Vol. 3, No. 9, Jan-Mar 2024,(93-113)
Journal of New Achievements in Electrical, Computer and Technology (JNAECT)
مجله دستاوردهای نوین در برق،کامپیوتر و فناوری
دوره ۳، شماره ۹، زمستان ۱۴۰۲ (۱۱۳-۹۳)
journal homepage: https://jnaect.ir

ارائه متدولوژی و طراحی کاشی‌های خودسامانی مبتنی بر دنا جهت انجام محاسبات موازی

فاطمه کاظمی حسن‌آبادی[1]، محمدرضا رشادی نژاد[2]، زهره بیکی[3]

[1] دانشگاه اصفهان، دانشکده مهندسی کامپیوتر، گروه معماری سیستم‌های کامپیوتری ایمیل fatemehkazemi@eng.ui.ac.ir
[2] دانشگاه اصفهان، دانشکده مهندسی کامپیوتر، گروه معماری سیستم‌های کامپیوتری ایمیل m.reshadinezhad@eng.ui.ac.ir
[3] دانشگاه اصفهان، دانشکده مهندسی کامپیوتر، گروه معماری سیستم‌های کامپیوتری ایمیل z.beiki@eng.ui.ac.ir

چکیده

محاسبات مولکولی پدیده‌ای نوظهور از محاسبات در مقیاس نانو متری می‌باشد. تمرکز محاسبات مولکولی طراحی و ساخت پردازشگرهای مولکولی بر بستری از ماکرومولکول‌های زیستی مانند دی‌اکسی‌ریبونوکلئوتیداسید[1]، ریبونوکلئوتیداسید[2] و پروتئین‌ها می‌باشد [۱]. محاسبات دنا زیر شاخه‌ای از محاسبات مولکولی است که از واکنش شیمیایی بین رشته‌های دنا برای انجام محاسبات استفاده می‌کند. این محاسبات از نظر مقیاس‌پذیری، سادگی انجام و ساخت، هزینه‌های طراحی و پیاده سازی و ... تابحال موفق‌ترین روش از بین همه روش‌های شناخته شده برای محاسبات مولکولی می‌باشد [۲].

مدل‌های محاسباتی گوناگون با کاربردهای مختلف مبتنی بر رشته‌های دنا تا به امروز ارائه شده است، از جمله این مدل‌های محاسباتی، مدل محاسباتی خودسامانی می‌باشد. این مدل محاسباتی با توجه به قدرت خودسامانی کاشی‌ها، توان موازی‌سازی بسیار بالا و قاعده‌مند بودن آن بسیار مورد توجه قرار گرفت. با وجود این مزایا چالش‌هایی نظیر عدم کنترل‌پذیری کاشی‌ها و درصد خطای بالا و ایجاد کریستال‌های بزرگ در خروجی نهایی مانع پیشرفت این روش شده است. با حل مشکل کنترل‌پذیری کاشی‌ها در [۲]، امکان ارائه مدل مناسبی برای ساخت توابع منطقی با استفاده از این روش فراهم شده است. در این مقاله روشی برای طراحی توابع منطقی توسط کاشی‌های خودسامانی ارائه شده است. علاوه بر این قابلیت گسترش و کاهش هزینه‌های ساخت و همچنین توسعه روش برای ایجاد روند طراحی خودکار در این مقاله ارائه شده است. متدولوژی معرفی شده در این مقاله می‌تواند سرآغاز امید بخشی برای انجام محاسبات به صورت موازی باشد.

واژه‌های کلیدی: محاسبات مبتنی بر دنا، کاشی‌های خودسامانی، روش طراحی دوخطی[3]، مدل‌های محاسباتی، محاسبات موازی.

[1] Deoxyribonucleic acid

[2] Ribonucleic acid

[3] Dual Rail

1. مقدمه

قدرت پردازش کامپیوترهای امروزی از سال 1970 افزایش یافت تا اینکه اخیراً این افزایش قدرت پردازش به دلیل چالش فناوری‌های ساخت اشباع شده است. برای حل این چالش‌ها از روش چند هسته‌ای کردن پردازنده‌ها و موازی‌سازی پردازش‌ها بر روی هسته‌های پردازشی استفاده شده است. قرار دادن چند هسته پردازشی بر روی یک تراشه به دلیل توان مصرفی زیاد و دمای بالای تراشه محدود به 16 هسته می‌شود [۳]. این چالش‌ها باعث شد ارائه تکنیک‌های دیگری برای بالا بردن قدرت پردازشی جز اهداف پژوهشگران قرار گیرد.

محاسبات مولکولی که از بارزترین آن‌ها محاسبات مبتنی بر پلیمر دنا می‌باشد؛ قدرت پردازش موازی و ذخیره‌سازی بالای اطلاعات را دارند. قدرت بالای پردازش موازی محاسبات مبتنی بر دنا از واکنش شیمیایی کاملاً موازی مولکول‌ها در یک محلول نشأت می‌گیرد. همچنین مولکول‌های دنا از قرن‌ها پیش تا به امروز اثبات کرده‌اند که قابلیت نگهداری و انتقال اطلاعات ژنتیکی و وراثتی بالایی را دارند [۴]. در ادامه به بحث و بررسی قابلیت موازی‌سازی بالای محاسبات مبتنی بر دنا پرداخته می‌شود.

قدرت بالای موازی‌سازی محاسبات مبتنی بر دنا ابتدا توسط Adlman با حل مسئله‌ی مسیر همیلتونی که جز دسته مسائل NP-complete می‌باشد به اثبات رسید [۵]. Adlman با تشکیل کل فضای جواب ممکن برای مسیرهای همیلتونی و فیلتر کردن نتایج تا رسیدن به جواب نهایی توانست اثبات کند که محاسبات مبتنی بر دنا قابلیت موازی‌سازی بالایی دارد. حل این مسئله در کامپیوترهای الکترونیکی با قدرت پردازش بالا برای مسائل پیچیده‌تر زمان محاسبات نمایی دارد. یکی از چالش‌های محاسبات مبتنی بر دنا عدم وجود مدل محاسباتی مناسب می‌باشد. Adlman [۵] و Lipton [۳] یک مدل محاسباتی براساس تجربه‌ی Adlman به نام مدل محاسباتی فیلترینگ ارائه کردند. همچنین Wang[۶] از مدل Adlman-Lipton استفاده کرد و اثبات کرد که فناوری محاسبات مبتنی بر دنا برای حل مسائل NP-hard گراف‌های بدون جهت با زیرمجموعه‌ای از جواب‌ها که در حالت عادی حل نمایی دارند را می‌توان با پیچیدگی $O(n^3)$ حل نمود. در ادامه به توضیح چند نمونه از بارزترین مدل-های محاسباتی فناوری محاسبات مبتنی بر دنا و توضیح مختصری جهت آشنایی با آن‌ها داده خواهد شد.

Winfree و Rothermund مدل محاسباتی سازنده مبتنی بر دنا را معرفی کردند. اساس کار این مدل بر پایه‌ی خودسامانی بودن رشته‌های دنا می‌باشد. تئوری فرش کردن یک سطح با استفاده از کاشی‌ها توسط wang مطرح گردید که Winfree از همین تئوری استفاده کرد و با طراحی چند نمونه کاشی از جنس رشته‌های دنا توانست کریستالی از این کاشی‌ها بسازد. ساخت این کریستال‌ها نماد انجام یک محاسبه می‌باشد. در[۷] Winfree با ساخت چند نمونه کاشی از جنس دنا توانست شمارش اعداد را انجام دهد. بعد از آن به دلیل ساخت کریستال بزرگ در خروجی و درصد خطای بالای محاسبات با این روش سرعت تحقیقات در این حوزه کاهش یافت.

مدل محاسباتی مبتنی بر مدارهای بولین از دیگر دستاوردها در این فناوری می‌باشد که با ساخت دروازه منطقی آلاکلنگ[۴] توسط Winfree و همکارانش به اوج خود رسید [۸]. این مدل براساس و پایه‌ی ساخت دروازه‌های منطقی با عملکرد دروازه-های منطقی دیجیتالی می‌باشد. با استفاده از دروازه منطقی آلاکلنگ قابلیت استفاده آبشاری از دروازه‌های منطقی در چندین سطح از مدارها فراهم گردید. استفاده از دروازه‌های منطقی با توجه به ذات محاسباتی آن‌ها که ورودی یک دروازه منطقی خروجی دروازه منطقی قبلی می‌باشد، قدرت انجام موازی محاسبات را کاهش می‌دهد. در این گونه مدارها تاخیر و پیچیدگی زمانی بسته به طولانی‌ترین مسیر دروازه‌های منطقی در مدارها دارد. در مدارهای منطقی پیچیدگی زمانی از مرتبه‌ی $O(n)$

[4] Seesaw

می‌باشد. همچنین این مدل‌ها برای حل مسائل NP-complete و NP-hard مناسب نمی‌باشد. علاوه بر اینکه روش ساخت مدارها با استفاده از آلاکلنگ قدرت موازی‌سازی ندارد، توانایی طراحی انعطاف‌پذیر و خودکار برای مدارهای منطقی مبتنی بر محاسبات دنا را نیز ندارد. در این مقاله مدل پردازش موازی برای ساخت مدارهای منطقی ارائه شده است. با استفاده از این مدل یک متدولوژی برای طراحی مدارهای منطقی ارائه شده که با استفاده از حل چالش‌های روش خودسامانی توانسته مدارهای منطقی را با درجه‌ی پیچیدگی O(1) به صورت کاملاً موازی پیاده‌سازی کند.

در ادامه، در بخش 2 به معرفی مفاهیم اساسی برای مدل ارائه شده، پرداخته شده است. در بخش 3 متدولوژی طراحی شده معرفی می‌گردد. بخش 4 پیاده‌سازی مدارهای منطقی با متدولوژی را مورد بررسی قرار داده است و در بخش نهایی نتیجه‌گیری این مقاله ارائه شده است.

2. مفاهیم اساسی

همانطور که در بخش 1 اشاره شد، پیش از این توسط نویسندگان این مقاله، یک متدولوژی جدید برای ساخت سیستم‌های دیجیتال کاملاً موازی مبتنی بر محاسبات دنا ارائه شده است. ساخت مدارهای منطقی در سیستم محاسبات مبتنی بر دنا مستلزم استفاده از دروازه‌های منطقی مانند آلاکلنگ می‌باشد که باعث از بین رفتن قدرت پردازش موازی رشته‌های دنا می‌شوند. به همین منظور در این مقاله از ساختارهای جدیدی برای کاشی‌های مدل خودسامانی استفاده شده که در این ساختارها قابلیت کنترل اتصال کاشی‌ها به آن‌ها اضافه شده است [۹]. همچنین اتصالات این نوع کاشی‌ها در نهایت منجر به تولید تک رشته دنا می‌شود که می‌توان از آن به عنوان ورودی در مراحل بعدی استفاده کرد. این ویژگی‌ها امکان طراحی توابع بولی به صورت کاملاً موازی را ایجاد می‌کند. بنابراین در این مقاله متدولوژی معرفی شده است که سیستم‌های دیجیتالی متشکل از مدارهای منطقی را به صورت کاملاً موازی با رشته‌ها دنا پیاده‌سازی می‌کند. در ادامه به معرفی اجمالی توابع منطقی پرداخته خواهد شد. همچنین منطق دوخطی بیان می‌شود تا بتوان با استفاده از آن عدم وجود تکنیک مناسب برای پیاده‌سازی دروازه منطقی[5] Not در محاسبات مبتنی بر دنا را برطرف کرد. سپس مدل محاسباتی خودسامانی و ساختار جدید معرفی شده برای کاشی‌ها معرفی می‌گردد.

