مقاله پژوهشی

تولید الگوی آزمون خودکار پیشرفته با استفاده از الگوریتم PSO-FAN

سید حمید ظهیری، احسان حق‌پرست و ابوالفضل بیجاری

چکیده: حوزه آزمایش مدارهای یکپارچه با مقیاس بسیار بزرگ (VLSI) در سال‌های اخیر پیشرفت‌های قابل توجهی را در روش‌های تشخیص خطا تجربه کرده است، به طوری که این پیشرفت‌ها در کاربردهای پیچیده اهمیت فراوانی یافته‌اند. هرچند که روش‌های سنتی برای شناسایی مدارهای سالم و معیوب کارآمد هستند، اما معمولاً در تشخیص دقیق انواع مختلف خطا محدودیت‌هایی دارند. این مقاله به بررسی پیاده‌سازی یک الگوریتم بهینه‌سازی شده به نام FAN (الگوریتم تولید آزمون مبتنی بر خروجی) می‌پردازد که با استفاده از تکنیک بهینه‌سازی ازدحام ذرات (PSO) بر روی مدارهای معیاری ISCAS'۸۹ انجام شده است. نتایج حاصل از این مطالعه نشان‌دهنده بهبود در پوشش خطا و افزایش دقت تشخیص می‌باشد. به علاوه، این بهینه‌سازی منجر به کاهش تعداد بردارهای آزمایشی مورد نیاز گردیده است که در نتیجه کارایی آزمایش را در شرایط تولید با حجم بالا افزایش می‌دهد. این تحقیق بر اهمیت توسعه معیارهای تشخیصی به منظور درک بهتر عیوب مدار تأکید می‌کند و به‌کارگیری تکنیک‌های طراحی که قابلیت آزمایش را تقویت کند، پیشنهاد می‌نماید. استفاده از PSO در الگوریتم FAN، نشان‌دهنده پتانسیل بالای این روش در فرآیند تولید آزمون است و موجب برقراری تعادلی مناسب بین دقت، پیچیدگی و کارایی عملیاتی می‌شود. این رویکرد نه تنها راهکارهای قوی‌تری را ارائه می‌دهد، بلکه تولید بردارهای آزمایشی با کیفیت بالا را نیز تسهیل می‌کند و در نهایت به بهبود تصمیم‌گیری در فعالیت‌های محاسباتی کمک می‌کند.

کلیدواژه: **الگوریتم FAN، الگوریتم های فراابتکاری، بهینه‌سازی، تولید خودکار الگوی آزمون، مدارات VLSI.**

۱- مقدمه

پیشرفت‌های سریع در فناوری‌های تولید نیمه‌هادی، تغییرات چشمگیری در تولید مدارهای مجتمع (IC) ایجاد کرده است. با کاهش اندازه‌ها و افزایش پیچیدگی طراحی‌ها، احتمال وقوع عیوب در این مدارها بیشتر شده است که می‌تواند تأثیر منفی جدی بر قابلیت اطمینان ICها داشته باشد. این وضعیت ضرورت به‌کارگیری پروتکل‌های آزمایش دقیق و سخت‌گیرانه را برای اطمینان از عملکرد صحیح دستگاه‌ها ایجاب می‌کند. از آن‌جا که روش‌های آزمایش سنتی معمولاً به افزایش تعداد بردارهای آزمایشی و مصرف بالای انرژی منجر می‌شوند، یافتن استراتژی‌های مؤثر در تشخیص عیوب به یک اولویت اصلی در این حوزه تبدیل شده است.

این مقاله در تاریخ ۱۳ دی ماه ۱۴۰۳ دریافت و در تاریخ ۱۳ خرداد ماه ۱۴۰۴ بازنگری شد.

سید حمید ظهیری (نویسنده مسئول)، دانشکده مهندسی برق و کامپیوتر، دانشگاه بیرجند، بیرجند، ایران، (email: hzahiri@birjand.ac.ir).

احسان حق‌پرست، دانشکده مهندسی برق و کامپیوتر، دانشگاه بیرجند، بیرجند، ایران، (email: ehsan_haghparast@birjand.ac.ir).

ابوالفضل بیجاری، دانشکده مهندسی برق و کامپیوتر، دانشگاه بیرجند، بیرجند، ایران، (email: a.bijari@birjand.ac.ir).

تولید الگوی آزمون خودکار (ATPG)[1] به‌ویژه در زمینه مدارهای دیجیتال ترتیبی به‌عنوان یک حوزه تحقیقاتی حیاتی شناخته می‌شود؛ زیرا این نوع مدارها به دلیل ویژگی‌های خاص خود، چالش‌های زیادی را ایجاد می‌کنند. تکنیک‌های مبتنی بر شبیه‌سازی، همچون الگوریتم‌های ژنتیک، به‌دلیل توانایی در ساده‌سازی فرآیند ATPG بدون کاهش کارایی، مورد توجه قرار گرفته‌اند. با این حال، اثربخشی این روش‌ها به ویژگی‌های توابع از پیش تعیین شده وابسته است. همچنین، در فضای آزمایش مدارهای مجتمع با مقیاس بسیار بزرگ (VLSI)[2]، شکاف‌های قابل توجهی در توانایی‌های محلی‌سازی خطاها مشاهده می‌شود. در حالی که آزمایش‌های تشخیصی می‌توانند دستگاه‌ها را به‌عنوان سالم یا معیوب طبقه‌بندی کنند، اغلب در شناسایی خطاهای خاص ناکام می‌مانند، که این موضوع محدودیت‌هایی را در کاربرد آن‌ها در محیط‌های پیچیده ایجاد می‌کند. با توجه به افزایش تقاضا برای محلی‌سازی دقیق عیوب، به‌ویژه در مراحل پیش‌تولید و ویژگی‌یابی، وجود معیارهای تشخیصی استاندارد ضروری به نظر می‌رسد. در این راستا، پیاده‌سازی تکنیک‌های طراحی برای آزمون‌پذیری (DFT)[3] می‌تواند افق‌های جدیدی را برای بهبود تشخیص خطا و کارآیی کلی فرآیند آزمایش فراهم کند. ترکیب الگوریتم‌های پیشرفته و استراتژی‌های نوآورانه در طراحی، به‌منظور رفع محدودیت‌های موجود در راه‌حل‌های تشخیصی کنونی، از اهمیت بالایی برخوردار است و در نهایت می‌تواند به ایجاد آزمایش‌های قابل اعتماد برای فناوری‌های نیمه‌هادی نسل بعدی منجر شود. در بخش‌های بعدی، به بررسی پیشینه تحقیقاتی موجود در این حوزه پرداخته خواهد شد.

در زمینه بهبود تشخیص خطاها در مدارهای دیجیتال، الگوریتم‌ها و تکنیک‌های مختلفی وجود دارند که به طور خاص بر روی بهینه‌سازی فرآیند آزمون تمرکز دارند. برای نمونه، GATEST [۱] با پیاده‌سازی یک تابع تناسب، تأثیرات خطاهایی که به فلیپ‌فلاپ‌ها می‌رسند را در اولویت قرار می‌دهد تا شانس تشخیص آن‌ها را افزایش دهد. همچنین، CRIS [۲] و GATTO [۳] در طراحی خود به فعالیت مدار توجه کرده و سعی می‌کنند با استفاده از اطلاعات مربوط به رفتار مدار، دقت بیشتری در تشخیص خطا داشته باشند. IGATE [۴] نیز از دنباله‌های تشخیصی استفاده می‌کند که می‌توانند خطاها را از فلیپ‌فلاپ‌ها به خروجی‌ها انتقال دهند. این روش به تنظیم، پاک کردن و توجیه شبه‌رجیسترها می‌پردازد تا وضعیت فعلی را توجیه کند. استفاده از الگوریتم‌های ژنتیک به عنوان یک رویکرد شناخته شده، هدفش ارتقاء نسل‌های "ژن‌های مفید" از طریق مکانیزم‌های طبیعی تناسب و ترکیب است. با این حال، شناسایی و به ارث بردن این "ژن‌ها" در مدارهای ترتیبی معمولاً به صورت تصادفی و نه بهینه انجام می‌شود. در طول فرآیند تکاملی، دنباله‌های "قوی" که

1. Automatic Test Pattern Generation
2. Very Large-Scale Integration
3. Design for Testability

توانایی بیشتری در تشخیص خطا دارند، در عین حال، دنباله‌های "ضعیف" که ممکن است اطلاعات مهمی درباره وضعیت‌ها و خطاها ارائه دهند، ممکن است نادیده گرفته شوند. این دنباله‌های "ضعیف" ممکن است خطاهایی را شناسایی کنند که در ارزیابی‌های بعدی قابل دسترسی نیستند. بنابراین، اجازه دادن به برخی از این دنباله‌های "ضعیف" برای ادامه حیات می‌تواند توانایی الگوریتم در پوشش خطا را افزایش دهد، موضوعی که در ادبیات موجود کمتر به آن پرداخته شده است. یک مشاهده دیگر برای افزایش پوشش خطا، ترکیب عناصر از دنباله‌های مختلف "ضعیف" است تا دنباله‌های جدیدی تولید شوند که می‌توانند خطاهایی را شناسایی کنند که در دنباله‌های قوی‌تر غایب هستند. با این حال، این تکنیک ممکن است به تولید دنباله‌های طولانی‌تر منجر شود که نیاز به فشرده‌سازی دارد تا کارایی آزمون را حفظ کند . به همین منظور، توسعه روش‌هایی برای تنظیم و فشرده‌سازی پویا دنباله‌های آزمایشی می‌تواند مهم باشد. استراتژی‌های فشرده‌سازی متعددی وجود دارند که می‌توانند به این منظور استفاده شوند، شامل روش‌های ایستا و پویا [۵] تا [۹]. به عنوان مثال، در [۱۰]، فشرده‌سازی ایستا بر روی دنباله‌های آزمایش تولیدشده با هدف‌گذاری خطاهای خاص اعمال شده است. این رویکردها می‌توانند به بهبود کارایی و قابلیت اطمینان آزمون‌های الکترونیکی کمک نمایند.

