}_2x+0)+(q_1x+q_0)^2 \quad (۱۱)$$
$$k_L=(q_3{}^2x^2+q_2q_3x+q_2q_3x+q_2{}^2)(x)$$
$$+(q_1{}^2x^2+q_0q_1x+q_0q_1x+q_0{}^2)$$
$$k_L=q_3x^3+q_2x+q_1x^2+q_0$$

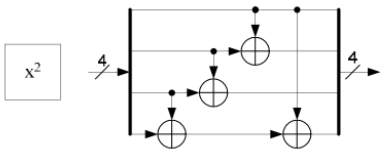
همان‌گونه که گفته شد مقدار x^2 را می‌توان به‌صورت $x^2=x+1$ نوشت. پس ازآنجاکه $x^3=x.x^2$ بنابراین $x^3=x^2+x=x+1+x=1$ بنابراین معادلات قبل را می‌توان به‌صورت روابط (۱۲) بازنویسی کرد.

$$k_L=q_3(1)+q_2x+q_1(x+1)+q_0 \quad (۱۲)$$
$$k_1x+k_0=(q_2+q_1)x+(q_3+q_1+q_0)\in GF(2)$$

پس در نهایت با توجه به روابط (۱۰) و (۱۲) مقدار مربع ورودی در میدان $FG(2^4)$ با استفاده از روابط (۱۳) قابل محاسبه است:

$$k_3=q_3$$
$$k_2=q_3\oplus q_2 \quad (۱۳)$$
$$k_1=q_2\oplus q_1$$
$$k_0=q_3\oplus q_1\oplus q_0$$

همچنین روابط (۱۳) را می‌توان به‌صورت مدار ترکیبی پیاده‌سازی نمود. شکل ۱۲ این مدار را نشان می‌دهد.



شکل ۱۲: مدار ترکیبی محاسبه مربع در میدان $GF(2^4)$ [۲۱]

پ) انجام عملیات ضرب با مقدار ثابت λ در میدان $GF(2^4)$

با فرض این‌که $k=q\lambda$ باشد و نیز $k=\{k_3\,k_2\,k_1\,k_0\}_2$ و $q=\{q_3\,q_2\,q_1\,q_0\}_2$ و $\lambda=\{1100\}_2$ المان‌هایی از $GF(2^4)$ باشند آنگاه داریم:

$$\begin{aligned} k &= \left(\underbrace{k_3k_2}_{k_H}\ \underbrace{k_1k_0}_{k_L}\right) = k_H x + k_L = \left(\underbrace{q_3q_2}_{q_H}\ \underbrace{q_1q_0}_{q_L}\right)\left(\underbrace{11}_{\lambda_H}\ \underbrace{00}_{\lambda_L}\right) \\ k &= (q_H x + q_L)(\lambda_H x + \lambda_L) \quad \lambda_L \text{ can be cancelled out since } \lambda_L = \{00\}_2 \\ k &= q_H\lambda_H x^2 + q_L\lambda_H x \end{aligned} \tag{۱۴}$$

مقدار x^2 را با استفاده از چندجمله‌ای تحویل‌ناپذیر نشان داده‌شده در رابطه (۳) ($x^2+x+\varphi$) را می‌توان به‌صورت $x^2=x+\varphi$ بازنویسی کرد. بنابراین معادلات (۱۵) حاصل می‌شود.

$$\begin{aligned} k &= q_H\lambda_H(x+\varphi) + q_L\lambda_H x \\ k &= \underbrace{(q_H\lambda_H + q_L\lambda_H)}_{k_H} x + \underbrace{(q_H\lambda_H\varphi)}_{k_L} \in GF(2^2) \end{aligned} \tag{۱۵}$$

همچنین با توجه به توضیحات بخش قبل می‌توان k_H و k_L را مطابق با روابط (۱۶) در میدان GF(2) تجزیه کرد.

$$\begin{aligned} k_H &= q_H\lambda_H + q_L\lambda_H \\ k_H &= (q_3q_2)(11_2) + (q_1q_0)(11_2) \\ k_H &= (q_3x+q_2)(x+1) + (q_1x+q_0)(x+1) \\ k_H &= q_3x^2 + (q_3+q_2)x + q_2 + q_1x^2 + (q_1+q_0)x + q_0 \end{aligned} \tag{۱۶}$$

پس با توجه به $x^2=x+1$ خواهیم داشت:

$$\begin{aligned} k_H &= q_3(x+1) + (q_3+q_2)x + q_2 + q_1(x+1) + (q_1+q_0)x + q_0 \\ k_H &= (q_3+q_3+q_2+q_1+q_1+q_0)x + (q_3+q_2+q_1+q_0) \\ k_3x + k_2 &= (q_2+q_0)x + (q_3+q_2+q_1+q_0) \in GF(2) \end{aligned} \tag{۱۷}$$

روند مشابهی نیز برای محاسبه K_L و تجزیه آن به میدان GF(2) صورت می‌گیرد.