2-1- توابع منطقی و منطق دوخطی

الف) توابع منطقی و نحوه‌ی پیاده‌سازی آن‌ها

سیستم‌های دیجیتال متشکل از مجموعه‌ای از مدارهای منطقی بر پایه‌ی جبر بول می‌باشد. یکی از مهمترین عملیات در جبر بولی، استفاده از دروازه‌ی NOT می‌باشد و تمامی توابع بولی را می‌توان با استفاده از دروازه‌های NOR یا NAND پیاده‌سازی کرد. پیاده‌سازی دروازه‌ی NOT در محاسبات مبتنی بر دنا چالش‌های بسیار فراوانی دارد [۱۰].

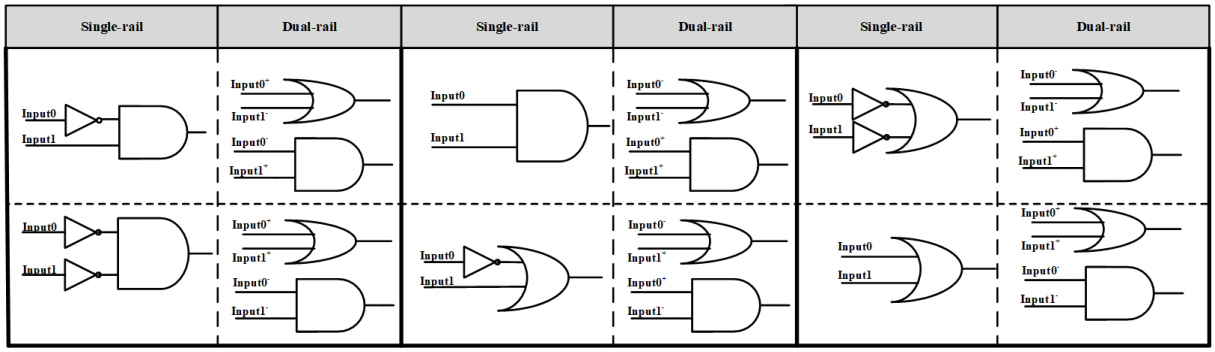
ب) منطق دوخطی

همانطور که گفته شد یکی از اصلی‌ترین چالش‌هایی که در محاسبات مبتنی بر دنا وجود دارد، عدم وجود تکنیک مناسب برای پیاده‌سازی عملیات NOT می‌باشد. با توجه به اینکه عمل NOT یکی از عمل‌های اصلی توابع بولی می‌باشد پیاده‌سازی مدارهای منطقی بر روی بستر دنا نیازمند پیاده‌سازی این عمل می‌باشد. تاکنون چندین روش برای پیاده‌سازی آن ارائه شده است که هیچ کدام کارایی مناسب نداشته‌اند.

در [۳] برای اینکه دروازه منطقی Not را پیاده‌سازی نکند از طراحی دوخطی استفاده کرده است. پیاده‌سازی منطق دوخطی مستلزم استفاده از تعداد دروازه‌های منطقی بیشتر می‌باشد. قوانین منطق دوخطی در شکل 1 نشان داده شده است. ورودی‌ها $Input0^+$ و $Input1^+$ و مکمل آن‌ها $Input0^-$ و $Input1^-$ می‌باشند. شکل 1 نحوه‌ی تبدیل از single rail را به دوخطی نشان

[5] Gate

می‌دهد. به عنوان مثال اگر تابع $\bar{x}y$ به صورت single rail پیاده‌سازی شود نیازمند یک دروازه منطقی Not بر روی ورودی x و یک دروازه منطقی AND برای انجام عمل AND بین y و x' می‌باشد. همانطور که در شکل 1 سطر اول نشان داده شده است. پیاده‌سازی دوخطی همین تابع نیازمند دو ورودی مثبت و منفی به ازای خود ورودی و مکمل آن می‌باشد. در واقع در منطق دوخطی بایستی هم پیاده‌سازی SOP و هم دوگان آن را داشته باشیم. به اینصورت که برای تابع $\bar{x}y$ باید $(x+\bar{y})$ را نیز پیاده‌سازی کرد. برای پیاده‌سازی $\bar{x}$ و $\bar{y}$ از ورودی‌های مکمل آن‌ها استفاده می‌شود. به عبارتی به صورت $(Input0^{+}+Input1^{-})$ و $Input0^{-}.Input1^{+}$ در سطرهای اول و دوم ستون‌های اول و دوم از سمت چپ شکل 1 نشان داده شده است. در شکل 1 چند نمونه از پیاده‌سازی توابع منطقی با استفاده از روش دوخطی نشان داده شده است.



شکل 1: نحوه‌ی تبدیل مدارهای single rail به دوخطی [۳].

2-2- مدل محاسباتی خودسامانی و ساختارهای خودسامانی

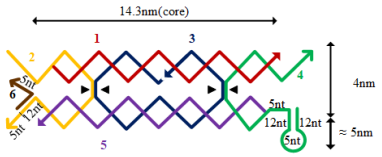
مدل خودسامانی توسط Winfree در سال 1998 مطرح گردید[۷]. این مدل بر پایه‌ی ساختارهای سازنده طراحی گردید. قرار گرفتن این ساختارها در کنار هم که با عنوان کاشی شناخته می‌شوند، می‌تواند نماد انجام یک محاسبه باشد. ایده‌ی اصلی آن از تئوری کاشی کردن سطوح که توسط Wang[۶] مطرح شده بود گرفته شد و توسعه یافت. در [2] ساختار اولیه‌ی کاشی‌ها ، نحوه‌ی اتصال آن‌ها، قدرت اتصال آن‌ها و قوانین اتصال آن‌ها نشان داده شده است. در ادامه در بخش 2-2-1 به معرفی ساختار تغییریافته کاشی‌ها که این مدل را مناسب برای ساخت متدولوژی ساخت مدارهای منطقی مبتنی بردنا این مقاله کرده است، پرداخته می‌شود.

2-2-2- ساختار تغییر یافته کاشی‌ها جهت بهبود مدل محاسباتی خودسامانی

Winfree با ساخت ساختارهای کاشی‌ها که در [2] نشان داده شده است توانست مدل محاسباتی خودسامانی را طراحی کند. این کاشی‌ها شامل 5 رشته بلند دنا می‌باشد که با نام کاشی DAE-E شناخته می‌شود [۱۱]. این کاشی یکی از ساختارهای DX یا Double Crossover (دو محل تقاطع) هستند. در[۲] ساختار کاشی DAE-E نشان داده شده در [2] را به گونه‌ای تغییر دادند که بازوهای اتصال آن‌ها در حالت عادی به صورت غیر فعال هستند. به این معنی که قابلیت اتصال به کاشی‌های مجاور را ندارند. این کاشی‌ها تا زمان دریافت ورودی از جنس تک رشته‌های دنا غیر فعال باقی می‌مانند. بنابراین این مدل محاسباتی قابلیت دریافت ورودی را با این تغییرات پیدا کرده است. زمانی که تمام ورودی‌ها دریافت شده و بازوها فعال شدند کاشی‌ها به صورت موازی به هم متصل می‌گردند و در نهایت با تغییر کاشی‌های نهایی که تحت عنوان INT شناخته می‌شوند، تک رشته‌ای از جنس دنا آزاد می‌گردد که خروجی این مدل محسوب می‌شود. این تغییر در ساختار کاشی‌ها با مخفی

کردن آغازگر[۶]های مناسب در قسمت حلقه‌ی ساختارهای سنجاق‌سر[۷] به وجود آمده است. شکل2 ساختار تغییر یافته کاشی معرفی شده در [2] را نشان می‌دهد.

نحوه‌ی فعال‌سازی، واکنش آن‌ها و تولید خروجی توسط این کاشی‌ها درشکل3 نشان داده شده است. شکل3 (الف) ساختار دو کاشی غیر فعال این مدل محاسباتی را نشان می‌دهد. شکل3(ب) کاشی غیر فعال سمت چپ با استفاده از رشته‌ی ورودی به صورت فعال در آمده و آمادگی اتصال به کاشی سمت راست را دارد. نحوه‌ی فعال‌سازی این کاشی به این صورت است که ورودی <a^ m> وارد محیط آزمایش شده و آغازگر a و a* واکنش می‌دهند و به مرور زمان با بالا رفتن انرژی سیستم قسمت ساقه‌ی سنجاق‌سر باز خواهد شد و آغازگر ta1 آشکار می‌گردد. در شکل3(ج) دو کاشی قسمت (الف) به صورت فعال هستند و آمادگی اتصال بهم دیگر را دارند. آغازگر ta1 به ta1* کاشی سمت راست متصل شده و تک رشته‌ی <ta2^ m> را به عنوان خروجی این واکنش آزاد می‌کند. در نتیجه با دریافت ورودی مناسب، خروجی مناسب تولید گردید و همچنین کریستال محاسبه نیز تشکیل گردید [۲]. با استفاده از این تغییرات در ساختار کاشی‌ها مدل خودسامانی قابلیت انجام محاسبات NP-complete به طور خاص یافتن مسیر همیلتون از گراف موردنظر را پیدا کرده است. همچنین ساختارهایی برای دروازه‌های منطقی AND و OR ارائه شده است [۲]. با استفاده از پیاده‌سازی گراف و همچنین استفاده از دروازه‌های منطقی در ادامه روش محاسباتی کاملا موازی برای طراحی مدارهای منطقی ارائه شده است.



شکل2: ساختار تغییریافته کاشی مدل خودسامانی [۲]

3-متدولوژی ساخت مدارهای منطقی با استفاده از مدل محاسباتی خودسامانی

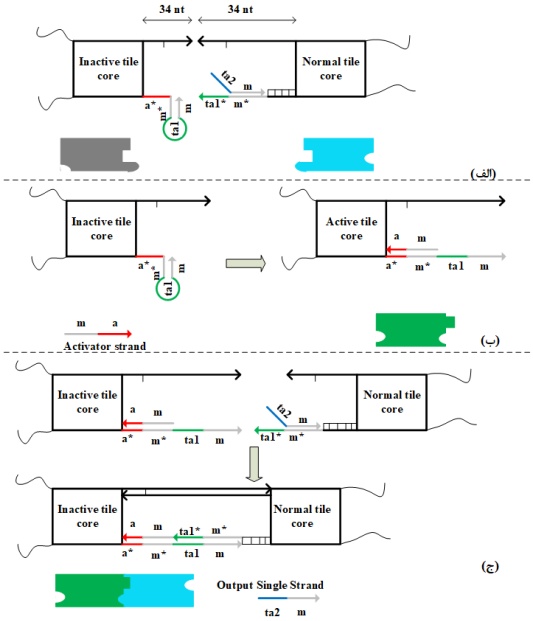
همانطور که قبلاً اشاره شد مدارهای منطقی توسط توابع منطقی نمایش داده می‌شوند که این تابع از جبر بولی پیروی می‌کنند. همچنین با توجه به عدم وجود سازوکار مناسب برای پیاده‌سازی دروازه منطقی NOT در محاسبات مبتنی بر دنا منطق دوخطی معرفی گردید. در این بخش به معرفی متدولوژی برای ساخت مدارهای منطقی با توجه به توابع آن‌ها به صورت دوخطی پرداخته می‌شود که در طراحی کاشی‌های مدل محاسباتی خودسامانی از کاشی‌های قابل کنترل[۲] استفاده می‌شود. طراحی کاشی‌ها در این مدل محاسباتی کاملاً موازی به گونه‌ای است که بازوی راست تمام کاشی‌ها غیرفعال هستند و با دریافت تک رشته‌ای از جنس دنا فعال می‌شوند. ساختار این بازوها با رشته‌های سنجاق‌سر طراحی می‌گردد. بازوی سمت چپ کاشی‌ها فعال می‌باشد و مخصوص اتصال دو کاشی فعال شده به همدیگر است. هدف اصلی در این مقاله طراحی یک سیستم اتوماتیک برای طراحی هر نوع مدار با هر پیچیدگی زمانی می‌باشد. به صورتی که تابع f(x) با هر درجه پیچیدگی که وجود داشته باشد با طراحی بازوها می‌توان آن را با این متدولوژی و پیچیدگی زمانی O(1) به صورت اتوماتیک طراحی نمود.

