Computational Intelligence in Electrical Engineering
Vol. 13, No. 4, 2023
Research Paper

SA-based Approach to Implement Digital Systems on 3D Integrated Circuits

Hemin Rahimi [1], Hadi jahanirad [2]

[1] M.Sc., Department of Electrical Engineering,University of Kurdistan, Sanandaj, Iran
[2] Assistant Professor, Department of Electrical Engineering, University of Kurdistan, Sanandaj, Iran

Abstract:
The 3D integrated circuit is emerged as a promising solution to integrate very large-scale circuits on electronics chips. In such chips, several layers of silicon substrates are stacked which are separated by insulator interfaces. Interconnection between two layers is realized using Through Silicon Via (TSV). Fabrication of TSVs is challenging due to their large size and complex process. Consequently, the number of TSVs should be minimized in the circuit's implementation. The 3D implementation consists of three main steps: Partitioning, Placement, and Routing. In this paper, the first two steps are accomplished using the Simulated Annealing-based optimization approach wherein minimization of the number of TSVs and total wire length are considered the main objectives. In this paper, an improved version of the pathfinder method has been developed which would efficiently generate the necessary interconnections among circuit modules. The results of simulations on MCNC benchmark circuits show that the proposed method outperforms the previous state-of-the-art methods in all aspects. In comparison with FSA, the number of TSVs is reduced by 6.15%, and the algorithm's runtime is decreased by 27.79%. Moreover, in comparison with the hMETIS method, the number of TSVs is reduced by 9.78%, and the algorithm's runtime is decreased by 31.73% .

Keywords: 3D IC, Evolutionary algorithms, SA algorithm, Partitioning, Placement, and Routing.

https://doi.org/10.22108/isee.2021.127039.1447

مقاله پژوهشی

پیاده‌سازی مدارهای دیجیتال روی تراشه‌های سه‌بعدی با استفاده از الگوریتم تبرید شبیه‌سازی‌شده

هیمن رحیمی[۱]، هادی جهانی‌راد[۲*]

۱- کارشناسی ارشد گروه مهندسی برق، دانشکده مهندسی، دانشگاه کردستان، سنندج، ایران
hemn.rahimi@uok.ac.ir

۲- استادیار گروه مهندسی برق، دانشکده مهندسی، دانشگاه کردستان، سنندج، ایران
h.jahanirad@uok.ac.ir

چکیده: تراشه‌های سه‌بعدی در سال‌های اخیر به‌منزلهٔ یک راه‌حل برای مجتمع‌سازی مدارهای الکترونیکی دیجیتال با اندازهٔ بسیار بزرگ مطرح شده‌اند. در این تراشه‌ها چند لایهٔ سیلیکونی روی هم قرار می‌گیرند که با یک واسط عایق از هم تفکیک شده‌اند. ارتباط بین لایه‌ها با اتصالات ویژه‌ای به نام TSV انجام می‌شود. اندازهٔ TSVها بسیار بزرگ‌تر از اندازهٔ گیت‌های منطقی است و همچنین، ساختن این نوع اتصالات بسیار پرهزینه است؛ بنابراین، ساختن تراشه‌های سه‌بعدی با شمار TSV کمتر، یکی از اهداف مهم در طراحی این تراشه‌هاست. پیاده‌سازی مدارهای منطقی دیجیتال روی تراشه‌های سه‌بعدی در سه مرحلهٔ کلی انجام می‌شود؛ بخش‌بندی، جانشانی و مسیردهی. در این مقاله مرحلهٔ بخش‌بندی و جانشانی با استفاده از الگوریتم فراابتکاری تبرید شبیه‌سازی‌شده یا SA انجام می‌شود که هدف اصلی این دو مرحله، کاهش تعداد TSVها و طول سیم به‌کاررفته در جانشانی بلوک‌های منطقی است. در این مقاله، یک نسخهٔ بهبودیافته از الگوریتم مسیریاب توسعه داده شده است که به‌صورت کارا سیم‌بندی لازم برای اتصال ماجول‌ها را ایجاد می‌کند. نتایج شبیه‌سازی مدارهای معیار MCNC نشان می‌دهند روند طراحی ارائه‌شده نسبت به روش‌های پیشین، بسیار کاراتر است. در روش بخش‌بندی ارائه‌شده نسبت به روش FSA، TSVها به اندازهٔ ۶/۱۵ درصد و زمان اجرا به میزان ۲۷/۷۹ درصد کاهش یافته‌اند. همچنین، در مقایسه با الگوریتم بخش‌بندی hMetis، به اندازهٔ ۹/۷۸ درصد کاهش در تعداد TSV ایجاد شده است. این میزان بهبود در حالی است که الگوریتم پیشنهادی به میزان ۳۱/۷۳ درصد سریع‌تر عمل می‌کند.

واژه‌های کلیدی: مدارهای مجتمع سه‌بعدی، الگوریتم‌های فراابتکاری، الگوریتم SA، بخش‌بندی، جانشانی و مسیردهی.

۱- مقدمه

در گذار از یک نسل تراشه‌های مجتمع دیجیتال به نسل بعد، همواره سعی بر آن است که براساس قانون مور [۱]، تعداد ترانزیستورهای مجتمع‌شده دو برابر شود. این امر معمولاً با کوچک‌کردن اندازهٔ ترانزیستورها انجام می‌شود؛ برای مثال، پردازندهٔ ساخته‌شده در شرکت NVIDIA موسوم به GA100 Ampere در سال ۲۰۲۰ با فناوری ۷ نانومتر ساخته شده که شامل ۵۴ میلیارد ترانزیستور است [۲۲]. با کاهش مقیاس ترانزیستورها به زیر ۱۰ نانومتر، ادامهٔ روند ترسیم‌شده با قانون مور ازطریق کاهش اندازهٔ ترانزیستورها، تقریباً غیرممکن شده است. برطبق آخرین گزارش ITRS در سال ۲۰۱۵، با کوچک‌شدن اندازهٔ ترانزیستورها، پس از سال ۲۰۲۱، کاهش مقیاس تقریباً امکان‌پذیر نخواهد بود [۲].

[۱] تاریخ ارسال مقاله: ۱۳۹۹/۱۰/۲۷
تاریخ پذیرش مقاله: ۱۴۰۰/۰۵/۱۷
نام نویسندهٔ مسئول: هادی جهانی‌راد
نشانی نویسندهٔ مسئول: ایران - سنندج - دانشگاه کردستان - دانشکده مهندسی - گروه مهندسی برق

با کاهش اندازهٔ ترانزیستورها در حوزهٔ نانومتری، اثرات نامطلوبی مانند افزایش توان نشتی و وابستگی بسیار زیاد عملکرد ترانزیستور به تغییرات ناشی از مشخصات فناوری ساخت، نمود بیشتری پیدا می‌کنند [۳, ٤]. همچنین، اتصالات[1] در تراشه‌های نانومتری سهم بزرگی از تأخیر را به خود اختصاص می‌دهند؛ برای مثال، در فناوری ۹۰ نانومتری، تقریباً ۷۵ درصد تأخیر کلی تراشه، ناشی از اتصالات داخلی و سیم‌بندی‌ها است [۵].

مدارهای مجتمع سه‌بعدی[2] به‌عنوان یک راه‌حل بسیار کارا برای حل مشکلات ذکرشده مطرح شده‌اند. در این تراشه‌ها، طول سیم لازم برای اتصال ماجول‌های مختلف کاهش می‌یابد که نتیجهٔ مستقیم آن، افزایش سرعت تراشه، بهبود مجتمع‌سازی و کاهش توان مصرفی است [٦]. در معماری تراشه‌های مذکور، تعداد N لایهٔ فعال روی یکدیگر قرار گرفته‌اند که هرکدام ساختاری شبیه به تراشه‌های دوبعدی دارند (یعنی ترانزیستورها روی سطح بالایی آنها ساخته می‌شوند و اتصالات لازم بین این ترانزیستورها، با چند لایهٔ سیم‌بندی در بالای لایهٔ فعال انجام می‌شود). همچنین، برای ایجاد ارتباط بین ترانزیستورها در لایه‌های مختلف، از اتصالات عمودی ویژه‌ای به نام TSV[3] استفاده می‌شود. برای ایجاد TSV بین لایه‌های فعال ۱ و ۲، باید کانالی (با سطح مقطع دایره‌ای، مربعی و ...) از رویهٔ زیرین لایهٔ فعال دوم به رویهٔ بالایی آن ایجاد شود. برای ایجاد ارتباط عمودی بین دو لایهٔ فعال غیرمجاور (مثلاً لایه‌های ۱ و ٤) باید کانال مدنظر از لایه‌های میانی (۲، ۳ و ٤) عبور کند. ساختن کانال‌های این چنینی با محدودیت‌های مکانیکی و پیچیدگی‌های ساخت فراوانی همراه است؛ بنابراین، مدار مدنظر باید به‌گونه‌ای بین لایه‌های فعال تقسیم‌بندی شود که به کمترین تعداد TSV نیاز داشته باشد.

مسئلهٔ مهم دیگر، حرارت تولیدشده در لایه‌های میانی است که باید به‌صورت مناسبی به یک Heatsink منتقل شود که پایین‌تر از لایهٔ اول قرار دارد. این انتقال با استفاده از کانال‌های حرارتی ساخته‌شده با استفاده از TSVها انجام می‌شود.

پیاده‌سازی مدارهای منطقی روی تراشه‌های سه‌بعدی با تعداد N_L لایهٔ فعال، در سه مرحله انجام می‌شود. ابتدا مدار مدنظر به N_L بخش مساوی تقسیم می‌شود. سپس هرکدام از بخش‌ها روی یک لایه پیاده‌سازی می‌شوند. پیاده‌سازی روی لایه‌ها در دو مرحلهٔ جانشانی (انتساب گیت‌ها به مکان‌هایی روی سطح لایه) و مسیردهی (ایجاد سیم‌بندی‌های لازم بین گیت‌ها) انجام می‌شود. از منظر پیچیدگی محاسباتی هر سه مرحلهٔ بخش‌بندی، جانشانی و مسیردهی از نوع مسائل NP-Hard هستند؛ بنابراین، حل تحلیلی آنها امکان‌پذیر نیست و برای یافتن راه‌حل‌های پذیرفتنی، مجبور به استفاده از روش‌های فراابتکاری هستیم.

