علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۳، شماره ۲، ۱۳۹۴
صفحات ۳۲-۴۶
مقاله عادی

نگاشت و زمان‌بندی همزمان وظایف و ارتباطات انرژی‌آگاه بی‌درنگ در ساختارهای چندهسته‌ای

امین‌اله مه‌آبادی فاطمه عسگری بیدهندی

دانشکده فنی و مهندسی، دانشگاه شاهد، تهران، ایران

چکیده

در این مقاله یک متدلوژی نگاشت و زمان‌بندی بی‌درنگ انرژی‌آگاه برای برنامه‌ریزی همزمان وظایف و ارتباطات با هدف حل سریع با جواب نزدیک بهینه درتراشه‌های چند هسته‌ای ارایه می‌شود. متدلوژی پیشنهادی با برخورداری از ساختار نوین کروموزوم در الگوریتم ژنتیک و برخورداری از تابع جهش شبیه‌سازی گداخت، دارای قابلیت جلوگیری از تولید راه‌حل‌های غیرممکن جهت کاهش زمان تولید جواب نزدیک بهینه است. تحلیل ما از نتایج آزمایشات در فضای نانو تکنولوژی نشان می‌دهد که در نگاشت و زمان‌بندی همزمان نسبت به روش سنتی ژنتیک از سرعت همگرایی بسیار خوبی برخورداری است و به‌طور متوسط در ساختار زمان‌بندی حدود ۱۰٪ و در ساختار نگاشت بیش از ۹۰٪ بهبود سرعت در زمان اجرا، همراه با تولید جواب نزدیک بهینه را نشان می‌دهد.

کلمات کلیدی: شبکه بر تراشه، زمان‌بندی وظایف و ارتباطات، ژنتیک الگوریتم، تابع جهش، شبیه‌سازی گداخت، زمان‌بندی انرژی‌آگاه.

۱- مقدمه

تراشه‌ها آینده می‌توانند شامل صدها هسته پردازشی از پیش طراحی شده باشند که درون یک تراشه کنار هم قرار گرفته‌اند و یک تراشه با پیچیدگی بسیار بالا را ایجاد کنند [۱]. از مهم‌ترین مسائل چنین تراشه‌هایی، نگاشت و زمان‌بندی منابع درون تراشه است که با ترکیب ابعاد نگاشت و زمان‌بندی همزمان وظایف و ارتباطات از نوع مسایل سخت NP و نیاز تقلیل زمان حل آن است [۲]. در این بخش به مساله نگاشت و زمان‌بندی، چالش‌های مساله، راهبرد حل مساله، و نوآوری‌های ارایه شده می‌پردازیم.

۱-۱- نگاشت و زمان‌بندی

اکنون حل مساله نگاشت و زمان‌بندی همزمان وظایف و ارتباطات آنها با هدف کمینه‌سازی مصرف انرژی جهت وظایف بی‌درنگ در تکنولوژیِ زیرمیکرون مورد توجه قرار دارد و با نگاه به چالش‌های جدید مصرف انرژی [۳]، حل سریع و نزدیک بهینه آن برای تراشه‌های چندهسته‌ای از مساله‌های بسیار سخت محسوب می‌شود. بیشینه فرکانس عملیاتی یک پردازنده تک هسته‌ای می‌تواند از طریق توان نشتی و اثرات فرکانس رادیویی به آن آسیب بزند. این مشکل سازندگان را به محدود کردن بیشینه فرکانس پردازنده و به طراحی تراشه‌های چندهسته‌ای با فرکانس‌های کمتر سوق می‌دهد [۴] [۵] هر چند با افزایش تقاضای کارآیی کاربردهای نهفته پیچیده مدرن، افزایش فرکانس پردازنده تک‌هسته‌ای یا مشتری‌سازی آن پردازنده نمی‌تواند پاسخگو باشد لذا نیاز به پردازنده‌های با هسته‌های زیاد و با ارتباطات داده‌ای بسیار، یک ضرورت است [۶]. مفهوم اصلی آن تشریح نگاشت و زمان‌بندی وظیفه‌ها و ارتباطات گوناگون کاربردها است که به‌صورت کارآ بتواند بر روی چندین هسته به‌صورت همزمان به‌منظور افزایش کارآیی، اجرا شوند [۷] [۸].

افزایش تقاضا برای سیستم‌های با کارایی[۱] بالا، سبب کوچک شدن زیاد تکنولوژی نیمه‌هادی[۲] در سیستم‌های پیچیده الکترونیکی شده است.این پیشرفت

تکنولوژی طراحان را قادر به تجمیع پردازنده‌های همگن زیاد در یک تراشه می‌سازد که به سیستم چندپردازنده برتراشه[۳] (MPSoC) معروف هستند. گرچه برای هسته‌های بسیار در کنار توجه به پردازش وظیفه‌ها، برای افزایش قابلیت مقیاس‌پذیری و کارآیی ارتباطات آنها نیازمند زیرساخت مناسب شبکه برتراشه (NoC) هستیم [۹]. کوچکتر شدن عناصر سیستم نیز سبب افزایش توان[۴] و انرژی مصرفی آن می‌شود [۱۰]. مطابق تحقیقات مقاله [۷]، افزایش توان مصرفی نشتی[۵] در تکنولوژی‌های زیر میکرون مشکلات بسیاری دارد. از سوی دیگر میزان توان نشتی مصرفی تراشه در تکنولوژی زیرمیکرون[۶]، به ۶۰٪ کل توان مصرفی خواهد رسید و ضرورت توجه به ذخیره‌سازی مصرف انرژی در تراشه را نشان می‌دهد [۱۰].

فرآیند طراحی سیستم‌های نهفته[۷] معمولا دارای سه گام تقسیم‌بندی وظیفه‌های کاربرد، نگاشت وظیفه بر عنصر پردازش و و زمان‌بندی وظیفه‌ها و ارتباطات است [۱۱] [۱۲]. تقسیم‌بندی کاربرد[۸] براساس قیود[۹] و محدودیت‌های سیستم به وظیفه‌های سخت‌افزاری[۱۰] و نرم‌افزاری[۱۱]، نگاشت وظیفه‌های بر روی عناصر پردازشی موجود (نگاشت وظیفه‌های سخت‌افزاری بر روی مدارهای خاص منظوره (ASIC) و FPGAها، و وظیفه‌های نرم‌افزاری برروی پردازنده‌های همه‌منظوره، DSPها[۱۲] و شتاب‌دهنده‌ها[۱۳]، و زمان‌بندی همزمان وظیفه‌ها و ارتباطات برای دستیابی به کارایی بهینه جهت برآوردن قیود مختلف سیستم مانند زمان‌بندی بی‌درنگ و توان مصرفی انجام می‌شود [۱۳]. گرچه در فرآیند نگاشت بر بسترهای ناهمگن، به تعیین نوع هسته برای نگاشت وظیفه و هزینه نگاشت هر وظیفه روی هسته‌های متفاوت (یعنی هزینه پیاده‌سازی مانند کارآیی، توان مصرفی و اشتغال منابع) نیز نیاز است [۱۴].

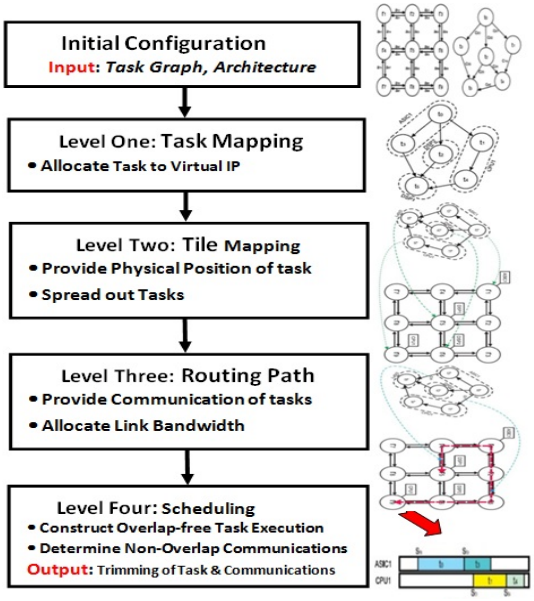
نگاشت وظیفه‌ها بر سیستم‌های چندهسته‌ای و بسیار هسته‌ای، شامل رعایت ترتیب اجرای وظیفه‌ها و ارتباطات آنها براساس بعضی معیارهای بهینه‌سازی مانند توان مصرفی و کارآیی محاسبات و ارتباطات است [۱]. یعنی ارتباطات وظیفه‌ها به‌منظور بهینه‌سازی تاخیرات ارتباطی و انرژی مصرفی یا روی یک هسته یا بر روی هسته‌ای نزدیک به‌یکدیگر نگاشت می‌شوند [۲]. بهینه‌سازی برای برآوردن قیود کارآیی اجرای کاربردها ضروری است. این ضرورت توسعه متدلوژی‌های کارآی نگاشت و زمان‌بندی را نشان می‌دهد که نیازمند مدل کاربرد، مدل بسترکاری، قیود (مانند کارآیی محاسبات و توان)، مدل کارآیی ارتباطات درونی (مانند زمان اجرا و توان مصرفی) و تخمین زمان بدترین اجرای وظیفه با پیاده‌سازی بر هسته‌های متفاوت (مانند پردانده‌های خاص منظوره و همه‌منظوره) است [۱] [۱۱].

۱-۲- چالش‌های نگاشت و زمان‌بندی

چالش‌های حل مساله زمان‌بندی کاربردها عبارت از پیچیدگی جستجوی فضای راه‌حل، زمان طولانی پاسخ مساله، دقت جواب پاسخ، اهداف همزمان طراحی با قیود فرآوان، افزایش ابعاد وظیفه‌ها یا هسته‌های پردازشی، سرعت همگرایی و دقت روش‌های ارایه شده است. در روش‌های فرامکاشفه‌ای مانند الگوریتم ژنتیک [۱۵]، بهینه‌سازی گروهی ذرات [۱۶] و شبیه‌سازی گداخت [۱۷] فضای راه‌حل برای زمان‌بندی بهینه جستجو می‌شود. روش‌های مکاشفه‌ای، ترکیبی از تکنیک‌های جستجوی شبه تصادفی و بهینه‌سازی هستند که کشف فضای مساله را براساس تجارب انجام می‌دهند. این روش‌ها زمانی به‌کار می‌روند که جستجوی جامع و متدهای قطعی، بسیار سخت یا بکاربردن آن‌ها غیرممکن باشد و اگر زمان جستجو با افزایش ابعاد مساله به‌صورت نمایی رشد کند [۱۸].

روش‌های مکاشفه‌ای یک راه‌حل مطلوب با زمان نسبتا کوتاه فراهم می‌آورند. گرچه به‌دلیل نیاز به افزایش سرعت ممکن است پاسخ این روش‌ها مطلوب باشد ولی بهینه یا نزدیک به بهینه نباشد. برای جواب دقیق بهینه روش‌های برنامه‌ریزی خطی صحیح[۱۴]، برنامه‌ریزی خطی مشروط[۱۵]، برنامه‌ریزی غیرخطی[۱۶] و برنامه‌ریزی خطی صحیح مخلوط[۱۷] مثال‌های قابل ارایه با پیچیدگی چند جمله‌ای و غیر آن هستند [۹]. برنامه‌ریزی ریاضی، متدهای کمینه یا بیشینه‌سازی اهداف با ارضاء قیود مساله را برای جواب دقیق یا بهترین جواب دقیق فراهم می‌کنند.

پیچیدگی فضای جستجوی مساله با نیاز به جواب بهینه، افزایش وظیفه‌ها، همزمانی زمان‌بندی و نگاشت با پشتیبانی از عملیات همجوار پردازش و ارتباطات، و افزایش قیود آنها، ارایه روش‌های حل این مساله را بسیار سخت می‌کند [۱۹]. این روش‌ها می‌توانند خاص منظوره یا همه‌منظوره، سازنده یا براساس بهبود تکرار باشند. مکاشفه‌ای‌های سازنده راه‌حل جزیی می‌سازند تا به راه‌حل کامل برسند. مکاشفه‌ای‌های قابل تبدیل سعی بر تبدیل و بزرگ کردن فضای مساله تا رسیدن به کل جواب را دارند. زمان‌بندی فهرستی و الگوریتم ژنتیک مثال‌هایی از روش‌های مکاشفه‌ای همه‌منظوره از کلاس سازنده و قابل تبدیل هستند.



شکل ۱- متدلوژی پیشنهادی نگاشت و زمان‌بندی همزمان وظایف و ارتباطات

افزایش تعداد هسته‌ها، وظیفه‌ها، ارتباطات و همزمانی نگاشت و زمان‌بندی به سختی مساله کمک بسیار می‌کند که حل سریع آن با یافتن پاسخ بهینه یا نزدیک بهینه از محورهای تحقیقاتی نوین محسوب می‌شود [۱۰]. از آنجا که یافتن جواب بهینه و برآوردن تمامی قیود بسیار سخت و زمان‌بر است لذا طراحی راه‌حل مکاشفه‌ای با ارایه جواب مطلوب یک ضرورت همیشگی برای پیاده‌سازی سیستم‌های نهفته محسوب می‌شود [۲۰]. با مطالعه کارهای تحقیقاتی ارایه شده در حوزه نگاشت و زمان‌بندی وظیفه‌ها می‌توان بیان کرد که توجه بسیاری به متدلوژی‌های انرژی‌آگاه شده ولی در تسریع حل همزمان مساله همچنان مشکل کاهش زمان تولید محصول تا بازار نیز وجود دارد [۲۱] [۲۲] [۲۳] [۲۴].

۱-۳- راهبرد متدلوژی پیشنهادی

از آنجاکه یافتن تمامی نگاشت‌های و زمان‌بندی‌های ممکن با کاربردهای بزرگ پیچیده و در ابعاد بزرگ یک قالب کار در زمان محدود ممکن نیست لذا به داشتن راهبردهای تحلیل سریعتر با اهداف طراحی برای کشف نگاشت و زمانبندی کارا نیازمند هستیم. ما با استفاده از داده‌های اولیه، کروموزوم‌های مناسبی برای بهینه‌سازی تاخیر و انرژی مصرفی، در دو مرحله نگاشت و زمان‌بندی در معماری مش دو بعدی با در نظر گرفتن الگوریتم مسیریابی به‌عنوان راهبرد اولیه انتخابی

ارایه می‌کنیم. تمرکز ما بر تعریف کروموزوم با امکان جلوگیری از ایجاد راه‌حل‌های غیرممکن در الگوریتم ژنتیک پیشنهادی (بعد از مراحل جهش و تقاطع) به‌منظور کاهش تعداد جستجوها است. برای تضمین بهبود این کروموزوم بعد از مرحله جهش و به‌عنوان فاز جهش ژنتیک پیشنهادی از الگوریتم بهبود شبیه‌سازی گداخت استفاده شده است.

