پیاده‌سازی سخت افزاری پر سرعت ضرب نقطه‌ای بر روی خم‌های باینری ادواردز و هشیان کلی‌شده

بهرام رشیدی[۱] **محمد عابدینی**[۲]

۱- دانشیار- دانشکده فنی و مهندسی- دانشگاه آیت الله العظمی بروجردی (ره)- آزمایشگاه تحقیقاتی میکروالکترونیک-بروجرد- ایران
b.rashidi@abru.ac.ir

۲- دانشیار- دانشکده فنی و مهندسی- دانشگاه آیت الله العظمی بروجردی (ره)- آزمایشگاه تحقیقاتی میکروالکترونیک-بروجرد- ایران

چکیده: در این مقاله پیاده‌سازی ساختارهایی پر سرعت برای محاسبه ضرب نقطه‌ای برای خم‌های بیضوی باینری ادواردز و هشیان کلی‌شده بر اساس الگوریتم نردبان مُنتگومری ارائه شده است. در ساختار پیشنهادی برای کاهش تعداد سیکل ساعت، ضرب‌کننده‌های میدانی برای انجام محاسبات جمع دو نقطه و دو برابر کردن یک نقطه به صورت موازی استفاده شده‌اند. ضرب-کننده‌ی میدانی استفاده شده با پایه نرمال گوسی می‌باشد، که به صورت خط لوله‌ای و دارای ساختار رقمی-سریال در پایه نرمال گوسی است. این ضرب‌کننده دارای ساختاری منظم با مسیر بحرانی کم و سخت‌افزار مصرفی مناسب می‌باشد. در ساختار ارائه شده عمل ضرب نقطه‌ای برای خم‌های بیضوی باینری ادواردز در دو حالت کلی و خاص آن به ترتیب از چهار و سه ضرب‌کننده‌ی میدانی استفاده شده است. همچنین از سه ضرب‌کننده‌ی میدانی برای خم باینری هشیان کلی‌شده استفاده شده است. ضرب‌کننده‌ها در طول محاسبات برای کاهش تعداد سیکل ساعت، زمان‌بندی و به اشتراک گذاشته شده‌اند. نتایج پیاده سازی معماری های پیشنهادی بر روی Virtex-5 XC5VLX110 FPGA نشان می دهد که زمان اجرای ضرب نقطه برای خم‌های بیضوی باینری ادواردز و هشیان کلی‌شده بر روی میدان های متناهی $GF(2^{163})$ و $GF(2^{233})$ به ترتیب ۸,۶۲ µs و ۱۱,۰۳ µs است. نتایج نشان می دهد که ساختارهای پیشنهادی، در مقایسه با ساختارهای قبلی، از نظر پارامترهای مانند تاخیر و بازدهی بهبود یافته اند.

واژه‌های کلیدی: سیستم رمزنگاری خم بیضوی، ضرب نقطه ای، ضرب کننده در پایه نرمال گوسی، رقمی-سریال، خم‌های بیضوی باینری ادواردز، خم های باینری هشیان کلی‌شده.

نوع مقاله: پژوهشی

DOI: 10.52547/jiaeee.21.1.105

تاریخ ارسال مقاله: ۱۴۰۱/۰۳/۱۱
تاریخ پذیرش مشروط مقاله: ۱۴۰۱/۰۸/۱۵
تاریخ پذیرش مقاله: ۱۴۰۲/۰۱/۱۸
نام نویسنده‌ی مسئول: دکتر بهرام رشیدی
نشانی نویسنده‌ی مسئول: دانشگاه آیت الله العظمی بروجردی (ره)- آزمایشگاه تحقیقاتی میکروالکترونیک- بروجرد- ایران

۱- مقدمه

رمزنگاری خم بیضوی[1] در اواسط دهه ۱۹۸۰ به طور مستقل توسط نیل کوبلیتز [۱] و ویکتور میلر [۲] ارائه شد. رمزنگاری خم بیضوی یک سیستم رمزنگاری کلید عمومی با اندازه کلید نسبتاً کوچک است که در آن از خم های بیضوی روی میدان های متناهی (میدان های Galois (GF)) اعمال می شود. به عبارت دیگر در رمزنگاری خم بیضوی استفاده از خم های بیضوی تعریف شده در میدان های متناهی برای طرح های رمزنگاری، از جمله مبادله کلید، رمزگذاری و امضای دیجیتال پیشنهاد شده است. ضرب نقطه ای یا ضرب اسکالر اصلی ترین عملیات در این سیستم رمزنگاری است. بنابراین، اجرای کارآمد این عمل می تواند منجر به عملکرد و سرعت بالای کل سیستم رمزنگاری شود. پیاده سازی سخت افزاری با سرعت بالا می تواند یک راه حل قابل قبول برای رسیدن به یک عملکرد مناسب در سیستم های کاربردی بر اساس خم های بیضوی باشد. در مقایسه با سایر سیسم های کلید عمومی، رمزنگاری خم بیضوی همان سطح امنیت را با استفاده از اندازه کلید کوچکتر [۱]-[۲] ارائه می دهد. بنابراین، پیاده سازی موثر رمزنگاری خم بیضوی از نظر زمان محاسبه و سطح مصرفی بسیار مهم است. برای کاربردهای مختلف، پیاده سازی با سرعت بالا مورد نیاز است در حالی که برای دستگاه های کوچک و جاسازی شده رمزنگاری سخت افزار مصرفی اصلی ترین دغدغه ای است که باید مورد توجه قرار گیرد. یک فرآیند سلسله مراتبی برای اجرای سخت افزاری رمزنگاری خم بیضوی استفاده می شود. عملیات محاسباتی میدان متناهی مانند ضرب میدانی، مربع کردن میدانی و معکوس کردن میدانی در فرآیند پیاده سازی دخیل هستند.

در سال های اخیر، برای پیاده سازی سخت افزاری سیستم های رمزنگاری خم بیضوی بیشتر میدان های متناهی باینری ($GF(2^m)$ در نظر گرفته شده است، زیرا عملیات جمع در میدان باینری بدون بیت نقلی است و می تواند توسط عمل XOR بیتی ساده و با تاخیر زمانی یک گیت XOR انجام شود. عناصر در میدان های باینری بر اساس یک پایه نشان داده می شوند. دو پایه کاربردی مهم شامل پایه چند جمله ای (PB) و پایه نرمال (NB) هستند. در پایه نرمال، عمل مربع کردن (به توان ۲ رساندن یک عضو میدان) با یک عمل شیفت چرخشی ساده اجرا می شود. این باعث می شود که میدان های متناهی باینری با پایه نرمال یک انتخاب مناسب برای اجرای سخت افزاری کارآمد سیستم های رمزنگاری خم بیضوی باشند. سرعت محاسباتی بالای پیاده سازی سخت افزاری سیستم های رمزنگاری خم بیضوی می توان عملکرد قابل ملاحظه ای را در مقایسه با پیاده سازی نرم افزار به دست آورد. بنابراین، پیاده سازی های سخت افزاری را می توان در کاربرد های مهم که نیاز به سرعت بالا در رمزگذاری یا رمزگشایی دارند استفاده کرد. ساختار موازی با پیاده سازی سخت افزاری کارآمد عملیات حسابی میدان متناهی برای افزایش سرعت عمل ضرب نقطه، که عمل اصلی در سیستم های رمزنگاری خم بیضوی است، استفاده می شود.

در ادامه، آثار اخیر در زمینه پیاده سازی سخت افزاری از ضرب نقطه در سیستم های رمزنگاری خم بیضوی باینری مبتنی بر FPGA را ارائه می دهیم. بسیاری از پیاده سازی های مختلف سخت افزاری مبتنی بر FPGA از ضرب نقطه روی خم های بیضوی باینری در [۳]-[۲۷] گزارش شده اند. به عنوان مثال، معماری پیشنهادی در مرجع [۷] بر اساس الگوریتم ضرب نقطه‌ای لوپز-دهاب اصلاح‌شده می‌باشد که در آن از ضرب‌کننده‌های میدانی با پایه نرمال گوسی بر روی میدان $GF(2^{163})$ استفاده شده است. در [۸] برای بهبود ساختار و کاهش تأخیر زمانی مسیر بحرانی، با سازماندهی و مرتب‌کردن ساختار ضرب نقطه‌ای، ضرب‌کننده به صورت موازی طراحی شده و عملیات در مسیر بحرانی به مسیرهایی که بحرانی نیستند انتقال داده شده‌اند. در [۱۲] یک مدل برای تقریب تأخیر واحدهای عملیاتی خم بیضوی بر روی FPGAهای دارای LUT با k ورودی ارائه شده است. این ساختار بر اساس یک ضرب‌کننده‌ی بیت-موازی طراحی شده است. در هر سیکل ساعت، واحد رجیستر فایل شش ورودی را به قسمت پردازشی اعمال کرده و در انتهای سیکل نتایج خروجی از طریق چهار باس در رجیستر فایل ذخیره می‌شوند. این واحد شامل ۸ رجیستر می‌باشد که هر کدام قابلیت ذخیره یک عنصر میدانی را دارند. عنصر اصلی قسمت پردازشی مربوط به ضرب‌کننده‌ی میدانی می‌باشد که دارای ساختار بیت-موازی می‌باشد. این واحد حجم زیادی از سخت افزار مصرفی و نیز تاخیر مسیر بحرانی را به خود اختصاص داده است. در [۱۳] ساختاری سریع برای ضرب نقطه ای با افزایش فرکانس کاری و کاهش تعداد سیکل ساعت ارائه شده است. فرکانس با استفاده از پیاده سازی بهینه عملیات میدانی به صورت خط لوله ای شده بر اساس یک تجزیه و تحلیل ریاضی افزایش می یابد. همچنین، برای کاهش تعداد سیکل های ساعت، یک برنامه زمان بندی ارائه شده است که امکان پردازش همزمان بیت های عدد اسکالر را در محاسبه ضرب نقطه ای را فراهم می کند. در [۱۴] یک پردازنده برای سیستم های رمزنگاری خم بیضوی روی $GF(2^m)$ ارائه شده است که به نسبت بسیار مناسبی از بازدهی/سطح مصرفی در FPGA دست می یابد. برای افزایش سرعت ضرب نقطه ای، از یک ضرب کننده میدانی خط لوله ای شده رقمی-سریال استفاده شده است. برای دستیابی به تاخیر کم، یک الگوریتم ترکیبی برای جمع نقطه و دو برابر کردن یک نقطه با زمان بندی مناسب ارائه شده است. در [۱۵] یک ساختار سخت افزاری برای ضرب نقطه ای بر اساس الگوریتم نردبان مُنتگومری ارائه شده است. جمع نقطه و دو برابر کردن یک نقطه به طور موازی توسط سه ضرب میدان میدان رقمی-سریال بر اساس پایه چند جمله ای انجام می شود. برای افزایش سرعت پردازش، از تکنیک استفاده از چند فرکانس برای سیگنال ساعت استفاده می شود. برای این منظور، یک مدار سوئیچ سیگنال کلاک برای مدیریت سیگنال کلاک استفاده می شود، به

طوری که مدار می تواند در حداکثر فرکانس ساعت خود که توسط تاخیرهای مختلف مسیرهای بحرانی عمل کند. در [۱۷] دو معماری برای پیاده سازی الگوریتم ضرب نقطه ای لوپز-دهاب براساس الگوریتم مُنتگومری ارائه شده است. این الگوریتم برای برنامه ریزی دقیق برای جلوگیری از وابستگی به داده ها و در نتیجه کاهش شدید تعداد چرخه های ساعت مورد نیاز تغییر یافته است. معماری اول از یک ضرب کننده میدانی برای پیاده سازی پردازنده خم بیضوی با کارایی بالا و سطح مصرفی کم استفاده می کند. معماری دوم بر اساس سه ضرب کننده میدانی برای پردازنده خم بیضوی برای دستیابی به بالاترین سرعت ممکن است.

