Journal of Soft Computing and Information Technology (JSCIT)
Babol Noshirvani University of Technology, Babol, Iran
Journal Homepage: jscit.nit.ac.ir
Volume 9, Number 3, Fall 2020, pp. 196-205
Received: 07/10/2019, Revised: 04/30/2020, Accepted: 06/29/2020

Performance Improvement in Multiprocessors Using Three Steps of Non Contiguous Migration for Online Mapping

Akram Reza[1], and Mahnaz Rafie[2*]
1- Department of Computer Engineering, Shahre-Qods Branch, Islamic Azad University, Tehran, Iran.
2*- Department of Computer Engineering, Ramhormoz Branch, Islamic Azad University, Ramhormoz, Iran.
[1]a.reza@qodsiau.ac.ir, and [2*]m.rafie@srbiau.ac.ir

Corresponding author address: Mahnaz Rafie, Faculty of Computer Engineering, Islamic Azad University of Ramhormoz, Ramhormoz, Iran, Post Code: 151-63825.

Abstract- In this paper, we have presented different concepts and parameters in the online mapping for different jobs in the network on chips. Thus, three essential steps are considered which are finding the appropriate size of sub-mesh, finding a sub-mesh place in integrating the mesh for online task allocation and finding the main place in sub-mesh for online task mapping. For this purpose, efficient previous models to select the dimensions of the sub-mesh, the processor migration methods based on the two-row boundary, limited left-right compaction, limited top-down compaction, online dynamic compaction-four corner (ODC-FC) and hybrid migrations for mesh topology are compared with the proposed algorithm to check the comparative performance. Also, the impact of different performance parameters which are average job execution time and average system utilization will be compared against the previous mechanisms to achieve the appropriate configuration of the network on chips. It is worth noting that in this article, 7 algorithms, which have achieved better performance, have been selected among the 29 ones. We have demonstrated that using hybrid migration strategies enable us to limit the number of processors migrations. Consequently, significant improvements have been achieved in the average job execution time (%36 ~ %38.1), and the average system utilization (%38.2~%48.5).

Keywords- Allocation, Fragmentation, Migration, Multiprocessors, Network on Chip.

مجله علمی پژوهشی رایانش نرم و فناوری اطلاعات
دانشگاه صنعتی نوشیروانی بابل
صفحه مجله: jscit.nit.ac.ir
جلد ۹، شماره ۳، پائیز ۱۳۹۹ ، صفحه ۱۹۶-۲۰۵
دریافت: ۱۳۹۸/۰۴/۱۹، بازنگری: ۱۳۹۹/۰۲/۱۱، پذیرش: ۱۳۹۹/۰۴/۰۹

بهبود کارایی در چندپردازنده‌ای‌ها با استفاده از سه مرحله مهاجرت ناپیوسته جهت نگاشت برخط

اکرم رضا[۱]، مهناز رفیعی[*۲]
۱- گروه کامپیوتر، واحد شهرقدس، دانشگاه آزاد اسلامی، تهران، ایران.
*۲- گروه کامپیوتر، واحد رامهرمز، دانشگاه آزاد اسلامی، رامهرمز، ایران
[1]a.reza@qodsiau.ac.ir, and [2*]m.rafie@srbiau.ac.ir

*نشانی نویسنده مسئول: مهناز رفیعی، رامهرمز، میدان توحید، دانشگاه آزاد اسلامی واحد رامهرمز، دانشکده مهندسی کامپیوتر، کدپستی: ۶۳۸۲۵-۱۵۱.

چکیده: در این مقاله مفاهیم و پارامترهای مختلف در نگاشت برخط برای کارهای متعدد در شبکه‌روی‌تراشه بررسی شده است. در این راستا سه گام اساسی پیدا نمودن اندازه زیرتوری مناسب، محل زیرتوری در همبندی‌توری جهت تخصیص و مکان اصلی در زیرتوری جهت نگاشت بر خط کار در نظر گرفته شده است. لذا الگوریتم‌های مؤثر پیشین جهت انتخاب ابعاد زیرتوری، الگوریتم‌های مهاجرت پردازنده مبتنی بر دو مرز سطری، فشرده‌سازی بالا پایین محدود شده، فشرده‌سازی چهارگوشه برخط پویا (ODC-FC) و روش‌های مهاجرت ترکیبی برای همبندی توری با الگوریتم پیشنهادی جهت بررسی کارایی مقایسه شده‌است. در این راستا، تأثیر پارامترهای کارآیی میانگین زمان اجرای کار و میانگین بهره‌وری سیستم با الگوریتم‌های پیشین جهت دستیابی به پیکربندی مناسب در شبکه‌های روی تراشه بررسی شده‌اند. در این مقاله بیست و نه الگوریتم مختلف پیاده‌سازی شده و از بین آن‌ها هفت الگوریتم که عملکرد بهتری نسبت به سایرین دارا هستند انتخاب شده‌است. در واقع، با استفاده از روش‌های مهاجرت تلفیقی کارا توانستیم تعداد مهاجرت‌های پردازنده‌ها را محدود نماییم و در نتیجه میانگین زمان اجرای کار بین ۳۶٪ تا ۳۸.۱٪ و میانگین بهره‌وری سیستم بین ۳۸.۲٪ تا ۴۸.۵٪ بهبود یافته است.

واژه های کلیدی: تخصیص، تکه‌تکه‌شدن، چندپردازنده‌ای‌ها، شبکه‌روی‌تراشه، مهاجرت.

۱- مقدمه

با افزایش تعداد هسته‌های پردازشی در یک تراشه مجتمع، عملکرد شبکه‌روی‌تراشه[۱] (*NOC*) به‌عنوان زیرساخت ارتباطی در پارامترهای مصرف انرژی، تأخیر انتقال‌داده‌ها و حرارت می‌تواند بسیار محدودکننده باشد. با رشد اندازه شبکه، تأخیر شبکه به صورت تصاعدی افزایش می‌یابد. در نتیجه بهره‌وری شبکه به طور قابل توجهی کاهش می‌یابد. بنابراین، اتصال الکتریکی سنتی با چالش‌های جدی توان مصرفی و مقیاس‌پذیری شبکه مواجه خواهد شد. یکی از مسائل مهمی که می‌تواند تأثیر به‌سزایی در طراحی تراشه داشته باشد، نگاشت کاربرد بر روی همبندی شبکه‌روی تراشه است. با استفاده از نگاشت، هسته‌های کاربرد موردنظر که هر کدام وظیفه‌ای خاص را در تراشه بر عهده دارند، در گره‌ای مناسب از شبکه‌روی‌تراشه جانشانی خواهند شد. نگاشت کارهای موازی بر روی یک تراشه چندپردازنده‌ای در زمان طراحی (نگاشت برون خطی[۲]) و یا در زمان اجرا (نگاشت برخط[۳]) انجام می‌شود. در