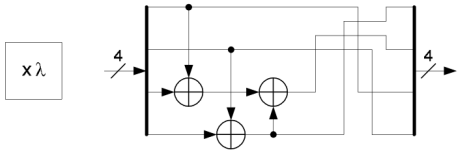
$$\begin{aligned} k_L &= q_H\lambda_H\varphi \\ k_L &= (q_3q_2)(11_2)(10_2) \\ k_L &= (q_3x+q_2)(x+1)(x) \\ k_L &= q_3x^3 + q_2x^2 + q_3x^2 + q_2x \end{aligned} \tag{۱۸}$$

پس با توجه به $x^2=x+1$ و $x^3=1$ خواهیم داشت:

$$\begin{aligned} k_L &= q_3(1) + q_2(x+1) + q_3(x+1) + q_2x \\ k_L &= (q_3+q_2+q_2)x + (q_3+q_3+q_2) \\ k_1x + k_0 &= (q_3)x + (q_2) \in GF(2) \end{aligned} \tag{۱۹}$$

بنابراین با توجه به روابط (۱۷) و (۱۹) محاسبه حاصل‌ضرب با مقدار ثابت λ از طریق معادلات (۲۰) در سطح سخت‌افزار قابل محاسبه است. همچنین شکل ۱۳ مدار ترکیبی این ضرب را نشان می‌دهد.

$$\begin{aligned} k_3 &= q_2 \oplus q_0 \\ k_2 &= q_3 \oplus q_2 \oplus q_1 \oplus q_0 \\ k_1 &= q_3 \\ k_0 &= q_2 \end{aligned} \tag{۲۰}$$



شکل ۱۳: مدار ترکیبی محاسبه ضرب با مقدار ثابت λ در میدان $GF(2^4)$ [۲۱]

ت) انجام عملیات ضرب در میدان $GF(2^4)$

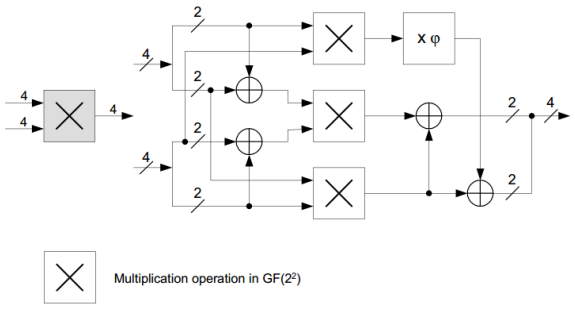
با فرض این‌که $k=qw$ باشد و نیز $k=\{k_3\,k_2\,k_1\,k_0\}_2$ و $q=\{q_3\,q_2\,q_1\,q_0\}_2$ و $w=\{w_3\,w_2\,w_1\,w_0\}_2$ المان‌هایی از $GF(2^4)$ باشند آنگاه داریم:

$$\begin{aligned} k &= \left(\underbrace{k_3k_2}_{k_H}\ \underbrace{k_1k_0}_{k_L}\right) = k_H x + k_L = \left(\underbrace{q_3q_2}_{q_H}\ \underbrace{q_1q_0}_{q_L}\right)\left(\underbrace{w_3w_2}_{w_H}\ \underbrace{w_1w_0}_{w_L}\right) \\ &= (q_H x + q_L)(w_H x + w_L) \\ k &= (q_Hw_H)x^2 + (q_Hw_L + q_Lw_H)x + q_Lw_L \end{aligned} \tag{۲۱}$$

با توجه به $x^2=x+\varphi$ خواهیم داشت:

$$\begin{aligned} k &= (q_Hw_H)(x+\varphi) + (q_Hw_L + q_Lw_H)x + q_Lw_L \\ k &= k_Hx + k_L = (q_Hw_H + q_Hw_L + q_Lw_H)x \\ &\quad + q_Hw_H\varphi + q_Lw_L \in GF(2^2) \end{aligned} \tag{۲۲}$$

معادلات بالا در میدان $GF(2^2)$ است. درصورتی‌که مداری برای انجام عملیات ضرب در این میدان در اختیار باشد، رابطه بالا را می‌توان با استفاده از مدار نشان داده‌شده در شکل ۱۴ به‌صورت سخت‌افزاری پیاده‌سازی کرد. همچنین مدار ضرب با مقدار ثابت در این میدان در بخش قبل توصیف شد.