پیاده‌سازی سخت‌افزاری ضرب نقطه‌ای بر روی خم‌های باینری ادواردز و هشیان کلی‌شده در [۱۹]-[۲۳] گزارش شده‌اند. در [۱۹] برای کاهش تعداد سیکل ساعت لازم برای انجام ضرب نقطه‌ای، یک آنالیز بر روی مسیر داده و حداکثر تعداد ضرب‌کننده‌ی میدانی موازی انجام گرفته است. همچنین از یک ضرب‌کننده‌ی رقمی-سریال ترکیبی با پایه نرمال گوسی برای کم کردن وابستگی داده‌ها و کاهش تعداد سیکل ساعت استفاده شده است. در مرجع [۲۰] یک پیاده‌سازی در سطح FPGA بر روی خم‌های باینری ادواردز ارائه شده است که در آن از ضرب کننده ی Karatsuba-Ofman استفاده شده است. در [۲۱] با استفاده از تکنیک موازی‌سازی حداکثر استفاده از منابع سخت‌افزاری در محاسبه جمع دو نقطه و دو برابر کردن یک نقطه در خم‌های باینری ادواردز و هشیان کلی‌شده بدست آمده است. در این پیاده‌سازی فرمول‌های تفاضلی برای محاسبه ضرب نقطه‌ای استفاده شده است. ضرب‌کننده‌های میدانی بکار رفته در [۲۱] دارای ساختار رقمی-سریال با پایه نرمال گوسی می‌باشد. این معماری از چندین بلوک شامل دو ضرب‌کننده‌ی میدانی، دو جمع‌کننده، دو واحد مربع‌کننده و قسمت مربوط به حافظه ساخته شده است. قسمت حافظه شامل دو بخش می‌باشد که در آن حافظه RAM برای ذخیره مقادیر میانی محاسبه شده در طول محاسبات و خروجی‌های نهایی و حافظه ROM برای ذخیره‌کردن دستورات پردازنده ضرب نقطه‌ای می‌باشد. انتقال اطلاعات برای پردازش و ذخیره‌سازی بین واحد پردازنده عملیات میدانی و حافظه‌های استفاده شده از طریق باس‌های بین این دو واحد اصلی صورت می‌گیرد. همچنین یک رجیستر ۱۶۳-بیتی برای ذخیره‌سازی موقت نتایج حاصل از واحد پردازنده برای پردازش‌های بعدی در نظر گرفته شده است. در مرجع [۲۲] یک ساختار با سرعت بالا بر اساس دو ضرب‌کننده‌ی میدانی برای پیاده‌سازی ضرب نقطه‌ای روی خم‌های باینری ادواردز ارائه شده است. این پردازنده شامل دو ضرب‌کننده‌ی میدانی با ساختار بیت-موازی، سه جمع‌کننده، دو واحد مربع‌کننده میدانی، دو رجیستر فایل و چندین مالتی‌پلکسر می‌باشد. در مقاله [۲۳] یک معماری با پیچیدگی کم برای محاسبات ضرب نقطه ای برای خم های بیضوی باینری ادواردز بر روی $GF(2^{233})$ ارائه می دهد. سه نوآوری عمده در این مقاله وجود دارد. اولین نوآوری کاهش پیچیدگی در سطح دستورالعمل برای قوانین یکپارچه جمع و دو برابر کردن نقطه با حذف چندین عملیات در یک فرمت دستورالعمل واحد است. نوآوری دوم بهینه سازی منابع سخت افزاری با به حداقل رساندن تعداد عناصر ذخیره سازی مورد نیاز است. در نهایت، نوآوری سوم کاهش تعداد سیکل‌های ساعت مورد نیاز با گنجاندن یک ضرب‌کننده موازی-رقمی میدانی با مسیر داده ۳۲-بیتی است. در مرجع [۲۴] یک معماری موازی جدید برای ضرب اسکالر خم بیضوی بر اساس الگوریتم اصلاح‌شده لوپز-دهاب-مونتگومری (LDM) ارائه شده است تا کل تاخیر زمانی برای محاسبه ضرب نقطه ای را کاهش دهد. در مرجع [۲۵] یک معماری جدید پر سرعت ضرب نقطه ای در رمزنگاری خم بیضوی معرفی می شود. یک ضرب‌کننده میدانی با دقت کامل برای کوتاه کردن تأخیر پیشنهاد شده است، و یک معماری برای عمل معکوس کردن میدانی جدید برای کاهش تعداد کل چرخه‌های ساعت در ضرب نقطه ای ادغام شده است. در [۲۷] یک معماری ضرب نقطه ای ایجاد شده برای الگوریتم نردبان مونتگومری اصلاح شده ارائه می شود. یک ضرب کننده میدانی سریال-رقمی برای انجام ضرب در الگوریتم نردبان مونتگومری اصلاح شده استفاده می شود.

مقاله حاضر بر پیاده سازی سخت افزاری معماری با سرعت بالا و بازدهی بالا از ضرب نقطه ای بر اساس الگوریتم نردبان مُنتگومری برای خم های بیضوی باینری ادواردز و هشیان کلی شده بر روی میدان $GF(2^m)$ متمرکز است. نتایج پیاده سازی نشان می دهد که ساختارهای پیشنهادی دارای کارایی بالا و زمان محاسبه کم و با مصرف سخت افزار کم در مقایسه با سایر کارهای مرتبط می باشند. به عنوان مثال در مورد خم های باینری ادواردز روی $GF(2^{233})$ (در Virtex-4 XC4VLX110 FPGA) ساختار پیشنهادی می تواند به ترتیب تا ۳۷% و ۴۰% از منابع سخت افزاری و زمان اجرا را کاهش و بازدهی را ۶۲% در مقایسه با بهترین کار موجود بهبود دهد.، بهبودهایی که در این مقاله در ساختارهای پیشنهادی داده شده است به شرح زیر هستند:

- در ساختار پیشنهادی، برای کاهش تعداد سیکل های ساعت، ضرب کننده های میدانی به صورت موازی برای محاسبه جمع نقطه و دو برابر کردن یک نقطه بکار گرفته شده اند.
- برای دستیابی به یک پیاده سازی بهینه، ما از تکنیک زمان بندی و تخصیص منابع عمل های میدانی برای کاهش تعداد سیکل های ساعت و مصرف سخت افزار استفاده کرده ایم.
- در ساختارهای پیشنهادی ضرب نقطه ای فقط از سه و چهار واحد ضرب کننده میدانی به ترتیب برای حالت خاص و کلی خم های باینری ادواردز استفاده می شود.
- از سه ضرب کننده میدانی برا خم های باینری هشیان کلی شده استفاده می شود. ضرب کننده ها برای کاهش تعداد سیکل های ساعت در طول فرآیند ضرب نقطه ای زمان بندی و به اشتراک گذاشته می شوند.

- ضرب کننده میدانی بر اساس ساختار رقمی-سریال با پایه نرمال گوسی طراحی شده و دارای یک ساختار بسیار منظم با تاخیر کم در مسیر بحرانی و منابع سخت افزاری کم است.

در ادامه مقاله در بخش دوم الگوریتم ضرب نقطه ای نردبان مُنتگومری برای خم های باینری ادواردز و هشیان کلی شده ارائه می شود. در بخش سوم ساختارهای پیشنهادی توضیح داده شده اند. ساختار پیشنهادی از ضرب کننده رقمی-سریال در پایه نرمال گوسی بر روی $\mathrm{GF}(2^m)$ در بخش چهار ارائه می شود. بخش پنجم مقایسه ای بین این کار و سایر آثار قبلی ارائه می دهد. در آخر مقاله در بخش ششم مورد نتیجه گیری قرار می گیرد.

۲- الگوریتم ضرب نقطه‌ای نردبان مُنتگومری برای خم‌های باینری ادواردز و هشیان کلی شده

الگوریتم ۱ ضرب نقطه‌ای برای خم‌های باینری ادواردز و هشیان کلی-شده بر اساس الگوریتم نردبان مُنتگومری را نشان می‌دهد. در این الگوریتم از فرمول‌های جمع و دو برابر کردن تفاضلی[۲] استفاده شده است.

Algorithm 1: Montgomery ladder point multiplication

Input: $\boldsymbol{k} = (\boldsymbol{k_{m-1}}.\boldsymbol{k_{m-2}}.\ldots.\boldsymbol{k_2}.\boldsymbol{k_1}.\boldsymbol{k_0})_2$ with $\boldsymbol{k_{m-1}} = \boldsymbol{1}$. $\boldsymbol{P} = (\boldsymbol{x}.\boldsymbol{y}) \in \boldsymbol{E}/\mathrm{GF}(2^m)$.
Output: $w(\boldsymbol{kP})$

1. Initial values

$\boldsymbol{w_0} \coloneqq w(\boldsymbol{P}).\boldsymbol{W_1} \coloneqq \boldsymbol{w_0}.\ \boldsymbol{Z_1} \coloneqq \boldsymbol{1}$;
$(\boldsymbol{W_2}.\boldsymbol{Z_2}) \coloneqq \mathrm{Double}(\boldsymbol{W_1}.\boldsymbol{Z_1})$;

2. Loop iterations part

For $\boldsymbol{i} = \boldsymbol{m-2}$ to 0 do
  If $\boldsymbol{k_i} = \boldsymbol{1}$ then
    $(\boldsymbol{W_1}.\boldsymbol{Z_1}) \coloneqq \mathrm{dAdd}\ (\boldsymbol{W_1}.\boldsymbol{Z_1}, \boldsymbol{W_2}.\boldsymbol{Z_2}, \boldsymbol{w_0})$;
    $(\boldsymbol{W_2}.\boldsymbol{Z_2}) \coloneqq \mathrm{Double}\ (\boldsymbol{W_2}.\boldsymbol{Z_2})$;
  Else
    $(\boldsymbol{W_2}.\boldsymbol{Z_2}) \coloneqq \mathrm{dAdd}\ (\boldsymbol{W_1}.\boldsymbol{Z_1}, \boldsymbol{W_2}.\boldsymbol{Z_2}, \boldsymbol{w_0})$;
    $(\boldsymbol{W_1}.\boldsymbol{Z_1}) \coloneqq \mathrm{Double}\ (\boldsymbol{W_1}.\boldsymbol{Z_1})$;
  End if;
End for;

Return $w(\boldsymbol{kP}) <= (\boldsymbol{W_1}.\boldsymbol{Z_1})$ and $w((\boldsymbol{k}+\boldsymbol{1})\boldsymbol{P}) <= (\boldsymbol{W_2}.\boldsymbol{Z_2})$

در ادامه سیستم مختصات تفاضلی استفاده شده در الگوریتم ۱ توضیح داده می‌شود. اگر w یک تابع گویا بر روی خم بیضوی E روی میدان $\mathrm{GF}(2^m)$ باشد، برای نقطه P در $E\big(\mathrm{GF}(2^m)\big)$، $w(P)$ متعلق به میدان $\mathrm{GF}(2^m)$ است. برای هر نقطه P در E داریم $w(P) = w(-P)$ که در آن $-P$ برابر معکوس جمعی نقطه P است. فرمول‌های جمع تفاضلی برای محاسبه $w(P+Q)$ بر اساس $w(P)$، $w(Q)$ و $w(P-Q)$ می‌باشند، همچنین فرمول‌های دو برابر کردن تفاضلی برای بدست آوردن $w(2mP)$ برحسب $w(P)$ هستند. در الگوریتم 1 فرمول‌های جمع و دو برابر کردن تفاضلی به ترتیب توسط توابع "dAdd" و "Double" انجام می‌شوند. در این الگوریتم برای عدد صحیح مثبت k و نقطه P روی خم، مقدار $w(kP)$ توسط الگوریتم نردبان مُنتگومری محاسبه می‌شود. در این محاسبات $w(kP)$ با یک روش تکراری بوسیله‌ی محاسبه $w(2mP)$ و $w((2m+1)P)$ بر اساس سه مقدار $w(iP)$، $w((i+1)P)$ و $w(P)$ توسط فرمول‌های جمع و دو برابر کردن تفاضلی بدست می‌آید. در این الگوریتم برای جلوگیری از انجام عمل معکوس‌گیری از مختصات تصویری استفاده می‌شود، بطوری که برای نقطه آفین P در E داریم $w(P) = \frac{W}{Z}$ که در آن $Z \in \mathrm{GF}(2^m)$ و W و در این حالت نقطه P به صورت $(W:Z)$ است.