نگاشت‌برخط، انتساب و تخصیص پردازنده به کارها همزمان با اجرای برنامه‌های کاربردی دیگر انجام می‌شود که سربار محاسباتی الگوریتم نگاشت ممکن است باعث افزایش تأخیر و توان مصرفی برنامه در حال اجرا شود [۱]. برای نگاشت برخط چند کار بر روی تراشه چندپردازنده‌ای، دو گام عمده تخصیص‌کار و نگاشت‌کار می‌بایست انجام شود. نگاشت چند کار بر روی NoC چند هسته‌ای را می توان تحت دو شرط به کار برد: زمانی‌که تعداد هسته‌ها در تراشه بیشتر از مجموع هسته‌های مورد نیاز برای اجرای تمام کار باشد (نگاشت نامحدود) و یا زمانی‌که تعداد هسته‌ها در تراشه کمتر از مجموع هسته‌های مورد نیاز برای اجرای تمام کارها باشد (نگاشت محدود). در ضمن، دو طرح برای تخصیص پردازنده وجود دارد، تخصیص‌پیوسته و تخصیص‌ناپیوسته. در تخصیص‌پیوسته، گره‌های اختصاص داده شده به یک کار در مجاورت یکدیگر و در یک زیرتوری است. در تخصیص‌ناپیوسته، کار قادر است روی چندین زیرتوری گسسته کوچک‌تر اجرا شود، نسبت به این‌که همواره منتظر بماند تا زمانی‌که یک زیرتوری منفرد از اندازه مورد تقاضا موجود شود. با بالابردن شرایط پیوستگی در تخصیص ناپیوسته، انتظار می‌رود که تکه‌تکه‌شدن پردازنده کاهش یابد و در نتیجه بهره پردازنده افزایش یابد که منجر به بار کاری ترافیکی بیشتر و تأخیر ارتباطی بالاتر می‌گردد (به‌دلیل فاصله طولانی‌تر پیغام‌های یک کاربرد که می‌بایست در طول زیرشبکه‌های تخصیص داده شده مختلف حرکت نماید) [۲]. مشکل اصلی در تخصیص‌پیوسته تکه‌تکه‌شدن خارجی است. زمانی‌که تعداد گره‌های آزاد بیشتر از گره‌های موردنیاز برای یک کار است و گره‌های آزاد به‌صورت مجزا وجود دارد، تخصیص پردازنده به کار متوقف می‌شود [۳]. در تخصیص غیر انحصاری برای مشکل تکه‌تکه‌شدن خارجی راه‌حلی وجود ندارد؛ با این حال، راه‌حل آن در تخصیص انحصاری، مهاجرت کار است. در این مقاله نگاشت برخط برای کارهای متعدد در شبکه‌روی‌تراشه با همبندی توری به‌منظور دستیابی به نگاشت محدود و غیرانحصاری مورد بررسی قرار گرفته است. برای انجام نگاشت موردنظر باید چهار گام انجام شود که شامل محاسبه اندازه زیرتوری بر اساس اندازه کار ورودی، تخصیص زیرتوری به کار، آزاد سازی زیرتوری بعد از اتمام کار، نگاشت کار بر روی فضای زیرتوری و مهاجرت (جابجایی) زیر توری‌ها در زمان رخ دادن تکه‌تکه شدن خارجی[۴] می باشند. لازم به‌ذکر است که در تخصیص غیر انحصاری گام مهاجرت زیرتوری‌ها حذف می‌گردد [۴]. با توجه به مطالب مذکور، در بخش ۲، الگوهای مبتنی بر نگاشت، الگوریتم‌های تخصیص پیوسته و روش‌های مهاجرت کاری که در گذشته انجام شده‌اند بررسی می‌شود. در بخش ۳، الگوریتم پیشنهادی مطرح شده است. سپس، شبیه ساز OM[۵] [۵-۷] در بخش ۴ بررسی شده است. در نهایت، در بخش ۵، نتایج شبیه‌سازی‌ها بر اساس دو پارامتر میانگین زمان اجرای کار و میانگین بهره‌وری سیستم مورد ارزیابی واقع شده‌اند.

۲- مروری بر پژوهش های پیشین در نگاشت برخط

در این مقاله بر اساس استراتژی تخصیص که می‌تواند انحصاری یا غیرانحصاری باشد، سه یا چهار تابع باید اجرا شوند. در مجموع، چهار گام جهت تخصیص انحصاری در نظر گرفته می‌شود. گام اول بر اساس اندازه کار ورودی، ابعاد زیرتوری تعیین می‌شود. در گام دوم یک زیر توری آزاد برای کار ورودی بر روی همبندی توری تعیین می‌شود. برای تخصیص در این مقاله از تخصیص پیوسته به‌منظور کاهش فاصله بین گره‌های مرتبط یک کار استفاده شده است. گام سوم پیدا کردن مکان اصلی در زیرتوری جهت نگاشت برخط‌کار است. لازم به‌ذکر است که تکه‌تکه‌شدن خارجی مشکل اصلی در تخصیص پیوسته است [۵]. لذا الگوریتم‌های مهاجرت کار به‌عنوان گام چهارم مطرح شده‌اند. در ادامه کارهای انجام شده پیشین در این راستا بررسی شده است.

۲-۱- محاسبه اندازه زیرتوری مناسب برای کار ورودی

در این مقاله دو الگوریتم حداقل قطر(MD[6]) و عدد اول کمینه و مینیمم آستانه[۷] ($MT\&MPN$) [۷] در نظر گرفته شده اند. یکی از اهدافی که در روش MD [۵، ۶] می‌توان در نظر گرفت حداقل نمودن میزان اتلاف است. برای حداقل نمودن میزان اتلاف در یک زیرتوری باید قطر زیرتوری حداقل باشد، زیرا قطر شبکه دارای تأثیر مستقیم بر روی ماکزیمم اتلاف شبکه است. الگوریتم مزبور جهت تعیین ابعاد زیرتوری مناسب برای کار ورودی است که در ابتدا اندازه کار ورودی به صورت لیستی از مضرب‌های دوتایی شکسته می‌شود و سپس از لیست به دست آمده زوجی از مضرب‌ها که دارای کمترین اختلاف نسبت به هم هستند به‌عنوان اندازه زیرتوری انتخاب می‌شود [۵]. هدف اصلی در روش $MT\&MPN$ نیز انتخاب زیرتوری، کاهش تأخیر و افزایش پیوستگی در چندپردازنده‌ای‌ها است. در ضمن، این روش بر اساس مینیمم آستانه و عدد اول می‌باشد. در این الگوریتم، مقدار آستانه نه در نظر گرفته شده است. به‌عنوان مثال، اگر تعداد هسته‌های کار ورودی مساوی با ده باشد، زیرتوری (۲و۵) یا (۵و۲) در نظر گرفته می‌شود. افزون‌براین، چنانچه تعداد هسته‌های کار ورودی نه باشد، ابعاد زیرتوری (۱و۹) یا (۹و۱) در نظر گرفته می‌شود. در رابطه با تعداد هسته‌هایی که قابل تقسیم بر عامل‌های اول نیستند نیز استراتژی همانند الگوریتم MPN[۸] در نظر گرفته شده است.

