سال سیزدهم، شماره‌های ۴۷ و ۴۸، بهار و تابستان ۱۴۰۰
صص: ۱۸۶_۱۹۷

دو فصلنامه علمی
فناوری اطلاعات و ارتباطات ایران

یک الگوریتم زمان‌بندی وظیفه چندهدفه بر اساس الگوریتم ژنتیک برای طراحی سیستم‌های نهفته

محدثه نیک سرشت* محسن راجی**

* کارشناسی ارشد، دانشکده مهندسی برق و کامپیوتر، دانشگاه شیراز

**عضو هیئت علمی، دانشکده برق و کامپیوتر، دانشگاه شیراز

تاریخ دریافت: ۱۳۹۹/۰۸/۲۳ تاریخ پذیرش: ۱۳۹۹/۱۲/۲۰

نوع مقاله: پژوهشی

چکیده

طراحان سیستم‌های نهفته با الزامات و اهداف متعددی در طراحی (مانند زمان اجرا، انرژی مصرفی و قابلیت اطمینان) مواجه هستند. ازآنجاکه در بیشتر مواقع، تلاش برای برآوردن یکی از این الزامات در تناقض با دستیابی به دیگر الزامات طراحی است، استفاده از رویکردهای چندهدفه در مراحل مختلف طراحی دستگاه‌های نهفته ازجمله زمان‌بندی وظایف امری اجتناب‌ناپذیر به نظر می‌رسد. در این مقاله، یک روش زمان‌بندی وظیفه ایستای چندهدفه برای طراحی دستگاه‌های نهفته ارائه‌شده است. در این روش، وظایف به‌صورت یک گراف مدل شده و با در نظر گرفتن یک زیرساخت سخت‌افزاری برای سیستم نهفته، روشی برای نگاشت و زمان‌بندی وظایف بر روی معماری سخت‌افزاری پیشنهاد می‌شود. در این روش زمان‌بندی، پارامترهای زمان اجرای وظایف، انرژی مصرفی و قابلیت اطمینان به‌عنوان اهداف بهینه‌سازی طی یک الگوریتم بهینه‌سازی ژنتیک بهینه می‌گردند. نتایج شبیه‌سازی‌ها نشان می‌دهد که روش پیشنهادی با در نظر گرفتن اهداف مختلف طراحی در مقایسه با روش‌های مشابه پیشین مانند EAG-TA، در زمان اجرای وظایف، انرژی مصرفی و قابلیت اطمینان به ترتیب ۲۱,۴، ۱۹,۲ و ۲۰ درصد بهبود داشته است. استفاده از یک راهبرد بهینه‌سازی چندهدفه این امکان را فراهم می‌کند که طی مرحله نگاشت و زمان‌بندی، گزینه‌های متعدد طراحی پیش روی طراح قرار گیرد تا بتواند بین پارامترهای مختلف طراحی سیستم (سخت‌افزاری/نرم‌افزاری) موازنه مدنظر خود را انجام دهد.

واژه‌های کلیدی: سیستم‌های نهفته، زمان‌بندی وظیفه، بهینه‌سازی چندهدفه، الگوریتم ژنتیک.

۱ مقدمه

سیستم‌های نهفته یا سیستم‌های تعبیه‌شده به‌طور گسترده‌ای در خدمات و محصولات صنعتی ، نظامی و تجاری استفاده می‌شود [۱]. این زمینه‌های مختلف کاربرد، منجر به ناهمگنی بیشتر و پیچیدگی روزافزون در سیستم‌های نهفته امروزی شده است و طراحی این سیستم‌ها را به یک چالش مهم مهندسی تبدیل کرده است [۲].

یکی از بخش‌های اصلی این چالش شامل مسئله نگاشت و زمان‌بندی کارها در طراحی سیستم‌های نهفته می‌باشد [۲]. در یک مسئله نگاشت و زمان‌بندی، فرآیند کاری که بر عهده سیستم نهفته است به‌صورت یکسری وظایف نرم‌افزاری مدل شده و بایستی با توجه به زیرساخت سخت‌افزاری موجود، ترتیب اجرای هرکدام از وظایف روی اجزای زیرساخت سخت‌افزاری تعیین گردد. با توجه به محدودیت‌های متعددی که سیستم‌های نهفته با آن روبه‌رو هستند، از قبیل زمان اجرای وظایف، توان مصرفی و صحت عملکرد آن‌ها

نویسنده مسئول: محسن راجی، mraji@shirazu.ac.ir

(قابلیت اطمینان)، مسئله زمان‌بندی بایستی با در نظر گرفتن هم‌زمان این پارامترها در قالب یک مسئله بهینه‌سازی چندهدفه حل شود [۳]-[۶]. به این ترتیب، این امکان برای طراح سیستمی فراهم می‌شود که طی مرحله نگاشت و زمان‌بندی، بتواند از بین گزینه‌های متعدد طراحی و بین پارامترهای مختلف طراحی سیستم (سخت‌افزاری/نرم‌افزاری) موازنه مدنظر خود را انجام دهد.

مسئله زمان‌بندی وظایف یک مسئله NP-سخت است که در [۷] بررسی‌شده است، به این معنی که با افزایش ابعاد مسئله، رسیدن به بهترین جواب مسئله در یک‌زمان معقول امکان‌پذیر نیست. به همین دلیل، از الگوریتم‌های اکتشافی و فرا اکتشافی برای رسیدن به جواب‌هایی که تا حد ممکن به بهترین جواب‌ها نزدیک باشند، استفاده می‌شود. یکی از روش‌های محبوب فرا اکتشافی که قابلیت حل مسائل بهینه‌سازی چندهدفه را نیز دارد، الگوریتم ژنتیک است. این الگوریتم، به دلیل کارآمد بودن و درعین‌حال، سادگی پیاده‌سازی به‌طور گسترده‌ای برای حل مسائل بهینه‌سازی مختلف مورداستفاده قرارگرفته است که در [۸] بررسی‌شده است. الگوریتم ژنتیک با الهام از سیر تکامل انسان‌ها، قادر است پس از مدل شدن مسئله به‌صورت کروموزم[1] ها، با اعمال عملگرهای ژنتیکی بر روی آن‌ها، به حل مسئله بهینه‌سازی نائل گردد که در [۹] بررسی‌شده است. نیاز به تأکید است که ابزار ما در مرحله طراحی سیستم‌های نهفته کار می‌کند. در این سطح هیچ سیستم‌عاملی وجود ندارد، فقط مجموعه‌ای از وظایف مورد انتظار و بستر سخت‌افزاری موجود است. این امر به طراح کمک می‌کند تا بفهمد مجموعه‌ای از کارها را می‌توان بر روی سخت‌افزار موردتقاضا در شاخص‌های موردنظر زمان‌بندی کرد یا خیر و در صورت لزوم تغییراتی را در بستر سخت‌افزاری یا مجموعه وظایف ایجاد کند. ابزارهای ارزیابی در مرحله اول طراحی با تجزیه‌وتحلیل دقیق سیستم، طراحان را قادر می‌سازد گزینه‌های مختلف طراحی را بررسی و بر اساس شاخص‌های طراحی انتخاب کنند. تصمیمات طراحی که در مراحل اولیه فرآیند طراحی گرفته می‌شوند برای جلوگیری از تغییرات بالقوه پرهزینه در مراحل پیشرفته‌تر فرآیند بسیار مهم هستند. درنتیجه، طراح باید در مراحل اولیه طراحی، تجزیه‌وتحلیل و ارزیابی لازم را انجام دهد تا اطمینان حاصل کند که سیستم تعبیه‌شده حاصل عملکرد مطلوبی را در مورد پارامترهای طراحی موردنظر برآورده می‌کند.

۲ کارهای پیشین

کارهای متعددی از الگوریتم ژنتیک به‌منظور حل مسئله نگاشت و زمان‌بندی وظایف در حین طراحی سخت‌افزاری/نرم‌افزاری سیستم‌های نهفته بهره برده‌اند. می‌توان این روش‌های متعدد را بر اساس نوع گراف ورودی سیستم به دو دسته ایستا و پویا تقسیم کرد. در نوع استاتیک گراف ورودی در لحظه ورود دریافت می‌شود و زمان‌بندی بر روی آن انجام می‌شود. درحالی‌که در حالت پویا گراف ورودی در زمان اجرا امکان دارد تغییر کند یا در هرلحظه وظیفه‌ای جدید به سیستم داده شود. کارهای ارائه‌شده در [۱۰] و [۱۱] بر روی گراف‌های ورودی پویا تمرکز کرده است. اما تمرکز این مقاله بر روی گراف‌های ورودی ایستا می‌باشد.