در زمینه فشرده‌سازی دنباله‌های آزمایشی و بهبود تشخیص خطا، چندین رویکرد و تکنیک متنوع وجود دارد که در تحقیقات مختلف معرفی شده‌اند .برای مثال، در [۷]، یک الگوریتم ژنتیکی طراحی شده که به فشرده‌سازی ایستا دنباله‌های آزمایشی می‌پردازد. این الگوریتم با سازماندهی مجدد دنباله‌ها، به کوتاه‌تر کردن برخی از آن‌ها و حذف کامل برخی دیگر، سطح پیچیدگی را کاهش می‌دهد و کارایی را بهبود می‌بخشد . در [۱۱]، یک روش جدید تحت عنوان انتخاب بردار معرفی شده است که به استخراج زیر دنباله‌های خاص از یک دنباله آزمایشی واحد (T) مربوط می‌شود. این زیر دنباله‌ها به نحوی انتخاب می‌شوند که بتوانند تمامی خطاها را شناسایی کنند. سپس با استفاده از یک تکنیک پوششی، یک زیرمجموعه حداقلی از دنباله‌ها شناسایی می‌شود که قادر به تشخیص تمامی خطاها هستند، در حالی که ترتیب زیر دنباله‌ها به همان ترتیبی که در مجموعه دنباله اصلی قرار داشته، حفظ می‌شود. کاجی‌هارا و همکاران [۱۲] و [۱۳]، رویکردی موازی را معرفی کرده‌اند که در آن چندین بردار آزمایش تولید می‌شود. این تحلیل ادامه می‌یابد تا زمانی که تمامی خطاها شناسایی شوند یا محدودیت خاصی برآورده گردد. با این حال، این فرآیند می‌تواند به ویژه برای مدارهای بزرگ زمان‌بر باشد. همچنین، اگرلال و همکاران [۱۴] برخی از خطاها را به وسیله تولید بردار آزمایشی شناسایی کرده و برای بقیه از یک تکنیک شاخه و محدود استفاده کردند که ممکن است به افزایش زمان پردازش منجر شود . ترکیب الگوریتم‌های ژنتیکی و تولید الگوی آزمایشی خودکار (ATPG) نیز در مقاله [۱۵] مورد بحث قرار گرفته است. این روش شامل انتخاب مکرر، ترکیب و فرآیندهای جهش برای یافتن راه‌حل‌های بهینه است که نیازمند ارزیابی تکنیک‌های مختلف الگوهای آزمون و مقایسه آن‌ها می‌باشد. این رویکردها و تکنیک‌ها در مجموع، در تلاش برای بهبود کارایی و کاهش زمان پردازش در تشخیص خطاها و فشرده‌سازی دنباله‌های آزمایشی طراحی شده‌اند و به بهبود عملکرد مدارهای دیجیتال کمک می‌کنند.

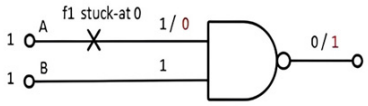
در قرن گذشته، استفاده از هوش مصنوعی به ویژه در زمینه استخراج ویژگی‌ها و تشخیص نقص‌ها به طرز چشمگیری رشد کرده است. این تکنیک‌ها به خصوص در مدارهای الکترونیکی و سیستم‌های پیچیده کاربرد دارند و به شناسایی نقص‌ها کمک می‌کنند. به عنوان مثال، استفاده از ماشین‌های بردار پشتیبان (SVM)[1] برای تشخیص نقص در [۱۶] و [۱۷] مطرح شده است، اگرچه این روش ممکن است به دلیل بروز مشکلاتی مانند حداقل‌های محلی و غیرخطی بودن، با چالش‌هایی مواجه شود. در مطالعاتی دیگر، استفاده از ماشین‌های بردار مرتبط چندکلاسه و جنگل‌های تصادفی [۱۸] و [۱۹] مورد بررسی قرار گرفته است. این رویکردها اگرچه می‌توانند نتایج خوبی ارائه دهند، اما ممکن است زمان زیادی برای رسیدن به نتایج مطلوب نیاز داشته باشند. شبکه‌های عصبی مصنوعی که شامل نورون‌های به‌هم‌پیوسته هستند و توابع پیچیده را از طریق تبدیلات غیرخطی یاد می‌گیرند، نیز در تشخیص نقص به کار رفته‌اند [۲۰]. این شبکه‌ها برای وظایف پیچیده‌ای همچون شناسایی گفتار [۲۱] بسیار مؤثر هستند و می‌توانند برای تشخیص نقص‌ها نیز به کار گرفته شوند [۲۲] و [۲۳]. با این حال، بسیاری از پژوهش‌ها بر روی شبکه‌های عصبی کم‌عمق تمرکز دارند و هنوز هم در مراحل ابتدایی کاربرد خود قرار دارند. یادگیری عمیق به عنوان یک تکنیک نوین برای استخراج خودکار ویژگی‌ها و تشخیص نقص‌ها مطرح شده است. این تکنیک می‌تواند ویژگی‌های سطح بالا را از مجموعه‌های بزرگ داده استخراج کند و معمولاً از روش‌های سنتی کارآمدتر است، که نیاز به طراحی دستی ویژگی‌ها دارند [۲۴] تا [۲۶]. تکنیک‌های ریاضی نیز می‌توانند در اطمینان از ایمنی و کیفیت خدمات در اتصالات نوت‌ها به کار روند [۲۷]. در [۲۸]، از شبکه‌های عصبی برای مدل‌سازی فرآیندهای تشخیص و تصحیح نقص‌ها استفاده شده است، و توجه به تلاش‌های آزمون که بر این فرآیندها تأثیر دارند، نشان‌دهنده اهمیت آن‌هاست. همچنین، برای حفظ حریم خصوصی پیام‌های منتقل‌شده در شبکه‌های سایبری-فیزیکی غیرمتمرکز، خدمات رمزنگاری به همراه الگوریتم‌های معتبر به کار گرفته می‌شوند تا امکان تشخیص نقص در برابر حملات تزریق نقص فراهم گردد [۲۹]. مدل خاص یادگیری عمیق به نام کدگذار اتوماتیک حفره‌ای انباشته (SSAE)[2] نیز می‌تواند مجموعه‌های ویژگی‌ها را یاد بگیرد و بدین ترتیب زمان و پیچیدگی محاسباتی را کاهش دهد، که نشان‌دهنده پیشرفت‌های اخیر در این زمینه است.

در ادامه، به بررسی مدل‌سازی خطاها پرداخته خواهد شد و به تحلیل جزئیات آزمایش VLSI و چالش‌های مرتبط با آن در زمینه الکترونیک مدرن خواهیم پرداخت. پس از این مرحله، روش‌های تولید آزمون و کارایی آن‌ها در شرایط دنیای واقعی ارزیابی خواهد شد. همچنین، در ادامه، بهینه‌سازی درخت‌های تصمیم با بهره‌گیری از الگوریتم FAN و بهینه‌سازی جمعیت ذرات (PSO)[3] بررسی خواهد شد. در نهایت، نتایج تجربی حاصل از این پژوهش ارائه خواهد شد و در ادامه، به بررسی پیامدهای این نتایج و جمع‌بندی یافته‌ها پرداخته می‌شود. همچنین پیشنهادهایی برای تحقیقات آینده ارائه خواهد گردید.

۲- مدل‌سازی خطاها

مدل خطا، به عنوان یک نمایه مهندسی، شرایطی را توصیف می‌کند که ممکن است در فرآیند تولید، توسعه یا عملکرد یک تجهیزات یا محصول به‌وجود آید. این مدل به طراح یا کاربر امکان می‌دهد تا به راحتی

1. Support Vector Machines
2. Stacked-Sparse-Autoencoder
3. Particle Swarm Optimization



شکل ۱: تصویر یک گیت NAND که در خطای چسبیده.

و بدون دردسر، پیامدهای ناشی از هر خطا خاص را پیش‌بینی کند. در سطح ساختاری، مدار معمولاً به شکل ۱ شماتیک نمایش داده می‌شود که بیشتر در سطح دروازه‌ها و فلیپ‌فلاپ‌ها وجود دارد. در این مدل، می‌توان چند فرض اصلی را مطرح کرد:

– بلوک‌ها (مانند دروازه‌ها) خود دچار خطا نیستند.

– اما ارتباطات بین بلوک‌ها ممکن است دچار مشکل شوند.

هدف اصلی این است که اطمینان حاصل شود این ارتباطات بدون خطا باقی مانده و قادر به انتقال سیگنال‌های منطقی ۰ و ۱ هستند. از مدل‌های خطا ساختاری مشهور می‌توان به مدل خطای چسبیدن[1]، مدل خطای پل-زدن[2]، و مدل خطای تاخیر[3] اشاره کرد.

۲-۱ مدل خطای چسبیدن

مدل رایج برای خطاهای منطقی، خطای «چسبیده» است. در اینجا فرض می‌کنیم که برخی از خطوط مدار به طور دائمی در حالت منطقی ۰ یا منطقی ۱ به دلیل بعضی خرابی‌ها ثابت مانده‌اند. خطاهای چسبیده به ۰ و ۱ اغلب به ترتیب به s-a-۰ و s-a-۱ خلاصه می‌شوند. خطاهای چسبیده را می‌توان بیشتر به دو بخش تقسیم کرد:

۲-۱-۱ خطاهای چسبیده تک

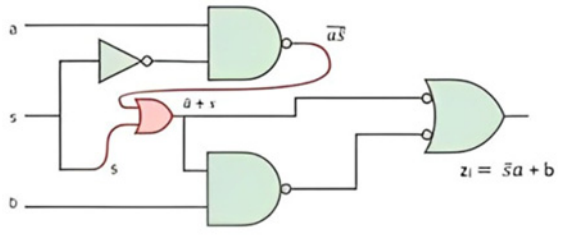
تنها یکی از سیم‌های مدار در هر زمان خاص دارای خطا چسبیده است. این مدل خطا، رایج‌ترین مدل در صنعت است. برای یک مدار با "n" سیم، تعداد کل خطاهای چسبیده تک برابر با "۲n" است.

