یک قدم روبه‌جلو در طراحی مدارات مقیاس نانو به‌وسیله هوش ماشین

جلال رستمی منفرد[۱]، دانشجوی دکتری؛ سید عبدالمجید موسوی[۲]، استادیار

۱- دانشکده فنی و مهندسی - دانشگاه لرستان - خرم‌آباد - ایران - rostami.ja@yahoo.com
۲- دانشکده فنی و مهندسی - دانشگاه لرستان - خرم‌آباد - ایران - mousavi.m@lu.ac.ir

چکیده: یکی از ایده‌های امیدبخش برای جایگزینی CMOS در مقیاس نانو ایده اتوماتای سلولی کوانتومی (QCA) است. بر این اساس، انواع مختلفی از مدارات الکترونیکی و منطقی طراحی و ارائه شده است که اساس طراحی آن‌ها گیت‌های اکثریت و اینورتر می‌باشد. در همین راستا، در کار پیش‌رو یک روش بهینه سازی مؤثر برای کاهش تعداد گیت‌های اکثریت و اینورتر در مدارات QCA چند ورودی-چند خروجی ارائه خواهد شد. روش پیشنهادی مبتنی‌بر برنامه‌نویسی ژنتیک کارتزین (Cartesian Genetic Programming) بوده که در آن مدار QCA به صورت دنباله‌ای از اعداد صحیح به‌عنوان ژنوتیپ (genotype) کد خواهد شد تا با اجرای برنامه، فنوتیپ (phenotype) بهینه با تعداد گیت‌ها، نوع گیت‌ها و اتصالات بهینه در خروجی حاصل شود. در ادامه و برای راستی‌آزمایی، روش پیشنهادی روی ۲۷ تابع منطقی پیاده‌سازی شده و نتایج گزارش خواهد شد. نتایج آزمون‌ها حاکی از آن است که روش پیشنهادی در مقایسه با روش‌های طراحی رقیب در یافتن جواب با کمترین تعداد گیت عملکرد بهتری دارد. بر این اساس، انتظار می‌رود با استفاده از این روش علاوه‌بر تعداد گیت کمتر، مدارهای QCA با سطح اشغالی و تأخیر کمتر طراحی شوند.

واژه‌های کلیدی: اتوماتای سلولی کوانتومی، برنامه‌نویسی ژنتیک کارتزین، بهینه‌سازی، مدارات کوانتومی، گیت اکثریت و اینورتر.

A Step Forward in the Design of Nano-Scale Circuits using Machine Intelligence

Jalal Rostami Monfared[1], PhD Student; Seyyed Abdolhamid Mousavi[2], Assistant Professor

1- Faculty of Electrical Engineering, Lorestan University, Khorramabad, Iran, Email: rostami.ja@yahoo.com
2- Faculty of Electrical Engineering, Lorestan University, Khorramabad, Iran, Email: mousavi.m@lu.ac.ir

Abstract: One of the promising ideas to improve over CMOS constrains in the nano-scales is Quantum Cellular Automata (QCA). So far, a variety of logic circuits are designed based on QCA where usually the majority and the inverter gates are the basic building blocks from which more complicated circuits are developed. In this paper, first we propose an approach to minimize the number of the majority and inverter gates in a given circuit with multiple inputs/outputs (MIMO). In our proposal, which is based on Cartesian Genetic Programming (CGP), a QCA circuit is coded as a series of integer numbers that constitutes a genotype for CGP. Applying CGP operators then, outputs the optimum phenotype including the number and the type of gates along with their interconnections. As for the verification of this approach, we apply it to 27 logic circuits and the results are reported, which show better solutions (in majority of cases) compared to the competing approaches. In addition to a fewer number of gates, our approach may provide a way to design QCA circuits with less power dissipation and/or less occupied areas.

Keywords: Quantum dot cellular automata, cartesian genetic programming, optimization, quantum circuits, majority and inverter gates.

تاریخ ارسال مقاله: ۱۳۹۷/۰۹/۰۷
تاریخ اصلاح مقاله: ۱۳۹۷/۱۱/۲۱
تاریخ پذیرش مقاله: ۱۳۹۸/۱۰/۰۴
نام نویسنده مسئول: عبدالمجید موسوی
نشانی نویسنده مسئول: ایران – خرم آباد - دانشگاه لرستان - دانشکده فنی و مهندسی.

۱- مقدمه

بعد از چند دهه، روند فشرده‌سازی مقیاس در CMOS با محدودیت‌هایی جدی مواجه شده به‌طوری‌که این محدودیت‌ها منجربه توسعه سریع تکنولوژی‌های جایگزین دیگر در مقیاس نانو گردیده است. یکی از روزنه‌های امیدبخش در حیطه نانوالکترونیک، ایده QCA است. مفهوم QCA اولین بار در سال ۱۹۹۳ [۱] مطرح و در سال‌های بعد توسعه پیدا کرد [۲]. از جمله ویژگی‌های مهم QCA چگالی بالا، مصرف توان پایین و تاخیر کم می‌باشد که طبیعتا می‌تواند یک جایگزین مناسب برای تکنولوژی CMOS در مدارات منطقی باشد. به‌عنوان مثال میزان مصرف توان و تاخیر یک گیت اینورتر پیاده‌سازی‌شده با QCA به‌ترتیب ۳۰/۲ meV و ۰/۲۵ Psec است [۳]، درحالی‌که همین کمیت‌ها برای یک گیت اینورتر CMOS به‌ترتیب در حدود چند نانووات و چند نانوثانیه می‌باشند (با توجه به شرایط بایاسینگ و غیره) [۴].

یکی از مشکلاتی که طراح هنگام طراحی مدارهای الکترونیکی و دیجیتالی براساس ماژول‌های[۱] پایه مثل گیت‌های[۲] منطقی با آن مواجه می‌شود کاهش تعداد ماژول‌ها و همچنین کاهش تأخیر ورودی به خروجی است. در این راستا، به‌طورکلی کارهای متنوعی در زمینه بهینه سازی مدارهای نانوالکترونیک انجام گرفته است. از جمله در [۵] با استفاده از نرم‌افزار متلب و الگوریتم ژنتیک[۳] تیوب‌های CNTFET با طول بهینه ارائه گردیده است. در [۶] نیز الگوریتم ژنتیک برای انتخاب بهترین ضخامت عایق گیت و طول در CNTFET استفاده شده است. این مقاله نیز از نرم‌افزار متلب برای شبیه سازی خود استفاده می‌کند. همچنین در [۷، ۸] ساختار جدیدی از ترانزیستورها در مقیاس نانو پیشنهاد گردیده که به‌ترتیب به‌نام ترانزیستور DM-DG و QSZ-MOSFET نام‌گذاری شده‌اند. در این کارها، از نرم‌افزار ATLAS برای شبیه‌سازی و بررسی عملکرد استفاده شده است.

در زمینه بهینه‌سازی مدارهای QCA به‌طور خاص، چون این مدارها براساس گیت‌های اکثریت[۴] و اینورتر[۵] ساخته می‌شوند خودبه‌خود تعداد و چیدمان این گیت‌ها از عوامل تاثیرگذار اصلی روی اندازه مدار و همچنین سرعت انتقال ورودی به خروجی خواهند بود [۹]. لذا مسأله بهینه‌سازی یک طرح مدار QCA در واقع با مسأله کاهش تعداد این گیت‌ها و نحوه چیدمان آن‌ها معادل خواهد بود.

