

## الگوریتم پر سرعت برای تقسیم کننده‌های مدولار $GF(2^m)$ در رمزنگاری

هادی شهریار شاه حسینی

دانشکده مهندسی برق

دانشگاه علم و صنعت ایران

hshsh@just.ac.ir

عبداله عبدالله زاده

پژوهشکده الکترونیک

دانشگاه علم و صنعت ایران

amir.abz@gmail.com

**چکیده:** در این مقاله، یک الگوریتم پرسرعت برای انجام عملیات تقسیم مدولار ارائه شده است. تقسیم کننده‌های مدولار در بسیاری از روش‌های رمزنگاری، بخصوص رمزنگاری منحنی بیضوی کاربرد فراوان دارند. الگوریتم ارائه شده بر پایه الگوریتم تقسیم GCD میباشد و در رمزنگارهای پرسرعت میتواند جایگزین الگوریتم‌های رایج پیشین برای تقسیم باشد. در تقسیم کننده ارائه شده با بردن الگوریتم عملیات از مبنای دو به مبنای چهار، سرعت الگوریتم دو برابر شده و سپس با ایجاد تغییرات ساده و بهبود الگوریتم، تعداد شرط‌های آن برای پیاده سازی نرم افزاری و سخت‌افزاری کمتر و بهینه تر شده است. شایان ذکر است که دو برابر شدن سرعت الگوریتم، تنها با افزایش ۲۹٪ در سطح تراشه به دست آمده است.

**واژه های کلیدی:** تقسیم کننده مدولار، میدان‌های محدود، الگوریتم GCD

### ۱- مقدمه

پیاده سازی اعمال ریاضی (مانند جمع، ضرب و تقسیم) روی میدان محدود، در سیستم‌های ارتباطی پرسرعت امروزی بسیار مورد توجه است. این توجه با گسترش فناوریهای ارتباطی و مخابراتی و همینطور نیاز به امنیت در ارتباطات و تصحیح خطا در آن هر روز بیشتر می‌شود. استفاده از میدان‌های محدود در کاربرد هایی مانند رمزنگاری<sup>۱</sup> و تصحیح خطا<sup>۲</sup> بسیار رایج میباشد [۱-۲].

از میان تمام اعمال ریاضی روی میدان‌های محدود، تقسیم، برای پیاده سازی از همه پر هزینه تر می‌باشد. سطحی که برای پیاده سازی واحد تقسیم کننده روی تراشه اشغال می‌شود، از بقیه اعمال ریاضی روی میدان‌های محدود بیشتر است. این

مساله در مورد زمان صرف شده برای بدست آوردن پاسخ در

حین اجرای عملیات نیز صدق می‌کند.

با ظهور رمزنگاری منحنی بیضوی، ارائه طرح هایی برای پیاده سازی بهینه تقسیم مورد توجه قرار گرفته‌است و بسیاری از دانشمندان سعی زیادی برای پیاده سازی آن کرده‌اند [۳]. از این میان تقریباً سه روش کلی ارائه شده که اغلب پیاده سازی‌های تقسیم کننده‌ها بر پایه این سه روش می‌باشد. روش اول پیاده سازی بر اساس رابطه فرما<sup>۳</sup> می‌باشد [۴-۷]. روش دوم بر اساس حل دستگاه معادلات خطی در میدان محدود  $GF(2)$  [۸]، و روش سوم بر اساس الگوریتم بسط دودویی بزرگترین مقسوم علیه مشترک (GCD) می‌باشد. که از این سه روش، روش سوم از سایر الگوریتمها برای پیاده سازی تقسیم شناخته شده تر است. [۹-۱۱].

<sup>1</sup> Cryptography

<sup>2</sup> Error correction code

<sup>3</sup> Fermat's theorem

$$A = \sum_{i=0}^{m-1} a_i \alpha^i \quad a_i \in GF(2) \quad (1)$$

جمع کردن روی این میدان به صورت جمع تک تک عنصرها و اعضا به صورت بیت به بیت و به پیمانه دو می باشد که به صورت یک گیت XOR قابل پیاده سازی می باشد. یعنی

$$A + B = \sum_{i=0}^{m-1} (a_i + b_i) \alpha^i = \sum_{i=0}^{m-1} (a_i \oplus b_i) \alpha^i \quad (2)$$

تفریق نیز در میدان محدود معادل جمع و بصورت XOR کردن دو عملوند (رابطه ۲) تعریف می شود.