- فرمول‌های جمع و دو برابر کردن تفاضلی *w*-coordinate برای خم‌های باینری ادواردز

فرمول‌های جمع تفاضلی روی خم‌های بیضوی باینری ادواردز در مرجع [۲۸] ارائه شده‌اند. برای نقطه $P = (x.y)$ روی خم بیضوی باینری ادواردز تابع w به صورت $w(P) = x + y$ تعریف می‌شود. در این حالت $w(-P) = w(P)$ است زیرا $-P = (y.x)$ می‌باشد. برای دو نقطه P_1 و P_2 روی خم E_{d_1,d_2}، $w(P_1)$ و $w(P_2)$ برابر $\frac{W_1}{Z_1}$ و $\frac{W_2}{Z_2}$ می‌باشند. اگر $P = P_1 - P_2$ بوده و $w_0 = w(P)$ به عنوان یک عنصر میدان تعریف شود، برای محاسبه‌ی جمع دو نقطه $P_a = P_1 + P_2$، مقدار $w(P_a) = \frac{W_a}{Z_a}$ بوده و فرمول‌های جمع به صورت زیر می‌باشند:

$$C = W_1 \times (Z_1 + W_1), D = W_2 \times (Z_2 + W_2),$$
$$E = Z_1 \times Z_2, F = W_1 \times W_2, V = C \times D$$
$$Z_a = V + (e_1 \times E + e_2 \times F)^2, W_a = V + w_0 \times Z_a$$

که در آن $e_1 = \sqrt{d_1}$ و $e_2 = \sqrt{\frac{d_2}{d_1} + 1}$ است. برای دو برابر کردن یک نقطه $P_d = 2P_1$ مقدار $w(P_a)$ برابر $\frac{W_d}{Z_d}$ است و فرمول‌های آن به صورت زیر می‌باشند:

$$C = W_1 \times (Z_1 + W_1), W_d = C^2$$
$$Z_d = W_d + ((e_2 \times Z_1 + e_4 \times W_1^2)^2)^2$$

که در آن $e_3 = \sqrt[4]{d_1}$ و $e_4 = \sqrt[4]{\frac{d_2}{d_1} + 1}$ است. حال اگر $d_1 = d_2$ باشد، فرمول‌های مربوط به جمع دو نقطه به صورت زیر است:

$$C = W_1 \times (Z_1 + W_1), D = W_2 \times (Z_2 + W_2),$$
$$E = Z_1 \times Z_2, V = C \times D$$
$$Z_a = V + d_1 \times E^2, W_a = V + w_0 \times Z_a$$

در این فرمول‌ها $w_0 = w(P) = x + y$ است و همچنین برای فرمول‌های دو برابر کردن در این حالت داریم:

$$C = W_1 \times (Z_1 + W_1), W_d = C^2, Z_d = d_1 \times (Z_1^2)^2 + W_d$$

- **فرمول‌های جمع و دو برابر کردن تفاضلی ترکیبی w-coordinate برای خم‌های باینری هشیان کلی شده**

برای نقطه $P=(x.y)$ بر روی خم باینری هشیات کلی‌شده، $w(P)$ به صورت $w(P)=c+dxy$ یا $w(P)=x^3+y^3$ محاسبه می‌شود. در اینجا $w(P)=w(-P)$ است زیرا $-P=(y.x)$ می‌باشد. در ادامه فرمول‌های جمع و دو برابر کردن تفاضلی ترکیبی w-coordinate بیان شده‌اند. برای دو نقطه P_1 و P_2 روی خم داریم $w(P_1)=\frac{W_1}{Z_1}$ و $w(P_2)=\frac{W_2}{Z_2}$. اگر w_0 برابر $w(P_1-P_2)$ باشد، برای انجام عمل جمع دو نقطه $P_a=P_1+P_2$ و دو برابر کردن $P_d=2P_1$، عناصر میدان $w(P_a)=\frac{W_a}{Z_a}$ و $w(P_d)=\frac{W_d}{Z_d}$ به صورت زیر محاسبه می‌شوند:

$$A=W_1\times Z_2, B=W_2\times Z_1, C=A\times B, U=h_2\times C,$$
$$Z_a=(A+B)^2, W_a=U+w_0\times Z_a$$

برای دو برابر کردن یک نقطه داریم:

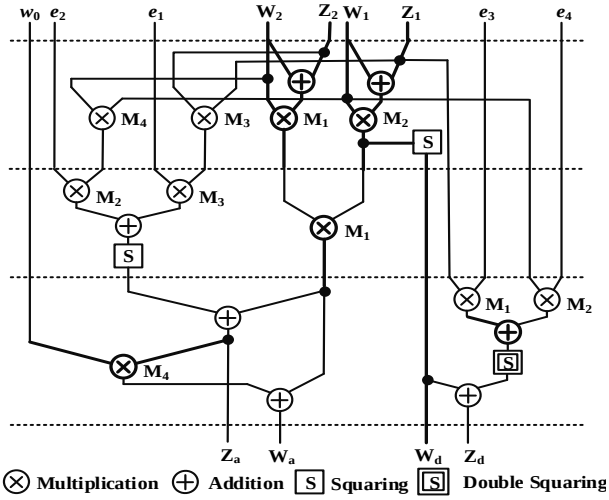
$$A=W_1^2, B=Z_1^2, C=A+h_1\times B, D=h_2\times B,$$
$$W_d=C^2, Z_d=A\times D$$

که در آن $h_1=\sqrt{c^3(d^3+c)}$ و $h_2=d^3$ است. ساختارهای ارائه شده برای محاسبه ضرب نقطه‌ای برای خم‌های باینری ادواردز و هشیان کلی‌شده بر اساس الگوریتم نردبان مُنتگومری در ادامه آمده است.

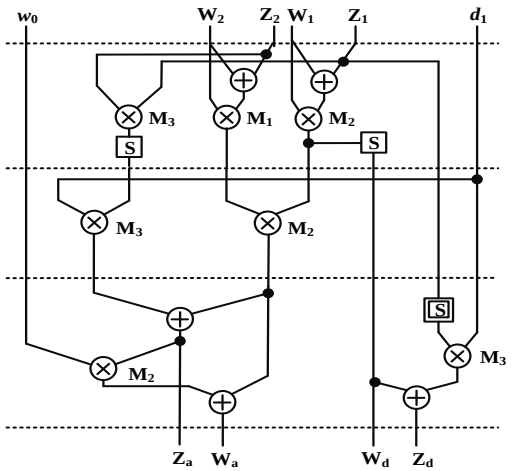
۳- ساختار پیشنهادی محاسبه ضرب نقطه‌ای برای خم های باینری ادواردز

پیاده‌سازی‌های ارائه شده در این قسمت بر اساس پایه نرمال گوسی می‌باشند. بنابراین ضرب‌کننده‌ی میدانی استفاده شده برای محاسبه‌ی ضرب نقطه‌ای همان ضرب‌کننده‌ی میدانی با پایه نرمال گوسی توضیح داده شده در بخش پنجم است. در خم‌های بیضوی باینری ادواردز عملیات میدانی مربوط به عمل جمع دو نقطه PA و دو برابر کردن یک نقطه PD به صورت موازی توسط سه مرحله ضرب میدانی انجام می‌شوند. برای مثال در حالت $d_1\neq d_2$، محاسبات PA و PD نیازمند ۱۰ عمل ضرب میدانی می‌باشند. بنابراین پیاده‌سازی این محاسبات به صورت موازی و با در نظر گرفتن وابستگی روابط به یکدیگر حداقل در سه مرحله ضرب میدانی انجام می‌شود. در این حالت در هر مرحله حداکثر چهار ضرب‌کننده‌ی میدانی مورد استفاده قرار می‌گیرد. در مرحله اول چهار عمل ضرب $C=W_1\times(Z_1+W_1)$، $D=W_2\times(Z_2+W_2)$، $E=Z_1\times Z_2$ و $F=W_1\times W_2$ به ترتیب توسط چهار ضرب‌کننده‌ی موازی M_2، M_1، M_3 و M_4 انجام می‌شوند. در مرحله دوم ضرب‌های میدانی $V=C\times D$، $F\times e_2$ و $E\times e_1$ به صورت مشابه محاسبه می‌شوند. سرانجام در مرحله آخر $Z_1\times e_3$، $W_1\times e_4$ و $w_0\times Z_a$ توسط ضرب‌کننده‌های M_1، M_2 و M_4 حساب می‌شوند. شکل‌های ۱ و ۲ به ترتیب زمان‌بندی ارائه شده برای محاسبات موازی PA و PD خم‌های باینری ادواردز برای دو حالت $d_1\neq d_2$ و $d_1=d_2$ را نشان می‌دهند.

در زمان‌بندی‌های ارائه شده تخصیص منابع بر اساس کاهش تعداد سیکل ساعت انجام شده است. ضرب‌کننده‌ی میدانی با پایه نرمال گوسی دارای ساختار رقمی-سریال می‌باشد، بطوری که رجیستر خروجی در مسیر مستقیم خروجی قرار نگرفته است، بنابراین نتیجه هر ضرب میدانی در w سیکل بدست می‌آید که در آن w برابر تعداد کلمه‌های ورودی می‌باشد. همچنین در ساختار ضرب‌کننده‌ی میدانی تکنیک خط لوله برای کاهش تاخیر مسیر بحرانی استفاده شده است، در نتیجه خروجی هر ضرب میدان در $w+1$ سیکل ساعت محاسبه می‌شود. بنابراین کل محاسبات مربوط به PA و PD به صورت موازی و در سه مرحله در $3(w+1)$ سیکل محاسبه می‌شود.



شکل (۱): زمان‌بندی ارائه شده برای محاسبات موازی PA و PD برای خم‌های باینری ادواردز در حالت $d_1\neq d_2$



شکل (۲): زمان‌بندی ارائه شده برای محاسبات موازی PA و PD برای خم‌های باینری ادواردز در حالت $d_1=d_2$

جدول (۱): عملکرد چهار ضرب‌کننده‌ی $\mathbf{M_1}$، $\mathbf{M_2}$، $\mathbf{M_3}$ و $\mathbf{M_4}$ در هر حلقه‌ی تکرار الگوریتم ۱ برای حالت $d_1 \neq d_2$ و $w = 4$

Cycles	M_1	M_2	M_3	M_4
1	Part 1 of $W_2(Z_2+W_2)$	Part 1 of $W_1(Z_1+W_1)$	Part 1 of Z_1Z_2	Part 1 of W_1W_2
2	Part 2 of $W_2(Z_2+W_2)$	Part 2 of $W_1(Z_1+W_1)$	Part 2 of Z_1Z_2	Part 2 of W_1W_2
3	Part 3 of $W_2(Z_2+W_2)$	Part 3 of $W_1(Z_1+W_1)$	Part 3 of Z_1Z_2	Part 3 of W_1W_2
4	Part 4 of $W_2(Z_2+W_2)$	Part 4 of $W_1(Z_1+W_1)$	Part 4 of Z_1Z_2	Part 4 of W_1W_2
5	Part 5 of $W_2(Z_2+W_2)$	Part 5 of $W_1(Z_1+W_1)$	Part 5 of Z_1Z_2	Part 5 of W_1W_2
6	Part 1 of CD	Part 1 of Fe_2	Part 1 of Ee_1	---
7	Part 2 of CD	Part 2 of Fe_2	Part 2 of Ee_1	---
8	Part 3 of CD	Part 3 of Fe_2	Part 3 of Ee_1	---
9	Part 4 of CD	Part 4 of Fe_2	Part 4 of Ee_1	---
10	Part 5 of CD	Part 5 of Fe_2	Part 5 of Ee_1	---
11	Part 1 of Z_1e_3	Part 1 of W_1e_4	---	Part 1 of w_0Z_a
12	Part 2 of Z_1e_3	Part 2 of W_1e_4	---	Part 2 of w_0Z_a
13	Part 3 of Z_1e_3	Part 3 of W_1e_4	---	Part 3 of w_0Z_a
14	Part 4 of Z_1e_3	Part 4 of W_1e_4	---	Part 4 of w_0Z_a
15	Part 5 of Z_1e_3	Part 5 of W_1e_4	---	Part 5 of w_0Z_a

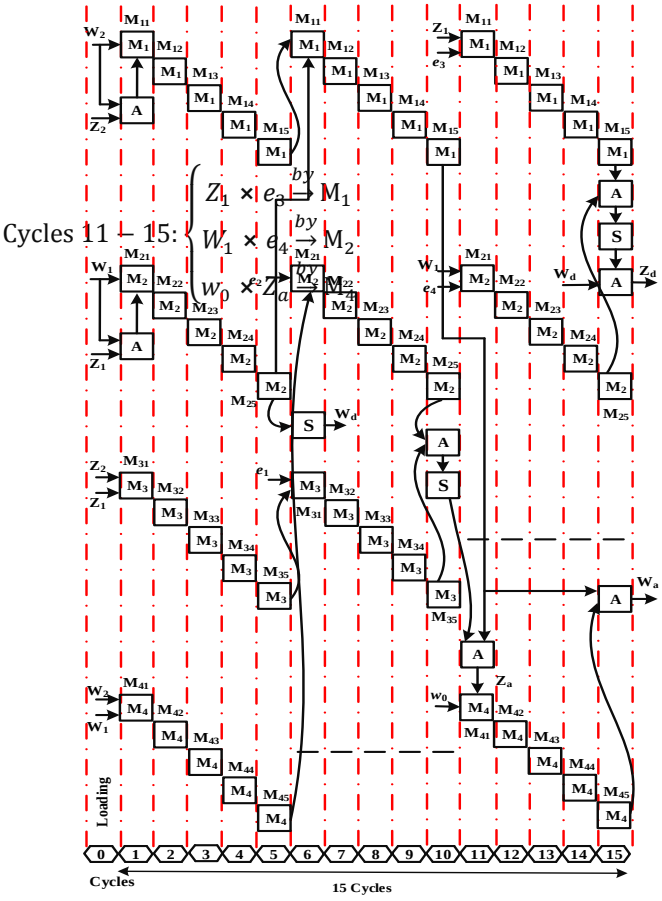
برای مثال در حالت $d_1 \neq d_2$ با $w = 4$، محاسبات در ۱۵ سیکل ساعت به صورت زیر محاسبه می‌شوند:

$$\text{Cycles } 1-5:\begin{cases} C=W_1\times(Z_1+W_1)\xrightarrow{by}\mathrm{M}_2 \\ D=W_2\times(Z_2+W_2)\xrightarrow{by}\mathrm{M}_1 \\ E=Z_1\times Z_2\xrightarrow{by}\mathrm{M}_3 \\ F=W_1\times W_2\xrightarrow{by}\mathrm{M}_4 \end{cases},$$