همچنین سعی داریم که از روش برنامه‌ریزی خطی عدد صحیح برای افزایش دقت جواب‌ها بهره ببریم. با این انتخاب‌ها، تعداد جستجو و در نتیجه زمان حل‌مساله کاهش و جواب بهینه حاصل می‌شود. معماری متدلوژی تکاملی ترکیبی پیشنهادی (مطابق شکل ۱) در سه مرحله "مقیدکردن وظایف به پردازشگرها"، "نگاشت وظایف و ارتباطات"، و "زمان‌بندی وظایف" با ساختار شبکه بر تراشه به حل مساله می‌پردازد. ورودی مساله گراف وظایف و معماری شبکه است. ابتدا وظایف با هدف بهینه‌سازی انرژی مصرفی مسیرها و هسته‌ها به هسته‌هایی که توانایی اجرای آن‌ها را دارند ملحق و نگاشت می‌گردند. سپس کار زمان‌بندی وظیفه‌ها و ارتباطات انجام می‌گیرد.

۱-۴- نوآوری

ما در این مقاله یک متدلوژی نگاشت و زمان‌بندی همزمان انرژی‌آگاه ایستا و شبه ایستا با در نظر گرفتن جواب مطلوب و نزدیک بهینه در سیستم‌های برپایه شبکه بر تراشه برای کاربردهای بی‌درنگ سخت ارایه می‌دهیم. براساس دانش ما، این مقاله اولین کاری نیست که با لحاظ کردن همزمان نگاشت و زمان‌بندی انرژی‌آگاه وظیفه‌ها را بیان می‌کند ولی سعی بر حل سریع و یافتن پاسخ نزدیک بهینه آن با کاهش فضای جستجو دارد. برای تقلیل زمان اجرای این قالب‌کار از الگوریتم ژنتیک، برای بهبود جواب‌های میانی از تکنیک شبیه‌سازی گداخت، و برای افزایش دقت جواب‌ها از برنامه‌ریزی صحیح بهره برده‌ایم که توانایی ارایه سریع جواب مطلوب، بهبود جواب‌ها در تکرار با هدف بهینه‌سازی انرژی مصرفی را برای حل همزمان پردازش و ارتباطات دارد. روش ارایه شده، مستقل از نوع الگوریتم زمان‌بندی انرژی‌آگاه است و هر الگوریتم جایگزین بهتر آن می‌تواند نتایج را بهبود بخشد. نتایج کار در تکنولوژی ۹۵ نانومتر ارزیابی شده و به‌طور خلاصه نوآوری‌های آن در این مقاله عبارتست از:

- ارایه یک قالب‌کار برای همزمانی تخصیص و زمان‌بندیِ وظایف و ارتباطات به‌صورت انرژی‌آگاه با افزایش سرعت همگرایی و دقت جواب و کاهش زمان تولید محصول،
- ارایه یک ساختار نوین کروموزوم الگوریتم ژنتیک با قابلیت جلوگیری از ایجاد راه‌حل‌های غیرممکن بعد از مراحل جهش و تقاطع،
- ارایه روش تضمین بهبود جواب هر نسل الگوریتم ژنتیک با فرار از تله بهینه‌های محلی جهت تولید بهینه سراسری،
- ارایه مدل دقیق حل مساله با ترکیبی از الگوریتم ژنتیک، شبیه‌سازی گداخت و برنامه‌ریزی خطی صحیح جهت ارایه سریع جواب مطلوب، بهبود جواب‌های مطلوب در تکرار جهت ارای جواب دقیق با هدف بهینه‌سازی انرژی مصرفی، و
- ایجاد فضای آزمون مناسب برای ارزیابی کارآیی الگوریتم در تکنولوژی ۹۵ نانومتر برای ساختارهای بر تراشه همگن[۱۸].

ما در ادامه مقاله و در بخش ۲ کارهای مرتبط با مساله را بررسی می‌کنیم. در بخش ۳ متدلوژی پیشنهادی را ارایه می‌دهیم. در بخش ۴، مدل نگاشت و زمان‌بندی پیشنهادی را برای نگاشت و زمان‌بندی همزمان انرژی‌آگاه تشریح می‌کنیم. نتایج شبیه‌سازی و آزمایشات را در بخش ۵ به‌طور مشروح ارایه می‌دهیم. نهایتاً در بخش ۶ نتیجه‌گیری از متدلوژی پیشنهادی را بیان می‌کنیم.

۲- کارهای مرتبط

نگاشت و زمان‌بندی وظایف روی چند هسته و پردازنده، مساله سخت[۱۹] است [۱]. لذا در ابعاد و اندازه بزرگ فقط می‌توان آن را با استفاده از روش‌های مکاشفه‌ای سازنده[۲۰] یا قابل تبدیل[۲۱] حل کرد. جواب مساله‌های با اندازه کوچک می‌توانند با روش‌های قطعی و با جواب بهینه یافت شود. روش‌های قطعی (مانند. روش شاخه و کران[۲۲]) به‌طور جامع راه‌حل‌های فضای مساله را کشف می‌کنند و بهترین جواب ارایه می‌دهند. برای حل مساله‌های بزرگ و یافتن سریع جواب نزدیک بهینه راه‌حل‌های مکاشفه‌ای[۲۳] زیادی مانند مکاشفه‌ای مبتنی بر لیست [۱۲] و فرامکاشفه‌ای [۱۱] ارایه شده‌اند.

جدول ۱- دسته‌بندی متدلوژی‌های زمان طراحی

مقاله	معماری	هدف بهینه‌سازی
ارسیلا و همکاران [۲۶]	همگن	زمان اجرا
راجیرو و همکاران [۳۱]	همگن	زمان اجرا
ساتیش و همکاران [۵۰]	همگن	زمان اجرا
بانی‌فتی و همکاران [۴۹]	همگن	زمان نگاشت و کیفیت
لین و همکاران [۸]	همگن	گذردهی، اشتغال منابع
وو و همکاران [۲۷]	همگن	انرژی مصرفی
راهی و همکاران [۳۴]	همگن	انرژی مصرفی
چن و همکاران [۱۸]	همگن	انرژی مصرفی
هو و همکاران [۲۲]	همگن	انرژی مصرفی، زمان اجرا
مارکون و همکاران [۲۳] [۲۴]	همگن	انرژی مصرفی، زمان اجرا
ایشیا و همکاران [۹]	همگن	انرژی مصرفی، زمان اجرا
می‌یر و همکاران [۲۵]	همگن	قابلیت اطمینان
تلی و همکاران [۵۱]	همگن	قابلیت اطمینان، دما
زانگ و همکاران [۳۸]	همگن	انرژی مصرفی
وو و همکاران [۵۲]	ناهمگن	زمان اجرا
مارکوسکی و همکاران [۵۳]	ناهمگن	زمان اجرا
چه و همکاران [۱۴]	ناهمگن	زمان اجرا
کاستریلون و همکاران [۱۳]	ناهمگن	زمان اجرا
مانولاچی و همکاران [۲۹]	ناهمگن	زمان کشف، دقت
جاوید و همکاران [۳۰]	ناهمگن	زمان کشف، دقت
وئو و همکاران [۵۲]	ناهمگن	انرژی مصرفی
هارتمن و همکاران [۲۰]	ناهمگن	قابلیت اطمینان
متدلوژی پیشنهادی	همگن	انرژی مصرفی، دقت، زمانِ اجرا

۲-۱- متدلوژی‌های نگاشت و زمان‌بندی

رده‌بندی‌هایی برای کلاس‌بندی متدلوژی‌های نگاشت مانند با اساس معماری هدف، با اساس معیارهای بهینه‌سازی، براساس بارکاری و غیره وجود دارد. متدلوژی‌های نگاشت در سطح زمان- طراحی و زمان اجرا براساس سناریوهای بارکاری ثابت و پویا، عمل بهینه‌سازی را انجام می‌دهند. براساس معماری هدف به سیستم‌های همگن و ناهمگن تقسیم می‌شوند.

زمان‌بندی زمان اجراء، نیازمند مدیری است که نگاشت زمان اجراء را انجام دهد و علاوه بر آن مسئولیت زمان‌بندی وظیفه [۲۵]، کنترل منابع، کنترل ساختار و مهاجرت وظیفه در زمان اجراء را بر عهده داشته باشد. این مدیر می‌تواند مدیریت متمرکز (استفاده از یک هسته به عنوان مدیر)، مدیریت توزیعی (تقسیم به نواحی کلاستری و استفاده از یک هسته در هر کلاستر به عنوان مدیر و ارتباط از طریق یک مدیر سراسری برای انتخاب بهترین کلاستر برای نگاشت) یا ترکیبی از هر دو داشته باشد.

متدلوژی‌های نگاشت زمان طراحی، مناسب سناریوهای بارکاری ثابت که برای مجموعه‌ای از کاربردهای از پیش مشخص با محاسبات و رفتار ارتباطی شناخته‌شده و دارای قالب‌کار ثابت بیان می‌شوند، قادر به پشتیبانی از پویایی زمان اجرا (مانند کاربردهای چند رسانه‌ای و شبکه‌ای) نیستند. لذا متدلوژی خاص خود را نیاز دارد که در راستای بررسی ما نیست.در متدلوژی‌های نگاشت زمان طراحی، نگاه سراسری به سیستم وجود دارد که تصمیم‌سازی بهتری برای استفاده از منابع سیستم صورت گیرد.

لذا در مقایسه با متدلوژی‌های نگاشت زمان اجرا که نگاه محلی و همسایگی دارند از کیفیت بهتر نگاشت و زمان‌بندی برخوردار است. بیشتر متدلوژی‌های نگاشت مربوط به زمان طراحی دارای قابلیت معماری همگن یا ناهمگن نیستند. در جدول ۱، کارهای اخیر نگاشت زمان طراحی، براساس معماری هدف و اهداف بهینه‌سازی ارایه شده است. بهینه‌سازی کارآیی محاسبات برای موفقیت ضرب‌الاجل‌ها یا کمینه‌سازی زمان اتمام کارها مهم است. کارآیی ممکن است به زمان اجراء، تاخیر، پریود، توان خروج و مانند آن که مرتبط با اطلاعات زمان‌بندی است اشاره کند.

روش‌های جستجوی متفاوتی به‌منظور یافتن نگاشت مطلوب بهینه یا نزدیک بهینه وظیفه‌ها مانند شبیه‌سازی گداخت [۸] [۲۶]، ژنتیک الگوریتم [۲۷]، جستجوی ممنوع [۲۹]، و برنامه‌ریزی خطی صحیح [۳۰] استفاده می‌شود. زمان اجرا و حافظه مصرفی در این کار بهینه‌سازی شده است. در کار دیگری مساله نگاشت وظیفه‌ها با توجه به بیشینه‌سازی گذردهی سیستم به بهبود ۲۰٪ دست یافت [۸]. در کار مشابهی با استفاده از GA برای اجرای کاربردهای جریان‌داده سنکرون روی یک سیستم چندهسته‌ای با توجه به تخصیص محدود حافظه به هر هسته صورت گرفت [۵۳]. در تحقیق دیگری با تمرکز بر فرموله کردن ILP مساله نگاشت حل شد. تمامی این متدلوژی‌ها با وجود دست‌یابی به پاسخ‌های مناسب، از هزینه بالا محاسباتی برای کاربردهای دارای وظیفه بسیار برخوردارند [۲۳].

هدف استراتژی‌های دیگر تمرکز بر فضای جستجو با هدف تقلیل هزینه محاسبات است. ترکیب برنامه‌ریزی خطی و برنامه‌ریزی قیود برای تسریع اجرا هدف است [۳۱]. معماری هدف این ساختار براساس ساختار باس بنا شده و مقیاس‌پذیر نبود. در کار دیگری از تکنیک تجزیه برای تسریع بهینه‌سازی قیود مساله استفاده شد [۵۰]. در تحقیقی دیگر کار بر روی بهینه‌سازی زمان ارتباطات و محاسبات صورت گرفت [۳۲]. متدهای بسیار دیگری که این راه را طی کردند گرچه دارای زمان کمتر جستجو بودند ولی از پاسخ‌های با کیفیت بالا برخوردا نبودند.

۲-۲- محدودیت‌های زمان طراحی

بیشتر متدلوژی‌های زمان طراحی، روش‌های جستجو مبنا (یعنی GA[۲۴]، ILP، SA[۲۵]) هستند که متحمل هزینه‌های محاسباتی بالا می‌شوند. گرچه آنها برای سیستم‌های کوچک پاسخ کارآ فراهم می‌کنند ولی در مقیاس بزرگ، ممکن است زمان ارزیابی آنها قابل‌پذیرش نباشد. این مساله در ترکیب ارتباطات و پردازش وظایف کار را پیچیده‌تر و این زمان را بیشتر می‌کند. این زمان می‌تواند بوسیله هرس فضای جستجو تقلیل یابد ولی ریسک این مساله، از دست دادن پاسخ‌های نگاشت با کیفیت بالا است. مقیاس‌پذیری حل این مساله با هدف جواب بهینه و ضرورت افزایش تعداد هسته‌ها در کنار نیازمندی به تکنولوژی شبکه بر تراشه یک ضرورت تحقیقاتی جدید است. مساله اصلی، کاهش فضای جستجو با هدف یافتن پاسخ‌های نگاشت و زمان‌بندی با همزمانی پردازش و ارتباطات و با کیفیت بالا یعنی سرعت پاسخ زیاد، جواب نزدیک بهینه، و سرعت همگرایی بالا است.