الگوریتم مزبور از ایده استفاده از مینیمم عدد اول (*MPN*) استفاده می‌نماید. روش پیشنهادی مبتنی بر کوچکترین عدد اولی است که قابل تقسیم بر تعداد هسته‌هاست. این عدد به‌عنوان تعداد سطرها یا ستون‌های زیرتوری در نظرگرفته می‌شود. به‌عنوان مثال، چنانچه تعداد هسته‌ها دوازده باشد، چهار زیرتوری موجود در *Array* را به شرح ذیل می‌توان در نظر گرفت، {(۲و۶)، (۶و۲)، (۴و۳)، (۳و۴)}=*Array*، از لیست مزبور اولویت بر انتخاب زیرتوری است که سطر و ستون آن کمترین اختلاف را نسبت به هم داشته باشند یعنی {(۴و۳)، (۳و۴)}. در صورتی‌که زیرتوری مورد نظر یافته نشد از لیست *Array* باقی زیرتوری‌ها بر اساس دومین اولویت در راستای کمترین اختلاف بین زیرتوری‌ها انتخاب می‌گردد. در ضمن، چنانچه تعداد هسته‌ها بر هیچ عامل اولی قابل تقسیم نباشند به تعداد هسته‌ها یک واحد اضافه شده و به‌این‌صورت بر عامل‌های اول قابل تقسیم می‌گردد و الگوریتم مزبور طبق روندی که بیان شد اجرا می‌گردد. به‌عنوان مثال چنانچه تعداد هسته‌ها ۱۷ باشد، یک واحد به تعداد آن‌ها افزوده شده و ۱۸ در نظر گرفته می‌شوند. سپس *Array* به‌صورت {(۲و۹)، (۹و۲)، (۶و۳)، (۳و۶)}=*Array* ایجاد می‌شود.

در مجموع روش *MT&MPN* یک مقدار آستانه جهت انتخاب زیرتوری‌ها به شکل سطری یا ستونی قائل می‌شود و منجر می‌شود تا این شیوه تخصیص ساده‌تر و کم هزینه‌تر از تخصیص مستطیلی گردد. در این راستا با توجه به اندازه توپولوژی، حد آستانه‌ای با توجه به رابطه (۱) تعریف می‌شود تا قطر زیرتوری شبکه مینیمم بماند [۷].

$$threshold = 2\sqrt{n} + 1 \qquad (۱)$$

۲-۲- تخصیص پیوسته

برای این‌که بتوانیم از تمام توان محاسباتی یک چند پردازنده‌ای بزرگ به شکلی بهینه استفاده کنیم، داشتن یک الگوریتم تخصیص پردازنده کارآمد بسیار حیاتی می‌باشد. تخصیص پردازنده عهده‌دار انتخاب مجموعه‌ای از پردازنده‌ها به‌منظور اجرای کارهای موازی بر روی آنهاست. در واقع، الگوریتم تخصیص پردازنده، پردازنده‌های آزاد را برای کار انتخاب شده پیدا می‌کند. در مجموع، روش‌های تخصیص پردازنده را می‌توان به دو دسته کلی تقسیم نمود: پیوسته و ناپیوسته. در روش‌های تخصیص پیوسته، کار ورودی به یک گروه از پردازنده‌های آزاد پیوسته موجود در شبکه تخصیص می‌یابد. در تخصیص ناپیوسته، یک کار بر روی چندین زیرتوری کوچک‌تر از آنچه که کار ورودی درخواست نموده است، قادر به اجرا خواهد بود و در انتظار آزاد شدن یک زیرتوری پیوسته نمی‌ماند [۳]. از آنجایی‌که در این مقاله از مفهوم تخصیص پیوسته استفاده شده است، در این بخش ارزیابی کلی بر روی روش‌های تخصیص پیوسته‌ای که در گذشته انجام شده است مطرح می‌گردد. این روش‌ها شامل الگوریتم مبتنی بر پشته بهبود یافته (*ISBA*[۹]) [۸]، الگوریتم تخصیص دو مرز سطری[۱۰] (*TRB*) [۵]، الگوریتم تخصیص دو مرز ستونی[۱۱] (*TCB*) [۶] و الگوریتم سطری[۱۲] [۹] است. الگوریتم *ISBA*، از چرخش کار استفاده می‌نماید تا قابلیت تشخیص زیرتوری را به طور کامل کسب نماید. هنگامی‌که ابعاد کار *J(p,q)* مقادیر یکسانی دارند، (p=q) نیازی به چرخش کار نیست. با استفاده از یک پشته به عنوان محلی جهت ذخیره بلوک‌های داوطلب، الگوریتم اولین بلوک پایه‌ای که بیابد را به عنوان نتیجه باز می‌گرداند و آنچه در پشته باقی مانده است کنار گذاشته می‌شود [۸]. در الگوریتم *TRB*، انتخاب گره پایانی متفاوت است و بر اساس گره‌پایه و اندازه کار ($p\times q$) محاسبه می‌گردد. در واقع، هنگامی‌که بیش از یک گره پایه وجود داشته باشد، زیرتوری انتخاب می‌شود که گره پایه با گره پایانی آن حداقل فاصله را از نقاط بالا و پایین توری داشته باشد. در ضمن، حداقل تعداد اتصالات آزاد نیز در این استراتژی در نظر گرفته شده است. با استفاده از این مکانیزم زیرتوری‌ها به مرزهای سطری توری هدایت می‌گردند و در نتیجه تخصیص گره‌های آزاد در وسط توری نگه داشته می‌شوند. در نتیجه، مشکل تکه‌تکه‌شدن خارجی حداقل می‌شود [۵]. در الگوریتم *TCB*، انتخاب گره‌پایانی متفاوت است و بر اساس گره پایه و اندازه کار ($p\times q$) محاسبه می‌گردد که با رابطه (۲) نشان داده شده است.

$$endnodex_i = basenodex_i + p$$
$$endnodey_i = basenodey_i + q \qquad (۲)$$

در این استراتژی، هنگامی‌که بیش از یک گره پایه وجود داشته باشد، زیرتوری انتخاب می‌شود که گره پایه با گره پایانی آن حداقل فاصله را از نقاط راست و چپ توری داشته باشد. در ضمن، حداقل تعداد اتصالات آزاد نیز در این استراتژی در نظر گرفته شده است. معادلات (۳) برای محاسبه فاصله گره پایه و گره پایانی یک زیرتوری از مرزها استفاده می‌شود. اندازه توری $m\times n$ در نظر گرفته شده است.

$$b.x_i = basenode.x_i - 0$$
$$e.x_i = m - endnode.x_i$$
$$mid.d_i = min(b.x_i, e.x_i) \qquad (۳)$$
$$m.d = min(all\ min.d_i)$$

با استفاده از این مکانیزم، زیرتوری‌ها به مرزهای ستونی توری هدایت می‌گردند و در نتیجه تخصیص گره‌های آزاد در وسط توری

نگه داشته می‌شوند. در نتیجه، مشکل تکه‌تکه‌شدن‌خارجی حداقل می‌شود [۶].

