علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد۳، شماره ۳ (الف)، پائیز ۱۳۸۴
صفحات ۵۵-۷۰
مقاله عادی

یک چارچوب کاری مبتنی بر شباهت پیکربندی‌ها برای افراز زمانی و طـراحی فیزیکی سیستم‌های محاسبات قابل باز پیکربندی

فرهاد مهدی‌پور[۱] مرتضی صاحب‌الزمانی[۱] حمیدرضا احمدی‌فر[۲] مهدی صدیقی[۱]

[۱]دانشکده مهندسی کامپیوتر و فناوری اطلاعات، دانشگاه صنعتی امیرکبیر
[۲]دانشکده فنی، دانشگاه گیلان

چکیده

در این مقاله یک چارچوب کاری برای کامپایل ایستای کاربردهای محاسبات قابل باز پیکربندی ارائه می‌گردد که ترکیب دو مفهوم سنتز و طراحی فیزیکی را برای انجام فرآیند کامپایل مطرح می‌کند. در یک روش جدید افراز زمانی، شباهت زوج‌نودها و اتصالات آنها مورد استفاده قرار می‌گیرد. یک روش تکمیلی با هدف افزایش تشابه پیکربندی‌ها با مطرح ساختن تکنیک درج نودهای ساختگی، زمان پیکربندی را برای سخت‌افزارهایی که قابلیت پیکربندی جزئی دارند کاهش می‌دهد. همچنین از ماجول‌های سفت و ماجول‌های سخت برای جایابی استفاده می‌شود. جایابی افزایشی برای کاهش زمان کامپایل و بهبود زمان سربار باز پیکربندی و نیز زمان کل اجرا پیشنهاد شده است. علاوه بر این یک الگوریتم جدید افراز زمانی نیز برای کاهش حافظه‌ی مصرفی در پیکربندی‌ها ارائه گردیده که با پیمایش عمودی نودها به جای پیمایش افقی، مقدار حافظه‌ی مصرفی پیکربندی‌ها و نیز مساحت سخت‌افزار مورد استفاده را کاهش می‌دهد. یک ابزار نیز برای انجام افراز زمانی و طراحی فیزیکی مجتمع و تولید پیکربندی‌های نهایی توسعه یافته است.

کلمات کلیدی: سیستم‌های محاسبات قابل باز پیکربندی، جریان طراحی، افراز زمانی، طراحی فیزیکی، جایابی، مسیریابی، نود ساختگی، مشابهت.

۱- مقدمه

ایده‌ی اولیهٔ سیستم‌های محاسبات قابل باز پیکربندی (RCS) در دههٔ ۱۹۶۰ مطرح گردید. در چند سال اخیر به دلیل رشد تکنولوژی مدارهای برنامه‌پذیر، امکان پیاده‌سازی چنین سیستم‌هایی به وجود آمده است [۱]. RCS ترکیبی از قابلیت‌های پردازنده‌های همه منظوره را در کنار افزاره‌های برنامه‌پذیر بکار می‌گیرد [۲]. این نوع سیستم‌ها را می‌توان از لحاظ قابلیت انعطاف‌پذیری، بالاتر از تراشه‌های با کاربرد خاص (ASIC[۱]) و از لحاظ کارآیی بالاتر از همهٔ پردازنده‌های همه منظوره در نظر گرفت. به عبارتی RCS قابلیت‌های انعطاف‌پذیری و کارآیی متوسطی را نسبت به دو نوع دیگر به همراه می‌آورد [۳،۴]. RCS بدلیل انعطاف پذیری و قابلیت استفاده از معماری‌های سریع، پتانسیل مناسبی برای به کارگرفتن در کاربردهایی مثل پردازش تصویر، رمزنگاری، شناسایی هدف و پردازش سیگنال دیجیتال دارد[۵، ۶]. در یک سیستم قابل بازپیکربندی، بازپیکربندی می تواند بصورت ایستا و یا پویا انجام شود. بازپیکربندی پویا به مفهوم برنامه‌ریزی بلوک‌های منطقی برنامه‌پذیر و منابع مسیریابی در زمان اجرا و در مقابل بازپیکربندی ایستا شامل تعیین پیکربندی‌های سیستم قبل از زمان اجرا جهت استفاده از آنها در زمان اجرا می‌باشد [۱، ۷]. در زمینهٔ پیاده‌سازی کاربردهای مختلف بر روی RCS، در حال حاضر مشکلات مختلفی وجود دارد. فقدان روش طراحی مشخص یا استاندارد، فقدان کامپایلرهای مخصوص، وجود مشکلات در تغییر جزیی پیکربندی‌ها، زمان طولانی بازپیکربندی افزاره‌های برنامه‌پذیر[۲] و نیز شناسایی کاربردهایی که برای پیاده‌سازی در قالب سیستم‌های قابل بازپیکربندی مناسب باشند، همگی از جمله چالش‌های موجود در پیاده‌سازی کاربردهای RCS می‌باشند[۶، ۸، ۹]. درحال حاضر به نظر می رسد که هنوز ابزارهایی با اهداف عمومی برای RCS که طراحی و پیاده سازی کاربردهای مختلف را پشتیبانی کنند به وجود نیامده‌اند.

ما در این مقاله قصد داریم به بررسی و ارائهٔ روشی برای طراحی کاربردهای مختلف که دارای ماهیت ایستا می‌باشند بر روی سیستم RCS بپردازیم. بدین منظور یک جریان طراحی برای RCS ارائه می‌گردد که ترکیبی از مراحل سطح

بالا و سطح پایین طراحی است. در جریان طراحی پیشنهادی دو فاز افراز زمانی[۳] و طراحی فیزیکی[۴] دیده می‌شود. افراز زمانی، مرحلهٔ بعد از سنتز مدار است که در سطوح بالای مراحل طراحی انجام می‌شود. از سوی دیگر طراحی فیزیکی شامل مراحل جایابی و مسیریابی است که در ادامهٔ افراز زمانی اجرا می‌شود. یک روش افراز زمانی مبتنی بر شباهت نودهای منفرد در [۱۰، ۱۱] ارائه شده بود (با توجه به اینکه هر نود معادل یک ماجول در مدار می‌باشد لذا در متن مقاله از نود و ماجول بطور معادل استفاده شده است). در این مقاله یک روش دیگر برای در نظر گرفتن زوج نود به جای نود منفرد ارائه می‌گردد. همچنین دو نوع روش طراحی متفاوت مبتنی بر ماجول‌های سفت[۵] و همچنین ماجول‌های سخت[۶] پیشنهاد می‌گردد. ماجول سفت به یک ماجولی گفته می‌شود که به صورت نت لیست ارائه شده است. ماجول سخت نیز یک ماجول جایابی شده است که دارای شکل ثابت می‌باشد. در ادامه یک روش افراز زمانی جدید نیز برای حل مشکل مربوط به حافظهٔ مصرفی بالا بین پیکربندی‌های متوالی ارائه می‌دهیم. بخش‌های مختلف این مقاله به شرح زیر است:

در بخش دوم به مراحل افراز زمانی و طراحی فیزیکی و مرورکارهای انجام شده در این زمینه می پردازیم. در بخش سوم جریان طراحی پیشنهادی را ارائه می‌کنیم. در بخش چهارم ضمن مرور روش قبلی ارائه شده که مبتنی بر شباهت نودهای منفرد بود، روش دیگری که مبتنی بر شباهت زوج نودها می‌باشد پیشنهاد می‌کنیم. الگوریتم تکمیلی درج نودهای ساختگی نیز که سعی در افزایش میزان شباهت پیکربندی‌های متوالی دارد در ادامه این بخش معرفی می‌شود. ارائهٔ روش افزایشی جایابی و مسیریابی مجتمع با افراز زمانی در بخش پنجم انجام می شود. همچنین نحوه انجام طراحی مبتنی بر ماجول‌های سفت و سخت در این بخش ارائه می‌گردد. در بخش ششم یک روش افراز زمانی دیگر که هدف آن کاهش میزان حافظهٔ مصرفی در پیکربندی‌ها است، پیشنهاد می‌گردد. بخش هفتم مقاله به ارائهٔ نتایج حاصل از آزمایش‌ها پرداخته و در بخش هشتم به یک جمع بندی از دست یافته های مقاله می پردازیم.

۲-کارهای انجام شدهٔ قبلی

برای پیاده سازی مدارهای بزرگ در یک افزاره‌ی برنامه‌پذیر مانند FPGA[۷] لازم است که مدار با استفاده از فرآیند افراز به چند بخش تقسیم بندی گردد. در این صورت اجرای کل مدار با ورود و خروج متوالی پیکربندی مربوط به هر بخش و اجرای هر یک انجام می‌شود. این نوع از افراز که با در نظر گرفتن مفاهیم زمانی و ترتیب اجرای عملگرها و پارتیشن‌ها انجام می‌گردد افراز زمانی نامیده می شود. در واقع افراز زمانی، افراز مدار به پارتیشن‌های مجزا از لحاظ زمانی است که یا نمی توانند و یا نیازی به اجرای همروند ندارند بلکه به ترتیب و به صورت متوالی اجرا می‌شوند. با استفاده از افزاره های با قابلیت باز پیکربندی جزئی[۸] بخشی از یک طرح می تواند توسط بخش دیگر جایگزین شود در حالی که سایر بخش‌ها فعال بوده و در حال اجرا هستند [۱۲، ۱۳، ۱۴].

در روش‌های ارائه شده برای حداقل‌سازی زمان سربار بازپیکربندی عمدتاً از مدارهای FPGA چند زمینه‌ای[۹] و به ویژه از معماری‌های دانه درشت استفاده شده است [۱۵]. Karthikeya و همکارانش [۱۶] الگوریتم‌هایی را برای افراز زمانی و زمان‌بندی اجرای مدارهای بزرگ بر روی سخت افزار قابل باز پیکربندی که دارای محدودیت مساحت است ارائه کرده‌اند. در روش ارائه شده محدودیت‌های زمانی و سربار زمانی باز پیکربندی در نظر گرفته نشده است. Bobda در [۸] دو روش برای حل مساله افراز زمانی ارائه کرده است. اولین روش مدل توسعه یافته‌ای از روش معروف list scheduling است و روش دوم از یک فضای بردار سه بعدی برای مکان یابی ماجول‌ها استفاده می کند که در آن ابعاد ماجول شامل عرض و ارتفاع به همراه زمان به عنوان بعد سوم در نظر گرفته شده است.

در [۱۷]، Spillane و Owen تلاش کرده‌اند تا ترتیب مناسبی از اجرای پارتیشن‌ها را ایجاد نمایند. این ترتیب مناسب با هدف دستیابی به سازش بین معیارهای طراحی مختلف همچون زمان باز پیکربندی، کارآیی زمانی و اندازه‌ی سخت‌افزار قابل باز پیکربندی حاصل می‌شود. SPARCS [۱۸] یک سیستم سنتز و افراز مجتمع است که افراز زمانی و زمان‌بندی طرح مورد نظر برای اجرا بر روی سیستم قابل بازپیکربندی را انجام می‌دهد. ابزار افراز زمانی مورد استفاده در SPARCS یک حد بالا را برای تعداد پارتیشن‌ها تخمین زده و سپس مساله را بصورت یک سیستم برنامه ریزی غیرخطی فرموله می‌کند. این روش زمان اجرای بالایی دارد. تابع هزینهٔ اصلی استفاده شده در SPARCS پهنای باند حافظه است [۱۹]. Luk و همکارانش [۲۰] مدلی برای بهره‌گیری از عملگرهای مشترک در پارتیشن‌های متوالی ارائه کرده‌اند. آنها روشی را ارائه نموده‌اند که تا حد امکان زمان باز پیکربندی و در نتیجه زمان اجرای کاربرد را کاهش می‌دهد. در این مدل محدودیت‌های زمانی در نظر گرفته نمی‌شوند. Tanougast و همکارانش در [۱۹] سعی کرده‌اند تا کوچک‌ترین سخت‌افزار قابل بازپیکربندی را با درنظر گرفتن محدودیت‌های زمانی پیدا کنند. در [۲۱] تکنیک پیش واکشی پیکربندی برای کاهش سربار پیکربندی ارائه شده است. همچنین در [۱۵] یک تکنیک پیش واکشی نمایش داده شده که از وقوع پیش بینی‌های نادرست و افزایش زمان اجرای کل سیستم جلوگیری می‌کند. Ganesan [۲۲]، روشی را برای همزمانی عملیات بازپیکربندی و اجرا برای کاهش سربار بازپیکربندی و کاهش تاخیر طراحی، ارائه کرده است. اکثر تکنیک‌های فوق برای بازپیکربندی زمان اجرا مناسب هستند.

طراحی فیزیکی سیستم‌های قابل بازپیکربندی اغلب بر اساس الگوریتم‌های جایابی و مسیریابی مرسوم برای FPGA انجام می‌گردد [۲۳، ۲۴]. دو فاز اصلی طراحی فیزیکی شامل مراحل جایابی[۱۰] و مسیریابی[۱۱] است. در فاز جایابی، مکان مناسب ماجول‌های مدار بر روی سخت‌افزار تعیین می‌شود. در این فرآیند، حداقل سازی طول اتصالات ، مساحت اشغال شده بر روی سخت‌افزار و طول طولانی ترین سیم از جمله مهم‌ترین معیارهای بهینه‌سازی می‌باشند. الگوریتم‌های متفاوتی برای حل مساله‌ی جایابی ارائه شده اند [۲۳، ۲۵، ۲۶]. Simulated Annealing یکی از الگوریتم‌های جایابی موفق در طی چند سال گذشته بوده است.