به منظور اثبات این ادعا یک تابع و دوگان آن در نظر گرفته می‌شود. سپس گراف موردنظر آن طراحی می‌گردد و کاشی‌های آن طبق سازوکار معرفی شده در [۲] طراحی می‌گردند. تابع $f(x) = \bar{A}B + AC$ و دوگان آن به صورت $\overline{F(x)} = (A + \bar{B}).(\bar{A} + \bar{C})$ درنظر گرفته می‌شود. برای طراحی گراف آن باید توجه داشت که اتصال سری کاشی‌ها

[6] Toehold

[7] Hairpin

در این متدولوژی معنای عملیات AND و مسیرهای موازی به معنای عملیات OR می‌باشد. پس برای قسمت‌های $\bar{A}B$ و AC مسیرهای سری جداگانه طراحی می‌گردد و برای $(A+\bar{B})$ و $(\bar{A}+\bar{C})$ به صورت جداگانه دو مسیر موازی طراحی می‌گردد که با کاشی‌های میانی رابط عملیات AND این دو برقرار می‌گردد. گراف موردنظر پیاده‌سازی شده در شکل4 دیده می‌شود. همچنین لازم به ذکر است با توجه به استفاده از منطق دوخطی ورودی A و $\bar{A}$ مربوط به خود ورودی و مکمل آن می‌باشد. این برای ورودی B نیز صدق می‌کند. شکل5 ساختار دروازه منطقی‌های منطقی این تابع را نمایش می‌دهد.



شکل3: (الف) ساختار کاشی‌های طراحی شده با بازوهای فعال و غیر فعال. (ب) نحوه‌ی فعال شدن بازوی سمت راست کاشی با استفاده از تک رشته‌ی دنا که به عنوان ورودی(<m a^>) دریافت می‌گردد. (ج) نحوه‌ی اتصال بازوی فعال شده به کاشی مجاور و تولید رشته دنا خروجی (<m ta2^>) [۲].

ابتدا در بخش 3-1 به توضیح ساختار طراحی شده برای تابع موردنظر با استفاده از کاشی‌های طراحی شده در[۲] پرداخته می‌شود و سپس در بخش 3-2 به توضیح روش بهینه شده برای پیاده‌سازی تابع پرداخته می‌شود.

3-1- معرفی روشی برای طراحی مدارهای منطقی کاملاً موازی با استفاده از مدل محاسباتی بهبود یافته (پیاده-سازی گراف پیمایشی)

مدل خود سامانی ابتدا توسط Winfree برپایه‌ی محاسبات مبتنی بر دنا ارائه گردید [۲]. Winfree ساختار کاشی-هایی را معرفی کرد که اتصال آن‌ها به همدیگر قابلیت انجام محاسبات را داشت. این مدل، محاسبات را به صورت کاملا موازی و قاعده‌مند انجام می‌دهد. در کنار این مزایای بارز مدل خود سامانی چالش‌هایی همچون کنترل‌پذیر نبودن اتصال کاشی‌ها، ساخت کریستال بزرگ در خروجی و خطای بالای محاسبات را داشت. چالش عدم کنترل‌پذیری کاشی‌ها توسط نویسندگان همین مقاله با غیرفعال کردن بازوهای چسبنده برطرف گردید. همین امر باعث شد امکان ارائه روشی برای طراحی مدارهای منطقی کاملا موازی با استفاده از مدل خودسامانی بهبودیافته [۲] فراهم گردد که در ادامه به بحث در مورد آن پرداخته می-شود. برای طراحی این روش ابتدا نمایش مناسبی برای توابع با استفاده از منطق دوخطی ارائه گردید. سپس در ادامه عنوان می‌شود که ارائه این روش به دو صورت پیاده‌سازی توسط گراف پیمایشی و پیاده‌سازی توسط دروازه‌های منطقی صورت می-گیرد. همانطور که در بخش قبل توضیح داده شد شکل 4 گراف پیمایشی برای تابع موردنظر را نشان می‌دهد. در این گراف از

سمت بالا به سمت پایین 4 مسیر وجود دارد که از گره‌ی S آغاز می‌شوند. مسیرها با رشته‌ی enable یا <m e0^> فعال می‌گردند. اگر این رشته در محیط آزمایش وجود نداشته باشد هیچ کدام از مسیرها فعال نمی‌شوند. این رشته به منظور هماهنگی آغاز واکنش‌ها قرار داده شده است. به صورت پیش‌فرض این رشته در محیط آزمایش قرار داده می‌شود. این رشته در 4مسیر مجزا دخیل می‌باشد به همین دلیل غلظت اولیه آن 4 برابر شده است.

کاشی گره‌ی S یا seed در شکل 6 دیده می‌شود. ساختار کلی کاشی‌ها در این پیاده‌سازی‌ها به اینصورت است که دامنه‌ی همگی بازوها با m نام‌گذاری شده است. نام‌گذاری آغازگرها بسته به استفاده‌ی کاشی‌ها متفاوت است. در این کاشی seed آغازگر *e0 برای واکنش با رشته‌ی enable استفاده می‌شود. آغازگر e0 از رشته‌ی enable با *e0 از بازوی سمت راست کاشی seed واکنش می‌دهد و باعث باز شدن رشته‌ی سنجاق‌سر و آزاد شدن آغازگر t1 که مخفی شده می‌شود. با انجام این واکنش کاشی seed دارای بازوی چسبنده‌ی <m t1^> می‌باشد.

مسیر اول و دوم که شامل گره‌های Q1، Q2، Q3، Q4و F^+ است. برای پیاده‌سازی تابع F که به صورت $f(x) = \bar{A}B + AC$ است استفاده می‌شوند. مسیر اول $\bar{A}B$ را پیاده‌سازی می‌کند. همانطور که در شکل 4 دیده می‌شود Q1، Q2 و F^+ به صورت سری به هم متصل هستند. اتصال سری معنای عملیات بولی AND را می‌دهد. در ادامه به معرفی ساختار کاشی‌های مسیر اول پرداخته می‌شود و نحوه‌ی واکنش آن‌ها بررسی می‌گردد.

گره‌ی Q1 با طراحی tile1 پیاده‌سازی می‌شود. این کاشی دارای دو بازوی سمت راست و چپ می‌باشد. بازوی سمت چپ با تک رشته‌ی <m* t1*> به بازوی سمت راست کاشی seed متصل می‌گردد. بازوی سمت راست آن با دریافت ورودی $\bar{A}$ همان رشته‌ی <m NA^> فعال می‌گردد. زمانی که NA از رشته‌ی ورودی $\bar{A}$ با *NA ازکاشی tile1 واکنش بدهند قسمت سنجاق‌سر باز شده و آغازگر t2^ آشکار می‌گردد. در این مرحله آمادگی اتصال به tile2 مربوط به گره‌ی Q2 را پیدا می‌کند. ساختار این کاشی در شکل 7 دیده می‌شود.

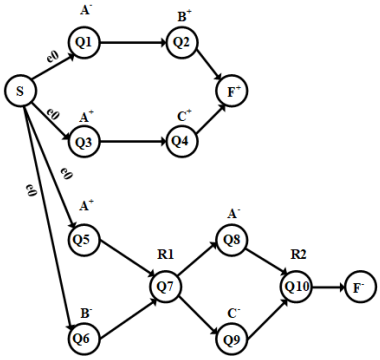
گره‌ی Q2 با طراحی کاشی tile2 پیاده‌سازی شده که در شکل8 دیده می‌شود. این کاشی دارای دو بازو می‌باشد. بازوی سمت چپ به صورت دو رشته‌ای و فعال می‌باشد. زمانی که بازوی سمت راست کاشی tile1 فعال شود t2 و *t2 واکنش می‌دهد و باعث آزاد شدن تک رشته‌ی <e1^ m> می‌شود. همچنین اتصال tile1 از سمت راست با tile2 از سمت چپ برقرار می‌گردد. بازوی سمت راست tile2 با دریافت ورودی B که با تک رشته‌ی <m B^> نشان داده شده فعال می‌گردد. زمانی که آغازگرهای B و *B واکنش بدهند قسمت سنجاق‌سر آن باز شده و آغازگرed1 آشکار می‌گردد.

گره‌های Q3، Q4 و F^+ به صورت سری متصل هستند و مسیر دوم را تشکیل می‌دهند. اتصال سری آن‌ها نماد عملیات AND بولی می‌باشد. این مسیر قسمت AC تابع F را پیاده‌سازی می‌کند.

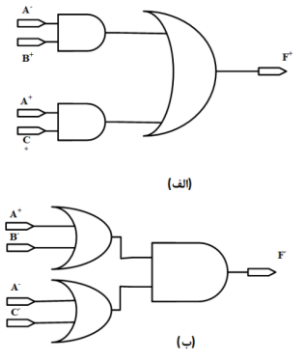
گره‌ی Q3 با طراحی tile3 که در شکل9 دیده می‌شود پیاده‌سازی می‌شود. این کاشی‌ها دارای دو بازوی سمت راست و چپ است که بازوی سمت چپ به منظور اتصال به tile0 که برای گره‌ی seed می‌باشد طراحی گردیده است. بازوی سمت راست آن برای دریافت ورودی A که تک رشته‌ی <m A^> است طراحی گردیده است. اگر قبلا t1 از بازوی سمت راست کاشی seed آشکار شده باشد بازوی سمت چپ کاشی tile3 با آن واکنش داده و بهم دیگر متصل می‌شوند. زمانی که آغازگر A و *A واکنش بدهد سنجاق‌سر باز شده و آغازگر t3 آشکار می‌گردد. به این ترتیب بازوی سمت راست کاشی tile3 هم فعال می‌گردد.

کاشی Tile4 برای گره‌ی Q4 طراحی گردیده است که در شکل10 دیده می‌شود. ساختار بازوی چپ آن به صورت دو رشته‌ای است که به محض فعال شدن بازوی سمت راست Tile3 به آن متصل می‌گردد و تک رشته‌ی <m e2^> آزاد می‌گردد.

همچنین بازوی سمت راست آن به منظور دریافت رشته‌ی <m c^> که همان ورودی C است طراحی گردیده است. با واکنش آغازگرهای c و *c سنجاق‌سر این بازو، باز خواهد شد و آغازگر ed2 آشکار می‌گردد. به این ترتیب این بازو به حالت فعال در می‌آید.



شکل4: گراف تابع موردنظر پیاده‌سازی شده برای معرفی متدولوژی موردنظر.