زمان‌بندی وظایف از نوع استاتیک خود به دو زیرشاخه اکتشافی و فراکتشافی تقسیم می‌شوند. از طرفی الگوریتم‌های اکتشافی شامل سه زیرشاخه لیست محور، خوشه‌ای و تکثیری هستند [۱۲].

در الگوریتم‌های لیست محور، به هر یک از وظایف لیست اولویتی داده می‌شود و وظایف بر اساس تقدم اولویت شروع به اجرا می‌کنند. به‌طورکلی وظیفه‌ای با اولویت بالاتر سریع‌تر به یک پردازنده منصوب می‌شود. برای مثال در سیستم‌های ناهمگن وظیفه‌ای با اولویت بالاتر به پردازشگر قدرتمندتر تخصیص داده می‌شود تا زمان اجرا آن به حداقل برسد. الگوریتم‌های HEFT و CPOP [۱۲] از مهم‌ترین نمونه‌های الگوریتم‌های لیست محور می‌باشند. در روش‌های تکثیری و خوشه‌ای با سپردن متعدد و مکرر وظایف به پردازنده‌های متفاوت زمان اجرا به حداقل رسانده می‌شود. در این روش‌ها با اجرای وظایف تکراری بر روی بیش از یک پردازنده و جلوگیری از انتقال نتیجه از یک پردازنده به پردازنده دیگر سربار زمانی ارتباط بین پردازنده‌ها را کاهش می‌دهیم. در سیستم‌های موازی و مجزا این روش راه‌حل مناسبی است تا تأخیر ارتباطی گراف را به حداقل برسانیم، زیرا وظایفی که با یکدیگر ارتباط زیادی دارند در یک دسته قرار می‌گیرند و به یک پردازنده محول می‌شوند. به‌طورکلی الگوریتم‌های اکتشافی قادر نیستند برای طیف وسیعی از مسائل به جواب پایدار دست یابند به‌خصوص زمانی که پیچیدگی‌های مربوط به زمان‌بندی وظایف افزایش پیدا می‌کند. از طرف دیگر، الگوریتم‌های اکتشافی هزینه بالای محاسباتی دارند و نسبت به الگوریتم‌های فرا اکتشافی دارای کارایی کمتری می‌باشند.

الگوریتم‌های فرا اکتشافی معمولاً به یک‌راه حل کلی برای حل مسائل بهینه‌سازی دست پیدا می‌کنند. زمانی که یک الگوریتم اکتشافی می‌باشد یعنی کاملاً بر اساس آزمون و خطا پیاده‌سازی شده است اما الگوریتم‌های فراشناختی می‌توانند بر اساس استراتژی‌های پایدار پیاده شوند که الگوریتم‌های اکتشافی را به سمت اکتشافات هدایت‌شده تغییر می‌دهند. به‌این‌ترتیب می‌توانند به نوآوری‌هایی تازه دست یابند. در حل مسائل ما به دنبال جواب می‌باشیم که برای تمام زیرمجموعه‌های مسئله بهینه باشد و نه‌تنها برای یک زیرمجموعه خاص از کل فضای جست‌وجو، از طرفی پیدا کردن پاسخ بهینه سراسری در اکثر مسائل حقیقی بسیار سخت می‌باشد و درنتیجه رسیدن به پاسخ‌های به‌اندازه مناسب رضایت‌بخش قابل‌قبول است. از الگوریتم‌های فراشناختی برای دست‌یابی به این دسته از پاسخ‌ها استفاده می‌شود. الگوریتم‌های فرا اکتشافی بر اساس سه

معیار سادگی، قابلیت انعطاف و گستردگی جستجو در فضای حل مسئله استوار می‌باشند. اکثر الگوریتم‌های فرا اکتشافی ساده و به‌راحتی قابل پیاده‌سازی می‌باشند و پیچیدگی کمی دارند، این الگوریتم‌ها انعطاف‌پذیر بوده و به‌راحتی می‌تواند طیف وسیعی از مسائل بهینه‌سازی را حل کنند حتی مسائلی که با استفاده از الگوریتم‌های سنتی قابل پوشش دادن نیست.

درزمینه گراف‌های ورودی ایستا اولین بار در [۱۳]، یک الگوریتم ژنتیک چندهدفه برای ترکیب سخت‌افزاری/نرم‌افزاری سیستم‌های نهفته توزیع‌شده ارائه‌شده است. در این کار، تنها پارامترهای توان مصرفی و زمان اجرای وظایف در نظر گرفته شده است. همچنین وابستگی وظایف در گراف وظایف در نظر گرفته نشده است. در [۱۴]، [۱۵] و [۱۶] رویکردی برای نگاشت و زمان‌بندی سیستم نهفته مبتنی بر الگوریتم ژنتیک ارائه‌شده است. در این کار نیز پارامتر قابلیت اطمینان لحاظ نشده است. در [۱۷]، چارچوبی یکپارچه برای ترکیب سیستم‌های نهفته در سطح سیستمی ارائه‌شده است. در این چارچوب، ضمن استفاده از الگوریتم ژنتیک به بهینه‌سازی چندهدفه پارامترهای نگاشت پذیری و ویژگی‌های مختلف معماری می‌پردازد. هدف این کار مشخصاً ارائه یک بن سازه یکپارچه برای ترکیب سطح سیستمی بوده و به جستجوی فضای طراحی با توجه به پارامترهایی مانند توان مصرفی و قابلیت اطمینان نپرداخته است. همچنین در [۱۸]، [۱۹] و [۲۰] که نسخه‌ای تکمیل‌شده از کار [۱۷] می‌باشد، از الگوریتم ژنتیک برای بهینه‌سازی زمان اجرای وظایف در حین زمان‌بندی برای سیستم‌های نهفته استفاده شده است. از نقطه‌ضعف‌های کارهای فوق می‌توان به نادیده گرفتن دیگر پارامترهای مهم سیستم‌های نهفته ازجمله توان مصرفی و قابلیت اطمینان اشاره کرد. در [۲۱] ، برای اولین بار با رویکرد جست جو در فضای طراحی و ارائه مجموعه جواب‌ها برای طرح با الهام گرفتن از کارهای پیشین قابلیت اطمینان در نظر گرفته‌شده است. اما از نکات منفی این روش می‌توان به نادیده گرفته شدن انرژی مصرفی و زمان اجرا اشاره کرد. همچنین شبیه‌سازی قابلیت اطمینان با فرمول‌های بسیار پیچیده انجام‌شده است که قدرت محاسبات را به‌شدت کاهش می‌دهد و علاوه بر این تنها به مسئله قابلیت اطمینان مختص به پارامتر سالخوردگی در ترانزیستورها پرداخته‌شده است. در کارهای اخیر صورت گرفته در این حوزه در سطح طراحی مقاله‌هایی مانند [۲۲] ، [۲۳] و [۲۴] وجود دارند که یک محیط مدل‌سازی و شبیه‌سازی را برای کاوش فضای طراحی سیستم‌های نهفته ناهمگن ایجاد کرده‌اند. بااین‌حال، این کارها بر روی مسئله نگاشت متمرکز بوده و از پرداختن به مشکل زمان‌بندی چشم‌پوشی کرده‌اند جمع‌بندی موارد گفته‌شده را می‌توانید در جدول ۱ مشاهده کنید.

به‌طور خلاصه کارهای پیشین دارای محدودیت‌های جدی‌ای هستند؛ ازجمله این‌که پارامترهای زمان اجرای وظایف، توان/انرژی مصرفی و قابلیت اطمینان به‌صورت هم‌زمان در حین حل مسئله نگاشت و زمان‌بندی وظایف در سیستم‌های نهفته در نظر گرفته نشده‌اند و یا این‌که تنها به مسئله نگاشت وظایف پرداخته‌شده است.