پیاده سازی ضرب کمی پیچیده تر است، به این گونه که اگر فرض کنیم  $C'$  یک عدد در پایه چند جمله ای با درجه  $2m-2$  باشد، پاسخ ضرب  $C = A \times B$  به صورت زیر تعریف می شود:

$$C = A \times B = \sum_{i=0}^{m-1} c_i \alpha^i = C' \bmod P(\alpha) \quad (3)$$

که  $C'$  ضرب دو چند جمله ای  $A$  و  $B$  و بصورت رابطه ۴ خواهد بود:

$$C' = \left( \sum_{i=0}^{m-1} a_i \alpha^i \right) \cdot \left( \sum_{j=0}^{m-1} b_j \alpha^j \right) = \sum_{k=0}^{2m-2} c_k \alpha^k \quad (4)$$

که در آن

$$c_k = \sum_{i+j=k} a_i \cdot b_j$$

$$0 \leq i, j \leq m-1 \quad \& \quad 0 \leq k \leq 2m-2$$

برای ساده کردن  $C'$  در رابطه ۳ به یک چند جمله ای از درجه  $m-1$  میتوان از رابطه ۵ و ضرایب آن کمک گرفت و ضرایب با درجه بزرگ تر از  $m-1$  را به چند جمله ای های کوچکتر آن تبدیل کرد.

$$x^m = \sum_{i=0}^{m-1} p_i x^i \quad (5)$$

تقسیم نیز بر مبنای چند جمله ای در میدان محدود به صورت زیر تعریف میشود:

$$C = \frac{A}{B} \quad (6)$$

که در آن

$$C \times B \bmod P(x) = A$$

در این مقاله ساختار و الگوریتمی برای پیاده سازی تقسیم کننده های بر مبنای روش سوم پیشنهاد شده است که سرعت آن دو برابر تقسیم کننده های معمول ارایه شده در این روش میباشد. اگرچه این تقسیم کننده سطح بیشتری را روی تراشه اشغال می کند، ولی با توجه به بالا رفتن سرعت و نیازهای ارتباطات پرسرعت در شبکه ها، استفاده از این تقسیم کننده برای رمزنگارهای آتی بسیار مفید خواهد بود.

ادامه این مقاله به صورت زیر دسته بندی شده است: بخش دوم به بیان مقدمه ای بر میدان های محدود می پردازد که در آن پیش زمینه ریاضی میدان های محدود بیان می شود. در بخش سوم تقسیم کننده های رایج بسط دودویی GCD مرور می شود. بخش چهارم به معرفی یک تقسیم کننده اولیه بر مبنای چهارم می پردازد. در بخش پنجم پیش فرض های اولیه برای تقسیم کننده مبنای چهارم مرور شده و در بخش ششم الگوریتم بهبود یافته برای تقسیم مدولار مبنای چهارم ارایه می شود. فصل هفتم به نتایج پیاده سازی سخت افزاری و فصل هشتم به جمع بندی مباحث مطرح شده می پردازد.

## ۲- مقدمه ای بر میدان های محدود $GF(2^m)$

میدان محدود  $GF(2^m)$ ، برای تمام مقادیر  $m$ ، شامل  $2^m$  عضو می باشد که اعضای آن مجموعه هایی از ضرایب  $GF(2)$  یعنی صفر و یک هستند. محدود بودن میدان به بسته بودن میدان نسبت به عملیات ریاضی اشاره دارد؛ بطوریکه مثلاً حاصل ضرب دو عدد  $m$  بیتی در میدان  $GF(2^m)$ ، عددی حداکثر  $m$  بیتی نتیجه خواهد داد.

پایه یا چند جمله ای مبنای، نمادی برای نمایش میدان محدود است که با یک چند جمله ای غیر قابل تقسیم<sup>۵</sup>، مانند  $P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i$ ،  $(p_i \in GF(2))$  تعریف می شود. اگر فرض کنیم که  $\alpha \in GF(2^m)$  ریشه  $P(x)$  باشد یعنی  $P(\alpha) = 0$  آنگاه مجموعه  $\{1, \alpha, \alpha^2, \dots, \alpha^{m-1}\}$  به پایه یا مبنای چند جمله ای  $GF(2^m)$  باز می گردد که برای هر  $A \in GF(2^m)$  داریم:

<sup>4</sup> Polynomial basis  
<sup>5</sup> Irreducible

```

input:  $A(x), B(x), P(x)$ 
output:  $A(x)/B(x) \bmod P(x)$ 
 $R = B(x); S = P(x); U = A(x); V = 0;$ 
for  $i = 1$  to  $2m$ 
  if (state = 0) then
    cnt = cnt + 1;
    if ( $r_0 = 1$ ) then
       $(R, S) = (S + R, R); (U, V) = (V + U, U);$ 
      state = 1;
  else
    cnt = cnt - 1;
    if ( $r_0 = 1$ ) then
       $(R, S) = (S + R, S); (U, V) = (V + U, V);$ 
    if (cnt = 0) then
      state = 0;
 $R = R/x;$ 
 $U = U/x \bmod P(x);$ 
return (V);
    
```

شکل ۱: الگوریتم رایج برای GCD مبنای دو ارایه شده در [۹]

### ۳- تقسیم کننده‌های معمول GCD مبنای دو

تقسیم کننده‌های معمول GCD مبنای دو به طور معمول عملیات تقسیم را برای میدان محدود  $GF(2^m)$ ، تقریباً در  $2m$  ضربه ساعت انجام میدهند که در آن طول میدان محدود میباشد. ثابت نبودن زمان انجام عمل تقسیم مدولار، استفاده از آن را در پردازنده های رمزنگار مشکل می کرد. در سال ۲۰۰۲ میلادی، الگوریتم بسط دودویی بزرگترین مقسوم علیه مشترک بهبود داده شد، به گونه‌ای که در آن با استفاده از متغیرهای حالت محلی و استفاده از دستور for بجای دستور while، زمان انجام عمل تقسیم به ازای تمام ورودی ها ثابت نگاه داشته می شود [۹]. این مساله برای پیاده‌سازی سخت‌افزاری بسیار با اهمیت بود زیرا از یک طرف پیاده سازی با فرمان for بسیار ساده‌تر از پیاده سازی سخت‌افزاری با فرمان while می‌باشد، و از طرف دیگر، در پیاده سازی با الگوریتم جدید مقدار دقیق کلاک هایی که صرف تقسیم می‌شود معلوم بود و این، استفاده از تقسیم کننده را در پردازنده های رمز نگار VLSI ممکن می‌کرد.

### ۴- الگوریتم اولیه برای تقسیم کننده های GCD

#### مبنای ۴

الگوریتم اولیه‌ای که برای تقسیم در  $m$  ضربه ساعت طراحی شده است، در حقیقت فشرده شده الگوریتم شکل ۱ می‌باشد که به صورت الگوریتم ۲ در شکل ۲ ارایه شده است.

اگر چه سرعت این الگوریتم دو برابر سرعت الگوریتم شکل ۱ است ولی در پیاده سازی سخت‌افزاری، این الگوریتم فضایی نزدیک دو برابر فضای الگوریتم شکل ۱ روی تراشه اشغال می‌کند. همین طور با توجه به زیاد بودن تعداد شرط‌ها، پیاده سازی نرم‌افزاری آن نیز چندان مناسب به نظر نمی‌آید. به همین دلیل تغییراتی روی این الگوریتم اعمال شده است تا بتوان آن را به صورت بهینه ای پیاده سازی نمود.

الگوریتم بسط دودویی GCD بر پایه سه فرضی که در زیر می‌آید قرار دارد. این سه فرض در جدول یک ارایه شده اند، که در آن  $A, S \in GF(2^m)$  بوده، و  $a_0$  و  $s_0$  بیت آخر یا کم ارزش هر کدام از آنها می‌باشند.

جدول ۱: پیش فرض های تقسیم کننده GCD مبنای دو

$a_0$	$s_0$	The equal GCDs
0	0	$GCD(A, S) \rightarrow 2GCD(A/2, S/2)$
0	1	$GCD(A, S) \rightarrow GCD(A/2, S)$
1	1	$GCD(A, S) \rightarrow GCD((A-S)/2, S)$

الگوریتم GCD مبنای دو که برای تقسیم، بر اساس این سه فرض ارایه شده، در شکل ۱ آمده است.

در این الگوریتم  $P(x)$ ، چند جمله ای غیر قابل تقسیم میدان و  $V(x)$  در انتهای الگوریتم برابر نتیجه  $A(x)/B(x) \bmod P(x)$  خواهد بود.





الگوریتم ۱ نصف می‌باشد و این الگوریتم بجای  $2m$  ضربه ساعت برای تقسیم به  $m$  ضربه ساعت نیاز دارد.