مسیریابی، فرآیند تعیین دقیق مسیرها و سوئیچ‌هایی است که باید برای اتصال پایانه‌های یک نت برنامه‌ریزی شوند. این مرحله بایستی برای تمامی نت‌های مدار انجام شود. مسیر یابی کلی[۱۲] و جزئی[۱۳] باید بعد از جایابی، برای ایجاد اتصالات بین ماجول‌های مدار انجام شود. بعد از یک جایابی مناسب، یک مسیریاب با احتمال زیاد می‌تواند سیم‌های بین ماجول‌ها را مسیر یابی کند در غیر اینصورت فرآیند جایابی باید دوباره اجرا شود [۲۶، ۲۷].

اغلب روش‌های ذکر شده برای استفاده در زمان اجرا پیشنهاد شده‌اند. برخی از این روش‌ها نیز طراحی ایستای کاربردها بر روی سیستم قابل باز پیکربندی را انجام می‌دهند. از ویژگی بارز این روش‌ها عدم پرداختن به جزئیات مربوط به فرآیندهای جایابی و مسیریابی است به گونه‌ای که اغلب تنها به فرآیند افراز زمانی پرداخته شده است. در معدود روش‌هایی که طراحی فیزیکی را نیز انجام می‌دهند، از ابزارهای آماده برای جایابی و مسیریابی استفاده شده و به جزئیات و نحوهٔ انجام این مراحل و ارتباط آنها با فرآیند افراز زمانی توجه نشده است.

از ویژگی‌های مهم کار ارائه شده در این مقاله، ارائهٔ یک جریان طراحی مجتمع[۱۴] برای افراز زمانی و طراحی فیزیکی برای طراحی کاربردهای ایستا جهت اجرا بر روی یک سخت‌افزار قابل باز پیکربندی است. افراز زمانی بعنوان یک مرحله بعد از سنتز و از جمله مراحل بالا یا میانی طراحی است در حالی که طراحی فیزیکی فرآیندی سطح پایین است. در چارچوب کاری ارائه شده برای طراحی کاربردهای ایستا، این امکان وجود دارد که مراحل افراز زمانی و طراحی فیزیکی در ارتباط با هم و با دریافت اطلاعات لازم از یکدیگر اجرا شوند. انجام این مراحل با استفاده از ماجول‌های سفت و سخت، دیگر نوآوری این مقاله می‌باشد. همچنین

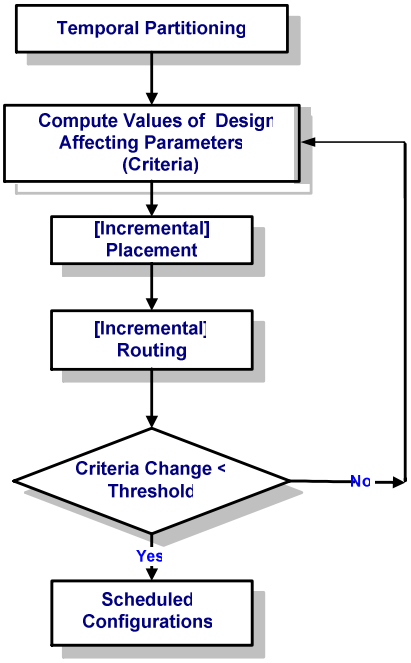
ارائه‌ی روش‌های افراز زمانی مبتنی بر شباهت با هدف کاهش زمان کامپایل و مهمتر از آن برای کاهش زمان باز پیکربندی و زمان کل اجرای کاربرد، انجام گرفته است. در مقالات پیشین [۱۰، ۱۱]، یک روش افراز زمانی مبتنی بر شباهت ماجول‌ها در پارتیشن‌های متوالی ارائه شد. در این مقاله یک روش افراز زمانی دیگر مبتنی بر شباهت زوج نود و نیز روش دیگری برای کاهش میزان حافظه‌ی مصرفی در پیکربندی‌ها ارائه می‌شود. ما یکی از ابزارهای طراحی فیزیکی معروف و متداول در کاربردهای آکادمیک برای FPGA یعنی VPR [۲۷، ۲۸] را بکار می‌بریم که با هدف پاسخ به نیازهای جریان طراحی ارائه شده تغییر داده شده است. ما هدف اصلی را حداقل سازی زمان سربار بازپیکربندی و زمان کل اجرای کاربرد، حافظه‌ی مصرفی و مساحت سخت‌افزار مورد استفاده قرار داده‌ایم.

۳- جریان طراحی پیشنهادی

یکی از مشکلات مهم موجود در زمینه‌ی سیستم‌های محاسبات قابل باز پیکربندی فقدان ابزارهای طراحی است که امکان پیاده‌سازی کاربردهای مورد نظر را بصورت قابل باز پیکربندی فراهم نموده و مجموعهٔ کامل مراحل طراحی را دارا باشد. یکی از اهداف ما در این مقاله، ارائه‌ی یک جریان طراحی است که امکان پیاده‌سازی کاربردهایی که دارای ماهیت ایستا می‌باشند را بر روی یک سخت‌افزار قابل باز پیکربندی فراهم نماید و ترکیبی از مراحل سطح بالا و پایین طراحی را داشته باشد. یک جریان طراحی در شکل ۱ پیشنهاد شده است که در آن افراز زمانی و طراحی فیزیکی به عنوان مراحل مهم طراحی در نظر گرفته شده‌اند. افراز زمانی مرحله‌ای است که بعد از سنتز مدار قابل اجرا است و در واقع یک مرحله‌ی میانی در مجموعه مراحل طراحی است. از سوی دیگر طراحی فیزیکی ازجمله مراحل پایین در مجموعه مراحل طراحی است. در واقع یکی از خصوصیات این روش ترکیب مراحل بالا و پایین طراحی و اجرای توأم این مراحل در ارتباط با یکدیگر است.

برای پیاده‌سازی یک کاربرد، طراح یک گراف جریان داده (DFG[۱۵]) را برای کاربرد مورد نظر خود ارائه می‌دهد. یک گراف جریان داده شامل مجموعه‌ای از نودها و کمان‌های جهت‌دار بین آنها است که وابستگی میان نودها را نشان می‌دهد. یک کمان جهت‌دار نه تنها مسیر انتقال داده را بین نودها تعیین می‌کند بلکه ترتیب اجرای آنها را نیز بیان می‌کند. در شکل ۲ نمونه‌ای از یک DFG دیده می‌شود. طراح کاربرد مورد نظر خود را در قالب یک گراف جریان داده ارائه می‌کند. نودهای این گراف سلول‌های از قبل طراحی شده می‌باشند که در یک فایل کتابخانه‌ای قرار گرفته‌اند. ما مجموعه‌ای از ماجول‌های سفت و سخت را تولید کرده و در یک فایل کتابخانه‌ای قرار داده‌ایم. ماجول سفت به صورت یک نت‌لیست از اجزای تشکیل دهنده‌ی آن ارائه می‌گردد. هر یک از اجزای این سلول به طور مستقیم بر روی یک سلول قابل برنامه‌ریزی به نام CLB[۱۶] از افزاره‌ی برنامه‌پذیر، قابل نگاشت می‌باشد. برای یک ماجول سخت نیز که جایابی آن قبلاً انجام گرفته است با توجه با ثابت بودن شکل ماجول، جایابی آن بر روی بخشی از افزاره بدون تغییر در ساختار داخلی آن امکان‌پذیر می‌باشد. به این ترتیب طراح می‌تواند حتی بدون استفاده از ابزار سنتز، گراف جریان داده مربوط به کابرد را ارائه نماید.

ما برای پیاده‌سازی جریان طراحی پیشنهادی، یک ابزار ارائه کرده‌ایم. در این ابزار یک گراف جریان داده به عنوان ورودی دریافت می‌گردد و سپس الگوریتم افراز زمانی پیشنهادی، این گراف را دریافت و بعد از انجام افراز مدار، زمان‌بندی اجرای هر پارتیشن را تعیین می‌کند. هر یک از پارتیشن‌های تولید شده در مرحله‌ی قبل به نسخه‌ی تغییر یافته‌ای از ابزار جایابی و مسیریابی VPR داده می‌شوند. VPR یکی از رایج‌ترین ابزارهای موجود برای جایابی و مسیریابی مدار بر روی افزاره‌های برنامه‌پذیر می‌باشد. در نسخه‌ی تغییر یافته‌ی VPR، امکان جایابی و مسیریابی افزایشی به قابلیت‌های نرم‌افزار اضافه شده است. در VPR، سخت‌افزار مقصد یک FPGA با معماری island-style است. فرض بر این است که این افزاره قابلیت برنامه‌ریزی جزیی را دارا می‌باشد. با بهره‌گیری از چنین افزاره‌ای می‌توان در زمان اجرا بخشی از سخت‌افزار بخش‌های دیگر آن را بدون ایجاد وقفه پیکربندی نمود.



شکل ۱- جریان طراحی پیشنهادی

۴- الگوریتم افراز زمانی

مدارهایی که اندازه‌ی آنها بزرگتر از اندازه‌ی افزاره‌ی برنامه‌پذیر است، به چند پارتیشن افراز و سپس زمان‌بندی اجرای پارتیشن‌ها تعیین می‌شود. افراز زمانی می‌تواند بعنوان یک تقسیم کنندهٔ گراف جریان داده به چندین افراز درنظرگرفته شود به گونه ای که هر پارتیشن در افزارهٔ مناسبی قرار گیرد و همچنین وابستگی داده‌ها نقض نشود. در این بخش به ارائه یک الگوریتم افراز زمانی می‌پردازیم که هدف اصلی آن کاهش زمان باز پیکربندی است چرا که عمدتاً زمان باز پیکربندی از مهم‌ترین عوامل تأثیرگذار در کارآیی سیستم‌های RCS است. دلیل اصلی این مسأله زمان باز پیکربندی نسبتاً بالا در افزاره‌های برنامه‌پذیر است.

با فراهم نمودن امکان برنامه‌ریزی بخشی از افزاره به جای کل آن می‌توان زمان باز پیکربندی را کاهش داد. یک ایده برای کاهش این زمان، سعی در افزایش شباهت میان دو پیکربندی متوالی است. برای این منظور ما عامل شباهت را برای دو پیکربندی متوالی تعریف می‌کنیم. منظور از شباهت دو نود این است که هر دو یک تابع را پیاده‌سازی کنند. در [۱۰، ۱۱] روش افراز زمانی مبتنی بر شباهت ماجول‌ها در پیکربندی‌های متوالی ارائه گردید که در این مقاله به ارائه‌ی خلاصه‌ای از آن خواهیم پرداخت. در این بخش از مقاله یک روش افراز زمانی دیگر مبتنی بر شباهت پیشنهاد می‌شود. در روش اخیر هر دو نود با اتصال بین آنها به عنوان یک زوج نود در نظر گرفته شده و شباهت بین دو پارتیشن متوالی مبتنی بر شباهت زوج نودها تعیین می‌شود در حالی که در روش قبل، شباهت پارتیشن‌های متوالی بر اساس شباهت نودهای منفرد صورت می‌گرفت. در این روش‌ها با استفاده از الگوریتم‌های ترکیبی متفاوت سعی می‌شود تعداد نودهای منفرد مشابه و یا زوج نودهای مشابه حداکثر شود.

۴-۱ الگوریتم افراز زمانی مبتنی بر شباهت نودهای منفرد

در روش پیشنهادی در [۱۰، ۱۱]، بعد از دریافت یک گراف جریان داده، تعیین اولویت اجرای نودهای آن بر اساس الگوریتم زمان‌بندی ASAP[۱۷] [۸، ۲۹] انجام

می‌شود. در الگوریتم ASAP، تعیین اولویت نودها بر اساس تأخیر اجرای آنها انجام می‌گردد. تأخیر اجرای هر نود نیز بر اساس تأخیر آنها نسبت به نودهای قبلی تعیین می‌شود. در ادامه سطح اولویت هر یک از نودها مبتنی بر تأخیر هر نود نسبت به نودهای قبل از آن مشخص می‌شود [۸]. الگوریتم افراز زمانی با در نظر گرفتن اندازه‌ی هر یک از ماجول‌ها، سطح اولویت اجرای آنها و نیز اندازه‌ی افزاره‌ی مقصد، پارتیشن‌بندی گراف را انجام می‌دهد. الگوریتم افراز زمانی یک مرحله‌ی تکمیلی دیگر را برای افزایش میزان شباهت بین دو پارتیشن اجرا می‌کند. برای هر دو پارتیشن متوالی، مجموعه نودهایی از پارتیشن اول که دارای بالاترین اولویت اجرا و مجموعه نودهایی از پارتیشن دوم که دارای پایین‌ترین اولویت می‌باشند انتخاب و نودهای متعلق به این مجموعه‌ها دوبدو جابجا می‌شوند. هر جابجایی که باعث بهبود در میزان شباهت دو پارتیشن گردد پذیرفته شده و در غیر اینصورت رد می‌شود (شکل ۲). به عنوان مثال در شکل ۲-الف پارتیشن‌های اولیه بعد از اجرای مرحله‌ی اول الگوریتم به دست آمده است. با جابجایی ماجول ADD متعلق به پارتیشن دوم با ماجول SUB از پارتیشن اول، میزان شباهت بین دو پارتیشن به اندزه‌ی دو واحد افزایش می‌یابد (شکل ۲-ب). به این ترتیب در تولید پیکربندی مربوط به پارتیشن دوم می‌توان بخش‌هایی از پیکربندی اول را بدون تغییر در افزاره نگهداشت و تنها باقیمانده‌ی پارتیشن دوم را که با پارتیشن اول متفاوت است بر روی سخت‌افزار برنامه‌پذیر انتقال داد. انجام این کار در نهایت زمان باز پیکربندی را کاهش می‌دهد که از اهداف مهم در کاهش زمان اجرا و بهبود کارآیی سیستم می‌باشد. اثبات کارآیی این روش در [۱۱] آمده است.