در تمامی سطوح فرآیند طراحی تکنولوژی زیرمیکرون، مساله همزمانی نگاشت و زمان‌بندی در طراحی انرژی‌آگاه ضروری است. کارهای قبلی یا فقط روی بهبود نگاشت تمرکز کرده‌اند و یا فقط به مساله زمان‌بندی توجه داشته‌اند. البته بیشتر تحقیقات یا تمرکز بر پردازش وظایف داشته‌اند و یا ارتباطات را مدل کرده‌اند و از همزمانی تمرکز بر مدل‌سازی پردازش وظایف و ارتباطات صرف‌نظر شده است. همچنین، ارایه بهبود دقت محاسبات برای ارایه جواب نزدیک بهینه در کارهای قبلی به اندازه کافی دقیق نیست و از همگرایی خوبی برخوردار نبوده است. این دلایل ما را بر آن داشت که در این مقاله روش جدید مکاشفه‌ای با مدل‌سازی همزمان ارتباطات در کنار محاسبات وظایف بی‌درنگ برای حل‌مساله همزمان نگاشت و زمان‌بندی انرژی‌آگاه تحت نگرش یافتن جواب سریع نزدیک به بهینه با ترکیب الگوریتم ژنتیک برای سرعت جواب، شبیه‌سازی گداخت برای بهبود تکرار و روش برنامه‌ریزی خطی صحیح برای افزایش دقت و یافتن جواب نزدیک بهینه ارایه دهیم.

۲-۳- متدلوژی‌های انرژی‌آگاه

مساله مهم در تحقیقات نهفته مدرن، بهینه‌سازی مصرف انرژی برای تداوم بیشتر باتری است و نیازمندی شدیدی به‌کار در زمان طراحی و تداوم زمان اجرا دارد. در کار مولی و همکاران، نتیجه ذخیره‌سازی ۵۴٪ توان مصرفی حاصلو متدلوژی با هدف برآوردن قیود کارآیی ارایه شد [۳۳]. نتیجه تحقیق راهی [۳۴] با اساس ILP بحث بهینه‌سازی نگاشت هسته‌ها در معماری توری به‌منظور کمینه کردن مصرف انرژی یا ازدحام NoC[۲۶] بود که ۸۱٪ ذخیره‌سازی انرژی حاصل شد. یک متدلوژی نگاشت چند مرحله‌ای برای بهینه‌سازی در [۱۸] ارایه شد. و همکاران یک روش مبتنی بر GA ارایه دادند که انرژی مصرفی را با روش ولتاژ مقیاسی پویا تا ۵۱٪ کاهش داد.

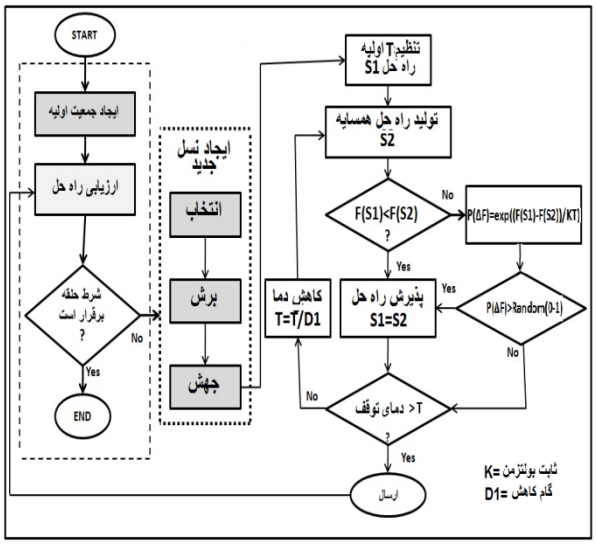
در بعضی از تحقیقات مساله بهینه‌سازی در دو بعد کارآیی محاسبات و انرژی مصرفی صورت گرفت [۹] [۲۲] [۲۴] در تحقیق هو و همکاران یک روش نگاشت برای تقلیل انرژی مصرفی از طریق کاهش انرژی ارتباطات در کنار تضمین کارآیی مورد نیاز با ۵۱٪ ذخیره انرژی ارایه شد [۲۲]. مارکون و همکاران با توسعه کار [۲۲] تکنیکی ارایه دادند که زمان‌بندی ارتباطات را علاوه بر میزان ارتباطات ارایه [۲۴] و علاوه بر تقلیل ۹۸٪ زمان اجرا به‌میزان قابل توجه ذخیره انرژی انجام دادند. اشیا و همکاران روشی برمبنای GA ارایه دادند که به پاسخ پرتو[۲۷] برای عوامل بهره‌وری و دقت دست یافتند در حالی که برای انرژی مصرفی و کارآیی نیز بهینه‌سازی انجام دادند [۹]. گرچه این کارها توانستند انرژی مصرفی را کاهش دهند ولی در زمینه دقت پاسخ نزدیک بهینه و سرعت تولید محصول تا بازار به نقطه خوبی دست نیافتند.

۳- متدلوژی پیشنهادی

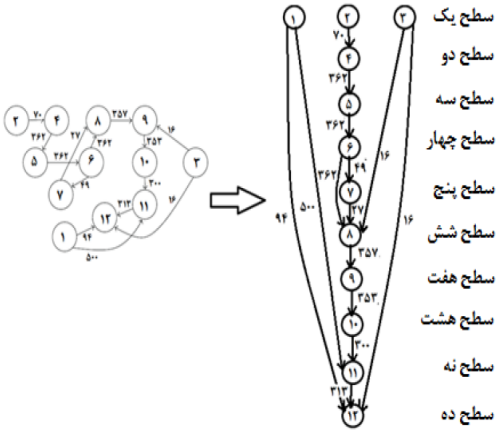
در این بخش به تعریف ارایه متدلوژی نگاشت و زمان‌بندی وظیفه‌ها می‌پردازیم. در ابتدا کروموزومی خوب برای مساله زمان‌بندی با امکان جلوگیری از ایجاد راه‌حل‌های غیرممکن بعد از مراحل جهش و تقاطع، ارایه می‌شود. سپس برای بهبود روش با الگوریتم ژنتیک در مرحله جهش، از الگوریتم شبیه‌سازی گداخت بهره‌برداری می‌شود تا بهبود کروموزوم بعد از مرحله جهش تضمین گردد. بعد مساله نگاشت با هدف کاهش انرژی با روش شبیه‌سازی خطی عدد صحیح فرموله می‌شود. نهایتاً برای کاهش جایگشت‌ها در حل قطعی مساله زمان‌بندی جهت جواب بهینه، راه‌حل پیشنهادی بیان می‌گردد.

۳-۱- کروموزم در زمان‌بندی

ما کروموزومی برای بخش زمان‌بندی طراحی کرده‌ایم که از ایجاد راه‌حل‌های غیر امکان‌پذیر پس از مراحل جهش و تقاطع جلوگیری می‌کند. برای این کار ابتدا گراف وظایف را به صورت زیر سطح بندی می‌کنیم.گره‌هایی که یال ورودی ندارند را در سطح اول قرار می‌دهیم. گره‌هایی که مبدا یال ورودی آن‌ها در سطح اول است در سطح دوم قرار می‌گیرند. به همین نحو سطح هر گره با توجه به بیشینه‌ی مبدا ورودی‌های آن به اضافه یک مشخص می‌شود. مثلا در شکل ۳، ۳ وظیفه در سطح اول و در مابقی سطح‌های یک وظیفه وجود دارد. سطح گره شماره ۸ که یک یال ورودی از گره ۷ (سطح ۵) و یک یال ورودی از گره ۶ (سطح ۴) دارد با توجه به سطح این دو گره، بیشینه سطح آن‌ها به اضافه یک (۶ = ۱ + ۵) خواهد بود. مسلما وظیفه موجود در هر سطح نمی‌تواند قبل از وظیفه والد خود که در سطح بالاتر از آن قرار دارد اجرا شود.

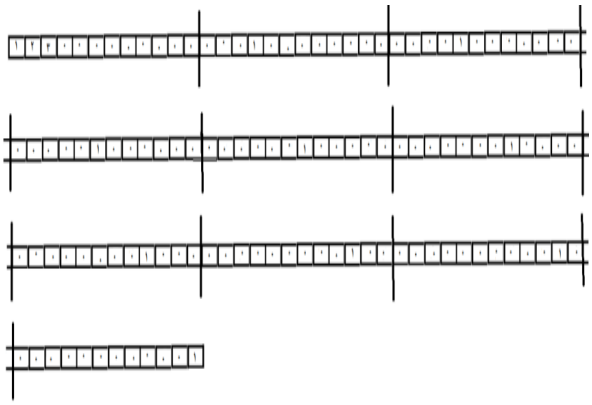


شکل ۲- فلوچارت الگوریتم ژنتیک بهبودیافته پیشنهادی



شکل ۳- سطح‌بندی گراف وظایف VOPD

برای پیاده‌سازی کروموزوم، گراف تغییر یافته را به برداری از اعداد صحیح با طول (تعداد سطح‌ها × تعداد وظایف) تبدیل می‌شود، به نحوی که بردار به تعداد سطح‌ها به زیر بردارهایی با طول تعداد وظایف تقسیم شده است. در هر سطح با توجه به اولویت وظایف، به وظایف موجود در آن، عدد طبیعی از ۱ تا تعداد وظایف آن سطح اختصاص داده می‌شود. با توجه به تعریف کروموزوم، تعداد کروموزوم‌های تولیدی برای زمان‌بندی گراف Vopd، !۳ یعنی ۶ عدد خواهد بود. بقیه کروموزوم‌های تولیدی، فقط در قسمت اول با این کروموزوم تفاوت دارند.



شکل ۴- کروموزوم ایجادی محک VOPD

در این شکل، ۱۲ عنصر اول برای سطح اول، ۱۲ عنصر بعدی برای سطح دوم و به همین ترتیب اختصاص داده شده است. شماره عناصر غیر صفر در هر سطح، شماره وظایف موجود در آن سطح را مشخص می‌کند. مثلا در سطح اول غیر صفر بودن عناصر ۱ و ۲ و ۳ حضور وظایف یک تا سه در سطح اول و عدد اختصاص داده شده به آن‌ها نشان‌دهنده اولویت اجرای وظایف آن سطح خواهد بود. مثلا در این سطح ترتیب ۱۲۳ خواهد بود. تعداد کروموزوم‌ها برای محک Consumer برابر !۲×!۴×!۳×!۲×۱ خواهد بود. یک نمونه از کروموزوم ایجادی برای این محک در شکل ۴ آورده شده است. عدد یک در خانه ۱۶ به این معنا است که در سطح دوم، وظیفه‌ی ۴ دارای اولویت اجرای یک است. به‌همین ترتیب وجود عدد ۳ در خانه ۳۱ به این معنا است که در سطح سوم وظیفه‌ی ۷ دارای اولویت اجرای سه است.

در اینجا نیز از رمزگذاری اعداد حقیقی مانند بالا برای بازنمایی در الگوریتم ژنتیک استفاده شده است. در ابتدا تعدادی کروموزوم به‌صورت تصادفی ایجاد می‌گردد. کروموزوم تولیدی برداری از اعداد صحیح با طول (تعداد سطح‌ها×تعداد وظیفه‌ها) خواهد بود، به نحوی که در ابتدا همه‌ی عناصر با صفر مقداردهی می‌شوند. سپس بردار به تعداد سطح‌ها به بردارهای با طول تعداد وظیفه‌ها تقسیم می‌شوند. هر کدام از این بردارها به‌سطح مربوطه اختصاص داده می‌شود و به‌صورت تصادفی به ژن‌های مربوط به وظیفه‌های موجود در آن سطح، عدد از ۱ تا تعداد وظیفه‌های آن سطح اختصاص می‌یابد. عملگر آمیزش به صورت تقاطع تک نقطه‌ای[۲۸] است و به این صورت پیاده‌سازی می‌شود که ابتدا یک نقطه از مضرب تعداد وظیفه‌ها برای تقاطع دو کروموزوم انتخاب و عمل تقاطع تک نقطه‌ای از آنجا انجام می‌شود. جهش نیز به این صورت پیاده‌سازی شده است که برای این عمل دو نقطه‌ی غیر صفر در یک سطح کروموزوم انتخاب و مقدار آن‌ها با یکدیگر جایگزین می‌گردد.

در آغاز هر نسل، ابتدا تابع هزینه برای هر یک از افراد جمعیت محاسبه و تابع تولید فرزند صدازده می‌شود تا از نسل فعلی به همان تعداد، یک نسل جدید تولید کند. نحوه عملکرد آن است که ابتدا افراد جمعیت جابجا می‌شوند، سپس درصدی از آن‌ها را دو به دو انتخاب کرده و عملیات آمیزش روی آن پیاده می‌شود. پس از آن درصدی از آن‌ها برای عملیات جهش انتخاب می‌گردد. در مرحله بعد میزان تابع برازش جمعیت فرزندان محاسبه می‌شود. سیاست انتخاب نسل جدید به این صورت است که ٪۲۰ والدها و ٪۸۰ فرزندان با استفاده از تابع چرخ رولت برای نسل جدید انتخاب می‌شوند. چرخ رولت براساس مقدار تابع برازش، احتمال انتخاب یک فرد را تعیین می‌کند. خروجی نیز بهترین راه‌حل از نظر تابع برازش در نسل آخر است. تابع برازش کروموزوم‌ها که باید کمینه شود، ترتیبی از وظیفه‌ها را

اخذ و وظیفه فعلی را نگاه می‌کند. اگر در ورودی‌های قبلی، وظیفه والد یا هم‌پردازنده‌ی این وظیفه بود، زمان شروع آن برابر با بیش‌ترین زمان خاتمه‌ی والدهای آن وظیفه به اضافه‌ی زمان انتقال داده از وظیفه والد به وظیفه فرزند و زمان خاتمه‌ی وظیفه‌های هم‌پردازنده آن خواهد بود. در غیر آن (یعنی فقدان وجود وظیفه والد و وظیفه هم‌سطح در میان وظیفه‌های قبلی) زمان شروع آن صفر خواهد بود.

۳-۲- بهبود الگوریتم ژنتیک

ما برای بهبود روش حل با الگوریتم ژنتیک در مرحله جهش، از الگوریتم شبیه‌سازی گداخت بهره گرفتیم تا بهبود کروموزوم بعد از مرحله جهش تضمین گردد (شکل ۲). در این بخش روش پیاده‌سازی تابع جهش عنوان شده است.

نگاشت با متد پیشنهادی: الگوریتم پیشنهادی در این قسمت در تابع جهش و عملگر انتخاب با الگوریتم ژنتیک معرفی شده برای نگاشت تفاوت دارد. در الگوریتم ژنتیک با توابع جهش و تقاطع موجود، امکان افزایش تابع هزینه کروموزوم انتخابی وجود دارد. در این الگوریتم تابع جهش برای تضمین عدم افزایش تابع هزینه با الگوریتم شبیه‌سازی گداخت پیاده‌سازی شده است.