شکل ۱۴: مدار ترکیبی محاسبه ضرب در میدان $GF(2^4)$ [۲۱]

همچنین به‌دلیل محدود بودن دامنه اعداد در میدان $GF(2^4)$ می‌توان نتایج حاصل از این ضرب را از پیش محاسبه کرده و در جدول‌هایی ذخیره و استفاده کرد. شکل ۱۵ نتایج پیش‌محاسبه ضرب دو المان در میدان $GF(2^4)$ را نشان می‌دهد.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
2	0	2	3	1	8	a	b	9	c	e	f	d	4	6	7	5
3	0	3	1	2	c	f	d	e	4	7	5	6	8	b	9	a
4	0	4	8	c	6	2	e	a	b	f	3	7	d	9	5	1
5	0	5	a	f	2	7	8	d	3	6	9	c	1	4	b	e
6	0	6	b	d	e	8	5	3	7	1	c	a	9	f	2	4
7	0	7	9	e	a	d	3	4	f	8	6	1	5	2	c	b
8	0	8	c	4	b	3	7	f	d	5	1	9	6	e	a	2
9	0	9	e	7	f	6	1	8	5	c	b	2	a	3	4	d
a	0	a	f	5	3	9	c	6	1	b	e	4	2	8	d	7
b	0	b	d	6	7	c	a	1	9	2	4	f	e	5	3	8
c	0	c	4	8	d	1	9	5	6	a	2	e	b	7	f	3
d	0	d	6	b	9	4	f	2	e	3	8	5	7	a	1	c
e	0	e	7	9	5	b	2	c	a	4	d	3	f	1	8	6
f	0	f	5	a	1	e	4	b	2	d	7	8	3	c	6	9

شکل ۱۵: نتایج پیش‌محاسبه ضرب دو المان در میدان $GF(2^4)$ [۲۱]

همچنین واضح است که نتایج حاصل از ضرب با مقدار ثابت λ و محاسبه مقدار مربع داده ورودی را نیز می‌توان با استفاده از این جدول به‌دست آورد.

ث) انجام عملیات ضرب در میدان $GF(2^2)$

با فرض این‌که $k=qw$ باشد و نیز $k=\{k_1\ k_0\}_2$ و $q=\{q_1\ q_0\}_2$ و $w=\{w_1\ w_0\}_2$ المان‌هایی از $GF(2^2)$ باشند آنگاه داریم:

$$k=(k_1k_0)=k_1x+k_0=(q_1q_0)(w_1w_0)=(q_1x+q_0)(w_1x+w_0) \quad (۲۳)$$
$$k=q_1w_1x^2+q_0w_1x+q_1w_0x+q_0w_0$$

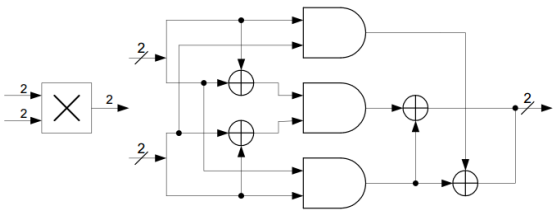
پس با توجه به $x^2=x+1$ خواهیم داشت:

$$k=q_1w_1(x+1)+q_0w_1x+q_1w_0x+q_0w_0 \quad (۲۴)$$
$$k_1x+k_0=(q_1w_1+q_0w_1+q_1w_0)x+(q_1w_1+q_0w_0)\in GF(2)$$

معادله بالا در میدان $GF(2)$ است و با توجه به آن مقادیر زیر برای k_1 و k_0 حاصل می‌شود:

$$k_1=q_1w_1\oplus q_0w_1\oplus q_1w_0 \quad (۲۵)$$
$$k_0=q_1w_1\oplus q_1w_0$$

همان‌گونه که دیده می‌شود این روابط تنها با استفاده از گیت‌های AND و XOR قابل پیاده‌سازی سخت‌افزاری است. شکل ۱۶ مدار ترکیبی حاصل از پیاده‌سازی این معادلات را به‌منظور محاسبه ضرب در میدان $GF(2)$ نشان می‌دهد.