۲-۴ الگوریتم افراز زمانی مبتنی بر شباهت زوج نود

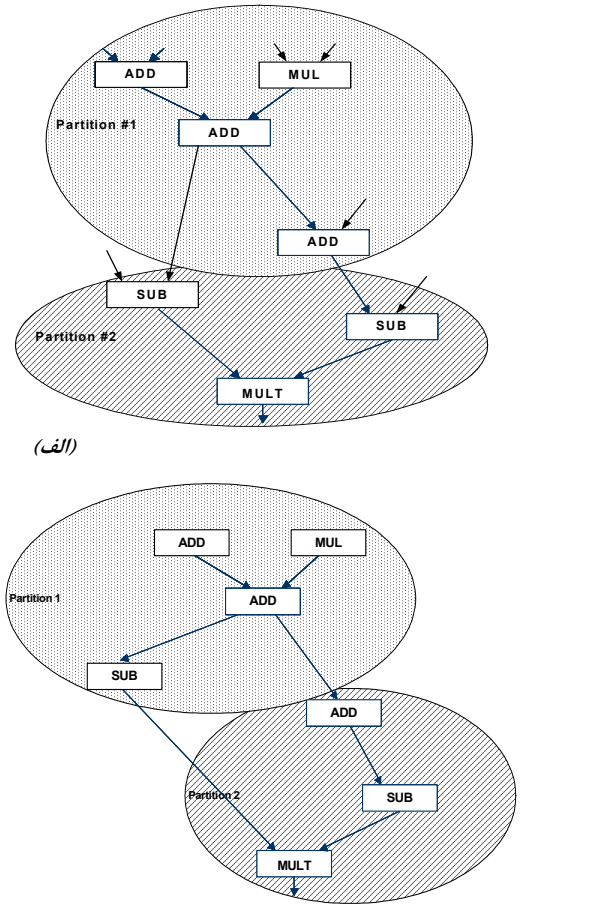
در یک سخت افزار با قابلیت بازپیکربندی جزئی، بخشی از مدار بدون آسیب رساندن به سایر بخش‌های طرح می‌تواند برنامه‌ریزی گردد و قسمت‌های مشترک دو پیکربندی متوالی، می‌توانند بدون تغییر باقی بمانند. عامل میزان تشابه که در [۱۰، ۱۱] تعریف شده است، سطح تشابه بین دو پیکربندی متوالی (در قسمت-هایی از توابع عملیاتی نودها) را ارزیابی می‌کند. هدف اصلی ما، کاهش زمان بازپیکربندی و سربار زمان اجرای کاربرد و فضای مورد نیاز برای پیاده سازی آنها است. در این بخش یک الگوریتم افراز زمانی را معرفی می‌کنیم که بر اساس عامل تشابه و شباهت زوج نودها کار می‌کند. نشان می‌دهیم که تشابه زوج نودها با درنظرگرفتن اتصالات بین آنها موجب کاهش پیچیدگی فرآیند مسیریابی و زمان بازپیکربندی می‌شود. الگوریتم پیشنهادی یک DFG را دریافت می‌کند که نودهای آن متناظر با ماجول‌های سفت و یا سخت از پیش طراحی شده‌ای هستند که در یک کتابخانه ذخیره شده‌اند. با استفاده از الگوریتم ارائه شده، تنها بخش‌های غیر-مشابه پیکربندی‌ها، جایابی و مسیریابی می‌شوند. این کار باعث کاهش زمان مورد نیاز برای تولید پیکربندی ها و همچنین کاهش تأخیر زمان اجرای بازپیکربندی می‌شود.

مسیریابی یک فرآیند زمان‌بر در فازهای کامپایل و بازپیکربندی زمان اجرا است. اغلب ، حدود ۷۰٪ تا ۹۰٪ از رشته بیتهای پیکربندی، مربوط به منابع مسیریابی و اتصالات است. بعلاوه، ۸۰٪ تأخیر مسیرها مربوط به اتصالات مدار است [۱۵] . الگوریتم پیشنهادی برای افراز زمانی، زوج نودها و اتصالات بین آنها را به عنوان یک واحد پایه در الگوریتم مبتنی بر تشابه به کار می‌گیرد. در ابتدا به معرفی برخی اصطلاحات و تعاریف می‌پردازیم:

تعریف ۱ : برای نود i در یک DFG ، $F(i)$ ماجولی را در کتابخانهٔ ماجول‌ها با عملیات محاسباتی مشابه با نود i نشان می‌دهد.

تعریف ۲ : برای دو نود i و j در یک DFG و پارتیشن P_k از DFG، $NP(i,j,k)$، یک زوج نود است که $i \in P_k$ و $j \in P_k$ و i و j دو نود در DFG هستند و یک لبه از i به j وجود دارد.

تعریف ۳: برای دو زوج نود $NP(i_1, j_1, k)$ و $NP(i_2, j_2, k+1)$ در یک DFG



شکل ۲- نمونه‌ای از یک گراف جریان داده و افراز آن (الف) شباهت بین دو پارتیشن وجود ندارد (ب) با جابجایی نودها در دو پارتیشن متوالی دو مجموعه نود مشابه به وجود می‌آید.

و دو افراز متوالی P_k و P_{k+1} از DFG، $SNP(i_1, j_1, i_2, j_2, k)$ یک زوج نود مشابه است که $i_1, j_1 \in P_k$ و $i_2, j_2 \in P_{k+1}$ و $F(i_1)= F(i_2)$، $F(j_1)= F(j_2)$.

تعریف ۴: مجموعهٔ زوج نودهای مشابه ($SNPS$) برای پارتیشن به صورت مجموعه‌ای از همه زوج نودهای موجود در P_k و P_{k+1} تعریف می‌شود:

$$SNPS(P_k) = \{SNP(i_1, j_1, i_2, j_2, k) \mid \begin{matrix} \forall i_1, j_1 \in P_k \\ and \quad \forall i_2, j_2 \in P_{k+1} \end{matrix} \}$$

(رابطه ۱)

تعریف ۵: تعداد زوج تشابه در یک پارتیشن ($PPSV$) برای P_k تعداد زوج نودهای مشابه بین دو پارتیشن متوالی k و $k+1$ است:

$$PPSV(P_k) = |SNPS(P_k)| \quad \text{(رابطه ۲)}$$

تعریف ۶: تعداد زوج تشابه در گراف، مجموع تعداد زوج نودهای مشابه در DFG است:

$$GPSV = \sum_{k=1}^{N-1} PPSV(P_k) \quad \text{(رابطه ۳)}$$

که N تعداد پارتیشن‌ها است.

تعریف ۷: نود ساختگی[۱۸] نودی است که عمل محاسباتی بی اثر انجام می دهد مثل جمع یک عدد با صفر یا ضرب در عدد ۱ و اثری در ترتیب اجرای عملیات در DFG ندارد.

در الگوریتم افراز زمانی مبتنی بر تشابه، ابتدا، DFG ورودی با استفاده از الگوریتم افراز زمانی [۱۰، ۱۱] که زمان سربار بازپیکربندی و محدودیتهای فضایی

را در نظر می‌گیرد، افراز می شود. سپس، لیستی از زوج نودها در هر افراز ایجاد شده و زوج نودهای مشابه در پیکربندی های متوالی به آن اضافه می‌شود. در شکل ۳-الف، یک DFG نشان داده شده است. در شکل ۳-ب این DFG افراز شده و زوج نودهای هر افراز مشخص شده است. شکل ۳-ج شباهت نودهای منفرد در افرازهای متوالی را نشان می‌دهد. نود SUB بین پارتیشن‌های اول، دوم و سوم و نود MUL در دو پارتیشن دوم و سوم مشابه هستند. شکل ۳-د زوج نودهای مشابه موجود در پارتیشن‌های متوالی از DFG را نشان می‌دهد. در این شکل، (MUL, SUB) زوج نود مشابه در پارتیشن‌های دوم و سوم است.

۳-۴ الگوریتم درج نودهای ساختگی

در پیکربندی‌های تولید شده توسط فرآیند طراحی، معمولاً فضاهای بدون استفاده در سخت‌افزار به وجود می‌آید. این فضاها ممکن است به اندازهٔ کافی بزرگ باشند تا بتوان در آن یک ماجول کوچک را اضافه کرد. در این حالت، ماجول‌های کوچک بصورت انتخابی می توانند به هر پارتیشن اضافه شوند تا تشابه بر حسب زوج نودها را افزایش دهند. نود جدید باید یک نود ساختگی باشدکه عمل بی اثری را انجام می‌دهد. الگوریتم درج نودهای ساختگی را به صورت زیر ارائه می‌کنیم:

۱. الگوریتم افراز زمانی را مطابق با روش پیشنهادی در [۱۰، ۱۱] انجام بده.

۲. لیست زوج نودها را برای هر پارتیشن ایجاد کن.

۳.برای $k=0$ تا $N-1$ (N تعداد پارتیشن‌ها است) مراحل زیر را انجام بده (در این مرحله عملیات درج نود بین هر دو پارتیشن متوالی انجام خواهد شد):

۳-۱. یک زوج نود مانند $NP(i, j, k)$ از لیست موجود برای زوج نودهای پارتیشن P_k انتخاب کن.

۳-۲. نود $t \in P_{k+1}$ را به گونه‌ای انتخاب کن که t مشابه با نود $i \in P_k$ باشد و $F(i)=F(t)$. یعنی هر دو نود t و i یک تابع یکسان را پیاده‌سازی کنند و به عبارتی مشابه با یکدیگر باشند. اگر هیچ نودی مشابه با نود i در P_{k+1} وجود نداشته باشد مراحل ۳-۱ و ۳-۲ را تکرار کن.

۳-۳. یک نود ساختگی l مشابه با نود $j \in P_k$ به P_{k+1} اضافه کن.

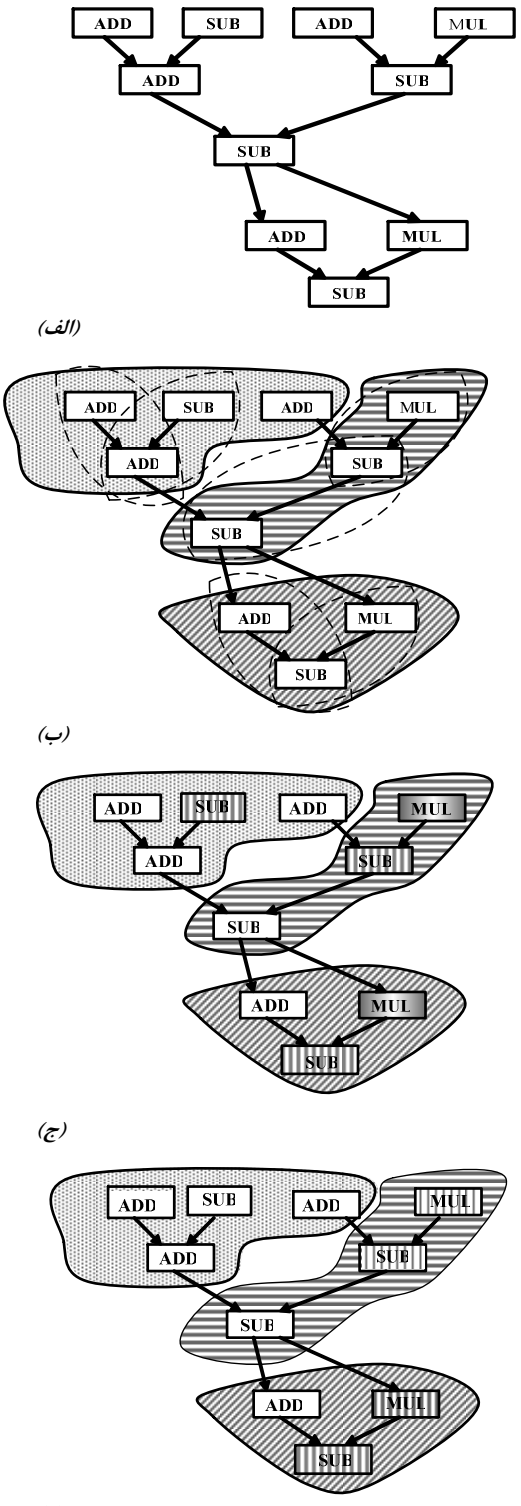
۳-۴. مجموع مساحت پیکربندی به همراه میزان حافظه مورد استفاده برای نگهداری داده‌های میانی بین P_k و P_{k+1} را برای پارتیشن P_{k+1} محاسبه کن.

۳-۵. اگر فضای مورد نیاز برای پارتیشن P_{k+1} کمتر از مساحت سخت-افزار قابل پیکربندی مورد استفاده باشد، در این صورت، عمل درج نود را تأیید کن و تغییرات لازم را برای اضافه شدن نود جدید در پارتیشن P_{k+1} انجام بده. در این حالت زوج نود $NP(t,l,k)$ به $SNPS(P_k)$ اضافه شده و در نتیجه $PPSV(P_k)$ و $GPSV$ هر دو به میزان یک واحد افزایش می‌یابند.

۳-۶. اگر فضای مورد استفاده توسط پارتیشن P_{k+1} بزرگتر از مساحت سخت‌افزار مورد استفاده باشد، نود ساختگی اضافه شده به P_{k+1} را حذف کن که در این حالت مقادیر $PPSV(P_k)$ و $GPSV$ ثابت باقی می‌مانند.

۳-۷. مراحل ۳-۲ الی ۳-۶ را برای تمامی زوج نودهای موجود در پارتیشن P_k تکرار کن.