در ارتباط با بهینه‌سازی مدارهای QCA کارهای متنوعی تاکنون صورت گرفته است [۱۹-۱۰]. در [۱۰] استراتژی جدیدی برای ساده‌سازی توابع منطقی و تبدیل عبارت‌های مجموع حاصل‌ضرب ارائه شده است که در آن ۱۳ تابع استاندارد برای نمایش همه توابع منطقی ۳ متغیره معرفی شده و عبارت‌های ساده‌شده مرتبط با آن‌ها استخراج گردیده است. در [۱۱] از ۲۰ تابع استاندارد برای ساده‌سازی توابع منطقی و ایجاد عبارت‌های معادل با گیت‌های اکثریت استفاده شده است. در این روش ابتدا یک تابع داده‌شده به شکل یکی از بیست تابع استاندارد بیان شده، سپس مدار مبتنی‌بر گیت اکثریت آن از یک جدول مرجع استخراج می‌شود. در [۱۲] یک الگوریتم جدید برای ایجاد عبارت‌های اکثریت بهینه ارائه شده است که طی آن یک جدول جستجو از عبارت‌های اکثریت ایجاد گردیده، سپس براساس آن از تکنیک ترکیب منطقی حداقل/حداکثر برای بهینه‌سازی مدار QCA استفاده می‌شود.

از طرف دیگر، استفاده از الگوریتم‌های تکاملی برای بهینه‌سازی مدارات QCA اولین بار در [۱۳] مطرح شد که در آن از ساختار درختی برنامه‌نویسی ژنتیک (GP) برای بهینه‌نمودن یک مدار تک خروجی بهره گرفته شد. در این روش توابع اکثریت و اینورترها گره‌های داخلی ساختار درختی می‌باشند. برهمین اساس، روش مشابه دیگری در [۱۴] برای مدارات با ۲ خروجی مطرح گردید. در [۱۵] علاوه‌بر توسعه کارهای قبلی، بهینه‌سازی همزمان کاهش تعداد گیت‌ها و تأخیر مدارات با تعداد دلخواه ورودی و خروجی پیشنهاد شد. روش کار به این صورت است که الگوریتم ژنتیک برای هر خروجی تکرار شده و بهترین کروموزوم‌ها[۶] و عبارات مشترک دو کروموزوم شناسایی می‌شوند. سپس با حذف قسمت‌های مشترک، کروموزوم‌های حاصل برای یافتن بهترین نتیجه با یکدیگر ترکیب می‌شوند.

در [۱۶] یک تابع برازندگی[۷] جدید با وزن‌های متفاوتی برای گیت‌های اکثریت و اینورتر در نظر گرفته‌شده و کروموزومی با کمترین تعداد سلول به‌عنوان نتیجه انتخاب می‌شود. از طرف دیگر، [۱۷] از ۲ تابع برازندگی متفاوت برای الگوریتم تکاملی برنامه‌نویسی ژنتیک (Genetic Programming) بهره می‌گیرد. در گام اول کروموزومی معادل با تابع منطقی ایجاد می‌شود و سپس با قیاس خروجی الگوریتم با خروجی مطلوب، مقدار تابع برازندگی موردنظر انتخاب می‌گردد. در گام دوم برای هر گیت اکثریت یک جدول کارنو ایجاد شده و با قیاس بین این جدول‌ها از تابع برازندگی دوم استفاده می‌کند. در این روش برای هر خروجی کروموزوم‌هایی با بهترین برازندگی انتخاب شده و نتایج با هم ترکیب می‌شوند. در نهایت نتیجه‌ای که دارای بیشترین گیت مشترک است به‌عنوان بهترین گزینه معرفی می‌گردد.

در [۱۸] براساس الگوریتم ژنتیک چند-هدفه[۸] تکنیکی برای ترکیب مطلوب مدار QCA، به‌خصوص مدار QCA چند-خروجی پیشنهاد شده است درحالی‌که از یک تابع برازندگی نسبی برای یافتن راه‌حل بهینه کلی استفاده می‌کند. در این روش، بهینه سازی برحسب مساحت با مینیمم‌سازی تعداد گیت و کاهش تاخیر مدار در طول فرایند ترکیب مدنظر بوده است. در مقاله [۱۹] نیز بهینه‌سازی توابع منطقی ۳ متغیره با استفاده از الگوریتم کلونی مورچه‌ها (Ant Colony) انجام شده است که در آن بهینه‌سازی مدارات QCA از لحاظ کاهش تعداد گیت‌ها و سطوح مورد نیاز انجام می‌گیرد.

همان‌طور که در کارهای فوق شرح داده شد در روش‌های فعلی بهینه‌سازی مدارات QCA، یا تعداد ورودی‌ها و خروجی‌ها محدود است، یا از روش دستی استفاده می شود و یا الگوریتم‌های تکاملی با ساختار کروموزوم درختی استفاده گردیده است. در کار پیش‌رو، به‌منظور

برطرف‌کردن محدودیت تعداد ورودی و خروجی مدار و نیز بهبود معیارهای بهینه سازی، روش برنامه‌نویسی ژنتیک کارتزین[۹] (CGP) در فرایند تکامل مدارات QCA استفاده خواهد شد. به‌طور خاص، معیاری که بهبود آن دنبال خواهد شد عبارت است از کاهش تعداد گیت (که می‌تواند به کاهش تأخیر و کاهش تعداد سیکل‌های کلاک هم منجر گردد). پس از عملیاتی‌کردن روش خود نتایج اجرای آن روی توابع موجود در کارهای مشابه، گزارش خواهد گردید. همان‌طوری که در بخش نتایج شبیه سازی مندرج است در اکثر موارد بهبود قابل‌توجهی در مدارات حاصل از اجرای روش پیشنهادی نسبت به روش‌های رقیب قابل مشاهده است.

این مقاله به‌صورت زیر تنظیم گردیده است. در بخش ۲، توضیح مختصری از تکنولوژی QCA ارائه شده و در بخش ۳ الگوریتم CGP شرح داده می‌شود. سپس در بخش ۴، بهینه‌سازی توابع منطقی QCA چند ورودی - چند خروجی با روش پیشنهادی ارائه خواهد گردید. بخش ۵ مقایسه نتایج با کارهای مشابه را شامل شده و نهایتا خلاصه‌ای از نتایج کار پیش‌رو به همراه پیشنهاداتی جهت توسعه و ادامه کار در بخش ۶ ارائه می‌شود.