شکل ۱۶: مدار ترکیبی محاسبه ضرب در میدان GF(2) [۲۱]

همان‌گونه که دیده می‌شود مدار ترکیبی نشان داده‌شده در شکل ۱۵ با رابطه نشان داده‌شده برای k_1 متفاوت است. قابل اثبات است معادله k_1 در رابطه (۲۵) از طریق روابط (۲۶) به‌دست آمده است.

$$k_1=(q_1\oplus q_0)(w_1\oplus w_0)\oplus(q_0w_0) \quad (۲۶)$$
$$k_1=(q_1w_1)\oplus(q_0w_1)\oplus(q_1w_0)\oplus(q_0w_0)\oplus(q_0w_0)$$
$$k_1=(q_1w_1)\oplus(q_0w_1)\oplus(q_1w_0)$$

ج) انجام عملیات ضرب با مقدار ثابت φ در میدان $GF(2^2)$

با فرض این‌که $k=q\varphi$ باشد و نیز $k=\{k_1\ k_0\}_2$ و $q=\{q_1\ q_0\}_2$ و $\varphi=\{10\}_2$ المان‌هایی از $GF(2^2)$ باشند آنگاه داریم:

$$k=k_1x+k_0=(q_1q_0)(10_2)=(q_1x+q_0)(x) \quad (۲۷)$$
$$k=q_1x^2+q_0x$$

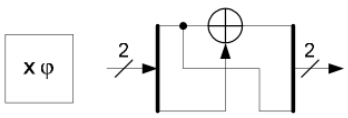
پس با توجه به $x^2=x+1$ خواهیم داشت:

$$k=q_1(x+1)+q_0x \quad (۲۸)$$
$$k=(q_1+q_0)x+(q_1)\in GF(2)$$

بنابراین با توجه به فرمول بالا مقادیر k_1 و k_0 برابرند با:

$$k_1=q_1\oplus q_0 \quad (۲۹)$$
$$k_0=q_1$$

با توجه به این روابط مدار ترکیبی برای محاسبه ضرب با مقدار ثابت φ در میدان $GF(2)$ مطابق شکل ۱۷ قابل پیاده‌سازی است.



شکل ۱۷: مدار ترکیبی محاسبه ضرب با مقدار ثابت φ در میدان GF(2) [۲۱]

چ) انجام عملیات معکوس حاصل‌ضرب در میدان $GF(2^4)$

نویسنده مقاله [۲۱] فرمولی را به‌منظور محاسبه معکوس حاصل‌ضرب q ارائه داده است. با فرض این‌که $q^{-1}=\{q_3^{-1},\ q_2^{-1},q_1^{-1},q_0^{-1}\}$ باشد و q المانی از $GF(2^4)$ باشد آنگاه مقدار هر بیت به‌صورت مجزا از دیگر بیت‌ها با استفاده از روابط (۳۰) قابل محاسبه است.

$$q_3^{-1}=q_3\oplus q_3q_2q_1\oplus q_3q_0\oplus q_2$$
$$q_2^{-1}=q_3q_2q_1\oplus q_3q_2q_0\oplus q_3q_0\oplus q_2\oplus q_2q_1 \quad (۳۰)$$
$$q_1^{-1}=q_3\oplus q_3q_2q_1\oplus q_3q_1q_0\oplus q_2\oplus q_2q_0\oplus q_1$$
$$q_0^{-1}=q_3q_2q_1\oplus q_3q_2q_0\oplus q_3q_1\oplus q_3q_1q_0\oplus q_3q_0$$
$$\oplus q_2\oplus q_2q_1\oplus q_2q_1q_0\oplus q_1\oplus q_0$$

همچنین شکل ۱۸ نتایج پیش محاسبه معکوس حاصل‌ضرب در میدان $GF(24)$ را نشان می‌دهد.

q	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
q^{-1}	0	1	3	2	f	c	9	b	a	6	8	7	5	e	d	4

شکل ۱۸: نتایج پیش محاسبه معکوس حاصل‌ضرب در میدان $GF(2^4)$ [۲۱]

مراجع

[۱] ذاکرحسینی و ملکیان، *امنیت داده‌ها*، ویراستهٔ باباخانی، ویرایش دوم، تهران، مؤسسه علمی - فرهنگی نص، ۱۳۸۷.

[۲] عطائی، روزبهانی، سعیدی و برنج‌کوب، «طراحی و پیاده‌سازی سخت‌افزاری رمزگذار و رمزگشای AES با طرح خط‌لوله فیدبک‌دار». *سومین کنفرانس انجمن رمز ایران*، صفحات ۳۶۱-۳۷۰، دانشگاه صنعتی اصفهان، ۱۳۸۴.

[3] F. Standaert, G. Rouvroy, J. Quisquater, and J. Legat, "Efficient implementation of rijndael encryption in reconfigurable hardware : improvements and design tradeoffs," *Cryptographic Hardware and Embedded Systems - CHES*, Springer Berlin Heidelberg, vol. 2779, pp. 334-350, 2003.