در شکل ۴-الف، یک DFG افراز شده دیده می‌شود. تنها یک زوج نود (MUL,SUB) در پارتیشن‌های دوم و سوم وجود دارد، بنابراین $PPSV(P_2)= 1$ و $GPSV= 1$. افزودن نود ساختگی ADD به پارتیشن دوم، زوج نود جدید (SUB,ADD) را به آن اضافه می‌کند. این زوج مشابه با زوج (SUB,ADD) در پارتیشن اول است. به علاوه، افزودن نود ساختگی SUB در پارتیشن سوم، زوج نود جدید دیگری بصورت (SUB,SUB) را به پارتیشن دوم و سوم اضافه می‌کند (شکل ۴-ب) . بنابراین، تعداد زوج‌های مشابه در DFG به ۳ افزایش پیدا می‌کند $(GPSV_{new}=3)$.



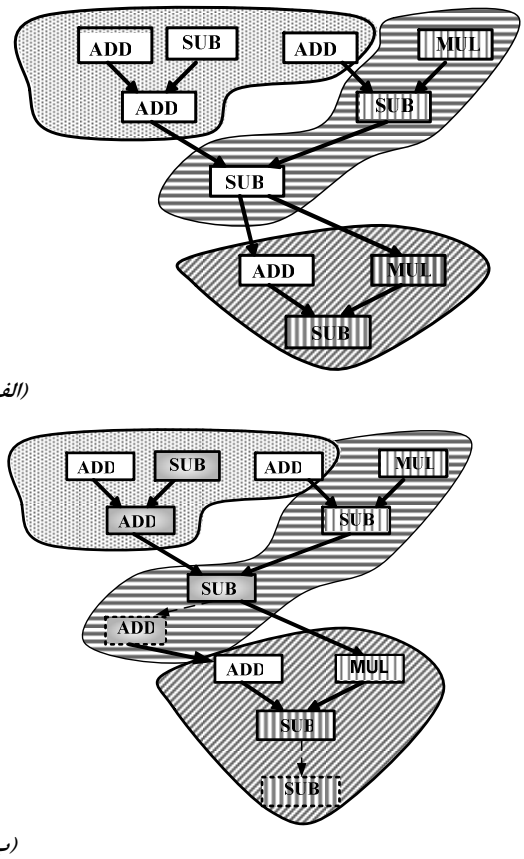
شکل ۳- (الف) یک گراف جریان داده (DFG)، (ب) یک DFG افراز شده و زوج نودهای آن (ج) نودهای منفرد مشابه در دو پارتیشن متوالی (د) زوج نودهای مشابه

۴-۴ انتخاب بین مشابهت زوج نود یا نود منفرد

در بخش‌های قبل، الگوریتم‌های مربوط به نودهای منفرد مشابه و زوج نودهای مشابه برای افراز زمانی ارائه شدند. در روش اول، تعداد نودهای منفرد مشابه در دو پیکربندی متوالی در نظرگرفته شد. این تشابه، زمان جایابی را کاهش می‌دهد و اثری در فرآیند مسیریابی ندارد. ازطرفی دیگر، در روش دوم که تکیه بر کشف زوج نودها دارد، اتصالات بین ماجول‌ها نیز در نظر گرفته می‌شود. بنـــابـــراین، نودهای

مشابه در دو افراز متوالی در طول فرآیند جایابی بر روی سخت‌افزار بدون تغییر باقی می‌مانند. بعلاوه، با استفاده از تشابه زوج نودها، پیچیدگی و زمان اجرا در هر دو فرآیند جایابی و مسیریابی کاهش می‌یابد. از طرفی در فرآیند بازپیکربندی زمان اجرا، تأخیر ناشی از باز پیکربندی نیز کاهش می یابد. در حالت تشابه منفرد، بخشی از رشته بیت‌هایی که مربوط به جایابی ماجول‌های مشابه هستند دوباره داخل سخت افزار بارگذاری نمی‌شوند و زمان بازپیکربندی بلوک‌های منطقی کاهش می‌یابد. از طرفی دیگر، در حالت تشابه زوج نود، رشته بیت‌های هر دو مرحلهٔ جایابی و مسیریابی مربوط به زوج نود مشابه و اتصالات بین آنها نیازی به بارگذاری مجدد در سخت افزار ندارند. بنابراین، زمان مورد نیاز برای باز پیکربندی بلوک‌های منطقی و منابع مسیریابی کاهش می‌یابد.

فرض کنید که S تعداد نودهای مشابه منفرد و P تعداد زوج نودهای مشابه در DFG باشد. بنابراین، تعداد S ماجول وجود دارد که مکان آنها در دو پیکربندی متوالی بدون تغییر باقی می‌ماند. مقدار S در زمان اجرای فرآیند جایابی (مرحله‌ی کامپایل) و زمان باز پیکربندی (مرحله اجرا) اثر دارد. از طرفی دیگر، P ×۲ نود مشابه و P اتصال مشابه در پیکربندی‌ها وجود دارد. بنابراین، جایابیP ×۲ماجول و مسیریابی P اتصال انجام نمی‌شود و همچنین نیازی نیست که در پیکربندی زمان اجرا، بازپیکربندی P×۲ بلوک منطقی و قسمتی از منابع مسیریابی انجام شود.



شکل ۴- (الف) یک DFG پارتیشن‌بندی شده که دارای یک زوج نود مشابه است، (ب) الگوریتم درج نود ساختگی با افزودن نود ساختگی ADD زوج نود (SUB,ADD) را به پارتیشن دوم اضافه می‌کند.

فرض می کنیم که γ تاثیراتصالات باشد که وزن مسیریابی اتصالات را در مقابل جایابی بلوک‌های منطقی نشان می‌دهد. به عنوان مثال، γ برابر ۲ به این مفهوم است که مسیریابی اتصالات بین دو ماجول اثری معادل با جایابی دو ماجول دارد. به این ترتیب بر اساس میزان تشابه حاصل، یکی از روش‌های تشابه منفرد یا زوج نود را برای تولید افرازها انتخاب می‌کنیم. اگر اثر مشابهت‌های منفرد (S) بزرگتر از $2p + 2P \times \gamma$ باشد، دراین صورت روش تشابه منفرد را انتخاب می‌کنیم و در غیر این صورت روش تشابه زوج نود انتخاب می‌شود. ما در پیاده‌سازی انجام شده، افراز زمانی را با هر دو روش انجام داده و سپس بر اساس معیار ذکر شده که به نوعی حجم رشته بیت مشابه در پیکربندی‌ها و میزان صرفه‌جویی در زمان باز پیکربندی را نشان می‌دهد، افراز حاصل از یکی از این دو روش را برمی‌گزینیم.

۵- افراز زمانی و طراحی فیزیکی مجتمع

همانطور که قبلاً نیز گفته شد، تأکید روش ما بر هر دو مرحلهٔ طراحی فیزیکی و افراز زمانی است. روش‌های جایابی می‌توانند در کاربردهای مختلف به صورت online یا offline مورد استفاده قرار گیرند. هر دو حالت جایابی online و offline در کاربردهای RCS دارای اهمیت هستند [۲] . نسخه‌ی online در سیستمهای پویا قابل استفاده است. بدلیل اینکه پیش بینی دقیق زمان اجرای یک الگوریتم عمومی در زمان کامپایل مشکل است، لذا انجام جایابی برای مدار پویا در زمان اجرا انجام می‌شود. در این حالت معمولاً عملیات پیش جایابی و مسیریابی ماجول‌ها که ماجول سخت نیز نامیده می شود قبل از زمان اجرا در راستای کاهش زمان جایابی در افزاره های قابل باز پیکربندی انجام می‌شود. از طرفی دیگر مهم-ترین ویژگی الگوریتم‌های جایابی offline کیفیت جایابی تولید شده است با وجود اینکه روند اجرای کندتری دارند. در نسخه offline جایابی بر روی یک نت لیست از ماجول‌های مدار انجام می‌شود و زمان کافی برای جایابی و رسیدن به شرایط بهینه وجود دارد. روش offline به عنوان زیربرنامه ای برای الگوریتم online (بعنوان مثال پیش جایابی ماجول‌ها) و همچنین به عنوان روشی پایه برای رسیدن به جایابی با کیفیت بالا در روش‌های online استفاده می‌شود [۲]. در بخش بعدی مقاله، یک روش مبتنی بر طراحی فیزیکی مجتمع با افراز زمانی مورد استفاده در یک ابزار طراحی کاربردهای ایستا معرفی می‌شود.

در این مقاله از دو روش متفاوت جایابی مبتنی بر ماجول‌های سفت و سخت استفاده می‌کنیم. در بخش بعد، نحوه‌ی به کارگیری ماجول‌های سخت پیش جایابی شده ارائه و پس از آن نحوه استفاده از ماجول‌های سفت بیان می‌شود. به کمک ماجول‌های سخت، جایابی سریع‌تری انجام می‌شود و در صورت استفاده از ماجول‌های سفت کیفیت بالاتری بدست می‌آید که فضای تلف شده‌ی (فضای خالی) کمتری دارد.

۵-۱ طراحی فیزیکی با استفاده از ماجول‌های سخت

یکی از اهداف ما در این مقاله، ارائه یک روش مجتمع برای انجام افراز زمانی و مراحل طراحی فیزیکی مطابق با متدولوژی طراحی ارائه شده است. بنابراین نیازمند ابزاری هستیم که جایابی و مسیریابی پارتیشن‌های حاصل از مرحله‌ی افراز زمانی را انجام دهد. VPR ابزار معروفی برای جایابی و مسیریابی افزاره های FPGA است. ابزارهای دیگری برای جایابی و مسیریابی افزاره های FPGA مثل Frontier [۲۶] موجود است که درحال حاضر در دسترس نیستند. به هر حال با توجه به در دسترس بودن و باز بودن متن VPR آن را برای پیاده‌سازی ایده‌های خود انتخاب کردیم. VPR الگوریتم‌های جایابی و مسیریابی FPGA مرسوم را استفاده می‌کند ولی ما VPR را برای جایابی و مسیریابی پیکربندهای حاصل از افراز زمانی تغییر دادیم. VPR الگوریتم Simulated Annealing را برای جایابی بکار می‌گیرد. الگوریتم‌های تکراری نظیر Simulated Annealing بدلیل سادگی مدل‌سازی محدودیتهای افزاره‌های FPGA پیچیده، روش‌های متداولی برای حل مسأله‌ی جایابی هستند [۲۷، ۲۳].

در فرآیند افراز زمانی بایستی توجه داشت که هر سیگنال خروجی در پارتیشن تولید شده باید تا زمانی که دوباره در پارتیشن‌های دیگر مورد نیاز شد در حافظه نگهداری شود. هزینه‌ی ارتباط هر پارتیشن با پارتیشن‌های دیگر برابر تعداد

سیگنال‌های خروجی پارتیشن مذکور است که در پارتیشن‌های دیگر به آنها نیاز وجود دارد. بنابراین باید تعدادی ثبات و یا سلول حافظه برای نگهداری داده و انتقال آنها بین پارتیشن‌ها مورد استفاده قرار گیرند. بر این اساس پارامتر بهره‌گیری حافظه[۱۹] در پیکربندی i ام را به صورت زیر تعریف می‌کنیم [۱۱].

MU_i = تعداد سیگنال‌های خروجی پیکربندی i که به عنوان سیگنال ورودی در پارتیشن بعدی مورد استفاده قرار می‌گیرد.

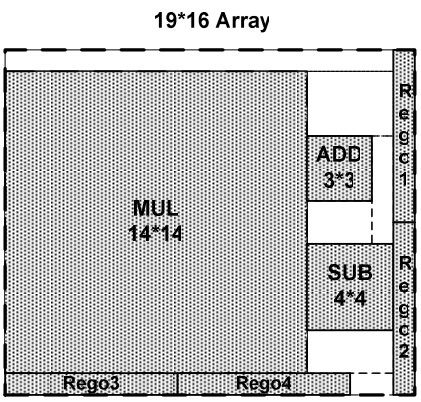
برای پیکربندی آخر ($i=N$ ، N تعداد پیکربندی‌ها است)، MU_N برابر صفر است چرا که همهٔ سیگنال‌های خروجی این پارتیشن نهایی بوده و در هیچ پارتیشن دیگری استفاده نمی‌شوند.

پارامتر بهره‌گیری از حافظه برای یک DFG[۲۰]، حداکثر تعداد سلول‌های حافظه است که در یک پیکربندی برای نگهداری مقدار سیگنال‌ها استفاده می‌شوند. به عبارت دیگر:

DMU= حداکثر بهره‌گیری از حافظه برای DFG= $Max(MU_i)_{i=1...N-1}$

در فرآیند طراحی بایستی برای هر پارتیشن یک نت‌لیست تولید و به ابزار طراحی فیزیکی اعمال شود. برای این منظور باید تعدادی سلول حافظه نیز متناسب با میزان بهره‌گیری DFG از حافظه داده شود. به این ترتیب با مشخص شدن اعضای هر پارتیشن، تعداد سلول‌های حافظه‌ی لازم برای آن برآورد شده و نت‌لیست مربوط به آن که شامل سلول‌های حافظه نیز می‌باشد تولید می‌گردد. نت‌لیست حاصل توسط نرم‌افزار تغییر یافته VPR جایابی و مسیریابی می‌شود تا پیکربندی نهایی بر روی یک FPGA با معماری *island-style* [۲۷، ۲۸] تولید گردد. این معماری متشکل از یک ماتریس از بلوک‌های منطقی قابل برنامه‌ریزی (CLB) است که توسط یک مش یکنواخت از منابع مسیریابی احاطه شده‌اند. هر CLB شامل یک یا چند LUT[۲۱] (عناصر قابل برنامه‌ریزی که امکان پیاده‌سازی انواع توابع منطقی را فراهم می‌کنند)، تعداد کمی ورودی، تعدادی گیت ساده‌ی منطقی و یک یا چند فلیپ فلاپ است [۲۶].