جدول ۱ . جمع‌بندی از کارهای پیشین

نقاط ضعف	نقاط مثبت	مرجع مربوط به الگوریتم
فرمول‌های شبیه‌سازی بسیار ساده، عدم پشتیبانی از گراف های ورودی پیچیده و دارای وابستگی، عدم در نظر گرفتن قابلیت اطمینان	در نظر گرفتن انرژی مصرفی و زمان اجرا	[۱۳], [۱۴], [۱۵], [۱۶]
عدم در نظر گرفتن دو پارامتر مهم قابلیت اطمینان و انرژی مصرفی	ترکیب نگاشت سخت‌افزاری با گراف وظایف ورودی سیستم، جست‌وجوی فضای طراحی برای اولین بار و قرار دادن نتایج مختلف شبیه‌سازی در اختیار طراح	[۱۷], [۱۸], [۱۹], [۲۰]
غفلت از سایر پارامترهای مهم طراحی مانند انرژی مصرفی و زمان اجرا، محدود بودن مدلسازی قابلیت اطمینان به سالخوردگی ترانزیستورها	مدلسازی قابلیت اطمینان در مسئله زمانبندی سیستم های نهفته برای اولین بار	[۲۱]
محدود بودن به مسئله نگاشت وظایف و عدم پرداختن به مسئله زمان‌بندی وظایف	در نظر گرفتن هر سه پارامتر مهم انرژی مصرفی، قابلیت اطمینان و زمان اجرا در سطح طراحی و حل مسئله نگاشت وظایف بر روی سخت افزار	[۲۲], [۲۳], [۲۴]

در این مقاله، یک روش زمان‌بندی وظیفه ایستای چندهدفه مبتنی بر الگوریتم ژنتیک برای طراحی سیستم‌های نهفته ارائه‌شده است. در این روش، وظایف به‌صورت یک گراف وظیفه مدل شده و با توجه به معماری سخت‌افزاری موجود، روشی برای نگاشت و

زمان‌بندی وظایف بر روی معماری سخت‌افزاری پیشنهاد می‌شود. به این منظور، فرآیند نگاشت و زمان‌بندی به‌صورت کروموزوم‌هایی مدل شده و هر کروموزوم با استفاده از تابع ارزیابی‌های مختلف برای پارامترهای زمان اجرای وظایف، توان مصرفی و قابلیت اطمینان ارزیابی می‌شوند. برای یک بهینه‌سازی چندهدفه، از انتخاب راه‌حل‌های پارتو استفاده‌شده و هر بار با توجه به نتایج ارزیابی پارامترهای سه‌گانه جمعیت جدید (زمان‌بندی‌های جدید) انتخاب می‌شوند. نتایج شبیه‌سازی‌ها نشان می‌دهد که روش پیشنهادی با در نظر گرفتن اهداف مختلف طراحی در مقایسه با روش‌های مشابه پیشین مانند EAG-TA، در زمان اجرای وظایف، انرژی مصرفی و قابلیت اطمینان به ترتیب۲۱,۴، ۱۹,۴ و۲۰ درصد بهبود داشته است. استفاده از یک راهبرد بهینه‌سازی چندهدفه این امکان را فراهم می‌کند که طی مرحله نگاشت و زمان‌بندی، گزینه‌های متعدد طراحی پیش روی طراح قرار گیرد تا بتواند بین پارامترهای مختلف طراحی سیستم (سخت‌افزاری/نرم‌افزاری) موازنه مدنظر خود را انجام دهد.

به‌طور خلاصه، نوآوری‌های مقاله به شرح زیر است:

۱. مدل‌سازی مسئله زمان‌بندی وظایف در سیستم‌های نهفته به‌صورت یک مسئله بهینه‌سازی چندهدفه در قالب الگوریتم ژنتیک

الف) توسعه نحوه نمایش کروموزوم‌ها برای حل مسئله زمان‌بندی

ب) توسعه عملگرهای ژنتیکی برای نحوه نمایش جدید کروموزوم‌ها

۲. لحاظ قابلیت اطمینان علاوه بر پارامترهای زمان اجرا، توان مصرفی به‌طور هم‌زمان در حل مسئله زمان‌بندی وظیفه سیستم‌های نهفته.

ادامه مقاله به این صورت زیر سازمان‌دهی شده‌اند: بخش ۳ به متدلوژی پیشنهادی طراحی می‌پردازد. طی این بخش، مدل استفاده‌شده برای وظایف و معماری سخت‌افزاری مدنظر بررسی می‌شود. همچنین الگوریتم ژنتیک توسعه داده‌شده برای حل مسئله نگاشت و زمان‌بندی وظایف در سیستم‌های نهفته ارائه می‌شود. در بخش ۴ نتایج تجربی ارائه می‌شود و درنهایت، مقاله در بخش ۵ جمع‌بندی می‌شود.

۳ روش پیشنهادی طراحی چندهدفه مبتنی بر الگوریتم ژنتیک

در این قسمت به نحوه طراحی چندهدفه طراحی مبتنی بر الگوریتم‌های ژنتیک پرداخته شده است. در قسمت اول مدل‌سازی وظایف و معماری سخت‌افزاری ارائه شده و در قسمت دوم، به رویه جست‌وجوی فضای طراحی با استفاده از الگوریتم‌های ژنتیک می‌پردازیم.

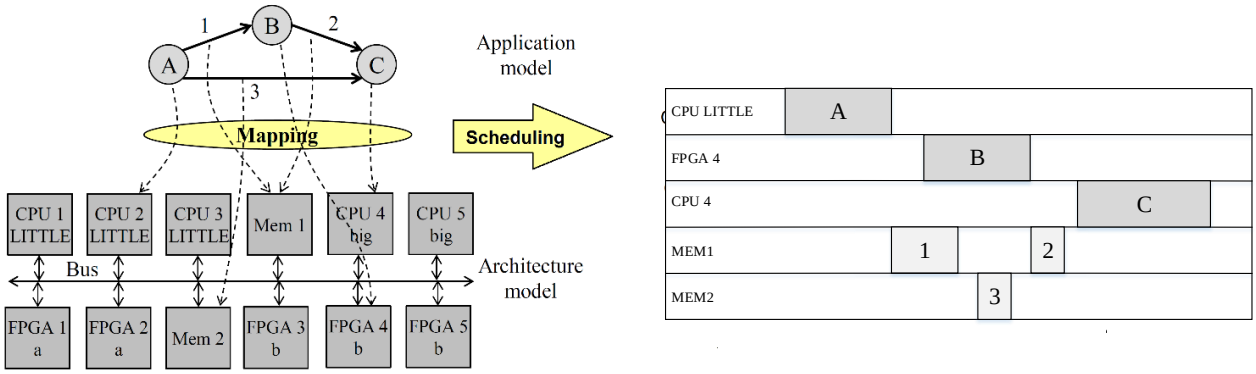
۳,۱ مدل‌سازی وظایف و معماری سخت‌افزاری

برای مدل کردن وظایف، از مدل معرفی‌شده در [۱۷] مبتنی بر شبکه فرآیند کاهن (KPN) استفاده می‌شود که یک مدل بسیار مشهور محاسباتی برای مدل کردن سیستم‌های نهفته است. KPN روشی برای نمایش وظیفه است که در آن، وظایف به صورت گراف جهت‌دار نشان داده می شود. KPN را به شکل گراف $G_{AP}(V_{AP}, E_{AP})$ نشان می‌دهند که در آن برای هر رأس $i \in \{1, \dots, |V_{AP}|\}$ یک وظیفه یا پردازه[۲] از گراف را نشان می‌دهد. برای هر رأس V_i مجموعه وظایفی که به یکدیگر مرتبط می‌باشند به شکل $B_i = \{e_j \in E_{AP}\}$ نشان داده می‌شود. زمانی که رأس V_i برای اجرا به یک مؤلفه سخت‌افزاری تخصیص داده می شود، ht_i نشان‌دهنده مدت‌زمان اجرا بر روی این سخت‌افزار می‌باشد.

زمانی که وظیفه‌ای امکان اجرا بر روی هسته‌های[۳] متعددی را داشته باشد. زمان اجرای آن به‌صورت مجموعه $ht_i = \{ht_{i1}, ht_{i2}, \dots, ht_{iU}\}$ نشان داده می شود که نشان دهنده تعداد سخت‌افزاری است که وظایف روی آن ها اجرا می شود.

زمانی که رأس به یک مؤلفه نرم‌افزاری برای زمان‌بندی تخصیص داده می‌شود، st_i نشان‌دهنده زمان اجرا بر روی آن مؤلفه نرم‌افزاری می باشد. زمانی که آن وظیفه بر روی مؤلفه های نرم‌افزاری متعدد قابلیت اجرا داشته باشد، زمان اجرای آن به‌صورت مجموعه $St_i = \{st_{i1}, st_{i2}, \dots, st_{iV}\}$ نشان داده می شود که نشان‌دهنده تعداد مؤلفه‌های نرم‌افزاری می‌با شد که این وظیفه می‌تواند بر روی آن اجرا شود.