جهش: همسایه‌های کروموزوم موردنظر را پیدا می‌کند که همسایه هر کروموزوم همان کروموزوم‌هایی هستندکه فقط در یک ژن از کروموزوم با هم متفاوت دارند. سپس با استفاده از شبیه‌سازی گداخت در یک حلقه بر روی همه همسایه‌های کروموزوم، آن‌ها را از نظر تابع برازش با وظیفه فعلی مقایسه می‌کند. اگر کروموزوم بهتری بود با آن جایگزین می‌گردد وگرنه با احتمالی جایگزین می‌شود که این احتمال رفته رفته کوچک و به مرور زمان احتمال انتخاب کروموزوم‌های بد کمتر می‌شود.

انتخاب: تعدادی از بهترین کروموزوم‌های نسل قبل، در نسل جدید کپی می‌شوند. روی درصدی از کروموزوم‌های دیگر عمل تقاطع اجرا و دو فرزند تولیدی جایگزین والدین می‌گردند. روی کروموزوم‌های باقیمانده عمل جهش اجرا و کروموزوم تولیدی جایگزین والد خود می‌شود. به این ترتیب جمعیت جدیدی با همان تعداد افراد ایجاد می‌گردد.

زمان‌بندی با متد پیشنهادی: الگوریتم پیشنهادیِ این بخش نیز در تابع جهش و عملگر انتخاب با الگوریتم ژنتیک معرفی شده برای زمان‌بندی تفاوت دارد. در الگوریتم ژنتیک با توابع جهش و تقاطع موجود، امکان افزایش تابع هزینه کروموزوم انتخابی وجود دارد. در این الگوریتم تابع جهش برای تضمین عدم افزایش تابع هزینه با الگوریتم شبیه‌سازی گداخت پیاده‌سازی شده است.

جهش: همسایه‌های کروموزوم مورد نظر را پیدا می‌کند که همسایه هر کروموزوم همان کروموزوم‌هایی هستند که فقط در یک قسمت از کروموزوم، بین ژن‌های $k \times (n+1)$ تا $n \times (k+1)$ کروموزوم، تنها مقدار موجود در دو ژن (اولویت اجرای دو وظیفه) با هم جابجا شده باشد که k یک عدد صحیح از صفر تا یکی کمتر از تعداد سطح‌ها و n تعداد وظایف است. سپس با استفاده از شبیه‌سازی گداخت در یک حلقه بر روی همسایه‌ها، آن‌ها را از نظر تابع برازش با وظیفه فعلی مقایسه می‌کند. اگر بهتر بود با آن جایگزین می‌گردد وگرنه با احتمالی جایگزین آن می‌شود که این احتمال به تدریج کوچک و به مرور زمان احتمال انتخاب کروموزوم‌هایِ بد کمتر می‌شود.

انتخاب: تعدادی از بهترین کروموزوم‌های نسل قبل، در نسل جدید کپی می‌شوند. روی درصدی از کروموزوم‌های دیگر عمل تقاطع اجرا شده و ۲ فرزند تولیدی جایگزین والدین خود می‌گردند. روی کروموزوم‌های باقیمانده عمل جهش اجرا و کروموزوم تولیدی جایگزین والد خود می‌شود. به این ترتیب جمعیت جدیدی با همان تعداد افراد ایجاد می‌گردد.

۳-۳- مدل‌سازی قطعی

مساله‌های در ابعاد و اندازه کوچک می‌تواند با روش قطعی به جواب بهینه دست یابد. روش‌های قطعی به‌طور جامع راه‌حل‌های فضای مساله را کشف می‌کنند و جواب بهتر را ارایه می‌دهند. روش شاخه و کران مثالی از روش‌های قطعی است. برنامه‌ریزی خطی یا بهینه‌سازی خطی، روشی ریاضیاست که به یافتن مقدار کمینه یا بیشینهِ یک تابع خطی روی یک چندضلعی محدب می‌پردازد. این چندضلعی محدب در حقیقت نمایش نموداری تعدادی قید از نوع نامعادله روی متغیرهای تابع است. به‌وسیله برنامه‌سازی خطی می‌توان بهترین نتیجه (مثلاً بیشترین سود یا کمترین هزینه) را در شرایط خاص و با قیود خاص به‌دست آورد [۳۵].

نگاشت با برنامه‌ریزی خطی عدد صحیح: مساله نگاشت وظایف ما بر روی معماری مش دو بعدی با روش برنامه‌ریزی خطی عدد صحیح حل شده است. معادلات مدل‌سازی خطی عدد صحیح و تابع هزینه استفاده شده در این مساله به صورت زیر خواهد بود. هدف در این روش بهینه‌سازی انرژی مصرفی برای اجرای وظایف بر روی پردازنده‌ها است. در معادلات این بخش از متغیرهای زیر استفاده شده است:

m : تعداد وظایف و n تعداد پردازنده‌ها است.
$\mathcal{M}$: عددی خیلی بزرگ در نقش بی‌نهایت است.
$\mathcal{E}_{ij}$: انرژی مصرفی توسط $task_i$ که بر روی پردازنده j اجرا می‌شود.
$\mathcal{F}_{ii'}$: حجم داده مبادله‌ای مابین $task_i$ و $task_{i'}$ است.
a_{ij} : اگر $task_i$ بر روی پردازنده‌ی j قابل اجرا باشد مقدار a_{ij} برابر یک و در غیر این صورت برابر صفر است.
$d_{jj'}$: فاصله مابین پردازنده j و j' که طبق رابطه (۱) به روش منهتن محاسبه می‌شود.

$$d_{jj'} = abs(x_{j'} - x_j) + abs(y_{j'} - y_j) \qquad (۱)$$

هدف این مدل کمینه‌سازی انرژی مصرفی است. انرژی مصرفی شامل دو بخش یعنی انرژی مصرفی اجرای $task_i$ در PE_j و انرژی مصرفی تبادل داده‌ها مابین عناصر پردازشی مختلف (در رابطه (۲)) است. بخش اول آن مجموع انرژی مصرفی هر وظیفه بر روی پردازنده انتخابی برای اجرای آن و بخش دوم انرژی مصرفی مربوط به ارتباطات است. ξ انرژی مصرفی جهت انتقال یک واحد داده در واحد مسافت را نشان می‌دهد.

$$Z = \sum_{i=1}^{n}\sum_{j=1}^{m} \mathcal{E}_{ij} \cdot \mathcal{X}_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{i'=1}^{n}\sum_{j'=1}^{m} \xi \cdot \mathcal{F}_{ii'} \cdot d_{jj'} \cdot r_{iji'j'} \qquad (۲)$$

با توجه به روابط (۳) تا (۵)، $\mathcal{X}_{ij}$ وقتی برابر یک است که پردازنده‌ی j برای $task_i$ انتخاب شده باشد.

$$\sum_{j=1}^{n} \mathcal{X}_{ij} = 1 \qquad (۳)$$

$$\mathcal{X}_{ij} \le a_{ij} \qquad \forall i \in \{1, \dots, m\}; \forall j \in \{1, \dots, n\} \qquad (۴)$$

$$\mathcal{X}_{ij} \in \{0,1\} \qquad \forall i \in \{1, \dots, m\}; \forall j \in \{1, \dots, n\} \qquad (۵)$$

$r_{iji'j'}$ یک متغیر باینری است و با توجه به وجود یال از $task_i$ به $task_{i'}$ به آن مقداردهی می‌شود. اگر $task_i$ در پردازنده j و i' در پردازنده j' اجرا شوند لذا $r_{iji'j'}$ برابر با یک است. در غیر این صورت اگر حداقل یکی از شرایط بالا صدق نکند این مقدار برابر با صفر خواهد بود.

$$r_{iji'j'} \le \mathcal{X}_{ij} \qquad (۶)$$

$$r_{iji'j'} \le x_{i'j'} \quad (۷)$$

$$r_{iji'j'} \ge x_{ij} + x_{i'j'} - 1 \quad (۸)$$

$$r_{iji'j'} \in \{0,1\} \qquad \forall i,i' \in \{1,\dots,m\}; \forall j,j' \in \{1,\dots,n\} \quad (۹)$$

زمان‌بندی با مدل‌سازی تقلیل جایگشت‌های روش قطعی: در نگاشت قطعی، مجموعه جواب‌ها به صورت آرایه‌هایی با طول تعداد وظایف می‌باشند که خانه‌های آن‌ها با شماره پردازنده‌ی مورد استفاده برای آن وظیفه تخصیص داده شده‌اند. همه جایگشت‌های ممکن برای مساله برابر با حاصل‌ضرب تعداد پردازنده‌های مورد استفاده برای کلیه وظایف است. مثلا برای محک Jpeg که از ۸ وظیفه موجود در گراف وظایف، ۲ وظیفه بر روی ۴ پردازنده مش دوبعدی[29] ۲×۲ و مابقی بر روی ۳ پردازنده قابل اجرا هستند، تعداد کل جایگشت‌هایی که باید بررسی شوند برابر با ۱۱۶۶۴ = ۴×۳×۳×۳×۳×۳×۳×۳×۴ است که برای همه آن‌ها تابع هزینه محاسبه شده و جایگشتی که کم‌ترین مقدار انرژی را برای نگاشت به‌دست می‌دهد به‌عنوان جواب مساله انتخاب می‌گردد.

در حل مساله زمان‌بندی به روش قطعی مجموعه جواب‌ها به‌صورت آرایه‌هایی با طول تعداد وظایف هستند که خانه‌های آن‌ها با اولویت اجرای آن وظیفه، از یک تا تعداد وظایف تخصیص داده می‌شود. همه جایگشت‌های ممکن برای مساله برابر با فاکتوریل تعداد وظایف است. مثلا برای محک Jpeg که ۸ وظیفه در گراف وظایف آن موجود باشد، تعداد کل جایگشت‌هایی که باید بررسی شوند برابر با ۴۰۳۲۰=۱×۲×۳×۴×۵×۶×۷×۸=!۸ است که برای همه آن‌ها تابع هزینه محاسبه شده و جایگشتی که کم‌ترین مقدار زمان اجرا را در زمان‌بندی (کمترین زمان اجرای آخرین وظیفه) به دست می‌دهد به عنوان جواب مساله انتخاب می‌گردد.

در اینجا برای پیاده‌سازی زمان‌بندی قطعی جهت حذف یکسری جایگشت‌های غیرممکن مساله و با هدف کاهش جستجو، گراف مانند شکل ۳ سطح‌بندی می‌شود. از آنجا که هر وظیفه نمی‌تواند پیش از وظیفه‌ی والد خود اجرا شود، از ابتدا راه‌حل‌هایی که عدد اختصاصی به وظیفه فرزند را در زمان‌بندی کوچکتر از عدد اختصاص یافته به وظیفه والد قرار می‌دهند، حذف می‌شود. مجموعه جواب‌ها به‌صورت آرایه‌هایی با طول تعداد (سطح‌ها × تعداد وظیفه‌ها) است که به تعداد سطح‌ها به بردارهای با طول تعداد وظیفه‌ها تقسیم شده است. هر کدام از این بردارها به‌سطح مربوطه اختصاص یافته و به خانه‌های مربوط به وظیفه‌های موجود در آن سطح، عدد صحیح از ۱ تا تعداد وظیفه‌های آن سطح (اولویت اجرای وظایف موجود در سطح) اختصاص داده می‌شود.

حال برای آن که یک سری جایگشت‌های امکان‌پذیر که با این کار حذف شده‌اند را نیز در محاسبات بیاوریم، گراف را با همه سطح‌بندی‌های ممکن دیگر با در نظر گرفتن مسیر بحرانی و عدم افزایش طول آن مسیر نیز حل می‌کنیم. با حل مساله تعداد جایگشت‌ها برای محک Jpeg از ۴۰۳۲۰ به ۶=!۱×!۱×!۱×!۳×!۱×!۱ خواهد رسید که این عدد با توجه به تعداد وظایف موجود در سطوح مختلف گراف سطح‌بندی شده است. مثلا در این محک تعداد وظایف موجود در سطوح اول تا پنجم آن به ترتیب برابر ۱، ۱، ۱، ۳، ۱ است که در این صورت حل همه حالت‌ها مدت زمان بسیار کمی نیاز دارد و به راحتی قابل اجرا است. این محک فقط یک سطح‌بندی خواهد داشت. همین طور محک Mpeg4 نیز دارای یک سطح‌بندی است و تعداد جایگشت‌های آن از !۱۲ به !۹×!۳ کاهش می‌یابد. محک Office automation نیز دارای ۲ سطح‌بندی و بهبود تعداد جایگشت‌ها از !۵ به ۲(!۲) است. این مقادیر برای محک‌های MWD و Networking و Consumer و VOPD به ترتیب برابر با !۱۲، !۱۳، !۱۲، !۱۲و مقادیر جدید برابر (!۲×!۲×!۲×!۱×!۱×!۲)+۱۰(!۳)،۴(!۲×!۲×!۲×!۲×!۲)،۴(!۳×!۳×!۳×!۴)، ۲(!۲×!۳×!۲×!۳×!۲)+۸(!۱×!۲×!۳×!۴×!۲)+۳(!۲×!۲×!۲×!۴×!۲) و ۴۰(!۳) است که این تعداد نسبت به تعداد قبلی بسیار کمتر شده‌اند.

۴- نگاشت و زمان‌بندی وظایف و ارتباطات

در این بخش با تعریف مساله و ارایه مدل‌های جریان و فرکانس، معماری سیستم و کاربرد، به بیان مساله می‌پردازیم. برای هر قالب خاص با اطلاعات تعیین شده، گراف جریان وظیفه‌های کاربرد، و پارامترهای مدل، یک مجموعه تخصیص و زمان‌بندی تعیین می‌شود. این روش به‌طور همزمان وظایف و ارتباطات را مدل و توان مصرفی تراشه را کمینه می‌کند. فرض می‌کنیم که نتایج زمان‌بندی حاصل ما یک زمان‌بندی بلاانقطاعی[30] است. در شکل ۵ ساختار مساله تخصیص و زمان‌بندی انرژی- آگاه وظیفه‌ها برای سیستم بر تراشه ارایه شده است. این مثال دارای مدل کاربرد در قالب گراف وظایف، مدل معماری، و زمان کامل شدن چند وظیفه است.

۴-۱- مدل جریان و فرکانس

تغییرات پارامترهای کلیدی ترانزیستور روی تغییرات فرکانس و توان نشتی تراشه اثر دارد. برای دما و V_{dd} ثابت، فرکانس و توان نشتی زیرآستانه هر هسته بر تراشه می‌تواند بوسیله V_{th} و L_{eff} تقریب زده شود.