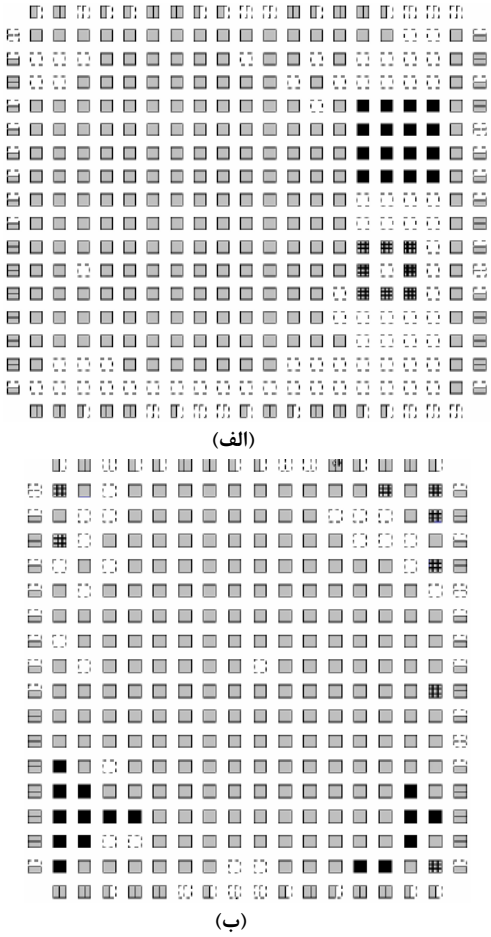
بعد از مرحلهٔ افراز زمانی، جایابی ماجول‌های سخت توسط الگوریتم SA انجام می‌گردد. کتابخانه‌ای از ماجول‌های سخت تولید شده و در دسترس می‌باشد. جایابی ماجول‌ها در طول فرآیند کامپایل مطابق با نت لیست هر پارتیشن انجام می‌گردد. شکل ۵ جایابی یک پارتیشن را مبتنی بر ماجول‌های سخت نشان می‌دهد. مشاهده می‌شود که در فرآیند جایابی ممکن است فضای نسبتاً بزرگی از سخت افزار به دلیل عدم انعطاف‌پذیری ماجول‌های سخت تلف گردد.



شکل ۵- جایابی ماجول‌های سخت در سخت‌افزار. مستطیل‌های سفید بیانگر فضاهای تلف شده هستند.

۲-۵ طراحی فیزیکی افزایشی با استفاده از ماجول‌های سفت

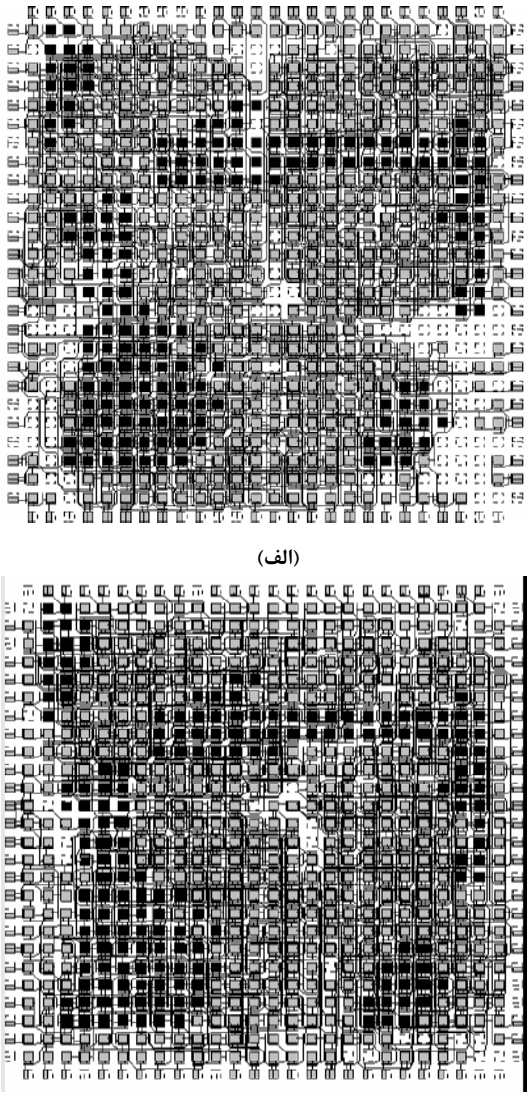
به کارگیری ماجول‌های سخت در مرحله‌ی جایابی می‌تواند فرآیند تولید پیکربندی را تسریع نماید. در نتیجه این روش برای سیستم‌های قابل بازپیکربندی پویا مناسب است. آزمایش‌ها نشان می دهند (به بخش ۷ مراجعه کنید) استفاده از ماجول‌های سخت فضای تلف شده‌ی بیشتر و پیکربندی با کیفیت پایین‌تر را در مقایسه با پیکربندی‌های تولید شده بر اساس ماجول‌های سفت دارد. شکل ۶ مکان CLBها را برای دو عملیات مختلف جایابی مبتنی بر ماجول‌های سخت (شکل ۶-الف) و سفت (شکل ۶-ب) نشان می‌دهد. در حالت دوم CLBهای هر ماجول روی FPGA توزیع می‌شوند تا کیفیت جایابی را به نحو مطلوب‌تری بهبود دهند. ما استفاده از ماجول‌های سفت به جای ماجولهای سخت را با وجود کندتر بودن آن برای به دست آوردن کیفیت بالا در پیکربندی‌ها پیشنهاد می‌کنیم. یکی از مزایای استفاده از ماجول‌های سفت که به طور ضمنی به آن دست پیدا می‌کنیم این است که میزان پراکندگی در سطح سخت‌افزار قابل برنامه‌ریزی نسبت به حالتی‌که از ماجول‌های سخت استفاده می‌شود تقلیل یابد. دلیل کاهش پراکندگی در سطح تراشه، انعطاف پذیری ماجول‌های سفت و امکان جابجایی اجزای آنها در نقاط مختلف تراشه می‌باشد.



شکل ۶- جایابی ماجول‌ها در FPGA، (الف) جایابی مبتنی بر ماجول‌های سخت (ب) جایابی مبتنی بر ماجول‌های سفت که در آن CLB های مربوط به ماجول‌ها در بخش‌های مختلف FPGA توزیع شده‌اند.

یک هدف دیگر این است که تا حد امکان از مزایای روش افراز زمانی پیشنهادی در فرآیند طراحی فیزیکی برای رسیدن به کارآیی بالاتر در سیستم قابل باز پیکربندی استفاده کنیم. بخش‌هایی از الگوریتم جایابی VPR تغییر داده شده است به گونه‌ای که بتواند بر مبنای وجود ماجول‌های مشابه در پیکربندی‌های متوالی تنها جایابی بخشی از ماجول‌ها را به جای تمامی آنها انجام دهد. در ابزار ارائه شده، بعد از تولید اولین پیکربندی، جایابی سایر افزارها بصورت افزایشی انجام می‌شود. بر اساس جریان طراحی پیشنهادی، CLB مشترک در دو پیکربندی متوالی در فاز جایابی پیکربندی دوم ثابت و بدون تغییر باقی می‌مانند. بدلیل اینکه در الگوریتم جایابی افزایشی، جابجایی و انتقال ماجول‌های ثابت شده انجـــام نمی

گیرد، لذا زمان اجرای فرآیند جایابی کاهش می‌یابد. شکل ۷ دو پیکربندی تولید شده توسط ابزار ارائه شده را نشان می‌دهد. این پیکربندی ها مربوط به دو پارتیشن متوالی از یک DFG هستند. مکان‌های قرارگیری ماجول‌های مشابه در دو پارتیشن تثبیت شده‌اند و به عبارت دیگر در پیکربندی دوم نیازی به جایابی مجدد ماجول‌هایی که در پارتیشن قبل نیز وجود داشته‌اند (ماجول‌های مشابه) نمی‌باشد. مربع‌های سیاه در این شکل، CLB های مشابه در پیکربندی‌های متوالی را نشان می‌دهند.



(الف)

(ب)

شکل ۷- (الف) و (ب) دو پیکربندی متوالی که به وسیله‌ی نسخه‌ی تغییر یافته VPR جایابی و مسیریابی شده‌اند. مربع‌های سیاه CLBهای مربوط به ماجول‌های مشابه در دو پیکربندی هستند که محل آنها بعد از انجام جایابی ثابت مانده است.

۶- کاهش مصرف حافظه

در روش اولیهٔ پیشنهادی برای افراز زمانی، یکی از مشکلات در ایجاد پیکربندی‌ها، سربار زیاد پین‌های ورودی/خروجی و سلول‌های حافظه می‌باشد. الگوریتم پیشنهادی، DFG را با ترتیب افقی یا سطری افراز می‌کند. با توجه به اینکه اندازهٔ آرایه انتخاب شده برای سخت‌افزار تا حد زیادی وابسته به تعداد پین‌های ورودی/خروجی است، لذا زیاد بودن تعداد پین‌های ورودی/خروجی ممکن است نسبت سلول‌های حافظه به CLBهای منطقی را افزایش دهد. بنابراین با توجه به اختصاص یک آرایه‌ی بزرگ برای مداری که چندان بزرگ نبوده ولی دارای پین های ورودی/خروجی زیاد می‌باشد، تعداد زیادی CLB در سطح FPGA بدون استفاده خواهد ماند. در این بخش، روش افراز زمانی دیگری ارائه می‌گردد که سعی دارد سربار پین‌های وردی/خروجی را به همراه مصرف حافظه کاهش دهد. جزئیات این الگوریتم به شرح زیر است:

الگوریتم افراز زمانی با هدف کاهش حافظه مصرفی:

۱. سطح زمان‌بندی اجرای هر نود از *DFG* را بر اساس الگوریتم زمان‌بندی *ASAP* تعیین کن.

۲. مقدار l را برابر صفر قرار بده (l بیانگر سطحی از *DFG* است که در حال حاضر باید پردازش شود).

۳. یک پارتیشن خالی در نظر گرفته و لیست نودهایی را که در *DFG* دارای سطح l هستند تهیه کن و در یک آرایه قرار بده.

۴. برای $i=0$ الی تعداد نودهای آرایه‌ی ایجاد شده مراحل زیر را انجام بده:

۴-۱. یک نود از آرایه انتخاب کن که تا به حال در هیچ پارتیشنی قرار نگرفته است.

۴-۲. مقدار l را در صورتی‌که هیچ نودی برای پارتیشن کردن وجود نداشته باشد کاهش بده.

۴-۳. نود انتخاب شده را در صورتی‌که اندازه‌ی پارتیشن از اندازه‌ی افزاره تجاوز نکند و تمامی والدین آن قبلاً پارتیشن شده باشند به پارتیشن جاری اضافه کن.

۴-۴. مقدار l را یک واحد افزایش بده تا نودهای موجود در سطح بعدی مورد پردازش قرار گیرند و به مرحله‌ی ۳ برو.

۴-۵. مقدار l را در صورتی که هیچ نودی شرایط قرار گرفتن در پارتیشن جدید را نداشته باشد کاهش بده.

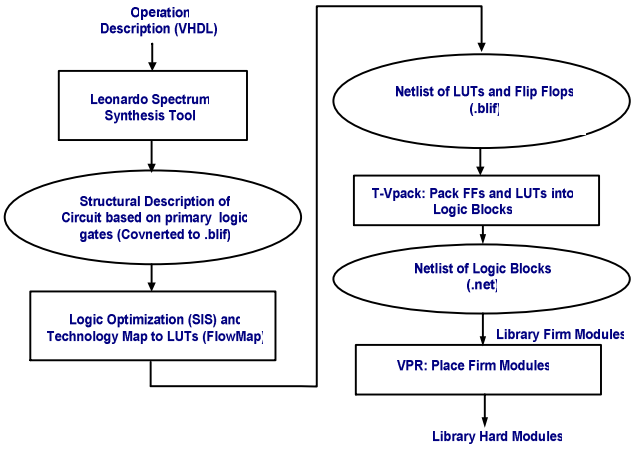
۴-۶. فرآیند افراز را در صورتی که تمامی نودهای *DFG* افراز شده باشند خاتمه بده، در غیر این صورت، مرحله‌ی ۴ را تا زمانی که اندازه‌ی پارتیشن جاری کمتر از اندازه‌ی سخت‌افزار باشد تکرار کن.

۵. یک پارتیشن جدید خالی ایجاد و مراحل ۳ و ۴ را تکرار کن.

شکل ۸ نتیجه‌ی دو افراز مختلف از یک DFG را نشان می‌دهد. در اولین حالت (شکل ۸-الف) افراز بر اساس الگوریتم افراز زمانی قبلی انجام شده است. الگوریتم افراز زمانی جدید، افراز DFG را با درنظرگرفتن وابستگی داده‌ها و در عمق آن انجام داده است. این الگوریتم به جای پیمایش افقی DFG، پیمایش آن را به صورت عمودی و در عمق انجام می‌دهد. در این روش مصرف حافظه برای DFG مورد نظر تا حداکثر ۸۰٪ کاهش می‌یابد که در نهایت منجر به کاهش اندازه آرایه نیز می‌شود (شکل ۸-ب) .آزمایش‌های ما بهبود در مصرف حافظه و اندازه‌ی FPGA را در الگوریتم افراز زمانی جدید نشان می دهد.