در مطالعات پیشین، روش‌های فراابتکاری متنوعی برای پیاده‌سازی مدارهای دیجیتال روی تراشه‌های سه‌بعدی ارائه شده است. مراجع [۷]، [۸] و [۹] از الگوریتم ژنتیک در حل مسئله استفاده کرده‌اند. در مرجع [۷] دو هدف اصلی کاهش طول سیم مصرفی (کاهش تأخیر مدار) و توزیع مناسب بلوک‌ها برای انتقال بهینهٔ حرارت تولیدشده در لایه‌های میانی تعریف شده است. نتایج نشان‌دهندهٔ کاهش میزان میانگین و بیشینهٔ حرارت تولیدشده در نقاط داغ تراشه است که با افزایش اندکی در طول سیم مصرفی همراه است. یک روش فراابتکاری چند هدفه بر مبنای الگوریتم ژنتیک در مرجع [۸] ارائه شده است که در آن شبکه‌های سه‌بعدی موجود در مدار (یعنی یک منبع و ماجول‌هایی که به خروجی این منبع وصل‌اند) روی لایه‌های تراشهٔ سه‌بعدی به‌صورت همزمان پیاده‌سازی می‌شوند. در این پیاده‌سازی سعی بر آن است که محل TSVها به‌گونه‌ای انتخاب شوند که نخست، در مکانی با چگالی توان بالا و تراکم سیم‌بندی کم باشند؛ دوم، طول سیم‌بندی شبکه‌های سه‌بعدی حداقل شود. در مرجع [۹] روش پیاده‌سازی دیگری بر مبنای الگوریتم ژنتیک ارائه شده است که در آن ابتدا با شروع از پایین‌ترین لایه، بلوک‌ها به لایه‌ها نسبت داده می‌شوند. با انتخاب توالی‌های مختلف، بخش‌بندی‌های مختلفی ایجاد می‌شوند که هرکدام یک کروموزم خواهند بود. در مرحلهٔ بهینه‌سازی، الگوریتم ژنتیک با اعمال عملگرهای ژنتیکی روی کروموزم‌ها جواب‌های مطلوب را جستجو می‌کند. تابع هزینه در این الگوریتم ترکیبی از چگالی توان و تعداد TSVها است.

در مرجع [۱۰]، رویکرد متفاوتی برای حل مسئلهٔ

پیاده‌سازی روی تراشه‌های سه‌بعدی بر مبنای الگوریتم تبرید شبیه‌سازی‌شده[۴] یا به اختصار SA ارائه شده است. در این روش، با شروع از یک پیاده‌سازی تصادفی، یک بلوک در بین لایه‌ها جابه‌جا می‌شود و این جابه‌جایی در صورتی پذیرفته می‌شود که به کاهش تابع هزینه منجر شود. در صورتی که حرکت مدنظر باعث افزایش تابع هزینه شود، با احتمال کوچکی پذیرفته خواهد شد. تابع هزینه در این رویکرد بر مبنای کاهش تعداد TSVها و متعادل‌کردن سطح مصرفی لایه‌ها تعریف شده است. در روش FSA مطرح‌شده در مرجع [۱۱]، با استفاده از رهیافت نیروی کشسانی فنر، مدلی احتمالاتی برای افزایش نرخ همگرایی SA ارائه شده است. در این روش اهداف اصلی، کاهش تعداد TSVها و متعادل‌کردن سطح مصرفی لایه‌ها هستند.

تمام روش‌های ذکرشده در بالا دو نقص اساسی دارند؛ سرعت رسیدن به جواب بهینه و حافظهٔ مورد نیاز در اجرای فرایند بهینه‌سازی. الگوریتم ژنتیک برای حل مسائلی با اندازهٔ بزرگ کارا نیست. در این الگوریتم، برای رسیدن به جواب بهینه (یا شبه‌بهینه با کیفیت بالا) باید تعداد کروموزم‌ها نسبتاً زیاد باشد؛ اما این مسئله به مصرف مقدار زیادی حافظه منجر خواهد شد و اعمال عملگرهای ژنتیکی نیز به زمان زیادی نیازمند است. به همین دلیل، روش‌های ارائه‌شده بر مبنای الگوریتم ژنتیک مقیاس‌پذیر نیستند و برای مدارهای با اندازهٔ بزرگ کاربردی نیستند. روش‌های ارائه‌شده بر مبنای SA، به‌دلیل استفاده از فقط یک حل اولیه و ایجاد تغییرات کوچک در ساختار آن مشکل حافظهٔ مصرفی بالا را حل کرده‌اند. مشکل مقیاس‌پذیری در روش‌های مبتنی بر SA همچنان وجود دارد. دلیل این امر، نیاز به فرایند سردسازی بسیار طولانی و طی‌کردن تعداد زیادی تغییرات کوچک در هر گام دمایی به‌منظور رسیدن به جواب بهینه برای مدارهای بسیار بزرگ است.

در روند پیاده‌سازی ارائه‌شدهٔ پیشنهادی، مراحل بخش‌بندی و جانشانی از هم جدا شده‌اند که به کاهش اندازهٔ مسئلهٔ بهینه‌سازی منجر می‌شود. روش فراابتکاری SA [۱۲] برای یافتن حالت بهینه در مراحل بخش‌بندی و جانشانی استفاده شده است. هدف در مرحلهٔ بخش‌بندی، کاهش تعداد TSVها و در مرحلهٔ جانشانی، کاهش طول مصرفی برای اتصال ماجول‌ها روی هرکدام از لایه‌ها است. مرحلهٔ مسیردهی براساس الگوریتم ابتکاری مسیریاب[۵] انجام می‌شود که در ابزارهای طراحی مدارهای مجتمع به‌صورت وسیع استفاده شده است [۱۳].

در ادامهٔ مقاله، در بخش دوم پیش‌نیازهای لازم و در بخش سوم، روش پیشنهادی بیان شده است. در بخش چهارم، نتایج حاصل از روش پیاده‌سازی بررسی شده و در بخش پنجم، نتیجه‌گیری مربوط به این تحقیق بیان شده است.

۲- پیش‌نیازها

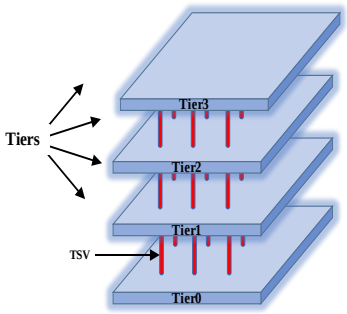
در این بخش ساختار مدارهای مجتمع سه‌بعدی و ساختار TSVها به‌صورت اجمالی معرفی می‌شوند. توضیحاتی دربارهٔ بخش‌بندی، جانشانی و مسیردهی ارئه خواهند شد و درنهایت الگوریتم SA معرفی می‌شود.

۲-۱- ساختار مدارهای مجتمع سه‌بعدی

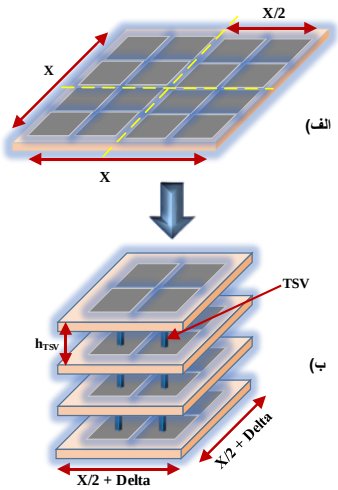
ساختار کلی یک مدار مجتمع سه‌بعدی در شکل (۱) نشان داده شده است. همان‌طور که دیده می‌شود، لایه‌های فعال به‌صورت انباشته[۶] روی یکدیگر قرار می‌گیرند و ایجاد اتصالات عمودی بین لایه‌ای با استفاده از TSVها انجام می‌شود.

یکی از مهم‌ترین مزیت‌های مدارهای مجتمع سه‌بعدی، کاهش طول و عرض تراشه (و به تبع آن مساحت تراشه) در مقایسه با معادل دوبعدی است. میزان این کاهش، به شدت به تعداد TSVها وابسته است.

برای درک بهتر این موضوع، شکل (۲) را در نظر بگیرید. همان‌طور که در شکل (۲) قسمت (الف) دیده می‌شود، تراشهٔ دوبعدی، دارای طول و عرض X است. حال اگر بخواهیم منابع موجود در این تراشهٔ دوبعدی را در یک تراشهٔ سه‌بعدی با چهار لایه توزیع کنیم (شکل ۲، قسمت ب)، سطح تراشهٔ دوبعدی به چهار قسمت مساوی تقسیم می‌شود که طول و عرض هرکدام به نصف حالت قبلی (یعنی $X/2$) کاهش می‌یابد.



شکل (۱): نمایی کلی از یک مدار مجتمع سه‌بعدی یا چهار لایه



شکل (۲): الف) تراشهٔ دوبعدی، ب) تراشهٔ سه‌بعدی معادل با چهار لایه

قرارگرفتن TSVها در ساختار تراشه‌های سه‌بعدی، نیازمند در نظر گرفتن فضای کافی در محل قرارگیری آنها است. به همین دلیل، همان‌گونه که در شکل (۲) قسمت (ب) مشاهده می‌شود، اندازهٔ طول و عرض تراشهٔ سه‌بعدی نسبت به مقدار مورد انتظار *X/2* بیشتر است. این مقدار افزایش با مشخصهٔ *Delta* نشان داده شده است. با افزایش تعداد TSVها در سطح تراشه، مقدار *Delta* بیشتر می‌شود. بیشینهٔ فاصلهٔ منهتن[۷] (از نقطه A به نقطه B) در تراشهٔ دوبعدی (شکل ۲، قسمت الف) مطابق معادلهٔ (۱) محاسبه می‌شود:

$$MD_{max,\ 2D} = X + X \qquad (۱)$$

همچنین، بیشینهٔ فاصلهٔ منهتن در تراشهٔ سه‌بعدی (شکل ۲، قسمت ب) مطابق معادله (۲) محاسبه می‌شود:

$$MD_{max,\ 3D} = (n-1) \times h_{TSV} + 2\times (X/2 + delta) \qquad (۲)$$

در این معادله، h_{TSV} برابر با ارتفاع TSV و n برابر با تعداد لایه‌های تراشهٔ سه‌بعدی است. هرچند ارتفاع TSV حدود ۲۰ میکرومتر است [۱٤]، با توجه به مشخصه‌های ذاتی TSV (در بخش‌های بعدی به آن اشاره خواهد شد)، در مقایسه با سیم‌های استاندارد، تأخیر بسیار کمتری دارد [۸]؛ بنابراین، سهم عبارت اول در معادله (۲) در مقدار تأخیر مسیر، قابل صرف‌نظر کردن است. به‌صورت تقریبی، نسبت تأخیر مسیرها با بیشترین طول در تراشهٔ سه‌بعدی و دوبعدی با استفاده از معادله (۳) محاسبه می‌شود:

$$MD_{max}(3D)/MD_{max}(2D) = 2/(1+2\times Delta/X) \qquad (۳)$$

با توجه به رابطه (۳)، افزایش تعداد TSVها که به افزایش مقدار *Delta* منجر می‌شود، سبب کاهش مزیت تراشهٔ سه‌بعدی نسبت به تراشهٔ دوبعدی می‌شود. نتیجهٔ کلی این است که باید کنترل جدی بر تعداد TSVها صورت بگیرد.

