پیاده‌سازی الگوریتم رمزنگاری Serpent در فنّاوری اتوماتای سلولی کوانتومی

محمّدامین امیری[۱]

مژده مهدوی[۲]، ستّار میرزاکوچکی[۳]

چکیده

اتوماتای سلولی کوانتومی (QCA)، یک فنّاوری نوظهور در عرصه نانوفنّاوری می‌باشد. بدلیل ویژگی‌هایی نظیر مصرف توان اندک، ابعاد نانو و سرعت بالا، رمزنگاری یکی از کاربردهای جذاب این فنّاوری می‌باشد. پیاده‌سازی الگوریتم رمزنگاری Serpent در فنّاوری اتوماتای سلولی کوانتومی، در این مقاله مورد بحث قرار گرفته است. ماژولهای پایه ای Serpent با استفاده از سلول‌های QCA پیاده‌سازی شده‌اند. براحتی می‌توان مشاهده کرد که پیاده‌سازی الگوریتم‌های رمزنگاری در فنّاوری QCA در مقایسه با روشهای پیاده‌سازی CMOS، نتایج بهتری را بدنبال خواهد داشت.

کلید واژه

اتوماتای سلولی کوانتومی، رمزنگاری، Serpent ، پیاده سازی CMOS

۱. دانشجوی دکتری برق، دانشگاه علم و صنعت ایران amiri@ee.iust.ac.ir

۲. هیأت علمی دانشگاه آزاد اسلامی

۳. دانشیار دانشکده مهندسی برق، دانشگاه علم و صنعت ایران

تاریخ دریافت:۸۹/۴/۸ تاریخ پذیرش: ۸۹/۵/۳

مقدمه

صنایع میکروالکترونیک طی دهه‌های گذشته با کاهش ابعاد ترانزیستورها، مجتمع سازی، توان مصرفی و سرعت مدارات مجتمع را بهبود داده است. اما بنظر می‌رسد با وجود کاهش ابعاد ترانزیستورها، بعضی مشکلات نظیر توان مصرفی قابل صرفنظر نیستند. QCA که اولین بار توسط لنت و همکاران معرفی گردید [۱] ، یک فنّاوری نوظهور را در سطح نانو ارائه می‌دهد. استفاده از فنّاوری QCA برای پیاده‌سازی مدارات منطقی، یکی از روش‌هایی است که علاوه بر کاهش ابعاد مدارات منطقی و افزایش فرکانس کلاک این مدارات، توان مصرفی را کاهش می‌دهد [۱-۲]. سلول‌های QCA دارای نقطه‌های کوانتومی هستند که موقعیت الکترونها در آنها، سطح باینری صفر و یک را تعیین خواهد کرد. این مهمترین ویژگی مدارات QCA در مقایسه با مدارات معمول CMOS است که در آنها حالت‌های منطقی بوسیله سطوح ولتاژ مشخص می‌گردند.

الگوریتم رمزنگاری بلوکی Serpent یکی از کاندیداهای نهایی استاندارد رمزنگاری AES بود. این الگوریتم دارای ۳۲ راند با طول بلوک ۱۲۸ بیت و طول کلید ۲۵۶ بیت می‌باشد [۳]. بعنوان یکی از کاربردهای فنّاوری QCA ، ما الگوریتم رمزنگاری Serpent را پیاده‌سازی نمودیم. نتایج شبیه‌سازی این پیاده‌سازی از نرم‌افزار QCADesigner v2.0.3 بدست آمده‌اند. نرم‌افزار فوق در آزمایشگاه ATIPS دانشگاه Calgary کانادا تهیه شده است. این نرم‌افزار دارای موتورهای شبیه‌سازی متفاوتی می‌باشد. در این مقاله از موتور شبیه‌سازی Coherence Vector استفاده شده است زیرا ارزیابی دقیق و با جزئیات بیشتر از مدارات QCA انجام می‌دهد.