۲-۱-۲ خطاهای چسبیده چندگانه

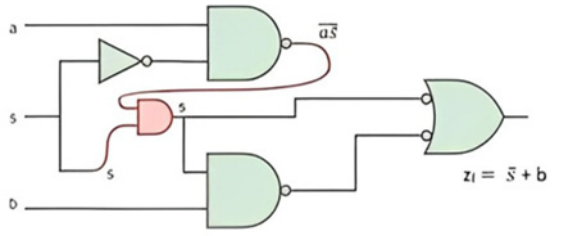
هر تعداد از سیم‌های مدار می‌توانند در هر زمان خاص دارای خطاهای چسبیده باشند. از آنجایی که برای هر سیم سه طبقه‌بندی کلی وجود دارد (خوب، خطای s-a-۱، خطای s-a-۰)، بنابراین برای یک مدار با "n" سیم، تعداد کل خطاهای چسبیده چندگانه برابر با "۳n–۱" است.

۲-۲ مدل خطای پل زدن

خطای پل‌زنی زمانی روی می‌دهد که دو یا چند خط سیگنال در یک مدار به‌طور تصادفی به هم متصل شوند. این مسأله به‌دلیل نقص در فرآیند تولید طراحی ممکن است اتفاق بیفتد. اگر یک المان به منبع تغذیه (VDD) یا زمین (VSS) متصل شود، خطای پل‌زنی در سطح دروازه به دو نوع تقسیم می‌شود: خطای پل‌زنی ورودی و خطای پل‌زنی بازخورد. خطای پل‌زنی ورودی به معنای اتصال کوتاه تعدادی از خطوط ورودی اولیه است. از طرف دیگر، خطای پل‌زنی بازخورد زمانی رخ می‌دهد که بین یک خط خروجی و یک خط ورودی اتصال کوتاه وجود داشته باشد. این نوع خطا می‌تواند باعث نوسان مدار شود یا آن را به یک مدار ترتیبی تبدیل کند. خطاهای پل‌زنی در یک مدار با سطح ترانزیستوری ممکن است بین پایانه‌های یک ترانزیستور یا بین دو یا چند خط سیگنال رخ‌دهند.



(الف)



(اب)

شکل ۲: تصویر خطای پل زدن به شیوه (الف) AND سیمی و (ب) OR سیمی.

دهند. این نوع خطاها را می‌توان مطابق شکل ۲ به دو بخش تقسیم‌بندی کرد: پل‌زنی به‌شیوه AND سیمی و پل‌زنی به‌شیوه OR سیمی.

۲-۳ مدل خطای تاخیر

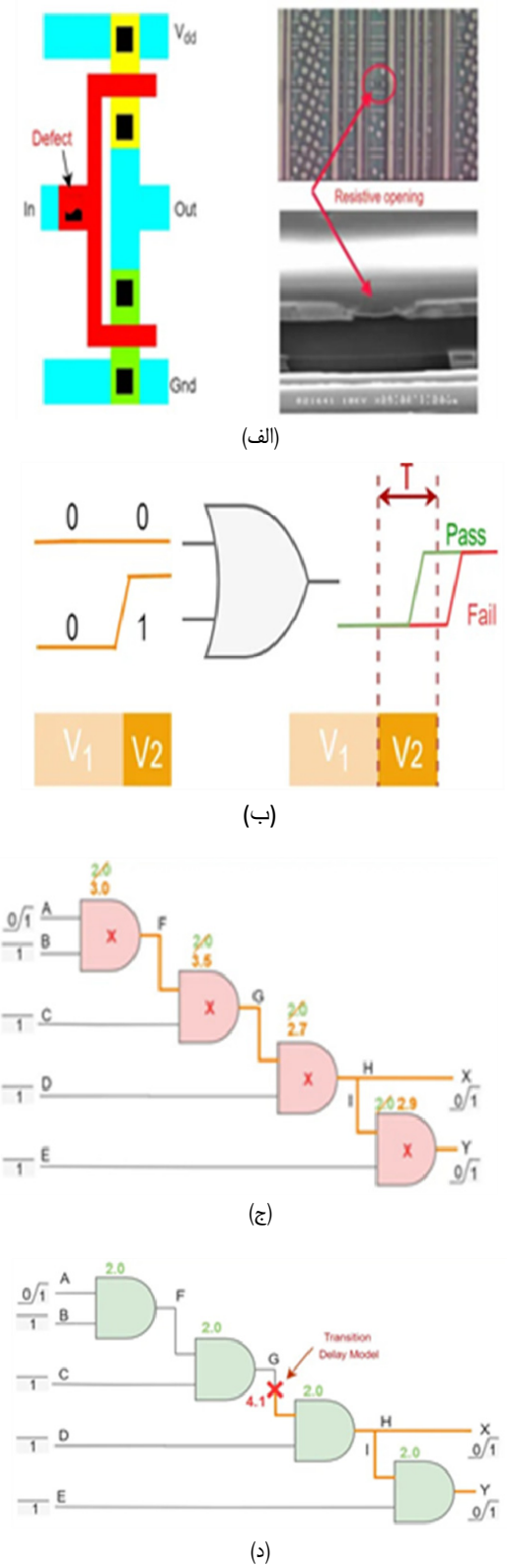
خرابی‌های تأخیری زمانی رخ می‌دهند که یک پین از یک دروازه به یک انتقال خاص در یک مجموعه خاص از محرک‌ها به‌طور بسیار کند پاسخ می‌دهد (شکل ۳).

– عیوب تصادفی: بازهای مقاومتی و برقراری مقاومتی. به عنوان مثال، تماس ضعیف بین دو دروازه. علاوه بر تأخیر دروازه‌ها، خروجی ممکن است تحت تأثیر تأخیر بین اتصالات (نازک شدن سیم) نیز قرار گیرد. به عنوان مثال، یک مشکل در فرایند باعث شده است که ویا (حفره) دروازه معکوس کننده به‌طور کافی پر نشود. بنابراین ورودی دارای چندین اهم مقاومت است. از آنجا که دروازه‌های CMOS مقدار قابل توجهی ظرفیت دارند، ثابت زمان RC ورودی بسیار بیشتر از آنچه انتظار می‌رفت، است که منجر به یک انتقال کندتر می‌شود. این نقص می‌تواند به‌عنوان برقراری مقاومتی مدل‌سازی شود.

– عیوب سیستماتیک: نویز متقاطع و تغییرات فرایندی در .Vt به‌عنوان مثال، اگر ما یک انتقال صعودی و یک انتقال نزولی در دو سیم بسیار نزدیک به هم داشته باشیم، برقراری خازنی بین آن‌ها ممکن است منجر به یک انتقال کندتر به خروجی شود. توجه داشته باشید که خرابی‌های تأخیری زمان‌بندی مدار را تغییر می‌دهند ولی عملکرد آن را نه. یک خرابی تأخیری باعث می‌شود که یک مدار با سرعت مشخصی دچار خطا شود اما ممکن است خروجی صحیحی در سرعتی کندتر تولید کند. بر خلاف خرابی‌های گیر کرده، خرابی تأخیری نیاز به یک آزمون دو الگو یا مجموعه‌ای از دو بردار آزمون دارد.

- V۱: وضعیت مدار مقداردهی اولیه شود؛ الگوی آزمون ۰۰ را برای دروازه OR اعمال گردد.
- V۲: انتقال آغاز شود و اثر خرابی به خروجی گسترش داده شود؛ الگوی آزمون ۰۱ برای دروازه OR اعمال گردد.

1. Stuck-at Fault Model
2. Bridging Fault model
3. Delay Fault Model



شکل ۳: تصویر خطای تاخیر. (الف) خطای تصادفی، (ب) خطای سیستماتیک، (ج) مدل خطای تاخیر مسیر، و (د) مدل خطای تاخیر انتقال.

زمان تأخیر واقعی با در نظر گرفتن تأخیر انتشار دروازه OR محاسبه می‌شود. بدترین حالت تأخیر 'T' توسط آزمون‌گر تعیین می‌شود و الگوی آزمون اعمال می‌شود. بردار 'V2' خروجی را به منطق۱ تغییر می‌دهد. اگر خروجی قبل از زمان مقرر 'T' نشان داده شود، آزمون قبول می‌شود و در غیر این صورت رد می‌شود. دو مدل اصلی خرابی تأخیری عبارتند از: مدل خطای تأخیر مسیر (PDF) و مدل خطای تاخیر انتقال (TDF).

خطاهای تأخیری در آزمون‌های چسبیده عبور می‌کنند. از آنجا که خطاهای s-a-۰ و s-a-۱ به‌عنوان موارد افراطی خطاهای کند به صعودی و آهسته به نزول در نظر گرفته می‌شوند، بنابراین می‌توان گفت که خطاهای تأخیری به‌عنوان یک فرامجموعه از خطاهای چسبیده محسوب می‌شوند. به این معنا که اگر یک مدار را برای خطاهای تأخیری آزمون کنیم، خطاهای چسبیده به‌صورت خودکار آزمون خواهند شد. به همین دلیل، مدل خطاهای تأخیری به‌مراتب بهتر و برتر از خطاهای چسبیده است.