هر یال e_j، $j \in \{1, \dots, |E_{AP}|\}$ نشان‌دهنده رابطه بین دو وظیفه از گراف G_{AP} می‌باشد. اگر این مسیر ارتباطی به یک حافظه تخصیص داده شود، mt_j نشان‌دهنده زمان دسترسی به حامی می‌باشد. اگر در زمان‌بندی بتوان آن را به حافظه‌های متعددی تخصیص داد، زمان دسترسی به شکل مجموعه $mt_j = \{mt_{j1}, mt_{j2}, \dots, mt_{jW}\}$ نشان داده می شود. این مجموعه نشان‌دهنده مجموعه وظیفه‌ای است که می‌توان آن‌ها را در زمان‌بندی تخصیص داد.



شکل ۱ . مسئله نگاشت و زمان‌بندی در طراحی سیستم‌های نهفته

همچنین مدل معماری به صورت گراف $G_{AR}(V_{AR}, E_{AR})$ نشان داده می‌شود که در آن V_{AR} و E_{AR} به ترتیب نشان‌دهنده مؤلفه‌های معماری و مسیر ارتباطی بین آن می‌باشد. مجموعه مؤلفه‌های معماری شامل دو مجموعه مستقل یعنی مجموعه پردازشگرها (P) که شامل مجموعه مؤلفه‌های سخت‌افزاری، نرم‌افزاری و همچنین مجموعه حافظه ها می باشد، $V_{AR} = P\ U\ M$. تأخیر بین مسیرهای ارتباطی به‌صورت lt_{pq} نشان داده‌شده است که در آن $p,q \in \{1,\dots,|E_{AR}|\}$ توان مصرفی در طول اجرا برای پردازنده p به شکل w_{pe} و w_{me} برای حافظه m و مسیر ارتباطی w_{le} در نظر گرفته شده است. در این مقاله، فرض بر این است که بن سازه معماری موجود و مشخص است چراکه هدف اصلی، حل مسئله تعیین بهترین معماری نیست بلکه روش پیشنهادی برای حل مسئله نگاشت زمان‌بندی در طراحی سیستم‌های نهفته است.

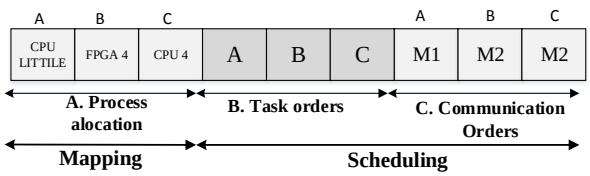
مسئله نگاشت و زمان‌بندی، مسئله یافتن بهترین تخصیص وظایف و ارتباطات بین آن ها بر روی یک بن سازه معماری سخت‌افزاری است. نمونه‌ای از نگاشت و زمان‌بندی وظایف بر بن سازه سخت‌افزاری در شکل (۱) آورده شده است. همان‌طور که در شکل مشاهده می شود، هر رأس یا یال از گراف که به ترتیب نماینده یک وظیفه یا ارتباط بین دو وظیفه است برای اجرا بر روی یک پردازشگر از بن سازه معماری سخت‌افزاری به آن تخصیص داده‌شده و ضمن لحاظ زمان اجرا وظیفه یا ارتباط مربوطه و همچنین ترتیب اجرای وظایف با توجه به گراف وظایف، زمان‌بندی اجرای آن‌ها نیز مشخص می‌گردد. در این مقاله برای نزدیک شدن به بهترین نگاشت و زمان‌بندی با توجه به اهداف مهم سیستم‌های نهفته اعم از بهترین زمان اجرا، توان مصرفی، قابلیت اطمینان از حل یک الگوریتم بهینه‌سازی چندهدفه مبتنی بر الگوریتم ژنتیک پیشنهاد می‌شود.

۳،۲ جستجوی فضای طراحی با استفاده از الگوریتم ژنتیک

به‌منظور جستجوی فضای طراحی برای یافتن بهترین نگاشت و زمان‌بندی وظایف از الگوریتم ژنتیک استفاده می‌شود. به علت پیچیدگی‌ها و محدودیت‌هایی که در طراحی سیستم‌های نهفته وجود دارد جستجوی فضای طراحی یک فرآیند چالش‌برانگیز است چراکه این کار، یک مسئله بهینه سازی چندهدفه است که این اهداف معمولاً در تضاد با یکدیگر می‌باشند. در ادامه، نحوه مدل سازی مسئله زمان‌بندی به کمک الگوریتم ژنتیک ارائه خواهد شد.

۳،۲،۱ ساختار کروموزوم‌ها

نگاشت و زمان‌بندی وظایف بر روی پردازنده‌ها و لبه‌های ارتباطی در هر کروموزوم کدگذاری می‌شود. فرض کنید یک گراف وظایف (G) با مجموعه وظایف (N) و گره‌های ارتباطی (M) و معماری سیستم با (P) پردازنده داشته باشیم، در این شرایط هر کروموزوم با مجموعه ژن‌های ۲M + N توصیف می‌شود. ساختار یک کروموزوم در شکل (۲) نشان داده‌شده است. هر کروموزوم از دو بخش اصلی تشکیل‌شده است:



شکل ۲ . نمایش کروموزوم‌ها

نگاشت: در این بخش، پردازنده منتخب برای اجرا بر روی هر وظیفه مشخص می‌شود. در شکل (۲)، N ژن اول نشان داده‌شده مربوط به نگاشت N وظیفه بر روی P پردازنده موجود است. برای مثال، وظیفه B بر روی $FPGA$ ۴ نگاشت شده است. ترتیب پر

شدن پردازنده‌ها بر روی وظایف بر اساس اولویت وظایف در بخش‌بندی است.

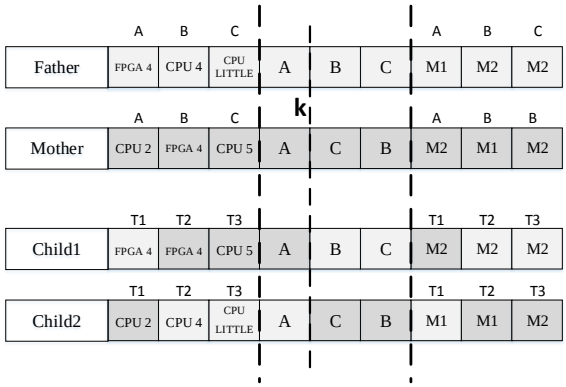
زمان‌بندی: $N + M$ ژن باقیمانده برای زمان‌بندی در نظر گرفته شده است. در قسمت اول(N ژن) ترتیب اجرای کارها بر روی پردازنده‌های منتخب شده را نشان می‌دهد. قسمت دوم(M ژن) ترتیب انجام انتقال داده از روی مسیرهای ارتباطی را نشان می‌دهد. در قسمت اول، وظایف را بر اساس ترتیب اختصاص داده شده درروش بخش‌بندی قرار می‌دهیم(حالت اولیه) و سپس به شکل تصادفی ترتیب قرارگیری وظایف هر بخش را تغییر می‌دهیم.

۳،۲،۲ عملگرهای ژنتیک

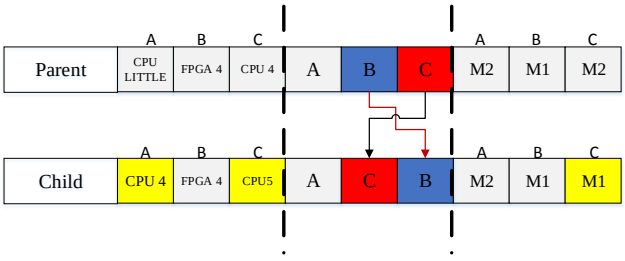
در الگوریتم ژنتیک از عملگرهای ژنتیک برای تولید و حفظ تنوع کروموزوم‌ها از یک نسل به نسل دیگر جمعیت استفاده می‌شود. در این مقاله از دو عملگر اصلی ژنتیک، جهش، حفظ تنوع بین اعضا کروموزوم‌های هر جمعیت، و تقاطع، کمک به همگرایی جمعیت به سمت پاسخ‌های بهتر، استفاده می‌شود. در ادامه، به بررسی دقیق این عملگرهای ژنتیک که در ابزار شبیه‌سازی پیشنهادی استفاده‌شده‌اند، می‌پردازیم.

عملگر برش[۴]: یکی از عملگرهای ژنتیک است، از این عملگر جهت ترکیب اطلاعات دو والدین[۵] و تولید فرزند جدید استفاده می‌شود. در پیاده‌سازی این الگوریتم در ابزار شبیه‌سازی پیشنهادی ابتدا یک از والدین به شکل تصادفی انتخاب می‌شوند و فن تقاطع محلی که در مقاله [۱۸] ارائه‌شده است بر روی آن اعمال می‌شود. طبقه‌بندی بین دسته کروموزوم به این شکل است که الگوریتم NSGAII [۲۶] بر روی جمعیت اعمال می‌شود و بر اساس غالب بودن کروموزوم‌ها، در دسته‌های مجزا تفکیک می‌شوند. پس از انتخاب شدن والد اول به شکل تصادفی، والد دوم به شکل تصادفی از دسته والد اول انتخاب می‌شود. برای مثال اگر والد اول در دسته یک قرارگرفته باشد، والد دوم نیز از دسته۱ انتخاب می‌شود. پس از انتخاب شدن والدین، دو فرزند با احتمال P_C تولید می‌شوند. شکل (۳)، نشان‌دهنده عملیات برش می‌باشد.