الگوریتم تخصیص سطری [۹] تلاش می‌کند تا گره‌های مشغول را در پایین‌ترین سطر مجاور با دیگر زیرتوری‌های مشغول تخصیص دهد. در الگوریتم تخصیص سطری، اگر بیش از یک گره پایه وجود داشته باشد، الگوریتم گره پایه‌ای با کم‌ترین ردیف و حداقل اتصالات آزاد برای زیرشبکه موردنظر را نزدیک به دیگر زیرتوری‌های مشغول انتخاب می‌نماید.

در روش‌های تخصیص مذکور، تنها نواحی پیوسته جهت اجرای یک کار در نظر گرفته شده‌اند. در نتیجه انتظار می‌رود مسیرهای ارتباطی مینیمم باشند. لازم به ذکر است که در روش‌های تخصیص پیوسته ممکن است تعداد کافی از پردازنده‌ها موجود باشند اما تخصیص یک کار به شکست منجر شود [۵-۷]. استفاده از این روش‌های تخصیص منجر به تکه‌تکه‌شدن خارجی می‌گردد. لذا در ادامه چندین روش مهاجرت پردازنده به‌عنوان راه‌حلی جهت بهبود تکه‌تکه شدن خارجی در تخصیص پیوسته مطرح شده است.

۲-۳- بررسی روش‌های مهاجرت کار

۲-۳-۱- روش‌های مهاجرت استاندارد

الگوریتم‌هایی همچون انتقال به‌سمت نزدیک‌ترین گوشه توری (*ODC-FC*[۱۳]) [۱۰]، الگوریتم مهاجرت سطری [۹]، الگوریتم مهاجرت دو مرز سطری[۱۴] (*TRBMA*) [۷] و الگوریتم فشرده‌سازی بالا-پایین محدود شده[۱۵] (*LTDC*) [۵]، روش‌های استاندارد نامیده می‌شوند که در مقالاتی که در گذشته بررسی شده‌اند حاکی از توانمندی دو الگوریتم مزبور هستند و به همین دلیل در این مقاله نیز استفاده شده‌اند. در *ODC-FC*، وظایف به‌سمت چهار گوشه از توری مهاجرت داده می‌شوند. بنابراین، یک فضای پیوسته بزرگ از گره‌های آزاد را در مرکز توری ایجاد می‌نماید [۱۰].

الگوریتم مهاجرت سطری برای اتصال گره‌های آزاد که به طور گسسته در فضای توری گسترده شده‌اند جهت حل مشکل تکه‌تکه‌شدن خارجی پیشنهاد شده است. مهاجرت هنگامی که سربار زمان مهاجرت کمتر از متوسط زمان اجرای کارهای در حال اجرا است انجام می‌شود. در الگوریتم مهاجرت سطری تلاش می‌شود تا زیرتوری‌ها به پایین‌ترین سطر در نزدیکی باقی زیرتوری‌های اشغال جابه‌جا شوند [۹]. در *TRBMA*، زیرتوری‌ها به نزدیک‌ترین مرز از بالا و پایین توری جابه‌جا می‌شوند. در الگوریتم مزبور ابتدا فاصله نزدیک‌ترین مرز از بالا و پایین توری برای همه زیرتوری‌ها محاسبه می‌شود تا گره‌های آزاد در میانه توری قرار گیرند. سپس، مینیمم فاصله به‌عنوان نزدیک‌ترین محدوده به مرز برای آن زیرتوری در نظر گرفته می‌شود. سپس، زیرتوری با مینیمم فاصله و مقدار غیر صفر از اتصالات آزاد از گره پایه و انتهایی به‌عنوان زیرتوری نهایی برای مهاجرت انتخاب می‌شود. در نهایت، زیرتوری به نزدیک‌ترین مرز از زیرتوری جابه‌جا می‌شود [۷]. در الگوریتم *LTDC*، جهت مهاجرت زیرتوری مرز بالای توری در نظر گرفته می‌شود. چنانچه امکان‌پذیر بود زیرتوری به مرز بالاترین سطر جابه‌جا می‌شود و در غیر این‌صورت به مرز پایین‌ترین سطر جابه‌جا می‌گردد. لازم به‌ذکر است که فاصله تمام زیرتوری‌های موجود در توری تا مرز بالاترین سطر محاسبه می‌شود، اگر این فاصله صفر باشد جابه‌جایی صورت نخواهد گرفت [۵]. شایان ذکر است که این الگوریتم عملکردی همانند الگوریتم مهاجرت *TRBMA* [۷] دارد با این تفاوت که غالباً مرز بالای توری در نظر گرفته می‌شود. هدف از این استراتژی در مقایسه با الگوریتم *TRBMA*، کاهش تعداد محاسبات جهت مهاجرت است.

۲-۳-۲- الگوریتم‌های مهاجرت ترکیبی

تخصیص به‌طور انحصاری یا غیرانحصاری رخ می‌دهد. چنانچه رویه مهاجرت اتفاق افتد، یک حالت انحصاری رخ می‌دهد و در غیراینصورت آن غیرانحصاری است. در ضمن، سه الگوریتم مهاجرت ترکیبی که در [۷] پیشنهاد شده‌اند نیز در این مقاله در نظر گرفته شده است. استراتژی‌هایی همچون *LTDC&TRBMA*، *TRBMA&ODC-FC* و *LTDC&ODC-FC* مزایای دو الگوریتم مهاجرت را دارا می‌باشند. در واقع، در هر تکرار یکی از الگوریتم‌ها اجرا می‌شود. به‌عنوان مثال، چنانچه تعداد مهاجرت‌های روش *TRBMA&ODC-FC* مساوی با بیست باشد، الگوریتم‌های مهاجرت *TRBMA* [۷] و *ODC-FC* [۱۰] هر کدام ده بار اجرا خواهند شد. در این رویه، یک متغیر جهت کنترل اجرای هر یک از دو الگوریتم مهاجرت در نظرگرفته شده است. در ضمن، ترکیب الگوریتم‌های دیگر نیز رویه مشابهی را در بر دارند.