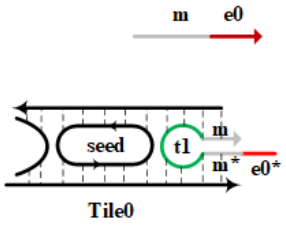


شکل5: ساختار دروازه منطقی‌های تابع موردنظر پیاده‌سازی شده برای معرفی متدولوژی موردنظر.

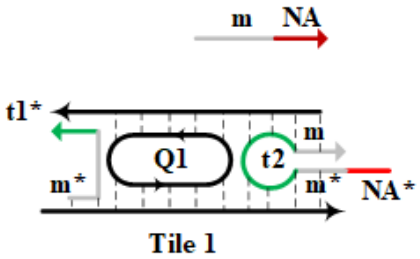
گره‌ی F^+ با طراحی tile11 و tile 12 که در شکل11 نشان داده شده‌اند، پیاده‌سازی می‌شوند. اتصال به کاشی‌های گره‌ی F^+ به معنی پایان محاسبات تابع f است. زمانی که آغازگر ed1 از بازوی سمت راست کاشی tile2 آشکار شد با *ed1 از بازوی سمت چپ tile11 واکنش می‌دهد و دامنه‌ی m جدا می‌گردد. در نتیجه‌ی بالا رفتن انرژی سیستم آغازگر e1 نیز جدا شده و باعث آزادسازی آغازگر *e1 می‌شود. اگر اتصال گره‌های Q1 و Q2 هم برقرار شده باشد و تک رشته‌ی <e1^ m> آزاد شده باشد، e1 با *e1 واکنش می‌دهد و دامنه‌ی m جدا می‌گردد. در نهایت تک رشته‌ی<res^ m> به عنوان خروجی مسیر شناخته می‌شود. همچنین بازوی سمت چپ tile12 برای تشکیل شدن مسیر دوم می‌باشد. اگر ed2 از بازوی سمت راست tile4 آشکار شده باشد با *ed2 از بازوی سمت چپ tile12 واکنش می‌دهد و باعث جدا شدن m می‌شود و آغازگر *e2 آزاد می‌گردد. همچنین اگر اتصال Q3 و Q4 برقرار شده باشد تک رشته‌ی <e2^ m> با *e2 واکنش می‌دهد و m را جدا می‌کند. در نهایت رشته‌ی <res^ m> به عنوان خروجی مسیر دوم آشکار می‌شود. این دو مسیر نمایان‌گر پیاده‌سازی تابع f می‌باشند اگر تک رشته‌ی خروجی با غلظت قابل قبول وجود داشته باشد این تابع خروجی یک تولید کرده است.

شکل 12 پیاده‌سازی مسیر اول از گراف شکل 6 که شامل گره‌های Seed، Q1، Q2 و F^+ با پیاده‌سازی کاشی‌های Tile0، Tile1، Tile2 و Tile11 را نشان می‌دهد. شکل12 (الف) نمایانگر کاشی‌ها و ورودی‌های استفاده شده در این مسیر می‌باشد. شکل 12 (ب) نحوه‌ی فعال‌سازی Tile0 با ورودی فعال‌کننده(enable) و همچنین اتصال آن به بازوی سمت چپ کاشی tile1 و فعال شدن بازوی سمت راست tile1 با ورودی اول را نشان می‌دهد. شکل 12 (ج) اتصال کریستال تشکیل شده در مرحله‌ی قبل از بازوی سمت راست tile1 به بازوی سمت چپ tile2 و آزاد شدن تک رشته‌ی <m e1^> را نشان می‌دهد. همچنین بازوی سمت راست کاشی tile2 با ورودی B فعال شده و آغازگر ed1 آشکار می‌گردد. شکل12 (د) اتصال کریستال تشکیل شده در مرحله قبل به tile11 با کمک تک رشته‌ی <m e1^> بهم دیگر را نشان می‌دهد که در نتیجه‌ی این اتصال تک رشته‌ی <res^ m> به عنوان خروجی تابع F^+ آزاد می‌گردد. آغازگر ed1 از بازوی سمت راست tile2 با بازوی سمت چپ tile11 واکنش می‌دهد و باعث آزاد شدن آغازگر *e1 می‌شود. اگر اتصال tile2 و tile1 برقرار شده باشد رشته‌ی <e1^ m> آزاد شده و می‌تواند با این آغازگر *e1 واکنش دهد و خروجی <res^ m> آزاد می‌کند. شکل 12 نحوه‌ی اتصال کاشی‌های هر

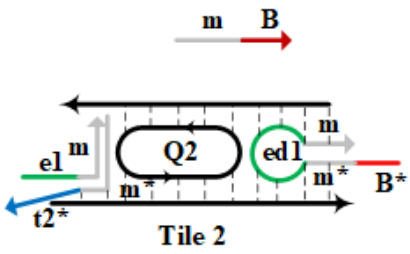
مسیر در صورت وجود تک رشته‌های فعال‌کننده و تولید تک رشته‌های خروجی را نشان می‌دهد. بقیه‌ی مسیرهای گراف شکل 6 هم به همین منوال تولید خواهند شد ولی به دلیل بزرگ شدن شکل‌های آن‌ها از قرار دادن آن‌ها در مقاله خودداری شد.



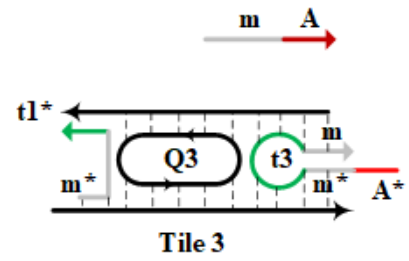
شکل6: ساختار کاشی مربوط به گره‌ی seed با استفاده از گراف پیمایشی.



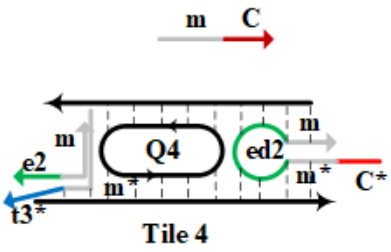
شکل7: پیاده‌سازی tile1 برای گره‌ی Q1.



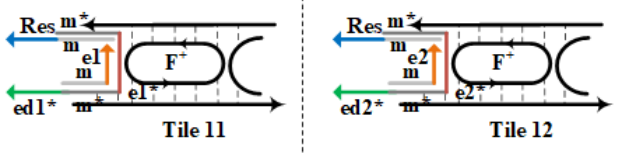
شکل8: ساختار tile2 برای گره‌ی Q2.



شکل 9: ساختار tile3 پیاده‌سازی شده برای گره‌ی Q3 .

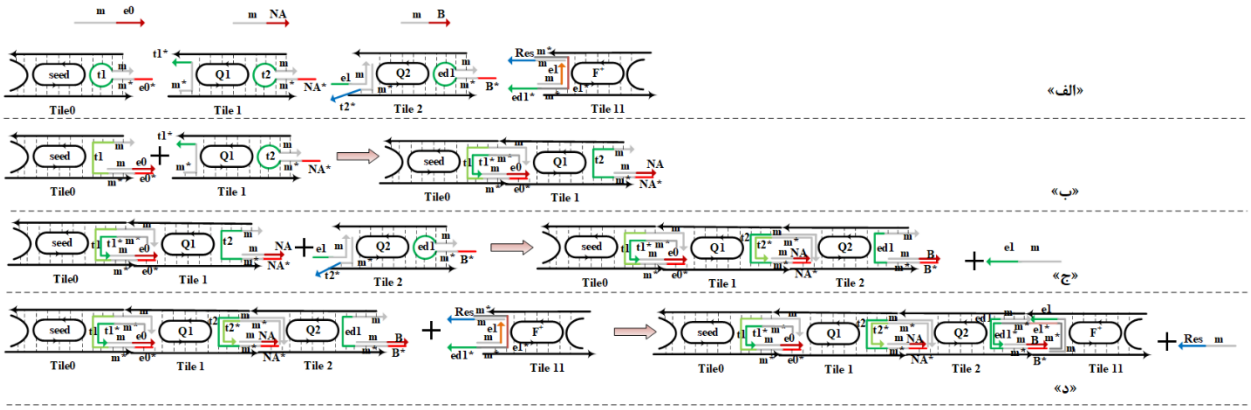


شکل10: ساختار tile4 پیاده‌سازی شده برای گره‌ی Q4.



شکل 11: ساختار tile11 و tile12 پیاده‌سازی شده برای گره‌ی F^+

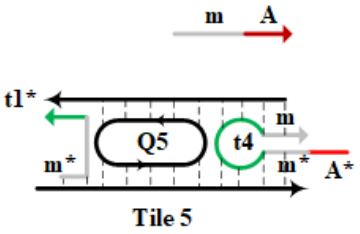
مسیر سوم و چهارم شامل گره‌های Q5، Q6، Q7، Q8، Q9، Q10 و F^- می‌باشند. این دو مسیر برای پیاده‌سازی دوگان تابع F یا همان $\overline{F(x)} = (A + \bar{B}).(\bar{A} + \bar{C})$ استفاده می‌شود. گره‌های Q5 و Q6 به صورت موازی باهم که برای پیاده‌سازی $(A + \bar{B})$ به کار می‌رود. بنابراین پیاده‌سازی موازی گره‌ها معنای عملیات OR از عملیات‌های بولی را می‌دهد. گره‌ی Q8 و Q9 هم به صورت موازی با همدیگر هستند که پیاده‌سازی $(\bar{A} + \bar{C})$ را برعهده دارند. اتصال این دو قسمت موازی توسط گره‌های Q7 و Q10 انجام می‌شود که عملیات AND بولی این دو پرانتز را برعهده دارند. اگر این اتصالات به خوبی برقرار شده باشند خروجی دوگان تابع F در گره‌ی F^- تولید خواهد شد. در ادامه پیاده‌سازی کاشی‌های هر کدام از این گره‌ها شرح داده خواهد شد.



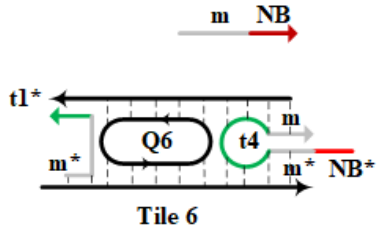
شکل12: پیاده‌سازی مسیر اول با استفاده از tile0، tile1، tile2 و tile11

گره‌ی Q5 با طراحی tile5 پیاده‌سازی شده که در شکل13 دیده می‌شود. بازوی سمت چپ آن مخصوص اتصال به tile0 متعلق به گره‌یseed است، طراحی گردیده است و بازوی سمت راست آن برای دریافت ورودی A طراحی شده است. زمانی که A با A* واکنش بدهد قسمت سنجاق‌سر بازوی سمت راست باز شده و آغازگر t4 آشکار می‌گردد.

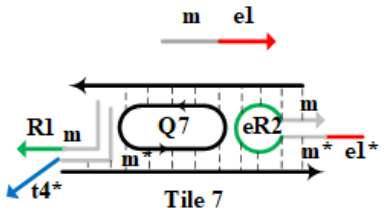
گره‌ی Q6 نیز با طراحی tile6 پیاده‌سازی شده که در شکل14 دیده می‌شود. این کاشی دارای دو بازوی سمت راست و چپ می‌باشد. بازوی سمت چپ آن همانند tile7 برای اتصال به tile0 از گره‌ی seed طراحی شده است و بازوی سمت راست آن برای دریافت ورودی $\bar{B}$ از پرانتز $(A+\bar{B})$ می‌باشد. زمانی که آغازگر NB با NB* واکنش دهد قسمت سنجاق‌سر آن باز شده و آغازگر t5 آشکار می‌گردد. گره‌ی Q5 و Q6 به صورت موازی پیاده‌سازی شده به این منظور که هرکدام از ورودی‌های A و $\bar{B}$ وجود داشته باشد مسیر از seed به Q7 برقرار گردد. همچنین تعبیه‌ی آغازگر t4 در بازوی سمت راست آن‌ها این اتصال را به بازوهای سمت چپ کاشی‌های گره‌ی Q7 را فراهم می‌کند.