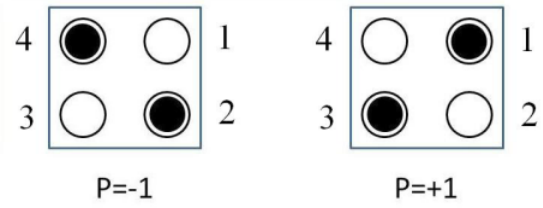
در بخش ۲، مروری کوتاه بر QCA ارائه شده است. در این بخش مروری بر الگوریتم رمزنگاری Serpent و بلوک‌های اصلی آن نیز خواهیم داشت. در بخش ۳ نتایج پیاده سازی و شبیه سازی بلوک‌های اصلی الگوریتم مورد بحث و بررسی قرار می‌گیرند. بخش ۴ شامل نتیجه‌گیری مقاله و بحث در ارتباط با نتایج بدست آمده خواهد بود.

موضوعات و روش‌ها

اتوماتای سلولی کوانتومی

در اتوماتای سلولی کوانتومی، یک سلول دارای چهار نقطه کوانتومی مانند شکل ۱ خواهد بود. نقطه های کوانتومی بصورت دایره‌های توخالی نمایش داده شده‌اند. هر سلول دارای دو الکترون است که بصورت دایره‌های توپر نشان داده شده‌اند.

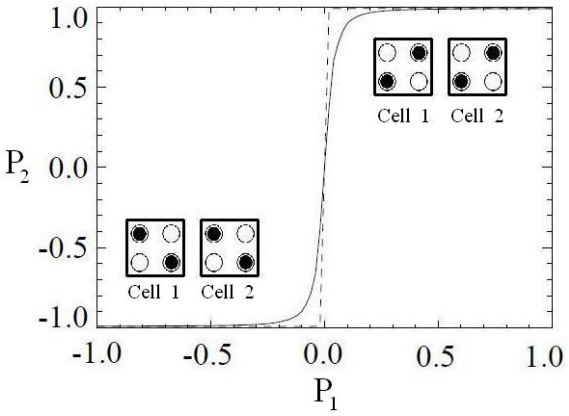
در یک سلول، الکترون‌ها با سازوکار تونل زنی مجاز به پرش بین نقطه‌های کوانتومی مجزا هستند ولی آنها مجاز به تونل زنی بین سلول‌ها نمی‌باشند. سد پتانسیل بین سلول‌ها بقدر کافی بزرگ فرض شده است تا بطور کامل از تونل زنی بین سلولی جلوگیری کند. با وجود اینکه دو الکترون دارای نیروی دافعه می‌باشند، آنها مجبور به اشغال نقطه‌های کوانتومی هستند.



شکل ۱: سلول QCA و حالت های پایه ای

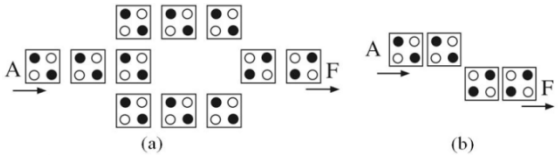
اگر آنها را بدون نیروی خارجی رها کنیم، آنها یکی از دو ترکیب پایه ای سلول را بخود می‌گیرند. واضح است که الکترون‌ها تمایل دارند که در نقطه‌های کوانتومی متمایز جای بگیرند و بخاطر نیروی کلمبی موجود، آنها در یک نقطه کوانتومی نخواهند ماند. با توجه به این پیش فرض‌ها، می‌توان نتیجه گرفت که حالت‌های پایه‌ای سامانه شامل دو ترکیب الکترونها در گوشه‌های مخالف سلول (شکل ۱) خواهد بود.