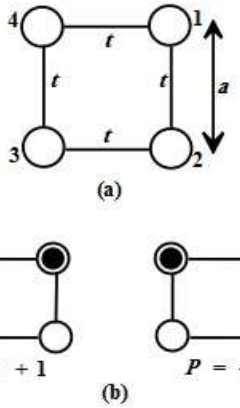
۲- اساس ایده QCA

عناصر اصلی در ایده QCA سلول‌های QCA هستند. مطابق شکل ۱- الف هر سلول شامل ۴ نقطه کوانتومی در ۴ گوشه سلول مربعی و ۲ الکترون (چگالی الکترون‌ها) متحرک می‌باشد که دو الکترون با تونل‌زنی بین نقاط کوانتومی جابه‌جا می‌گردند [۲۰]. با توجه به نحوه استقرار الکترون‌ها در داخل سلول، سه حالت برای آن پدید می‌آید. حالت null (قطبیت صفر) حالتی است که ارتفاع سد پتانسیل نقاط پایین بوده و الکترون‌ها جایگاه معینی ندارند. دو حالت دیگر با افزایش ارتفاع سد پتانسیل رخ می‌دهند و به‌عنوان قطبیت p=-1 و p=+1 شناخته می‌شوند (شکل ۱- ب). در این حالت‌ها به‌سبب فعل و انفعالات کولمبی، الکترون‌ها دو موقعیت قطری مانند شکل ۱- ب را اشغال می‌کنند. با کدکردن قطبیت‌های p=-1 و p=+1 به منطق باینری ۰ و ۱، سلول‌های QCA می‌توانند در پیاده‌سازی توابع باینری به‌کارگرفته شوند به‌طوری‌که ورودی‌ها در نتیجه انتشار قطبیت بین سلولی (به‌سبب فعل و انفعالات کولمبی) به خروجی‌ها منتقل می‌شوند.

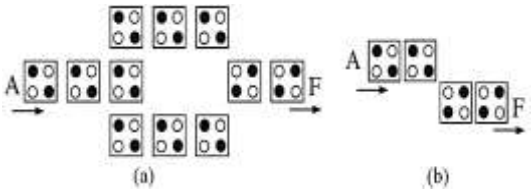
اساسی‌ترین المان‌های مدارات QCA گیت‌های اکثریت و اینورتر می‌باشند که مانند گیت‌های پایه AND و OR در مدارات منطقی، در ساخت مدارات QCA استفاده می‌گردند [۲۱، ۲۲]. از طرف دیگر، گیت‌های اکثریت و اینورتر با چیدمان مناسب سلول‌های QCA در کنار هم به‌دست می‌آیند. برای نمونه شکل ۲ انواع مختلف و متداول پیاده سازی اینورتر و شکل ۳ انواعی از گیت اکثریت را نشان می‌دهند [۲۱، ۲۳]. تابع گیت اکثریت (۳-ورودی) با رابطه (۱) توصیف می‌گردد.

$$f = M(a,b,c) = ab + ac + bc \quad (۱)$$

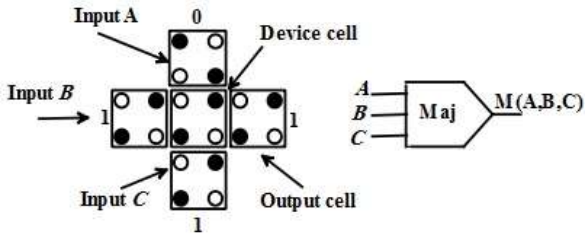
که در آن a, b, c ورودی‌ها و f خروجی تابع می‌باشند. به‌این‌ترتیب، گیت‌های منطقی AND و OR را می‌توان با ثابت نگه‌داشتن قطبیت یکی از ترمینال‌های ورودی گیت اکثریت به‌ترتیب با p=-1 و p=+1 ایجاد کرد (شکل ۴) [۲۲].



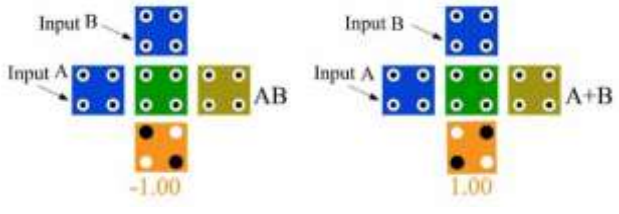
شکل ۱: الف) یک سلول QCA با ۴ نقطه کوانتومی، ب) نمایش قطبیت p=+1 و p=-1 در سلول [۲]



شکل ۲: دو مدار QCA برای پیاده‌سازی گیت اینورتر [۲۱]



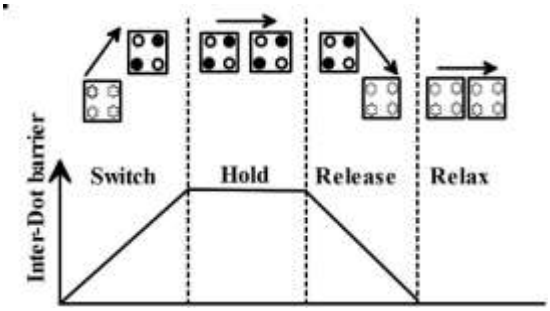
شکل ۳: گیت اکثریت ۳- ورودی [۲۲]



شکل ۴: اجرای گیت‌های AND و OR با استفاده از گیت اکثریت [۲۲]

یکی از ویژگی‌های مشترک مدارات QCA نیاز به پالس‌زنی ساعت[۱۰] مناسب برای کنترل گردش اطلاعات و انتقال صحیح سیگنال ورودی به خروجی می‌باشد. علاوه‌بر این سیگنال پالس‌زنی ساعت توان ضروری برای درایو مدار و محاسبات متوالی را فراهم می‌کند [۲، ۲۴]. سیگنال پالس‌زنی ساعت یک ولتاژ زمان‌بندی شده است که فعالیت سلول را مهیا کرده و می‌تواند آن را به آرامی از یک حالت فعال به یک حالت

null سوئیچ کند [۲۵]. در واقع هر سیگنال پالس ساعت در QCA شامل چهار فاز است: Switch, Hold, Release, Relax که در شکل ۵ نشان داده شده‌اند. همان‌طوری که قبلا بیان شد، مدارات QCA با استفاده از گیت‌های پایه‌ای اکثریت و اینورتر ایجاد می‌گردند. لذا برای داشتن طراحی های بهینه، تعداد و چیدمان این گیت ها عامل تعیین‌کننده خواهد بود. بدین سبب، در بخش بعدی روش پیشنهادی (که از الگوی CGP بهره می‌گیرد) برای بهینه‌سازی مدارات QCA شرح داده می‌شود.



شکل ۵: چهار فاز مختلف سیگنال پالس ساعت [۲]

۳- برنامه‌نویسی ژنتیک کارتزین (CGP)

الگوریتم CGP اولین بار در [۲۶] ارائه شد و سپس در سال ۲۰۰۰ به‌عنوان شکلی از برنامه‌نویسی ژنتیک توسعه داده شد [۲۷]. در CGP راه‌حل‌های ممکن به‌وسیله رشته‌ای از اعداد صحیح کد می‌شوند که آن را ژنوتیپ[11] می‌نامند و این راه‌حل ها به‌صورت یک گراف جهت‌دار دوبعدی از گره‌هایی که نماینده عناصر پایه هستند ساخته می شوند. به‌طورخاص در مدارات منطقی هر گره در گراف یک تابع یا گیت پایه را نشان می‌دهد. به‌علاوه، هر گیت با یک شناسه خاص در یک جدول جستجو لیست شده و در ژنوتیپ نیز به‌وسیله تعدادی ژن (آلل) کد می‌گردد. برای هر گره یکی از ژن‌ها آدرس تابع گره در جدول جستجو را نشان می‌دهد که به‌عنوان ژن تابع (function-gene) شناخته می‌شود. بقیه ژن‌های گره، آدرس ترمینال‌های ورودی گره را مشخص می‌کنند که به آن‌ها ژن اتصال (connection gene) می‌گویند. در این ساختار ماکزیمم تعداد ورودی‌های یک گره را arity آن گره می‌نامند.