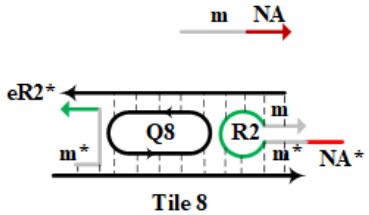
شکل13: ساختار tile5 پیاده‌سازی شده برای گره‌ی Q5.



شکل14: ساختار tile6 پیاده‌سازی شده برای گره‌ی Q6.



شکل15: ساختار tile7 پیاده‌سازی شده برای گره‌ی Q7.



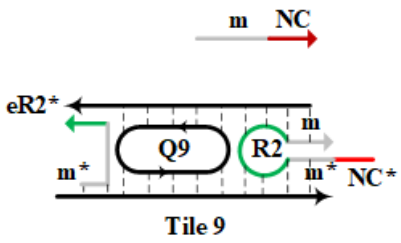
شکل16: ساختار tile8 پیاده‌سازی شده برای گره‌ی Q8.

گره‌ی Q7 با طراحی کاشی tile 7 که در شکل 15 نشان داده شده پیاده‌سازی می‌شود. Tile7 برای اتصال به بازوی سمت راست کاشی tile5 و tile6 طراحی شده است. آغازگر t4 از بازوی سمت راست کاشی‌های tile5 و tile6 با آغازگر t4* از بازوی سمت چپ کاشی tile7 واکنش می‌دهند و با بالا رفتن انرژی سیستم باعث جدا شدن دامنه‌ی m می‌شوند. با اتصال این کاشی

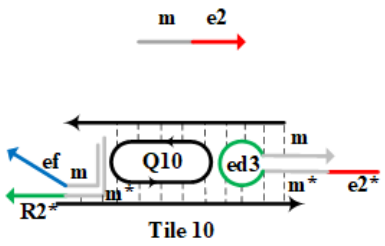
تک رشته <m R1^> آزاد می‌گردد که این تک رشته برای تولید نتیجه‌ی نهایی کاربرد دارد. رشته‌ی <m e1^> رشته‌ی فعال‌سازی است که باعث باز شدن بازوی سمت راست Tile7 می‌گردد. زمانی که این سنجاق‌سر باز شود، آغازگر eR2^ آشکار می‌شود و آمادگی اتصال به یکی از کاشی‌های مربوط به گره‌های Q8 وQ9 را پیدا می‌کند. رشته‌ی <e1^ m> همانند رشته‌ی enable به صورت پیش فرض در محیط آزمایش وجود دارد و این رشته به علت همگام بودن طراحی‌ها تعبیه شده است.

گره‌ی Q8 با طراحی کاشی tile8 پیاده‌سازی می‌شود که در شکل 16نشان داده شده است. بازوی سمت چپ آن برای اتصال به بازوی سمت راست tile7 طراحی گردیده است. بازوی سمت راست این کاشی برای دریافت ورودی $\bar{A}$ از پرانتز $(\bar{A}+\bar{C})$ می‌باشد. زمانی که آغازگر NA با *NA واکنش بدهد، سنجاق‌سر آن باز شده و آغازگر R2 آشکار می‌گردد که قابلیت اتصال به کاشی گره‌ی Q10 را پیدا می‌کند.

گره‌ی Q9 با طراحی دو کاشی tile9 پیاده‌سازی می‌شود که در شکل17 دیده می‌شود. بازوی سمت چپ Tile9 برای اتصال به کاشی tile7 از گره‌ی Q7 طراحی شده است. بازوی سمت راست این کاشی برای دریافت $\bar{C}$ از پرانتز $(\bar{A}+\bar{C})$ طراحی گردیده‌اند. زمانی که NC با *NC واکنش بدهد، آغازگر R2 آشکار می‌گردد و این کاشی‌ها آمادگی اتصال به کاشی گره‌ی Q10 را پیدا می‌کند.



شکل 17: ساختار tile9 پیاده‌سازی شده برای گره‌ی Q9



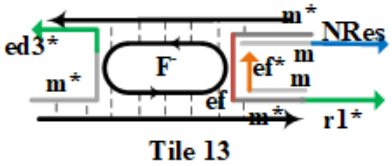
شکل 18: ساختار tile10 پیاده‌سازی شده برای گره‌ی Q10

گره‌ی Q10 با پیاده‌سازی tile10 طراحی می‌شود که در شکل18 دیده می‌شود. بازوی سمت چپ tile10 برای اتصال به یکی از کاشی‌های گره‌های Q8 و Q9 تعبیه شده است. زمانی که R2 با *R2 واکنش بدهد تک رشته‌ی <ef^ m> آزاد می‌گردد که در تولید نتیجه‌ی نهایی دخیل می‌باشد. تک رشته‌ی <m e2^> فعال‌ساز می‌باشد که باعث باز شدن رشته‌ی سنجاق‌سر بازوی سمت راست کاشی tile10 می‌شود و آغازگر ed3 آشکار می‌گردد. این رشته هم جهت همگام بودن پیاده‌سازی کاشی‌ها استفاده می‌شود.

گره‌ی F^- با طراحی کاشی tile13 که در شکل19 دیده می‌شود، پیاده‌سازی می‌گردد. اگر بازوی سمت راست tile10 فعال شده باشد، آغازگر ed3 با *ed3 از بازوی سمت چپ tile13 واکنش می‌دهد و اتصال این دو کاشی برقرار می‌شود. اگر طی مراحل قبل رشته‌های <ef^ m> و <r1^ m> آزاد شده باشد با رشته‌ی سمت راست tile13 واکنش می‌دهند و باعث آزاد شدن تک رشته‌ی <m Nres^> می‌شود که نمایانگر خروجی تابع $\bar{F}$ می‌باشد. آغازگر r1 از رشته‌ی <r1^ m> که در نتیجه‌ی اتصال یکی از کاشی‌های tile5 و tile6 با tile7 آزاد گردیده با *r1 از بازوی سمت راست کاشی tile13 واکنش می‌دهد و با بالارفتن انرژی سیستم دامنه‌ی m را نیز جدا می‌کند. در نتیجه آغازگر *ef آزاد می‌گردد. آغازگر ef از رشته‌ی <ef^ m> که حاصل اتصال tile8 یا tile9 به tile10 می‌باشد با *ef از بازوی سمت راست tile13 واکنش می‌دهد و دامنه‌ی m را جدا می‌کند. در نتیجه<m NRes^> به عنوان رشته‌ی خروجی دوگان تابع F آزاد می‌گردد.

با استفاده از پیاده‌سازی تابع بالا نشان داده شد که متدولوژی طراحی اتوماتیک برای ساخت موازی مدارهای منطقی به چه صورت عمل می‌کند. این متدولوژی قابلیت پیاده‌سازی هر تابع منطقی را دارد. لازم به ذکر است پیچیدگی زمانی آن نیز O(1)

می‌باشد که نسبت به تمامی مدل‌های موجود پیشرفت چشم‌گیری داشته است. همچنین این متدولوژی مشکلات روش خودسامانی را نداشته و توانسته تک رشته‌ی واحدی را در خروجی به نمایش بگذارد و از نظر کاهش تعداد tileها و قانونمند کردن آن‌ها نیز موفق عمل کرده است. در ادامه به توضیح نتایج این پیاده‌سازی‌ها برای تابع مثال زده شده پرداخته می‌شود.



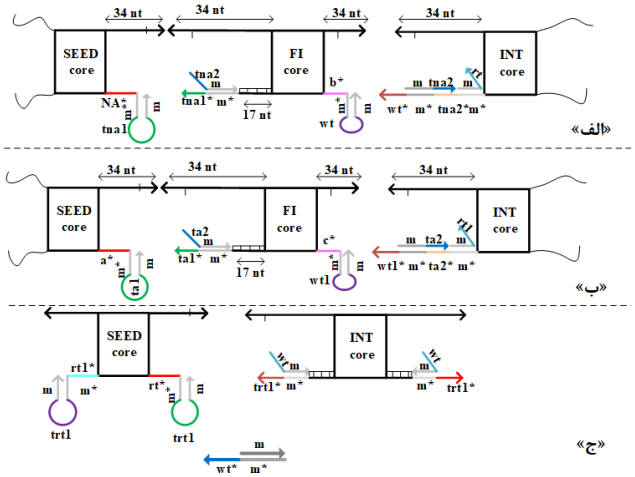
شکل 19: ساختار tile13 پیاده‌سازی شده برای گره‌ی F^-

2-3- پیاده‌سازی تابع موردنظر با استفاده از دروازه‌های منطقی AND و OR معرفی شده در [۲]

در این بخش متدولوژی پیاده‌سازی توابع منطقی با استفاده از دروازه‌های منطقی AND و OR پیاده‌سازی شده در [۲] و تکنیک دوخطی معرفی شده است. در[۲] با استفاده از تکنیک مخفی کردن آغازگرها در بازوهای کاشی‌ها دروازه‌های منطقی AND و OR به روش سازنده پیاده‌سازی شده است. همچنین اثبات شده که می‌توان آن‌ها را به صورت آبشاری در ساخت مدارهای بزرگتر استفاده کرد. در این بخش متدولوژی پیاده‌سازی توابع منطقی توسط این مدارهای منطقی نشان داده می‌شود. برای این کار تابع منطقی بحث شده در بخش قبل با این دروازه‌های منطقی پیاده‌سازی می‌شود. شکل6 تابع منطقی را به صورت توالی دروازه‌های AND-OR و OR-AND را نشان می‌دهد. با توجه به همین شکل می‌توان از دروازه‌های منطقی مورد نظر استفاده نمود. ساختار دروزاه‌ی منطقی AND و OR در [2] نشان داده شده است. این دروازه‌های منطقی به ترتیب دارای 3 و 2 کاشی می‌باشد. همچنین پیاده‌سازی آن‌ها با استفاده از شبیه‌ساز ISU-TAS[12] نشان داده شده است. که در این شبیه‌سازی‌ها قدرت اتصال و نوع کاشی‌ها مشخص می‌شود.