عملگر جهش[۶]: عملگر جهش یا جهش یک عملگر ژنتیک است که برای حفظ تنوع ژنتیک از یک نسل کروموزوم‌ها به نسل بعد استفاده می‌شود. این تغییرات نباید بر روی اولویت وظایف تأثیر بگذارد. در ابزار شبیه‌سازی پیشنهادی، برای پیاده‌سازی این عملگر یک والد به شکل تصادفی از جمعیت فعلی انتخاب می‌شود و دو فرزند از والد انتخاب‌شده تولید می‌شوند.. این تغییرات نباید گراف وابستگی وظایف را از بین ببرد. ازاین‌رو تغییر در کروموزوم با استفاده از بخش‌بندی صورت می‌گیرد. شکل (۴) نشان‌دهنده عملیات جهش است.



شکل ۳. عملیات برش.



شکل ۴. عملیات جهش.

عملگر انتخاب: انتخاب جمعیت بر اساس الگوریتم NSGAII [۲۶] است.

۳،۳ توابع هدف

در ابزار پیشنهادی، سه شاخص طراحی، زمان اجرا، انرژی مصرفی و قابلیت اطمینان برای رسیدن به پاسخ‌های بهینه در نظر گرفته‌شده‌اند. توابع هدف نسبت داده‌شده به هر یک از این شاخص‌های طراحی برای ارزیابی پاسخ‌های به‌دست‌آمده (زمان‌بندی‌های ارائه‌شده توسط الگوریتم) استفاده می‌شود. این مقادیر پس مشخص شدن ساختار کروموزوم‌ها و تخصیص پردازنده به وظایف محاسبه می‌شوند.

۳،۳،۱ هدف اول: زمان اجرا

اولین تابع هدف حداقل کردن زمان اجرا یا زمان پردازش بر روی مسیر بحرانی است. این مسیر شامل مجموعه‌ای از کل مسیرهای اجرا است. با توجه به نشان‌گذاری‌هایی که پیش‌ازاین انجام شده است، تابع هدف مربوط به زمان اجرای وظایف پس از هر بار نگاشت و زمان‌بندی از عبارت زیر به دست می‌آید [۲۲]:

$$\min\left\{\max path\left\{\sum_{i\in V_{AP},i\in Path} ht_{iu}\,x_{iu} + \sum_{i\in V_{AP},i\in Path} st_{iv}\,x_{iv} + \sum_{j\in E_{AP},j\in Path} [lt_{kl} + (mt_{jw} + lt_{mn})\,x_{jw}]x_j\right\}\right\} \quad (۱)$$

اولین قسمت معادله نشان‌دهنده تأثیر پردازنده‌های سخت‌افزاری بر روی زمان اجرای وظایف در طولانی‌ترین توالی اجرای وظایف (که تحت عنوان مسیر بحرانی تعریف می‌شود) است. قسمت دوم معادله نشان‌دهنده تأثیر مسیرهای برنامه کاری بر روی مدل کردن زمان اجرا است. آخرین قسمت نیز نشان‌دهنده تأثیر مسیرهای ارتباطی بر روی زمان اجرا است درصورتی‌که مسیر ارتباطی j به مؤلفه معماری متصل شده باشد. در اینجا mt_{jw} نشان‌دهنده زمان دسترسی به حافظه برای $w \in \{1, \dots, M\}$، از طرفی lt_{kl} نشان‌دهنده تأخیر مسیر هسته معماری k و $l \in \{1, \dots, |V_{AR}|\}$ و همچنین lt_{mn} نشان‌دهنده تأخیر بین هسته معماری m و n و همچنین با $m, n \in \{1, \dots, |V_{AR}|\}$ می باشد.

متغیرهای x_{iu}، x_{iv} و x_{jw} و x_j متغیرهای تصمیم‌گیری هستند. این متغیرها نشان می‌دهند برای مثال در هر مرحله، وظیفه i به سخت‌افزار u و یا نرم‌افزار v، مسیر ارتباطی درون‌هسته متصل شده است (در این صورت $x_j = 0$ در نظر گرفت می‌شود) به‌صورت متغیرهای تصمیم‌گیری نشان می‌دهد هر وظیفه برای اجرا بر روی کدام‌یک از هسته‌ها تخصیص داده‌شده است. مقدار این متغیرهای تصمیم‌گیری بر روی زمان‌بندی‌های مختلف می‌تواند مختلف باشد، این مقادیر در طول جستجوی فضای حل مسئله بر اساس الگوریتم ژنتیک تولیدشده‌اند.

۳،۳،۲ هدف دوم: توان مصرفی

دومین تابع هدف، حداقل کردن میزان توان مصرفی است که از رابطه زیر محاسبه می‌شود [۲۲]:

$$\min\left\{\sum_{j\in 0} t_p^e W_{pe} + \sum_{j\in E_{AR}} (t_l^c W_{le} + t_m\, w_{me})\right\} \quad (۲)$$

در معادله بالا مدت‌زمان سپری‌شده روی هسته پردازنده برابر t_p^e و مدت‌زمان سپری شده روی مسیر ارتباطی t_l^c و t_m زمان کل پردازش بر روی هسته می‌باشد.

۳،۳،۳ هدف سوم: قابلیت اطمینان

سومین تابع هدف قابلیت اطمینان سیستم است، که نیاز است حداکثری باشد. برای تخمین قابلیت اطمینان به دلیل سادگی از روش‌های ارائه شده در [۲۷][۲۸] استفاده می شود. توجه داشته باشید که روش‌های دیگر تخمین قابلیت اطمینان نیز می‌توان استفاده کرد و انتخاب روش ارزیابی هرکدام از این پارامترهای هدف به کلیت روش پیشنهادی خدشه‌ای وارد نمی‌شود. مدل قابلیت انعطاف بر اساس روش زنجیره مارکوفی[۷] (DTMC) است. این مدل‌های گرافیکی شامل یک سری حالت محدود هستند. برای یک‌زمان بندی داده شده مدل DTMC از معماری سیستم گرفته‌شده است و سپس قابلیت اطمینان محاسبه می‌شود:

$$R = S(1,n)R_n \quad (۳)$$

که در آن S ماتریس پایه DTMC بوده و $S(i,j)$ نشان‌دهنده تعداد بارهای رفتن به حالت j از حالت پایه i هست و n تعداد حالات در مدل است. حداکثر کردن میزان قابلیت اطمینان به شکل حداقل کردن میزان عدم اطمینان نوشته می‌شود:

$$\min\{1-R\} \quad (۴)$$

۳،۳،۴ حل یک مسئله بهینه‌سازی چندهدفه

پس از مدل کردن توابع هدف، می‌توان شکل کلی معادله بهینه‌سازی چندهدفه را به شکل معادله (۵) بازنویسی کرد:

$$\text{Min(x)}\ \ z=f(x)= (f_1(x), f_2(x), f_3(x))^T \quad \text{s.t. } x \in X \quad (۵)$$

در معادله ذکرشده X نشان‌دهنده یک جواب دقیق از مجموعه جواب‌های قابل‌قبول هست. در این مسئله یک پاسخ مناسب برای زمان‌بندی با انتخاب مناسب متغیرهای تصمیم‌گیری که در قسمت ۱-۲-۲ که درباره این‌که چطور با استفاده از این متغیرهای تصمیم‌گیری وظایف به هسته‌ها تخصیص داده می‌شوند توضیح

داده شد، به دست می‌آید. هر سه تابع هدف f_1، f_2، و f_3 به‌طور مشخص از معادله‌های (۱)، (۲) و (۴) به‌دست‌آمده‌اند. تابع نهایی بهینه‌سازی $z = f(X)$ یک پاسخ از مجموعه جواب x از محیط تصمیم‌گیری است که با استفاده از متغیرهای تصمیم‌گیری به یک نقطه از فضای بهینه‌سازی منتسب خواهد شد. در مثال ما فضای هدف سه‌بعدی هست و دارای وزن‌های یکسان هست که با استفاده از سه تابع از معادلات (۱)، (۲) و (۴) به دست می‌آید.