گره‌ها ورودی‌شان را با یک روش رانش روبه جلو (feed-forward) از خروجی گره‌های ستون‌های قبلی یا از ورودی‌های اصلی برنامه دریافت می‌کنند. این مسأله که ورودی‌های گیت فعلی از خروجی‌های چند سطح قبل آن قابل دریافت است، به‌وسیله پارامتر سطح برگشتی (level-back) کنترل می‌شود. شکل ۶ شماتیک ژنوتیپ و فنوتیپ[12] CGP را نشان می‌دهد. فرم کلی یک ژنوتیپ را می‌توان به‌صورت زیر نمایش داد [۲۸]:

$$\mathbf{F_0C_{0,0}\ldots.C_{0,a}F_1C_{1,0}\ldots.C_{1,a}\ldots.F_{(c+1)r-1}C_{(c+1)r-1,0}\ldots.C_{(c+1)r-1,\,a}O_0O_1\ldots.O_m} \quad (۲)$$

که در آن a مقدار arity هر گره، C_{ij} ژن‌های اتصال، F_i ژن‌های تابع، O_k ژن‌های خروجی، n تعداد ورودی‌ها و m تعداد خروجی‌های برنامه است.

مطابق شکل ۶، هر کروموزوم (ژنوتیپ) در CGP دارای سه نوع ژن است: ژن تابع، ژن اتصال و ژن‌های خروجی. ژن‌های خروجی در واقع مکانی که خروجی برنامه از آنجا گرفته می‌شود را آدرس‌دهی می‌کنند.

با اعمال فرایندهای جهش، انتخاب و ... بر روی کروموزوم ها و سپس دی‌کد کردن آن‌ها فنوتیپ‌ها (راه‌حل‌ها) به‌دست می‌آیند. وقتی‌که یک کروموزوم دی‌کد می‌شود بعضی از گره‌های آن ممکن است غیرفعال باشند و این زمانی است که خروجی آن گره‌ها در داده خروجی برنامه تأثیری نداشته باشند. لذا در CGP طول کروموزوم ثابت اما طول فنوتیپ (تعداد گره‌ها یا توابع) متغیر می‌باشد. لذا یک فنوتیپ ممکن است تعداد گره فعال کمتری نسبت به تعداد گره‌های کدشده اولیه دارا باشد.

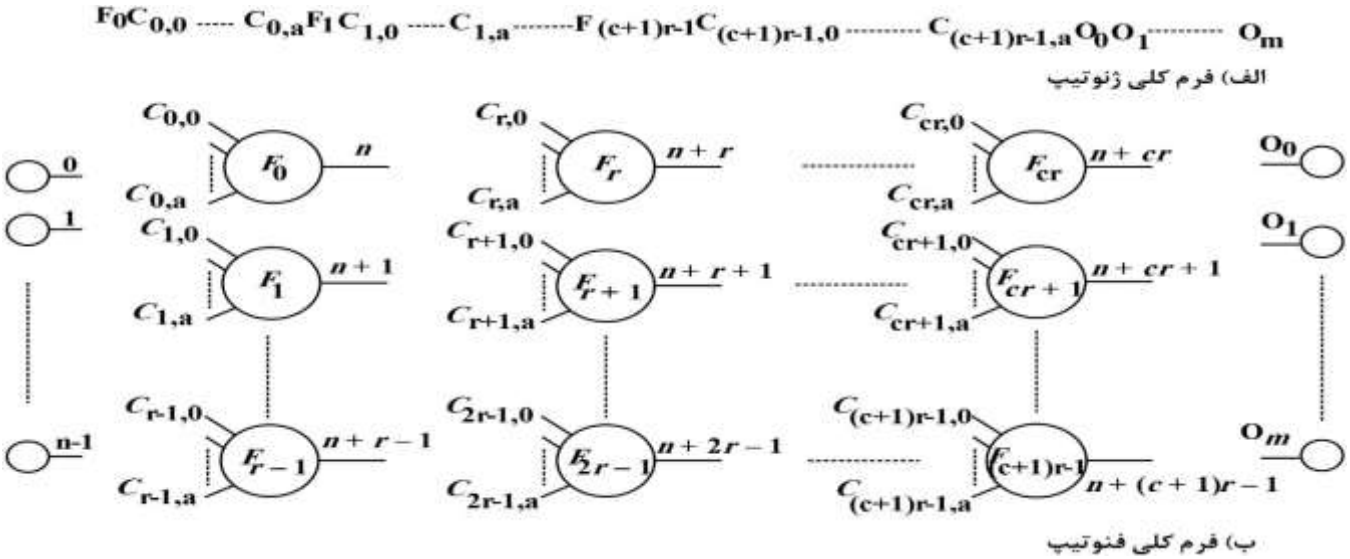
۴- روش پیشنهادی

۴-۱- کلیات

همان‌طوری که عنوان گردید هدف ما بهینه‌سازی مدارات مبتنی‌بر تکنولوژی QCA با بهره‌گیری از روش CGP است به این معنی که کمترین تعداد ممکن از گیت‌های اینورتر و اکثریت در مدار استفاده شود. به همین منظور در این مقاله از زبان برنامه نویسی C برای پیاده سازی الگوریتم CGP استفاده می شود. ورودی برنامه یک فایل داده‌ای است که حاوی اطلاعاتی درباره تعداد ورودی‌ها و خروجی‌ها، جدول صحت، همچنین دو ورودی ثابت با مقادیر ۰ و ۱ برای پیاده‌نمودن توابع منطقی AND و OR با استفاده از گیت اکثریت است. افزودن دو ورودی ثابت ۰ و ۱ قدرت جستجو و انتخاب الگوریتم CGP را افزایش داده و باعث می‌شود برنامه به‌طور مستقیم از گیت‌های AND و OR ایجاد شده به‌وسیله ترکیب گیت اکثریت و ورودی‌های ساختگی ۰ و ۱ استفاده نماید. بعد از اجرای برنامه، ژنوتیپ و در نتیجه فنوتیپ نهایی و بهینه‌شده با توجه به پارامترهای برنامه ایجاد می‌گردد. جدول ۱ پارامترهای استفاده‌شده در روش پیشنهادی را نشان می‌دهد.

جدول ۱: پارامترها و فرایندهای استفاده‌شده در الگوریتم CGP

Evolutionary Strategy	(۱+۴) -ES
Mutation	Point Mutation
Mutation rate	۰/۰۵
Fitness function	Supervised learning
Selection scheme	Select fittest
Reproduction scheme	Mutation random parent



شکل ۶: فرم کلی ژنوتیپ و فنوتیپ؛ الف) ژنوتیپ، ب) فنوتیپ [۲۸]

ا ستراتژی تکاملی ا ستفاده شده به‌و سیله CGP عموماً انتخاب یک والد از هر نسل و استفاده از آن برای ایجاد ۴ فرزند از طریق جهش است؛ یعنی نسل بعدی والد انتخاب شده و ۴ فرزند تولیدشده را شامل می‌شود. بنابراین اندازه جمعیت برابر ۵ خواهد شد. این استراتژی را (μ+λ)-ES یا ES- (۴+۱) می‌نامند [۲۹]. شکل ۷ استراتژی تکاملی ES- (۴+۱) را نشان می‌دهد.

```
The (1+4) evolutionary strategy
1: for all i such that 0≤ i <5 do
2:     Randomly generate individual i
3: end for
4: Select the fittest individual, which is promoted as the parent
5: while a solution is not found or the generation limit is not reached   do
6:     for all i such that 0≤ i <4 do
7:         Mutate the parent to generate offspring i
8:     end for
9:     Generate the fittest individual using the following rules:
10:    if an offspring genotype has a better or equal fitness than the parent then
11:        Offspring genotype is chosen as fittest
12:    else
13:        The parent chromosome remains the fittest
14:    end if
15: end while
```