۳- آزمون VLSI

آزمون‌گیری فرآیندی است که برای تأیید اینکه مدار VLSI تولیدشده به صورت مورد انتظار یا طراحی‌شده عمل می‌کند، استفاده می‌شود. مدار یا دستگاهی که مورد آزمایش قرار می‌گیرد (CUT/DUT)[1] تحت تأثیر سیگنال‌های آزمایشی در ورودی‌های خود قرار می‌گیرد. سپس پاسخ‌های خروجی نظارت شده و با نتایج مورد انتظار مقایسه می‌شوند (شکل ۴). اگر خروجی مشاهده‌شده با پاسخ صحیح ذخیره‌شده مطابقت داشته باشد، CUT به عنوان عملیاتی در نظر گرفته می‌شود؛ در غیر این صورت، به عنوان معیوب طبقه‌بندی می‌شود. داده‌های ورودی به CUT به عنوان الگوی آزمون یا وکتور آزمون شناخته می‌شوند. به طور کلی، آزمون‌ها در این مرحله بر روی شناسایی خطا تمرکز دارند. هدف آزمون شناسایی این است که مشخص شود CUT معیوب است یا بدون عیب. با این حال، در آزمون‌های تشخیصی، الگوهای آزمون اضافی ممکن است برای تعیین علت یا محل خرابی استفاده شوند. برای مدارهای ترکیبی با ورودی‌های اصلی (PI) و خروجی‌های اصلی (PO) متعدد، پاسخ‌های خروجی بلافاصله بعد از اعمال وکتورهای ورودی (صفر و یک منطقی) در PIها، با در نظر گرفتن تأخیرهای گیت، قابل مشاهده است. انواع مختلفی از آزمون وجود دارد:

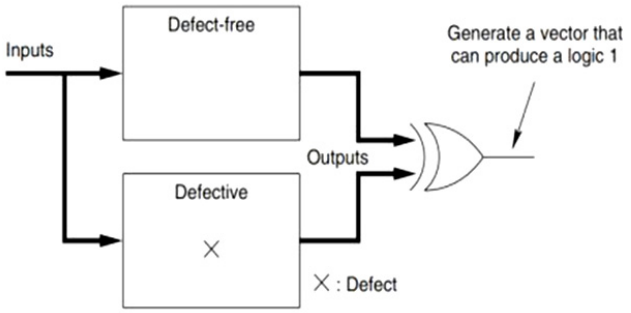
– آزمون اعتبارسنجی، آزمون ویژگی یا دیباگ طراحی:
- دقت طراحی و صحت فرآیند آزمون را تضمین می‌کند که ممکن است نیاز به تنظیمات در یکی یا هر دو مورد داشته باشد.

– آزمون تولید:
- شامل آزمایش همه چیپ‌های تولیدشده برای عیوب پارامتریک و منطقی و همچنین تطابق با مشخصات آنالوگ است.
- شامل آزمون‌های سوختگی یا فشار نیز می‌شود.

– آزمون پذیرش (بازرسی ورودی):
- توسط کاربران (مشتریان) برای تأیید کیفیت قطعات خریداری‌شده انجام می‌شود.

تولید آزمون شامل ایجاد الگوهای مؤثر برای دستیابی به پوشش خطای بالا در تراشه‌ها است که ممکن است به دلیل طراحی یا تولید ناقص دچار نقص شوند. هدف اصلی تولید الگوهای آزمون است که قادر به شناسایی خطاها در یک تراشه باشند و آن را از نسخه بدون نقص متمایز کنند. تولید الگوی آزمون خودکار (ATPG) نقش حیاتی در بهینه‌سازی منطقی، تأیید صحت، طراحی برای آزمون‌پذیری (DFT) و آزمایش‌های درون‌ساخت دارد. با پیشرفت فناوری و افزایش پیچیدگی داده‌ها، تولید الگوهای آزمون ترکیبی به طور مؤثر ضروری می‌شود. معمولاً طراحان مدار تنها ۷۰ تا ۷۵ درصد از کل نقص‌ها را پوشش می‌دهند که نشان‌دهنده نیاز به الگوریتم‌های ATPG قوی است. سیستم‌های ATPG

1. Circuit Under Test/ Device Under Test



شکل ۴: نمای مفهومی تولید آزمون.

با یک مدل نقص شروع می‌شوند تا عیوب را شناسایی کنند و اغلب از مدل نقص "مکث در یک وضعیت" [1](SAF) استفاده می‌کنند که خطاها را به دو دسته "مکث در ۰" و "مکث در ۱" طبقه‌بندی می‌کند. با وجود سادگی آن، محدودیت‌های SAF در شناسایی خطاها در مدارهای CMOS نشان‌دهنده نیاز به بهبود است. ATPG ترکیبی بخش اساسی از آزمایش‌های مدرن است، به‌ویژه با زنجیرهای اسکن که امکان آزمایش توالی را فراهم می‌آورد. در فرآیندهای ATPG، تصمیمات کلیدی شامل انتخاب مسائل حل نشده و راه‌حل‌های مؤثر است که به چالش‌های کنترل‌پذیری و مشاهده‌پذیری وابسته است. در حالی که هدف تولید الگوهای آزمون با کیفیت بالا با مجموعه‌های حداقلی است، دستیابی به پوشش کامل خطاها به‌ویژه چالش‌برانگیز است. پیشرفت‌های مداوم در روش‌های ATPG برای مقابله با پیچیدگی رو به افزایش مدارهای مجتمع ضروری است.

چندین الگوریتم برای تولید آزمون پیشنهاد شده‌اند، که سه مورد به‌خصوص برای مدارهای منطقی مؤثر هستند: الگوریتم D [31]، PODEM[2] [۳۲] و FAN[3] [۳۳]. الگوریتم D که توسط راث در سال ۱۹۶۶ توسعه‌یافته، پایه‌ای است اما به دلیل ناکارآمدی در تولید آزمون مورد انتقاد قرار گرفته است. PODEM که توسط گویل در سال ۱۹۸۱ ایجاد شده، سرعت را بهبود می‌بخشد اما هنوز با مشکلات برگشت‌پذیری مواجه است. الگوریتم FAN با اطمینان از اینکه اگر آزمونی وجود داشته باشد، قابل تولید باشد، این محدودیت‌ها را برطرف می‌کند.

٤- الگوریتم FAN

در این بخش، عملکرد و مزایای الگوریتم FAN در تولید آزمون تحلیل خواهد شد. نمودار جریان الگوریتم FAN در شکل ۵ ارائه شده است.

٤-١ چندین بازگشت و تخصیص ورودی

۱. اهداف اولیه: این فرایند با تعیین اهداف اولیه شروع می‌شود، مانند انتشار یک سیگنال معیوب یا توجیه خطوط غیرموجه. این اهداف به عنوان راهنما برای اقدام‌های بعدی در چارچوب تصمیم‌گیری عمل می‌کنند.
۲. شناسایی اهداف نهایی: از طریق چندین بازگشت، الگوریتم اهداف نهایی را شناسایی می‌کند. از بین این اهداف، یک هدف مشخص انتخاب می‌شود که (V, L) نام گذاری می گردد. در اینجا تخصیص مقدار (V) به خط (L) مشخص می‌شود که احتمال زیادی برای پیشبرد به سمت اهداف اولیه دارد.
۳. تخصیص مقادیر: الگوریتم مقدار انتخاب‌شده (V) را به خط (L) تخصیص می‌دهد و سپس این مقدار برای نتیجه‌گیری‌ها مورد استفاده قرار می‌گیرد. اهداف نهایی باقی‌مانده برای نتیجه‌گیری‌های بیشتر محفوظ می‌مانند، تا زمانی که موثر باقی بمانند.
۴. اثربخشی اهداف: اهداف نهایی باقی‌مانده تحت شرایط خاصی غیرموثر می‌شوند:
- اگر هدف اولیه انتشار یک سیگنال معیوب بوده باشد، و مرز نشان‌دهنده آن خطا بعد از نتیجه‌گیری تغییر کرده باشد.
- اگر هدف توجیه خطوط غیرموجه بوده باشد و تمام این خطوط توجیه شده باشند.

٤-٢ فرآیند بازگشت به عقب

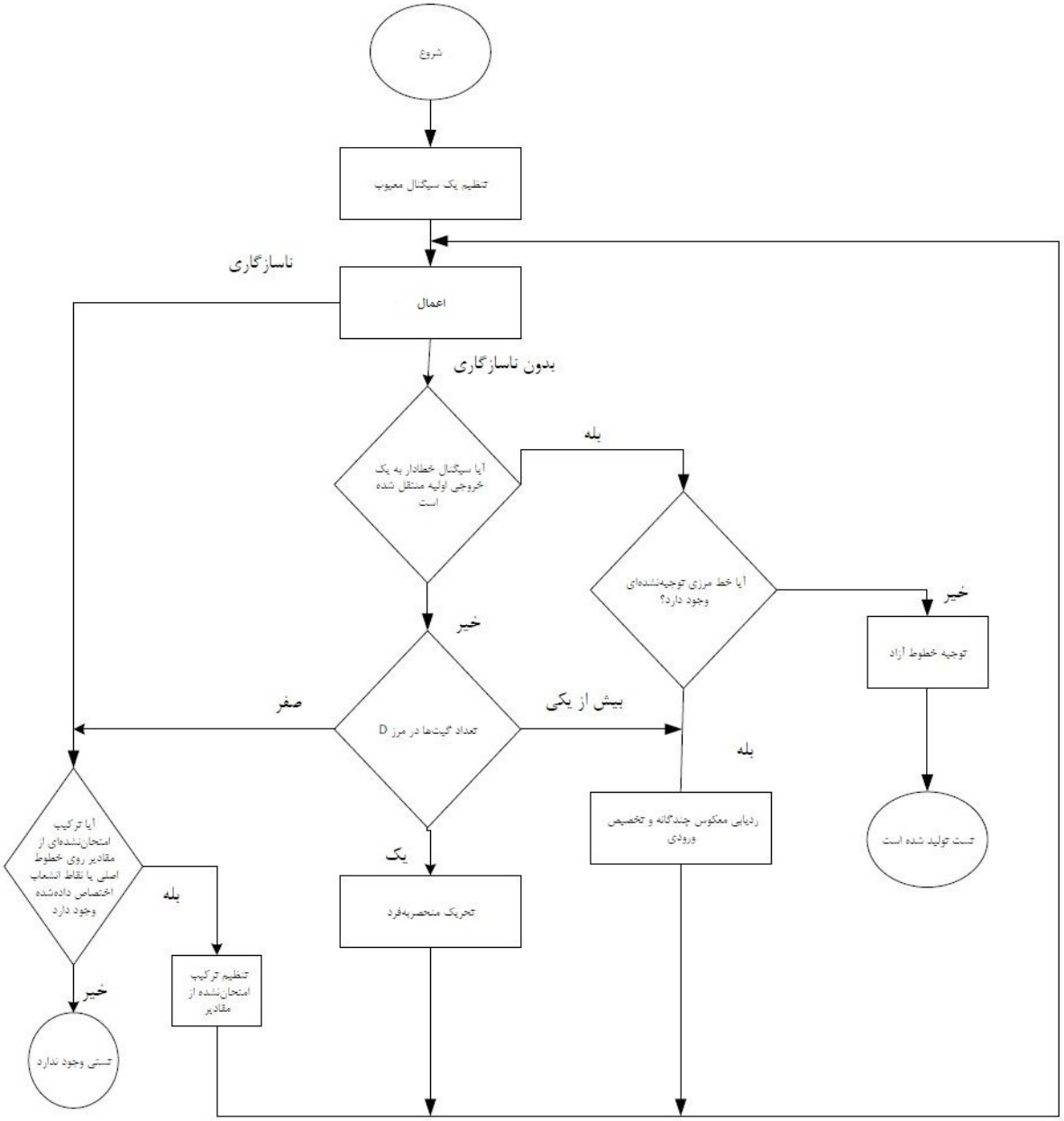
۱. ساختار درخت تصمیم: درخت تصمیم به طور مشابه با الگوریتم PODEM عمل می‌کند که به صورت یک لیست مرتب از گره‌ها سازماندهی شده است. هر گره نشانگر یک تخصیص فعلی از (۰) یا (۱) به یک خط سرسری خاص است که به ترتیب تخصیص‌ها سازماندهی شده‌اند.
۲. علامت‌گذاری گره‌ها و آزمایش‌های جایگزین: وقتی یک تخصیص اولیه در یک گره رد می‌شود، آن گره علامت‌گذاری می‌شود و الگوریتم به بررسی یک تخصیص جایگزین می‌پردازد.
۳. حذف گره‌ها: اگر هر دو گزینه تخصیص در یک گره رد شوند، آن گره از بررسی حذف می‌شود. سپس الگوریتم به گره والد برمی‌گردد و تخصیص فعلی آن را هم رد می‌کند.