مدل فرکانس: در این کار، ما از مدل فرکانس هسته ارائه شده در [۴۸] استفاده می‌کنیم. تاخیر دروازه می‌تواند با توجه به V_{th}، بوسیله رابطه (۱۰) مدل شود.

$$T_g \propto \frac{1 + V_{th}/V_{th0}}{(V - V_{th})^{\alpha}} \quad (۱۰)$$

جایی که $\alpha = 1.3$ است. مقدار V_{th} دارای توزیع نرمال است. برای مقدار قابل قبول V_{th}، رابطه (۱۰) در بازه مورد نظر تقریبا خطی است لذا T_g بوسیله توزیع نرمال تقریب‌زده می‌شود. فرکانس کار هر هسته بستگی به تاخیر مسیرهای بحرانی[31] T_{cp} دارد. در پردازنده‌های جدید هزاران مسیر بحرانی وجود دارد [۳۷]. احتمال تاخیر طولانی‌ترین مسیر بحرانی تراشه ($\max T_{cp}$) در [۳۷] محاسبه شده است. لذا عکس تاخیر طولانی‌ترین مسیر می‌تواند به‌عنوان فرکانس هسته و به‌صورت رابطه (۱۱) بیان شود.

$$f = \frac{1}{delay} = \frac{1}{\max T_{cp}} \quad (۱۱)$$

مدل جریان نشتی: ما برای مدل کردن توان نشتی از مدل ارایه شده مقاله [۴۷] استفاده می‌کنیم. تفاوت این مدل با مدل مقاله [۳۶] آن است که از جریان نشتی تولیدی توسط اثر DIBL صرف نظر شده است. از این جریان صرف نظر شده زیرا در تکنولوژی‌های آتی، منبع اصلی جریان نشتی همان جریان نشتی زیر آستانه است. براساس این فرضیات رابطه (۱۲) را برای جریان نشتی داریم.

$$I_{leakage} \propto v_t^2 e^{\frac{(V_{off} - V_{th})}{\eta v_t}} \quad (۱۲)$$

جایی که $v_t = k\frac{T}{q}$ همان ولتاژ گرمایی (k ثابت بولتزمن، T دمای تراشه، و q شارژ الکترونی است) و $V_{th} = Tc_1 + c_2$ ولتاژ آستانه (c_1 فاکتور دمایی ولتاژ آستانه است)، η ضریب سوئینگ آستانه، و V_{off} یک پارامتر تجربی است. پارامترهای

c_1, c_2, η, V_{off} باید بوسیله برازش منحنی رابطه (۱۲) برای داده‌های استخراجی در شبیه‌ساز SPICE تعیین شود. به این منظور پارامترهای فوق با استفاده از مدل‌های کتابخانه‌ای قابل پیش‌بینی BSIM4 مربوط به روش دروازه فلزی hi-K تحت تکنولوژی ۹۵ نانومتر تعیین شده‌اند. برای تقریب اثر تغییر V_{th} روی کل توان نشتی، توزیع V_{th} برای تمامی ترانزیستورهای تراشه در رابطه (۱۲) جایگزین می‌شود لذا تابع توزیع توان مصرفی نشتی لاگ نرمال[۳۲] است.

مدل توان: براساس توزیع V_{th}، و آستانه همبسته وابسته به تکنولوژی ϕ، توزیع مورد نیاز فرکانس و توان مصرفی نشتی را استخراج می‌شود. حداقل میزان فرکانس، به‌عنوان فرکانس سیستم تعیین می‌شود. برای محاسبه توان‌نشتی از نرخ توان‌نشتی تحت تغییرات جریان نشتی بدون تغییر استفاده می‌کنیم. توان نشتی و ولتاژ آستانه V_{th} هر رخداد، از رابطه (۱۳) بدست می‌آید.

$$P_{leak}^{event} / P_{leak0}^{event} = e^{\frac{(V_{th0} - V_{th}^{event})}{\eta v_t}} \qquad (۱۳)$$

جایی که P_{leak}^{event} میزان توان نشتی برای ولتاژ آستانه V_{th}، و P_{leak}^{event} توان نشتی برای مقدار اسمی V_{th} است. جریان نشتی هر رخداد معادل میانگین مقدار توان نشتی نقطه شروع و پایان است. مثلا اگر توان نشتی نقطه شروع و پایان حالت انتخابی برابر a و b باشد لذا جریان نشتی این حالت معادل $c = \frac{(a+b)}{2}$ است.

۴-۲- مدل معماری و کاربرد

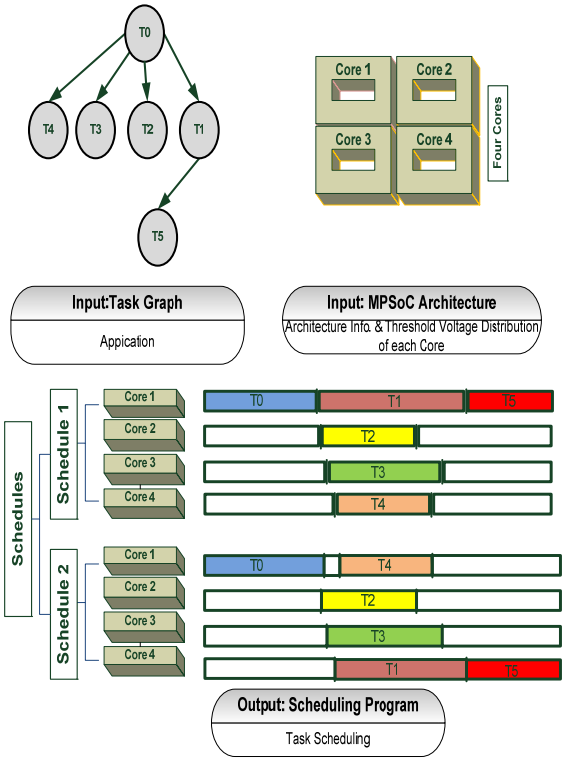
به‌طورکلی سیستم‌های نهفته دارای هسته‌های ناهمگن هستند. این سیستم‌ها از مجموعه‌ای از هسته‌ها $C=\{c_i : 1 \le i \le m\}$ تشکیل شده‌اند. توان پویا و نشتی هر هسته $\text{core}(c_i)$ وقتی که وظیفه t_j را اجرا می‌کند بوسیله P_{ij}^{dyn} و P_{ij}^{leak} ارائه می‌شود. شکل ۶ ارائه‌کننده یک مدل معماری نمونه با هم‌بندی توری[۳۳] است.

مدل کاربرد بوسیله یک گراف مستقیم غیرمدور DAG[۳۴] به‌صورت G(N,E) مدل می‌شود جایی که N مجموعه‌ای از وظیفه‌هایی است که متعلق به آن کاربرد است. همچنین E مبین مجموعه کمان‌هایی است که ارایه‌کننده وابستگی‌های داده و کنترل بین وظیفه‌های آن کاربرد هستند (شکل ۷). نماد وظیفه‌ها در گراف به صورت دایره ارایه می‌شود و هر وظیفه دارای سه نوع منبع[۳۵] (بدون کمان ورودی)، چاهک[۳۶] (بدون کمان خروجی)، و معمولی[۳۷] (دارای ورودی و خروجی) است. در این مدل، زمان رهاسازی و ضرب‌الاجل هر وظیفه t_i به‌وسیله r_i و d_i ارایه می‌شود. اگر وظیفه‌ای در محک دارای ضرب‌الاجل نباشد مقدار آن بی‌نهایت فرض می‌شود. برای هر وظیفه t_i، زمان اجرای بدترین حالت زمانی که روی هسته c_i اجرا می‌شود بوسیله et_{ij} ارایه می‌گردد. شکل ۷ یک نمونه گراف DAG و ضرب‌الاجل‌های مربوطه آن را نشان می‌دهد.

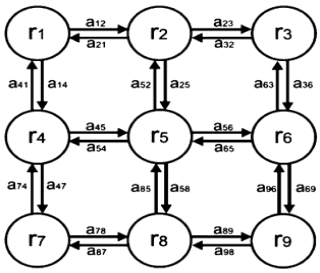
۴-۳- نگاشت و زمان‌بندی

ما در این بخش به شرح مساله تخصیص و زمان‌بندی می‌پردازیم. روال الگوریتم پیشنهادی در الگوریتم ۱ بیان شده است. این الگوریتم به پنج مرحله تقسیم شده است. در مرحله اول (سطح‌بندی گراف) که بر اساس یال‌های ورودی گراف مرجع ایجاد می‌شود. در مرحله دوم (تولید کروموزوم‌ها) که در ابتدا به‌صورت تصادفی ایجاد می‌شود و سپس بر اساس ورودی‌ها، بهبود می‌یابند. سپس در مرحله سوم (گزینش وظایف) با یک روش الویتی، n وظیفه از یک سطح وظایف برگزیده می‌شود. این روش از جهش روی کروموزوم‌های باقی‌مانده به عنوان جایگزین والد آن کروموزوم بهره می‌برد. در مرحله چهارم (زمان‌بندی)، الگوریتم نگاشت و زمان‌بندی برای کروموزوم‌های انتخابی، اجرا می‌شود. تمامی تخصیص‌ها و زمان‌بندی‌های تولیدی این مرحله برای تمامی کروموزوم‌ها اجرا می‌شود. کل اطلاعات لازم و موردنیاز تخصیص و زمان‌بندی مانند برآوردن قیود کارایی و مقدار انرژی کمینه در زمان اجرای کاربرد در مرحله پنجم (محاسبات)، محاسبه می‌شود.

ما براساس قیود نواحی انرژیِ مصرفی برای محاسبات و مبادله داده‌های ارتباطی، k زمان‌بندی را از میان n زمان‌بندی نامزد انتخاب می‌کنیم. این عمل در مرحله ششم (گزینش زمان‌بندی) انجام می‌شود. پس از فرآیند تقلیل، نگاشت فرکانس واقعی تراشه بوسیله تکنیک‌های الحاق سرعت بالغ‌شده[۳۸] فراهم می‌شود [۶]. زمان‌بند[۳۹] براساس این اطلاعات، از میان زمان‌بندی‌های انتخابی، یک زمان‌بندی مناسب را بر می‌گزینند.

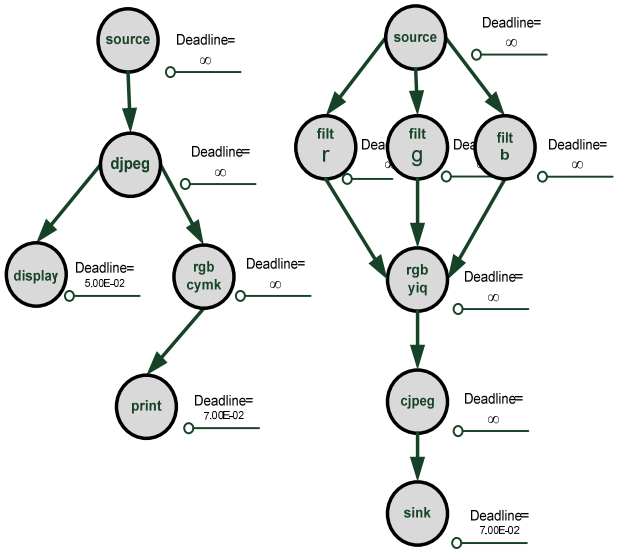


شکل ۵- تعریف مساله تخصیص و زمان‌بندی همزمان وظایف و ارتباطات



شکل ۶- مدل معماری توری شبکه پردازنده و ارتباطات نمونه ۳×۳

در این مرحله، الگوریتم تعدادی زمان‌بندی را به‌عنوان زمان‌بندی نامزد تولید می‌کند. ما برای تولید این نامزدها، یک الگوریتم تخصیص و زمان‌بندی انرژی‌آگاه را مورد استفاده قرار می‌دهیم. برای پشتیبانی از سیستم‌های دوره‌ای براساس روش تحقیقی مقاله [۴۰]، اصلاحات و تغییرات لازم را ایجاد و برای ابردوره[۴۰] وظیفه‌ها، مساله تخصیص و زمان‌بندی را حل کرده‌ایم (خط ۱۳ در الگوریتم ۱). این مساله بطور کامل در الگوریتم ۱ نشان داده شده است.



شکل ۷- نمونه گراف DAG از محک Consumer در مجموعه E3S [۳۹]

الگوریتم ما یک الگوریتم مکاشفه‌ای است که برای عملیات تخصیص و زمان‌بندی از ایده بهبود کروکوزوم استفاده می‌کند. در ایده، تقلیل زمان و تعداد جستجوی برای تولید کروموزوم و جمعیت تولیدی مساله ژنتیک یک معیار اندازه‌گیری مهم است. سطح وظیفه i برابر با اختلاف بین سطوح آن وظیفه است که تعریف می‌شود. این معیار به عنوان یک شاخص الویت وظیفه i در طول مدت زمان عملیات تخصیص و زمان‌بندی بکار می‌رود. هر چه مقدار این معیار وظیفه کمتر باشد، آن وظیفه به الویت نزدیکتر است.

در وظیفه i زودترین زمان شروع EST[۴۱] و دیرترین زمان شروع LST[۴۲] آن وظیفه است. به این دلایل در هر تکرار، وظیفه‌های آماده براساس سطح‌شان مرتب می‌شوند. یک وظیفه آماده وظیفه‌ای است که قبل از زمان تصمیم‌گیری تمامی وظیفه‌های اجدادش خاتمه یافته باشد. همچنین بدیهی است که در زمان تصمیم‌گیری، یک وظیفه آماده وارد سیستم شده است. سپس الگوریتم، یک هسته برای اجرای آن وظیفه آماده را بر می‌گزیند. هسته مورد نظر برای اجرای این وظیفه نیازمند چندین شرط است:

۱- این هسته باید توانایی اجرای آن وظیفه را داشته باشد (خط ۱۰ الگوریتم ۱).

۲- هسته موردنظر در زمان زمان‌بندی باید آزاد باشد (خط ۱۱ الگوریتم ۱).

۳- ضمناً باید برای اجرای آن وظیفه باندازه کافی سریع باشد تا ضرب‌الاجل آن را بر آورده سازد (خط ۱۳ الگوریتم ۱).

۴- اگر هسته‌ای باتوانایی انجام قیود زمان‌بندی یافت نشد، الگوریتم خاتمه و غیرممکن بودن این زمان‌بندی را گزارش می‌دهد (خطوط ۲۹ الی ۳۰ الگوریتم ۱).