شکل ۷: استراتژی تکاملی پیشنهادی

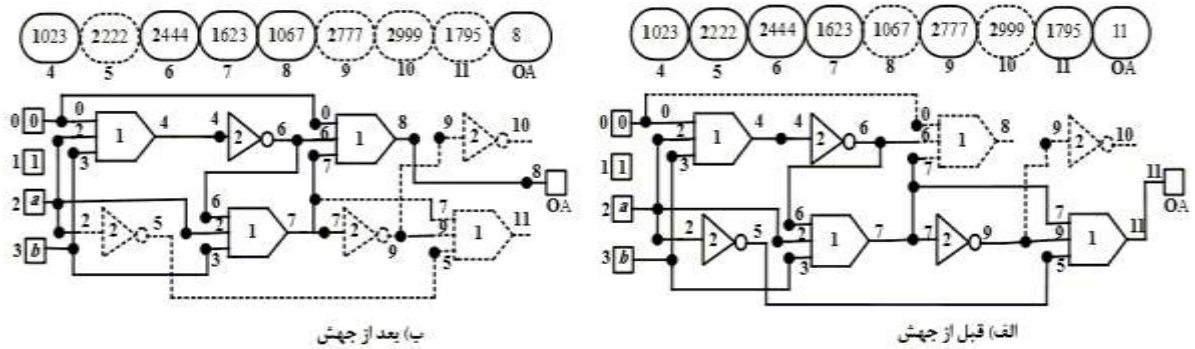
۴-۲- تابع برازندگی و جهش

جهش[۱۳] یکی از ابزارهای روش‌های تکاملی جهت تحول در جمعیت است و یک روش متداول برای این امر جهش نقطه‌ای (point-mutation) است که درصدی از ژن‌های ژنوتیپ به‌صورت تصادفی انتخاب و به مقادیر تصادفی دیگر تغییر می‌یابند. یک مثال از اعمال جهش نقطه‌ای برای بهینه‌سازی تابع XOR در شکل ۸ نشان داده شده است. مطابق شکل، عمل جهش در ژن خروجی برنامه رخ داده که باعث تغییر گره خروجی از ۱۱ به ۸ گردیده است. همچنین با این عمل گره‌های ۱۱، ۹ و ۵ غیرفعال و گره ۸ فعال می‌گردد که منجربه کاهش تعداد گیت‌های فعال و درنتیجه به‌دست‌آمدن جواب بدون خطا و دراین‌حال بهینه از نظر تعداد گیت می‌گردد.

در پیاده‌سازی الگوریتم پیشنهادی از تابع برازش (۳) استفاده شده است.

$$fitness = \sum_{i}^{n} \sum \left| y_{O,i} - y_{d,i} \right| \qquad (۳)$$

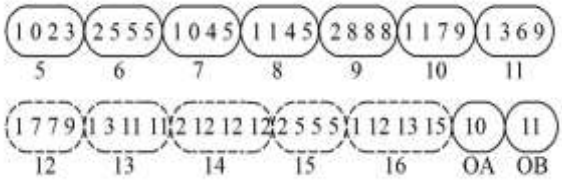
در رابطه (۳)، n اندازه جدول صحت اولیه، y_o خروجی واقعی برنامه و y_d خروجی مطلوب فایل داده ورودی است.



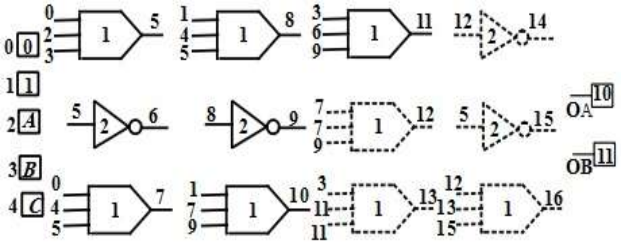
شکل ۸: مثالی از عملکرد جهش نقطه‌ای؛ الف) قبل از جهش، ب) بعد از اعمال جهش به یک ژنوتیپ به همراه فنوتیپ مرتبط با آنها (گیت‌های غیرفعال بصورت خط‌چین مشخص شده‌اند)

۴-۳- عملکرد الگوریتم

برای بررسی بیشتر روش بهینه سازی خود فرض کنید قرار است توابع $f1=\Sigma(0,2,4,7)$ و $f2=\Sigma(0,2,3,4)$ با استفاده از گیت‌های اکثریت و اینورتر که در جدول جستجو به‌ترتیب با ۱ و ۲ شماره‌گذاری شده‌اند با کمترین تعداد گیت طراحی شوند. با تعریف پارامترهای برنامه به‌صورت تعداد گره‌ها ۱۲، arity=۳، سطح برگشتی برابر تعداد گره‌ها، تعداد خروجی ۲ و تعداد ورودی ۵، ژنوتیپ نهایی حاصل از اجرای الگوریتم پس از ۲۰۰۰ تکرار برای توابع $f1$ و $f2$ به‌صورت شکل ۹ خواهد بود. همان‌طوری که قبلاً در بخش ۴-۱ ذکر گردید مقادیر ثابت ۰ و ۱ برای ایجاد مستقیم گیت‌های AND و OR با استفاده از گیت اکثریت به دیگر ورودی‌ها اضافه شده‌اند. در نهایت فنوتیپ یا جواب نهایی برای پیاده‌سازی دو تابع $f1$ و $f2$ در شکل ۱۰ نشان داده شده است.



شکل ۹: ژنوتیپ نهایی در نتیجه اجرای الگوریتم



شکل ۱۰: فنوتیپ نهایی ناشی از ژنوتیپ شکل ۹

با توجه به شکل ۱۰، از حداکثر ۱۲ گیت مجاز در ورودی برنامه تنها ۸ گیت در راه‌حل نهایی استفاده شده و گیت‌های دیگر غیرفعال می‌باشند. در بخش بعدی توابع بیشتری با استفاده از روش پیشنهادی طراحی خواهد گردید.

۵- نتایج شبیه‌سازی

در این بخش نتایج شبیه‌سازی مدارات QCA چند ورودی - چند خروجی با استفاده از روش پیشنهادی ارائه شده و با کارهای مشابه قبلی مقایسه می شوند. برای تمامی این توابع اندازه جمعیت اولیه برابر ۵ و تعداد تکرار بین مقادیر ۱۰۰۰ تا ۱۰۰۰۰ انتخاب می‌شوند. در جدول ۲، اجرای ۱۵ تابع ۳- ورودی/ ۱- خروجی استاندارد نشان داده شده است. با توجه به جدول ۲ روش پیشنهادی ما تقریبا نسبت به توابع [۱۰] به اندازه ۲۶/۹٪، نسبت به توابع [۱۹] به اندازه ۲۲/۸۵٪ و نسبت به توابع [۱۲] به اندازه ۲۰/۰۰٪ کاهش در تعداد گیت‌های اینورتر و اکثریت استفاده‌شده نشان می‌دهد. برای نمونه، شکل‌های ۱۱- الف و ۱۱- ب به‌ترتیب مدارهای به‌دست‌آمده از روش [۱۰] و روش پیشنهادی برای تابع $\sum m(0,3,6,7)$ در جدول ۲ را نشان می‌دهند. همان‌طور که در شکل ۱۱ قابل مشاهده است، برای این تابع خاص مدار به‌دست‌آمده از روش ما ۴ گیت اکثریت و ۱ گیت اینورتر داشته در حالی‌که مدار [۱۰] حاوی ۵ گیت اکثریت و ۳ گیت اینورتر می‌باشد.