این رویکرد که ترکیبی از چندین پشتوانه‌سازی و برگشتی ساختاریافته در درخت تصمیم است، به‌شدت دامنه راه‌حل‌های ممکن را محدود کرده و در عین حال پیچیدگی آزمایش‌ها و پیامدها را مدیریت می‌کند. با شناسایی سریع اهداف ناموثر، جستجو برای تخصیص‌های معتبر را بهینه می‌سازد و کارایی کلی در دستیابی به اهداف اولیه را افزایش می‌دهد. در زمینه تولید آزمون، به‌ویژه در الگوریتم تولید آزمون مبتنی بر انشعاب (FAN)، درخت‌های تصمیم نقش مهمی در سازماندهی گزینه‌های موجود در هر گره تصمیم دارند. درخت تصمیم یک الگوریتم طبقه‌بندی شناخته شده است که به خانواده الگوریتم‌های یادگیری نظارت شده تعلق دارد. به‌ویژه، این الگوریتم می‌تواند به مسائل رگرسیون و طبقه‌بندی اعمال شود و به همین دلیل در برنامه‌های مختلف یادگیری ماشین کاربردهای متنوعی دارد. درخت تصمیم با یک گره ریشه آغاز می‌شود که نمایانگر یک انتخاب یا ویژگی اولیه است. با اجرای الگوریتم، هر گره تصمیم به چندین انتخاب یا مسیر ممکن تقسیم می‌شود که می‌توان آنها را بررسی کرد. این ساختار امکان ارزیابی منعطف و سیستماتیک راه‌حل‌های احتمالی را فراهم می‌آورد. هنگام پیش‌بینی یک برچسب کلاس یا مقدار برای یک رکورد، الگوریتم مقادیر ویژگی‌ها را در هر گره مقایسه کرده و به سمت شاخه‌هایی که با آن مقادیر تطابق دارند، حرکت می‌کند تا به یک گره پایانی برسد.

از نقاط قوت الگوریتم FAN می توان به موارد ذیل اشاره کرد:

۱. انتخاب‌های متنوع در گره‌های تصمیم‌گیری: در تولید آزمون، هر گره تصمیم‌گیری امکان بررسی مسیرهای مختلف را فراهم می‌آورد و ارزیابی جامع‌تری از وکتورهای آزمون احتمالی انجام می‌دهد.

1. Stuck at Fault
2. Path-Oriented Decision Making
3. Fan-Out-Oriented



شکل ۵: نمودار جریان الگوریتم FAN.

۲. انتخاب اولیه دلخواه: انتخاب اولیه‌ای که در ریشه گره انجام می‌شود می‌تواند دلخواه باشد؛ اما با پیشرفت فرآیند، ممکن است نیاز به بازگشت وجود داشته باشد تا مسیرهای جایگزین بررسی شوند اگر مسیر کنونی نتواند راه‌حل معتبری ارائه دهد

۳. کاهش بازگشت‌ها: یکی از اهداف اصلی الگوریتم FAN کاهش بازگشت‌ها است، زیرا این امر می‌تواند کارایی را مختل کند. الگوریتم با شناسایی سریع مسیرهایی که به هیچ راه‌حلی نمی‌رسند، از تحقیقات غیرضروری جلوگیری کرده و زمان پردازش را تسریع می‌کند.

۴. کارایی از طریق تقسیم و محدودیت: در کنار روش‌هایی مانند تقسیم و محدودیت، درخت تصمیم به طور سیستماتیک به جستجوی راه‌حل‌های ممکن کمک می‌کند. اگر واضح شود که هیچ راه‌حلی در زیر یک گره کنونی وجود ندارد، نیمه‌کاره بازگشت آغاز می‌شود و از جستجوهای بی‌فایده در درخت جلوگیری می‌گردد.

۵. شتاب کلی الگوریتم: با ساختاردهی مسئله به عنوان یک درخت تصمیم و بهینه‌سازی مدیریت بازگشت، درخت تصمیم به طور قابل توجهی عملکرد کلی الگوریتم FAN را بهبود می‌بخشد.

درخت‌های تصمیم‌گیری می‌توانند بر اساس نوع متغیر هدفی که به آن پرداخته می‌شود، دسته‌بندی شوند. این دسته‌ها عبارتند از:

- درخت‌های تصمیم‌گیری با متغیر طبقه‌بندی: این درخت‌ها زمانی به کار می‌روند که متغیر هدف طبقه‌بندی باشد و درخت طوری ساختاردهی می‌شود که رکوردها را به کلاس‌های مختلف تقسیم کند.
- درخت‌های تصمیم‌گیری با متغیر پیوسته: این درخت‌ها زمانی استفاده می‌شوند که متغیر هدف پیوسته باشد و پیش‌بینی مقادیر عددی را تسهیل می‌کند.

با استفاده از درخت‌های تصمیم‌گیری در تولید آزمون، الگوریتم FAN می‌تواند به طور مؤثری کارایی و دقت تولید بردارهای آزمون را افزایش دهد و اهمیت عملی ساختارهای تصمیم‌گیری در فرآیندهای محاسباتی را نشان دهد.

۵- بهینه‌سازی درخت تصمیم در الگوریتم FAN با بهره‌گیری از الگوریتم PSO

در زمینه بهبود درخت‌های تصمیم برای تولید آزمایش با استفاده از الگوریتم FAN، بهینه‌سازی ازدحام ذرات (PSO) می‌تواند برای بهبود عملکرد مدل استفاده شود. PSOشکل یک روش محاسباتی الهام‌گرفته از رفتار اجتماعی پرندگان و ماهی‌ها است که برای یافتن راه‌حل‌های بهینه یا نزدیک به بهینه در فضاهای جستجوی پیچیده استفاده می‌شود.

۵-۱ ساختار و عملکرد الگوریتم PSO

مقداردهی اولیه: الگوریتم PSO با مقداردهی اولیه گروهی از ذرات آغاز می‌شود، جایی که هر ذره یک راه حل بالقوه برای پارامترهای درخت تصمیم، مانند عمق درخت، حداقل نمونه در هر برگ، یا معیارهای تقسیم را نشان می‌دهد. به روز رسانی سرعت و موقعیت: هر ذره موقعیت خود را در فضای محلول بر اساس تجربه خود و ذرات همسایه تنظیم می‌کند. قوانین به‌روزرسانی برای سرعت و موقعیت برای اطمینان از اکتشاف و بهره‌برداری از فضای جستجو فرموله شده است.

ارزیابی تناسب: تناسب هر ذره با استفاده از یک تابع هزینه از پیش تعریف شده که عملکرد درخت تصمیم را در زمینه تولید آزمایش منعکس می کند، ارزیابی می‌شود.

۵-۲ تابع هزینه برای PSO

تابع هزینه کلید هدایت الگوریتم PSO به سمت راه‌حل های بهینه است. برای درخت های تصمیم در الگوریتم FAN، مولفه های زیر را می‌توان در تابع هزینه در نظر گرفت:

$$C = \alpha . Accyracy + \beta Complexity + \gamma . Backtrack \quad (۱)$$

که در آن:

دقت[1]: این مؤلفه عملکرد درخت تصمیم را بر روی داده‌های آزمون اندازه‌گیری می‌کند که معمولاً به عنوان نسبت پیش‌بینی‌های صحیح محاسبه می‌شود.

پیچیدگی[2]: این مولفه مدل های بیش از حد پیچیده، مانند درختان عمیق یا درختان با گره‌های زیاد را جریمه می‌کند، که می‌تواند منجر به بیش از حد برازش شود. یک رویکرد رایج برای اندازه‌گیری پیچیدگی، شمارش تعداد گره‌ها یا عمق درخت است.

نسبت Backtrack[3]: این فرکانس بک ترک مورد نیاز در طول عملیات درخت تصمیم در الگوریتم FAN را اندازه گیری می کند. هرچه نسبت پسرفت کمتر باشد، راه حل کارآمدتر است. ضرایب α، β، γ فاکتورهای وزنی هستند که بسته به اهداف خاص بهینه سازی قابل تنظیم هستند. به عنوان مثال، اگر سادگی مدل در اولویت قرار گیرد، β را می‌توان نسبت به α و γ افزایش داد.