۲-۴- نگاشت در شبکه‌روی‌تراشه

یکی از مفاهیم اساسی در شبکه های‌روی‌تراشه نحوۀ چیدمان هسته‌ها در همبندی مورد نظر است. نتایج نگاشت تأثیر زیادی بر روی میزان مصرف انرژی، میانگین تأخیر و سایر پارامترهای کیفیت سرویس دارد. اهمیت این موضوع در این‌است‌که در تکنولوژی نانو کاهش میزان مصرف انرژی مهم‌ترین اولویت طراحی است و یک نگاشت مناسب می‌تواند ما را در رسیدن به این هدف یاری نماید [۱۱]. لازم به‌ذکر است که روش‌های نگاشت به دو دسته پویا و ایستا تقسیم می‌شوند. در نگاشت پویا بر خط، تخصیص و ترتیب کارها در طول اجرای برنامه کاربردی انجام می‌شود. در نگاشت پویا

هدف یافتن گلو گاه کارایی و توز یع ح جم کاری در بین پردازنده‌هاست. با توجه به این‌که نگاشت به بار کاری پرداز نده ها وابسته است می‌بایست به راه‌حل بهتری منجر گردد. در ضمن، سربار محاسباتی الگوریتم نگاشت منجر به افزایش تأخیر و ا نرژی مصرفی برنامه کاربردی در طول زمان ا جرا می گردد. در نگا شت ایستا، نگاشت وظایف به‌صورت برون خط قبل از ا جرای برنا مه کاربردی انجام می‌شود. نگاشت ایستا همواره تلاش می‌نما ید تا برای یک کاربرد مشخص و یک ساختار ارت باطی هدف، به ترین گمارش کارها را در زمان طراحی تعریف نماید. در ضمن، الگوریتم نگاشت بعد از تکمیل، تنها یک بار اجرا می‌گردد. با توجه به این‌که سربار ارتباطی در نگاشت پویا بسیار بالا است و کارایی سی ستم را به طور قابل توجهی تحت تأثیر قرار می‌د هد و منجر به ا فزایش تأخیر سیستم می‌گردد، ارائه راهکارهایی در جهت کاهش تأخیر حائز اهمیت است [۱۲]. شایان ذکر است که در این مقاله نگا شت پویا در نظر گرفته شده است.

۳- الگوریتم های ترکیبی پیشنهادی

در این مقاله نه الگوریتم از طرح‌های انتخاب زیرتوری/تخصیص/مهاجرتی که در مقالات پیشین [۶-۸]، به آن‌ها اشاره‌ای نشده است را مورد بررسی قرار داده‌ایم. این طرح‌ها در جدول ۱ مطرح شده‌اند. در ضمن، الگوریتم *TSM*[۱۶] نیز در ادامه پیشنهاد شده است.

۳-۱- الگوریتم مهاجرت سه مرحله ای (*TSM*)

در این الگوریتم، همانطوری که در شکل ۱ ملاحظه می‌گردد، چنانچه تعداد کل مهاجرت‌ها در یک طرح ۳۰ باشد، الگوریتم مهاجرت *LTDC*، ۲۰ بار و الگوریتم *ODC-FC*، ۱۰ بار تکرار می‌شوند. دلیل انتخاب این دو الگوریتم نیز برتری نتایج شبیه‌سازی این دو الگوریتم نسبت به الگوریتم‌های مهاجرت دیگر با توجه به [۵، ۷] است که در این میان چون الگوریتم *LTDC* غالباً عملکرد بهتری نسبت به الگوریتم *ODC-FC* دارد تعداد مهاجرت‌های *LTDC* در این‌جا بیشتر در نظر گرفته شده است. شایان ذکر است که در الگوریتم *TSM*، فراخوانی تابع مهاجرت بر اساس ترتیب هزینه الگوریتم‌های مهاجرت انجام می‌شود. لذا با توجه به اینکه الگوریتم *LTDC* نسبت به الگوریتم *ODC-FC* منجر به تعداد مهاجرت‌های کمتری می‌گردد، در نتیجه در روش *TSM* نیز این الگوریتم تعداد دفعات بیشتری فراخوانی می‌گردد. در ضمن هنگامی‌که فرضاً الگوریتم مهاجرت *LTDC* فراخوانی می‌گردد و این الگوریتم پیوستگی در فضای توری دوبعدی ایجاد می‌نماید در فراخوانی مجدد الگوریتم مهاجرت، دوباره الگوریتم *LTDC* صدا زده نمی‌شود زیرا با فراخوانی الگوریتم مهاجرت دیگر یعنی *ODC-FC*، احتمال پیوسته نمودن فضا بیشتر می‌شود. در ضمن، کنترل تعداد مهاجرت‌ها از طریق متغیر *flag* انجام می‌شود.

۴- بررسی شبیه‌ساز *OM*

در این مقاله جهت ارزیابی الگوریتم پیشنهادی از شبیه‌ساز *OM* که تحت زبان برنامه‌نویسی *C#* می‌باشد استفاده شده است. این شبیه‌ساز شامل چهار گام اساسی انتخاب زیرتوری، نگاشت‌برخط، مهاجرت و تخصیص (به‌صورت انحصاری و غیرانحصاری) است [۵]. در ضمن، پیکربندی شبیه‌ساز بر اساس پارامتر کار (نوع کار، اندازه‌کار، زمان‌زندگی‌کار و زمان‌ورودکار)، پارامتر شبکه (اندازه شبکه، نرخ ارتباطات)، تعداد وظایف و کل زمان شبیه‌سازی می‌باشد [۵، ۶] که در جدول ۲ پیکربندی شبیه‌ساز نشان داده شده است. در ضمن، گراف‌های ورودی شبیه‌ساز بر اساس سناریوی تعریف شده در [۵، ۶] است. لازم به‌ذکر است که در همه شبیه‌سازی‌های انجام شده از روش سوئیچینگ خزشی و مسیریابی *XY* استفاده شده است.

لازم به ذکر است که جهت ارزیابی الگوریتم‌های پیشنهادی سه مرحله محاسبه اندازه زیرتوری مناسب بر اساس اندازه کار ورودی، تخصیص زیرتوری به کار و مهاجرت زیرتوری‌ها انجام شده است. در این راستا جهت محاسبه اندازه زیرتوری مناسب بر اساس اندازه کار ورودی، دو الگوریتم زیرتوری مختلف *MD* [۵، ۶] و *MT&MPN* [۷] در نظر گرفته شده است. جهت تخصیص زیرتوری به کارهای ورودی، الگوریتم‌های تخصیص *ISBA* [۸]، *TRB* [۵] و *TCB* [۶] در شبیه‌ساز *OM* [۵] پیاده‌سازی شده‌اند. در نهایت جهت مهاجرت زیرتوری‌ها، الگوریتم‌های مهاجرت *ODC-FC* [۱۰]، *LTDC* [۵]، *TCB* [۶]، *TRBMA* [۷] و همچنین ترکیبی از الگوریتم‌های مهاجرت مزبور در نظر گرفته شده است. لازم به‌ذکر است که پارامترهایی همچون اندازه تصادفی، زمان ورود و زمان پردازش در کار ورودی در نظر گرفته شده است. همچنین، شرایط ترافیکی یکسانی برای همه شبیه‌سازی‌ها در نظر گرفته شده است. پارامترهای مورد استفاده در این روند شامل میانگین زمان اجرای کار و میانگین بهره‌وری سیستم می‌باشند. زمان اجرای یک کار، زمانی است که کار جهت خاتمه ارتباطات صرف می‌نماید [۱۳]. میانگین بهره‌وری سیستم، متوسط سازش پردازنده‌ها در یک سیستم تا حد امکان شلوغ است که مقداری بین صفر و یک را دارا می‌باشد [۱۴، ۱۵].