کوپلینگ بین دو سلول بوسیله اندرکنش کلمبی بین الکترون‌های سلولهای مختلف انجام می‌گیرد. شکل ۲ چگونگی تاثیر پذیری یک سلول از سلول همسایه اش را نشان می‌دهد [۴]. این شکل دو سلول را نشان می‌دهد که پلاریزاسیون سلول ۱ توسط پلاریزاسیون سلول همسایه اش تعیین می‌شود. فرض شده است که P2 در یک مقدار معینی فیکس شده است و روی سلول ۱ تاثیر گذاشته است و بنابراین پلاریزاسیون آن را تعیین می‌کند. نتیجه این است که کوپلینگ سلول به سلول در مدارات QCA شدیدا غیرخطی است. همانطور که می‌بینیم، حتی در صورت پلاریزه شدن جزئی سلول ۲، سلول ۱ بطور کامل پلاریزه شده است [۲،۴].



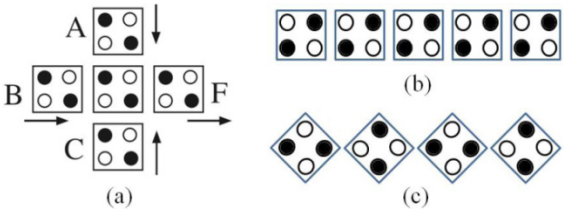
شکل ۲: کوپلینگ سلول های QCA

می‌توان از اندرکنش فیزیکی بین سلول‌ها جهت پیاده‌سازی توابع منطقی بولی پایه استفاده نمود. گیت‌های منطقی پایه در QCA شامل تابع منطقی اکثریت‌گیر و معکوس‌کننده می‌باشد که در شکل‌های ۳ و ۴ نشان داده شده‌اند. تابع منطقی اکثریت‌گیر تنها با ۵ سلول قابل پیاده‌سازی است [۵]. تابع AND منطقی را می‌توان با تابع منطقی اکثریت‌گیر و تنظیم یکی از ورودی‌هایش به منطق صفر پیاده‌سازی نمود. تابع OR منطقی را می‌توان با تابع منطقی اکثریت‌گیر و تنظیم یکی از ورودی‌هایش به منطق یک پیاده‌سازی نمود.



شکل ۳: (a) گیت معکوس کننده افزون شده، (b) گیت معکوس کننده

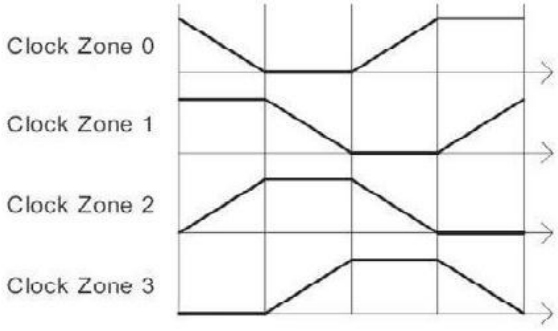
کلاکینگ در QCA سازوکاری جهت حرکت همزمان داده ها در مدار مهیا می‌کند. باید این مطلب را در نظر گرفت که کلاک همچنین جهت حرکت داده در مدارات QCA را کنترل می‌کند. کلاک همچنین توان مورد نیاز جهت عملکرد مدار را تامین می‌نماید.



شکل ۴: (a) گیت اکثریت گیر، (b) سیم باینری،(c) سیم ۴۵ درجه

بطور دقیقتر، از کلاک QCA جهت کنترل ارتفاع سد تونل زنی در سلول‌ها استفاده می‌شود. هنگامی که سطح کلاک پائین است، الکترون‌ها در موقعیت‌های مربوطه‌شان به تله افتاده و نمی‌توانند به نقطه‌های کوانتومی دیگر تونل‌زنی کنند و بنابراین مقدار منطقی سلول حفظ می‌شود. این عمل با ماکزیمم نگه داشتن ارتفاع سد تونل‌زنی محقق می‌شود. هنگامی که سطح کلاک بالا است، سلول به حالت پلاریزاسیون بی اثر خواهد رفت. این عمل با حداقل نگه داشتن ارتفاع سد تونل زنی محقق می‌شود. بین این دو حالت سلول‌ها یا در حال latch شدن و یا در حال relax شدن می‌باشند.