۵-۳ کاربرد PSO در بهبود درخت تصمیم

- بهینه‌سازی پارامترها: الگوریتم PSO می‌تواند پارامترهایی مانند عمق حداکثر، حداقل نمونه‌ها در هر برگ و معیارهای تقسیم را در درخت تصمیم بهینه کند؛ به طوری که ساختار درخت برای تولید آزمون مؤثر باشد.
- اجتناب از تخصص بیش از حد: با گنجاندن پیچیدگی و نسبت بازگشت به تابع هزینه، الگوریتم PSO می‌تواند به جلوگیری از مدل‌های بسیار تخصصی کمک کند که در داده‌های آموزشی خوب عمل می‌کنند، اما در کاربردهای واقعی ضعیف هستند.
- پایداری: استفاده از PSO می‌تواند به تولید درخت‌های تصمیم قوی‌تری منجر شود که دقت و کارایی را متوازن می‌کند و عملکرد کلی الگوریتم FAN را بهبود می‌بخشد.
- اصلاح تدریجی: طبیعت تکراری PSO این امکان را می‌دهد که ساختار درخت تصمیم به‌طور مداوم بهینه‌سازی شود، به طوری که تغییرات بر اساس بازخورد آنی از ارزیابی‌های Fitness انجام شود.

با ادغام PSO در بهینه‌سازی درخت‌های تصمیم برای الگوریتم FAN، می‌توان به فرآیند تولید آزمون بهینه‌تر و مؤثرتری دست یافت. استفاده از یک تابع هزینه به خوبی تعریف شده این امکان را برای الگوریتم فراهم می‌کند که بین دقت، پیچیدگی و کارایی عملی تعادل برقرار کند و در نهایت عملکرد درخت‌های تصمیم را در کاربردهای مختلف یادگیری ماشین بهبود بخشد. از طریق این رویکرد، امکان تولید بردارهای آزمون کیفیت بالا که عملکرد کلی فرآیندهای تصمیم‌گیری در وظایف محاسباتی را بهبود می‌بخشد، وجود دارد.

۵-٤ شبه‌کد بهینه‌سازی درخت تصمیم در الگوریتم PSO-FAN

شبه‌کد بهینه‌سازی درخت تصمیم در الگوریتم FAN با الگوریتم PSO در شکل ۶ مشاهده می‌شود.

مؤلفه‌های کلیدی کد بهینه‌شده عبارتند از:

۱. راه‌اندازی: تعداد ذرات و ابعاد مربوط به پارامترهای درخت تصمیم که باید بهینه‌سازی شوند، تعیین می‌شود.
۲. راه‌اندازی ذرات: ذرات با موقعیت‌ها و سرعت‌های تصادفی راه‌اندازی می‌شوند.
۳. ارزیابی تناسب: برای هر ذره، تناسب با استفاده از یک تابع هزینه که بر اساس دقت، پیچیدگی و نسبت بازگشت است، ارزیابی می‌شود.
۴. به‌روزرسانی سرعت و موقعیت: هر ذره سرعت خود را بر اساس بهترین موقعیت خود و بهترین موقعیت جهانی به‌روزرسانی کرده و سپس موقعیت خود را در فضای حل به‌روزرسانی می‌کند.
۵. خاتمه: الگوریتم به تکرار ادامه می‌دهد تا حداکثر تعداد تکرارها به پایان برسد.

٦- نتایج تجربی

در این بخش، نتایج بررسی مدارهای benchmark ISCAS'۸۹ را که در جدول ۱ آورده شده است با استفاده از الگوریتم FAN و نسخه بهینه‌سازی‌شده آن که با استفاده از بهینه‌سازی اجتماع ذرات (PSO) تقویت شده است، ارائه می‌گردد. همان‌طور که در بخش قبلی ذکر شد، الگوریتم FAN با استفاده از رویکرد درخت تصمیم، بالاترین پوشش خطا را با کمترین تعداد بردارهای آزمون به‌دست می‌آورد که در جدول ۲ نشان داده شده است. جدول ۳، نتایج به‌دست‌آمده با استفاده از الگوریتم PSO-FAN را نمایش می‌دهد.

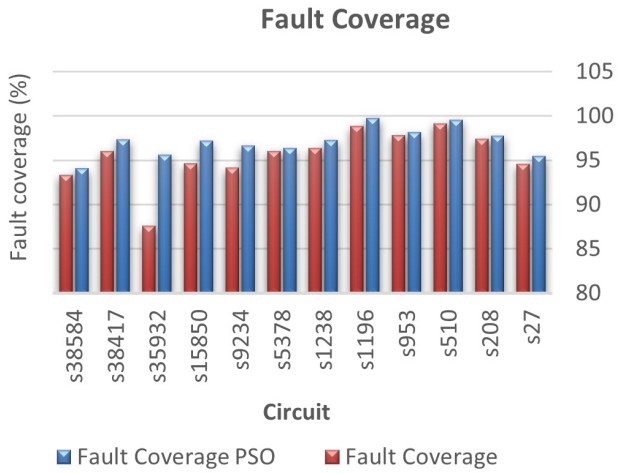
در بررسی مدارات مرجع از مجموعه آزمایش ISCAS'۸۹، می‌توان بهبودهای قابل توجهی در پوشش عیوب و شناسایی عیوب هنگام استفاده

1. Accuracy
2. Complexity
3. Backtrack Ratio

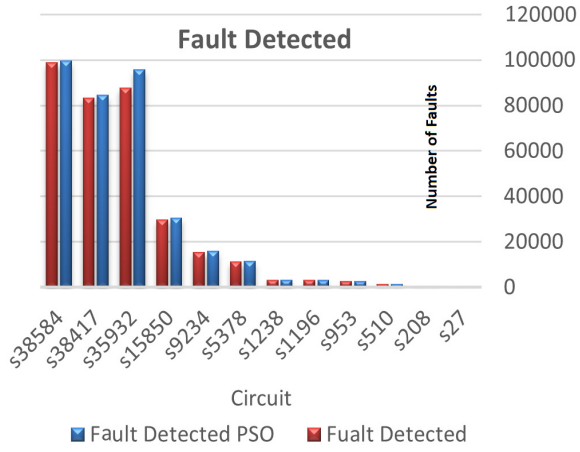
```
Initialize PSO Parameters:
   N ← Number of particles
   D ← Dimension (number of parameters to optimize)
   α, β, γ ← Weight factors for cost function components
   MaxIterations ← Maximum number of iterations

Initialize Particles:
   For i ← 1 to N do
      Particle[i].Position ← Randomly initialized position in
solution space
      Particle[i].Velocity ← Random initial velocity
      Particle[i].BestPosition ← Particle[i].Position
      Particle[i].BestCost ← Infinity  // Initialize best cost for each
particle

GlobalBestPosition ← Array of size D initialized to zeros
GlobalBestCost ← Infinity  // Initialize global best cost

Iterate until MaxIterations:
   For i ← 1 to N do
      // Evaluate fitness of each particle
      Cost ← EvaluateFitness(Particle[i].Position)

      // Update personal best
      If Cost < Particle[i].BestCost then
         Particle[i].BestCost ← Cost
         Particle[i].BestPosition ← Particle[i].Position

      // Update global best
      If Cost < GlobalBestCost then
         GlobalBestCost ← Cost
         GlobalBestPosition ← Particle[i].Position

   For i ← 1 to N do
      // Update velocity
      Particle[i].Velocity ← UpdateVelocity(Particle[i].Velocity,
                              Particle[i].BestPosition,
                              GlobalBestPosition)

      // Update position
      Particle[i].Position ← UpdatePosition(Particle[i].Position,
                              Particle[i].Velocity)
```

```
Function EvaluateFitness(Position):
   // Construct decision tree using parameters from Position
   Tree ← ConstructDecisionTree(Position)

   // Calculate accuracy, complexity, and backtrack ratio
   Accuracy ← CalculateAccuracy(Tree)
   Complexity ← CalculateComplexity(Tree)
   BacktrackRatio ← CalculateBacktrackRatio(Tree)

   // Compute cost using the defined cost function
   Cost ← (α * Accuracy) + (β * Complexity) + (γ *
BacktrackRatio)
   Return Cost

Function UpdateVelocity(Velocity, BestPosition,
GlobalBestPosition):
   // Use PSO update rules
   Inertia ← w * Velocity
   Cognitive ← c1 * Random() * (BestPosition - Particle.Position)
   Social ← c2 * Random() * (GlobalBestPosition -
Particle.Position)
   Velocity ← Inertia + Cognitive + Social
   Return Velocity

Function UpdatePosition(Position, Velocity):
   Position ← Position + Velocity
   Return Position

Function ConstructDecisionTree(Position):
   // Build decision tree based on the parameters represented in
Position
   Return decisionTree

Function CalculateAccuracy(Tree):
   // Evaluate the accuracy of the decision tree on test data
   Return accuracy

Function CalculateComplexity(Tree):
   // Measure the complexity of the decision tree (e.g., node count)
   Return complexity

Function CalculateBacktrackRatio(Tree):
   // Analyze the tree to determine the backtracking ratio during
FAN execution
   Return backtrackRatio
```

شکل۶. شبه کد بهینه سازی درخت تصمیم در الگوریتم FAN با الگوریتم PSO.



شکل ۸: مقایسه درصد پوشش خطاهای یافت‌شده توسط الگوریتم.



شکل۷: مقایسه تعداد خطاهای یافت‌شده توسط الگوریتم FAN با الگوریتم PSO-FAN.

جدول ۱: مشخصات مدارات معیار سری ISCAS'۸۹.

مدار معیار	تعداد ورودی‌ها	تعداد خروجی‌ها	تعداد گیت‌ها	تعداد کلی خطاها
s۲۷	۴	۱	۸	۱۱۰
s۲۰۸	۱۱	۲	۶۱	۶۲۲
s۵۱۰	۱۹	۷	۱۷۹	۱۴۰۲
s۹۵۳	۱۶	۲۳	۳۱۱	۲۷۰۲
s۱۱۹۶	۱۴	۱۴	۳۸۸	۳۱۰۴
s۱۲۳۸	۱۴	۱۲	۴۲۸	۳۳۵۴
s۵۳۷۸	۳۵	۴۹	۱۰۰۴	۱۱۸۲۲
s۹۲۳۴	۳۶	۳۹	۲۰۲۷	۱۶۴۷۶
s۱۵۸۵۰	۷۷	۱۵۰	۳۴۴۸	۳۱۴۵۶
s۵۹۳۲	۳۵	۳۲۰	۱۲۲۰۴	۱۰۰۰۱۸
s۳۸۴۱۷	۲۸	۱۰۶	۸۷۰۹	۸۶۸۲۴
s۳۸۵۸۴	۳۸	۳۰۴	۱۱۴۴۸	۱۰۵۸۴۲

جدول ۲: الگوریتم FAN، با استفاده از رویکرد درخت تصمیم [۳۱].