جدول ۱: طرح‌های جدید انتخاب زیرتوری/تخصیص/مهاجرت

فهرست	انتخاب زیرتوری/تخصیص/مهاجرت
۱	*MT&MPN [7]/ (TCB/ODC-FC) [6]*
۲	*MD [5, 6]/ TCB [6]/ TRBMA [7]*
۳	*MT&MPN [7]/ TCB [6]/ TRBMA [7]*
۴	*MD [5, 6]/ TCB [6]/ (LTDC&ODC-FC) [7]*
۵	*MT&MPN [8]/ TCB [7]/ (LTDC&ODC-FC) [7]*
۶	*MD [5, 6]/ TCB [6]/ LTDC& TRBMA [7]*
۷	*MT&MPN [7]/ TCB [6]/ LTDC&TRBMA [7]*
۸	*MD [5, 6]/ TCB [6]/ (TRBMA&ODC-FC) [7]*
۹	*MT&MPN [7]/ TCB [6]/ (TRBMA&ODC-FC) [7]*

Algorithm 1 LTDC&ODC-FC&LTDC

```
If Flag = 0 then
    Call LTDC migration method.
    Flag ← 1
Else if Flag = 1 then
    Call ODC-FC method.
    Flag ← 2
Else
    Call LTDC migration method.
    Flag ← 0
End if
```

شکل ۱: شبه کد الگوریتم *TSM*

جدول ۲: پیکربندی شبیه‌سازی، شبیه‌ساز *OM* [۵]

Simulation Parameter	Value
Topology size	16×16, 10×10
Communication rate	1 to 0.1e+4 bit/s
job type	Video & media
job size	Random between 9 to 32 core
job lifetime	Random between 0.1e+6 and 0.1e+7
job arrival time	Random between 0 and 0.3e+6
Total time of the simulation	0.2e+8 cycles
Number of jobs	Random between 0.5e+2 and 0.2e+3

۵- نتایج شبیه‌سازی

۵-۱- ارزیابی میانگین زمان اجرای کار

الگوهای ترافیکی با توجه به پیکربندی شبیه‌ساز در جدول ۲ بیان شده است. به این صورت که ۷۰ کار ورودی با اندازه بین ۹ تا ۳۲ هسته در نظر گرفته شده است. زمان ورود کار به صورت تصادفی بین ۰ تا ۰.۳e+۶، زمان زندگی کار ورودی نیز به صورت تصادفی بین ۰.۱e+۶ تا ۰.۱e+۷ و کل زمان شبیه سازی ۰.۲e+۸ سیکل در نظر گرفته شده است. شکل ۲، میانگین زمان اجرای کار برای الگوهای ترافیکی که ذکر شد را نمایش داده است. بهبودهای کسب شده جهت میانگین زمان اجرای کار با توجه به جدول ۳ دارای مقادیری بین ۳۴.۸٪ تا ۳۸.۱٪ است. با توجه به شکل ۲ و جدول ۳، طرح *MT&MPN/TCB[6]/TRBMA&ODC-FC* با ۳۸.۱٪ دارای بیشترین بهبود است. در واقع، روش مذکور دارای کمترین میانگین زمان اجرای کار نسبت به دیگر طرح‌ها است. این روش جهت تخصیص، زیرتوری‌ها را به ستون‌های مرزی گمارش می‌نماید و جهت مهاجرت آن‌ها را به سطرهای مرزی یا گوشه‌های توری هدایت می‌نماید. در واقع، از قانون‌مندی بالایی جهت تخصیص و مهاجرت زیرتوری‌ها استفاده می‌نماید. در ضمن، زمان اجرای کار با استفاده از رابطه (۴) محاسبه می‌گردد:

$$t_{execution} = t_{wait} + max(t_{mapping}, t_{allocation}) + t_{migration} + t_{process} + t_{maintaindeallocation} \quad (۴)$$

با توجه به رابطه (۴)، t_{wait} بازه زمانی جهت بررسی تعداد گره‌های آزاد در توری جهت تخصیص است. سپس، با استفاده از الگوریتم تخصیص موقعیت گره پایه کسب می‌شود. در نهایت، با استفاده از الگوریتم نگاشت، مکان واقعی گره در توری محاسبه می‌گردد. از آن‌جایی‌که رویه‌های نگاشت و تخصیص همزمان با یکدیگر انجام می‌شوند، در نتیجه بیشینه زمانی که این دو روند متحمل می‌شوند در رابطه (۴) جهت محاسبه زمان اجرای کار در نظر گرفته شده است. سپس، رویه اجرای کار آغاز می‌گردد که در این روند اگر مهاجرتی صورت پذیرد اجرای کار متوقف و بعد از اتمام مهاجرت، کار روند اجرایی خود را ادامه خواهد داد. لازم به‌ذکر است که چنانچه مهاجرتی برای کار صورت نگیرد این زمان صفر در نظر گرفته می‌شود و اگر مهاجرت صورت بگیرد، با توجه به رابطه (۵) $t_{migration}$ شامل ماکزیمم زمان نگاشت و تخصیص به انضمام زمان سوئیچ کاربردها بین گره‌های قبلی و جدید است.

$$t_{migration} = max(t_{mapping}, t_{allocation}) + t_{switch} \quad (۵)$$

شایان ذکر است که در شبیه‌سازی‌ها از $t_{maintaindeallocation}$ صرفنظر شده است و زمان‌سوئیچ نیز صفر در نظر گرفته شده است.

۵-۲- ارزیابی میانگین بهره‌وری سیستم

متوسط استفاده بهینه از سیستم، درصد استفاده از پردازنده‌های سیستم در طول اجرا است که به صورت رابطه (۶) محاسبه می‌شود:

$$\text{SystemUtilization} = \sum_{i=1}^{t} \frac{w \times h - n_i}{(w \times h) \times t} \quad (۶)$$

n_{i} تعداد پردازنده‌های آزاد سیستم در زمان i و t کل زمان صرف شده می‌باشد و $w \times h$ تعداد پردازنده‌های سیستم است. بارگذاری

سیستم نیز یک متغیر مستقل در سیستم است که با متوسط زمان ورود کارها نسبت عکس دارد و محاسبه آن به صورت رابطه (۷) می‌باشد.

$$\lambda = \frac{N \times T_e}{SystemLoad \times P} \quad (۷)$$

در این رابطه P تعداد کل پردازنده‌هاست و کارها با توزیع پواسون و با نرخ λ کار در واحد زمان وارد سیستم می‌شوند. N تعداد متوسط پردازنده‌های درخواست شده توسط هر کار است. T_e توزیع توانی متوسط زمان اجرا می‌باشد [۱۶].