ازآنجاکه مسئله بهینه‌سازی چند هدفه معمولاً دارای جواب یکتای بهینه نیست که بتواند تمام اهداف موردنظر را هم‌زمان بهینه کند، علاقه‌مند به پیدا کردن مجموعه جواب‌هایی هستیم که به آن‌ها راه‌حل‌های پارتو [۸] می‌گویند. پاسخ‌هایی که مجموعه راه‌حل‌های پارتو را شکل می‌دهند مجموعه پاسخ‌هایی هستند که جواب‌های دیگر فضای هدف نمی‌تواند آن‌ها را مغلوب کند. برای حل مسئله ژنتیک الگوریتم و تولید راه حل های پارتو از الگوریتم‌های تکاملی بهره می‌بریم چراکه این دسته از الگوریتم‌ها قادر هستند هم‌زمان چندین تابع هدف مستقل را بهینه کنند. به‌طور مشخص‌تر در این مقاله، از الگوریتم NSGAII [۲۶] استفاده می‌شود چون این الگوریتم نشان داده است که در مقابل سایر الگوریتم‌های تکاملی دارای مزایای فراوان ای ازجمله راحتی پیاده‌سازی و کم بودن سربار محاسباتی این الگوریتم است [۲۶]. شبه کد این الگوریتم در شکل (۵) نشان داده‌شده است. الگوریتم ژنتیک به شکل تکراری فرزندان جدید در جمعیت تولید می‌کند با استفاده از والدین نسل قبل با استفاده از دو عامل گر برش و جهش که در الگوریتم تعبیه‌شده است. در ابتدا الگوریتم نیاز به تولید یک مجموعه جمعیت اولیه دارد؛ جمعیت اولیه، که در ابتدای کار برای سادگی آن را به شکل تصادفی تولید کرده‌ایم. روشی که برای زمان‌بندی کردن به کاربرده‌ایم مانند روش ارائه‌شده در [۵] است. هر کروموزوم شامل مجموعه گره‌های وظیفه‌ای است که به مؤلفه‌های نرم‌افزاری، مجموعه گره‌های وظیفه‌ای که به مؤلفه‌های سخت‌افزاری و یال‌های ارتباطی است که به مؤلفه‌های حافظه تخصیص داده‌شده‌اند.

Input: N population size, M max number of generations
Output: Pareto frontier, non-dominated solutions in P_M

P_0 = GenerateInitialPopulation(); // size N
$Q_0 = \emptyset$; // start with children set empty
EvaluateObjectiveFunction(P_0); // calculate fitness
RankPopulation(P_0); // done according to fitness values
for ($i = 0$ to $M - 1$) **do**
  // Create children population:
  Q_i = SelectionCrossoverMutation(P_i);
  // Uncertainty aware, Monte Carlo based:
  EvaluateObjectiveFunction(Q_i);
  P_{i+1} = CombineParentsAndChildren(P_i, Q_i);
  RankPopulation(P_{i+1});
  // Elitism: keep non-dominated:
  P_{i+1} = SelectNIndividuals(P_{i+1});
end for

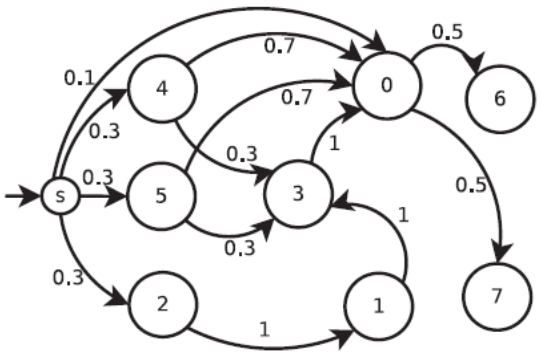
شکل ۵. الگوریتم NSGAII

۴ نتایج تجربی

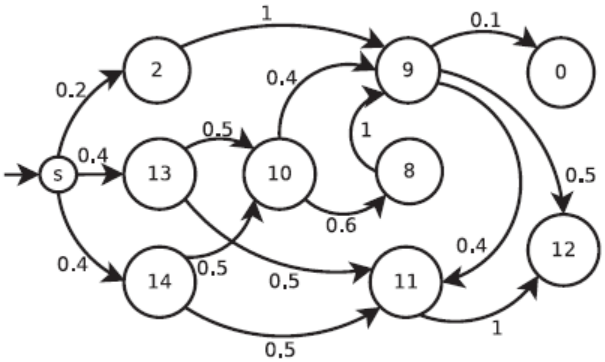
روش نگاشت و زمان‌بندی پیشنهادی به زبان برنامه‌نویسی C (که در مقایسه با زبان متلب و یا سایر زبان‌ها، از سرعت بالاتری برای اجرای برنامه به علت نزدیک بودن به زبان ماشین برخوردار است) انجام‌شده است. برای شبیه‌سازی از دو نمونه آزمایشی استفاده‌شده که هردوی آن‌ها در دامنه نرم‌افزارهای خودرو می‌باشند، سیستم ترمز ضد قفل [۹] ABS و سیستم کنترلی سرعت اقتباسی ACC است. سیستم کنترلی ABS و ACC در شکل ۶ نشان داده شده است. گراف وظایف ABC و ACC دارای ۱۴ گره و ۳۳ یال است. این دو نمونه آزمایشی از مرجع [۱۶] گرفته‌شده است. متغیرهای وابسته به الگوریتم NSGAII [۱۸] به ترتیب زیر مقداردهی شده‌اند، احتمال برش مقدار ۰٫۷، احتمال جهش ۰٫۰۵ در نظر گرفته شده است. تعداد دفعات تکرار الگوریتم ۹۰۰۰ هزار بار با اندازه جمعیت ۱۰۰ است. به دلیل آن که قابلیت اطمینان، توان مصرفی و زمان اجرای وظایف به‌صورت تابع اهداف چندگانه نشان داده‌شده است، تنها محدودیت‌هایی که در صورت‌بندی مسئله استفاده‌شده است شامل بن سازه سخت‌افزاری و افراز سخت‌افزاری/ نرم‌افزاری برنامه موردنظر است. به‌طور خاص در این مقاله، فرض می‌شود که بن سازه سخت‌افزاری شامل ۱۲ قسمت است که شامل ۱۰ پردازشگر و ۲ واحد ارتباطی (حافظه و گذرگاه باس) است، فرض شده است که یال‌های ارتباطی در گراف بر اساس نگاشت حافظه پیاده‌سازی شده‌اند و به این شکل عمل می‌کند که وظیفه منبع در مؤلفه حافظه می‌نویسد و وظیفه مقصد از مؤلفه حافظه می‌خواند. در معماری فرضی ما که در شکل شماره ۱ آمده است. مانند طرح پیشنهادی ارائه شده در [۱۹] ما دو نوع CPU در نظر گرفته‌ایم نوع اول کم‌مصرف با توان پرداز شی پایین و نوع دوم پرمصرف با

توان پردازشی بالا است. برای مؤلفه سخت‌افزاری ما دو نوع FPGA متفاوت را در نظر گرفته‌ایم، نوع اول کم‌مصرف با توان مصرفی پایین و نوع دوم پرمصرف با توان مصرفی بالا است. در کل FPGA ها سریع‌تر از CPU در نظر گرفته‌شده‌اند چون با استفاده از موازات بین دستوری با سرعت بالاتری دستورات را اجرا کنند. ممکن است که FPGA ها سریع‌تر از ASIC ها نباشند اما قابلیت بازپیکربندی دارند. ما زمان اجرا و نرخ وقوع خطا و میزان توان مصرفی را از گزارش‌ها ارائه‌شده در [۲۹] اقتباس کرده‌ایم.

در جدول ۲ نتیجه اجرا روش پیشنهادی و مقایسه آن با روش حریصانه بر روی‌داده آزمایشی واقعی ABS و ACC آورده شده است. شکل شماره ۶ نشان‌دهنده نمودار ABS وACC می‌باشد. الگوریتم حریصانه، همواره بهترین انتخاب در هر گام از حل مسئله انجام می‌دهد. در هر بار اجرای الگوریتم حریصانه، بر روی نمونه آزمایشی یک تابع هدف برای بهینه‌سازی انتخاب می‌شود و نتیجه نهایی به‌دست‌آمده از این الگوریتم با خروجی روش پیشنهادی مقایسه شده است، همان‌طور که در جدول ۲ ملاحظه می‌شود خروجی به‌دست‌آمده توسط روش پیشنهادی به روش حریصانه غالب بوده است یا نتیجه‌ای مشابه داشته است. برای مثال در جدول ۲.ج برای محاسبه میزان انرژی مصرفی به خروجی بهتری در مقایسه با الگوریتم حریصانه رسیده‌ایم این مقدار در الگوریتم حریصانه ۰،۱۷۶۴ در مقابل مقدار ۰،۱۵۶۳ در الگوریتم پیشنهادی می‌باشد.