هر سلول در منطقه کلاکینگ خاص به یکی از چهار فاز کلاک QCA ممکن که در شکل ۵ نشان داده شده است، مرتبط شده است. هر سلول در ناحیه کلاک بطور همزمان با تغییرات سیگنال کلاک latch و unlatch می‌شود و به این ترتیب داده‌ها از طریق سلول‌ها منتقل می‌شوند [۶-۸].



شکل ۵: نواحی کلاک QCA

الگوریتم رمزنگاری Serpent

الگوریتم رمزنگاری Serpent یک شبکه SPN می‌باشد که روی ۴ کلمه ۳۲ بیتی عمل می‌کند و بنابراین دارای بلوک‌هایی به طول ۱۲۸ بیت خواهد بود[۳][۹-۱۴]. این الگوریتم با استفاده از ۳۳ زیر کلید و طی ۳۲ راند، داده ورودی کشف ۱۲۸ بیتی را به داده رمز ۱۲۸ بیتی رمزنگاری می‌نماید. الگوریتم رمز شامل یک جایگشت اولیه IP ، ۳۲ راند و یک جایگشت نهایی FP می‌باشد. هر راند شامل یک عملیات ترکیب با کلید، یک گذر از S-Box ها و یک تبدیل خطی است. در راند آخر، تبدیل خطی با یک ترکیب با کلید اضافی جایگزین شده است.

ترکیب با کلید: در هر راند، یک زیرکلید ۱۲۸ بیتی K_i با داده میانی B_i فعلی xor می‌شود.

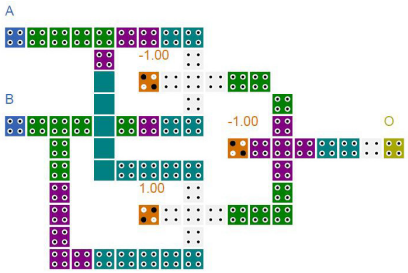
S-Box ها: الگوریتم رمزنگاری Serpent دارای ۸ S-Box با ابعاد ۴*۴ می‌باشد که در هر ۸ راند تکرار می‌شوند. هر ورودی ۱۲۸ بیتی به این S-Box ها به ۳۲ بلوک ۴ بیتی تقسیم شده و هر ۴ بیتی به یک S-Box اعمال می‌شود. خروجی این S-Box ها نهایتاً برای شکل دهی بلوک ۱۲۸ بیتی به هم ملحق می‌شوند.

تبدیل خطی: خروجی ۱۲۸ بیتی S-Box ها به ۴ بلوک ۳۲ بیتی تقسیم شده و این بلوک‌ها با عملیاتی نظیر شیفت، چرخش و xor با هم بصورت خطی ترکیب می‌شوند. یک راند کامل الگوریتم رمزنگاری Serpent را می‌توان بصورت زیر نمایش داد:

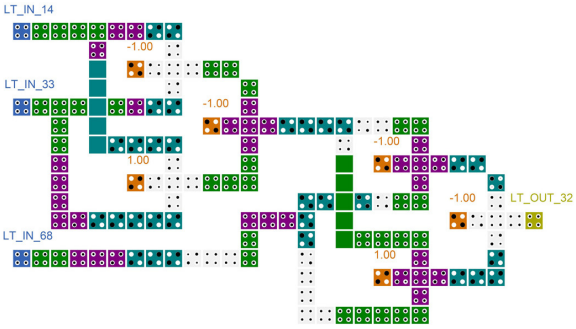
$$X_0, X_1, X_2, X_3 := S_i(B_i \oplus K_i) X_0, X_1, X_2, X_3 := S_i(B_i \oplus K_i)$$
$$X_0 := X_0 <<< 13 X_0 := X_0 <<< 13$$
$$X_2 := X_2 <<< 3 X_2 := X_2 <<< 3$$
$$X_1 := X_1 \oplus X_0 \oplus X_2 X_1 := X_1 \oplus X_0 \oplus X_2$$
$$X_3 := X_3 \oplus X_2 \oplus (X_0 <<< 3\ ) X_3 := X_3 \oplus X_2 \oplus (X_0 <<< 3\ )$$
$$X_1 := X_1 <<< 1 X_1 := X_1 <<< 1$$
$$X_3 := X_3 <<< 7 X_3 := X_3 <<< 7$$
$$X_0 := X_0 \oplus X_1 \oplus X_3 X_0 := X_0 \oplus X_1 \oplus X_3$$
$$X_2 := X_2 \oplus X_3 \oplus (X_1 <<< 7\ ) X_2 := X_2 \oplus X_3 \oplus (X_1 <<< 7\ )$$
$$X_0 := X_0 <<< 5 X_0 := X_0 <<< 5$$
$$X_2 := X_2 <<< 22 X_2 := X_2 <<< 22$$
$$B_{i+1} := X_0, X_1, X_2, X_3 B_{i+1} := X_0, X_1, X_2, X_3$$