میانگین بهره‌وری سیستم برای همه طرح‌های نگاشت در معماری شبکه‌روی‌تراشه در شکل ۳ نشان داده شده است. الگوریتم *MT&MPN/TCB[6]/TRBMA* با مقدار ۰.۹ بهترین بهره‌وری را در میان باقی روش‌ها دارد. با توجه به جدول ۴، روش مذکور نسبت به الگوریتم (*ISBA/...*) در [۵]، ٪۴۸.۵ بهبود داشته است. در ضمن، بهبودهای کسب شده جهت پارامتر مزبور در بازه ٪۲۶.۴ تا ٪۴۸.۵ قرار دارند.

شایان ذکر است که تمام بهبودهای کسب شده در جدول‌های ۳ و ۴ نسبت به الگوریتم‌هایی که بدترین نتایج را نسبت به سایر روش‌ها کسب نموده اند در نظر گرفته شده است. لذا درصد بهبودها جهت میانگین زمان اجرای کار نسبت به الگوریتم (*MT&MPN/ISBA/...*) [۷] و میانگین بهره‌وری سیستم نسبت به الگوریتم (*MD/ISBA/...*) [۵، ۶] نمایش داده شده است. لذا میانگین زمان اجرای کار و میانگین بهره‌وری سیستم برای الگوریتم *MT&MPN[7]/TCB[6]/TSM* طبق روابط (۸) و (۹) به ترتیب ٪۳۶ و ٪۴۴.۳ بهبود کسب شده است.

$$\text{محاسبه کارایی } AJET = \frac{\text{نتیجه ارزیابی روش i} - \text{نتیجه ارزیابی بدترین روش}}{\text{نتیجه ارزیابی بدترین روش}} \times ۱۰۰ \quad (۸)$$

$$= \frac{8739.3 - 5589.7}{8739.3} \times 100 = \%36.03 \sim \%36$$

$$\text{محاسبه کارایی } ASU = \frac{\text{نتیجه ارزیابی بدترین روش} - \text{نتیجه ارزیابی روش } i}{\text{نتیجه ارزیابی روش } i} \times ۱۰۰ \quad (۹)$$

$$= \frac{0.8687 - 0.4839}{0.8687} \times 100 = \%44.29 \sim \%44.3$$

۵-۳- تحلیل انرژی مصرفی

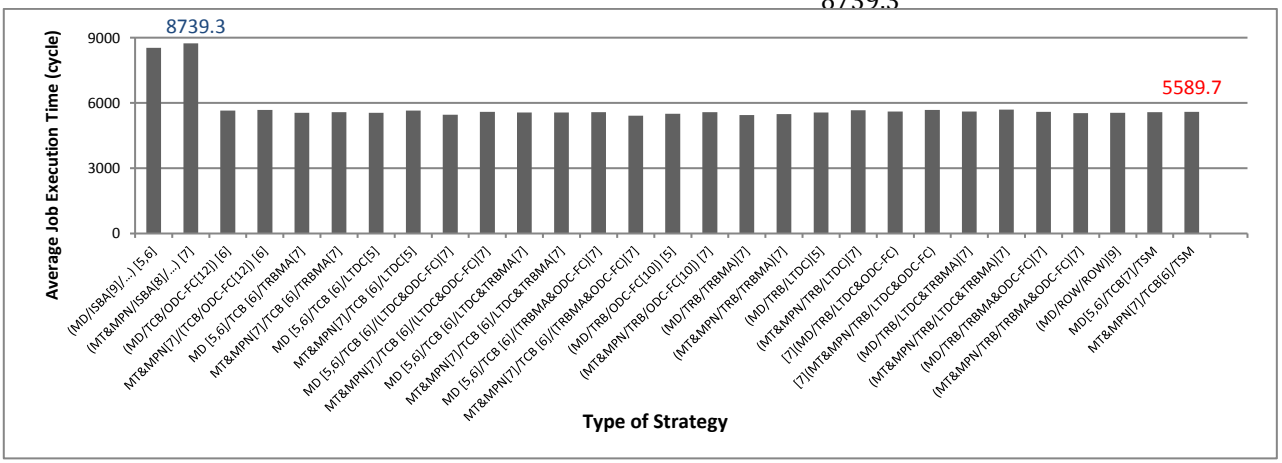
تأخیر و انرژی مصرفی شامل تأخیر پردازش $T_{Process}$ و ارتباطات $T_{Comunication}$ است که در روابط (۱۰) و (۱۱) ارائه شده است. در روش پیشنهادی که برای نگاشت چند کاربرد پیشنهاد شده است، اولاً با استفاده از توابع تخصیص ناحیه و نگاشت داخل ناحیه غالباً کمترین قطر انتخاب می‌شود تا تأخیر ارتباطات حداقل شود. ثانیاً با استفاده از تابع تخصیص و مهاجرت سعی می‌شود بیشتر کاربردهای ممکن همزمان اجرا شوند و منجر به کاهش زمان انتظار و در نتیجه کاهش زمان پردازش می‌شود. لذا با کاهش زمان ارتباطات و زمان پردازش زمان اجرای کاربردها نیز کاهش می‌یابد که تأثیر مستقیم روی انرژی مصرفی دارد. در مجموع، به‌منظور کاهش انرژی مصرفی، توان پردازش، توان ارتباطات و یا زمان اجرای کارها می‌بایست کاهش یابد. با توجه به رابطه (۱۱) با کاهش زمان اجرا، انرژی مصرفی نیز کاهش می‌یابد [۱۷].

$$T_{Execution} = T_{Comunication} + T_{Process} \quad (10)$$

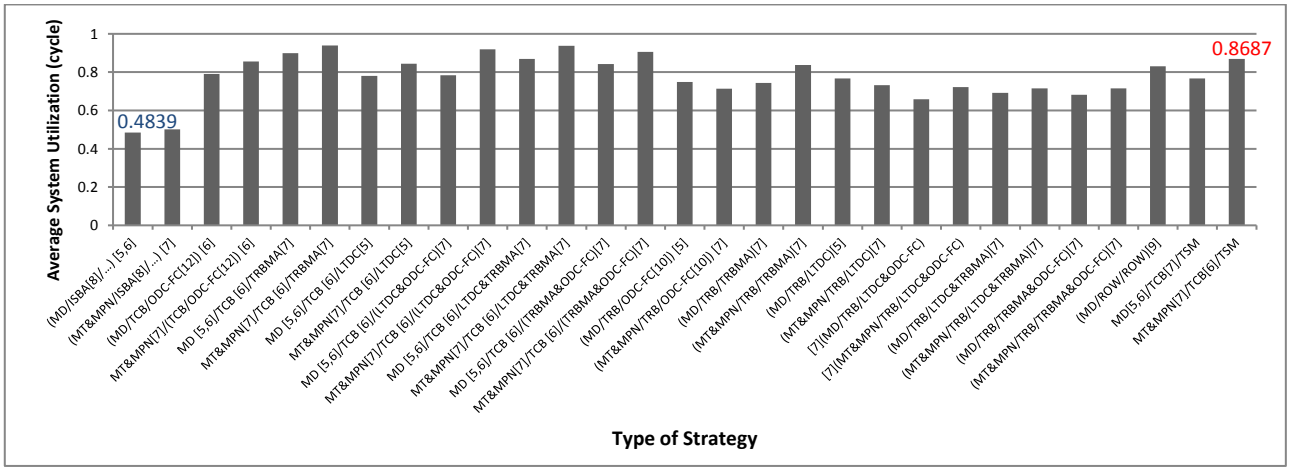
$$E = T_{Execution} . P = T_{Comunication} . P_{Comunication} + T_{Process} . P_{Process} \quad (11)$$