معماری سخت افزاری این الگوریتم در شکل ۴ ارائه شده است. در این معماری در اولین ضربه ساعت، رجیسترهای ورودی  $(A(x), B(x), P(x))$  مقدار دهی شده و پس از  $m$  ضربه ساعت، حاصل تقسیم در خروجی قرار خواهد گرفت. از جهت پیاده سازی سخت افزاری نیز الگوریتم ۳ به علت آنکه متغیرهای کمتری در شرط‌ها شرکت می‌کنند پیاده سازی ساده تری خواهد داشت.

با توجه به این که این الگوریتم مانند الگوریتم ۱ با طراحی شده برای پیاده سازی سخت افزاری و نیز نرم افزاری مناسب می‌باشد و نیز استفاده از آن در پردازنده‌های پرسرعت مشکلات بکارگیری تقسیم کننده‌های با فرمان while را ندارد.

## ۷- نتایج پیاده سازی

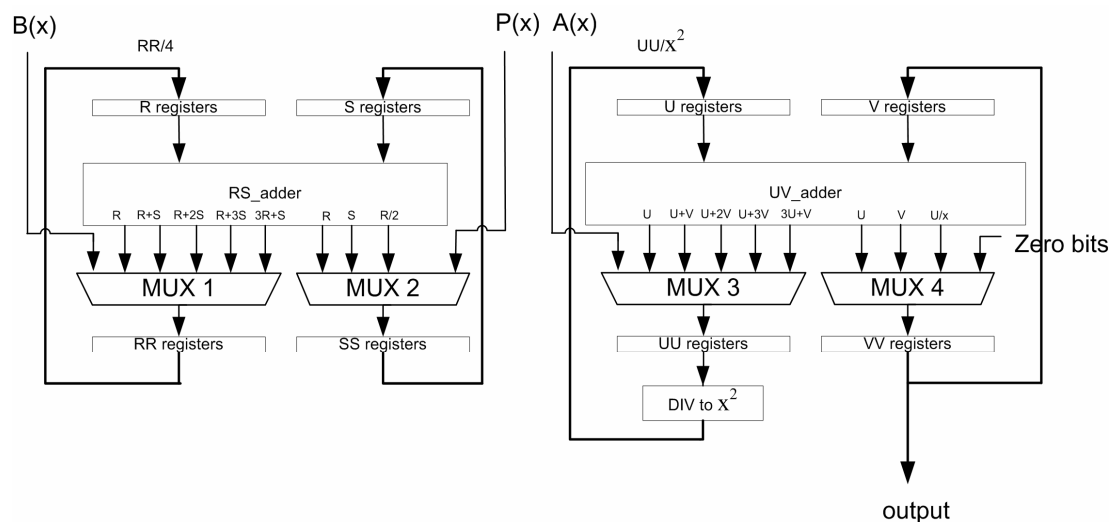
الگوریتم تقسیم کننده ارائه شده، بر روی FPGA (سری Xilinx Spartan3 XC3S1500-5) در محیط نرم‌افزاری ISE پیاده سازی شده است و نتایج شبیه سازی در جدول ۱ آورده شده است. همچنین در جدول ۱ نتایج پیاده سازی تقسیم کننده مبنای دو که در [۹] ارائه شده نیز آورده شده است.

```

input : A(x), B(x), P(x)
output : A(x)/B(x) mod P(x)
R = B(x); S = P(x); U = A(x); V = 0;
1. for i = 1 to m
2.   if(r0 = 0 and state = 0) then cnt = cnt + 1;
     elseif(state = 1 and cnt ≠ 0) then cnt = cnt - 1;
3-1.  (R, U) ← (R, U) when (r1r0 = 00)
       else (R + S, U + V) when (r1r0 + s1s0 = 00)
       else (R + 2S, U + 2V) when (r1r0 + s00 = 00)
       else (R + S + 2R, U + V + 2U)
           when (r1r0 + s1s0 + r00 = 00 and state = 0)
       else (R + S + 2S, U + V + 2V);
3-2.  (S, V) ← (R, U) when (r0 = 1 and state = 0)
       else (R/2, U/x) when (r1 = 0 and state = 0)
       else ((R+S)/2, (U+V)/x) when ((cnt = 0 and state = 1)
           and ((s1 = 0 and r1r0 = 11) or (s1 = 1 and r1r0 = 01)))
       else (S, V);
4.   if(state = 0 and cnt = 0) then
       if(r1 = 0) then state = 0;
       else state = 1;
     elseif(state = 1 and cnt ≠ 0) then state = 1;
     elseif(cnt = 0) then state = 0;
     else state = 1;
5.   U = U/x2 mod P(x); R = R/x2 mod P(x);
6. end loop;
7. return(V);

```

شکل ۳: الگوریتم ۳: تقسیم کننده بهبود یافته GCD مبنای ۴.