همان‌طور که گفته شد، TSVها به‌عنوان اتصالات عمودی در تراشه‌ها استفاده می‌شوند و توان تلفاتی کمتر و تأخیر کمتر در مقایسه با سیم‌های ارتباطی بین تراشه‌ها دارند.

مطالعات اخیر در حوزهٔ TSVها بیشتر روی طراحی بر مبنای قابلیت اطمینان[۸] و مقرون‌به‌صرفه[۹] بودنِ ساخت آنها تمرکز دارند. فاکتورهای اساسی تأثیرگذار بر نحوهٔ عملکرد یک TSV عبارت‌اند از ماده پرکننده[۱۰] (ماده‌ای که برای پرکردن حفرهٔ TSV استفاده می‌شود)، طول، شکل و قطر TSVها. از رایج‌ترین مواد پرکنندهٔ TSV می‌توان مس، تنگستن و پلی کریستال سیلیکون[۱۱] را نام برد که در زمان حاضر مس از همه رایج‌تر است. به‌تازگی نانو اتصال‌گرهای بر مبنای گرافن[۱۲]، رقیبی برای مس مطرح شده‌اند. دلیل واردشدن این فناوری به عرصهٔ رقابت با مس، خواص و رفتار یکتایی است که ازنظر الکتریکی، مکانیکی، شیمیایی و گرمایی دارد [۱۵]. TSVها در شکل‌های گوناگونی مانند دایره‌ای، حلقوی، مربعی، مستطیلی و مخروطی ساخته شده‌اند؛ اما ساختار فیزیکی یک TSV همچنان یک چالش

به حساب می‌آید و پژوهشگران حوزهٔ نیمه‌هادی و نانوالکترونیک، توجه بیشتری به این زمینه دارند.

بر طبق نقشه‌راه اعلام‌شده توسط *ITRS 2015* برای مقطع زمانی ۲۰۱۹ تا ۲۰۲۲، قطر یک TSV برابر با ۲-۰/۵ میکرومتر، گام‌های[۱۳] آن برابر با ۴-۱ میکرومتر و ارتفاع آن برابر با ۲۰-۵ میکرومتر است.

۲-۲- بخش‌بندی، جانشانی و مسیردهی

فرض کنید گراف مدار $G(V, E)$ را در اختیار داریم که در آن، سلول‌های منطقی (معادل همان گیت‌های منطقی) و ارتباطات داخلی (یا همان سیم‌بندی بین المان‌ها) به‌ترتیب با رأس‌ها و یال‌ها معادل شوند.

در گراف مدار، $V = \{v_1, v_2, v_3, \dots, v_n\}$ مجموعه رأس‌ها و $E = \{e_1, e_2, e_3, \dots, e_n\}$ مجموعه یال‌ها هستند. در مسئلهٔ بخش‌بندی، هدف تقسیم‌کردن مجموعه رأس‌های V از گراف $G(V, E)$ به k زیرگراف $V_1, V_2, V_3, \dots V_k$ است؛ به طوری که داشته باشیم:

$$|V_1| = |V_2| = \dots = |V_k|,$$
$$V_i \cap V_j = \Phi, \quad i \neq j$$

تابع هزینه در مسئلهٔ بخش‌بندی، براساس تعداد یال‌های متصل‌کنندهٔ بخش‌های مختلف تعریف می‌شود که اندازهٔ برش[۱۴] نامیده می‌شود. مسئلهٔ بخش‌بندی گراف‌ها از دیدگاه ریاضی به‌عنوان یک مسئلهٔ NP-hard شناخته شده است.

مراحل بعد از بخش‌بندی، جانشانی و مسیردهی[۱۵] هستند. جانشانی عبارت است از پیداکردن بهترین مکان برای هر رأس (بلوک)، به طوری که در هر مختصات مکانی، بیش از یک رأس قرار نگیرد. یک جانشانی ضعیف، باعث اشغال مساحتی بیشتر در سطح تراشه و همچنین افت کیفیت در مرحلهٔ مسیردهی می‌شود. در فرایند جانشانی زیرگراف‌ها روی لایه‌های تراشهٔ سه‌بعدی، بلوک‌های مربوطه به‌گونه‌ای جانشانی می‌شوند که کمترین طول سیم مصرفی مصرف شود.

مسئلهٔ مسیردهی در تراشه‌های دیجیتال، عبارت است از اختصاص‌دادن شبکه[۱۶] (منظور از شبکه، مجموعه اتصالات خروجی یک ماجول منبع به تمام ماجول‌های مقصد مربوط به آن است) به منابع بخش اتصالات، به گونه‌ای که هیچ‌یک از منابع با بیش از یک شبکه به اشتراک گذاشته نشود. این مرحله به کیفیت مرحلهٔ قبل از خود، یعنی جانشانی وابسته است [۱۶].

۲-۳- الگوریتم SA

الگوریتم SA یکی از الگوریتم‌های فراابتکاری است که کاربرد زیادی در طراحی‌های فیزیکی دارد. الگوریتم SA یک روش بهینه‌سازی است که از فرایند خنک‌سازی آهستهٔ فلزات الهام گرفته شده است. در این فرایند با کاهش تدریجی حرکت اتم‌ها، قرارگیری نهایی آنها به‌گونه‌ای است که وضعیت کمترین انرژی پایدار در شبکه حاصل شود [۱۲]. شبه‌کد مربوط به SA، در الگوریتم (۱) مشاهده می‌شود.

الگوریتم (۲-۱): شبه‌کد الگوریتم SA

```
Input: G (V, E)
Output: Optimized solution

1: Initialization
2: S ⟵ S_init
3: Cost_Current = Cost(S)
4: while T > T_min
5:     while stopping criterion is not met yet
6:         neighbor(S')
7:         Cost_new = Cost_Current
8:         ΔCost = Cost_new - Cost_Current
9:         if (ΔCost < 0)
10:            S ⟵S'
11            Cost_new = Cost_Current
12:        else
13:            r ⟵ Rand (0,1)
14:            if (r < exp(-ΔCost/T))
15:                S ⟵ S'
16:                Cost_new = Cost_Current
17:            end if
18:        end if
19:    end while
20:    T = G (T, K)
21: end while
```

عملکرد این الگوریتم (به‌صورت خلاصه) بدین صورت است: در ابتدا مشخصه‌های اساسی (دمای اولیه، ضریب خنک کنندگی و ...) مقداردهی اولیه می‌شوند و یک راه‌حل مسئله به‌صورت تصادفی تولید می‌شود. هزینهٔ این راه‌حل اولیه، محاسبه ($Cost_{Current}$) و الگوریتم وارد یک حلقه می‌شود. در هر تکرار از حلقه، تغییر کوچکی به‌صورت

تصادفی در حل مرحلهٔ قبل ایجاد می‌شود ($neighbor(S')$). اگر این تغییر، در راستای کاهش هزینه باشد (یعنی اختلاف تابع هزینهٔ جدید و قبلی، منفی باشد)، پذیرفته و تابع هزینه با توجه به این تغییر و تأثیرات آن به‌روز می‌شود؛ اما اگر این تغییر به کاهش مقدار تابع هزینه منجر نشود، به احتمال کمی مطابق خطوط ۱۳ تا ١٦ از الگوریتم پذیرفته می‌شود. این راه‌کار، سبب جلوگیری از گیرافتادن الگوریتم در یک کمینه محلی خواهد شد. در پایان بدنهٔ حلقه، مشخصهٔ دما بر طبق یک تابع خاص (خط ۲۰ از الگوریتم) به‌روز می‌شود. این تابع و فرایند خنک‌سازی، نباید آنقدر زود اتفاق بیافتد که مانعی برای کاهش تابع هزینه به اندازهٔ کافی شود و نیز نباید آنقدر با سرعت کمی پیش برود که زمان اجرا به‌صورت غیرقابل قبولی افزایش یابد. این فرایند به‌صورت مستقیم از مشخصه‌های الگویتم، ازجمله دمای اولیه، دمای نهایی و ضریب خنک‌سازی تبعیت می‌کند.

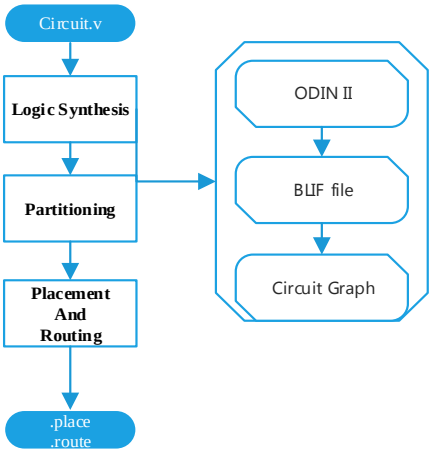
۳-فرایند پیاده‌سازی پیشنهادی

در روش پیاده‌سازی توسعه داده شده در این مقاله، ابتدا توصیف سخت‌افزاری وریلاگ مدار به سطح گیت تبدیل می‌شود. سپس گراف جهت‌دار مدار با استفاده از توصیف سطح گیت استخراج می‌شود. گراف مدار با استفاده از رهیافت فراابتکاری، به N_L بخش (زیرگراف) با اندازهٔ مساوی تقسیم می‌شود؛ به گونه‌ای که تعداد برش‌های بین آن‌ها کمینه باشد. سپس هر زیرگراف روی لایهٔ مربوطه با استفاده از الگوریتم فراابتکاری SA جانشانی می‌شود. ایجاد اتصالات (سیم‌بندی‌های لازم در شبکه‌های موجود در لایه‌ها) در مرحلهٔ مسیردهی انجام می‌شود.