در حالی که آن وظیفه در حال اجرا روی هسته مورد نظر است باید تمام قیود انرژیِ آن برآورده شود (خط ۱۶ الگوریتم ۱). اگر این قیود برآورده نشوند، الگوریتم برای رفع این مشکل از تکنیک درج تاخیر استفاده می‌کند (خط ۲۳ الگوریتم ۱). شرح درج تاخیر در مقاله [۴۱] آمده است. نهایتاً این الگوریتم یک هسته مناسب از بین هسته‌های نامزد انتخاب می‌کند که بوسیله آن تقریب زمان پایان آن وظیفه و وظیفه فرزندش کمینه می‌شود (خط ۱۹ الگوریتم ۱). وظیفه فرزند، فرزندی از آن وظیفه در DAG آن کاربرد است که سطح بیشتری از آن وظیفه را دارد (خط ۱۷ الگوریتم ۱). انتخاب هسته براساس سریعترین زمان اجرای آن وظیفه و وظیفه فرزند بحرانی، در بیشینه کردن پویایی جانشینان آن وظیفه بسیار مهم و اساسی است.

باید توجه شود که ما برای بیان زمان اجرای وظیفه فرزند در گام انتخاب هسته، در تابع درج تاخیر (DelayInsertion) تغییرات موردنیاز را ایجاد کرده و آن را بهبود داده‌ایم. اگر وظیفه فرزند قابلیت اجرا روی هسته نامزد را نداشته باشد، ما زمان $2 \times hyperperiod$ را به عنوان زمان پایان وظیفه آن فرزند استفاده می‌کنیم. سپس تمامی هسته‌هایی که قبل از نقطه زمان‌بندی بعدی بیکار می‌شوند را به‌عنوان هسته آزاد علامت‌گذاری می‌کند (خط ۳۷ الگوریتم ۱). برای کامل شدن کارها در هر تکرار، مقادیر EST وظیفه‌های زمان‌بندی شده را به‌روزرسانی و تجدید می‌کند (خط ۳۹ الگوریتم ۱).

در نهایت ما در این مرحله از نتایج تخصیص‌ها و زمان‌بندی‌های مرحله قبل استفاده کرده و تعداد k عدد تخصیص و زمان‌بندی از از بین تمامی n تخصیص و زمان‌بندی را انتخاب می‌کنیم. این انتخاب براساس میزان دمای کمینه محاسبه شده برای تخصیص‌ها و زمان‌بندی‌ها صورت می‌گیرد. این انتخاب باید با وجود تقلیل جستجو از نظر توان مصرفی کمینه باشد در حالی که در همان زمان مقدار امید ریاضی دمای تراشه را نیز کمینه می‌کند. ما $F(n,k)$ را برای فرموله کردن تعریف این معیار بکار می‌بریم. $F(n,k)$ را به‌عنوان کمینه مقدار مورد انتظار دمای k عدد تخصیص و زمان‌بندی از بین تمامی n تخصیص‌ها و زمان‌بندی‌ها قرار می‌دهیم.

الگوریتم ۱- الگوریتم تخصیص و زمان‌بندی انرژی‌آگاه پیشنهادی

```
1  compute EST(j), LST(j) and level(j)  for all tasks;
2  compute  avgE; // Average of execution time
3  CurrentTime = 0;
4  while there are Unscheduled tasks do
5        CurrentDelay = 0;
6        RT = ready tasks in non-decreasing order of level;
7      for each  j∈ RT  do
8        initial invalidCount and fastest Core info variables;
9        for each  m∈ M  do
10         if proc m can execute task  j then
11          if core  m at CurrentTime is free then
12               calculate end time of task_j on proc m;
13               if deadline constraints of task_j is satisfied and
                                        isWrappAroundValid( ) then
14                    compute energy profile for one Hyper period;
15                    calculate peak temperature for all cores;
16                  if energy constraints is satisfied then
17                       cchild  = find child of task_j;
18                       calculate end time cchild;
19                  if endTime[j]+endTime[cchild] is minimum then
20                          update core for task_j and its child;
21                          CurrentDelay = 0;
22       else
23        CurrentDelay=DelayInsertion(G(V,E), j, m, currentTime);
24                       update core for task_j;
25               else if core m is not busy then
26                                    InvalidCount++;
27          else if core m is not busy then
28                                    InvalidCount++;
29       if InvalidCount = | M | then
30                                  return  INFEASIBLE;
31       else if core is valid then
32              assign task_j on best core ;
33              best core is busy for end time of task_j;
34       if CurrentDelay> o then
35    break; //allow no tasks to start executing between currentTime
              // and currentTime + currentDelay
36    calculate nextSchedulePoint;
37    set idle all cores that become idle at NextSchedulePoint;
38    CurrentTime  =NextSchedulePoint;
39    update EST(j), level(j), for all unscheduled tasks;
40   return  FEASIBLE;
```

الگوریتم ۲- الگوریتم انتخاب زمان‌بندی نهایی

```
1  for i= 0 to nSchedule do
2      for j= 0 to nSelect do
3          if j= 0 then
4              for l= 0 to k do
5                  dpVec[i][j].meet [l]=0;
6                  dpVec[i][j].maxTemp[l]=L_max;
7          else if i= 0 then
8              for o= 0 to k do
9                  dpVec[i][j].meet[o]=meet[i][o];
10                 dpVec[i][j].maxTemp[o]=maxTSchadule[i][o];
11         else
12             // Calculate candidate Γ i-1 j-1
13             for l= 0 to k do
14                 calculateΓ i-1 j-1.meet[l];
15                 calculateΓ i-1 j-1.maxTemp[l];
16             F i-1 j  = calculateexpected peak-temp dpVec[i-1][j];
17             F i-1 j-1= calculateexpected peak-temp Γ i-1 j-1;
18             if F i-1 j-1≤ F i-1 j then
19                 dpVec[i][j]=Γ i-1 j-1;
20                 mark i_th schedule;
21             else dpVec[i][j]=dpVec[i-1][j];
```

جدول ۲- محک‌های داده‌ای و ابعاد شبکه مورد آزمایش

Benchmark	Type	Task No.	Edge No.
Auto- 3×3	Homogenus	24	21
Consumer-3×3	Hetrogenus	12	12
Networking-3×3	Homogenus	13	9
Office- 2×2	Homogenus	5	5
Mpeg4- 3×3	Homogenus	12	13
MWD- 3×3	Hetrogenus	12	12
Jpeg- 2×2	Homogenus	8	9
vopd- 3×3	Hetrogenus	12	14

جدول ۳- پردازنده‌های انتخابی حل مساله پیشنهادی

Index	Core
1	AMD K6-2E 400- MHZ/ACR
2	AMD K6-2E+ 500- MHZ/ACR
3	AMD K6-IIIE+ 550- MHZ/ACR
4	IBM PowerPC 405GP- 266 MHZ
5	Motorola MPC555- 40 MHZ

این تعریف واضح است زیرا بعضی از زمان‌بندی‌ها که توانسته‌اند قیود زمانی را برآورده کنند مقدار دمای آنها معتبر نیست. برای این موارد، ما از $L_{\max}$ به جای مقدار دما استفاده می‌کنیم که این پارامتر یک عدد ثابت است که با تنظیم مقدار آن می‌توان یک کران بالا برای حداکثر دمای تراشه در زمان اجرای کاربرد اعمال کرد. بر این اساس ما برای هر برنامه‌ریزی i یک پارامتر اندازگیری دما $T_{\max}^{sel}(i)$ به صورت رابطه (۱۴) تعریف می‌کنیم.

$$T_{\max}^{sel}(i) = \min_{s \in S_{sel}} \left[\max(V_{ins} \times T_{is}^{\max} \ , \ (1 - V_{ins}) \times L_{\max}) \right] \qquad (۱۴)$$

جایی که S_{sel} مجموعه زمان‌بندی‌های انتخابی، V_{is} موید این است که زمان‌بندی s^{th} می‌تواند برنامه i^{th} قیود زمان‌بندی را برآورده کند، و $T_{is}^{\max}$ حداکثر دمای اجرای زمان‌بندی تحت شرایط برنامه i^{th} (درصورت برآوردن قیود زمانی) است. پارامترهای V_{is} و $T_{is}^{\max}$ خروجی‌های مرحله قبل هستند. براساس تعریف $T_{\max}^{sel}$، ما می‌توانیم مقدار $F(n,k)$ را بوسیله رابطه (۱۵) بیان کنیم.

$$F(n,k) = \sum_{i=1}^{n_{event}} \rho(i) \times T_{\max}^{sel}(\mathrm{i}) \qquad (۱۵)$$

جایی که $\rho(i)$ همان احتمال وقوع برنامه i است و در مرحله اول محاسبه شده است. این مساله یک نمونه از مساله کوله‌پشتی است. لذا برای حل آن الگوریتمی بر پایه روش برنامه‌ریزی پویا ارایه می‌کنیم. برای انتخاب k زمانبندی از بین n زمان‌بندی نماد $F(n,k)$ نمایش دهنده آن است که $F(n,k)$ کمینه (یعنی مبین جواب مساله) است. به‌عبارت دیگر، $F(n,k)$ می‌تواند بوسیله رابطه بازگشتی رابطه (۱۶) ارایه شود.

$$F(n,k) = \min\{F(n-1,k), union(F(n-1,k-1),n)\} \qquad (۱۶)$$

فرض کنید که هر کدام از زمان‌بندی‌ها از ۱تا n شاخص‌بندی شود. آیا عنصر n^{th} در مجموعه انتخابی می‌تواند باشد. اگر به آن مجموعه تعلق نداشته باشد، $F(n,k)$ معادل $F(n-1,k)$ است. اگر عنصر n^{th} به آن مجموعه تعلق دارد، $T_{is}^{\max}$ باید تجدید و به‌روز شود. مقادیر $T_{is}^{\max}$ یا از زمانبندی n^{th} حاصل شده یا از (k-1) زمان‌بندی دیگر که در قبلا برگزیده شده‌اند و مقادیر آن در $F(n-1,k-1)$ بیان شده است. عمل union رابطه (۱۶) عملیات به‌روزرسانی مقادیر $T_{is}^{\max}$ را انجام می‌دهد. الگوریتم ۲ با زمان چندجمله‌ای مبین شبه کد آن است. مقدار اولیه $F(i,0)$ با $L_{\max}$ تنظیم می‌شود (خط ۳ الی ۶ الگوریتم ۲) و مقدار اولیه $F(0,j)$ با حداکثر دمای تولیدی در مرحله چهارم تنظیم می‌شود (خط ۷ الی ۱۰ الگوریتم ۲). تمامی مقادیر $F(n,k)$ با استفاده از روش پایین به بالا محاسبه می‌شود. اگر زمان‌بندی i^{th} به مجموعه انتخابی متعلق باشد، زمان‌بندی i^{th} علامت زده می‌شود (خط ۲۰ الگوریتم ۲). زمان‌بندی علامت زده شده همان نتیجه نهایی متد پیشنهادی است.

۵- آزمایشات تجربی

در این بخش به بررسی کارایی روش و الگوریتم پیشنهادی می‌پردازیم و نتایج شبیه‌سازی را در سه بخش تنظیمات آزمایشات، اعتبارسنجی مدل، و نتایج آزمایشات تجربی، ارزیابی و تحلیل می‌کنیم.

۵-۱- تنظیمات آزمایشات

الگوریتم زمان‌بندی انرژی‌آگاه ارائه شده در زبان C++ پیاده‌سازی و بوسیله محک‌های مختلف E3S[۴۳] [۳۹] در تکنولوژی ۹۵ نانومتر با توجه به پیش‌بینی‌های ITRS [۱۰] و کارهای موجود مورد ارزیابی قرارگرفته است. ما مجموعه داده‌ای E3S را بر اساس فاکتور مقیاس‌بندی تکنولوژی[۴۴] تغییر داده‌ایم. به این منظور اطلاعاتی مانند توان‌مصرفی، ابعاد عناصر و غیره را براساس فاکتورهای مقیاس‌بندی گزارش‌شده در [۴۲] [۴۳] تطابق داده‌ایم. E3S دارای چندین محک است که هرکدام ارایه‌کننده یک کاربرد خاص هستند. بعضی وظیفه‌ها در هر کاربرد دارای ضرب‌الاجل بی‌درنگ سخت هستند. مشابه [۴۴]، ما از ساختار معماری 2×2 که در جدول ۲ برای ساختارهای همگن و ناهمگن استفاده کرده‌ایم. اسامی پردازنده‌های استفاده شده در جدول ۳ آمده است. البته با توجه به‌محدودیت شدید زمانی محک Auto، مقدار تمامی ورودی‌های زمانی (ضرب‌الاجل و دوره) ۱.۵ برابر و پارامترهای مدل جریان نشتی را برای تکنولوژی مورداستفاده استخراج کرده‌ایم. برای بهینه‌سازی از ابزار CPLEX [45] و برای دما از HotSpot [46] بهره

برده‌ایم. ضمناً برای مقایسه نتایج حاصل، دو روش الگوریتم ژنتیک سنتی و پیشنهادی، یک روش شبیه‌سازی گداخت و یک روش قطعی برنامه‌ریزی عدد صحیح را پیاده‌سازی کرده‌ایم.

۵-۲- اعتبارسنجی مدل پیشنهادی

مساله را در ابعاد کوچک با روش قطعی برنامه‌ریزی خطی حل و با مقایسه نتایج اعتبار سنجی کرده‌ایم که به جواب دقیق بهینه دست می‌یابد. در ابعاد بزرگ به استناد جواب‌های یافت شده موجود بعضی مقالات و پاسخ‌های شبیه‌سازی گداخت، اطمینان از جواب مطلوب نزدیک بهینه حاصل شده است.

۵-۳- نتایج آزمایش‌های تجربی

نتایج آزمایش‌ها را از نظر الگوریتم نگاشت، الگوریتم زمان‌بندی و ارزیابی متدلوژی پیشنهادی برای زمان اجرا، سرعت همگرایی، و توان مصرفی با سایر روش‌ها ارزیابی می‌کنیم. برای ارزیابی الگوریتم‌ها، آن‌ها را با ۱۰۰ تکرار و چندین نسل تولیدی، با محک‌های مختلف اجرا کرده‌ایم.