برای پیاده‌سازی شکل5 (الف) کافی است از دو عدد دروازه‌ی AND و یک دروازه‌ی OR استفاده شود. شکل20 ساختار کاشی‌های این دو دروازه‌ی منطقی AND و یک دروازه OR را نشان می‌دهد. اگر هر دو ورودی NA و B وجود داشته باشد شکل 20 (الف) تک رشته‌ی <m rt^> را تولید می‌کنند. ورودی <m NA^> با آغازگر *NA واکنش می‌دهد و باعث باز شدن رشته سنجاق‌سر بازوی سمت راست کاشی seed می‌شود و آغازگر tna1 آشکار می‌شود. همزمان با آشکار شدن آغازگر tna1 با آغازگر *tna1 از بازوی سمت چپ کاشی FI می‌تواند واکنش بدهد و باعث آزاد شدن تک رشته‌ی <tna2^ m> می‌شود. ورودی <m b^> هم همین روند برای بازوی سمت راست کاشی FI انجام می‌دهد و اتصال آن با بازوی سمت چپ کاشی INT برقرار می‌شود. اگر قبلا رشته‌ی <tna2^ m> آزاد شده باشد باعث آزاد شدن *tna2 شده و تک رشته‌ی <m rt^> به عنوان خروجی این ساختار AND آزاد می‌گردد. همین روال برای ورودی‌های A و C با استفاده از کاشی‌های شکل 20 (ب) صورت می‌گیرد و تک رشته‌ی <m rt1^> را به عنوان خروجی آزاد می‌کند. این دو تک رشته‌ی <m rt^> و <m rt1^> توسط کاشی‌های شکل 20 (ج) که متعلق به دروازه‌ی منطقی OR می‌باشد استفاده شده و خروجی دروازه‌ی منطقی OR ایجاد خواهد شد. ورودی <m rt^> با آغازگر *rt از بازوی سمت راست کاشی seed واکنش داده و به مرور زمان با بالارفتن انرژی سیستم باعث باز شدن ساقه ساختار سنجاق‌سر شده و آغازگر trt1 را آزاد می‌کند. اگر ورودی <m rt1^> نیز وجود داشته باشد با آغازگر *rt1 بازوی سمت چپ کاشی seed واکنش داده و باعث باز شدن سنجاق‌سر شده و آغازگر trt1 را آزاد می‌کند. این کاشی بعد از فعال شدن هر کدام از بازوهایش قادر به متصل شدن به کاشی INT خواهد شد و خروجی <m wt^> را آزاد

می‌کند. این خروجی برای اینکه در محدوده‌ی حاشیه نویز معرفی شده قرار بگیرد توسط رشته‌ی {wt^*}[m] ترشولد گرفته می‌شود. به صورت کلی این کاشی‌ها در شکل 20 ساختار تابع $f(x) = \bar{A}B + AC$ را پیاده‌سازی می‌کنند.



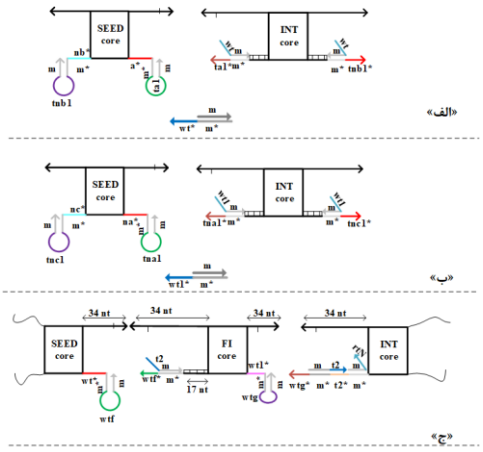
شکل20: ساختار کاشی‌های پیاده‌سازی شده برای شکل 5 (الف) توسط دروازه‌های منطقی AND-OR پیاده‌سازی شده در[۲]. (الف) ساختار AND برای ورودی‌های NA و B (ب) ساختار دروازه‌ی AND برای ورودی‌های A و C (ج) ساختار دروازه‌ی OR برای خروجی‌های قسمت‌های (الف) و (ب).

برای پیاده‌سازی شکل5(ب) از دو دروازه منطقی OR و یک دروازه منطقی AND استفاده می‌شود. ساختار این کاشی‌ها در شکل21 نشان داده شده است. این شکل ساختار $(\bar{A} + \bar{C})(A + \bar{B})$ را پیاده‌سازی می‌کند. دو کاشی بالای شکل 21 جهت پیاده‌سازی پرانتز $(A + \bar{B})$ می‌باشد هر کدام از ورودی‌های <m a^> ویا <m nb^> که وجود داشته باشند به ترتیب باعث بازشدن سنجاق‌سرهای بازوهای سمت راست و چپ کاشی seed می‌شوند. با آزاد شدن آغازگر مربوط به هر ورودی آن بازو می‌تواند با کاشی INT متصل شده و تک رشته‌ی <wt^ m> را آزاد کند. پس آزادسازی این تک رشته اگر غلظت خروجی از حد مجاز 200 nM برای صفر منطقی بالاتر رفته باشد از آن به وسیله‌ی رشته‌ی {wt^*}[m] ترشولد گرفته می‌شود. تا خروجی در حدود آستانه موردنظر قرار بگیرد. دو کاشی پایین شکل 21 نیز برای پیاده‌سازی پرانتز$(\bar{A} + \bar{C})$ می‌باشد. عملکرد این دو کاشی هم همانند قبلی‌ها می‌باشد با این تفاوت که برای ورودی‌های NAو NC این عملیات انجام می‌دهند. خروجی این دو دروازه منطقی OR به ترتیب تک رشته‌های <wt^ m> و <wt1^ m> هستند. این دو تک رشته به عنوان ورودی به دروازه‌ی منطقی AND که در آخرین سطر شکل 23 نشان داده شده در نظر گرفته می‌شوند. اگر هر دو این ورودی‌های وجود داشته باشند دروازه منطقی AND خروجی <m rtn^> را تولید می‌کند. عملکرد دروازه منطقی AND هم مانند دروازه AND توضیح داده شده در قسمت (الف) می‌باشد. بنابراین متدولوژی پیاده‌سازی توابع منطقی توسط دروازه‌های منطقی پیاده‌سازی شده در[۲] نیز به اثبات رسید. در بخش بعد به گزارش نتایج شبیه‌سازی آن‌ها پرداخته می‌شود.

بخش 4- نتایج متدولوژی معرفی شده

در این مقاله متدولوژی برای طراحی توابع جبری با کمک روش دنا خودسامانی ارائه شده است. برای طراحی این متدولوژی از ساختار کاشی‌های تغییر داده شده در [۲] استفاده شده است. در این مقاله نشان داده شده که هر تابعی با هر درجه از پیچیدگی را می‌توان با کمک tileهای طراحی شده و روش دوخطی پیاده‌سازی کرد. Tileها برای یک تابع نمونه طراحی شده و با استفاده از شبیه‌ساز visual DSD[۱۳] شبیه‌سازی شده است. این شبیه‌ساز در ساخت مدارهای مبتنی بر دنا

کاربرد فراوان دارد. پارامترهای مهم شبیه‌سازی و نرخ‌های واگرایی و اتصال در [2] نشان داده شده است. این شبیه‌ساز از مدل Deterministic استفاده کرده است.



شکل21: ساختار کاشی‌های پیاده‌سازی شده برای شکل5(ب) با استفاده از ساختار دروازه‌های منطقی AND و OR در[۲]

در ادامه به مقایسه متدولوژی معرفی شده با متدولوژی‌های فیلترینگ و خودسامانی پرداخته می‌شود. همچنین نحوه‌ی استفاده از کاشی‌های ارائه شده در[۲] به صورت پیاده‌سازی گراف و دروازه‌های منطقی آن در این مقاله بررسی شده و از نظر پیچیدگی‌های ساخت و هزینه برای متدولوژی معرفی شده بررسی می‌گردد. همچنین نتایج شبیه‌سازی برای حالت‌های مختلف در دو روش ارائه شده بررسی می‌گردد و سرعت واکنش‌ها مورد تحلیل قرار می‌گیرد.

4-1- مقایسه متدولوژی ارائه شده با متدولوژی‌های فیلترینگ و خودسامانی

نیاز به پردازش موازی در دنیای امروز برکسی پوشیده نیست. روش‌های پردازشی متداول به علت وجود محدودیت‌هایی پاسخ‌گوی این نیاز نبوده‌اند به همین علت محققان جذب روش‌های جدیدی همچون پردازش‌های مولکولی از جمله پردازش مبتنی بر دنا شده‌اند. در این روش‌های پردازشی معرفی یک متدولوژی مناسب برای پیاده‌سازی توابع به صورت موازی یکی از چالش‌های اساسی بوده است. آدلمن با ارائه روش فیلترینگ بنیان‌گذار محاسبات موازی با استفاده از رشته‌های دنا بود [۵]. این روش بعدها آغازی برای طراحی متدولوژی ساخت مدارها با روش فیلترینگ شد. اما این روش نیازمند دخالت مسئول آزمایشگاه در هر مرحله از آزمایش‌ها بود. این مراحل بایستی با اضافه کردن مدار خاص شیمیایی یا عملیات‌های دیگر انجام می‌گردید. همین امر سبب کاهش سرعت انجام محاسبات به صورت موازی گردید. با توجه به این که این روش نیازمند مواد، آنزیم‌ها و ابزارهای پرهزینه است، هزینه ساخت این روش را به شدت افزایش می‌دهد. بعد از آن Winfree با ارائه روش خودسامانی توانست مدل خودسامانی را ارائه کند [۷]. در این مدل قطعات به صورت خودسامانی به همدیگر متصل می‌گردند و محاسبات را به صورت کاملا موازی جلو می‌برد. این روش نیازمند مداخله مسئول آزمایش در جریان انجام واکنش‌ها ندارد. همچنین نیازمند وجود آنزیم‌ها و مواد پرهزینه دیگر نیز نیست. با وجود تمام مزایای روش سازنده این روش کنترل‌پذیر نبوده و درصد خطای آن بالا می‌باشد. همچنین امکان گرفتن خروجی و ارسال ورودی به صورت تک رشته‌ی دنا وجود نداشت. خروجی این روش کریستال‌های بزرگی بود که امکان استفاده از آن‌ها در مدارهای دیگر وجود نداشت. در این مقاله با تغییر کاشی‌های ارائه شده در [۲] متدولوژی برای طراحی مدارهای منطقی در مقیاس بزرگ با استفاده از کاشی‌های ارائه شده است. این کاشی‌ها امکان دریافت ورودی به صورت تک رشته‌ی دنا و خروجی به صورت تک رشته‌ی دنا را دارند.

برای معرفی این متدولوژی از منطق دوخطی به دلیل نبود سازوکار مناسب برای پیاده‌سازی دروازه منطقی NOT استفاده شده است. مزایا و معایب این مدل در جدول 1 نشان داده شده است.