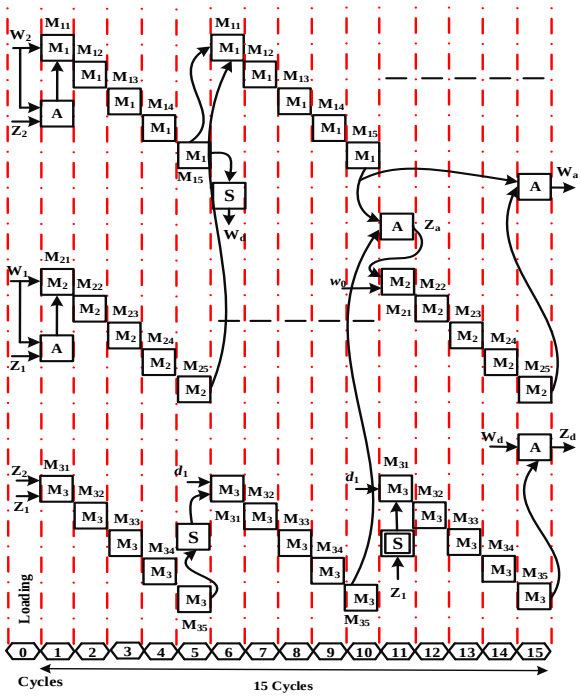
$$\text{Cycles } 6-10:\begin{cases} V=C\times D\xrightarrow{by}\mathrm{M}_1 \\ F\times e_2\xrightarrow{by}\mathrm{M}_2 \\ E\times e_1\xrightarrow{by}\mathrm{M}_3 \end{cases},$$

جزئیات عملکرد چهار ضرب‌کننده‌ی M_1، M_2، M_3 و M_4 در هر حلقه‌ی تکرار الگوریتم ۱ در جدول ۱ بر اساس شماره سیکل ساعت مربوطه نشان داده شده است.

شکل‌های ۳ و ۴ گراف محاسبات مربوط به PA و PD را به ترتیب برای دو حالت $d_1 \neq d_2$ و $d_1 = d_2$ نشان می‌دهند. همان طور که دیده می‌شود عمل ضرب‌کننده‌ها به صورت موازی بوده و محاسبات همزمان انجام می‌شوند. در حالت $d_1 \neq d_2$، دو ضرب‌کننده‌ی M_1 و M_2 در تمام ۱۵ سیکل محاسبات بکار گرفته شده‌اند. همچنین دو ضرب‌کننده‌ی M_3 و M_4 در ۱۰ سیکل بکار گرفته شده‌اند. بنابراین ضریب استفاده برای ضرب‌کننده‌های میدانی M_1 و M_2 برابر $\frac{15}{15}\times 100 = 100\%$ و برای M_3 و M_4 برابر $\frac{10}{15}\times 100 = 66.67\%$ است. همچنین برای حالت $d_1 = d_2$ ضریب استفاده برای ضرب‌کننده‌های M_1، M_2 و M_3 به ترتیب برابر 66.67%، 66.67% و 100% می‌باشد. ساختارهای پیشنهادی برای پیاده‌سازی ضرب نقطه‌ای خم‌های باینری ادواردز بر اساس الگوریتم نردبان مُنتگومری برای حالت‌های $d_1 \neq d_2$ و $d_1 = d_2$ به ترتیب در شکل‌های ۵ و ۶ نشان داده شده‌اند. در این ساختارها الگوریتم نردبان مُنتگومری با استفاده از چهار و سه ضرب‌کننده‌ی میدانی برای حالت‌های $d_1 \neq d_2$ و $d_1 = d_2$ پیاده‌سازی شده است. همانطور که در الگوریتم ۱ دیده می‌شود، در اولین مرحله از الگوریتم، مقادیر اولیه پارامترهای W_1 و Z_1 توسط نقطه



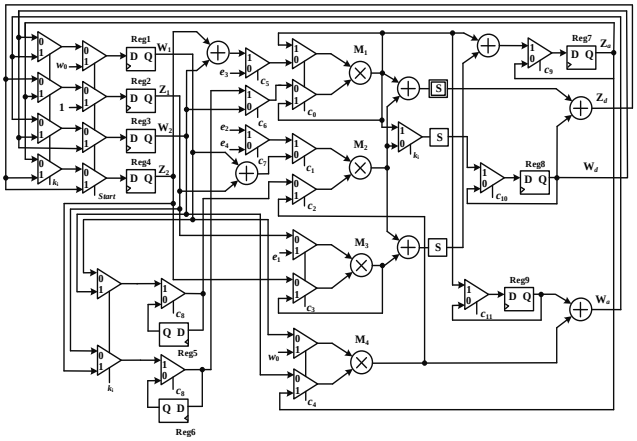
شکل (۳): گراف محاسبات مربوط به PA و PD برای حالت $d_1 \neq d_2$



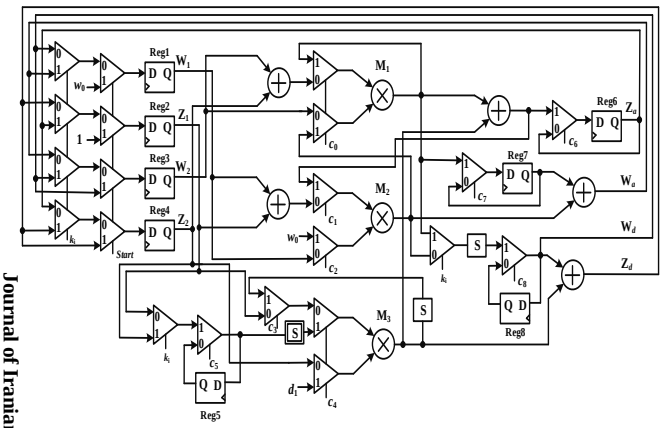
شکل (۴): گراف محاسبات مربوط به PA و PD برای حالت $d_1 = d_2$

ورودی P، و مقادیر W_2 و Z_2 توسط عمل دو برابر کردن $(W_2.Z_2) = \text{Double}(W_1.Z_1)$ محاسبه می‌شوند. عمل دو برابر کردن برای خم‌های باینری ادواردز در حالت‌های $d_1 \neq d_2$ و $d_1 = d_2$ به ترتیب با سه و یک عمل ضرب میدانی انجام می‌شود. در شکل‌های ۴ و ۵ سیگنال *Start* برای محاسبه مقادیر اولیه در الگوریتم ۱ در نظر گرفته شده است. در اولین سیکل ساعت سیگنال *Start* برابر '1' است. در این حالت مقادیر $W_1=w_0$ و $Z_1=1$ بارگذاری شده و در رجیسترهای Reg1 و Reg2 ذخیره می‌شوند. سپس سیگنال *Start* برابر مقدار '0' شده و عمل دو برابر کردن توسط ضرب‌کننده‌های M_1 و M_2 برای محاسبه W_2 و Z_2 انجام می‌شود. بعد از محاسبه عمل دو برابر کردن، خروجی‌های $W_2 = W_d$ و $Z_2 = Z_d$ برای شروع محاسبات مربوط به حلقه‌های تکرار آماده می‌شوند. در این زمان سیگنال *Start* دوباره برابر '1' می‌شود و مقادیر اولیه محاسبه شده‌ی W_1، Z_1، W_2 و Z_2 به ترتیب در رجیسترهای Reg1، Reg2، Reg3 و Reg4 ذخیره می‌شوند.

در مرحله دوم از الگوریتم، حلقه‌های تکرار بر اساس مقدار بیت‌های k_i، iامین بیت از عدد اسکالر k، محاسبه می‌شوند. دو مالتی‌پلکسر با سیگنال کنترل k_i برای انتخاب ورودی‌های عمل دو برابر کردن استفاده شده‌اند. بدین صورت که برای حالت '1'$=k_i$ مقادیر W_2 و Z_2 به عنوان ورودی عمل دو برابر کردن، و در حالت '0'$=k_i$ مقادیر W_1، Z_1 به عنوان ورودی عمل دو برابر کردن انتخاب می‌شوند. رجیسترهای Reg5 و Reg6 برای ذخیره مقادیر ورودی عمل دو برابر کردن که در طول زمان انجام عملیات حلقه‌ی تکرار نیاز می‌باشند، استفاده شده‌اند. مالتی‌پلکسر با سیگنال کنترل k_i در خروجی ضرب-کننده‌های M_1 و M_2 برای انتخاب مقادیر محاسبه شده توسط عمل دو برابر کردن قرار گرفته است. همچنین چهار مالتی‌پلکسر با سیگنال کنترلی k_i در قسمت ورودی ساختار برای تعیین رجیسترهای مقصد عملیات دو برابر کردن و جمع دو نقطه استفاده شده‌اند.



شکل (۵): ساختار پیشنهادی برای پیاده‌سازی ضرب نقطه‌ای خم باینری ادواردز براساس الگوریتم نردبان مُنتگومری در حالت $d_1 \neq d_2$



شکل (۶): ساختار پیشنهادی برای پیاده‌سازی ضرب نقطه‌ای خم باینری ادواردز براساس الگوریتم نردبان مُنتگومری در حالت $d_1 = d_2$

در پیاده‌سازی ارائه شده برای محاسبه ضرب نقطه‌ای برای خم‌های باینری ادواردز محاسبات مربوط به مقادیر اولیه در $2(w+1)$ و $(w+1)$ سیکل ساعت به ترتیب برای حالت‌های $d_1 \neq d_2$ و $d_1 = d_2$ انجام می‌گیرند. همچنین هر حلقه‌ی تکرار توسط $3(w+1)+1$ سیکل محاسبه می‌شود، بطوری که در اولین سیکل مقادیر W_1، Z_1، W_2 و Z_2 به ترتیب در رجیسترهای Reg1، Reg2، Reg3 و Reg4 ذخیره می‌شوند و سپس سه مرحله عملیات ضرب میدانی در $3(w+1)$ سیکل انجام می‌شود. بنابراین کل تعداد سیکل ساعت الگوریتم ۱ برای محاسبه ضرب نقطه‌ای برای خم‌های بیضوی باینری ادواردز بر روی میدان $\text{GF}(2^m)$ به ترتیب برای دو حالت $d_1 \neq d_2$ و $d_1 = d_2$ برابر

$(m-1)(3(w+1)+1)+2(w+1)+1$ و $(m-1)(3(w+1)+1)+(w+1)+1$ می‌باشد.