شمای کلی روش پیشنهادی در شکل (۳) نشان داده شده است. در ابتدا توصیف وریلاگ مدار مدنظر وارد فرایند می‌شود. در مرحلهٔ دوم با استفاده از سنتزگر odin_II [۲۳]، کد وریلاگ به یک توصیف سطح گیت تبدیل می‌شود. این سنتزگر، مستقل از فناوری و خروجی آن یک فایل *BLIF*[۱۷] است. فایل *BLIF* از ورودی - خروجی‌های مدار و لیست گیت‌های تشکیل‌دهندهٔ مدار ساخته شده است. برای هرکدام از گیت‌ها، اسامی پایه‌های ورودی و خروجی مشخص شده است که با استفاده از اسامی گره‌ها می‌توان گراف مدار را استخراج کرد. در گراف مدار، هرکدام از گیت‌ها نقش یک رأس را دارند و یال‌های خروجی هر رأس، به رأس‌هایی (گیت‌هایی) وصل می‌شوند که ورودی‌های هم‌نام با گیت مدنظر را دارند. همچنین، در فایل *BLIF* برای هرکدام از گیت‌ها با استفاده از جدول صحت (*LUT*)، نوع گیت‌ها (*NAND, XOR, ...*) مشخص شده است.

در این پژوهش، از روش طراحی شبه - سفارشی[۱۸] استفاده شده است که در آن سطح دوبعدی هر لایهٔ تراشه به تعدادی سطر با ارتفاع یکسان تقسیم‌بندی می‌شود. گیت‌های منطقی استفاده‌شده در فایل *BLIF* مدار، در محدودهٔ این سطرها پیاده‌سازی می‌شوند. بین دو سطر متوالی از سطح لایه، فضایی برای انجام مسیردهی در نظر گرفته می‌شود که کانال‌های مسیردهی نامیده می‌شوند.

خروجی مرحلهٔ اول، گراف جهت‌دار مدار است که در مرحلهٔ بخش‌بندی استفاده می‌شود. زیرگراف‌های ایجادشده در مرحلهٔ جانشانی و سپس مرحلهٔ مسیردهی روی لایه‌ها پیاده‌سازی می‌شوند. نتیجهٔ نهایی فرایند، تولید دو فایل *.place* و *.route*. برای هر لایه است که دربرگیرندهٔ اطلاعات مربوط به چینش ماجول‌ها روی لایه‌ها و همچنین نحوهٔ ایجاد اتصالات بین آنها است. در ادامه هرکدام از این مراحل تشریح می‌شوند.



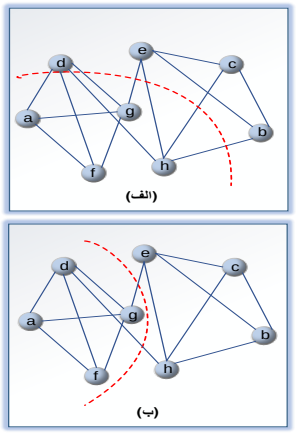
شکل (۳): نمایی از فلوی کلی روش پیشنهادی

۱-۳- بخش‌بندی

در حالت کلی هدف از بخش‌بندی، تقسیم یک گراف به دو یا چند زیرگراف است که این تقسیم‌بندی با اهدافی متفاوت و همچنین با الگوریتم‌های مختلفی می‌تواند صورت گیرد. در این مقاله، هدف از بخش‌بندی تولید N_L زیرگراف به‌صورتی است که تا حد امکان کمترین ارتباط بین لایه‌ها (تعداد TSV) ایجاد شود و توازن در تعداد رأس‌های مربوط به زیرگراف‌ها به وجود آید.

در شکل (٤)، دو بخش‌بندی متعادل (هرکدام از بخش‌ها شامل ٤ رأس‌اند) نشان داده شده است. این بخش‌بندی‌ها براساس تعداد اتصالات ایجادشده بین دو بخش، بخش‌بندی بد (قسمت الف) و بخش‌بندی عالی (قسمت ب) نام‌گذاری شده‌اند. تعداد TSVهای لازم برای بخش‌بندی بد و بخش‌بندی عالی به‌ترتیب ۸ و ۲ است که نشان‌دهندهٔ اهمیت بالای مرحلهٔ بخش‌بندی در کاهش تعداد TSVها است.

همان‌طور که در بخش مقدمه اشاره شد، در فرایند پیاده‌سازی پیشنهادی، بخش‌بندی براساس الگوریتم فراابتکاری SA انجام می‌شود. شبه‌کد مربوطه، در الگوریتم (۲) ارائه شده است. در این الگوریتم، ورودی یک گراف جهت‌دار با $2n$ رأس است (در صورتی که تعداد رئوس گراف مدار فرد باشد، یک رأس مجازی به آن اضافه می‌شود) و خروجی آن شامل فایل‌هایی است که بلوک‌های مربوط به هر لایه را دربردارد.



شکل (٤): گراف G با دو رویکرد متفاوت در بخش‌بندی. در قسمت الف) سایز برش ۸ است و در قسمت ب) سایز برش ۲ است.

در خط ۱، گراف ورودی به‌صورت تصادفی به N_L بخش (لایه) تقسیم می‌شود. سپس تابع Setup() اجرا می‌شود. در خط سوم، مشخصه‌های الگوریتم مقداردهی می‌شوند. این مشخصه‌ها روی عملکرد الگوریتم بسیار تأثیرگذارند. در خط چهارم و پنجم، تابع هزینهٔ اولیه برآورد می‌شود. در ادامه، اجرای حلقهٔ اصلی (خط ٦ تا ۲۷) ادامه می‌یابد تا زمانی که شرط حلقه (خط ٦) برقرار باشد و به محض نقض شرط حلقه یا عدم پیشرفت[19]، الگوریتم خاتمه می‌یابد. همان‌گونه که در شکل (۵) دیده می‌شود، عدم پیشرفت زمانی رخ می‌دهد که تابع هزینه برای چند گام پیاپی بدون تغییر باقی بماند یا میزان تغییرات آن کم باشد. در این شرایط، به‌دلیل صرف زمان زیاد تا رسیدن به یک جواب مطلوب، بهتر است الگوریتم با جواب فعلی خاتمه یابد.

در هر گام دمایی، یک حلقهٔ محلی با N گام اجرا می‌شود (خط ۸ تا ۲۳). مقدار N، در خط ۷ از الگوریتم تعیین می‌شود ((Setting Local Parameters(alpha, N)).

در این حلقهٔ محلی، در ابتدا تابع Selection() اجرا می‌شود. با اجرای این تابع، دو رأس برای جابه‌جایی بین دو یا چند لایه به‌صورت تصادفی انتخاب می‌شوند. در خط ۱۰ و ۱۱ تابع هزینهٔ جدید ناشی از این جابه‌جایی و اختلاف آن با تابع هزینهٔ فعلی محاسبه می‌شود. اگر این مقدار منفی باشد، یعنی جابه‌جایی انجام‌شده در راستای کاهش هزینه بوده است و این جابه‌جایی پذیرفته می‌شود (خط ۱۲ تا ١٤)؛ اما اگر اختلاف بین تابع هزینهٔ فعلی و تابع هزینهٔ جدید منفی نباشد، با احتمال کمی جابه‌جایی پذیرفته می‌شود (خط ۱۵ تا ۱۹). هدف از این کار جلوگیری از گیرافتادن الگوریتم در *مینیمم محلی* (شکل ۵) است. در ادامه توابع موجود در الگوریتم شرح داده می‌شوند.

شبه‌کد مربوط به تابع Setup() در الگوریتم (۳) نشان داده شده است. گراف مدار توسط تابع دریافت می‌شود و اطلاعات لازم توسط سه زیرتابع Vertice_Gain()، Adjacency_matrix()، و Connectivity() استخراج می‌شوند.

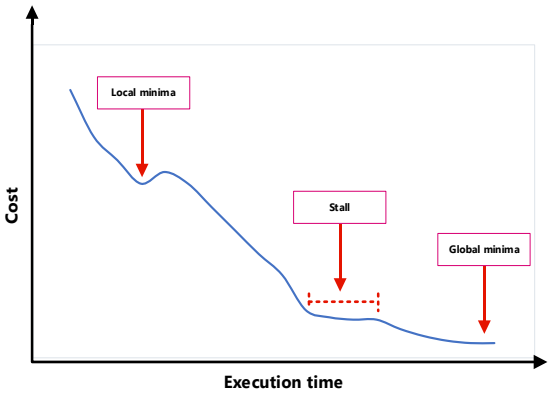
در خط ۱ با استفاده از تابع Vertice_Gain()، بهرهٔ همه رأس‌های گراف محاسبه می‌شود. بهرهٔ رأس i با استفاده از معادله (٤) به دست می‌آید:

$$D_i = E_i - I_i \qquad (٤)$$

الگوریتم (۲): الگوریتم بخش‌بندی بر مبنای الگوریتم SA

```
Input: G (V, E)
Ensure: Optimized partitioned Graph to l tiers

Randomly_partitioning( )
Setup( )
Setting parameters (T, T_min)
S ← S_init
Cost_Current ← Cost(S)
while T > T_min
    Setting local parameters (alpha, N)
    for i = 0: N
        Selection(S')
        Cost_new ← Cost_Current
        ΔCost = Cost_new - Cost_Current
        if (ΔCost < 0)
            S ← S'          // updating
            Cost_Current ← Cost_new
        else
            r ← Rand (0, 1)
            if (r < exp(-ΔCost/T))
                S ← S'      // updating
                Cost_Current ← Cost_new
            end if
        end if
    end for
    if (stall)
        break;
    else
        T = α × T
    endif
end while
```



شکل (۵): نمایی از مقادیر یک تابع هزینه در گذر زمان

الگوریتم (۳-۲): فاز Setup

```
Input: G (V, E)
Ensure: Extract the required data
Vertice_Gain( )
Adjacency_matrix( )
Connectivity( )
```

در این معادله، E_i ارزش خارجی[۲۰] و I_i ارزش داخلی[۲۱] برای رأس iام است. ارزش داخلی برای رأس i، برابر تعداد یال‌هایی است که این رأس با رأس‌های داخلی بخش مربوط به i دارد و ارزش خارجی رأس i برابر با تعداد یال‌هایی است که این رأس را به بخش‌های دیگر متصل می‌کنند.