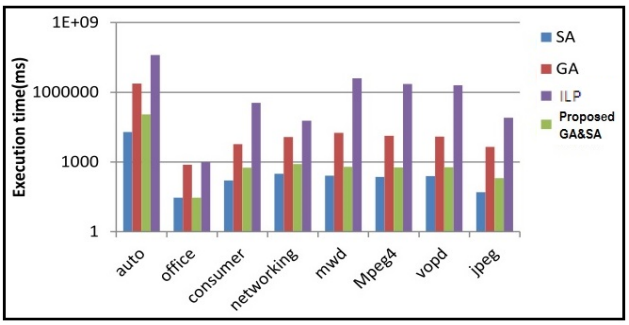
۵-۳-۱- ارزیابی الگوریتم نگاشت

زمان اجرای مرحله نگاشت الگوریتم پیشنهادی از الگوریتم ژنتیک کمتر و از الگوریتم شبیه‌سازی گداخت بیشتر است (شکل ۸). با افزایش تعداد وظایف، این اختلاف بسیار بیش‌تر می‌شود. البته احتمال‌گیر افتادن الگوریتم شبیه‌سازی گداخت در بهینه محلی زیاد و در صورت تکرار الگوریتم نیز مقادیر پاسخ تقریبی به دست آمده بهبود نخواهند یافت (جدول ۴). انرژی مصرفی مرحله نگاشت الگوریتم پیشنهادی از الگوریتم ژنتیک و شبیه‌سازی گداخت کمتر و از الگوریتم برنامه‌ریزی خطی بیشتر است (شکل ۹). در الگوریتم شبیه‌سازی گداخت احتمال گیرافتادن در بهینه محلی زیاد می‌باشد که این مورد در نمودار مشهود است.

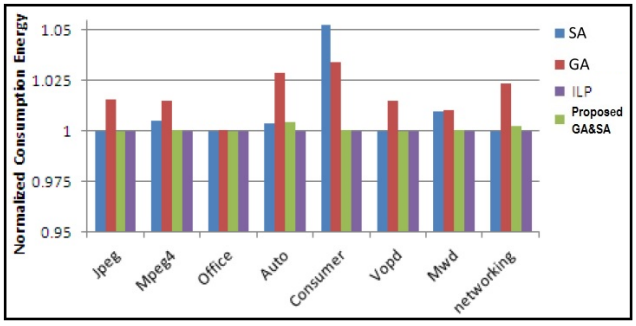
در نمودار شکل ۱۰، سرعت همگرایی الگوریتم‌ها در ۱۰۰۰ نسل متوالی برای محک‌های Jpeg و Office رسم شده است. همان‌طور که مشاهده می‌شود با استفاده از الگوریتم شبیه‌سازی گداخت و الگوریتم پیشنهادی برای این دو محک، بعد از گذشت تعداد کمی نسل، صد در صد به جواب بهینه دست می‌یابیم ولی با الگوریتم ژنتیک بعد از ۱۰۰۰ نسل به ترتیب به نرخ موفقیت ٪۶۲ و ٪۹۹ دست خواهیم یافت. البته الگوریتم ژنتیک در نهایت با تعداد تکرار بیش‌تر، پتانسیل رسیدن به جواب بهینه را دارد. الگوریتم پیشنهادی با تعداد تکرار بسیار اندک به جواب بهینه دست یافته و از سرعت همگرایی بسیار بیشتری برخوردار است.

جدول ۴- زمان پردازش مرحله نگاشت الگوریتم‌های ژنتیک، شبیه‌سازی گداخت، و پیشنهادی

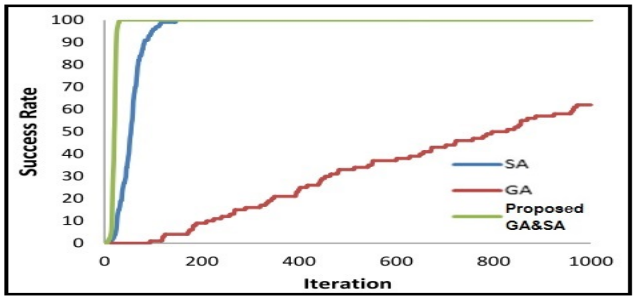
محک، الگوریتم	شبیه‌سازی گداخت	ژنتیک	پیشنهادی
auto	۱۹.۴۲	۲۳۷۱.۱۶	۱۱۳.۰۲
office	۰.۰۲۹	۰.۷۷	۰.۰۲۹
consumer	۰.۱۶	۵.۸۹	۰.۵۷
networking	۰.۳۱	۱۱.۹۳	۰.۸۲
mwd	۰.۲۶	۱۸.۱۵	۰.۶۲
Mpeg4	۰.۲۳	۱۳.۳۴	۰.۵۸
vopd	۰.۲۵	۱۲.۳۲	۰.۵۹
Jpeg	۰.۰۵	۴.۴۶	۰.۲



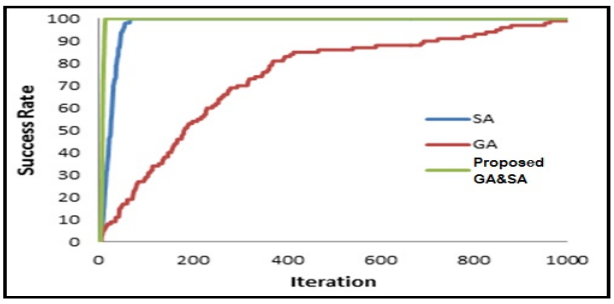
شکل ۸- سرعت اجرای نگاشت الگوریتم‌های ژنتیک، شبیه‌سازی گداخت، بهینه‌سازی و پیشنهادی



شکل ۹- انرژی مصرفی نگاشت الگوریتم‌های ژنتیک، شبیه‌سازی گداخت، بهینه‌سازی و پیشنهادی



(الف) همگرایی روش نگاشت در محک Consumer



(الف) همگرایی روش نگاشت در محک Office

شکل ۱۰- سرعت همگرایی الگوریتم‌های ژنتیک، شبیه‌سازی گداخت و پیشنهادی

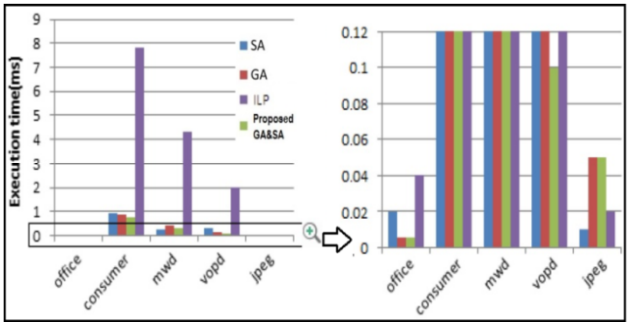
۵-۳-۲- ارزیابی الگوریتم‌های زمان‌بندی

زمان اجرای زمان‌بندی الگوریتم پیشنهادی از الگوریتم ژنتیک کمتر و از الگوریتم شبیه‌سازی گداخت بیشتر است (شکل ۱۱). با افزایش تعداد وظایف، این اختلاف

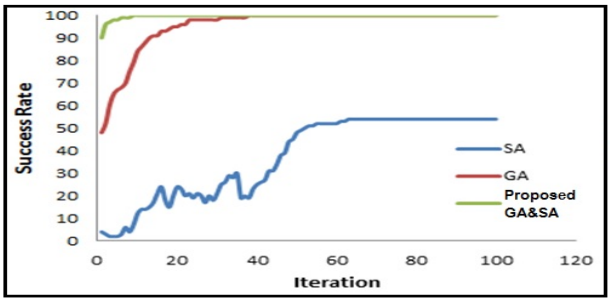
بسیار بیش‌تر می‌شود. البته احتمال‌گیر افتادن الگوریتم شبیه‌سازی گداخت در بهینه محلی زیاد و در صورت تکرار الگوریتم نیز مقادیر تقریبی به‌دست آمده بهبود نخواهند یافت (جدول ۵). الگوریتم شبیه‌سازی گداخت به صورت صد در صد به جواب نمی‌رسد و با تکرار الگوریتم نیز مقادیر تقریبی به دست آمده بهبود پیدا نخواهند کرد ولی الگوریتم ژنتیک بهبودیافته با دقت صد درصد به جواب می‌رسد. و زمان آن در برابر روش‌های قطعی بسیار ناچیز است (جدول ۵). انرژی مصرفی الگوریتم پیشنهادی نیز مانند نگاشت، از الگوریتم ژنتیک و شبیه‌سازی گداخت کمتر ولی از الگوریتم برنامه‌ریزی خطی بیشتر است.

جدول ۵- زمان اجرای زمان‌بندی الگوریتم‌های ژنتیک، شبیه‌سازی گداخت، قطعی و پیشنهادی

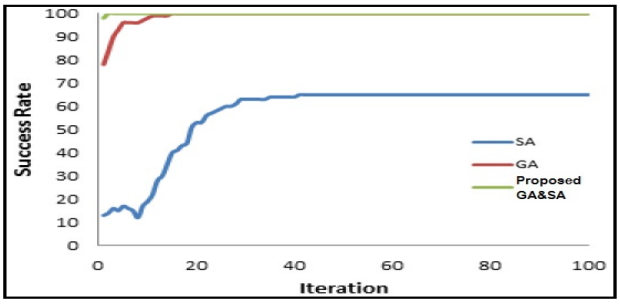
محک، الگوریتم	شبیه‌سازی گداخت	ژنتیک	پیشنهادی	قطعی
office	۰.۰۲	۰.۰۰۶	۰.۰۰۶	۰.۰۴
consumer	۰.۹۱	۰.۸۸	۰.۷۸	۷.۸۳
mwd	۰.۲۵	۰.۴	۰.۳۱	٤.۳٤
vopd	۰.۳	۰.۱۲	۰.۱	۲
jpeg	۰.۰۱	۰.۰۵	۰.۰۵	۰.۰۲



شکل۱۱- سرعت اجرای زمان‌بندی الگوریتم‌های ژنتیک، شبیه‌سازی گداخت، بهینه‌سازی و پیشنهادی



(الف) همگرایی روش زمان‌بندی در محک Consumer



(الف) همگرایی روش زمان‌بندی در محک MWD

شکل ۱۲- سرعت همگرایی الگوریتم‌های ژنتیک، شبیه‌سازی گداخت و پیشنهادی

در نمودار شکل ۱۲ سرعت همگرایی الگوریتم‌های پیاده‌سازی شده برای رسیدن به مقدار بهینه در ۱۰۰ نسل متوالی برای محک‌های Consumer و MWD ترسیم شده است. با مشاهده آنها بعد از گذشت تعدادی نسل به صورت ۱۰۰٪ به جواب بهینه دست می‌یابیم که دارای شیب خوب همگرایی هستند. همگرایی روش پیشنهادی در مقابل الگوریتم ژنتیک با افزایش نسل‌ها بیش‌تر استولی با الگوریتم شبیه‌سازی گداخت در ۱۰۰ بار اجرا به ترتیب به نرخ موفقیت ۵۴٪ و ۶۷٪ دست خواهیم یافت. به‌عبارتی در محک اول از ۱۰۰ آزمایش در تکرار ۶۳ تنها ۵۴ مورد به جواب بهینه دست یافته‌اند و مابقی در ۴۶ مورد به‌جواب رسیده‌اند. حتی با تکرار نیز به جواب بهینه نمی‌رسند و در تله بهینه‌های محلی گرفتار می‌شوند.

۵-۳-۳- ارزیابی متدلوژی پیشنهادی

با توجه به نتایج شبیه‌سازی و مقایسه زمان اجرا و انرژی مصرفی می‌توان بیان کرد، در مواردی که تعداد هسته‌های معماری و وظایف کاربرد کم باشد استفاده از روش‌های قطعی ممکن و تضمین‌کننده جواب بهینه است. به‌دلیل افزایش نمایی زمان محاسبات با افزایش تعداد وظایف کاربرد، این روش‌ها برای کاربردهای بزرگتر در مدت زمان خیلی زیاد جواب می‌دهند. وقتی که این تعداد از حد مشخصی بیشتر می‌شود مساله به دلیل افزایش نمایی فضای حافظه کامپیوتر غیر قابل حل خواهد بود لذا استفاده از روش‌های مکاشفه‌ای اجتناب ناپذیراست و جواب نزدیک به بهینه برای مساله را در مدت زمان کوتاهی ارایه می‌دهند. اگر پارامترهای اولیه خوبی برای روش شبیه‌سازی گداخت انتخاب گردد احتمال به دام افتادن آن در بهینه محلی کم و جواب خوبی ارایه می‌دهد.

روش الگوریتم ژنتیک تضمین‌کننده جواب خوب نخواهد بود زیرا ممکن است مقادیر نزدیک‌بهینه که در یک نسل به‌وجود آمده‌اند در نسل‌های بعدی با عملیات جهش و تقاطع بر روی آن‌ها حذف گردند و مجموعه جواب حاصل در نسل‌های بعدی یک نسل، نسبت به آن نسل دارای میانگین تابع ارزیاب بدتر باشد. ارایه روش ما تضمین‌کننده بهبود پس از مرحله‌جهش در الگوریتم پیشنهادی، این مشکل الگوریتم ژنتیک را برطرف می‌کند. استفاده از این روش موجب بهبود جواب در هر نسل می‌گردد که ضمن برخورداری از سرعت همگرایی بالا، در همزمانی نگاشت و زمان‌بندی وظایف و ارتباطات از سرعت اجرای خوبی نیز برخوردار است.

۶- نتیجه‌گیری

ما یک متدلوژی نگاشت و زمان‌بندی بی‌درنگ انرژی‌آگاه برای برنامه‌ریزی همزمان وظایف و ارتباطات با هدف حل سریع با جواب نزدیک بهینه در تراشه‌های چند هسته‌ای ارایه داده‌ایم که همزمان با کمینه‌سازی انرژی مصرفی و کاهش زمان اجرا، کار تولید جواب دقیق مساله را با کاهش تعداد جستجوها و با فرار از تله پاسخ‌های بهینه محلی انجام می‌دهد. متدلوژی پیشنهادی با برخورداری از ساختار نوین کروموزوم در الگوریتم ژنتیک و برخورداری از تابع جهش شبیه‌سازی گداخت، دارای قابلیت جلوگیری از تولید راه‌حل‌های غیرممکن جهت کاهش زمان تولید جواب نزدیک بهینه است. تحلیل نتایج آزمایشات نشان می‌دهد که در نگاشت و زمان‌بندی همزمان نسبت به روش سنتی ژنتیک از سرعت همگرایی و تولید جواب بسیار خوب همراه با تولید جواب نزدیک بهینه برخورداری است. همچنین قابل کاربرد برای طراحی سیستم‌های بزرگ و بی‌درنگ همگن و ناهمگن است. ما در برنامه آینده تحقیقاتی خود در حال افزودن قابلیت اطمینان به متدلوژی پیشنهادی و توسعه آن برای پشتیبانی از ساختارهای چندولتاژی هستیم.