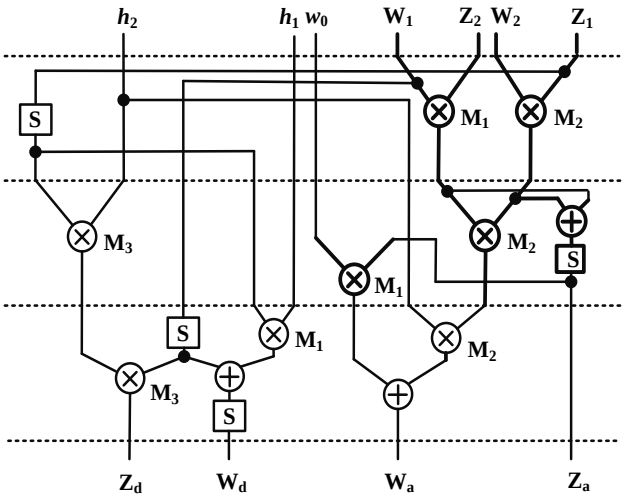
۴- ساختار پیشنهادی محاسبه ضرب نقطه‌ای برای خم‌های باینری هشیان کلی شده

ساختار ضرب نقطه‌ای برای خم‌های باینری هشیان کلی‌شده مشابه خم‌های باینری ادواردز می‌باشد. عملیات جمع دو نقطه و دو برابر کردن یک نقطه در سه مرحله عمل ضرب میدانی مطابق شکل ۸ انجام می‌شود. زمان‌بندی ارائه شده با هدف کاهش تعداد سیکل ساعت طراحی شده است. برای مثال در حالتی که تعداد کلمه مربوط به عنصر ورودی در ضرب میدانی برابر $w=۴$ باشد، خروجی هر ضرب‌کننده که به صورت خط لوله‌ای می‌باشد در ۵ سیکل محاسبه می‌شود. بنابراین کل محاسبات مربوط به PA و PD به صورت زیر در ۱۵ سیکل ساعت زمان‌بندی می‌شود:

$$\text{Cycles } 1-5:\begin{cases} A=W_1\times Z_2 \overset{by}{\rightarrow} \mathrm{M}_1 \\ B=W_2\times Z_1 \overset{by}{\rightarrow} \mathrm{M}_2 \end{cases}.$$

$$\text{Cycles } 6-10:\begin{cases} l=(A+B)^2\times w_0 \overset{by}{\rightarrow} \mathrm{M}_1 \\ C=A\times B \overset{by}{\rightarrow} \mathrm{M}_2 \\ U=h_2\times Z_1^2 \overset{by}{\rightarrow} \mathrm{M}_3 \end{cases}.$$

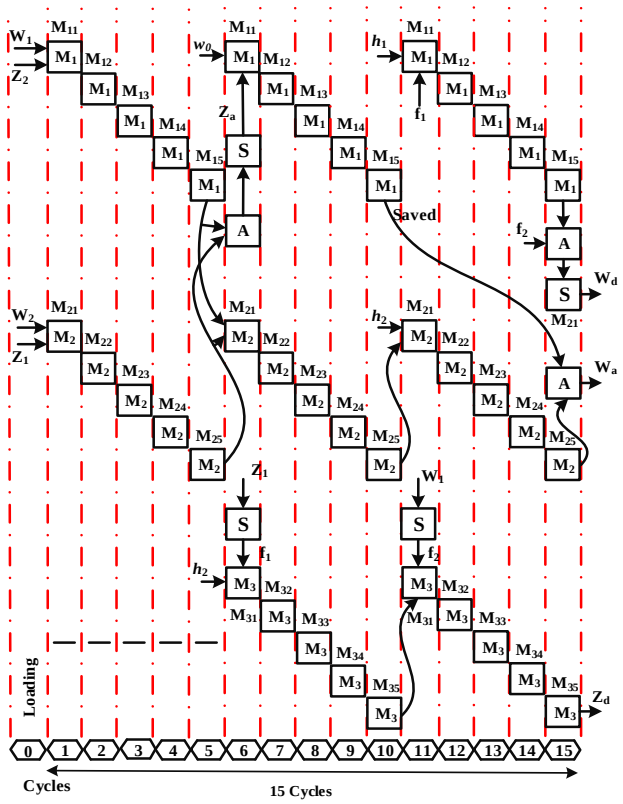
$$\text{Cycles } 11-15:\begin{cases} h_1\times Z_1^2 \overset{by}{\rightarrow} \mathrm{M}_1 \\ h_2\times C \overset{by}{\rightarrow} \mathrm{M}_2 \\ W_1^2\times U \overset{by}{\rightarrow} \mathrm{M}_3 \end{cases}$$



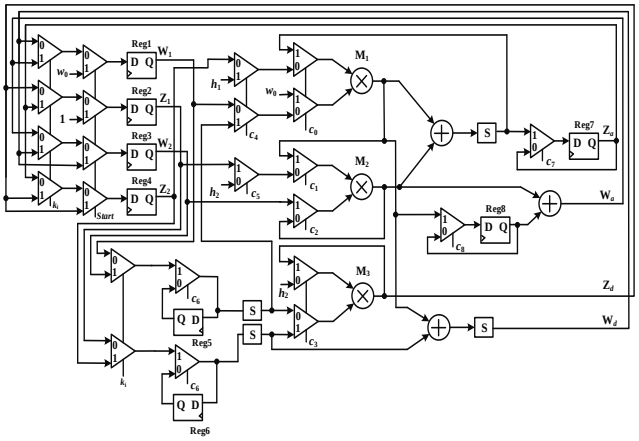
شکل (۷): زمان‌بندی ارائه شده برای محاسبات موازی PA و PD برای خم‌های باینری هشیان کلی‌شده

شکل ۸ گراف محاسباتی مربوط به هر حلقه‌ی تکرار از الگوریتم ۱ برای خم‌های باینری هشیان کلی‌شده را نشان می‌دهد. همان طور که در شکل دیده می‌شود در هر حلقه‌ی تکرار سه ضرب‌کننده‌ی میدانی M_1، M_2 و M_3 به صورت موازی عمل می‌کنند. ضرب‌کننده‌های M_1 و M_2 در تمام ۱۵ سیکل مربوط به عملیات PA و PD بکار گرفته شده‌اند و ضرب‌کننده‌ی M_3 برای ۱۰ سیکل مورد استفاده قرار می‌گیرد. بنابراین، ضریب استفاده برای ضرب‌کننده‌های M_1، M_2 و M_3 به ترتیب برابر 100%، 100% و 66.67% است.

ساختار پیشنهادی مربوط به ضرب نقطه‌ای بر اساس الگوریتم نردبان مُنتگومری برای خم‌های باینری هشیان کلی‌شده در شکل ۹ نشان داده شده است. مقادیر ورودی W_i و Z_i، که در آن i=1,2 است در طول محاسبات دو برابر کردن نقطه ثابت هستند. این مقادیر در رجیسترهای Reg5 و Reg6 ذخیره می‌شوند. خروجی‌های PA و PD لازم است در انتهای هر حلقه‌ی تکرار بطور همزمان آماده باشند. به این منظور خروجی Z_a که در انتهای سیکل پنجم محاسبه شده است در رجیستر Reg7 ذخیره می‌شود تا دیگر خروجی‌های W_a، Z_d و W_d نیز محاسبه شوند. همچنین برای محاسبه W_a، ابتدا خروجی ضرب‌کننده‌ی M_1 در سیکل ۱۰ در رجیستر Reg8 ذخیره می‌شود، سپس این مقدار در سیکل ۱۵ با خروجی ضرب‌کننده‌ی M_2 جمع تا W_a محاسبه شود. بنابراین هر چهار خروجی Z_a، W_a، Z_d و W_d به صورت همزمان در انتهای حلقه محاسبه و آماده شده‌اند. مقادیر اولیه در الگوریتم ۱ برای خم‌های باینری هشیان کلی‌شده در $2(w+1)$ سیکل ساعت توسط ضرب‌کننده‌های M_1 و M_3 محاسبه می‌شوند. همچنین هر حلقه‌ی تکرار در $3(w+1)+1$ سیکل انجام می‌شود. در اولین سیکل، ورودی‌های W_1، Z_1، W_2 و Z_2 به ترتیب در رجیسترهای Reg1، Reg2، Reg3 و Reg4 ذخیره می‌شوند و سپس برای انجام سه مرحله عملیات ضرب میدانی به $3(w+1)$ سیکل نیاز است. بنابراین تعداد کل سیکل‌های ساعت لازم برای محاسبه ضرب نقطه‌ای برای خم‌های باینری هشیان کلی‌شده بر روی میدان $\mathrm{GF}(2^m)$ برابر $(m\text{-}1)(3(w+1)+1)+2(w+1)+1$ می‌باشد.



شکل (۸): گراف محاسبات مربوط به PA و PD برای خم‌های بایزی هشیان کلی‌شده



شکل (۹): ساختار پیشنهادی مربوط به ضرب نقطه‌ای بر اساس الگوریتم نردبان مُنتگومری برای خم‌های بایزی هشیان کلی‌شده

۵- پایه نرمال گوسی و ضرب کننده رقمی-سریال میدانی

پایه نرمال تنها برای میدان‌های متناهی با مشخصه ۲ نمی‌باشد. این پایه برای هر میدان متناهی $GF(q^m)$ قابل تعریف می‌باشد که در آن q توانی از یک عدد اول است. یک پایه نرمال برای میدان $GF(2^m)$ بر روی $GF(2)$ به صورت مجموعه‌ی زیر نمایش داده می‌شود:

$$B=\{\beta^{2^{m-1}},\beta^{2^{m-2}},...,\beta^{2^2},\beta^{2^1},\beta^{2^0}\}$$

که در آن $\beta \in GF(2^m)$ و آن را مولد پایه $\boldsymbol{B}$ می‌نامند. اجزای این مجموعه بر روی میدان $GF(2)$ نسبت به هم مستقل خطی هستند. برای هر میدان باینری یک عنصر نرمال β و یک مجموعه پایه‌ی نرمال $\boldsymbol{B}$ وجود دارد. هر عضو میدان مانند A می‌تواند به صورت زیر نمایش داده شود:

$$A=\sum_{i=0}^{m-1}a_i\beta^{2^i}=a_{m-1}\beta^{2^{m-1}}+a_{m-2}\beta^{2^{m-2}}+...+a_2\beta^{2^2}+a_1\beta^2+a_0\beta$$

در نمایش برداری عنصر A به صورت $(a_{m-1}.a_{m-2}.....a_1.a_0)$ نمایش داده می‌شود که در آن ضرایب a_i عضو میدان $GF(2)$ می‌باشند. در این پایه اعداد صفر و یک بترتیب به صورت $0=(0.0.....0.0)$ و $1=(1.1.....1.1)$ نمایش داده می‌شوند.

پایه نرمال گوسی، یک کلاس خاص از پایه نرمال است که در آن عملیات ضرب به صورت موثری قابل انجام است [28]. پیچیدگی محاسباتی عمل ضرب در پایه نرمال گوسی توسط نوع آن اندازگیری می‌شود. نوع یک پایه نرمال گوسی توسط یک عدد صحیح مثبت که به تعداد عناصر غیر صفر ماتریس ضرب مرتبط است بیان می‌شود. تعداد عناصر غیر صفر در ماتریس ضرب بزرگتر از $2m-1$ است. هر چه عدد مربوط به نوع کوچکتر باشد ضرب دارای پیچیدگی کمتری است.

برای یک میدان متناهی $GF(2^m)$ اگر m قابل قسمت بر عدد ۸ نباشد، می‌توان عدد کوچک صحیح و مثبت T را طوری پیدا کرد که عدد $(p=Tm+1)$ یک عدد اول شود. همچنین اگر عددی مانند k وجود داشته باشد به طوری که مرتبه ۲ در پیمانه عدد p برابر k باشد، در این حالت میدان $GF(2^m)$ دارای یک پایه نرمال از نوع T می‌باشد اگر و تنها اگر $gcd(h.m)=1$ باشد که در آن $h=\frac{Tm}{k}=\frac{p-1}{k}$ است. تحت این شرایط، همیشه برخی عناصر مانند β که شرایط $\beta \in GF(2^m)$، $\beta \neq 1$ و $\beta^{Tm+1}=1$ را داشته باشند وجود دارند. در این حالت به منظور یافتن یک مولد برای پایه نرمال با پیچیدگی کم در $GF(2^m)$ یک عملگر شبه-اثر بر روی β اعمال می‌شود تا آن را از $GF(2^{Tm})$ به $GF(2^m)$ انتقال دهد. در حالتی که $T=2$ است این عملگر برابر $\varepsilon=\beta+\beta^{-1}$ می‌باشد. بنابراین برای mهایی که قابل قسمت بر عدد ۸ نیستند، می‌توان با استفاده از دوره‌های دایره بُری گوسی یک پایه نرمال موثر برای میدان $GF(2^m)$ تعریف نمود.

ضرب میدانی با پایه نرمال به صورت زیر محاسبه می‌شود:

$$C=A\times B:(a_{m-1},a_{m-2},...,a_1,a_0)\times(b_{m-1},b_{m-2},...,b_1,b_0)$$
$$=(c_{m-1},c_{m-2},...,c_1,c_0)$$

عبارت ضرب بر حسب ضرایب c_k برای $0\le k\le m-1$، به صورت زیر می‌باشد:

$$C = A \times B = \sum_{i=0}^{m-1} a_i \beta^{2^i} \times \sum_{j=0}^{m-1} b_j \beta^{2^j} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \beta^{2^j}$$

$$= \sum_{k=0}^{m-1} c_k \beta^{2^k}$$

برای حاصل ضرب $\beta^{2^i}\beta^{2^j}$ داریم:

$$\beta^{2^i}\beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^k}, \lambda_{i,j}^{(k)} = 0 or 1$$

با جایگذاری این عبارت در معادله C، ضرایب c_k به صورت زیر بدست می‌آید:

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{i,j}^{(k)}$$

ماتریس $\lambda^{(k)}$ را می‌توان بر اساس ماتریس $\lambda^{(0)}$ که ماتریس ضرب نیز نامیده می‌شود به صورت $\lambda_{i.j}^{(k)} = \lambda_{i-k.j-k}^{(0)}$ نوشت. با بازنویسی فرمول c_k داریم:

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{i,j}^{(0)}$$

پیاده‌سازی هر ترم c_k، مشابه پیاده‌سازی ترم c_0 و با همان سخت‌افزار و تنها با شیفت چرخشی به چپ ورودی‌ها به اندازه عدد k می‌باشد. به عبارت دیگر برای محاسبه c_k تنها ورودی‌های $(a_{m-1}.a_{m-2}.\dots.a_1.a_0)$ و $(b_{m-1}.b_{m-2}.\dots.b_1.b_0)$ با ورودی‌های $(a_{m-k-1}.\dots.a_1.a_0.a_{m-1}.a_{m-2}.\dots.a_{m-k})$ و $(b_{m-k-1}.\dots.b_1.b_0.b_{m-1}.b_{m-2}.\dots.b_{m-k})$ جایگزین و سپس به مدار ضرب‌کننده اعمال می‌شوند. پیچیدگی سخت‌افزاری یک ضرب-کننده با پایه نرمال به تعداد یک‌های ماتریس $\lambda^{(0)}$ که با m_λ نشان داده می‌شود وابسته است. در عمل محدوده مقادیر m_λ برابر $2m-1 \le m_\lambda \le m^2$ می‌باشد. در پایه نرمال بهینه $m_\lambda = 2m-1$ است.