برای ارزیابی بیشتر، روش پیشنهادی برروی توابع چند ورودی - چند خروجی اعمال شده و نتایج به‌ترتیب در جداول ۳، ۴، ۵ و ۶ آورده شده‌اند. جدول ۳ مقایسه بین توابع ۳- ورودی/۲- خروجی را نشان می‌دهد. همان‌طوری که در جدول ۳ نمایان است، روش ما نسبت به توابع [۱۷] به اندازه ۳۰٪، نسبت به توابع [۱۴] به اندازه ۲۲/۲٪ و نسبت به توابع [۱۸] به اندازه ۳۰٪ کاهش در تعداد گیت‌های اینورتر و اکثریت استفاده‌شده نشان می‌دهد. در جدول ۴ نتایج شبیه‌سازی برای توابع ۳- ورودی/۴- خروجی ارائه شده است. مطابق این جدول روش پیشنهادی ما نسبت به [۱۸] و [۱۵] به‌ترتیب به اندازه ۶۶٪ و ۲۶٪ کاهش در تعداد گیت اینورتر و اکثریت نتیجه می‌دهد. درنهایت جداول ۵ و ۶ به‌ترتیب نتایج شبیه سازی توابع ۴- ورودی/۲- خروجی و توابع ۴- ورودی/۴- خروجی را نشان می‌دهند. همان‌طور که در این دو جدول هم نمایان است، عموما روش ما عملکرد بهتری در کاهش تعداد عناصر پایه مدارات QCA داشته و سبب کاهش پیچیدگی (و تأخیر) در طرح های QCA می‌شود.

۶- خلاصه و نتیجه‌گیری

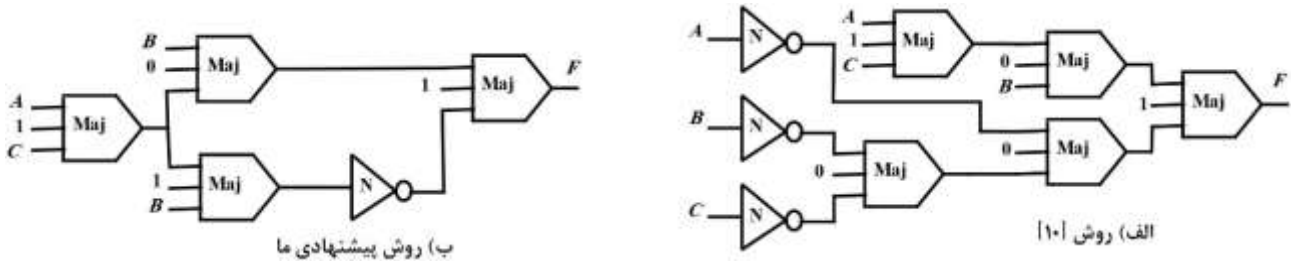
هدف این مقاله ارائه یک روش جدید بهینه‌سازی مبتنی‌بر برنامه‌نویسی ژنتیک برای کاهش تعداد عناصر پایه در طرح‌های QCA بود. روش پیشنهادی دارای مزیت‌هایی نسبت به روش‌های دیگر است ازجمله اینکه از ساختار ماتریسی به جای درختی در بهینه‌نمودن توابع استفاده کرده و به‌آسانی قابل توسعه و اجرا برروی مدارات با چندین خروجی است. در این روش، ابتدا جدول صحت توابع به‌عنوان یک فایل ورودی برای برنامه تعریف می‌شود. سپس، با اجرای الگوریتم، در خروجی تعداد و نوع گیت‌ها و همچنین نحوه اتصال بین آن‌ها نتیجه می‌شود. نتایج شبیه سازی مدارات با ۳ و ۴ ورودی و ۱، ۲ و ۴ خروجی نشان می‌دهد که عموما روش ما در مقایسه با روش‌های دیگر تعداد گیت اینورتر و اکثریت کمتری دارد. این کاهش به‌ویژه در مدارات QCA می‌تواند منجر به کاهش سطح اشغالی مدارات نهایی شود که پارامتری تعیین‌کننده برای تکنولوژی‌های ساخت در مقیاس نانومتر مانند QCA می‌باشد. در کارهای آتی می‌توان علاوه‌بر کاهش پیچیدگی مدارات QCA، با تمرکز بر تعداد سطوح مدارات ایجاد شده توسط کروموزوم‌ها، کاهش تأخیر کل از ورودی‌ها به خروجی‌ها بررسی شود.

جدول ۲: نتایج شبیه‌سازی توابع ۳- ورودی/۱- خروجی

تابع	[۱۰]		[۱۹]		[۱۲]		روش پیشنهادی	
$\sum m(7)$	M(M(A,B,0),C,0)	2Maj	M(M(A,B',C),B,0)	2Maj 2Not	-	-	M(M(A,B,0),C,0)	2Maj
$\sum m(6,7)$	M(A,B,0)	1Maj	M(A,B,0)	1Maj	-	-	M(A,B,0)	1Maj
$\sum m(0,7)$	M(M(M(A,B,0),C,0),M(M(A',B',0),C',0),1)	5Maj 3Not	M(M(B,C,1)',M(A,C,0),M(A',B,1)	4Maj 3Not	M(M(A,1,B), M'(B,1,C),M(A',1,C))	4Maj 2Not	M(1,M'(1,A,M((A,C,1)),M(0,A,M(B,C,0))	4Maj 2Not
$\sum m(1,6,7)$	M(M(A,B,0),M(M(A',B',0),C,0),1)	4Maj 2Not	M(M(A,B,1)',M(M(B,C,1),A,B'),B)	4Maj 2Not	M(M(A,0,B), M'(A,1,B),M(A,1,C))	4Maj 1Not	M(M(1,B,C),M'(A,B,1),M(A,B,M'(A,B,1)))	4Maj 1Not
$\sum m(4,5,6,7)$	M(A,0,1)	1Maj	A	1Maj	-	-	A	1Maj
$\sum m(1,5,6,7)$	M(M(A,B,0),M(B',C,0),1)	3Maj 1Not	M(M(B,C,1),M(A,B,1)',A)	3Maj 1Not	M(M(A,0,B),1,M(B',0,C))	3Maj 1Not	M(A,M(B,C,1),M(B',C,0))	3Maj 1Not
$\sum m(0,1,6,7)$	M(M(A,B,0),M(A',B',0),1)	3Maj 2Not	M(M(A,B,0),M(A,B,1)',1)	3Maj 2Not	-	-	M(B,M'(A,B,1),M(A,1,M'(A,B,1)))	3Maj 1Not
$\sum m(3,6,7)$	M(B,M(A,C,1),0)	2Maj	M(M(A,B,C),B,0)	2Maj 1Not	-	-	M(0,B,M(A,B,C))	2Maj
$\sum m(0,3,6,7)$	M(M(B,M(A,C,1),0,M(A',M(B',C',0),0),1)	5Maj 3Not	-	-	M(M(A',1,B),M'(B,1,C),M(A,B,C))	4Maj 2Not	M(1,M(B,0,M(A,1,C)),M'(1,B,M(A,1,C)))	4Maj 1Not
$\sum m(4,7)$	M(M(A,C,0),M(A,B,C'),M(A',B',C'))	3Maj 2Not	M(M(B,C,0),M(B,C,1)',A)	3Maj 2Not	M(M(B,0,C),A, M'(B,1,C))	3Maj 1Not	M(A,M'(A,B,C),M(B,C,0))	3Maj 1Not
$\sum m(2,4,7)$	M(M(A,C,0),M(A,B,C'),M(A',B',C'))	4Maj 3Not	-	-	-	-	M(M(A,C,0),M'(A,B,C),M(A,B,M'(A,B,C)))	4Maj 1Not
$\sum m(3,5,6,7)$	M(A,B,C)	1Maj	-	-	-	-	M(A,B,C)	1Maj
$\sum m(1,2,4,7)$	M(M(A',B,C),M(A,B',C),C')	3Maj 3Not	M(M(A,B,C)',M(A,B,C'),C)	3Maj 2Not	M(M'(A,B,C),A,M(A',B,C))	3Maj 2Not	M(B,M'(A,B,C),M(A,C,M'(A,B,C)))	3Maj 1Not
$\sum m(2,4,6,7)$	M(A,B,C')	1Maj 1Not	-	-	M(M(A,0,C),M'(A,B,C),M(A,B,C'))	4Maj 2Not	M(A,B, C')	1Maj 1Not
$\sum m(0,1,2,7)$	M(M(A',M(B,C,0),1),M(C',M(A,B',1),1),0)	5Maj 3Not	-	-	-	-	M(M(M(B,C,0),A,0),M'(1,A,M(0,B,C)),1)	4Maj 1Not