بردار آزمون	پوشش خطا (٪)	خطاهای یافت‌شده	تعداد کلی خطاها	مدار معیار
۵	۹۴٫۵۵	۱۰۴	۱۱۰	s۲۷
۲۹	۹۷٫۴۳	۶۰۶	۶۲۲	s۲۰۸
۵۹	۹۹٫۱۴	۱۳۹۰	۱۴۰۲	s۵۱۰
۸۹	۹۷٫۸۵	۲۶۴۴	۲۷۰۲	s۹۵۳
۱۳۴	۹۸٫۸۴	۳۰۶۸	۳۱۰۴	s۱۱۹۶
۱۴۵	۹۶٫۳۶	۳۲۳۲	۳۳۵۴	s۱۲۳۸
۱۱۷	۹۶٫۰۴	۱۱۳۵۴	۱۱۸۲۲	s۵۳۷۸
۱۵۶	۹۴٫	۱۵۵۱۱	۱۶۴۷۶	s۹۲۳۴
۱۳۳	۹۴٫۶۲	۲۹۷۶۳	۳۱۴۵۶	s۱۵۸۵۰
۲۱	۸۷٫۵۸	۸۷۵۹۴	۱۰۰۰۱۸	s۵۹۳۲
۱۰۵	۹۶٫۰۰	۸۳۳۴۷	۸۶۸۲۴	s۳۸۴۱۷
۱۳۳	۹۳٫۳۳	۹۸۷۸۲	۱۰۵۸۴۲	s۳۸۵۸۴

۲. شناسایی عیوب: تعداد عیوب شناسایی‌شده به‌طور عمده افزایش می‌یابد، که نشان‌دهنده کارایی بالاتر این روش در شناسایی نقص‌ها در رویکرد PSO-FAN است.
۳. کارایی بردار های آزمون: در چندین مورد، تعداد بردارهای آزمون مورد نیاز کاهش یافته است، که این موضوع برای پیاده‌سازی عملی مفید است، زیرا ممکن است زمان و هزینه‌های آزمون را کاهش دهد.

به‌طور کلی الگوریتم PSO-FAN، بهبودهای قابل توجهی در پوشش عیوب و کارایی شناسایی عیوب در تمام مدارات مرجع نسبت به الگوریتم اصلی FAN نشان می‌دهد.

۷- نتیجه‌گیری

نتایج این تحقیق به وضوح نشان می‌دهد که حوزه آزمون مدارهای مجتمع با مقیاس بسیار بزرگ (VLSI) به‌ویژه در زمینه بهبود روش‌های تشخیص عیب، پیشرفت‌های قابل توجهی را تجربه کرده است. با توجه به

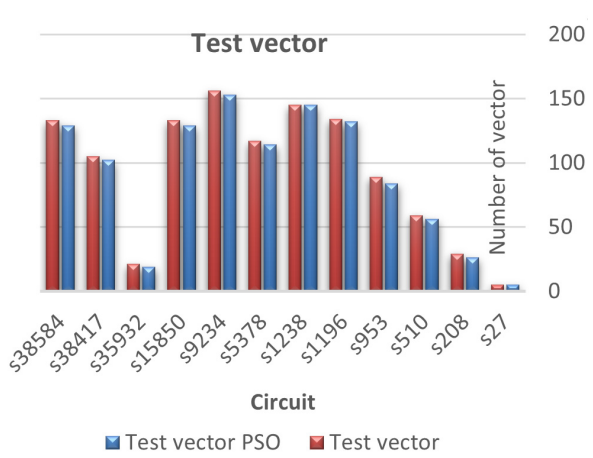
جدول ۳: الگوریتم PSO-FAN.

بردار آزمون	پوشش خطا (٪)	خطاهای یافت‌شده	تعداد کلی خطاها	مدار معیار
۵	۹۵٫۴۵۴۵	۱۰۵	۱۱۰	s۲۷
۲۶	۹۷٫۷۴۹	۶۰۸	۶۲۲	s۲۰۸
۵۶	۹۹٫۵۷۲	۱۳۹۶	۱۴۰۲	s۵۱۰
۸۴	۹۸٫	۲۶۵۲	۲۷۰۲	s۹۵۳
۱۳۲	۹۹٫۷۴۲۳	۳۰۹۶	۳۱۰۴	s۱۱۹۶
۱۴۵	۹۷٫۲۸۶۸	۳۲۶۳	۳۳۵۴	s۱۲۳۸
۱۱۴	۹۶٫۳۴۵۸	۱۱۳۹۰	۱۱۸۲۲	s۵۳۷۸
۱۵۳	۹۶٫۶۴۳۶	۱۵۹۲۳	۱۶۴۷۶	s۹۲۳۴
۱۲۹	۹۷٫۱۸۰۲	۳۰۵۶۹	۳۱۴۵۶	s۱۵۸۵۰
۱۹	۹۵٫۶۱۴۸	۹۵۶۳۲	۱۰۰۰۱۸	s۵۹۳۲
۱۰۲	۹۷٫۳۴۹۸	۸۴۵۲۲	۸۶۸۲۴	s۳۸۴۱۷
۱۲۹	۹۴٫۰۶۶۶	۹۹۵۶۲	۱۰۵۸۴۲	s۳۸۵۸۴

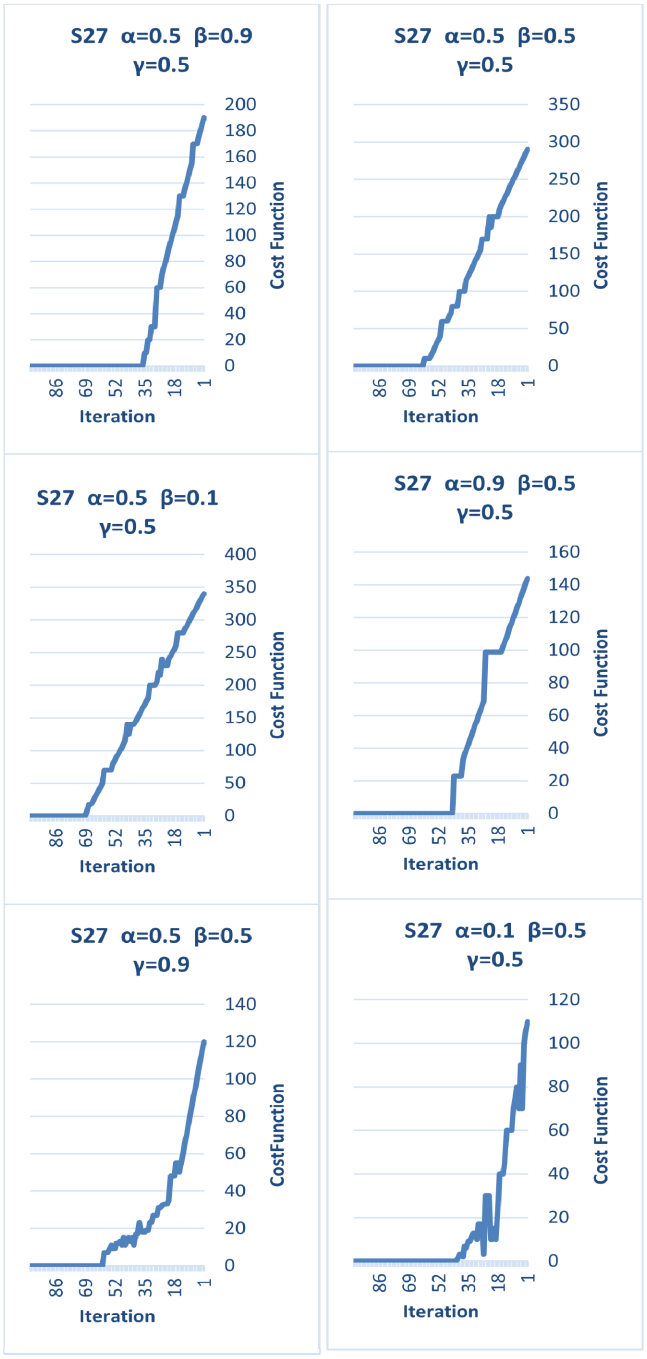
جدول ۴: مقایسه با پژوهش‌های گذشته.

مدار معیار	پژوهش گذشته [۱]	پژوهش گذشته [۲]	پژوهش گذشته [۳]	الگوریتم PSO-FAN
s۲۹۸	۴۲٫۶۶	—	۸۴	۹۷
s۳۸۶	۳۴٫۴۳	۵۱٫۸۹	۶۲	۷۳
s۵۲۶	۶۶٫۳	۵۵٫۲۴	۸۶	۹۱
s۱-۳۴۲۰	۲۷٫۱۹	—	۲۲	۳۷
s۱۱۹۶	۳۱٫۳۲	۱۰٫۳	۶۲	۹۹
s۱۲۳۸	—	۳٫۸۳	۸۷	۹۷
s۱-۱۳۲۰۷	۸۷٫۱۴	۲۵٫۱۳	۹۴	۹۷
s۱-۱۵۸۵۰	۸۳٫۰۵	۳٫۱	۵۳	۸۹

پیچیدگی‌های روزافزون طراحی مدارها و تنوع در فرآیندهای تولید، لزوم مکان‌یابی دقیق عیوب در مراحل پیش‌تولید و آزمایش به‌ویژه حس می‌شود. از طریق پیاده‌سازی الگوریتم بهینه‌سازی شده FAN با استفاده از تکنیک بهینه‌سازی ازدحام ذرات (PSO) بر روی مدارهای معیاری ISCAS'۸۹، بهبودهایی در پوشش عیب و دقت تشخیص حاصل شده است. این بهینه‌سازی نه تنها پوشش عیب را افزایش داده، بلکه همچنین منجر به کاهش تعداد بردارهای آزمایشی مورد نیاز شده است که این امر کارایی آزمایش را در سناریوهای تولید با حجم بالا به طرز چشمگیری ارتقا می‌دهد و متعاقبا منجر به کاهش زمان پردازش نیز خواهد شد. این تحقیق بر اهمیت توسعه معیارهای تشخیصی و به‌کارگیری تکنیک‌های طراحی برای آزمون‌پذیری تأکید دارد. به‌علاوه، استفاده از الگوریتم PSO-FAN پتانسیل بالای این روش را در فرآیند تولید آزمون و ایجاد تعادلی مناسب میان دقت، پیچیدگی و کارایی عملیاتی نشان می‌دهد (شکل ۱۰). با توجه به این پیشرفت‌ها، انتظار می‌رود که این رویکرد بتواند به تولید بردارهای آزمایشی با کیفیت بالا و بهبود تصمیم‌گیری در فرآیندهای محاسباتی کمک کند. به‌طور کلی، این مطالعات ضرورت توجه بیشتر به راه‌حل‌های تشخیصی قوی‌تر را در مواجهه با چالش‌های پیچیده سخت‌افزاری و بهبود قابلیت اطمینان مدارهای مجتمع به‌روشنی نمایان می‌سازد.