۷- نتایج آزمایش

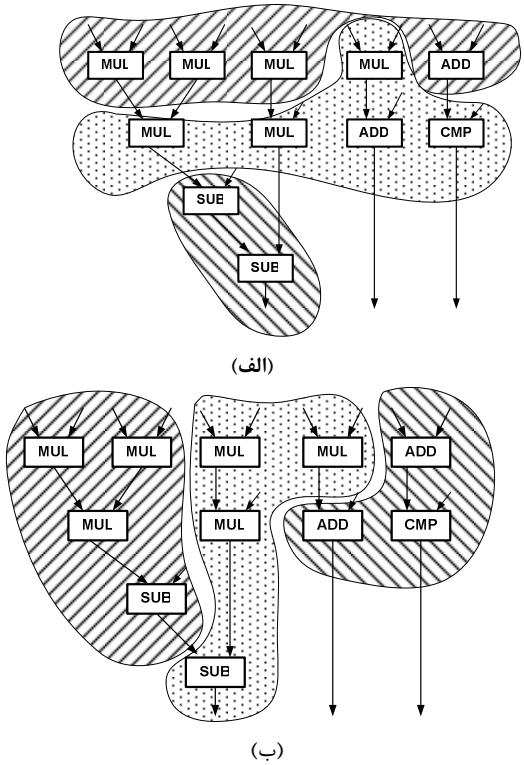
در روش پیشنهادی، ورودی بصورت یک DFG دریافت می‌شود. نودها در DFG ورودی ، ماجول‌های سفت یا سخت هستند. یک کتابخانه شامل ماجول‌های سفت و کتابخانه دیگری برای ماجول‌های سخت تهیه شده است. شکل ۹ جریان طراحی مورد استفاده برای تولید ماجول‌ها را مشخص می‌کند. ماجول‌های سفت بصورت نت‌لیست و ماجول‌های سخت بصورت نسخه‌ی جایابی شدهٔ همان ماجول‌ها می‌باشند. در ابتدا، هر ماجول با کد VHDL توصیف شده و سپس توسط ابزار سنتز Leonardo Spectrum برای به دست آوردن توصیف ساختاری ماجول‌ها، بر اساس گیت‌های منطقی سنتز می‌گردد. مجموعه ابزار سنتزSIS [۳۰] برای بهینه-سازی منطقی مستقل از تکنولوژی هر ماجول استفاده می‌شود. سپس هر مدار توسط ابزار Flowmap [۳۱] به مداری شامل LUT و فلیپ فلاپ‌ها سنتز می‌شود. خروجی Flowmap نت لیستی از LUTها و فلیپ فلاپ‌ها با فرمت blif است.



شکل ۹- مراحل تولید ماجول‌های کتابخانه

جدول ۱- تعدادی عملگر در نظر گرفته شده برای کتابخانه‌ی ماجول‌ها به همراه تعداد CLB های اشغال شده توسط آنها

نوع عملگر	تعداد CLB ها در ماجول سفت	ابعاد ماجول سخت
ADD	۸	۳×۳
SUB	۱۶	۴×۴
XOR	۸	۳×۳
CMP	۹	۳×۳
MUX	۸	۳×۳
MULT	۱۷۲	۱۴×۱۴
ROT	۸	۳×۳



شکل ۸- (الف) افراز با پیمایش افقی (ب) افراز با پیمایش عمودی

سپس T-VPACK [۲۷، ۲۸] نت لیست متشکل از LUT و فلیپ فلاپ‌ها را داخل بلوک‌های منطقی دانه درشت قرار داده و نت لیستی با فرمت net. تولید می‌کند و VPR [۲۷] ماجول را برای تولید ماجول سخت جایابی می‌نماید. به این ترتیب، نودهای DFG، بصورت ماجول‌های سفت و سخت تولید می‌شوند و به کتابخانه مخصوصی اضافه می‌گردند.

معماری افزاره‌ی برنامه پذیر مقصد معماری island-style می‌باشد. VPR یک پروفایل بکار می‌گیرد که در آن می‌توان جزئیات معماری را مشخص کرد. پروفایل معماری بکار رفته با نام 4-LUTsanitized.arch در [۲۸] قابل دسترسی است. فرض می‌کنیم که این معماری دارای قابلیت بازپیکربندی جزئی است. جدول ۱ برخی از ماجول‌های سفت و سخت ذخیره شده در کتابخانه را همراه با اندازه‌ی آنها بر مبنای تعداد CLBها نشان می‌دهد. تا جایی که اطلاع داریم نمونه‌های استاندارد DFG ایستا به طور مشخص وجود ندارد.

ما شش DFG ایستا را انتخاب کرده و آنها را به عنوان ورودی به ابزار بکار می‌بریم. پنج نمونهٔ اول از [۸] و [۲۹] انتخاب شده‌اند و ششمی یک DFG برای الگوریتم رمزنگاری FEAL (شکل ۱۰) [۳۲] است. به عنوان یک نمونه، DFG مربوط به توابع اصلی الگوریتم FEAL را در شکل ۱۰-الف مشاهده می‌کنید. توابع اصلی الگوریتم FEAL شامل *ADD* ، ثبات چرخش دو بیتی و *XOR* می باشد. الگوریتم افراز زمانی، پنج افراز را برای الگوریتم FEAL مطابق شکل ۱۰-ب تولید کرده است.

DFGهای ورودی در ابتدا بر اساس ماجول‌های سخت روی افزارهٔ قابل بازپیکربندی پیاده‌سازی شدند. مطابق شکل۱۱-الف برای هر پارتیشن از DFG، نت لیستی از ماجول‌های سخت تولید شده و سپس جایابی ماجول‌های سخت انجام شد. در نهایت، VPRبرای مسیریابی اتصالات پیکربندی بکارگرفته شد. جدول ۲ نتایج این پیاده سازی را نشان می‌دهد. در مرحلهٔ دوم، ماجول‌های سفت به جای ماجول‌های سخت استفاده شد. شکل ۱۱-ب جریان طراحی را برای تولید پیکربندی نهایی بر اساس ماجول‌های سفت نشان می‌دهد. بعد از تولید پارتیشن‌ها، یک نت لیست کامل ایجاد و سپس جایابی و مسیریابی برای ایجاد پیکربندی‌های نهایی انجام می‌شود. جدول ۳، نتایج حاصل از تولید پیکربندی مبتنی بر ماجول‌های سفت را نشان می‌دهد. تعداد CLBهای استفاده شده برای هر پیکربندی شامل تعداد CLBهای تخصیص یافته به ثبات‌های ورودی/خروجی (سلول‌های حافظه) در جدول ۳ نشان داده شده است. ستون آخر جدول ۳ نسبت سلول‌های حافظه به مجموع تعداد منابع منطقی را نشان می‌دهد. ثبات‌های ورودی و خروجی سربار زیادی در پیاده‌سازی قابل بازپیکربندی دارند.

نتایج بدست آمده از به کارگیری ماجول‌های سفت و سخت را برای ارزیابی اثرات استفاده از ماجول‌های سفت به جای سخت با هم مقایسه می‌کنیم. جدول ۴ مقایسهٔ این دو حالت را نشان می‌دهد. اعداد مثبت بیانگر میزان بهبود حاصل از استفاده‌ی ماجول‌های سفت نسبت به سخت و اعداد منفی بیانگر بهبود به کارگیری ماجول‌های سخت نسبت به سفت است. استفاده از ماجول‌های سخت، بلوک‌های منطقی و منابع مسیریابی بیشتری را تلف می‌کند. همچنین هزینهٔ جایابی نیز افزایش می یابد. در این حالت حداکثر تاخیر مسیر بحرانی معمولاً بزرگتر است، چون منابع مسیریابی بیشتری مصرف شده است. از طرفی دیگر به کارگیری ماجول‌های سخت زمان جایابی کمتری را در مقایسه با ماجول‌های سفت مصرف می‌کند (جدول ۵) که ممکن است برای کاربردهای online مناسب‌تر باشد. الگوریتم‌ها بر روی رایانه‌ی شخصی Pentium IV 2400MHz با 512MB RAM اجرا شده‌اند.

حال نشان می دهیم که چگونه افراز زمانی مبتنی بر تشابه و طراحی فیزیکی افزایشی می تواند کارایی کاربردها را در سیستم‌های قابل بازپیکربندی افزایش دهد.

طراحی مورد استفاده برای تولید پیکربندی‌های مبتنی بر شباهت نودهای منفرد و زوج نود را مشاهده می‌کنید. DFG ورودی بعد از افراز زمانی توسط هر دو الگوریتم تکمیلی ارائه شده پردازش شد و شباهت پارتیشن‌های متوالی مبتنی بر نودهای منفرد و زوج نودها به دست آمد. خروجی یکی از این دو روش در نهایت انتخاب و نت‌لیست هر پارتیشن بر مبنای آن تولید گردید. در انتهای کار جایابی و مسیریابی پارتیشن انجام و پیکربندی نهایی تولید شد.

جدول ۲- نتیجه پیاده‌سازی DFGهای ورودی با استفاده از ماجول‌های سخت

گراف جریان داده	اندازه سخت-افزار	حداکثر هزینه جایابی	پهنای کانال مسیریابی	حداکثر تأخیر مسیر بحرانی(نانوثانیه)
DFG1	۱۹×۱۶	۱۹/۳۰	۸	۶۱/۱۵
DFG2	۱۹×۱۷	۲۷/۷۴	۸	۶۱/۱۷
DFG3	۱۵×۱۶	۱۸/۸۴	۶	۶۱/۷۹
DFG4	۱۹×۱۶	۳۲/۰۹	۹	۶۰/۵۷
DFG5	۲۹×۲۹	۸۳/۲۴	۸	۶۵/۳۰
DFG6 (FEAL)	۱۵×۱۵	۱۹/۷۳	۷	۳۵/۴۱

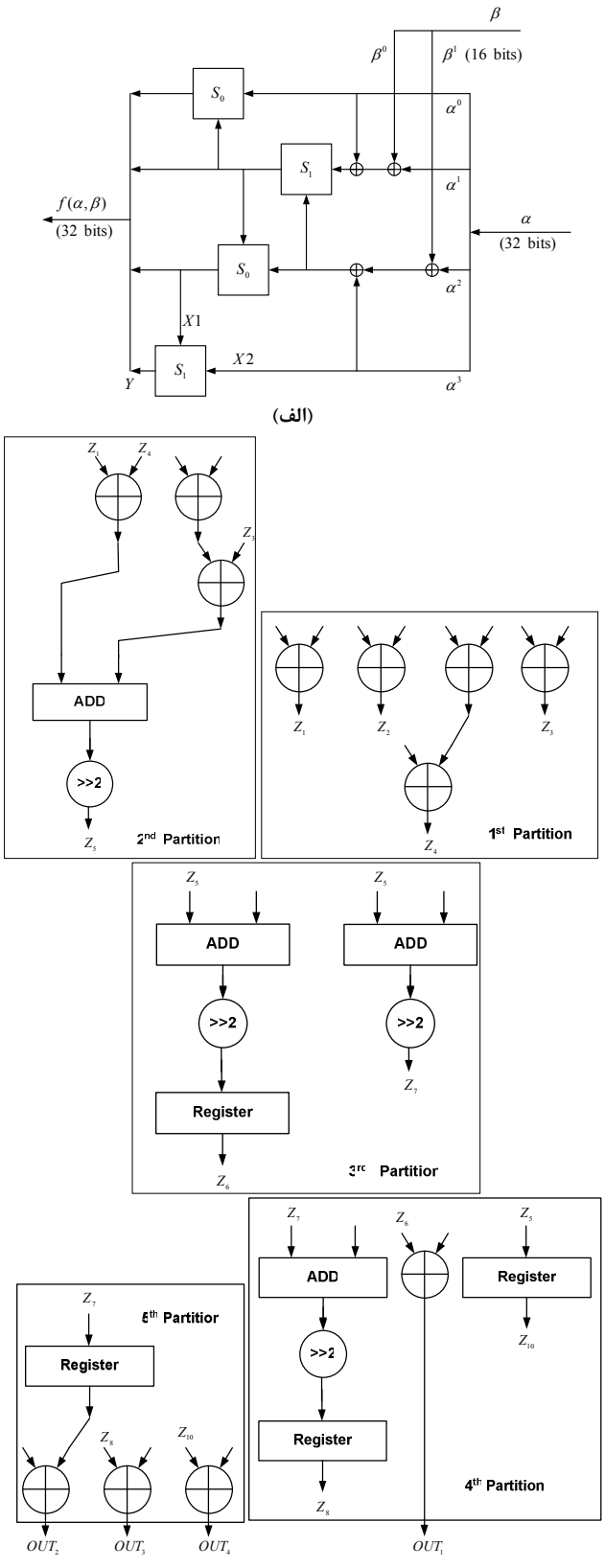
دو الگوریتم تکمیلی مبتنی برشباهت درجدول ۶ با هم مقایسه شده‌اند. آزمایش‌ها نشان می‌دهند که تکنیک درج نود ساختگی می‌تواند تعداد زوج نودها را در پیکربندی‌های متوالی افزایش دهد. علاوه بر این زمان اجرای فرآیند جایابی نیز متناسب با آن کاهش می‌یابد. تا جائی که مطلع هستیم، ابزاری که بتواند زمان پیکربندی منابع مسیریابی را در زمان اجرا به صورت واقعی تخمین زده و همچنین نسبت زمان برنامه‌ریزی منابع مسیریابی به زمان برنامه‌ریزی بلوک‌های منطقی را اعلام نماید وجود ندارد. بنابراین مطابق با [۱۵]، فرض می‌کنیم که رشته بیت مربوط به برنامه‌ریزی منابع مسیریابی حداقل ۷۰ درصد حجم کل رشته بیت مربوط به برنامه‌ریزی سخت‌افزار را شامل می‌شود. جدول ۶ تأثیر الگوریتم کشف شباهت نودهای منفرد را بر روی جایابی و همچنین تأثیر روش مبتنی بر کشف شباهت زوج نودها بر روی هر دو فرآیند جایابی و مسیریابی را در زمان کامپایل طرح نشان می‌دهد. علاوه بر این، افزایش سرعت بازپیکربندی در جدول ۷ نشان داده شده است.