در ادامه عمل ضرب میدانی با پایه نرمال گوسی براساس مرجع [29] بیان می‌شود. اگر میدان $GF(2^m)$ دارای یک پایه نرمال گوسی از نوع T بوده و $p = mT+1$ یک عدد اول باشد، و نیز u یک عدد صحیح از مرتبه T به پیمانه عدد p باشد، در این صورت مجموعه زیر را داریم:

$$Z = \{z_{i,j} : z_{i,j} = 2^i u^j \mid i \in \{0,1,\dots,m-1\}, j \in \{0,1,\dots,T-1\}\}$$

هر عدد صحیح مثبت x کمتر از p می‌تواند به طور واحد توسط $x = z_{i.j} \bmod p$ نمایش داده شود. اگر F یک تابع به صورت زیر باشد:

$$F : \{1,2,\dots,p-1\} \to \{0,1,\dots,m-1\}$$
$$F(x) = i, \quad x = z_{i,j} \bmod p \qquad (۱)$$

برای هر عدد زوج T، اولین بیت نتیجه ضرب $C = A \times B$ به صورت زیر محاسبه می‌شود:

$$c_0 = \sum_{k=1}^{p-2} a_{F(k+1)} b_{F(p-k)} \qquad (۲)$$

دیگر بیت‌های نتیجه ضرب، c_iها، که در آن $1 \le i \le m-1$ است، بطور مشابه توسط یک بیت شیفت چرخشی به راست ورودی‌ها محاسبه می‌شوند.

در این بخش ضمن بررسی و بازبینی فرمول‌های ریاضی مربوط به عمل ضرب، یک ساختار منظم با تاخیر مسیر بحرانی کم و سخت‌افزار مصرفی مناسب ارائه می‌گردد. ساختار پیشنهادی بر اساس توان رسانی توسط عدد ۲ و مدار ضرب در عنصر نرمال طراحی شده است. ابتدا با بازبینی فرمول‌های ضرب، عنصر نرمال β در فرمول جداسازی می‌شود. بدین ترتیب کل عملیات ضرب به دو دسته شامل عملیات توان رسانی توسط عدد ۲ و ضرب در عنصر نرمال β تبدیل می‌شود. توان رسانی توسط عدد ۲ در پایه نرمال تنها با شیفت چرخشی قابل پیاده‌سازی است. این ویژگی یک فاکتور مهم برای سادگی طراحی و کاهش سخت‌افزار مصرفی می‌باشد. بلوک‌های ضرب در β توسط مدار ترکیبی و با استفاده از گیت‌های XOR پیاده می‌شود که در آن تعدادی از گیت‌های XOR به صورت مشترک استفاده شده‌اند. این امر باعث کاهش تعداد گیت‌ها می‌شود. ساختار ضرب‌کننده‌ی پیشنهادی برای ارقام مختلف قابل پیکربندی است. پیاده‌سازی در سطح لی‌اوت مربوط به این ضرب‌کننده در این بخش آورده شده است.

برای توان رسانی به عدد 2^n این عمل از طریق تکرار عمل مربع‌کردن انجام می‌شود. برای عدد صحیح مثبت n، محاسبات عمل توان رسانی A^{2^n} از طریق n-بیت شیفت چرخشی به چپ به صورت زیر محاسبه می‌شود:

$$A^{2^n} = (a_{m-n-1}, a_{m-n-2}, \dots, a_1, a_0, a_{m-1}, \dots, a_{m-n+1}, a_{m-n})$$

و به صورت مشابه $A^{2^{-n}}$ توسط n-بیت شیفت چرخشی به راست محاسبه می‌شود:

$$A^{2^{-n}} = (a_{n-1}, a_{n-2}, \dots, a_1, a_0, a_{m-1}, \dots, a_{n+1}, a_n)$$

در مرجع [30] یک نمونه پیاده‌سازی بیت-سریال از ضرب‌کننده با پایه نرمال ارائه شده است که در آن ضرب دو عنصر A و B به صورت زیر انجام می‌شود:

$$C = AB = A(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \dots + b_0\beta)$$
$$= b_{m-1}(A\beta^{2^{m-1}}) + b_{m-2}(A\beta^{2^{m-2}}) + \dots + b_0(A\beta)$$
$$= b_{m-1}(A^{2^{-(m-1)}}\beta)^{2^{m-1}} + b_{m-2}(A^{2^{-(m-2)}}\beta)^{2^{m-2}} + \dots + b_0(A\beta)$$
$$= (\dots((b_{m-1}A^{2^{-(m-1)}}\beta)^2 + b_{m-2}A^{2^{-(m-2)}}\beta)^2 \dots)^2 + b_0(A\beta).$$

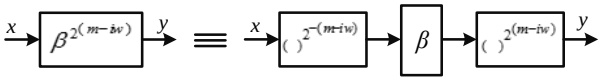
همچنین یک ساختار رقمی-سریال برای ضرب با پایه نرمال در این مرجع ارائه شده است. در این حالت B به d کلمه با w بیت تقسیم شده است:

$$C=(...((b_{m-1}A^{2^{-(w-1)}}\beta^{2^{m-w}})^2+b_{m-2}A^{2^{-(w-2)}}\beta^{2^{m-w}})^2 +...)^2+b_{m-w}A\beta^{2^{m-w}}+(...((b_{m-w-1}A^{2^{-(w-1)}}\beta^{2^{m-2w}})^2 +b_{m-w-2}A^{2^{-(w-2)}}\beta^{2^{m-2w}})^2+...)^2+b_{m-2w}A\beta^{2^{m-2w}} +...+(...((b_{m-(d-1)w-1}A^{2^{-(w-1)}}\beta^{2^{m-dw}})^2+ b_{m-(d-1)w-2}A^{2^{-(w-2)}}\beta^{2^{m-dw}})^2+...)^2+b_0A\beta.$$

در محاسبه عبارت فوق به محاسبه‌ی توان‌های عنصر β نیاز می‌باشد که در [30] توسط یک ساختار پیچیده سخت‌افزاری انجام شده است. در پیاده‌سازی حاضر برای کاهش پیچیدگی محاسبه $\beta^{2^{(m-iw)}}$، محاسبات مربوط به $y=x\beta^{2^{(m-iw)}}$ در سه مرحله انجام می‌گیرد؛ ابتدا ورودی x به توان $2^{-(m-iw)}$ می‌رسد، سپس ضرب در عنصر β انجام می‌شود، و سرانجام نتایج این دو مرحله به توان $2^{(m-iw)}$ می‌رسد. چگونگی انجام این عمل در عبارت زیر نشان داده شده است:

$$y=x\beta^{2^{(m-iw)}}=((x^{2^{-(m-iw)}})\beta)^{2^{(m-iw)}}$$

همچنین شکل ۱۰ روش پیشنهادی برای مرحله‌ی ضرب در $\beta^{2^{(m-iw)}}$ را نشان می‌دهد.



شکل (۱۰): روش پیشنهادی برای ضرب در $\beta^{2^{(m-iw)}}$.

در ساختار پیشنهادی ضرب‌کننده دو مرحله توان رسانی به $2^{-(m-iw)}$ و $2^{(m-iw)}$ تنها بوسیله‌ی شیفت چرخشی و بدون سخت‌افزار اضافی دیگر انجام می‌گیرد و تنها ضرب در عنصر β نیاز به سخت‌افزار دارد. در ساختار ارائه شده، واحد ضرب در عنصر β دارای ساختاری ثابت و مستقل از عملیات شیفت در ساختار ضرب‌کننده می‌باشد. در مرجع [30] مدار مربوط به ضرب در عنصر β وابسته به عملیات شیفت بوده که منجربه پیچیدگی سخت‌افزاری برای میدان‌های بزرگ می‌گردد.

در ادامه ساختار ارائه شده برای ضرب‌کننده‌ی رقمی-سریال با پایه نرمال گوسی توضیح داده می‌شود. اگر دو عنصر A و B اعضای میدان $GF(2^m)$ باشند، عنصر $B=(b_{m-1}.b_{m-2}.....b_2.b_1.b_0)$ بعنوان یک آرایه $d\times w$ به صورت زیر در نظر گرفته می‌شود:

$$\begin{pmatrix} b_{m-1} & b_{m-2} & \cdots & b_{m-w} \\ b_{m-w-1} & b_{m-w-2} & & b_{m-2w} \\ \vdots & & \ddots & \vdots \\ b_{m-(d-1)w-1} & b_{m-(d-1)w-2} & \cdots & b_{m-dw} \end{pmatrix}$$

در اینجا برای $i\le 0$ مقادیر $b_i=0$ است. بر اساس ستون‌های آرایه‌ی فوق داریم:

$$B=B_1+B_2+...B_w,$$

که در آن B_i برای $i=1.....w$ به صورت زیر تعریف می‌شود:

$$B_i=\sum_{k=1}^{d}b_{m-(k-1)w-i}\beta^{2^{m-(k-1)w-i}}$$

حال ضرب دو عنصر A و B به صورت زیر نوشته می‌شود:

$$C=AA=A\sum_{i=1}^{w}B_i$$
$$=A\sum_{i=1}^{w}\sum_{k=1}^{d}b_{m-(k-1)w-i}\beta^{2^{m-(k-1)w-i}}$$
$$=\sum_{i=1}^{w}\sum_{k=1}^{d}b_{m-(k-1)w-i}A\beta^{2^{m-(k-1)w-i}}$$
$$=\sum_{i=1}^{w}(\sum_{k=1}^{d}b_{m-(k-1)w-i}A^{2^{-(w-i)}}\beta^{2^{m-kw}})^{2^{w-i}}$$

به عبارت دیگر داریم:

$$C=\sum_{i=1}^{w}c_i^{2^{w-i}}=((...((c_1^2+c_2)^2+c_3)^2+....)^2+c_w),$$

که در آن برای $i=1.....w$، داریم:

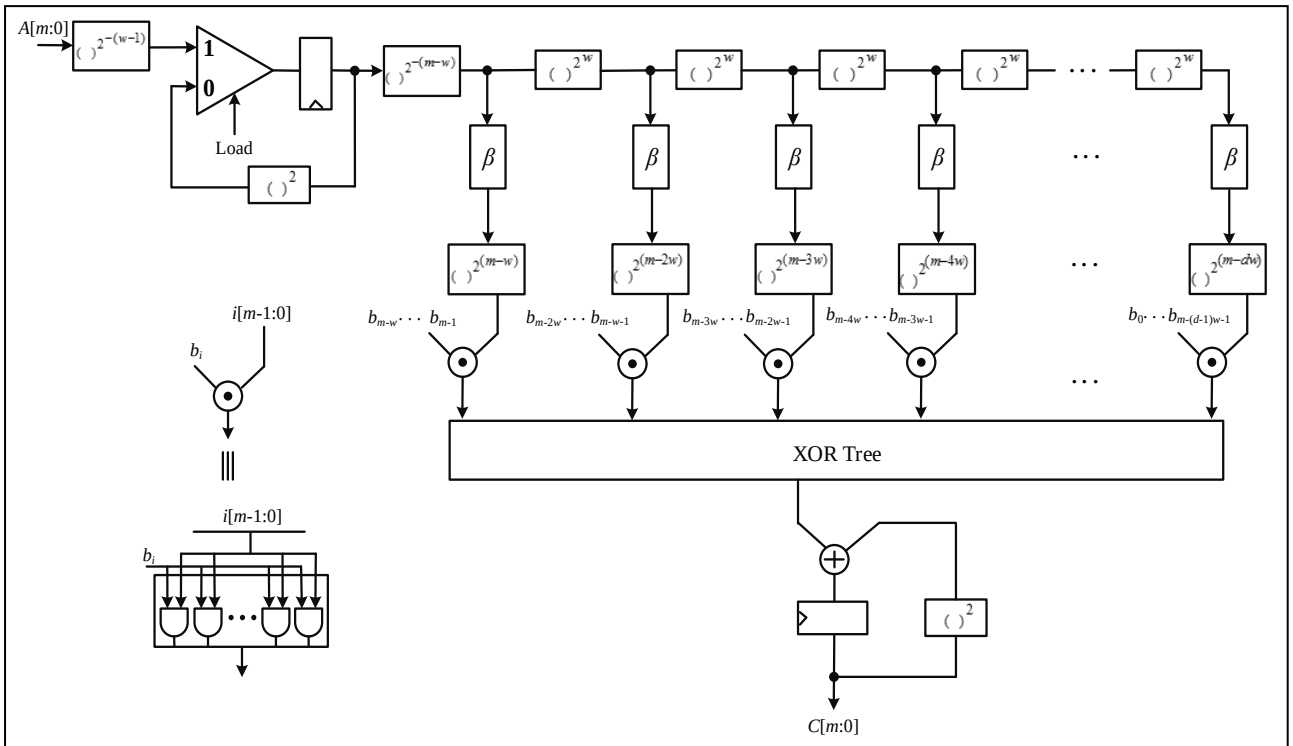
$$c_i=\sum_{k=1}^{d}b_{m-(k-1)w-i}A^{2^{-(w-i)}}\beta^{2^{m-kw}}$$
$$=\sum_{k=1}^{d}b_{m-(k-1)w-i}(((A^{2^{-(w-i)}})^{2^{(i-1)}})^{2^{-2(m-kw)}}\beta)^{2^{m-kw}}.$$