با استفاده از تابع Adjacency_matrix()، ماتریس مجاورت برای گراف اصلی ساخته می‌شود. ماتریس مجاورت برای گراف $G(V, E)$ با n رأس، ماتریسی متقارن با ابعاد n در n به شکل زیر است:

$$M = [c_{ij}] = \begin{cases} c_{ij} = 1, & \text{i and j are connected} \\ c_{ij} = 0, & \text{i and j are not connected} \end{cases}$$

این ماتریس نقش مهمی در سرعت‌بخشیدن به الگوریتم دارد. با ایجاد این ماتریس، هنگام نیاز به محاسبهٔ وزن یال‌های متصل بین دو رأس، بدون محاسبهٔ مجدد در هر تکرار و بدون درگیرکردن پردازنده در یک جستجوی طولانی، به وزن مدنظر می‌توان دست یافت.

بعد از این مرحله (در خط سوم)، تابع Connectivity() اجرا می‌شود. با اجرای این تابع، برای هر رأس یک لیست ایجاد می‌شود که نشان‌دهندهٔ شماره رأس‌هایی است که به آن متصل‌اند. این تابع، باعث کاهش چشمگیر زمان اجرای الگوریتم (۲)، در خطوط ۱۳ و ۱۸ می‌شود.

زمانی که دو رأس از دو لایهٔ مختلف (V_i و V_j) توسط تابع Selection() برای جابه‌جایی انتخاب می‌شوند، تابع هزینهٔ مربوطه و اختلاف آن با تابع هزینهٔ فعلی باید محاسبه شود تا مجاز بودن یا نبودن جابه‌جایی تعیین شود. برای محاسبهٔ تابع هزینه، فرض کنید ارزش داخلی رأس‌های مربوطه، برابر با I_i و I_j و ارزش خارجی آنها برابر با E_i و E_j باشند، در این صورت تابع هزینه از معادله (٤) به دست می‌آید که در آن C_{ij} برابر با تعداد یال‌های مشترک بین دو رأس V_i و V_j است.

$$Cost = (I_i + I_j) - (E_i + E_j) + (2.\ C_{ij}) \qquad (٤)$$

۲-۳- جانشانی و مسیردهی

مجموعه رأس‌های $V1, V2, V3, \ldots Vn$ و مجموعه

مکان‌های $P1, P2, P3, \ldots Pk$ $(n \leq k)$ را روی سطح تراشه در نظر بگیرید. مسئلهٔ جانشانی یعنی اختصاص‌دادن بهترین مکان به هر رأس، به طوری که دو رأس در یک مکان یکسان قرار نگیرند [۱۶]. کیفیت انجام جانشانی در تعیین مقدار سطح مصرفی و سرعت مدار پیاده‌سازی‌شده نقشی اساسی دارد.

مسئلهٔ جانشانی ازنظر ریاضی، یک مسئلهٔ *NP-hard* است. به همین دلیل، حل این مسئله با استفاده از الگوریتم‌های فراابتکاری، بهترین رهیافت ممکن است. در این مقاله، الگوریتم *SA* برای حل مسئلهٔ جانشانی استفاده می‌شود.

با توجه به مشخص‌شدن گیت‌های هر لایه در مرحلهٔ بخش‌بندی، فرایند جانشانی از لایهٔ اول شروع می‌شود. ابتدا یک جانشانی به‌صورت کاملاً تصادفی تولید می‌شود. با توجه به طراحی نیمه‌سفارشی، گیت‌هایی که در هر ردیف جانمایی می‌شوند، ممکن است عرض‌های مختلفی داشته باشند؛ درحالی‌که ارتفاع همه آنها برابر است. سپس برای هر دما (T)، تعداد مشخصی جابه‌جایی تصادفی بین مکان گیت‌ها صورت می‌گیرد. اولین معیار برای پذیرفته‌شدن یک جابه‌جایی این است که پس از انجام جابه‌جایی، همپوشانی بین ناحیهٔ اشغال‌شده توسط گیت‌ها ایجاد نشود. این همپوشانی معمولاً هنگامی به وجود می‌آید که یک گیت با عرض بزرگ‌تر با گیتی با عرض کوچک‌تر جابه‌جا می‌شوند.

در صورت نبود همپوشانی، پذیرفته‌شدن هر جابه‌جایی مشروط به مقدار تابع هزینهٔ جانشانی در روابط (۵) تا (۷) است:

$$\text{Cost\_W}_{3D} = \sum_{n=1}^{N_{net}} \text{K(n)} \left[XW \times xspan(n) + YW \times yspan(n) \right] \quad (۵)$$

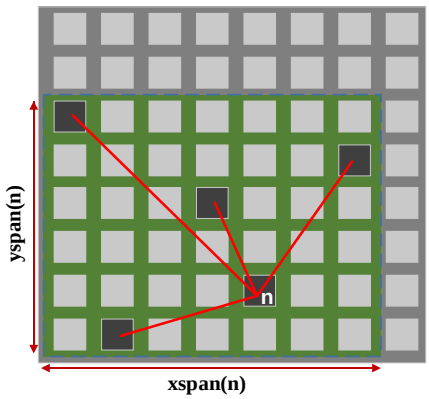
$$\text{Cost\_T}_{3D} = \sum_{\forall i,j \subset circuit} delay(\Delta xij, \Delta yij) \times \text{t\_p} \quad (۶)$$

$$\text{Cost}_{3D} = \text{V} \times \text{Cost\_T}_{3D} + (1 - \text{V}) \times \text{Cost\_W}_{3D} \quad (۷)$$

در رابطه (۵)، $xspan(n)$ و $yspan(n)$ برابر با طول گذر در جهت X و Y در Bounding Box کمینهٔ مربوط به شبکه n است؛ برای مثال، در شکل (۶) شبکهٔ تشکیل‌شده از بلوک n و بلوک‌های متصل به آن نشان داده شده است. XW, YW وزن‌هایی برای تعیین اولویت شبکه‌ها در بهینه‌سازی‌اند؛ برای مثال، شبکه‌های بحرانی دارای بالاترین اولویت‌اند و حداکثر مقدار XW و YW برای آنها لحاظ می‌شود. همچنین، $K(n)$ مشخصه‌ای قابل تنظیم در الگوریتم است.

در رابطه (۶)، متغیر t_p ضریب قابل تنظیم الگوریتم و $delay\ (\Delta xij, \Delta yij)$ مقدار تأخیر زمانی بین دو نقطه (x_1, y_1) و (x_2, y_2) را بیان می‌کند. میزان دقیق تأخیر زمانی، تنها بعد از انجام کامل مسیردهی محاسبه می‌شود؛ بنابراین، در مرحلهٔ جانشانی، یک درخت استینر[۲۲] [۲۴] تقریبی بین منبع شبکه و مقاصد در نظر گرفته می‌شود. سپس براساس مدل تأخیر المور[۲۳] [۲۵] مقدار تأخیر تقریبی شبکه استخراج می‌شود. تابع هزینهٔ کلی در رابطه (۷) بیان شده است. ضریب V سهم هزینهٔ زمان (Cost_T_{3D}) و هزینهٔ طول سیم (Cost_W_{3D}) در تابع هزینهٔ نهایی را تعیین می‌کند. اگر این ضریب، مقدار بزرگ (یعنی نزدیک به یک) داشته باشد، به تابع هزینهٔ زمان اهمیت بیشتری داده می‌شود. همچنین اگر این مقدار نزدیک به صفر باشد، هزینهٔ طول سیم غالب خواهد شد. در الگوریتم پیشنهادی ما با آزمودن مقادیر متفاوت، این ضریب برابر با ۰/۷ در نظر گرفته شده است.

جانشانی لایه‌های بعدی، به‌ترتیب و با رویکردی مشابه انجام می‌گیرد. شایان ذکر است به زیرگراف لایهٔ iام تعدادی رأس مجازی اضافه می‌شود که هرکدام بیان‌کنندهٔ TSVهایی‌اند که بعد از جانشانی لایه‌های پایین‌تر وارد لایهٔ iام می‌شوند.



شکل (۶): کمینه bounding box

در مرحلهٔ آخر، اتصالات لازم بین بلوک‌های موجود، با استفاده از منابع و شبکهٔ مسیردهی انجام می‌شود. در این مرحله، الگوریتم پیشنهادی بر مبنای الگوریتم مسیریاب [۱۳] گسترش داده شده است. این دسته از الگوریتم‌ها، از رهیافت پیداکردن کوتاه‌ترین مسیر[۲۴] در تئوری گراف بهره می‌برند. هدف در این رهیافت، یافتن مسیری بین دو رأس از گراف است؛ به گونه‌ای که مجموع وزن یال‌های مربوطه به کمترین مقدار ممکن برسد. هستهٔ اصلی الگوریتم پیشنهادی در حل مسئله کوتاه‌ترین مسیر، الگوریتم دایکسترا[۲۵] [۱۷] است.

تابع هزینهٔ کلی در مرحلهٔ مسیردهی، از رابطه (۸) محاسبه می‌شود:

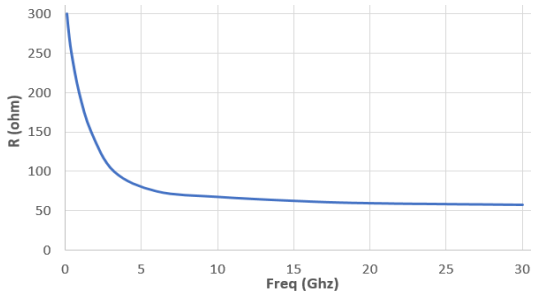
$$Cost_n = Criticality\ (i, j) \times delay_n \quad (۸)$$ [1]

$$Criticality\ (i, j) = \frac{Dij}{Dmax} \quad (۹)$$

در این رابطه، *Criticality* از رابطه (۹) محاسبه می‌شود که در آن، *Dij* برابر با ماکزیمم تأخیر مسیرِ ترکیبی از منبع i به مقصد j مربوط به شبکه n است و *Dmax* برابر با تأخیر مسیر بحرانی مدار است. در رابطه (۸)، $Delay_n$ با تقریب بسیار خوب، از روش تأخیر المور محاسبه می‌شود. در مدل المور با توجه به طول سیم‌ها و ویژگی‌های الکتریکی آنها (مقدار مقاومت و اثر خازنی سیم‌ها)، بین منبع یک شبکهٔ مسیردهی و مقاصد آن یک شبکهٔ RC معادل ایجاد می‌شود و درنهایت با استفاده از قواعد توسعه داده شده توسط المور، تأخیر بین منبع و هرکدام از مقاصد محاسبه می‌شود. در مدل تأخیر المور، برای TSVها از رابطه (۱۰) استفاده می‌کنیم:

$$Delay_{TSV}=0.69(R_{down} \cdot C_{down}) + 0.69(0.5(R_{TSV}) + R_{down})\, C_{TSV} + 0.69(R_{TSV}+ R_{up} + R_{down})\, C_{up} \quad (۱۰)$$

در این رابطه، R_{up} و R_{down} مقاومت‌های لایهٔ بالایی و لایهٔ پایینی در صورت وجود و C_{up} و C_{down} خازن‌های لایهٔ بالایی و لایهٔ پایینی‌اند. C_{TSV} خازن و R_{TSV} مقاومت مربوط به TSV هستند. مطالعهٔ موجود در [۱۸] روی مدل الکتریکی TSVها نشان می‌دهد خازن مربوط به TSV (C_{TSV}) بیشترین اثر را بر جای می‌گذارد. همچنین، چون حداکثر مقدار مقاومت TSV (R_{TSV}) برابر با ۱ اهم است [۱۹]، می‌توان از مقاومت TSV در مقایسه با R_{up} و R_{down} چشم‌پوشی کرد. خطوط انتقال سیگنال اصلی دارای عرض ۵۰۰ نانومتر با گام‌های ۱ میکرومتر در نظر گرفته شده‌اند. مشخصهٔ اندازهٔ امپدانس خطوط انتقال، در شکل (۷) نشان داده شده است. مقادیر مقاومت و خازن به‌ترتیب برابر با ۶۰ اهم بر میلی‌متر (60Ω/mm) و ۲۰۰ پیکوفاراد بر میکرومتر (200 pf/µm) است.