جدول1: مزایا و معایب مدل‌های مورد بررسی

مدل‌ها	هزینه ساخت	استفاده از آنزیم‌ها و مواد دیگر در طول آزمایش	لزوم دخالت متصدی آزمایشگاه در طول آزمایش	یکپارچگی و قانونمند بودن	امکان دریافت ورودی و تولید رشته خروجی
مدل فیلترینگ[۵]	بالا	√	√	×	×
مدل سازنده[۷]	متوسط	×	×	×	×
مدل سازنده کنترل-پذیر[۲]	متوسط	×	×	√	√

2-4- شبیه‌سازی متدولوژی ارائه شده با کاشی‌های کنترل‌پذیر[۲] به صورت پیاده‌سازی گراف

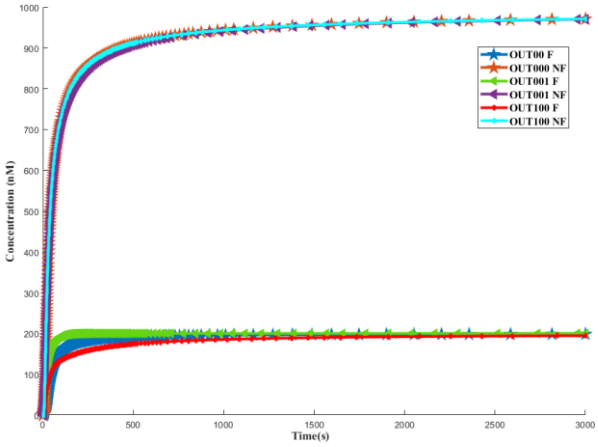
در این مقاله متدولوژی طراحی مدارهای منطقی با استفاده از کاشی‌های کنترل‌پذیر ارائه شده در [۲] معرفی گردید. این متدولوژی معرفی شده با استفاده از تکنیک دوخطی و برای یک تابع f و $\bar{f}$ طراحی شده است. در این طراحی گرافی در شکل4 برای تابع f و $\bar{f}$ در نظر گرفته شد. ساختار این تابع‌ها نیز در شکل5 (الف) و (ب) نشان داده شده است. طراحی این تابع‌ها با استفاده از این کاشی‌ها نشان دهنده‌ی توانایی پیاده‌سازی این متدولوژی برای تمامی توابع می‌باشد. در شبیه‌سازی غلظت ورودی‌ها با مقدار یک منطقی برابر 900nM و ورودی‌ها با مقدار صفر منطقی 100nM در نظر گرفته شدند. همچنین اگر ورودی یا مکمل ورودی در چند مسیر دخیل باشد به ازای هر مسیر غلظت معادل آن به غلظت اولیه اضافه می‌گردد. به عنوان مثال ورودی مکمل A در دو مسیر گراف شکل 4 شرکت کرده است بنابراین اگر ورودی برای این سیگنال یک منطقی در نظر گرفته شود غلظت معادل آن برابر 1800nM در نظر گرفته می‌شود و اگر صفر منطقی در نظر گرفته شود غلظت معادل آن 200nM در نظر گرفته می‌شود. دلیل این امر استفاده غلظت اولیه در هر دو مسیر می‌باشد. غلظت تمام کاشی‌ها برابر 1000nM در نظر گرفته شد. خروجی‌ها باید برای صفر منطقی در بازه‌ی 0nM تا 200nM و برای یک منطقی بین 800nM تا 1000nM قرار بگیرند. بازه‌ی 100nM تا 200nM حاشیه‌ی نویز صفر منطقی و 800nM تا 900nM حاشیه‌ی نویز یک منطقی می‌باشد. بنابراین اگر خروجی‌ها در این بازه‌ها قرار بگیرند صحیح می‌باشند.

شکل‌های 22و 23 خروجی حالت‌های متفاوت تابع f و $\bar{f}$ برای متدولوژی ارائه شده با استفاده از کاشی‌های کنترل‌پذیر [۲] و گراف پیاده‌سازی شده را نشان می‌دهد. در این شکل‌ها OUT نماد خروجی f و NOUT نماد خروجی $\bar{f}$ است. هر کدام از ورودی‌ها و مکمل آن‌ها در شکل‌ها با خطوط جداگانه نشان داده شده‌اند. محور yها غلظت رشته‌ها و محور xها زمان محاسبه را نشان می‌دهد.

شکل 22 نتیجه حالت‌های 000، 001و 100 را نشان می‌دهد. خطوط آبی و قهوه‌ایی با نشان‌ها ستاره به ترتیب نتایج تابع F و مکمل تابع F را نشان می‌دهد که ورودی‌ها همگی با غلظت 100nM و مکمل آن‌ها با غلظت 900nM در نظر گرفته شده‌اند. همانطور که مشاهده می‌شود خروجی تابع f کمتر از 200nM (در حاشیه نویز تنظیمی قرار دارد) می‌باشد. همچنین خروجی مکمل آن بر روی 900nM قرار دارد. همچنین خطوط سبز و بنفش با نشان‌های مثلث خروجی حالت 001 نشان می‌دهند که خط سبز برای خروجی تابع F و خط بنفش برای خروجی مکمل تابع F است. دو ورودی A و B با غلظت 100 nM و مکمل آن‌ها با غلظت 900 nM در نظر گرفته شده‌اند. ورودی C با غلظت 900 nM و مکمل آن با غلظت 100 nM در نظر گرفته

شده است. همانطور که مشاهده می‌شود خروجی تابع f کمتر از 200 nM (در حاشیه نویز تنظیمی قرار دارد) می‌باشد. همچنین خروجی مکمل آن بر روی 900 nM قرار دارد. همانطور که قبلا اشاره شد ورودی‌های NA و A به دلیل استفاده در دو مسیر جداگانه غلظت اولیه‌ی دو برابر دارند یعنی در این پیاده‌سازی غلظت NA برابر 1800 nM است و غلظت A برابر 200 nM است. خطوط قرمز و فیروزه‌ایی با نشان‌های مستطیل نشان دهنده‌ی به ترتیب خروجی‌های F و NF برای حالت 100 است که دو ورودی C و B با غلظت 100 nM و مکمل آن‌ها با غلظت 900 nM در نظر گرفته شده‌اند. ورودی A با غلظت 900 nM و مکمل آن با غلظت 100 nM در نظر گرفته شده است (به صورت دوبرابر در نظر گرفته شده است). همانطور که مشاهده می‌شود خروجی تابع f کمتر از 200 nM (در حاشیه نویز تنظیمی قرار دارد) می‌باشد. همچنین خروجی مکمل آن بر روی 900 nM قرار دارد.

شکل 23 خروجی حالت‌های 011و 111 نشان می‌دهد. ورودی A با غلظت 200 nM و مکمل آن با غلظت 1800 nM در نظر گرفته شده‌اند. ورودی‌های B و C با غلظت 900 nM و مکمل آن‌ها با غلظت 100 nM در نظر گرفته شده است. همانطور که مشاهده می‌شود خروجی تابع f بر روی 900 nM قرار دارد. همچنین خروجی مکمل آن کمتر از 200 nM (در حاشیه نویز تنظیمی قرار دارد) می‌باشد. خطوط آبی و نارنجی با نشان مثلث به ترتیب خروجی‌های تابع F و مکمل آن را برای حالت 011 نشان می‌دهند. خطوط سبز و بنفش بدون نشان خروجی تابع F و مکمل آن را برای حالت 111 نشان می‌دهند. ورودی‌ها با غلظت 900 nM و مکمل آن‌ها با غلظت 100 nM در نظر گرفته شده است (غلظت ورودی A و NA دو برابر شده‌اند). همانطور که مشاهده می‌شود خروجی تابع f بر روی 900 nM قرار دارد. همچنین خروجی مکمل آن کمتر از 200 nM (در حاشیه نویز تنظیمی قرار دارد) می‌باشد.



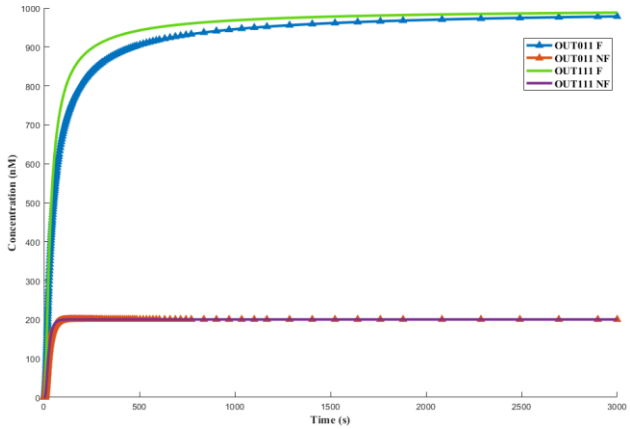
شکل 22: نتیجه شبیه‌سازی پیاده‌سازی متدولوژی موردنظر با استفاده از گراف برای ورودی‌های 000، 001و 100

جدول2 غلظت‌های دقیق خروجی‌های تابع f و $\bar{f}$ را برای ورودی‌های متفاوت نشان داده می‌شود. این نتایج صحت عملکرد متدولوژی پیاده‌سازی توابع موردنظر با استفاده از کاشی‌های پیاده‌سازی شده در[۲] را نشان می‌دهد.

3-4- شبیه‌سازی متدولوژی ارائه شده با کاشی‌های کنترل‌پذیر[۲] با استفاده از دروازه‌های منطقی AND و OR

در این مقاله متدولوژی طراحی مدارهای منطقی با استفاده از کاشی‌های کنترل‌پذیر ارائه شده در [۲] معرفی گردید. این متدولوژی معرفی شده با استفاده از تکنیک دوخطی و برای یک تابع f و $\bar{f}$ طراحی گردید. ساختار این تابع‌ها نیز در شکل5 (الف) و (ب) نشان داده شده است. طراحی این تابع‌ها با استفاده از این کاشی‌ها نماد توانایی پیاده‌سازی این متدولوژی برای تمامی توابع می‌باشد. در این بخش نتایج پیاده‌سازی با استفاده از دروازه‌های منطقی طراحی شده در[۲] بررسی می‌شود.

شکل 24 نتایج حالت‌های ورودی متفاوت را برای تابع F گزارش می‌کنند. تابع F با استفاده از دو عدد دروازه‌های منطقی AND و یک عدد دروازه‌ی منطقی OR پیاده‌سازی شده است. این تابع جبر $f(x) = \bar{A}B + AC$ را پیاده‌سازی می‌کند.



شکل 23: نتیجه شبیه‌سازی پیاده‌سازی متدولوژی موردنظر با استفاده از گراف برای ورودی‌های 011 و 111

جدول2: غلظت‌های خروجی‌ها به ازای ورودی‌های متفاوت برای پیاده‌سازی با استفاده از گراف

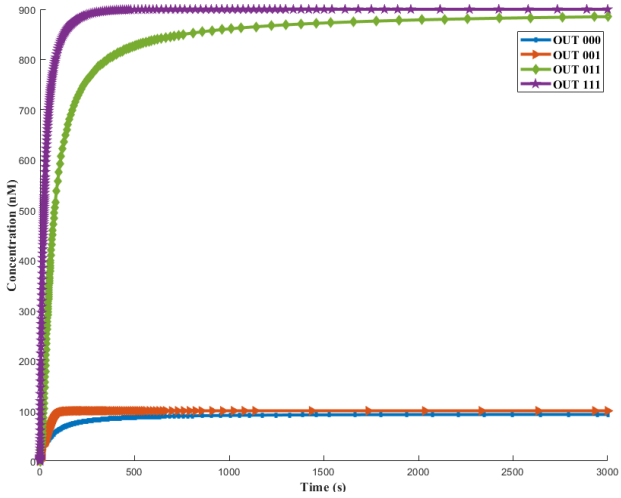
ورودی	A	B	C	NA	NB	NC	خروجیf	خروجی $\bar{f}$
000	200	100	100	1800	900	900	199.9	993.8
001	200	100	900	1800	900	100	200	993.8
011	200	900	900	1800	100	100	998.3	199.9
100	1800	100	100	200	900	900	199.6	993.8
111	1800	900	900	200	100	100	999.1	199.6

شکل 24 نتیجه‌های شبیه‌سازی حالت‌های 000، 001، 011 و 111 را برای تابع F گزارش می‌کند. همانطور که مشاهده می‌شود خروجی برای حالت 000 در حاشیه نویز و کمتر از 100 nM قرار گرفته است. غلظت‌های اولیه‌ی A، B و C برای این حالت معادل 100 nM می‌باشد. غلظت ورودی NA برابر 900 nM است. خط آبی رنگ با نشان‌های مستطیل در شکل 24 این خروجی را نمایش می‌دهد. خط نارنجی با نشان‌های مثلث در شکل 24 نتیجه شبیه‌سازی حالت 001 را برای تابع F گزارش می‌کند. همانطور که مشاهده می‌شود خروجی در حاشیه نویز و کمتر از 100 nM قرار گرفته است. غلظت ورودی A و B برابر 100 nM و غلظت NA و C معادل 900 nM قرار گرفته است. خط سبز رنگ با نشان‌های لوزی نتیجه شبیه‌سازی حالت 011 را برای تابع F گزارش می‌کند. همانطور که مشاهده می‌شود خروجی حدود 900 nM قرار گرفته است. غلظت ورودی‌های NA، C و B برابر 900 nM و غلظت A برابر 100 nM می‌باشد. خط بنفش با نشان‌های ستاره نتیجه شبیه‌سازی حالت 111 را برای تابع F گزارش می‌کند. همانطور که مشاهده می‌شود خروجی حدود 900 nM قرار گرفته است. غلظت ورودی‌های A، B و C برابر 900 nM قرار گرفته و غلظت ورودی NA بر روی 100 nM قرار گرفته است.