نتایج

پیاده‌سازی جایگشت اولیه و جایگشت نهایی که فقط توابع باز ترتیب بیت هستند [۳]، با استفاده از سیم‌بندی ورودی‌ها به خروجی‌های مطلوب صورت می‌گیرد. ترکیب با کلید یا تابع xor با استفاده از سه تابع اکثریت‌گیر پیاده‌سازی شده است. شکل ۶ پیاده‌سازی تابع ترکیب با کلید را نشان می‌دهد. یکی از ورودی‌های این تابع، همان ورودی راند است و ورودی دیگر، بیت متناظر از کلید مربوط به همان راند می‌باشد. همانطور که در شکل نیز دیده می‌شود، برای پیاده‌سازی یک تابع xor دو ورودی در فنّاوری QCA نیاز به تعداد ۷۱ سلول QCA می‌باشد.



شکل ۶: پیاده‌سازی تابع ترکیب با کلید یا xor

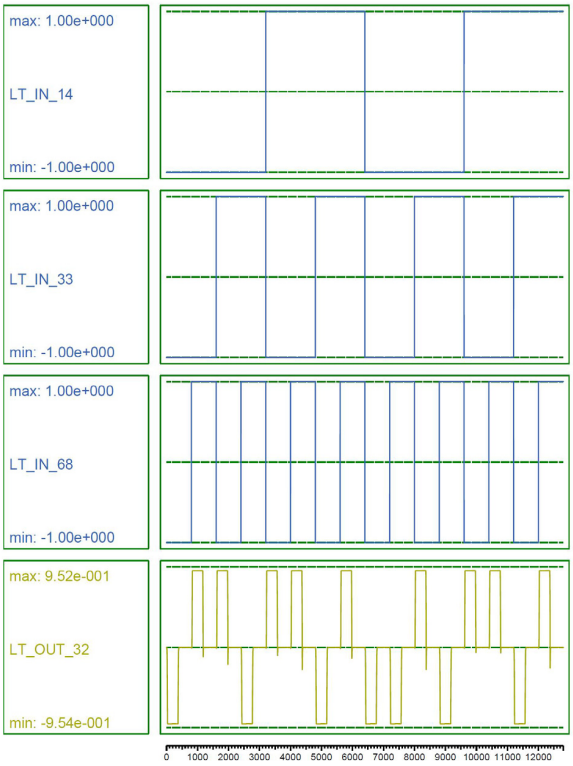


شکل ۷: پیاده‌سازی بیت ۳۳ام خروجی تبدیل خطی

برای پیاده‌سازی تابع تبدیل خطی، ابتدا خروجی توابع شیفت، چرخش و xor ، به xor ورودی‌ها که خروجی مرحله S-Box هستند، ساده شده است. بعنوان نمونه، خروجی‌های ۳۳ام و ۳۴ام تبدیل خطی پس از ساده شدن بصورت زیر می‌باشند:

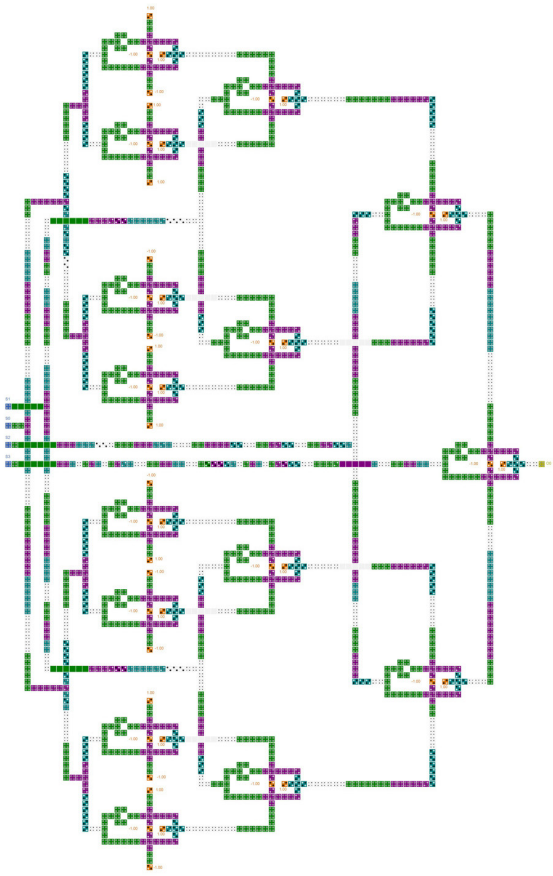
$$LTO(32) = LTI(14) \oplus LTI(33) \oplus LTI(68)$$
$$LTO(32) = LTI(14) \oplus LTI(33) \oplus LTI(68)$$
$$LTO(33) = LTI(15) \oplus LTI(34) \oplus LTI(69)$$
$$LTO(33) = LTI(15) \oplus LTI(34) \oplus LTI(69)$$

شکل ۷ پیاده‌سازی بیت خروجی ۳۳ام از تبدیل خطی را نمایش می‌دهد. همانطور که در این شکل دیده می‌شود، برای بدست آوردن این خروجی، ۲ گیت xor روی ۳ ورودی اعمال می‌شوند. یک شبیه‌سازی فراگیر برای این پیاده‌سازی انجام شده است. نتایج شبیه‌سازی خروجی ۳۳ام از تبدیل خطی در شکل ۸ دیده می‌شود.



شکل ۸: شبیه‌سازی بیت ۳۳ام خروجی تبدیل خطی

برای پیاده‌سازی تابع S-Box دو گزینه وجود دارد: روش مبتنی بر منطق (Logic-based) و روش مبتنی بر جدول جستجو (Look Up Table-based). در روش مبتنی بر منطق، خروجی S-Box تابعی منطقی از ورودی‌های آن در نظر گرفته می‌شود. این تابع منطقی بوسیله توابع اکثریت‌گیر پیاده‌سازی می‌شوند و خروجی طی عملیات S-Box ، محاسبه خواهد شد. ولی در روش مبتنی بر جدول جستجو، خروجی‌های S-Box در یک حافظه ذخیره می‌شوند و ورودی‌های S-Box به خطوط آدرس آن حافظه متصل می‌شوند. در این روش، با اعمال ورودی به خطوط آدرس، خروجی مطلوب روی خط داده خروجی ظاهر می‌گردد. در این مقاله، نتایج پیاده‌سازی و شبیه‌سازی روش مبتنی بر جدول جستجو ارائه شده‌اند. پیاده‌سازی چهارمین بیت خروجی S0 S-Box از الگوریتم رمزنگاری Serpent در شکل ۹ دیده می‌شود.