شکل (۷): مشخصهٔ اندازهٔ امپدانس برحسب فرکانس در خطوط انتقال

٤- نتایج شبیه‌سازی

فرایند طراحی سه‌بعدی پیشنهادی با استفاده از زبان برنامه‌نویسی C++ در بستر لینوکس توسعه داده شده و روی پردازندهٔ چهار هسته‌ای (۲/۹ گیگاهرتز) و حافظهٔ RAM ۸ گیگابایتی پیاده‌سازی شده است.

در این بخش، از مدارهای معیار MCNC برای اعتبارسنجی شبیه‌سازی‌ها استفاده شده است. ویژگی‌های اصلی این مدارهای معیار، شامل تعداد ورودی – خروجی‌های اصلی و تعداد سلول‌ها (گیت‌های منطقی) در جدول (۲) گزارش شده‌اند. یکی از ویژگی‌های سنتزگر ODIN_II امکان تعیین تعداد ورودی‌های گیت‌های منطقی استفاده‌شده در فایل BLIF است. برای انجام شبیه‌سازی‌ها از سه ساختار با گیت‌های ٤ ورودی (K= ٤)، ٦ ورودی (K= ٦) و ٨ ورودی (K= ٨) استفاده شده است. با توجه به یکسان‌بودن ارتفاع گیت‌ها در سلول‌های استاندارد، عرض گیت‌ها با افزایش تعداد ورودی‌ها افزایش می‌یابد؛ بنابراین، مقایسهٔ نتایج این سه نوع ساختار، تأثیر تغییر اندازهٔ سلول‌ها را نشان می‌دهد.

برای طراحی جانمایی[۲۶] مربوط به گیت‌های منطقی اصلی (NAND, NOR, …) از ترانزیستورهای ساخته‌شده با فناوری ۲۲ نانومتریِ PTM استفاده شده است. مشابه کار انجام‌شده در [۲۶]، یک کتابخانهٔ استاندارد برای گیت‌های پایه‌ای با تعداد ورودی‌های مختلف طراحی شده است. همان‌طور که پیش‌تر اشاره شد، این گیت‌های استاندارد دارای ارتفاع یکسان و عرض‌های متفاوت‌اند.

محاسبهٔ تأخیر انتشار هر گیت یک گام اساسی در انجام تحلیل زمانی است. با استفاده از مدل HSPICE برای ترانزیستورهای PTM [۲۰] و با در نظر گرفتن طراحی گیت‌های پایه‌ای در سطح ترانزیستور، میزان تأخیرها با استفاده از شبیه‌سازی HSPICE محاسبه شده است؛ برای مثال، تأخیر هر گیت NAND دو ورودی به‌طور میانگین ۴۰ پیکوثانیه است. در این نوع گیت، نسبت W/L برای ماسفت‌های نوع N و P به‌ترتیب برابر با ۱۰۰:۲۰ و ۲۵۰:۲۰ است.

همان‌طور که در بخش ۳ اشاره شد، مدل‌سازی تأخیر برای مسیرها با استفاده از مدل RC و رهیافت المور انجام می‌گیرد. مشخصه‌های در نظر گرفته شده در شبیه‌سازی برای TSVها در جدول (۱) آمده‌اند.

مشخصه‌های اصلی در الگوریتم SA عبارت‌اند از ضریب خنک‌کنندگی α، دمای نهایی (T_{min}) و مقدار دمای اولیه (T_0). این مشخصه‌ها باید متناسب با اندازه و پیچیدگی مسئله و توسط کاربر تعیین شوند. انتخاب بهینهٔ این مشخصه‌ها، نقشی اساسی در عملکرد الگوریتم SA و ایجاد مصالحه بین سرعت اجرا و کیفیت نتایج دارد. برای تعیین مقدار مشخصه‌های α و T_0 در فاز بخش‌بندی، شبیه‌سازی‌های متعددی انجام شده است. نتایج شبیه‌سازی‌ها در شکل‌های (۸) تا (۱۱) نشان داده شده‌اند. در این نمودارها محور عمودی، میزان بهبود در تعداد TSV و زمان اجرا را در مدارهای معیار با K=6 نشان می‌دهد. همچنین در نمودارهای ۱۰ و ۱۱، C برابر اندازهٔ بلوک‌ها یا سلول‌های استاندارد در مدار معیار است.

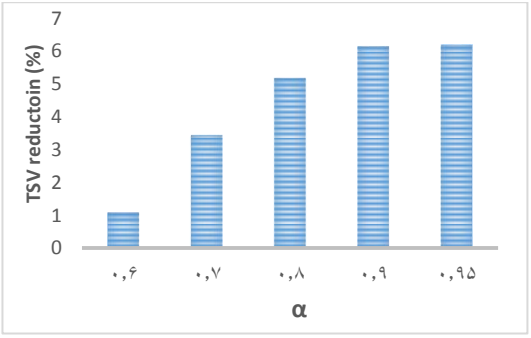
درصد کاهش تعداد TSVها، با افزایش مقدار ضریب خنک‌کنندگی روندی افزایشی دارد. دلیل این امر افزایش تعداد دورهای الگوریتم (افزایش تعداد گام‌های دمایی) با افزایش α است که به یافتن جواب‌های بهتری توسط الگوریتم SA منجر می‌شود. زمان اجرا با افزایش α روندی افزایشی دارد و دلیل آن، اجرای حلقهٔ داخلی الگوریتم SA به تعداد دفعات بیشتر است. روند افزایشی نمودار کاهش تعداد TSVها (شکل ۸) و روند نزولی نمودار بهبود زمان اجرا (شکل ۹) در α = ۰/۹ تلاقی می‌کنند.

مشخصهٔ مهم دیگر، مقدار دمای اولیه (T_0) است که براساس تعداد گیت‌های مدار (C) بیان می‌شود. با افزایش دمای اولیه مقدار کاهش در تعداد TSVهای مورد نیاز، روندی افزایشی دارد. دلیل این روند را می‌توان در افزایش تعداد گام‌های دمایی در صورت انتخاب مقدار T_0 بزرگ دانست که به یافتن جواب‌های بهتر توسط الگوریتم SA منجر می‌شود. همچنین، همین امر منجر به صرف زمان بیشتری توسط الگوریتم برای رسیدن به جواب نهایی خواهد شد. روند افزایشی نمودارِ کاهش تعداد TSVها (شکل ۱۱) و روند کاهشی نمودار بهبود زمان اجرا (شکل ۱۰) در T_0 = C تلاقی می‌کنند؛ بنابراین، بهترین مقادیر ممکن برای α و T_0 به‌ترتیب برابر ۰/۹ و C هستند.

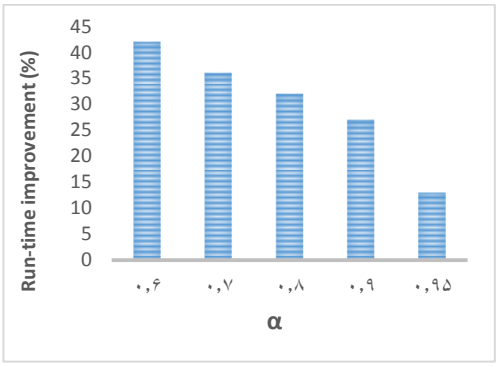
با انجام شبیه‌سازی‌های مشابه، مشخصه‌های الگوریتم SA در فاز جانشانی به‌صورت زیر به دست آمده‌اند: ضریب خنک‌کنندگی α برابر ۰/۹۵، T_{min} (دمای نهایی) برابر ۰/۱ و مقدار T_0 (دمای اولیه) برابر با ربع تعداد سلول‌های هر مدار (ستون آخر از جدول ۲).

جدول (۱) مشخصه‌های مدل TSV

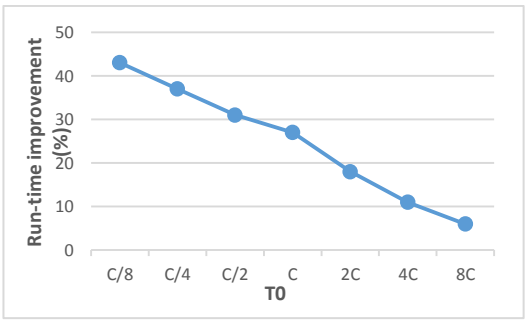
Resistance	0.35 Ω
Capacitance	3 fF
Diameter	2 μm
Pitch	4 μm
Height	20 μm



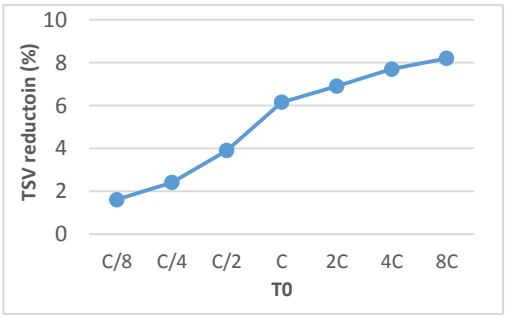
شکل (۸): میزان وابستگی تعداد TSVها به ضریب α



شکل (۹): میزان وابستگی زمان اجرا به ضریب α



شکل (۱۰): میزان وابستگی زمان اجرا به مشخصهٔ T0



شکل (۱۱): میزان وابستگی تعداد TSVها به مشخصهٔ T0

برای انجام مقایسه، دو روش پیاده‌سازی تراشه‌های سه‌بعدی براساس الگوریتم‌های بخش‌بندی FSA و hMetis در نظر گرفته شده‌اند. روش بخش‌بندی hMetis، به‌عنوان یک روش استاندارد در نرم‌افزارهای طراحی تراشه‌های دوبعدی کاربرد وسیعی دارد. همچنین، روش FSA با توجه به سرعت و نتایج قابل قبول، در بخش‌بندی تراشه‌های سه‌بعدی درخور توجه پژوهشگران قرار گرفته است؛ بنابراین، مقایسهٔ عملکرد روش پیشنهادی با این دو روش، مزیت‌های آن را به وضوح بیان می‌کند.