[4] A. Hodjat, and I. Verbauwhede, "A 21.54 Gbits/s fully pipelined AES processor on FPGA," *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines,* pp. 308-309, 2004.

[5] A. Hodjat, and I. Verbauwhede, "Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors," *IEEE Trans. Comput.*, vol. 55, no. 4, pp. 366-372, 2006.

[6] H. Qin, T. Sasao, and Y. Iguchi, "An FPGA design of AES encryption circuit with 128-bit keys," *Proceedings of the 15th ACM Great Lakes symposium on VLSI - GLSVSLI '05*, pp. 147-151, 2005.

[7] D. Theodoropoulos, A. Siskos, and D. Pnevmatikatos, "CCproc : a custom VLIW cryptography co-processor for symmetric-key ciphers," *Reconfigurable Computing: Architectures, Tools and Applications*, pp. 318-323, 2009.
[8] L. Gaspar, V. Fischer, F. Bernard, L. Bossuet, and P. Cotret, "Hcrypt: a novel concept of crypto-processor with secured key management," *International Conference on Reconfigurable Computing and FPGAs*, pp. 280-285, 2010.
[9] A. Bouhraoua, "Design feasibility study for a 500 Gbits/s advanced encryption standard cipher/decipher engine," *IET Comput. Digit. Tech*, vol. 4, no. 4, pp. 334-348, 2010.
[10] S.K. Mathew, F. Sheikh, M. Kounavis, S. Gueron, A. Agarwal, S.K. Hsu, H. Kaul, M.A. Anders, and R.K. Krishnamurthy, "53 Gbps native GF(2^4)^2 composite-field AES-encrypt/decrypt accelerator for content-protection in 45 nm high-performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 767-776, 2011.
[11] M. Grand, L. Bossuet, B. Gal, G. Gogniat, and D. Dallet, "Design and implementation of a multi-core crypto-processor for software defined radios," *Reconfigurable Computing: Architectures, Tools and Applications SE - 5*, Springer Berlin Heidelberg, vol. 6578, pp. 29-40, 2011.
[12] M.I. Soliman, and G.Y. Abozaid, "FPGA implementation and performance evaluation of a high throughput crypto coprocessor," *J. Parallel Distrib. Comput.*, vol. 71, no. 8, pp. 1075-1084, 2011.
[13] L. Bossuet, M. Grand, L. Gaspar, V. Fischer, and G. Gogniat, "Architectures of flexible symmetric key crypto engines—a survey," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 1-32, 2013.
[14] Helion. *AES-GCM (galois counter mode) core for FPGA (xilinx, altera, microsemi, lattice) and ASIC - helion technology*, http://www.heliontech.com/aes_gcm.htm/, 2014.
[15] Helion. *AES core - Xilinx, Altera, Microsemi, Lattice and ASIC - Helion Technology*, http://www.heliontech.com/-aes_giga.htm/, 2014.
[16] FIPS-197, *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*, Federal Information Processing Standards Publication 197. United States National Institute of Standards and Technology (NIST). November 26, 2001, Retrieved October 2, 2012.
[17] M. Khalil, and M. Hani, *Verilog Design of a 256-Bit AES Crypto Processor Core*, Universiti Teknologi Malaysia, Faculty of Electrical Engineering, 2007.
[18] V. Rijmen, *Efficient Implementation of the Rijndael s-box, Kathol*, Univ. Leuven, Dept. ESAT. Belgium, 2000.
[19] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact rijndael hardware architecture with s-box optimization," *Adv. Cryptol. — ASIACRYPT 2001 SE - 15*, vol. 2248, pp. 239-254, 2001.
[20] C.P. Su, C.L. Horng, C.T. Huang, and C.W. Wu, "A configurable AES processor for enhanced security," *Proceedings of the 2005 conference on Asia South Pacific design automation - ASP-DAC '05*, pp. 361-366, 2005.
[21] X. Zhang, and K.K. Parhi, "High-speed VLSI architectures for the AES algorithm," *Very Large Scale Integr. Syst. IEEE Trans.*, vol. 12, no. 9, pp. 957–967, 2004.

زیرنویس‌ها

[1] Cryptography
[2] Advanced Encryption Standard
[3] Data Encryption Standard
[4] Rijndael
[5] National Institute of Standards and Technology
[6] General Purpose Processor
[7] Field Programmable Gate Array
[8] Application Specific Integrated Circuit
[9] Byte Substitution
[10] Throughput
[11] Lookup Table
[12] Master Key
[13] Round Key