مراجع

[1] A. Kumar, M. Shafique, A. Kumar, and J. Henke, "Mapping on Multi/Many-core Systems: Survey of Current and Emerging Trends," *Proc. DAC*, pp. 338-342, 2013.

[2] S. Borkar, "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation," *IEEE Micro*, vol. 25, pp. 10-16, 2005.

[3] O. Eduard, and et. al., "Energy Efficiency and Renewable Energy Iintegration in Data Centres," Strategies and modelling review, Renewable and Sustainable Energy Reviews, no.42, pp. 429-445, 2015.

[4] S. Mittal, "A Survey of Techniques for Improving Energy Efficiency in Embedded Computing Systems," *International Journal of Computing Aided Engineering and Technology*, vol. 6, no. 4, pp. 450-459, 2014.

[5] P. Nathaniel, D. Blaauw, and D. Sylvester, "Low-Power Near-Threshold Design: Techniques to Improve Energy Efficiency Energy-Efficient Near-Threshold Design Has Been Proposed to Increase Energy Efficiency Across a Wid," IEEE Solid-State Circuits Magazine, vol. 7, no.2, pp. 49-57, 2015.

[6] J. Sartori, A. Pant, R. Kumar, and P. Gupta, "Variation-aware Speed Binning of Multi-core Processors," *11th ACM/IEEE International Symposium on Quality Electronic Design, ISQED*, San Jose, 2010.

[7] R. Viswanath, V. Wakharkar, A. Watew, and V. Lbonheur, "Thermal Performance Challenges from Silicon to Systems," *Intel Technology Journal*, vol.4, no.3, pp. 1-16, 2000.[Online].Available:http://www.intel.com/technology/itj / q32000/articles/art_4.htm.

[8] L. Y. Lin, and et. al., "Communication-driven Task Binding for Multiprocessor with Latency Insensitive Network-on-chip," *ASP-DAC*, pp. 39–44, 2005.

[9] G. Ascia, V. Catania, and M. Palesi, "Multi-objective Mapping for Mesh-based NoCArchitectures," *CODES+ISSS*, pp. 182–187, 2004.

[10] International technology roadmap for semiconductors, 2010, http://www.itrs.net/Links/2010ITRS/Home2010.htm.

[11] A. Mahabadi, SM. Zahedi, and A. Khonsari, "Reliable Energy-aware Application Mapping and Voltage–frequency IslandPartitioning for GALS-based NoC," *Journal of Computer and System Sciences*, vol. 79, no.4, pp.57–74, 2013.

[12] B. Khodabandeloo, A. Khonsari, F. Gholamian, M. H. Hajiesmaili, A. Mahabadi, and H. Noori, "Scenario-based Quasi-static Task Mapping and Scheduling for Temperature-efficient MPSoCDesign under Process Variation," *Microprocessors and Microsystems*, vol. 38, pp. 399–414, 2014.

[13] J. Castrillon, A. Tretter, R. Leupers, and G. Ascheid, "Communication-aware Mapping of KPN Applications onto Heterogeneous MPSoCs," *DAC*, pp. 1266–1271, 2012.

[14] W. Che, and K. S. Chatha, "Unrolling and Retiming of Stream Applications onto Embedded Multicore Processors," *DAC*, pp. 1272–1277, 2012.

[15] S. Gupta, G. Agarwal, and V. Kumar, "Task Scheduling in Multiprocessor System Using Genetic Algorithm," *Machine Learning and Computing (ICMLC), Second International Conference*, pp. 267-271, 2010.

[16] S. Sivanandam, and P. Visalakshi, "Dynamic Task Scheduling with Load Balancing using Parallel Orthogonal Particle Swarm Optimisation," *International Journal of Bio-Inspired Computation*, vol. 1, pp. 276-286, 2009.

[17] S. Ninomiya, K. Sakanushi, Y. Takeuchi, and M. Imai, "Task Allocation and Scheduling for Voltage-Frequency Islands Applied NoC-based MPSoC Considering Network Congestion," *Embedded Multicore Socs (MCSoC), IEEE 6th International Symposium*, pp. 107-112, 2012.

[18] G. Chen, F. Li, S. and Son, M. Kandemir, "Application Mapping for Chip Multiprocessors," *DAC*, pp. 620–625, 2008.

[19] S. Banerjee, and N. Dutt, "Efficient Search Space Exploration for HW-SW Partitioning," *International Conference on Hardware/Software Codesign and System Synthesis*, pp. 122-127, 2004.

[20] A. Hartman, D. Thomas, and B. Meyer, "A Case for Lifetime-aware Task Mapping in Embedded Chip Multiprocessors," *CODES+ISSS*, pp. 145 –154, 2010.

[21] E. Seo, Y. Koo, and J. Lee, "Dynamic Repartitioning of Real-time Schedule on a Multicore Processor for Energy Efficiency," *Proc. Int. Conf. Embedded and Ubiquitous Computing*, pp. 69–78, Aug. 2006.

[22] J. Hu, and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.*, no. 4, pp. 551–562, 2005.

[23] C. Marcon, A. Borin, A. Susin, L. Carro, and F. Wagner, "Time and energy efficient mapping of embedded applications onto NoCs," *ASP-DAC*, pp. 33–38, 2005.

[24] C. Marcon, E. Moreno, N. Calazans, and F. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption. Computers Digital Techniques," *IET*, pp. 471–482, 2008.

[25] B. H. Meyer, A. S. Hartman, and D. E. Thomas, "Cost-effective Slack Allocation for Lifetime Improvement in NoC-based MPSoCs," *DATE*, pp. 1596–1601, 2010.

[26] H. Orsila, and et. al., "Automated Memory-aware Application Distribution for Multi-processor System-on-Chips," *J. Syst. Archit.*, no. 11, pp.795–815, 2007.

[27] X. Wu, and et. al., "Genetic Algorithms for Integrating Cell Formation with Machine Layout and Scheduling," *Computers & Industrial Engineering*, vol. 5, no. 2, pp. 277-289, 2007.

[28] J. Choi, H. Oh, S. Kim, and S. Ha, "Executing Synchronous Dataflow Graphs on a SPM-based Multicore Architecture," *DAC*, pp. 664–671, 2012.

[29] S. Manolache, P. Eles, and Z. Peng, "Task Mapping and Priority Assignment for Soft Real-time Applications under Deadline Miss Ratio Constraints," *ACM Trans. Embed. Comput. Syst.*, vol. 19, pp.13–19, 2008.

[30] H. Javaid, and S. Parameswaran, "A Design Flow for Application Specific Heterogeneous Pipelined Multiprocessor Systems," *DAC*, pp. 250–253, 2009.

[31] M. Ruggiero, and et. al., "Communication-aware Allocation and Scheduling Framework for Stream-oriented Multi-processor Systems-on-chip," *DATE*, pp. 3–8, 2006.

[32] L. Thiele, I. Bacivarov, W. Haid, and K. Huang, "Mapping Applications to Tiled Multiprocessor Embedded Systems," *ACSD*, pp. 29–40, 2007.

[33] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "A methodology for Mapping Multiple Use-cases onto Networks on Chips," *DATE*, pp. 118–123, 2006.

[34] C.-E. Rhee, H.-Y. Jeong, and S. Ha, "Many-to-Many Core-Switch Mapping in 2-D Mesh NoC Architectures," *ICCD*, pp. 438–443, 2004.

[35] C. M. Chen, and C. T. King, "Using Integer Linear Programming for Instruction Scheduling and Register Allocation in Multi-issue Processors," *Multi-Issue Processors. Computers and Mathematics with Applications*, 1997.

[36] Y. Zhang, D. Parikh, K. Sankaranarayanan, K. Skadron, and M. Stan, "HotLeakage: A temperature-aware Model of Sub Threshold and Gate Leakage for Architects," *Tech. Rep. CS-2003-05*, University of Virginia, 2003.

[37] K. Bowman, S. Duvall, and J. Meindl, "Impact of Die-to-die and within Die Parameter Fluctuations on the Maximum Clock Frequency Distribution for Giga Scale Integration," *IEEE J. Solid State Circuits*, vol. 37, no. 2, pp. 183-190, 2002.

[38] W. Zhang, G. Sun, and S. Bin, "A Novel Task Communication and Scheduling Algorithm for NoC-based MPSoC," *International Journal of Smart Home*, vol. 9, no. 10, pp. 179-188, 2015.

[39] R. Dick. "Embedded Systems Synthesis Benchmarks Suite (e3s)," http://www.ece.northwestern.edu/ dickrp/e3s/.

[40] E. L. Lawler, and C. U. Martel, "Scheduling Periodically Occurring Tasks on Multiple Processors," *Information Processing Ltrs.*, vol. 7, no. 1, pp. 9-12,1981.

[41] T. Chantem, X.S. Hu, and R.P. Dick, "Temperature-aware Scheduling and Assignment for Hard Real-time Applications on MPSoCs," *IEEE Trans. VLSI System*, pp. 1884-1897, 2011.

[42] G. Link, and N. Vijaykrishnan, "Thermal Trends in Emerging Technologies," In Proc. Int. Symp. Quality of Electronic Design, pp. 625–632, 2006.

[43] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, "Scaling with Design Constraints Predicting the Future of Big Chips," *IEEE Micro special issue on Big Chips*, 2011.

[44] M. Momtazpour, E. Sanaei, and M.Goudarzi, "Power-yield Optimization in MPSoCTask Scheduling under Process Variation," ISQED, pp. 747-754, 2010.

[45] CPLEX 11.1ILOG: http://www.ilog.com/product/cplex/, 2013.

[46] HotSpot: http://lava.cs.virginia.edu/HotSpot/, 2013.

[47] R. Teodorescu, B. Greskamp, J. Nakano, S. Sarangi, A. Tiwari, and J. Torrellas, "VARIUS: A Model of Parameter Variation and Resulting Timing Errors for Microarchitects," 2nd Workshop *on Architectural Support for Gigascale Integration*, San Diego, USA, 2007.

[48] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A Model of Process Variation and Resulting Timing Errors for Microarchitects," *IEEE Trans. on Semiconductor Manufacturing*, vol. 21, no. 1, 2008.

[49] A. Bonfietti, L. Benini, M. Lombardi, and M. Milano, "An Efficient and Complete Approach for Throughput-maximal SDFAllocation and Scheduling on Multi-core Platforms," *DATE*, pp. 897–902, 2010.

[50] N. Satish, K. Ravindran, and K. Keutzer, "ADecomposition-based Constraint Optimization Approach forStatically Scheduling Task Graphs with Communication Delaysto Multiprocessors," *DATE*, pp. 57–62, 2007.

[51] L. Thiele, L. Schor, H. Yang, and I. Bacivarov, "Thermal-awareSystem Analysis and Software Synthesis for Embedded Multi-processors," *DAC*, pp. 268–273, 2011.

[52] D. Wu, B. M. Al-Hashimi, and P. Eles,"Scheduling and Mapping of Conditional Task Graphs for the Synthesis of LowPower Embedded Systems," *DATE*, pp. 10090, 2003.

[53] Y. Markovskiy, E. Caspi, R. Huang, J. Yeh, M. Chu, J. Wawrzynek, and A. DeHon, Analysis of Quasi-Static Scheduling Techniques in a Virtualized Reconfigurable Machine. Proceedings of ACM/SIGDA Tenth International Symposium on Field-Programmable Gate Arrays, pp. 196-205, 2002.

امین‌اله مه‌آبادی تحصیلات خود را در رشته مهندسی برق سخت‌افزار و معماری کامپیوتر به انجام رسانده و اکنون استادیار گروه مهندسی کامپیوتر و فناوری اطلاعات دانشگاه شاهد است. تحقیقات مورد علاقه نامبرده طراحی مدارات مجتمع، مدیریت منابع بر تراشه، زمان‌بندی شبکه بر تراشه و طراحی ابزارهای شبیه‌سازی هوشمند است.
آدرس پست‌الکترونیکی ایشان عبارت است از:
mahabadi@shahed.ac.ir

فاطمه عسگری بیدهندی لیسانس خود را در رشته مهندسی کامپیوتر نرم‌افزار از دانشگاه الزهرا و فوق‌لیسانس خود را در رشته معماری کامپیوتر از دانشگاه شاهد اخذ نموده است. تحقیقات مورد علاقه نامبرده معماری کامپیوتر، زمان‌بندی و مدیریت منابع است.
آدرس پست‌الکترونیکی ایشان عبارت است از:
fateme.asgari@gmail.com

اطلاعات بررسی مقاله:

تاریخ ارسال: ۱۳۹۴/۰۸/۲۸
تاریخ اصلاح: ۱۳۹۴/۰۹/۲۹
تاریخ قبول شدن: ۱۳۹۴/۱۰/۱۵
نویسنده مرتبط: دکتر امین‌اله مه‌آبادی، دانشکده فنی و مهندسی، دانشگاه شاهد، تهران، ایران.

[1]Performance
[2]Semiconductor
[3]Multiprocessor System on Chip (MPSoC)
[4]Power Density
[5]Leakage Power
[6]Deep Submicron
[7]Embedded System
[8]Application
[9]Constraint
[10]Hardware Task
[11]Software Task
[12]Digital Signal Processing(DSP)
[13]Accelerator
[14] Integer Linear Programming (ILP)
[15] Condition Linear Programming (CLP)
[16]Non Linear Programming (NLP)
[17] Mixed Integer Linear Programming (MILP)
[18]Homogenous
[19]NP-Hard
[20] Constructive
[21] Transformative
[22] Branch-and-Bound (BB)
[23]Heuristic
[24] Genetic Algorithm (GA)
[25] Simulated Annealing (SA)
[26] Network on Chip (NoC)
[27] Pareto
[28]One-point Crossover
[29] 2D Mesh
[30]Non-Preemptive Scheduling
[31]Critical Path
[32]Lognormal
[33] Mesh
[34] Directed Acyclic Graph (DAG)
[35] Source
[36]Sink
[37]General
[38]Speed Binding
[39]Scheduler
[40]Hyper Periodic
[41]Earliest Start Time (EST)
[42]Latest Start Time (LST)
[43]Benchmarks
[44]Technology Scaling