برای ارزیابی کارآیی روش مبتنی بر مشابهت نودها، جایابی و مسیریابی غیرافزایشی و افزایشی را برای نمونه‌های آزمایشی موجود انجام دادیم. نتایج آزمایش‌های انجام شده کاهش سربار زمان باز پیکربندی را با استفاده از روش افزایشی در شکل ۱۳ نشان می‌دهد. در این شکل بهبود زمان باز پیکربندی و بدنبال آن کاهش زمان اجرای کل نشان داده شده است. برای همه DFGها، زمان باز پیکربندی به دلیل وجود بخش‌های مشابه و مشترک در پیکربندی‌های متوالی، کاسته شده است. به دنبال آن، زمان مورد نیاز برای اجرای مرحلهٔ جایابی نیز کاهش یافته چرا که CLBهای مشابه در مکان‌های خود تثبیت شده‌اند.

در یک آزمایش دیگر، روش‌های افزایشی و غیر افزایشی را با معیارهای مختلفی نظیر هزینهٔ جایابی (طول سیم‌ها [۳۳])، تاخیر مسیر بحرانی و تعداد کانال‌های مسیریابی بکار رفته در سخت‌افزار مقایسه کردیم (جداول ۸ و ۹).

کیفیت پیکربندی‌های تولید شده در روش افزایشی در مقایسه با نتایج بدست آمده از روش غیرافزایشی قابل قیاس بوده و در روش افزایشی زمان کمتری برای تولید پیکربندی‌ها مصرف شده است (جدول ۸).

به بیان دیگر، افزایش تشابه پیکربندی‌ها و بکارگیری روش افزایشی، کاهش هزینهٔ جایابی و اغلب سرعت بالاتر را برای پیکربندی‌های تولید شده به دنبال دارد. همچنین جدول ۹ نشان می دهد که برای بعضی از DFGها ، تعداد کانال‌های مسیریابی کاهش می‌یابد به این دلیل که در روش افزایشی کیفیت بهتری برای جایابی حاصل شده است.



شکل ۱۰- (الف) توابع اصلی الگوریتم رمز FEAL (ب) پارتیشن‌های حاصل از چارچوب کاری ارائه شده

فرض می کنیم که زمان بازپیکربندی می تواند توسط یک تابع خطی از مجموع فضای سخت‌افزار تخمین زده شود. از سوی دیگر زمان پیکربندی کامل برای یک FPGA مشخص ثابت است. تعدادی آزمایش بر اساس الگوریتم افراز زمانی مبتنی بر شباهت نودهای منفرد و زوج نـــودها انجام گرفتـــه است. در شکل ۱۲، مراحل

جدول ۳- نتایج پیاده‌سازی DFG ها به صورت قابل باز پیکربندی

گراف جریان داده	نعداد پیکربندی‌ها	شماره پیکربندی	CLBتعداد	تأخیر مسیر بحرانی (نانو ثانیه)	پهنای کانال مسیریابی	حافظه مصرفی
DFG1	۲	۱	۲۳۳	۶۰/۰۳	۴	٪۲۴
		۲	۸۱	۲۹/۳۴	۳	٪۴۱
DFG2	۲	۱	۲۸۲	۶۲/۲۴	۵	٪۱۷
		۲	۲۵۸	۶۴/۶۷	۵	٪۱۶
DFG3	۲	۱	۲۴۲	۵۳/۱۹	۴	٪۲۷
		۲	۴۹	۸/۴۴	۳	٪۵۱
DFG4	۳	۱	۳۵۴	۶۰/۵۵	۴	٪۲۷
		۲	۳۳۸	۶۳/۹۱	۴	٪۱۹
		۳	۵۷	۲۴/۳۳	۳	٪۴۴
DFG5	۲	۱	۷۱۶	۶۹/۴۲	۵	٪۱۶
		۲	۷۵۹	۷۲/۲۹	۵	٪۱۴
DFG6 (FEAL)	۵	۱	۱۹۳	۱۲/۲۲	۳	٪۵۸
		۲	۱۶۱	۲۲/۲۵	۳	٪۳۵
		۳	۱۷۷	۱۲/۸۲	۳	٪۳۷
		۴	۱۸۵	۱۱/۶۵	۳	٪۴۳
		۵	۱۶۰	۱۱/۵۰	۳	٪۳۹

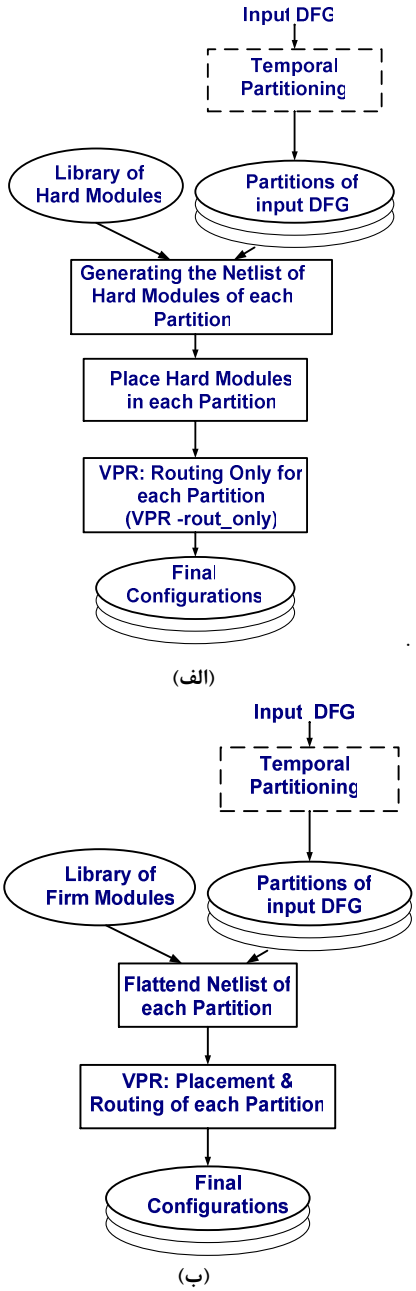
جدول ۴- بهبود حاصل از پیاده‌سازی بر اساس ماجول‌های سفت در مقابل ماجول‌های سخت

گراف جریان داده	اندازه آرایه	هزینه جایابی	پهنای کانال مسیریابی	حداکثر تأخیر مسیر بحرانی (نانو ثانیه)
DFG1	+٪۱۵/۷	+٪۵۰/۰	+٪۳۷/۵	-٪۶/۸
DFG2	+٪۲۰/۷	+٪۳۷/۴	+٪۳۷/۵	-٪۵/۴۱
DFG3	+٪۱۸/۳	+٪۴۷/۳	+٪۱۶/۶	-٪۶/۹
DFG4	+٪۴/۹	+٪۴۸/۹	+٪۴۴/۴	-٪۶/۴
DFG5	+٪۱۹/۶	+٪۴۵/۲	+٪۳۷/۵	-٪۱۰/۷
DFG6 (FEAL)	٪۰	+٪۱۹/۸۳	+٪۵۷/۱	-٪۳۷/۱۶

جدول۵- مقایسه زمان جایابی بر اساس ماجول‌های سفت در مقابل ماجول‌های سخت

گراف جریان داده	زمان جایابی برای ماجولهای سفت(ثانیه)	زمان جایابی برای ماجولهای سخت(ثانیه)	میزان تسریع عملیات جایابی
DFG1	۳/۵۳	۰/۲	۱۷/۶۵
DFG2	۲/۸۷	۰/۱۵	۱۹/۱۳
DFG3	۲/۷۳	۰/۱۶	۱۷/۰۶
DFG4	۶/۶۶	۰/۳۵	۱۹/۰۲
DFG5	۲۳/۲۵	۰/۳۵	۶۶/۴۲
DFG6 (FEAL)	۳/۷۶	۰/۵۶	۶/۷۱

ما الگوریتم افراز زمانی خود را با استفاده از روش افراز زمانی مبتنی بر پیمایش عمودی DFGها ارزیابی کردیم. آزمایش‌ها نشان می‌دهند که به کارگیری این الگوریتم بهبود نسبی و گاه قابل توجهی در حافظه مورد نیاز و اندازهٔ آرایه و تعداد پین‌های ورودی/خروجی دارد. برای DFGهای دوم و سوم هیچ بهبودی در مصرف حافظه مشاهده نشد چون در الگوریتم افراز زمانی، پارتیشن‌های حاصل برای هر دو روش یکسان بود. جدول ۱۰ نتایج آزمایش‌ها را نشان می‌دهد.



شکل ۱۱- مراحل طراحی بر اساس (الف) ماجول‌های سخت (ب) ماجول‌های سفت

۸- نتیجه گیری

در این مقاله، یک چارچوب کاری برای مراحل طراحی فیزیکی و افراز زمانی به صورت مجتمع برای تولید پیکربندی‌ها در سیستم‌های قابل باز پیکربندی ارائه شد. الگوریتم‌های افراز زمانی مطرح شده سعی داشتند تا با کشف شباهت بین پیکربندی‌های متوالی زمان کامپایل طرح و نیز زمان باز پیکربندی و در نتیجه زمان اجرای کاربرد را بهبود دهند. همچنین پیمایش عمودی نودهای DFG به جای پیمایش افقی آنها موجب می‌شود که سربار زیاد حافظهٔ مصرفی بین پیکربندی‌ها کاهش یابد. در این روش تعداد پین‌های ورودی/خروجی و مساحت سخت‌افزار مورد استفاده نیز بهبود می‌یابد.

دو روش مختلف برای فرآیند طراحی فیزیکی استفاده شد. در اولین روش، ماجول‌های سخت برای تولید پیکربندی‌ها بکار گرفته شد. در روش دوم، ماجول-های سفت به جای ماجول‌های سخت استفاده شدند که موجب کاهش مساحت سخت‌افزار، بهبود کیفیت جایابی، کاهش پهنای کانال مسیریابی و کاهش پراکندگی در سطح سخت‌افزار گردید. استفاده از ماجول‌های سخت برای جایابی-های online می‌تواند مناسب باشد، زیرا زمان جایابی را کاهش می دهد. برای ماجول‌های سفت، یک روش افزایشی تکراری برای مرحله‌ی جایابی تعریف و مورد استفاده قرار گرفت. این روش کیفیت پیکربندی خوبی را در زمان کمتری در مقایسه با الگوریتم غیرافزایشی تولید می‌کند. به علاوه، مجموع زمان اجرای کاربرد نیز کاهش پیدا می‌کند.

تشکر و قدردانی

این کار تحت حمایت مرکز تحقیقات مخابرات ایران انجام شده است.

مراجع

[1] M. Barr, *A Reconfigurable Computing Primer*, Miller Freeman Inc., 1998.

[2] K. Bazargan, and R. Kastner, and M. Sarrafzadeh, "Fast Template Placement for Reconfigurable Computing Systems," *IEEE Design and Test of Computers*, pp. 68-83, January-March 2000.

[3] K. Bazargan, and M. Orgenci, and M. Sarrafzadeh, "Integrating Scheduling and Physical Design into a Coherent Compilation Cycle for Reconfigurable Computing Architectures," *Proc. Of Design Automation Conference* , pp. 635-640, 2001.

[4] R. Kastner, and A. Kaplan, and M. Sarrafzadeh, *Synthesis Techniques and Optimizations for Reconfigurable Systems*, Kluwer-Academic Publishers, 2004.

[5] R. Enzler, The Current Status of Reconfigurable Computing, Technical report, Swiss Institute of Technology, July 1999.

[6] R. Tessier, and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey," *Journal of VLSI Signal Processing*, no. 28, pp. 7-27, 2001.

[7] R. Maestre, and F.J. Kurdahi, and N. Bagherzadeh, and H. Singh, and R. Hermida, and M. Fernandez, "Kernel Scheduling in Reconfigurable Computing, " *Proc. of Design, Automation and Test in Europe Conference,* pp.90-96, 1999.

[8] C. Bobda , Synthesis of Data Flow Graphs for Reconfigurable Systems Using Temporal Partitioning and Temporal Placement, Ph.D. Dissertation, Faculty of Computer Science, Electrical Engineering and Mathematics, University of Paderborn, 2003.

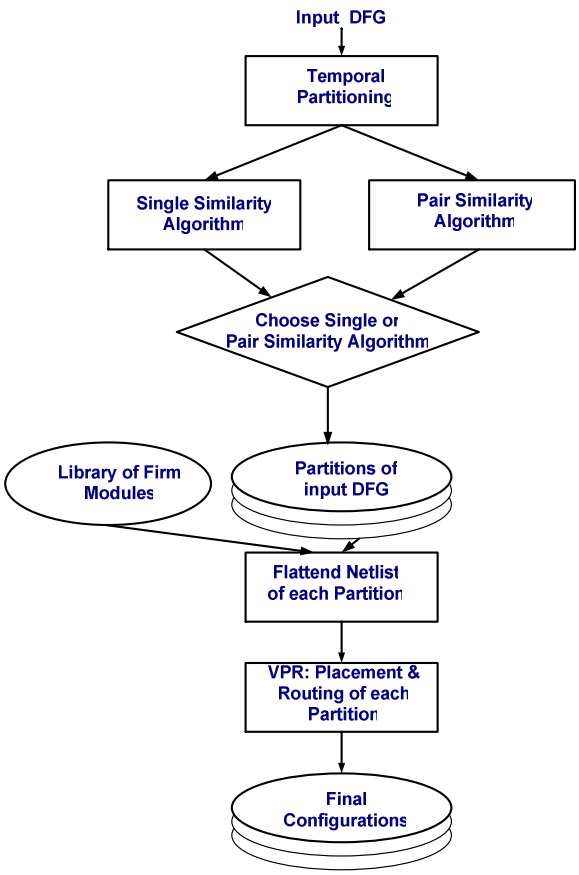
[9] K. Compton and S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171-210, 2002.