جدول ۳: نتایج شبیه‌سازی برای توابع ۳- ورودی/۲- خروجی

روش	تابع	خروجی	Maj.	Inv.	تعداد کل	
					Maj.	Inv.
[۱۷]	$F_1=\sum m(0,2,4,7)$	F_1=M(M(1,A,C)',M(A,B',C),M(1,B,C'))	۴	۳	۶	۴
	$F_2=\sum m(0,2,3,4)$	F_2= M(0,M(0,A,B)' ,M(1,B,C'))	۳	۲		
[۱۴]	$F_1=\sum m(0,2,4,7)$	F_1=M(M(C,1,M(A,B,C))',M(A',C,M(1,B,A')),A)	۵	۲	۷	۲
	$F_2=\sum m(0,2,3,4)$	F_2=M(M(1,B,A),A',M(1,C,M(A,B,C)'))	۴	۲		
[۱۸]	$F_1=\sum m(0,2,4,7)$	F_1 =M(M'(A,C,B),M(A,C',1),M(B,0,C))	۴	۲	۶	۴
	$F_2=\sum m(0,2,3,4)$	F_2 =M'(M'(0,B,C),M(B,0,A),C)	۳	۲		
روش پیشنهادی	$F_1=\sum m(0,2,4,7)$	F_1=M(M(C,0,M(A,B,0)),M'(C,1,M(A,B,0)),1)	۴	۱	۵	۲
	$F_2=\sum m(0,2,3,4)$	F_2=M(B,M'(A,B,0),M'(C,1,M(A,B,0)))	۳	۲		



شکل ۱۱: مدارهای به‌دست‌آمده از روش [۱۰] و روش ما برای تابع $\sum m(0,3,6,7)$ در جدول ۲

جدول ۴: نتایج شبیه‌سازی برای توابع ۳- ورودی/۴- خروجی

تابع	مرجع [۱۸]			مرجع [۱۵]			روش پیشنهادی		
	Output	Maj.	Inv.	Output	Maj.	Inv.	Output	Maj.	Inv.
$F_1=\sum m(1,4,5,7)$	M(C,B',A)	۱	۱	M(A,B',C)	۱	۱	M(A,B',C)	۱	۱
$F_2=\sum m(3,4,6)$	M(M(0,C',A),M(A,C,0)',M(B,C,0))	۴	۲	M(M(M(C,B,A),1,A),A',M(A,1',C')	۴	۳	M(M(C,1,M(A,B',C)),M(0,A,C'),M'(A,B',C))	۴	۳
$F_3=\sum m(0,2,5,6)$	M(M'(M'(C,0,A),A,B)),B,M(M'(B,0,C),A,C')	۵	۴	M'(M(A,1,C),M(1',B,C),M'(C,B,A))	۴	۳	M(M'(A,B,M(0,A,C')),A,M'(1,C,M(A,C,B')))	۵	۴
$F_4=\sum m(4,6,7)$	M(A,M(0,C',A),M(B,C,0))	۳	۱	M(M(A,B',C),B,M(A,1',C'))	۳	۳	M(A,B,M(0,A,C'))	۲	۱
Total		۹	۶		۹	۶		۶	۵

جدول ۵: نتایج شبیه‌سازی برای توابع ۴- ورودی/۲- خروجی

تابع	مرجع [۱۸]			مرجع [۱۵]			روش پیشنهادی		
	Output	Maj.	Inv.	Output	Maj.	Inv.	Output	Maj.	Inv.
$F_1=\sum m(0,2,6,12,13,14)$	M(M(B,1,A'),M(M'(B,C,1),C,D'),M(A,0,M(0,B,C')))	۶	۴	M(M'(A,1',C),M(D',M'(B,A,1),M(A,B,C)),M(1',B,A))	۶	۵	M(M(1,A',B),M(0,A,M(B,0,C')),M'(D,M(1,B,A'),M(0,B,C')))	۵	۳
$F_2=\sum m(1,3,4,5,7,12,13,15)$	M(D,M(1,A',B),M(C',0,B))	۳	۲	M(B,M'(A,C,1'),M(M'(B,A,1),D,M(A,B,C)))	۵	۳	M(M(1,B,A'),D,M(B,0,C'))	۳	۲
Total		۷	۴		۸	۵		۶	۳

جدول ۶: نتایج شبیه‌سازی برای توابع ۴- ورودی/۴- خروجی

تابع	مرجع [۱۷]			مرجع [۱۵]			روش پیشنهادی		
	Output	Maj.	Inv.	Output	Maj.	Inv.	Output	Maj.	Inv.
$F_1=\sum m(3,4,7,15)$	M(M(A',B,D),M(1,C,D)',M(0,C,D))	۴	۲	M(M(C, 0, D),M(D, M(C, B, A)',B), D')	۴	۳	M(B,M'(A,B,M(B,C,D)),M(0,C,M(B,D,M'(A,B,M(B,C,D)))))	۵	۱
$F_2=\sum m(1,3,4,9,13,15)$	M(M(A',B,D),M(B,C,D)',M(0,D,M(A,B',D)))	۵	۳	M(M(0, M(D', A, 1), B), M(D, M(C, B, A)', B), B')	۵	۴	M(M(B,D,M'(A,B,M(B,C,D))),M'(A,B,M(B,C,D))),M(0,A,D))	۵	۱
$F_3=\sum m(3,6,7,11,13,14,15)$	M(0,M(1,A,C),M(B,C,D))	۳	۰	M(C, D, M(0, M(D', A, 1), B))	۳	۲	M(C,M(0,A,D),M(B,C,D))	۳	۰
$F_4=\sum m(2,6,10,11,14)$	M(C,M(A',B,D),M(0,C,D'))	۳	۲	M(M(M(C, 0, D),M(D,M(C, B, A)',B), D')',0, C)	۵	۵	M(0,C,M'(B,D,M'(A,B,M(B,C,D)))	۴	۲
Total		۱۲	۵		۹	۴		۹	۲

مراجع

[1] C. S. Lent, P. D. Tougaw, W. Porod and G. H. Bernstein, "Quantum cellular automata," Nanotechnology, vol. 4, pp. 49–57, 1993.