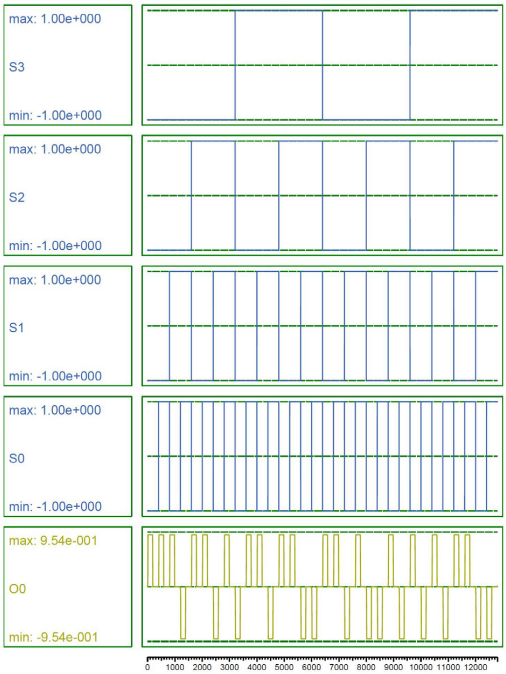
شکل ۹: پیاده‌سازی چهارمین بیت خروجی S0 S-Box

ورودی، خروجی و چهارمین بیت خروجی S0 S-Box در جدول ۱ دیده می‌شوند. مقادیر ورودی و خروجی به چارچوب هگزادسیمال نمایش داده شده‌اند.

جدول ۱: ورودی، خروجی و چهارمین بیت خروجی S0 S-Box

Input(hex)	0	1	2	3	4	5	6	7
Output(hex)	3	8	F	1	A	6	5	B
4th Output bit	1	0	1	1	0	0	1	1
Input(hex)	8	9	A	B	C	D	E	F
Output(hex)	E	D	4	2	7	0	9	C
4th Output bit	0	1	0	0	1	0	1	0

چهارمین بیت خروجی، در واقع بیت سمت راست خروجی است. یک شبیه‌سازی فراگیر برای این بیت خروجی انجام شده است. نتایج شبیه‌سازی چهارمین بیت خروجی S0 S-Box در شکل ۱۰ نمایش داده شده است. توالی خروجی "۱۰۱۱۰۰۱۱۰۱۰۰۱۰۱۰" که متناظر با تغییر ورودی از ۰ تا ۱۵ می‌باشد، در نتایج شبیه‌سازی و پس از ۱۰ کلاک تاخیر مدار دیده می‌شوند.



شکل ۱۰: شبیه‌سازی چهارمین بیت خروجی S0 S-Box

با فرض ابعاد 18 nm برای طول و عرض سلول‌های QCA و فاصله مرکز به مرکز 20 nm برای دو سلول مجاور [۱۵،۱۶]، نتایج پیاده‌سازی S-Box های الگوریتم در جدول ۲ نمایش داده شده است.

جدول ۲: نتایج پیاده‌سازی S-Box ها

	O3	O2	O1	O0	S0 S-Box
Complexity(Cells)	1200	1200	1200	1200	4800
Area(µm2)	2.626	2.626	2.626	2.626	10.504
Delay(Clocks)	10	10	10	10	10

بحث

در این مقاله، پیاده‌سازی الگوریتم رمزنگاری Serpent که یکی از کاندیداهای نهایی AES بود، مورد بحث و بررسی قرار گرفت. بلوک‌های اصلی این الگوریتم پیاده‌سازی و شبیه‌سازی شده‌اند. تعداد سلول‌ها و سطح مصرفی برای ماژول‌های ترکیب با کلید و S-Box ارائه شده‌اند. پیاده‌سازی‌های این مقاله در سطح سلول‌های QCA می‌باشد و بسته به نوع فنّاوری فلزی، مولکولی، نیمه‌هادی و ... که سلول‌های QCA در آن ساخته شوند، توان مصرفی و فرکانس کار متفاوت خواهد بود و در حال حاضر ابزاری برای محاسبه آنها در اختیار نداریم. از آنجایی که الکترون‌ها درون سلول حرکت می‌کنند، توان مصرفی بسیار ناچیز است. با توجه به ساختار خط لوله‌ای مدارات QCA، می‌توان از نتایج پیاده‌سازی دریافت که پس از چند کلاک تاخیر اولیه که مجموع تاخیر لایه‌های مختلف است، مثلاً ۱۰ کلاک برای لایه S-Box و ۲ کلاک برای لایه ترکیب با کلید، خروجی ۱۲۸ بیتی در هر کلاک معتبر خواهد بود و این نشان دهنده بازدهی بسیار زیاد (بازدهی در حد چندین ترابایت در ثانیه با فرض فرکانس کار در حد تراهرتز که یکی از ویژگی‌های این فنّاوری است) این گونه پیاده‌سازی می‌باشد.