نتایج مربوط به کاهش تعداد TSVها و همچنین کاهش زمان اجرا در مقایسهٔ روش بخش‌بندی پیشنهادی با روش FSA، در جدول‌های (۳، ٤ و ٥) گزارش شده‌اند. این جدول‌ها به‌ترتیب برای حالت‌هایی است که سنتزگر ODIN_II تعداد ورودی‌های مجاز گیت‌های مدار را به‌ترتیب ٤ (K=4)، ٦ (K=6) و ٨ (K=8) در نظر گرفته است.

در ستون آخر از جدول‌های (۳، ٤ و ٥) میزان بهبود پارامترهای زمان اجرا و تعداد TSVها برای هرکدام از مدارها نشان داده شده است. همان‌طور که دیده می‌شود، در جدول (۳) فقط در دو مدار معیار {elliptic, ex5p} تعداد TSVها بهبود نیافته‌اند. میزان بهبود متوسط در این مدارها، ٦/٢٤ درصد برای TSVها و ٢٧/٦٤ درصد برای زمان اجرا است.

جدول (۲): مشخصات مدارهای معیار MCNC با استانداردهای مختلف

Circuit	I/O	Cell #		
		K = 8	K = 6	K = 4
ex5p	71	76	748	1072
des	501	787	810	1847
alu4	22	807	1187	1536
tseng	174	1187	1233	1482
diffeq	103	1211	1308	1934
seq	76	1150	1366	1791
apex2	42	1275	1517	1917
bigkey	460	1395	1177	2193
ex1010	20	850	3103	4608
elliptic	245	3428	3385	4854
s38417	135	4039	4583	7587
s38584.1	343	4041	5461	7579
clma	465	5445	7102	8769

در جدول (٤)، در همه مدارهای معیار بهبود وجود دارد که میانگین آن، ٦/١٥ درصد برای TSV و ٢٧/٧٩ درصد برای زمان اجرا است. همان‌طور که در جدول (٥) مشخص شده است، تنها در یک مورد در تعداد TSVها بهبودی حاصل نشده است. ضمن اینکه در حالت کلی، ٥/٤٩ درصد برای تعداد TSVها و ٢٣/٩٩ درصد برای زمان اجرا بهبود مشاهده می‌شود. به‌طور میانگین، الگوریتم پیشنهادی نسبت به روش FSA به اندازهٔ ٦/١٥ درصد از تعداد TSVها کاسته (که قطعاً به کاهش مساحت هم منجر می‌شود) و زمان اجرای الگوریتم به میزان چشمگیری (٢٧/٧٩ درصد)

کاهش یافته است.

از مقایسهٔ تعداد TSVهای لازم برای پیاده‌سازی هرکدام از مدارهای معیار از جدول‌های ۳ تا ۵ می‌توان دریافت با افزایش اندازهٔ گیت‌ها از ٤ ورودی به ۸ ورودی، تعداد اتصالات بین لایه‌ها روندی کاهشی دارد. دلیل این امر کاهش اندازهٔ مدار سنتزشده ازنظر تعداد گیت‌ها و در پی آن کاهش تعداد یال‌ها در گراف مدار است. نکتهٔ دیگر مربوط به کاهش زمان اجرای الگوریتم‌های بخش‌بندی با افزایش اندازهٔ گیت‌های پایه‌ای است. دلیل اصلی این رفتار، کاهش اندازهٔ گراف مدار و کمترشدن پیچیدگی فضای جستجوی الگوریتم SA است.

در الگوریتم ابتکاری hMetis [۲۱]، گراف مدار ابتدا براساس کاهش تعداد اتصالات مشترک به دو زیرگراف تقسیم می‌شود. سپس طی چند مرحله همین فرایند دوبخشی‌کردن برای زیرگراف‌ها تکرار می‌شود تا زیرگراف‌هایی شامل تنها دو رأس تولید شوند. در فاز دوم، زیرگراف‌های تولیدشده به‌گونه‌ای با هم ادغام می‌شوند که هدف اصلی بخش‌بندی (ساختن دو بخش با کمترین اتصال مشترک) محقق شود. فرایند بخش‌بندی در الگوریتم hMetis براساس بخش‌بندی به دو گراف است که می‌توانند اندازه‌های متفاوت داشته باشند. کیفیت نتیجهٔ الگوریتم به میزان زیادی به کیفیت جستجو در فاز بازترکیب زیرگراف‌ها وابسته است.

در جدول (٦)، الگوریتم بخش‌بندی پیشنهادی با الگوریتم hMetis، با استاندارد ٦ ورودی (K = 6) مقایسه شده است. زمان اجرا به میزان ۳۱/۷۳ درصد و تعداد TSVها به میزان ۹/۷۸ درصد کاهش یافته‌اند. الگوریتم بخش‌بندی hMetis در مقایسه با الگوریتم پیشنهادی و الگوریتم FSA (جدول ٤) که هردو بر مبنای الگوریتم SA توسعه یافته‌اند، از لحاظ کاهش تعداد TSV، عملکرد بدتری دارد. دلیل این امر را در قدرت جستجوی بسیار بهتر الگوریتم SA نسبت به الگوریتم ابتکاری به‌کاررفته در hMetis می‌توان دانست. زمان اجرای الگوریتم hMetis نیز به‌دلیل پیچیده‌بودن فرایند تبدیل هر گراف به دو زیرگراف و نیز جستجو برای ادغام بهینهٔ زیرگراف‌ها، به میزان چشمگیری از الگوریتم پیشنهادی بیشتر است.

در جدول (۷)، جانشانی و مسیردهی رهیافت FSA و فرایند پیشنهادی این مقاله از نقطه‌نظر تأخیر مسیر بحرانی[۲۷] (CPD) مقایسه شده‌اند. مسیر بحرانی در پیاده‌سازی یک مدار مجتمع دیجیتال، مسیری از یک ورودی اصلی به یک خروجی اصلی است که انتشار تغییرات سیگنال (0 به 1 یا 1 به 0) در آن از سایر مسیرها بیشتر است. پس از انجام مسیردهی، برای محاسبهٔ تأخیر مسیر بحرانی، مقدار تأخیر مسیر بین خروجی گیت‌های منبع و ورودی گیت‌های مقصد با استفاده از مدل المور محاسبه می‌شود. سپس گراف وزن‌دار مدار ساخته می‌شود که در آن تأخیر از ورودی به خروجی گیت‌ها و تأخیر مسیر بین گیت‌ها به‌صورت وزن یال‌های گراف در نظر گرفته می‌شوند. الگوریتم تحلیل زمانی، این گراف را با شروع از ورودی‌های اصلی و عبور از یال‌ها و رأس‌ها سطح به سطح طی می‌کند. در طی این فرایند، حداکثر تأخیر برای رسیدن سیگنال به هرکدام از رأس‌ها مشخص می‌شود. در مرحلهٔ آخر، با فرایند بازگشت به عقب از خروجی اصلیِ با بیشترین زمان تأخیر به سمت ورودی‌ها، مسیر با بیشترین تأخیر شناسایی می‌شود.

نتایج جدول ۷ نشان می‌دهند تأخیر مسیر بحرانی در روش پیاده‌سازی پیشنهادی نسبت به FSA، برای حالت‌های K=4، K=6 و K=8 به‌ترتیب به میزان ۸/۱۱، ۱۰/٤٤ و ۱۱/۵۸ درصد کاهش یافته است. دلیل اصلی این کاهش را در کیفیت بهتر مرحلهٔ بخش‌بندی و توجه به میزان تأخیر شبکه‌ها در مرحلهٔ جانشانی و مسیردهی در الگوریتم پیشنهادی می‌توان مرتبط دانست. با افزایش اندازهٔ سلول‌های استاندارد از K=4 تا K=8 میزان تأخیر مسیر بحرانی کاهش چشمگیری دارد. دلیل این امر کاهش تعداد گیت‌ها و به تبع آن کاهش پیچیدگی شبکهٔ مسیردهی است.

۵- نتیجه‌گیری

در این مقاله، الگوریتم‌های بخش‌بندی، جانشانی و مسیردهی، با هدف پیاده‌سازی مدارهای مجتمع روی تراشه‌های سه‌بعدی ASIC گسترش داده شده‌اند. در فاز بخش‌بندی نسبت به روش FSA، تعداد TSVها به اندازهٔ ٦/۱۵ درصد و زمان اجرا به میزان ۲۷/۷۹ درصد کاهش

داشته است. همچنین، نسبت به الگوریتم بخش‌بندی hMetis به اندازهٔ ۹/۷۸ درصد کاهش در تعداد TSVها صورت گرفته است؛ در حالی که الگوریتم پیشنهادی به میزان ۳۱/۷۳ درصد سریع‌تر عمل می‌کند. در فاز جانشانی و مسیردهی، الگوریتم پیشنهادی نسبت به FSA به‌صورت میانگین به اندازهٔ ۱۰/٤٤ درصد به مدار سرعت بخشیده است.