شکل شماره۶.الف- نمودار ABS



شکل شماره۶.ب- نمودار ACC

شکل شماره ۶. نشان‌دهنده نمودار ABS و ACC [۲۱]

مهم‌ترین عامل اندازه‌گیری عملکرد سیستم بر روی الگوریتم زمان‌بندی گراف محاسبه طول گراف نتیجه از روی خروجی الگوریتم و Speedup می‌باشد که در ادامه این دو معیار را تعریف می کنیم. ازآنجایی‌که تعداد زیادی گراف با خصوصیات متفاوت استفاده‌شده‌اند، لازم است تا این گراف‌ها را به‌گونه‌ای یکسان‌سازی کنیم به کوتاه‌ترین طول ممکن در هر گراف برسیم تا بتوان نرخ مناسبی برای مقایسه آن‌ها به دست آورد، این مقدار طبق الگوریتم ارائه‌شده در [۱۷] به‌صورت رابطه (۶) است:

$$SLR = \frac{makespan}{\sum_{T_i \in CP_{min}} min\, p_k \in p\, (W(T_i, P_k))} \qquad (۶)$$

معیار دیگری که برای مقایسه انتخاب شده است معیار Speedup است. این متغیر از تقسیم، اجرای سریال وظایف بر روی سریع ترین وظیفه به زمان اجرای الگوریتم مورد نظر به دست می آید. معادله ی آن در زیر نشان داده شده است.

$$Speedup = \frac{min\, P_k \in p\, (\sum_{T_i \in T} W(T_i, P_k))}{makespan} \qquad (۷)$$

در معادلات بالا CP نشان‌دهنده مسیر بحرانی و $W(T_i, P_k)$ نشان‌دهنده زمان اجرا کار T_iبر روی پردازشگر می‌باشد. در زمان‌بندی به دست آمده، اگر هزینه محاسبات برای هر وظیفه به‌عنوان حداقل زمان اجرا روی پردازشگر در نظر گرفته شود به شکل CP_{min} نشان داده می‌شود.

جدول شماره ۲.الف. مقایسه زمان اجرا روش پیشنهادی با روش حریصانه

روش پیشنهادی	حریصانه	واحد	تابع هدف
۰،۰۵۴۵	۰،۵۴۵۴	ثانیه	زمان اجرا
۰،۸۰۸۷	۰،۸۱۶۹	وات	توان مصرفی
۰،۶۵۸۱	۰،۷۵۹۹	خرابی در سال	عدم قابلیت اطمینان

جدول شماره ۲.ب. مقایسه میزان عدم اطمینان درروش پیشنهادی با روش حریصانه

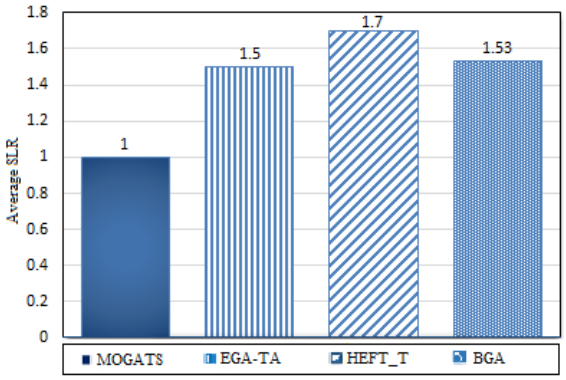
روش پیشنهادی	حریصانه	واحد	تابع هدف
۰،۱۹۵۴	۰،۲۷۲۷	ثانیه	زمان اجرا
۰،۳۹۰۹	۰،۲۱۵۴	وات	توان مصرفی
۰،۰۱۴۰	۰،۰۷۵۸	خرابی در سال	عدم قابلیت اطمینان

جدول شماره ۲.ج. مقایسه میزان انرژی مصرفی درروش پیشنهادی با روش حریصانه

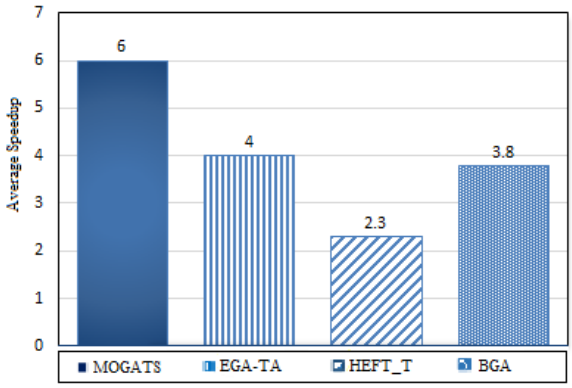
روش پیشنهادی	حریصانه	واحد	تابع هدف
۰،۳۹۰۹	۰،۲۷۲۸	ثانیه	زمان اجرا
۰،۱۵۶۳	۰،۱۷۶۴	وات	توان مصرفی
۰،۴۳۰۴	۰،۹۹۹۹	خرابی در سال	عدم قابلیت اطمینان

برای مقایسه عملکرد الگوریتم پیشنهادی با سایر روش‌ها چندین گراف تصادفی با استفاده از نرم‌افزار تولید گراف تصادفی تولید می شود. در هر گراف تصادفی تعداد گره‌ها (اندازه گراف وظایف) و تعداد یال‌های هر گره به شکل ایستا از مجموعه SET_V و SET_{out_degre} گرفته می شود گراف به‌طور با استفاده از توزیع یکنواخت با مقدار میانگی $\sqrt{n}/\propto$ تولید می‌شود و تعداد گره‌ها در هر سطح با توزیع یکنواخت و مقدار میانگین $\propto \times \sqrt{n}$ به شکل تصادفی تولید می‌شود در یک گراف تصادفی هزینه ارتباط برای تمام گره‌ها به شکل یکسان در نظر گرفته می‌شود مقدار SLR از نتیجه الگوریتم های زمانبندی مختلف برای ۱۰۰ گراف تصادفی با اندازه ۲۰ محاسبه شده است. نتایج به دست آمده از معیار SLR که در شکل (۷) نشان داده شده است، نسبت به SLR روش پیشنهادی نرمال شده اند. همانطور که در شکل (۷) مشاهده می شود الگوریتم پیشنهادی از منظر معیار SLR نسبت به تمام الگوریتم های پیشین بهتر عمل کرده است. به عنوان مثال، MOGATS نسبت به بهینه‌ترین الگوریتم پیشین یعنی EGA-TS [۱۹] توانسته SLR را به میزان ۵۰ درصد بهبود بخشد؛ به این معنی که میزان پارتو های نهایی و مجموعه پاسخ‌های بهینه به دست آماده به نسبت ۱،۵ به ۱ بهبود داشته‌اند. مشابه همین روند برای الگوریتم BGA و HEFT_T نیز قابل مشاهده می باشد تا حدی که می توان گفت MOGATS به طور میانگین در مقایسه با سه الگوریتم منتخب پیشین ۵۸،۶ درصد SLR را بهبود داده است.

نتایج به دست آمده از معیار Speedup برای الگوریتم های مختلف در شکل شماره ۸ نشان داده شده است. شاخص Speedup مشخص می‌کند بهترین پاسخ بهدست‌آمده توسط یک الگوریتم در هر سه شاخص انرژی مصرفی، قابلیت اطمینان و زمان اجرا نسبت به حالتی که به‌صورت متوالی که روی بهترین سخت‌افزار موجود اجرا شوند چقدر بهبود داشته است. همان‌طور که در شکل شماره ۸ مشاهده می شود، الگوریتم MOGATS توانسته است نسبت به EGA-TS به در این شاخص نیز به بهبود چشمگیری دست پیدا کند و میزان Speedup را از ۶ به ۸ رسانده است. البته زمان اجرا MOGATS و میزان حافظه موردنیاز جهت اجرای برنامه به ترتیب درصد نسبت به EGA-TS افزایش پیداکرده است که دلیل آن این است در MOGATS چند هدف بهینه سازی وجود دارد و در نتیجه اطلاعات حجم بیشتری از حافظه را اشغال خواهند نمود.