در این رابطه برای محاسبه بخش توان رسانی $2^{-(m-kw)}$، این قسمت به صورت زیر بازنویسی می‌شود:

$$((A^{2^{-(w-i)}})^{2^{(i-1)}})^{2^{-2(m-kw)}}=(...(((A^{2^{-(w-i)}})^{2^{(i-1)}})^{2^{-(m-w)}})^{2^w}...)^{2^w}$$

در این رابطه، ابتدا توان رسانی $2^{-(m-w)}$ محاسبه می‌شود، سپس برای $k=2.3.....d$، توان رسانی‌های $2^{-(m-kw)}$ توسط دنباله‌ای از توان‌های 2^w با طول $d-1$ انجام می‌شود. بدین ترتیب ساختار ضرب‌کننده منظم و ساده می‌گردد و نیز برای اندازه‌های رقم دیگر قابل پیکربندی می‌باشد. شکل ۱۱ ساختار ضرب‌کننده‌ی رقمی-سریال با پایه نرمال گوسی را نشان می‌دهد.

در پیاده‌سازی ضرب در عنصر β در میدان $GF(2^m)$ با نوع T، از روش ارائه شده در مرجع [31] استفاده شده است. در این روش ابتدا بر اساس ماتریس ضرب M، ماتریسی با ابعاد $T\times(m-1)$ به نام R بدست می‌آید. عناصر ماتریس R به صورت $r_{i,j}\in\{0.1.2.....m-1\}$ هستند که در آن $i=0.1.....m-2$ و $j=0.1.....T-1$ می‌باشند. در ادامه دو روش برای محاسبه‌ی ماتریس R بیان می‌شود. با اشاره به اینکه نوع T عددی زوج می‌باشد ماتریس ضرب M متقارن است، بطوری که $\lambda_{i.\,0}^{(k)}=\lambda_{i.k}^{(0)}$ که در آن $i.k$ در مجموعه $\{0.1.2.....m-1\}$ قرار دارند. بنابراین داریم:



شکل (۱۱): ساختار ضرب‌کننده‌ی رقمی-سریال با پایه نرمال گوسی

$$\beta^{2^i}\beta=\sum_{k=0}^{m-1}\lambda_{i,0}^{(k)}\beta^{2^k}=\sum_{k=0}^{m-1}\lambda_{i,k}^{(0)}\beta^{2^k}.$$

این بدین معنی است که، i امین ردیف از ماتریس ضرب M معادل نمایش m-بیتی عبارت $\beta^{2^i}\beta$ است. بنابراین، تعداد اعضای '1' در اولین ردیف برابر یک و در دیگر ردیف‌ها برابر یک عدد زوج کوچکتر یا مساوی T می‌باشد. در روش اول محاسبه‌ی ماتریس R، اعضای i امین ردیف ماتریس R بر اساس شماره ستون اعضای '1' در $(i+1)$ امین ردیف ماتریس M محاسبه می‌شوند. اگر تعداد اعضای '1' در ردیف $(i+1)$ ماتریس M برابر عدد T باشد همه‌ی اعضای ردیف i در ماتریس R طبق روش گفته شده محاسبه می‌شوند. در غیر این صورت بقیه اعضای ماتریس R با یک مقدار ثابت مقداردهی می‌شوند. در روش دوم، ماتریس R بر اساس تابع F که در رابطه (1) بیان شده است محاسبه می‌شود. در این روش، برای $k=1.2.\ldots.p-2$، که در آن $p=mT+1$ است زوج‌های $(F(k+1),F(p-k))$ محاسبه می‌شوند. برای هر زوج محاسبه شده اگر $F(k+1)\geq 1$ باشد مقدار $F(p-k)$، برابر عضو ماتریس R در ردیف $F(k+1)-1$ این ماتریس می‌باشد. بنابراین ضرب در عنصر β بر اساس ماتریس R به صورت زیر می‌باشد:

$$\begin{aligned}\beta B&=\beta\sum_{i=0}^{m-1}b_i\beta^{2^i}=\sum_{i=0}^{m-1}b_i\beta\beta^{2^i}=\sum_{i=0}^{m-1}b_i\sum_{k=0}^{m-1}\lambda_{i,0}^{(k)}\beta^{2^k}\\&=\sum_{i=0}^{m-1}\sum_{k=0}^{m-1}b_i\lambda_{i,0}^{(k)}\beta^{2^k}\\&=\sum_{k=0}^{m-1}\sum_{i=0}^{m-1}b_i\lambda_{i,0}^{(k)}\beta^{2^k}\\&=\sum_{k=0}^{m-1}(\sum_{i=0}^{m-1}b_i\lambda_{i,0}^{(k)})\beta^{2^k}\\&=\sum_{k=0}^{m-1}\sum_{j=0}^{T-1}b_{r_{k,j}}\beta^{2^k}=\sum_{k=0}^{m-1}s(k,B)\beta^{2^k},\end{aligned}$$

که در آن $s(k.B)=\sum_{j=0}^{T-1}b_{r_{k.j}}$ برای $0\leq k\leq m-2$ است. به طور خلاصه داریم:

$$\beta B=(s(m-1,B),\ldots,s(2,B),s(1,B),b_1)$$

جدول (۲): مقایسه ساختارهای پیشنهادی و دیگر کارهای ارائه شده برای خم‌های بیضوی باینری ادواردز و هشیان کلی‌شده

Works	Field size	Device	Area	Fmax (MHz)	Latency (Cycle)	Time (µs)	Efficiency
[20] BECs $(d_1 \neq d_2)$,	233	Virtex-4 XC4VLX140	21816 Slices	47.384	9008	190	56
[21] BECs $(d_1 \neq d_2)$ d=33	163	Virtex-5 XC5VLX110	4681 Slices	265.8	7542	28.3	1230
[21] BECs $(d_1 \neq d_2)$ d=41		Virtex-5 XC5VLX110	5788 Slices	264.5	6709	25.3	1113
[21] BECs $(d_1 = d_2)$ d=33, 2015		Virtex-5 XC5VLX110	4681 Slices	265.8	5911	22.2	1569
[21] BECs $(d_1 = d_2)$ d=41		Virtex-5 XC5VLX110	5788 Slices	264.5	5243	19.8	1422
[21] GBHCs d=33		Virtex-5 XC5VLX110	4681 Slices	268.2	5415	20.1	1732
[21] GBHCs d=41		Virtex-5 XC5VLX110	5788 Slices	267.1	4747	17.7	1591
[21] BECs $(d_1 \neq d_2)$ d=55		Virtex-4 XC4VLX100	12834 Slices	---	---	22.9	555
[21] BECs $(d_1 = d_2)$ d=55		Virtex-4 XC4VLX100	12834 Slices	---	---	23.3	545
[21] BHCs (c=1) d=55		Virtex-4 XC4VLX100	12834 Slices	---	---	20.8	610
[19] BECs $(d_1 \neq d_2)$ d=33	163	Virtex-4 XC4VLX160	27778 Slices	217.2	3808	17.5	335
[19] BECs $(d_1 \neq d_2)$ d=26	233	Virtex-4 XC4VLX160	29252 Slices	198.4	7212	36.3	219
[19] GBHCs d=33	163	Virtex-4 XC4VLX160	15992 Slices	218.2	3471	15.9	641
[19] GBHCs d=26	233	Virtex-4 XC4VLX160	16940 Slices	205.1	6791	33.1	416
[22] BECs $(d_1 = d_2)$	233	Virtex-5 XC5VLX110	32874 LUTs	25	---	132	284
[23] BECs $(d_1 \neq d_2)$ d=26	233	Virtex-4 XC4VLX100	17164 Slices	127.575	---	25.4	534
[23] BECs $(d_1 \neq d_2)$ d=26	233	Virtex-5 XC5VLX110	2662 Slices	165.115	---	19.64	4456
[25] BECs $(d_1 = d_2)$	163	Virtex-5 XC5VLX110	29309 LUTs	211	547	2.6	2139
Proposed BECs $(d_1 \neq d_2)$ d=33	**163**	**Virtex-4 XC4VLX100**	**22957 Slices**	**253.873**	**3091**	**12.18**	**583**
Proposed BECs $(d_1 \neq d_2)$ d=41		**Virtex-4 XC4VLX100**	**27365 Slices**	**247.396**	**2603**	**10.52**	**566**
Proposed BECs $(d_1 = d_2)$ d=33		**Virtex-4 XC4VLX100**	**17125 Slices**	**254.996**	**3085**	**12.1**	**787**
Proposed BECs $(d_1 = d_2)$ d=41		**Virtex-4 XC4VLX100**	**20853 Slices**	**247.750**	**2598**	**10.49**	**745**
Proposed GBHCs d=33		**Virtex-4 XC4VLX100**	**17052 Slices**	**254.808**	**3091**	**12.13**	**788**
Proposed GBHCs d=41		**Virtex-4 XC4VLX100**	**20752 Slices**	**247.037**	**2603**	**10.54**	**745**
Proposed BECs $(d_1 \neq d_2)$ d=33		**Virtex-5 XC5VLX110**	**9624 Slices**	**331.363**	**3091**	**9.33**	**1815**
Proposed BECs $(d_1 \neq d_2)$ d=41		**Virtex-5 XC5VLX110**	**11397 Slices**	**302.081**	**2603**	**8.62**	**1659**
Proposed BECs $(d_1 = d_2)$ d=33		**Virtex-5 XC5VLX110**	**7314 Slices**	**331.363**	**3085**	**9.31**	**2394**
Proposed BECs $(d_1 = d_2)$ d=41		**Virtex-5 XC5VLX110**	**8645 Slices**	**302.093**	**2598**	**8.6**	**2192**
Proposed GBHCs d=33		**Virtex-5 XC5VLX110**	**7313 Slices**	**331.363**	**3091**	**9.33**	**2389**
Proposed GBHCs d=41		**Virtex-5 XC5VLX110**	**8645 Slices**	**302.093**	**2603**	**8.62**	**2187**
Proposed BECs $(d_1 \neq d_2)$ d=26	**233**	**Virtex-4 XC4VLX100**	**18278 Slices**	**333.970**	**7213**	**21.6**	**590**
Proposed BECs $(d_1 \neq d_2)$ d=59		**Virtex-4 XC4VLX100**	**37053 Slices**	**277.691**	**3723**	**13.41**	**467**
Proposed BECs $(d_1 = d_2)$ d=26		**Virtex-4 XC4VLX100**	**13786 Slices**	**333.970**	**7203**	**21.57**	**784**
Proposed BECs $(d_1 = d_2)$ d=59		**Virtex-4 XC4VLX100**	**31702 Slices**	**277.681**	**3718**	**13.39**	**549**
Proposed GBHCs d=26		**Virtex-4 XC4VLX100**	**14052 Slices**	**333.970**	**7213**	**21.6**	**768**
Proposed GBHCs d=59		**Virtex-4 XC4VLX100**	**27933 Slices**	**277.681**	**3723**	**13.41**	**622**
Proposed BECs $(d_1 \neq d_2)$ d=26		**Virtex-5 XC5VLX110**	**6547 Slices**	**391.932**	**7213**	**18.40**	**1934**
Proposed BECs $(d_1 \neq d_2)$ d=59		**Virtex-5 XC5VLX110**	**14343 Slices**	**337.603**	**3723**	**11.03**	**1473**
Proposed BECs $(d_1 = d_2)$ d=26		**Virtex-5 XC5VLX110**	**4987 Slices**	**391.932**	**7203**	**18.38**	**2542**
Proposed BECs $(d_1 = d_2)$ d=59		**Virtex-5 XC5VLX110**	**11494 Slices**	**337.603**	**3718**	**11.01**	**1841**
Proposed GBHCs d=26		**Virtex-5 XC5VLX110**	**5045 Slices**	**391.932**	**7213**	**18.40**	**2510**