شکل ۹: مقایسه تعداد بردار آزمون استفاده‌شده.



شکل۱۰: تاثیر تغییر پارامترهای α, β, γ در تابع هزینه

مراجع

[1] Y. Lai, *et al.*, "Chipformer: Transferable chip placement via offline decision transformer," in *Proc. Int. Conf. on Machine Learning*, pp. 18346-18364, Honolulu, HI, USA, 23-29 Jul. 2023.

[2] L. Gaber, A. I. Hussein, and M. Moness, "Improved automatic correction for digital VLSI circuits," in *Proc. 31st Int. Conf. on Microelectronics*, pp. 18-22, Cairo, Egypt, 15-18 Dec. 2019.

[3] J. Shanthi, D. G. N. Rani, and S. Rajaram," A C4.5 decision tree classifier based floorplanning algorithm for system-on-chip design," *Microelectronics Journal*, vol. 121, Article ID:105361, Mar. 2022.

[4] M. Alateeq and W. Pedrycz, "A comparative analysis of bio-inspired optimization algorithms for automated test pattern generation in sequential circuits," *Applied Soft Computing*, vol. 101, Article ID:106967, Mar. 2021.

[5] S. H. Lee, S. J. Lee, J. Park, E. C. Lee, and H. G. Kang, "Development of simulation-based testing environment for safety-critical software," *Nuclear Engineering and Technology*, vol. 50, no. 4, pp. 570-581, May 2018.

[6] M. Gaudesi, I. Pomeranz, M. S. Reorda, and G. Squillero, "New techniques to reduce the execution time of functional test programs," *IEEE Trans. on Computers*, vol. 66, no. 7, pp. 1268-1273, Jul. 2017.

[7] T. S. Letia and A. O. Kilyen, "Fuzzy logic enhanced time Petri Net models for hybrid control systems," in *Proc. IEEE Int. Conf. on Automation, Quality and Testing, Robotics*, 6 pp., Cluj-Napoca, Romania, 19-21 May 2016.

[8] A. Floridia, E. Sanchez, and M. S. Reorda, "Fault grading techniques of software test libraries for safety-critical applications," *IEEE Access*, vol. 7, pp. 63578-63587, 2019.

[9] M. Gaudesi, M. S. Reorda, and I. Pomeranz, "On test program compaction," in *Proc. 20th IEEE European Test Symp.* 6 pp. Cluj-Napoca, Romania, 25-29 May 2015.

[10] S. Eggersglüß, S. Milewski, J. Rajski, and J. Tyszer, "On reduction of deterministic test pattern sets," in *Proc. IEEE Int. Test Conf.*, pp. 260-267, Anaheim, CA, USA, 10-15 Oct 2021.

[11] M. A. Kochte and H. J. Wunderlich, "Self-test and diagnosis for self-aware systems," *IEEE Design & Test*, vol. 35, no. 5, pp. 7-18, Oct. 2018.

[12] P. Wang, C. J. Moore, A. M. Gharehbaghi, and M. Fujita, "An ATPG method for double stuck-at faults by analyzing propagation paths of single faults," *IEEE Trans. on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 1063-1074, Mar. 2018.

[13] L. Malihi and R. Malihi, "Single stuck-at-faults detection using test generation vector and deep stacked-sparse-autoencoder," *SN Applied Sciences*, vol. 2, no. 10, Article ID: 1715, 2020.

[14] T. Shah, A. Matrosova, M. Fujita, V. Singh, "Multiple stuck-at fault testability analysis of ROBDD based combinational circuit design," *Journal of Electronic Testing*, vol. 34, pp. 53-65, 2018.

[15] V. K. Singh, T. Sarkar, and S. N. Pradhan, "Power-aware testing for maximum fault coverage in analog and digital circuits simultaneously," *IETE Technical Review*, vol. 39, no. 6, pp. 1395-1409, 2022.

[16] L. Gaber, A. I. Hussein, M. Moness, "Fault detection based on deep learning for digital VLSI circuits," *Procedia Computer Science*, vol. 194, 122-131, 2021.

[17] H. Hu, F. Feng, T. Wang, "Open-circuit fault diagnosis of NPC inverter IGBT based on independent component analysis and neural network," *Energy Reports* , vol. 6, Supp. 9, 134-143, Dec. 2020.

[18] T. Wang, H. Xu, J. Han, E, Elbouchikhi, and M. El Hachemi Benbouzid, "Cascaded H-bridge multilevel inverter system fault diagnosis using a PCA and multi-class relevance vector machine approach," *IEEE Trans. Power Electronics,* vol. 30, no. 12, pp. 7006-7018, Dec. 2015.

[19] Z. Yin, L. Wang, Y. Zhang, and Y. Gao, "A novel arc fault detection method integrated random forest, improved multi-scale permutation entropy and wavelet packet transform," *Electronics*, vol. 8, no. 4, Article ID: 396, Apr. 2019.

[20] K. Simonyan and A, Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, arXiv preprint, arXiv:1409.1556, 2015,

[21] Z. Wu, *et al.*, "A comprehensive survey on graph neural networks," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, Jan. 2021.

[22] H. Wu and J. Zhao, "Deep convolutional neural network model based chemical process fault diagnosis," *Computers & Chemical Engineering*, vol. 115, pp. 185-197, Jul. 2018.

[23] Z. Zhang and J. Zhao, "A deep belief networkbased fault diagnosis model for complex chemical processes," *Computers & Chemical Engineering*, vol. 107, pp. 395-407, Dec. 2017.

[24] W. Liu, *et al.,* "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11-26, Apr. 2017.
[25] H. Wang *et al.*, "Scientific discovery in the age of artificial intelligence," *Nature*, vol. 620, no. 7972, pp. 47-60, 2023.
[26] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, Article ID:160, 2021.
[27] Y. Xie, H. Zeng, C. Hu, and Z. Ji, "Security/timming-aware design space exploration of CAN FD for automative cyber-phsical systems," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 2, pp.1094-1104, Feb. 2019.
[28] H. Xiao, M. Cao, R. Peng, "Artificial neural networkbased software fault detection and correction prediction models considering testing effort," *Applied Soft Computing*, vol. 94, Article ID: 106491, Sept. 2020.
[29] W. Jiang, L. Wen, J. Zhan, K. Jiang, "Design optimization of confidentiality-critical cyber physical systems with fault detection," *Journal of Systems Architecture*, vol. 107, Article ID:101739, Aug. 2020.
[30] M. Y. Ali, *Transition Delay Test Generation Using Stuck-At-Fault Test Patterns*, M.Sc. Thesis, Tallinn University of Technology, 2020.
[31] D. M. Miller and M. A. Thornton, *Multiple-Valued Logic: Concepts and Representations*, Springer Nature, 2022.
[32] R. C. Dorf, *The Electrical Engineering Handbook-Six Volume Set*, CRC Press, 2018.
[33] H. Fujiwara, "Fan: a fanout-oriented test pattern generation algorithm," in *Proc*. of *IEEE Int. Symp. on Circuits and Systems*, pp. 671-674, Kyoto, Japan, 5-7 Jun. 1985.
[34] S. Khan and P. Sarkar, "A comprehensive review of machine learning applications in VLSI testing: Unveiling the future of semiconductor manufacturing," *in Proc. 7th Int. Conf. on Electronics, Materials Engineering & Nano-Technology,* 5 pp., Kolkata, India, 18-20 Dec. 2023.

سید حمید ظهیری مدرک کارشناسی، کارشناسی ارشد و دکترای خود را در رشته مهندسی الکترونیک به ترتیب در سال‌های ۱۳۷۲، ۱۳۷۴ و ۱۳۸۴ از دانشگاه صنعتی شریف، تهران، دانشگاه تربیت مدرس و دانشگاه فردوسی مشهد دریافت کرد. در حال حاضر، ایشان استاد دانشکده مهندسی الکترونیک دانشگاه بیرجند هستند. زمینه‌های تحقیقاتی ایشان شامل تشخیص الگو، الگوریتم‌های تکاملی، الگوریتم‌های هوش ازدحامی و محاسبات نرم است.

احسان حق‌پرست مدرک کارشناسی خود را در رشته مخابرات از دانشگاه بیرجند در سال ۱۳۸۹ و مدرک کارشناسی ارشد خود را در رشته مهندسی برق در سال ۱۳۹۶ از دانشگاه آزاد اسلامی واحد بیرجند دریافت کرد. او در حال حاضر مشغول به تحصیل در مقطع دکترا در گروه مهندسی الکترونیک، دانشگاه بیرجند است.

ابوالفضل بیجاری مدرک کارشناسی مهندسی مخابرات، کارشناسی ارشد و دکترای مهندسی الکترونیک خود را به‌ترتیب در سال‌های ۱۳۸۳، ۱۳۸۶ و ۱۳۹۱ از دانشگاه فردوسی مشهد دریافت کرد. در حال حاضر، ایشان دانشیار دانشکده مهندسی الکترونیک دانشگاه بیرجند هستند.