جدول ۶- مقایسه الگوریتم‌های افراز زمانی مبتنی بر شباهت نود منفرد و زوج نود

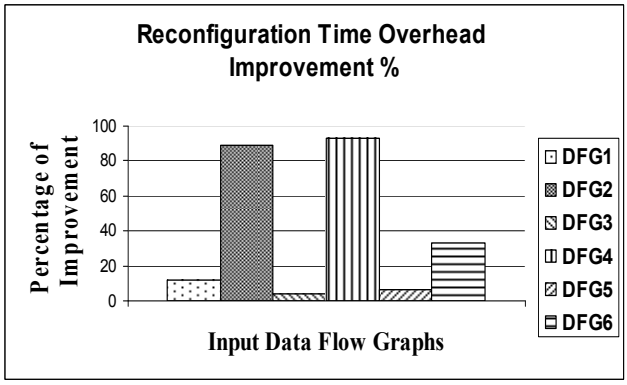
گراف جریان داده	الگوریتم شباهت نود منفرد		الگوریتم شباهت زوج نود	
	تعداد نودهای باشباهت منفرد (قبل از اجرای الگوریتم مبتنی بر شباهت)	تعداد نودهای باشباهت منفرد (بعد از اجرای الگوریتم مبتنی بر شباهت)	تعداد زوج نودهای مشابه (قبل از اجرای الگوریتم افزودن نود ساختگی)	تعداد زوج نودهای مشابه (بعد از اجرای الگوریتم افزودن نود ساختگی)
DFG1	۱	۲	۱	۲
DFG2	۳	۳	۰	۰
DFG3	۰	۰	۰	۰
DFG4	۴	۴	۰	۲
DFG5	۵	۵	۰	۲
DFG6 (FEAL)	۹	۱۱	۵	۷

جدول ۷- تأثیر الگوریتم‌های مبتی بر شباهت نود بر روی زمان کامپایل و افزایش سرعت بازپیکربندی

گراف جریان داده	الگوریتم شباهت نود منفرد		الگوریتم شباهت زوج نود		
	درصد بهبود زمان جایابی	میزان بهبود بازپیکربندی	درصد بهبود زمان جایابی	درصد بهبود زمان مسیریابی	میزان بهبود بازپیکربندی
DFG1	٪۱۰	۱/۰۳	٪۱۳/۷	٪۱۴	۱/۱۶
DFG2	٪۵/۸	۱/۰۲	٪۰/۰	٪۰/۰	۰
DFG3	٪۰/۰	۰/۰	٪۰/۰	٪۰/۰	۰
DFG4	٪۵۰	۱/۱۷	٪۴۸	٪۱۳/۵	۱/۳۱
DFG5	٪۶۴	۱/۲۴	٪۱۸	٪۷/۵	۱/۱۲
DFG6 (FEAL)	٪۳۷/۵	۱/۱۲	٪۵۸	٪۱۶/۶	۱/۳۲



شکل ۱۲- مراحل تولید پیکربندی‌ها با استفاده از الگوریتم‌های کشف شباهت نود منفرد و زوج نود



شکل ۱۳- بهبود زمان بازپیکربندی و کل زمان اجرای کاربرد با استفاده از چارچوب کاری ارائه شده

جدول ۸- نتایج حاصل از روش‌های افزایشی و غیر افزایشی

گراف جریان داده	غیر افزایشی			افزایشی		
	هزینهٔ جایابی	تاخیر مسیر بحرانی (نانوثانیه)	پهنای کانال مسیریابی	هزینهٔ جایابی	تاخیر مسیر بحرانی (نانوثانیه)	پهنای کانال مسیریابی
DFG1	۲/۶۴	۲۹/۳۴	۳	۲/۵۹	۳۶/۴۸	۳
DFG2	۱۶/۳۳	۶۴/۶۷	۵	۱۴/۵	۶۴/۶۷	۵
DFG3	۱/۰۵	۸/۴۴	۳	۰/۹۷	۷/۸۴	۲
DFG4	۱۶/۵۰	۲۴/۳۳	۴	۱۶/۲۰	۲۷/۳۴	۲
DFG5	۴۲/۵۸	۷۲/۲۹	۵	۴۱/۳۵	۷۰/۰۱	۵
DFG6 (FEAL)	۴/۵۴	۲۲/۲۵	۳	۴/۴۳	۱۳/۴۴	۳

جدول ۹- مقایسه روش‌های افزایشی و غیر افزایشی

گراف جریان داده	درصد بهبود هزینهٔ جایابی	درصد بهبود حداکثر سرعت	درصد کاهش کانال مسیریابی
DFG1	٪+۱۹	٪-۹/۵۷	٪۰/۰
DFG2	٪+۱۱/۲	٪۰/۰	٪۰/۰
DFG3	٪+۷/۶۱	٪+۷/۱۰	٪+۳۳
DFG4	٪+۱/۸۲	٪-۱۱/۰	٪+۵۰
DFG5	٪+۲/۸۹	٪+۳/۱۵	٪۰/۰
DFG6 (FEAL)	٪+۲/۴۲	٪+۳۹/۵۹	٪۰/۰

جدول ۱۰- میزان کاهش حافظه مصرفی حاصل از الگوریتم افراز زمانی مبتنی بر پیمایش عمودی

گراف جریان داده	درصد بهبود هاCLB در تعداد	درصد بهبود I/Oدر تعداد پینهای	درصد بهبود دراندازهٔ آرایه
DFG1	٪۸/۰	٪۳۵/۰	٪۱۲/۰
DFG2	٪۰/۰	٪۰/۰	٪۰/۰
DFG3	٪۰/۰	٪۰/۰	٪۰/۰
DFG4	٪۱۲/۰	٪۲۰/۰	٪۵۵/۰
DFG5	٪۴/۶۱	٪۲۶/۴	٪۰/۰
DFG6 (FEAL)	٪۵/۰	٪۸/۰	٪۱۶/۰

[10] F. Mehdipour, and M. Saheb Zamani, and M. Sedighi, "A New Iterative Design Flow for Static Compilation of Reconfigurable Computing Systems," *Proc. of the 10th International Symposium on Integrated Circuits, Devices and Systems,* Singapore, 2004.

[11] F. Mehdipour, and M. Saheb Zamani, and M. Sedighi, "An Integrated Temporal Partitioning and Physical Design Framework for Static Compilation of Reconfigurable Computing System," The International Journal of Microprocessors and Microsystems, 2005, Accepted for publishing.

[12] W.K. Mak, and E.F.Y. Young "Temporal Logic Replication for Dynamically Reconfigurable FPGA Partitioning", *IEEE Trans. of Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 7, pp. 952-959, 2003.

[13] X. Zhang, and K.Y. NG, "A Review of High-Level Synthesis for Dynamically Reconfigurable FPGAs," *The International Journal of Microprocessors and Microsystems*, vol. 24, pp. 199-221, 2000.

[14] N. Shirazi, and W. Luk, and P.Y.K. Cheung, "Automating Production of Run-Time Reconfigurable Designs," *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines,* IEEE Computer Society Press, pp. 147-156, 1998.

[15] J. Resano, and D. Verkest, and D. Mazos, and S. Varnalde, and F. Catthoor, "A Hybrid Design-Time/Run-Time Scheduling Flow to Minimize the Reconfiguration Overhead of FPGAs," *International Journal of Microprocessors and Microsystems,* vol. 28, 291-301, 2004.

[16] M. Karthikeya and P. Gajjala and D. Bhatia, "Temporal Partitioning and Scheduling Data Flow Graphs for Reconfigurable Computers," *IEEE Trans. on Computers*, vol. 48, no. 6, pp. 579-590, 1999.

[17] J. Spillane, and H. Owen, "Temporal Partitioning for Partially Reconfigurable Field Programmable Gate Arrays," *Proc. Of IPPS/SPDP Workshops*, pp. 37-42, 1998.

[18] S. Govindarajan and I. Ouaiss, and M. Kaul, and V. Srinivasan, and R. Vemuri, "An Effective Design Approach for Dynamically Reconfigurable Architectures," *Proc. Of IEEE Symposium on FPGAs for Custom Computing Machines*, pp.312-320, 1998.

[19] C. Tanougast, and Y. Berviller and P. Brunet, and S. Weber and H. Rabah, "Temporal Partitioning Methodology Optimizing FPGA Resources for Dynamically Reconfigurable Embedded Real-Time System," *The International Journal of Microprocessors and Microsystems*, vol. 27, pp.115-130, 2003.

[20] W. Luk and N. Shirazi and P.Y.K Cheung, "Modeling and Optimizing Run-Time Reconfiguration Systems," *Proc. of IEEE Symposium on FPGA's Custom Computing Machines,* IEEE Computer Society Press, pp.167-176, 1996.

[21] Z. Li, Configuration Management Techniques for Reconfigurable Computing, Ph.D. Dissertation, Dept. of ECE, Northwestern University, 2002.

[22] S. Ganesan, and R. Vemuri, "An Integrated Temporal Partitioning and Partial Reconfiguration Technique for Design Latency Improvement," *Proc. Of Design Automation and Test in Europe,* pp. 320, March 2000.

[23] R. Jayaraman, "Physical Design for FPGAs," *Proc. Of International Symposium on Physical Design,* pp. 214-221, 2001.

[24] N. Sherwani, *Algorithms for VLSI Physical Design Automation*, Kluwer-Academic Publishers, Third Edition, 1999.

[25] M. Hossain, and B. Thumma, and S. Ashtaputre, "A New Faster Algorithm for Iterative Placement Improvement," *Proc. Of Great Lakes Symposium on VLSI,* pp. 44-49, 1996.

[26] R. Tessier, Fast Place and Route Approaches for FPGAs, Ph.D. Dissertation, Massachussetts Institute of Technology, 1999.

[27] V. Betz, and J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs,* Kluwer Academic Publishers, 1999.

[28] V. Betz, VPR and T-VPack1 User's Manual (Version 4.30), *http://www.eecg.toronto.edu/~vaughn,* 2000.

[29] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[30] E.M. Sentovich, SIS: A System for Sequential Circuit Analysis, Tech. Report No.UCB/ERLM92/41, University of California, Berkeley, 1992.

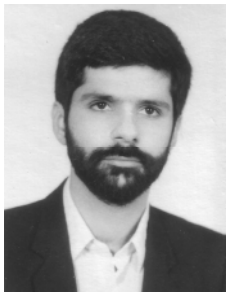
[31] J. Cong, and Y. Ding, "Flowmap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. on CAD,* pp. 1-12, 1994.

[32] A. Shimizu, and S. Miyaguchi,"Fast Data Encipherment Algorithm FEAL," *Trans. of IECE of Japan*, vol. J70-D, no. 7 , pp. 1413-1423, 1987.

[33] V. Betz, and J. Rose, "Directional Bias and Non-Uniformity in FPGA Global Routing Architectures, " *Proc. of the International Conference on Computer Aided Design*, pp.642-659, 1996.

1- Application Specific Integrated Circuit (ASIC)
2- Programmable Devices
3- Temporal Partitioning
4- Physical Design
5- Firm Module
6- Hard Module
7- Field Programmable Gate Array
8- Partially Programmable
9- Multi-Context
10- Placement
11- Routing
12- Global Routing

فرهاد مهدی‌پور مدارک کارشناسی و کارشناسی ارشد خود را در رشته مهندسی کامپیوتر به ترتیب از دانشگاه‌های صنعتی شریف و امیرکبیر در سال‌های ۱۳۷۵ و ۱۳۷۸ دریافت نمود. نامبرده اکنون دانشجوی دکترای در رشته مهندسی کامپیوتر در گرایش معماری سیستم‌های کامپیوتری در دانشگاه صنعتی امیرکبیر می باشد. زمینه‌های پژوهشی مورد علاقه ایشان سیستم‌های محاسبات قابل بازپیکربندی و طراحی سیستم‌های دیجیتال است.



مرتضی صاحب‌الزمانی مدرک کارشناسی خود را از دانشگاه صنعتی اصفهان در رشته مهندسی کامپیوتر در سال ۱۳۶۸ و مدارک کارشناسی ارشد و دکتری خود را در رشته مهندسی کامپیوتر از دانشگاه New South Wales استرالیا در سال‌های ۱۳۷۱ و ۱۳۷۵ دریافت نمود. ایشان از سال ۱۳۷۵ عضو هیأت علمی دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر هستند. زمینه‌های کاری ایشان، خودکارسازی طراحی مدارهای VLSI و سیستم‌های قابل بازپیکربندی است.

حمیدرضا احمدی فر عضو هیات علمی دانشگاه گیلان و فارغ التحصیل از دانشگاه صنعتی امیرکبیر در مقطع کارشناسی ارشد در رشته معماری سیستمهای کامپیوتری و فارغ‌التحصیل مقطع کارشناسی از دانشگاه شهید بهشتی هستند. زمینه های تخصصی ایشان زبانهای توصیف سخت افزاری، برنامه نویسی میکروکنترلر، PIC و AVR، برنامه نویسی تراشه های FPGA می‌باشد.

مهدی صدیقی در سال ۱۳۶۹ مدرک کارشناسی خود را در رشته مهندسی برق-سخت افزار از دانشگاه صنعتی شریف و در سال های ۷۳ و ۷۷ مدارک کارشناسی ارشد و دکترای خود را در رشته مهندسی برق و کامپیوتر از دانشگاه کلرادو در بولدر دریافت نمود. در سال‌های ۱۳۷۵ تا ۱۳۸۰ ایشان به عنوان مشاور و طراح ارشد سخت افزار شرکت های کامپیوتری Silicon Valley مشغول به کار بوده و هم اکنون استادیار دانشکده مهندسی کامپیوتر و فناوری اطلاعات دانشگاه صنعتی امیرکبیر می‌باشد. زمینه های پژوهشی مورد علاقه ایشان شامل طراحی VLSI، طراحی سخت افزار و مدارهای مجتمع برای کاربردهای ویژه می‌باشد.