[2] C. S. Lent and P. D. Tougaw, "A device architecture for computing with quantum dots," Proc. IEEE, vol. 85, no. 4, pp. 541-557, April 1997.

[3] W. Liu, S. Srivastava, L. Lu, M. O'Neill and E. E. Swartzlander, "Are QCA cryptographic circuits resistant to power analysis attack?," IEEE Transactions on Nanotechnology, vol. 11, no. 6, pp. 1239-1251, November 2012.

[4] P. Singh and R. Chandel, "Design and performance analysis of digital circuits using carbon nanotube transistors," In Inventive Communication and Computational Technologies (ICICCT), 2017 International Conference on, pp. 166-171. IEEE, 2017.

[5] A. Karimi and A. Rezai, "Improved device performance in CNTFET using genetic algorithm," ECS Journal of Solid State Science and Technology, 6(1), pp.M9-M12, 2017.

[6] A. Karimi and A. Rezai, "A design methodology to optimize the device performance in CNTFET," ECS Journal of Solid State Science and Technology, 6(8), pp.M97-M102, 2017.

[۷] حامد نجفعلی زاده و علی اصغر اروجی، «طراحی ساختاری از ترانزیستور ماسفت دو گیتی با به کارگیری دو ماده اکسید هافنیم (HfO_2) و سیلیسیم-ژرمانیوم (SiGe) در کانالی از جنس سیلیسیم (DM-DG)»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۷، شماره ۱، صفحات ۲۹۹-۳۰۴، ۱۳۹۶.

[۸] مهسا مهراد و میثم زارعی، «ارائه ساختاری جدید از ترانزیستورهای اثر میدان در مقیاس نانو به منظور بالا بردن قابلیت اطمینان»، مجله مهندسی برق دانشگاه تبریز، جلد ۴۸، شماره ۳، پاییز ۱۳۹۷.

[9] W. Liu, L. Lu, M. O'Neill and E. E. Swartzlander, "A first step toward cost functions for quantum-dot cellular automata designs," IEEE Transactions on Nanotechnology, vol. 13, no. 3, pp.476-487, May 2014.

[10] R. Zhang, K. Walus, W. Wang and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," IEEE Transactions on Nanotechnology, vol. 3, no. 4, pp. 443-450, December 2004.

[11] Z. Huo, Q. Zhang, S. Haruehanroengra and W. Wang, "Logic optimization for majority gate-based nanoelectronic circuits," Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 1307–1310, 2006.

[12] P. Wang, M. Y. Niamat, S. R. Vemuru, M. Alam and T. Killian, "Synthesis of majority/minority logic networks," IEEE Transactions on Nanotechnology, vol. 14, no. 3, pp.473-483, 2015.

[13] M. R. Bonyadi, S. M. R. Azghadi, N. M. Rad, K. Navi and E. Afjei, "Logic optimization for majority gate-based nanoelectronic circuits based on genetic algorithm," Proceedings of the IEEE International Conference on Electrical Engineering, pp. 1-5, April 2007.

[14] M. Houshmand, S. H. Khayat and R. Rezaei, "Genetic algorithm based logic optimization for multi-output majority gate-based nano-electronic circuits," Proceedings of the IEEE International Conference on Intelligent Computing and Intelligent Systems, pp. 584–588, November 2009.

[15] R. Rezaee, M. Houshmand and M. Houshmand, "Multi-objective optimization of QCA circuits with multiple outputs using genetic programming," Genet Program Evolvable Mach, pp. 95-118, 2013.

[16] Z. Beiki, M. Soryani and S. Mirzakuchaki, "Cell number optimization for quantum cellular automata based on genetic algorithm", Proceedings of the 3rd International Conference on Electronic Computer Technology, pp. 370–373, April 2011.

[17] M. A. Tehrani, K. Navi and A. Kia-kojoori, "Multi-output majority gate-based design optimization by using evolutionary algorithm," Swarm and Evolutionary Computation, vol. 10, pp. 25-30, 2013.

[18] M. H. Mahalat, M. Goswami, A. Mondal and B. Sen, "Synthesis and optimization of multi-objective multi-output QCA circuit using genetic algorithm," arXiv preprint arXiv: 1705.04099, 2017.

[19] G. Khademi, S. Soltani Fahraj, M. T. Moradgholi and M. Houshmand, "Logic optimization of QCA circuits using Ant colony optimization," Proceedings of the 22rd Iranian Conference on Electrical Engineering, May 2014.

[20] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent and G. L. Snider, "Realization of a functional cell for quantum dot cellular automata," Science, vol. 277, pp. 928-930, August 1997.

[21] I. E. Arani and A. Rezai, "Novel circuit design of serial–parallel multiplier in quantum-dot cellular automata technology," Journal of Computational Electronics, vol. 17, no. 4, pp.1771-1779, 2018.

[22] P. D. Tougaw and C. S. Lent, "Logical devices implemented using quantum cellular automata," Journal of Applied Physics, vol. 75, no. 3, pp. 1818–1825, 1994.

[23] H. Rashidi, A. Rezai and S. Soltany, "High-performance multiplexer architecture for quantum-dot cellular automata," Journal of Computational Electronics, vol. 15, no. 3, pp.968-981, 2016.

[24] H. Cho and E. E. Swartzlander, "Adder and multiplier design in quantum-dot cellular automata," IEEE Trans. Computers, vol. 58, No. 6, pp. 721–727, June 2009.

[25] C. S. Lent, M. Liu and Y. Lu, "Bennett clocking of quantum-dot cellular automata and the limits to binary logic scaling," Nanotechnology, vol. 17, pp. 4240-4251, 2006.

[26] J. F. Miller, "An empirical study of the efficiency of learning Boolean functions using a Cartesian genetic programming approach," Proc. Genetic and Evolutionary Computation Conference, pp. 1135–1142, 1999.

[27] J. F. Miller and P. Thomson, "Cartesian genetic programming," Proc. European Conference on Genetic Programming, vol. 1802, pp. 121–132, 2000.

[28] J. F. Miller, *Cartesian Genetic Programming*, Springer, Berlin Heidelberg, 2011.

[29] I. Rechenberg, *Evolutionsstrategie-Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution*, Ph.D. Dissertation, Technical University of Berlin, Germany, 1971.

زیرنویس‌ها

[1] Modules
[2] Gates
[3] Genetic Algorithm
[4] Majority
[5] Inverter
[6] Chromosome
[7] Fitness function
[8] Multi-Objective
[9] Cartesian genetic programming
[10] Clocking
[11] genotype
[12] phenotype
[13] Mutation