جدول 3 غلظت‌های دقیق نتایج بدست آمده برای متدولوژی پیاده‌سازی شده با استفاده از دروازه منطقی AND را گزارش می‌کند. در این جدول خروجی‌های تابع F نشان داده شده‌اند.

شکل‌های 25 نتایج حالت‌های ورودی متفاوت را برای تابع $\bar{F}$ گزارش می‌کنند. تابع $\bar{F}$ با استفاده از دو عدد دروازه‌های منطقی OR و یک دروازه‌ی منطقی AND پیاده‌سازی شده است. این تابع جبر $(\bar{A} + \bar{C})(A + \bar{B})$ را پیاده‌سازی می‌کند. شکل 25 نتایج شبیه‌سازی حالت‌های 000، 001، 011و 100 را برای تابع $\bar{F}$ گزارش می‌کند. خط آبی با نشان مستطیل خروجی حالت

000 را نشان می‌دهد. همانطور که مشاهده می‌شود خروجی در حاشیه نویز 800 nM تا 900 nM قرارگرفته است. غلظت‌های ورودی‌های NA، NB و NC برابر 900 nM و غلظت ورودی A برابر 100 nM قرار گرفته است. خط نارنجی نتیجه شبیه-سازی حالت 001 را برای تابع $\bar{F}$ گزارش می‌کند. همانطور که مشاهده می‌شود خروجی در حاشیه نویز 800 nM تا 900 nM قرارگرفته است. خط قرمز با نشان مثلث نتیجه شبیه‌سازی حالت 011 را برای تابع $\bar{F}$ گزارش می‌کند. همانطور که مشاهده می‌شود خروجی در حاشیه نویز 100 nM تا 200 nM قرارگرفته است. غلظت ورودی NA برابر 900 nM قرار می‌گیرد و غلظت ورودی‌های NB، NC و A برابر 100 nM قرار گرفته است. خط بنفش نتیجه شبیه‌سازی حالت 111 را برای تابع $\bar{F}$ گزارش می‌کند. همانطور که مشاهده می‌شود خروجی در حاشیه نویز 100 nM تا 200 nM قرارگرفته است. غلظت ورودی-های NA، NB و NC برابر 100 nM قرار گرفته و غلظت ورودی A برابر 900 nM قرار گرفته است.



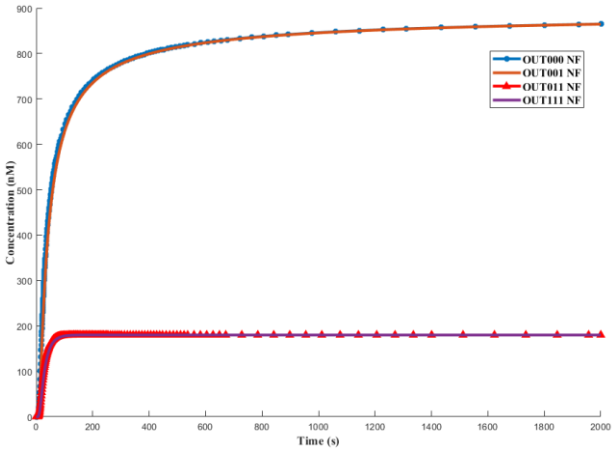
شکل 24: نتیجه شبیه‌سازی پیاده‌سازی متدولوژی موردنظر با استفاده از دروازه منطقی AND برای ورودی‌های 000، 001، 011 و 111

جدول3: غلظت‌های خروجی‌ها به ازای ورودی‌های متفاوت برای پیاده‌سازی با استفاده از دروازه منطقی AND

غلظت (nM) ورودی	A	B	C	NA	خروجیF
000	100	100	100	900	**92.8**
001	100	100	900	900	**100**
011	100	900	900	900	**890.7**
111	900	900	900	100	**899.9**
000	150	150	150	900	**189.2**
111	850	850	850	100	**849.9**

جدول4 غلظت‌های دقیق نتایج بدست آمده برای متدولوژی پیاده‌سازی شده با استفاده از دروازه منطقی OR و AND را گزارش می‌کند. در این جدول خروجی‌های تابع $\bar{F}$ نشان داده شده‌اند.

با توجه به نتایج گزارش شده در بخش‌های قبل متدولوژی طراحی مدارهای منطقی با استفاده از کاشی‌های بهبود یافته در [۲] هم به صورت پیاده‌سازی گراف و هم به صورت استفاده از دروازه‌های منطقی به خوبی قابل اجرا می‌باشد. این نتایج اثبات می-کند که این متدولوژی می‌تواند برای پیاده‌سازی مدارهای منطقی به صورت کاملاً موازی و نیمه موازی کارآمد باشد. این متدولوژی می‌تواند نقطه امیدی برای پیاده‌سازی مدارهای منطقی به صورت کاملاً موازی و با درجه پیچیدگی O(1) باشد.



شکل 25: نتیجه شبیه‌سازی پیاده‌سازی متدولوژی موردنظر با استفاده از دروازه‌های منطقی OR و AND برای ورودی 111

جدول4: غلظت‌های خروجی‌ها به ازای ورودی‌های متفاوت برای پیاده‌سازی با استفاده از دروازه منطقی OR و AND

غلظت (nM) ورودی	A	NA	NB	NC	خروجی $\bar{F}$
000	100	900	900	900	**879.5**
001	100	900	900	100	**879.4**
011	100	900	100	100	**179.9**
111	900	100	100	100	**179.9**
000	200	900	900	900	**858.5**
000	50	850	850	850	**806.7**

5-نتیجه‌گیری

امروزه انجام محاسبات به صورت موازی یکی از نیازهای اصلی بشریت می‌باشد. محاسبات مبتنی بر دنا به صورت ذاتی توان انجام محاسبات موازی بالایی را دارد. تاکنون روش‌های زیادی برای انجام محاسبات با استفاده از این روش ارائه شده است اما همچنان وجود متدولوژی طراحی یک‌پارچه و قوی برای طراحی مدارهای منطقی از چالش‌های اصلی این روش محاسباتی می‌باشد. متدولوژی‌های معرفی شده در گذشته مانند مدل فیلترینگ [۵] هر کدام دارای چالش‌هایی از جمله نیاز به مداخله در طول آزمایش، استفاده از مواد و آنزیم‌های پر هزینه دارند. متدولوژی سازنده [۷] بخشی از این چالش‌ها را برطرف نمود ولی همچنان امکان دریافت ورودی به صورت تک رشته‌ی دنا و ارسال تک رشته‌ی دنا به عنوان خروجی را نداشت و تنها تشخیص یک کریستال بزرگ نماد تکمیل محاسبات بود. این روش درصد خطای بالایی داشت و همچنین قابلیت استفاده به صورت آبشاری را نداشت. در[۲] کاشی‌های کنترل‌پذیر ارائه شد که این چالش‌ها را به صورت کامل مرتفع می‌نمود. با استفاده از این کاشی‌ها در این مقاله متدولوژی طراحی مدارهای منطقی ارائه گردید. با استفاده از این متدولوژی می‌توان تمام تابع‌های منطقی را به راحتی و به صورت کاملاً موازی تولید نمود. استفاده از این متدولوژی در محاسبات می‌تواند قدرت محاسبات موازی را بسیار بالا ببرد و هر تابعی را با O(1) انجام داد.

12. منابع

1. Dagdag, O., El Harfi, A., El Gana, L., S Safi, Z., Guo, L., Berisha, A., ... & El Gouri, M. (2021). Designing of phosphorous based highly functional dendrimeric macromolecular resin as an

effective coating material for carbon steel in NaCl: Computational and experimental studies. Journal of Applied Polymer Science, 138(2), 49673.

2. HassanAbadi, F. K., Reshadinezhad, M. R., Beiki, Z., & Dehghanian, F. (2023). Cascadable-controllable self-assembly DNA tiles for large-scale DNA logic circuits. IEEE Transactions on Biomedical Circuits and Systems.
3. Beiki, Z., & Jahanian, A. (2023). RTL2DNA: An automatic flow of large-scale DNA-based logic circuit design. Scientia Iranica, 30(4), 1279-1295.
4. Ceze, L., Nivala, J., & Strauss, K. (2019). Molecular digital data storage using DNA. Nature Reviews Genetics, 20(8), 456-466.
5. Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problems. science, 266(5187), 1021-1024.
6. Wang, H., & Wang, H. (1990). Dominoes and the AEA case of the decision problem. Computation, Logic, Philosophy: A Collection of Essays, 218-245.
7. Winfree, E. (1998). Algorithmic self-assembly of DNA. California Institute of Technology.
8. Qian, L., & Winfree, E. (2011). A simple DNA gate motif for synthesizing large-scale circuits. Journal of the Royal Society Interface, 8(62), 1281-1297.
9. Winfree, E. (2004, February). DNA computing by self-assembly. In 2003 NAE Symposium on Frontiers of Engineering (pp. 105-117).
10. Kumar, A. A. (2016). Fundamentals of digital circuits. PHI Learning Pvt. Ltd.
11. Evans, C. G., & Winfree, E. (2017). Physical principles for DNA tile self-assembly. Chemical Society Reviews, 46(12), 3808-3829.
12. Patitz, M. J. (2011). Simulation of self-assembly in the abstract tile assembly model with ISU TAS. arXiv preprint arXiv:1101.5151.
13. Lakin, M. R., Youssef, S., Polo, F., Emmott, S., & Phillips, A. (2011). Visual DSD: a design and analysis tool for DNA strand displacement systems. Bioinformatics, 27(22), 3211-3213.

Proposed methodology and DNA self-assembly tiles design to parallel computation

Fatemeh Kazemi Hassanabadi[1], Mohammad Reza Reshadinezhad[2], Zohreh Beiki[3]
Faculty of Computer Engineering, University of Isfahan, Isfahan,
fatemehkazemi@eng.ui.ac.ir[1], m.reshadinezhad@eng.ui.ac.ir[2], z.beiki@eng.ui.ac.ir[3]

Abstract—Molecular computing is an emerging Nanotechnology. Molecular computing is the base of designing and implementing molecular processors on a platform of biological macromolecules such as deoxyribonucleotide acid, ribonucleotide acid, and proteins [1]. DNA computing is a branch of molecular computing that uses DNA strands and chemical reactions for computation. DNA computing is the most successful molecular computing method in terms of scalability, simplicity of execution and implementation, design, and costs [2].
Recently, according to the features of the problem, many DNA computational models were proposed. The Self-assembly model consists of a lot of tiles that reacted parallel. this model has some advantages and disadvantages, for example, its massively parallel computing and regularity and vice versa uncontrollability tiles, high error rate, and large crystals on output (unscalable).
This article proposed a design method for the self-assembly logic functions. Also, this method has advantages such as scalability, automatic design flow, and low-cost implementation. The proposed methodology can be a promising starting point for performing computations in parallel.

Keywords: DNA-based computing, DNA self-assembly tile, Dual-Rail designing method, Computational methods, Parallel computation.