جدول (۳): مقایسهٔ روش FSA و روش پیشنهادی با استاندارد ٤ ورودی (K = 4)، از نقطه‌نظر زمان اجرا و تعداد TSV

Circuit	FSA		Proposed		Improvement (%)	
	TSV#	Time (MS)	TSV#	Time (MS)	TSV#	Time
ex5p	139	457	134	381	3.59	16.63
des	182	1673	177	1375	2.74	17.81
alu4	268	1718	256	1197	4.47	30.32
tseng	162	1648	169	973	-4.32	40.95
diffeq	189	1861	177	1498	6.34	19.50
seq	257	1807	246	1307	4.28	27.67
apex2	152	1592	134	1201	11.84	24.56
bigkey	227	2173	215	1564	5.28	28.02
ex1010	365	4951	341	3750	6.57	24.25
elliptic	458	5432	429	4002	6.33	26.32
s38417	319	8724	280	6825	12.22	21.76
s38584.1	486	8640	455	6638	6.37	23.17
clma	897	9347	863	7309	3.79	21.80
Avg					5.49	23.99

جدول (٤): مقایسهٔ روش FSA و روش پیشنهادی با استاندارد ٦ ورودی (K = 6)، از نقطه‌نظر زمان اجرا و تعداد TSV

Circuit	FSA		Proposed		Improvement (%)	
	TSV#	Time (MS)	TSV#	Time (MS)	TSV#	Time
ex5p	44	401	42	247	4.54	38.40
des	91	635	86	413	5.49	34.96
alu4	109	1230	102	927	6.42	24.63
tseng	218	1185	209	862	4.12	27.25
diffeq	157	1602	144	1086	8.28	32.21
seq	206	1417	202	958	1.94	32.39
apex2	153	1792	151	1201	1.30	32.97
bigkey	219	1342	208	944	5.02	29.65
ex1010	324	3524	305	2413	5.86	31.52
elliptic	458	3786	427	2897	6.76	23.48
s38417	207	4937	185	3815	10.62	22.72
s38584.1	258	5761	239	4239	7.36	26.41
clma	758	9173	705	6558	6.99	28.51
Avg					6.15	27.79

جدول (۵): مقایسهٔ روش FSA و روش پیشنهادی با استاندارد ۸ ورودی (K = 8)، از نقطه‌نظر زمان اجرا و تعداد TSV

Circuit	FSA			Proposed		Improvement (%)	
	TSV#	Time (MS)		TSV#	Time (MS)	TSV#	Time
ex5p	32	165		35	89	-9.37	46.06
des	110	438		103	354	6.36	19.17
alu4	97	1095		81	836	16.49	23.65
tseng	248	1136		235	718	5.24	36.79
diffeq	234	1307		213	891	8.97	31.82
seq	226	1183		209	872	7.52	26.28
apex2	91	1376		88	1008	3.29	26.74
bigkey	199	1270		186	982	1.50	22.67
ex1010	294	1073		279	814	5.10	24.13
elliptic	463	4157		478	3016	-3.24	27.44
s38417	215	5032		197	4211	8.37	16.31
s38584.1	321	5268		298	3957	7.16	24.88
clma	1106	8241		997	5218	+9.85	36.68
Avg						6.24	27.64

جدول (٦): مقایسهٔ الگوریتم hMetis از نقطه‌نظر زمان اجرا و تعداد TSV

Circuit	hMetis		Proposed		Improvement (%)	
	TSV#	Time (MS)	TSV#	Time (MS)	TSV#	Time
ex5p	53	462	42	247	20.75	46.53
des	98	670	86	413	12.24	38.35
alu4	117	1403	102	927	12.82	33.92
tseng	234	1109	209	862	10.68	22.27
diffeq	163	1761	144	1086	11.65	38.33
seq	213	1502	202	958	5.16	36.21
apex2	169	1863	151	1201	10.65	35.53
bigkey	228	1417	208	944	8.77	33.38
ex1010	337	3623	305	2416	9.49	33.39
elliptic	456	3679	427	2897	6.35	21.25
s38417	216	5135	185	3815	14.35	25.70
s38584.1	271	6034	239	4239	11.80	29.74
clma	776	10248	705	6558	9.14	36.01
Avg					9.78	31.73

جدول (۷): مقایسهٔ خروجی اعمال‌شده به فاز جانشانی و مسیردهی حاصل از خروجی بخش‌بندی ناشی از الگوریتم پیشنهادی و الگوریتم FSA

Circuit	Proposed			FSA		
	CPD (ns)			CPD (ns)		
	K = 8	K = 6	K = 4	K = 8	K = 6	K = 4
ex5p	1.51	2.75	6.79	1.72	2.53	5.84
des	2.48	2.83	5.94	2.94	3.27	6.37
alu4	3.67	4.09	6.21	4.29	4.52	7.93
tseng	3.21	4.57	5.06	3.06	4.77	6.32
diffeq	4.16	4.51	6.12	4.34	4.64	6.28
seq	4.32	4.34	6.71	4.25	4.31	7.08
apex2	5.07	4.89	6.93	5.22	4.73	7.49
bigkey	3.61	2.93	4.69	4.57	3.82	5.26
ex1010	4.49	7.59	8.78	4.38	8.71	7.68
elliptic	6.12	6.64	8.94	6.78	6.47	9.51
s38417	4.26	5.62	7.79	5.89	6.39	9.07
s38584.1	3.74	5.16	6.84	5.10	7.26	8.14
clma	6.43	9.14	8.86	7.48	11.23	10.61
Avg	4.08	5.01	6.89	4.61	5.58	7.50
Imp (%)	11.58	10.44	8.11			

مراجع

[1] Moore, G. E., , “Cramming More Components On to Integrated Circuits”, Electronics Magazine, Vol. 38, No. 8, April 1965.

International Technology Roadmap for Semiconductors (ITRS). 2015 edition. [online], available: https://www.dropbox.com/sh/3jfh5fq634b5yqu/AADYT8V2Nj5bX6C5q764kUg4a?dl=0

[2] Jahanirad, H., “Efficient reliability evaluation of combinational and sequential logic circuits”, J Comput Electron, Vol. 18, No. 1, March 2019.

[3] Jahanirad, H.,”CC-SPRA: Correlation coefficients approach for signal probability-based reliability analysis”. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 27, No. 4, April 2019.

[4] Sharma, R., “Design of 3D integrated circuits and systems”, CRC Press, 2014.

[5] Ababei, C., Yan Feng, Goplen, B., Mogal, H., Tianpei Zhang, Bazargan, K., & Sapatnekar, S., “Placement and Routing in 3D Integrated Circuits”, IEEE Design and Test of Computers, Vol. 22, No. 6, April 2005.

[6] Cuesta, D., Risco-Martin, J.L., Ayala, J.L. and Hidalgo, J.I., “3D thermal-aware floorplanner using a MOEA approximation”, Integration, Vol. , No. 1, Januray 2013.

[7] Saha, D., and Sur-Kolay, S., “Guided GA-Based Multiobjective Optimization of Placement and Assignment of TSVs in 3-D ICs”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 27, No. 8, April 2019.

[8] Meitei, N.Y., Baishnab, K.L. and Trivedi, G., “3D-IC partitioning method based on genetic algorithm”, IET Circuits, Devices & Systems, Vol. 14, No. 7, November 2020.

[9] Sait, S.M., Oughali, F.C. and Al-Asli, M., “Design partitioning and layer assignment for 3D integrated circuits using tabu search and simulated annealing”, Journal of applied research and technology, Vol. 14, No. 1, Februrey 2016.

[10] Fakheri Tabrizi, A., Behjat, L., Swartz, W., & Rakai, L., “A fast force-directed simulated annealing for 3D IC partitioning”, Integration, the VLSI Journal, Vol. 55, No. 1, September 2016.

[11] Siddique, N., & Adeli, H., “Simulated Annealing, Its Variants and Engineering Applications”, International Journal on Artificial Intelligence Tools, Vol. 25, No. 6, December 2016.

[12] McMurchie, L., Ebeling, C., “PathFinder: a negotiation based performance driven router for FPGAs”, in Conference of Field Programmable Gate Arrays FPGA, pp. 291–301, 1995.

[13] Lee, M., Pak, J. S., & Kim, J., “Electrical Design of Through Silicon Via”, Springer, 2014.

[14] Kaushik, B. K., Majumder, M. K., and Kumari, A., “Fabrication and Modelling of Copper and Carbon Nanotube Based Through-Silicon Via, Design of 3D Integrated Circuits and Systems”. CRC Press/Taylor & Francis Group, 2014.

[15] Sherwani, N., “Algorithms for VLSI Physical Design Automation”, Springer Science & Business

Media, 2012..
[16] Gbadamosi, O. A. and Aremu, D. R., "Design of a Modified Dijkstra's Algorithm for finding alternate routes for shortest-path problems with huge costs.", International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS), pp. 1-6, 2020.
[17] Lau, J.H., "Heart and Soul of 3D IC Integration", posted at 3D InCites on June 2010.
[18] Bamberg, L., & Garcia-Ortiz, A., " High-Level Energy Estimation for Submicrometric TSV Arrays", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 25, No. 10, June 2017.
[19] Predictive technology model, available [online] : http://ptm.asu.edu/latest.html.
[20] Karypis, G., Aggarwal, R., Kumar, V., & Shekhar, S., "Multilevel hypergraph partitioning: applications in VLSI domain", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 7, No. 1, March 1999.
[21] Nvidia Ampere architecture, (May 2020), available [online] : https://www.nvidia.com/en-gb/data-center/nvidia-ampere-gpu-architecture/.
[22] Jamieson,P., Kent,K., Gharibian,F., and Shannon, L., "Odin ii-an open-source verilog hdl synthesis tool for cad research", In International Symposium on Field-Programmable Custom Computing Machines, pp. 149–156. 2010.
[23] Jahanirad, H., Mohammadi, K., "Reliable Implementation on SRAM-based FPGA using Evolutionary Methods", IETE Journal of Research, Vol. 59, No. 5, September 2013.
[24] Rabaey, J. M., Chandrakasan, A. P., & Nikolić, B., "Digital integrated circuits: a design perspective", Pearson education 2003.
[25] Østerhus, S., "Subthreshold CMOS Cell Library by 22 nm FDSOI Technology", MS thesis. NTNU, 2018.

[1] Interconnect Delays
[2] 3D Integrated Circuits
[3] Through Silicon Via
[4] Simulated Annealing
[5] Pathfinding
[6] Stack
[7] Manhattan distance
[8] Reliable
[9] Cost effective
[10] filler material
[11] Polycrystalline silicon
[12] Nano-interconnects Graphene-based
[13] Pitch
[14] Cut-size
[15] Place and Route
[16] Net
[17] Berkeley Logic Interchange Format
[18] Semi-Custom
[19] Stall
[20] External Cost
[21] Internal Cost
[22] Steiner Tree
[23] Elmore Delay Model
[24] Shortest path
[25] Dijkstra
[26] Layout
[27] Critical Path Delay