۶- نتایج و مقایسه

در این بخش، سخت افزار و پیچیدگی زمانی ساختارهای پیشنهادی با سایر ساختارهای ضرب نقطه ای روی خم های باینری ادواردز و هشیان کلی شده مقایسه می شوند. در جدول ۲ مقایسه‌ای نتایج پیاده‌سازی کار پیشنهادی و کارهای قبلی ارائه شده برای ضرب نقطه ای روی خم‌های بیضوی باینری ادواردز و هشیان کلی‌شده بر روی دو میدان ($GF(2^{163})$ و $GF(2^{233})$) آورده شده است. در این جدول پارامترهای مورد مقایسه بر اساس منابع سخت‌افزاری، حداکثر فرکانس، زمان اجرا و ارزیابی کلی عملکرد مدار می‌باشند. مقدار بازدهی با استفاده از رابطه زیر محاسبه شده است:

$$Efficiency = \frac{Number\ of\ bits}{Time \times Slices}$$

ساختار ارائه شده برای خم‌های باینری ادواردز در حالت $d_1 \neq d_2$ روی میدان $GF(2^{233})$ دارای مشخصات سخت‌افزاری و زمانی بهتری نسبت به [۱۹] است. برای خم‌های باینری ادواردز در حالت $d_1 \neq d_2$ برای میدان $GF(2^{163})$، ساختار ارائه شده روی Virtex-5 دارای بترتیب ۶۷% و ۲۵% پارامترهای زمان اجرا و حداکثر فرکانس کاری بهتر نسبت به [20] می‌باشد، اما سخت‌افزار مصرفی در [20] نسبت به کار ارائه شده کمتر می‌باشد. با این وجود کار ارائه شده دارای بازدهی بهتری نسبت به [20] است. همچنین نتایج مشابه‌ای برای خم‌های بیضوی باینری ادواردز در حالت $d_1 = d_2$ و خم‌های باینری هشیان کلی‌شده روی میدان $GF(2^{163})$ نسبت به [20] وجود دارند. روش پیشنهادی برای خم‌های باینری ادواردز حالت $d_1 \neq d_2$ برای میدان-های $GF(2^{233})$ و $GF(2^{163})$ روی Virtex-4 به ترتیب ۴۰% و ۳۰% زمان اجرای کمتری نسبت به [21] دارد. علاوه بر این بازدهی و سخت‌افزار مصرفی نسبت به [21] بهبود داده شده است. برای خم‌های باینری هشیان کلی‌شده روی میدان‌های $GF(2^{163})$ و $GF(2^{233})$ نیز ساختار ارائه شده در [21] به ترتیب دارای ۲۴% و ۳۵% زمان اجرای بیشتر نسبت به کار پیشنهادی برای اندازه رقم یکسان می‌باشد.

۷- نتیجه‌گیری

در این مقاله، طراحی و پیاده‌سازی ضرب نقطه‌ای خم‌های باینری ادواردز و هشیان کلی‌شده بر اساس الگوریتم نردبان مُنتگومری بر روی میدان $GF(2^m)$ با پایه نرمال گوسی انجام شده است. مختصات استفاده شده مختصات تفاضلی w-coordinate می‌باشد. ضرب-کننده‌ی میدانی استفاده شده با پایه نرمال گوسی و به صورت خط لوله‌ای و دارای ساختار رقمی-سریال است. در ساختار ارائه شده عمل ضرب نقطه‌ای برای خم‌های بیضوی باینری ادواردز برای دو حالت کلی و خاص آن به ترتیب از چهار و سه ضرب‌کننده‌ی میدانی استفاده شده است. در این خم‌ها عملیات میدانی مربوط به عمل جمع دو نقطه و دو برابر کردن یک نقطه به صورت موازی توسط سه مرحله ضرب میدانی انجام می‌شوند. همچنین از سه ضرب‌کننده‌ی میدانی برای خم باینری هشیان کلی‌شده استفاده شده است. ضرب‌کننده‌ها با هدف کاهش تعداد سیکل ساعت، زمان‌بندی و به اشتراک گذاشته شده‌اند. نتایج نشان می دهد بهبود کلی از نظر زمان اجرا ، منابع سخت افزاری و کارایی در مقایسه با آثار قبلی گزارش شده است. به عنوان مثال، در مورد خم های باینری ادواردز روی ($GF(2^{233})$ (در Virtex-4 XC4VLX110 FPGA) ساختار پیشنهادی می تواند به ترتیب تا ۳۷% و ۴۰% از منابع سخت افزاری و زمان اجرا را کاهش و بازدهی را ۶۲% در مقایسه با بهترین کار موجود بهبود دهد.

مراجع

[1] Koblitz, N., "Elliptic curve cryptosystems", Mathematics of Computation, 1987, pp. 203-209.

[2] Miller , V. S., "Use of elliptic curve in cryptography", Advances in Cryptology, Crypto'85 Proceedings, 1986, pp. 417-426.

[3] Sutter, G.D., Deschamps, J.P., Imana, J.L., "Efficient elliptic curve point multiplication using digit-serial binary field operations", IEEE Trans. Ind. Electron., Vol. 60, No. 1, 2013, pp. 217-225.

[4] Jarvinen, K., and Skytta, J., "On Parallelization of High-Speed Processors for Elliptic Curve Cryptography", IEEE Trans. on Very Large Scale Integration (VLSI) Systems, Vol. 16, No. 9, 2008, pp. 1162-1175.

[5] Masoumi, M., Mahdizadeh, H., "Efficient Hardware Implementation of an Elliptic Curve Cryptographic Processor over GF (2^{163})", World Academy of Science, Engineering and Technology, Vol. 65, 2012, pp. 1223-1230.

[6] Chelton, W. N. and Benaissa, M., "Fast elliptic curve cryptography on FPGA", IEEE Trans. on VLSI Systems, Vol. 16, No. 2, 2008, pp. 198-205.

[7] Choi, H. M., Hong, C. P., Kim, C. H., "High Performance Elliptic Curve Cryptographic Processor Over $GF(2^{163})$", in Proc. of 4th IEEE International Symposium on Electronic Design, Test & Application, 2008, pp. 290-295.

[8] Mahdizadeh, H. and Masoumi, M., "Novel Architecture for Efficient FPGA Implementation of Elliptic Curve Cryptographic Processor Over $GF(2^{163})$", IEEE Trans. on VLSI Systems, Vol. 21, Iss. 12, 2013, pp. 2330-2333.

[9] Zhang, Y., Chen, D., Choi, Y., Chen, L., and Ko, S.-B., "A high performance ECC hardware implementation with instruction-level parallelism over GF (2^{163})", Microprocess. Microsyst., Vol. 34, No. 6, 2010, pp. 228-236.

[10] Fayed, M. A., Watheq, El-Kharashi, M., Gebali, F., "A High-Speed, High-Radix, Processor Array Architecture for Real-Time Elliptic Curve Cryptography over $GF(2^m)$", in Proc. of IEEE International Symposium on Signal Processing and Information Technology, 2007, pp. 56-61.

[11] Cinnati Loi, K. C., Sen A., and Ko, S.B., "FPGA Implementation of Low Latency Scalable Elliptic Curve Cryptosystem Processor in $GF(2^m)$", in Proc. of IEEE

international Symposium on Circuits and Systems (ISCAS), 2014, pp. 822-825.
[12] Roy, S.S., Rebeiro, C. and Mukhopadhyay, D., "Theoretical Modeling of Elliptic Curve Scalar Multiplier on LUT-Based FPGAs for Area and Speed", IEEE Trans. on VLSI Systems, Vol. 21, No. 5, 2013, pp. 901-909.
[13] Rebeiro, C., Roy, S.S. and Mukhopadhyay, D., "Pushing the Limits of High-Speed GF(2^m) Elliptic Curve Scalar Multiplication on FPGAs", in Proc. of First International Workshop Cryptographic Hardware and Embedded Systems (CHES), 2012, pp. 494-511.
[14] Khan, Z.U.A. and Benaissa, M., "Throughput/Area Efficient ECC Processor using Montgomery Point Multiplication on FPGA", IEEE Trans. circuits and systems-II express briefs, Vol. 62, Iss. 11, 2015, pp. 1078-1082.
[15] Rashidi, B., Sayedi, S.M., and Rezaeian Farashahi, R., "High-speed Hardware Architecture of Scalar Multiplication for Binary Elliptic Curve Cryptosystems", Microelectronics Journal, Vol. 52, 2016, pp. 49-65.
[16] Rashidi, B., Rezaeian Farashahi, R., and Sayedi, S.M., "High-performance and high-speed implementation of polynomial basis Itoh-Tsujii inversion algorithm over GF(2^m)", IET Inf. Secur., Vol. 11 Iss. 2, 2017, pp. 66-77.
[17] Khan, Z.U.A. and Benaissa, M., "High-Speed and Low-Latency ECC Processor Implementation over GF(2^m) on FPGA", IEEE Trans. on VLSI Systems, Vol. 25, No. 1, 2017, pp. 165-176.
[18] Li, L. and Li, S., "High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication over GF(2^m)", IEEE Trans. Very Large Scale Integr. Syst., Vol. 24, Iss. 4, 2016, pp. 1223-1232.
[19] Chatterjee, A., Sengupta I., "Design of a high performance Binary Edwards Curve based processor secured against side channel analysis", Integration, the VLSI Journal, Vol. 45, No. 3, 2012, pp. 331-340.
[20] Azarderakhsh, R. and Reyhani-Masoleh, A., "Efficient FPGA Implementations of Point Multiplication on Binary Edwards and Generalized Hessian Curves Using Gaussian Normal Basis", IEEE Trans. on VLSI Systems, Vol. 20, No. 8, 2012, pp. 1453-1466.
[21] Azarderakhsh, R. and Reyhani-Masoleh, A., "Parallel and High-Speed Computations of Elliptic Curve Cryptography Using Hybrid-Double Multipliers", IEEE Trans. on VLSI Systems, Vol. 26, Iss. 6, 2015, pp. 1668-1677.
[22] Fournaris, AP., Sklavos, N. and Koulamas, C., "A High Speed Scalar Multiplier for Binary Edwards Curves", in Proc. of Third Workshop on Cryptography and Security in Computing Systems, ACM, 2016, pp. 41-44.
[23] Asher Sajid, Muhammad Rashid, Malik Imran and Atif Raza Jafri,"A Low-Complexity Edward-Curve Point Multiplication Architecture", Electronics, Vol. 10, No. 9, 2021, pp. 1-16.
[24] M. Kalaiarasi, V. R. Venkatasubramani, V. Vinoth Thyagarajan & S. Rajaram, "A parallel elliptic curve crypto-processor architecture with reduced clock cycle for FPGA platforms", The Journal of Supercomputing, Vol. 78, 2022, pp. 15567-15597.
[25] Jiakun Li; Shun'an Zhong; Zhe Li; Shan Cao; Jingqi Zhang, "Speed-Oriented Architecture for Binary Field Point Multiplication on Elliptic Curves", IEEE Access, Vol. 7, 2019, pp. 32048- 32060.
[26] MD. Mainul Islam; MD. Selim Hossain; MD. Shahjalal; MOH. Khalid Hasan, "Area-Time Efficient Hardware Implementation of Modular Multiplication for Elliptic Curve Cryptography", IEEE Access, Vol. 8, 2020, pp. 73898-73906.
[27] Pradeep Kumar Goud Nadikuda & Lakshmi Boppana, "Low area-time complexity point multiplication architecture for ECC over GF(2^m) using polynomial basis", Journal of Cryptographic Engineering, Vol 68, 2022, pp. 1-10.
[28] Ash, D.W., Blake, I.F., and Vanstone, S.A., "Low Complexity Normal Bases", Discrete Applied Math., Vol. 25, 1989, pp. 191-210.
[29] IEEE P1363: Editorial Contribution to standard for Public Key Cryptography, 2003.
[30] Reyhani-Masoleh, A., "Efficient Algorithms and Architectures for Field Multiplication Using Gaussian Normal Bases", IEEE Trans. Computers, Vol. 55, No. 1, Jan. 2006, pp. 34-47.
[31] Sukcho, Y., Yeon Choi, J., "A new Word-parallel bit-serial Normal basis multiplier over GF(2^m)", International Journal of control and Automation, Vol. 6, No. 3, June 2013, pp. 209-216.

[1] Elliptic Curve Cryptography (ECC)

[2] Differential addition and doubling