شکل شماره ۷. میزان SLR درروش پیشنهادی در مقایسه با سایر روش‌ها



شکل شماره ۸. میزان Speed up درروش پیشنهادی در مقایسه با سایر روش‌ها

۵ جمع‌بندی

در این مقاله، یک روش زمان‌بندی وظیفه ایستای چندهدفه برای طراحی سیستم‌های نهفته ارائه‌شده است. در این روش، وظایف به‌صورت یک گراف وظیفه مدل شده و با توجه به معماری سخت‌افزاری موجود، روشی برای نگاشت و زمان‌بندی وظایف بر روی معماری سخت‌افزاری با استفاده از الگوریتم ژنتیک پیشنهاد می‌شود. روش پیشنهادی به بهینه‌سازی چندهدفه برای پارامترهای زمان اجرای وظایف، توان مصرفی و قابلیت اطمینان می‌پردازد و به این منظور، از انتخاب راه‌حل‌های پارتو استفاده می‌نماید. استفاده از یک راهبرد بهینه‌سازی چندهدفه این امکان به طراح سیستمی می‌دهد تا بتواند بین پارامترهای مختلف طراحی سیستم (سخت‌افزاری/نرم‌افزاری) موازنه مدنظر خود را انجام دهد. همان‌طور که در بخش نتایج نشان داده‌شده است در مقایسه با روش EGATS در معیار SLR و Speedupبه ترتیب ۲۸,۶ و ۲۷,۸ درصد بهبود دست‌یافته‌ایم، که نشان می‌دهد توانسته‌ایم مجموعه وظایف را بر روی پردازنده‌های بهینه‌تر اجرا کنیم و درنتیجه به مجموعه پاسخ بهبودیافته نسبت به روش EGATSدست پیداکرده‌ایم.

مراجع

[۱] J. P. Roth, “Diagnosis of automata failures: a calculus and a method ”,*IBM Res Dev* ,pp. ۲۷۸-۲۸۱, ۱۹۶۶ .

[۲] S. A. Cook, “The complexity of theorem proving procedures ”,*ACM symposium on theory of computing* ,pp. ۱۵۱-۱۵۸, ۱۹۷۱ .

[۳] J. Shi, G. Fey, R. Drechsler, A. Glowatz, and F. Hapke, “PASSAT:efficient SAT-based test pattern generation ”, *Proceedings of the IEEE annual symposium on VLSI*, pp. ۱۸۷-۱۹۶, ۲۰۰۵ .

[٤] Salimi, Maghsood, et al. "Multi-objective Optimization of Real-Time Task Scheduling Problem for Distributed Environments." Proceedings of the ٦th Conference on the Engineering of Computer Based Systems. ACM, ۲۰۱۹.

[٥] J. Balcarek, P. Fiser, and J. Schmidt, “Test patterns compression technique based on a dedicated SAT-based ATPG ”,*In Digital System Design: Architectures, Methods and Tools (DSD) ۱۳th Euromicro Conference*, pp. ۸۰۵-۸۰۸, ۲۰۰۹

[٦] Majd, Amin, et al. "NOMeS: Near-optimal metaheuristic scheduling for MPSoCs." ۲۰۱۷ ۱۹th International Symposium on Computer Architecture and Digital Systems (CADS). IEEE, ۲۰۱۷.

[۷] A. Czutro, I. Polian, M. Lewis, P. Engelke, S. Reddy and B. Becker, “Thread parallel integrated test pattern generator utilizing satisfiability analysis ”,*Int J Parallel Progr* ,pp. ۱۸۵-۲۰۲, ۲۰۱۲ .

[۸] H. Marijn , M. Järvisalo and A. Biere, “Efficient CNF simplification based on binary implication graphs ”,*International Conference on Theory and Applications of Satisfiability Testing* ,pp. ۲۰۱-۲۱۵, ۲۰۱۱ .

[۹] P. Jackson and D. Sheridan, “Clause form conversions for boolean circuits ”,*International Conference on Theory and Applications of Satisfiability Testing* ,Berlin Heidelberg, ۲۰۰٤ .

[۱۰] Salimi, Maghsood, et al. "Multi-objective Optimization of Real-Time Task Scheduling Problem for Distributed Environments." Proceedings of the ٦th Conference on the Engineering of Computer Based Systems. ACM, ۲۰۱۹.

[۱۱] Majd, Amin, et al. "NOMeS: Near-optimal metaheuristic scheduling for MPSoCs." ۲۰۱۷ ۱۹th International Symposium on Computer Architecture and Digital Systems (CADS). IEEE, ۲۰۱۷.

[۱۲] Topcuoglu, S. Hariri and Min-You Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", IEEE Transactions on Parallel and Distributed Systems, vol. ۱۳, no. ۳, pp. ۲٦۰-۲۷٤, ۲۰۰۲.

[۱۳] G. Tseitin, *On the complexity of derivation in propositional calculus*, Berlin Heidelberg: Springer , ۱۹۸۳.

[۱٤] L. T, “Test pattern generation using boolean satisfiability ”,*IEEE Transactions on Computer-Aided Design* ,pp. ٤-۱۵, ۱۹۹۲.

[۱۵] Z. Wang and T. Fang, "Task Scheduling Model Based on Multi-Agent and Multi-Objective Static Scheduling Algorithm", Journal of Networks, vol. ۹, no. ٦, ۲۰۱٤. Available: ۱۰,٤۳۰٤/jnw.۹,٦,۱۵۸۸-۱۵۹۵

[۱٦] L. Zhou, "Integrated Multi-objective Scheduling for Multi-task on Perishable Products", Journal of Information and Computational Science, vol. ۱۲, no. ۱۸, pp. ٦٦۵۳-٦٦٦٤, ۲۰۱۵ .

[۱۷] M. Davis and H. Putnam, “A computing procedure for quantification theory ”,*Journal of the ACM (JACM)* ,Vol. ۷, No. ۳, pp. ۲۰۱-۲۱۵, ۱۹٦۰.

[۱۸] S. Eggersglüß and R. Drechsler, *High Quality Test Pattern Generation and Boolean Satisfiability*, Berlin: Springer Science & Business Media, ۲۰۱۲.

[۱۹] Akbari, M., Rashidi, H. and Alizadeh, S. (۲۰۱۷). An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. Engineering Applications of Artificial Intelligence, ٦۱, pp.۳۵-٤٦.

[۲۰] P. Beame, H. Kautz and A. Sabharwal, “Towards understanding and harnessing the potential of clause

learning ",*Journal of Artificial Intelligence Research* ,Vol. ۲۲, N۰. ۱, pp. ۳۱۹-۳۵۱, ۲۰۰٤.

[۲۱] W. Lin and X. Jin, "Toward The Flexible Operation Of Integrated Community Energy System: A Two-Stage Multi-Objective Scheduling Method", Science Trends, ۲۰۱۸.

[۲۲] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming*, Addison-Wesley Professional, ۲۰۱۰.

[۲۳] S. AlEbrahim and I. Ahmad, "Task scheduling for heterogeneous computing systems", The Journal of Supercomputing, vol. ۷۳, no. ٦, pp. ۲۳۱۳-۲۳۳۸, ۲۰۱٦. Available: ۱۰,۱۰۰۷/s۱۱۲۲۷-۰۱٦-۱۹۱۷-۲.

[۲٤] S. Parikh, S. Shah, N. M Patel, "Multi-objective Prediction based Task Scheduling Method in Cloud Computing", International Journal of Recent Technology and Engineering, vol. ۸, no. ٤, pp. ۹۳۸۸-۹۳۹٤, ۲۰۱۹. Available: ۱۰,۳۵۹٤۰/ijrte.d۹۷۰۲,۱۱۸٤۱۹.

[۲۵] M. Bushnell, V. Agrawal, Essentials of electronic testing for digital,memory and mixed-signal VLSI circuits, Boston, USA: Kluwer, ۲۰۰۰.

[۲٦] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiojective genetic algorithm: NSGA-II,"IEEE Trans. On Evolutionary Computation, ۲۰۰۲.

[۲۷] M. C. Hansen, H. Yalcin and J. P. Hayes,"Unveiling the ISCAS-۸۵ benchmarks: A case study in reverse engineering ",*IEEE Design & Test* , Vol. ۱٦, No. ۳, pp. ۷۲-۸۰, ۱۹۹۹.

[۲۸] G. E. Moore, "Cramming more components onto integrated circuits," *IEEE Solid-State Circuits Newsletter,* ۱۹۹٦.

[۲۹] I. Meedeniya, A. Aleti, and L. Grunske, "Architecture-driven reliability optimization with uncertain model parameters", J.of Systems and Software, ۲۰۱۲.

---

۱ Chromosome

۳ Process
٤ Crossover
۵ Parents

٦ Mutation
۷ Markov Chain
۸ Pareto frontier
۹ Anti-Break System