شکل ۴: شمای کلی پیاده سازی الگوریتم ۳

- Design, Codes, and Cryptography*, volume 25(2), pp. 207–216, February 2002.
- [6] Wei S.W., “VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in  $GF(2^m)$ ” *Processing IEEE Transactions on Circuits and Systems II: Analog and Digital Signal*, Volume 44, pp. 847–855, Oct. 1997.
- [7] Kaihara M.E., Takagi N., “A VLSI algorithm for modular multiplication/division” *Proceedings of the 16th IEEE Symposium on Computer Arithmetic, 2003*. pp. 220 – 227, June 2003.
- [8] Hasan M.A., Bhargava V.K., “Bit-serial systolic divider and multiplier for finite fields  $GF(q^m)$ ” *IEEE Transaction on Comput.*, Vol.41, pp. 972-980, Aug 1992.
- [9] Kim C.H., Hong C.P., “High-speed division architecture for  $GF(2^m)$ ” *Electronics Letters IEEE*, Volume 38, pp. 835 - 836 , July 2002.
- [10] Gura N. H. et al, “An end-to-end systems approach to elliptic curve cryptography” In CHES 2002 Workshop on Cryptographic Hardware and Embedded Systems, Lecture Note in Computer Science. Springer-Verlag, 2002.
- [11] Kenneth Weber, Kent State University, “The Accelerated Integer GCD Algorithm”, *ACM Transactions on Mathematical Software*, Vol. 21, pp. 111-122, March 1995.

## ۸- نتیجه گیری

در این مقاله، الگوریتمی جدید برای انجام عملیات تقسیم مدولار ارائه شده است. سرعت این الگوریتم تقسیم کننده مدولار ۲ برابر سرعت تقسیم کننده‌های مدولار مشابه پیشین می‌باشد. به این منظور ابتدا یک الگوریتم تقسیم کننده مبنای ۴ با کمک الگوریتم های پیشین ارائه شد و سپس با کمک توسعه پیش فرض‌های الگوریتم تقسیم کننده مبنای ۲ به مبنای ۴، الگوریتم ارائه شده تغییر یافت تا برای پیاده سازی با سخت افزار و نرم افزار بهینه شود. نتایج پیاده سازی سخت افزاری تقسیم کننده ارائه شده، نشان می دهد که سرعت در تقسیم کننده جدید دو برابر شده در حالی که سطح اشغالی فقط ۲۹ درصد افزایش یافته است.

## سپاسگزاری

نویسندگان این مقاله از مرکز تحقیقات مخابرات ایران که حمایت مالی این تحقیق را بر عهده داشتند کمال سپاسگزاری را دارند.

همچنین از سرکار خانم مهندس فاطمه فراتی و آقای مهندس بهامین ثبوتی که در بررسی نسخه اولیه این مقاله همکاری نمودند تشکر می‌گردد.

## ۹- مراجع

- [1] Reyhani-Masoleh A., “Efficient algorithms and architectures for field multiplication using Gaussian normal bases”, *IEEE Transactions on Computers*, Volume 55, pp. 34-47, Jan. 2006.
- [2] Reyhani-Masoleh A., Hasan M.A., “A new construction of Massey-Omura parallel multiplier over  $GF(2^m)$ ”, *IEEE Transactions on Computers*, Volume 51, pp. 511–520, May 2002.
- [3] Savas E., Naseer M., Gutub A.A-A, Koc C,K “Efficient unified Montgomery inversion with multibit shifting”, *IEE Proceeding of Comput. Digit. Tech.*, Vol.152, No.4, July 2005
- [4] Rodríguez-Henríquez F., Morales-Luna G., Saqib N. A., Cruz-Cortés N., “Parallel Itoh-Tsujii Multiplicative Inversion Algorithm for a Special Class of Trinomials” *Cryptology ePrint Archive*, Report 2006/035
- [5] Guajardo J., Paar C., "Itoh-Tsujii Inversion in Standard Basis and Its Application in Cryptography and Codes”, *Proceeding of the*