مراجع

1.C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, ”Quantum Cellular Automata,” Nanotechnology, Vol. 4, No. 1, pp. 49-57, 1993.

2.P. D. Tougaw and C. S. Lent, “Dynamic Behavior of Quantum Cellular Automata,” Journal of Applied Physics, Vol. 80, No. 8, pp. 4722-4735, Oct. 1996.

3.R. Anderson, E. Biham, and L.Knudsen,”Serpent: A proposal for the Advanced Encryption Standard,” NIST AES Proposal, 1998.

4.P. D. Tougaw, C. S. Lent, and W. Porod, ”Bistable Saturation in Coupled Quantum-dot Cells,” Journal of Applied Physics, Vol. 74, No. 5, pp. 3558-3565, Sep. 1993.

5.P.D. Tougaw and C.S. Lent, ”Logical Devices Implemented Using

Quantum Cellular Automata," Journal of Applied Physics, Vol. 75(3), pp. 1818-1825, 1994.

K. Hennessy and C. S. Lent, "Clocking of Molecular Quantum-dot Cellular 6.Automata," Journal of Vacuum Science and Technology B, Vol. 19, No. 5, pp. 1752-1755, Sep. 2001.

7.C. S. Lent and Beth Isaksen, "Clocked Molecular Quantum-dot Cellular Automata," IEEE Transaction on Electron Devices, Vol. 50, No. 9, Sep. 2003.

8.M. A. Amiri, M. Mahdavi, S. Mirzakuchaki, "QCA Implementation of a Mux-Based FPGA CLB," Proc. of International Conf. On Nanoscience and Nanotechnology, Melbourne, Australia, pp. 141-144, Feb. 2008.

9.R. Anderson, E. Biham, and L.Knudsen, "Smart Card. Research and Applications," Springer, Berlin/Heidelberg, 2006.

J. L´azaro, A. Astarloa, J. Arias, U. Bidarte and C. Cuadrado, "Field 10.Programmable Logic and Application," Springer, Berlin/Heidelberg, 2004.

11.D. Ivancic, D. Runje, and M. Kovac, "Implementation of Serpent encryption algorithm on 24-bit DSPprocessor", Proc. of the 2nd International Symposium on Image and Signal Processing and Analysis, Pula, Croatia, pp. 411-416, June 2001.

12.B. Najafi, B. Sadeghian, M. Saheb Zamani, A. Valizadeh, "High speed implementation of Serpent algorithm," Proc. of the 16th International Conf. on Microelectronics, pp. 718- 721, dec. 2004.

13.A. J. Elbirt, and C. Paar, "An FPGA implementation and performance

evaluation of the Serpent block cipher", Proc. of the 2000 ACM/SIGDA 8th international symposium on Field programmable gate arrays, Monterey, California, United States, pp. 33-40, 2000.

14.I. Damaj, M. Itani, H. Diab, "Serpent Cryptography on Static and Dynamic Reconfigurable Hardware," Proc. of the IEEE International Conf. on Computer Systems and Applications, United States, pp. 680-684, 2006.

15.Heumpil Cho, Earl E. Swartzlander, Adder Designs and Analyses for Quantum-Dot Cellular Automata, IEEE Trans. on Nanotechnology, vol. 6, n. 3, May 2007, pp. 374–383.

16.Heumpil Cho, Earl E. Swartzlander, Adder and Multiplier Design in Quantum-Dot Cellular Automata, IEEE Trans. on Computers, vol. 58, n. 6, June 2009, pp. 721–727.