۵-۴- ارزیابی هفت روش انتخابی

با توجه به جدول‌های ۳، ۴ و بررسی بیست و نه طرح زیرتوری/تخصیص/نگاشت ملاحظه می‌گردد که هفت روش نتایج بهتری را نسبت به الگوریتم‌های دیگر با توجه به دو پارامتر کارایی کسب نموده‌اند که این نتایج در جدول ۵ لیست شده است.



شکل ۲: میانگین زمان اجرای کار برای طرح‌های مختلف زیرتوری/تخصیص/مهاجرت



شکل ۳: میانگین بهره‌وری سیستم برای طرح‌های مختلف زیرتوری/تخصیص/مهاجرت

جدول ۳: بهبودهای کسب شده جهت میانگین زمان اجرای کار برای طرح‌های مختلف انتخاب زیرتوری/تخصیص/مهاجرت

فهرست	مهاجرت/تخصیص/زیرتوری	AJET (%)
۱	(MD/ISBA [8]/…) [5,6]	2.4
۲	(MT&MPN/ISBA [8]/…) [7]	0
۳	(MD/TCB/ODC-FC[10]) [6]	35.4
۴	MT&MPN[7]/(TCB/ODC-FC[10]) [6]	35
۵	MD[5,6]/TCB [6]/TRBMA [7]	36.5
۶	MT&MPN[7]/TCB [6]/TRBMA[7]	36.2
۷	MD[5,6]/TCB[6]/LTDC[5]	36.5
۸	MT&MPN[7]/TCB [6]/LTDC[5]	35.3
۹	MD[5,6]/TCB[6]/(LTDC&ODC-FC)[7]	37.5
۱۰	MT&MPN[7]/TCB[6]/(LTDC&ODC-FC)[7]	36.1
۱۱	MD[5,6]/TCB[6]/(LTDC&TRBMA)[7]	36.4
۱۲	MT&MPN[7]/TCB[6]/(LTDC&TRBMA)[7]	36.3
۱۳	MD[5,6]/TCB [6]/(TRBMA&ODC-FC)[7]	36.1
۱۴	MT&MPN[7]/TCB[6]/(TRBMA&ODC-FC)[7]	38.1
۱۵	(MD/TRB/ODC-FC[10]) [5]	37
۱۶	(MT&MPN/TRB/ODC-FC[10]) [7]	36.2
۱۷	(MD/TRB/TRBMA)[7]	37.7
۱۸	(MT&MPN/TRB/TRBMA)[7]	37.2
۱۹	(MD/TRB/LTDC)[5]	36.3
۲۰	(MT&MPN/TRB/LTDC)[7]	35.2
۲۱	(MD/TRB/LTDC&ODC-FC)[7]	35.8
۲۲	(MT&MPN/TRB/LTDC&ODC-FC)[7]	35.1
۲۳	(MD/TRB/LTDC&TRBMA)[7]	35.8
۲۴	(MT&MPN/TRB/LTDC&TRBMA)[7]	34.8
۲۵	(MD/TRB/TRBMA&ODC-FC)[7]	36.1
۲۶	(MT&MPN/TRB/TRBMA&ODC-FC)[7]	36.6
۲۷	(MD/ROW/ROW)[9]	36.5
۲۸	MD[5,6]/TCB[6]/TSM	36.1
۲۹	MT&MPN[7]/TCB[6]/TSM	36

جدول ۴: بهبودهای کسب شده جهت میانگین بهره‌وری سیستم برای طرح‌های مختلف انتخاب زیرتوری/تخصیص/مهاجرت

فهرست	مهاجرت/تخصیص/زیرتوری	ASU (%)
۱	(MD/ISBA [8]/…) [5,6]	0
۲	(MT&MPN/ISBA [8]/…) [7]	3.4
۳	(MD/TCB/ODC-FC[10]) [6]	38.7
۴	MT&MPN[7]/(TCB/ODC-FC[10]) [6]	43.4
۵	MD[5,6]/TCB [6]/TRBMA [7]	46.1
۶	MT&MPN[7]/TCB [6]/TRBMA[7]	48.5
۷	MD[5,6]/TCB[6]/LTDC[5]	37.9
۸	MT&MPN[7]/TCB [6]/LTDC[5]	42.7
۹	MD[5,6]/TCB[6]/(LTDC&ODC-FC)[7]	38.2
۱۰	MT&MPN[7]/TCB[6]/(LTDC&ODC-FC)[7]	47.4
۱۱	MD[5,6]/TCB[6]/(LTDC&TRBMA)[7]	44.3
۱۲	MT&MPN[7]/TCB[6]/(LTDC&TRBMA)[7]	48.4
۱۳	MD[5,6]/TCB [6]/(TRBMA&ODC-FC)[7]	42.5
۱۴	MT&MPN[7]/TCB[6]/(TRBMA&ODC-FC)[7]	46.5
۱۵	(MD/TRB/ODC-FC[10]) [5]	35.3
۱۶	(MT&MPN/TRB/ODC-FC[10]) [7]	32.2
۱۷	(MD/TRB/TRBMA)[7]	34.9
۱۸	(MT&MPN/TRB/TRBMA)[7]	42.2
۱۹	(MD/TRB/LTDC)[5]	36.9
۲۰	(MT&MPN/TRB/LTDC)[7]	33.8
۲۱	(MD/TRB/LTDC&ODC-FC)[7]	26.4
۲۲	(MT&MPN/TRB/LTDC&ODC-FC)[7]	32.9
۲۳	(MD/TRB/LTDC&TRBMA)[7]	30
۲۴	(MT&MPN/TRB/LTDC&TRBMA)[7]	32.4
۲۵	(MD/TRB/TRBMA&ODC-FC)[7]	29
۲۶	(MT&MPN/TRB/TRBMA&ODC-FC)[7]	32.4
۲۷	(MD/ROW/ROW)[9]	41.7
۲۸	MD[5,6]/TCB[6]/TSM	36.9
۲۹	MT&MPN[7]/TCB[6]/TSM	44.3

جدول ۵: بهترین طرح‌های منتخب جهت بهبود کارایی در نگاشت برخط

فهرست	مهاجرت/تخصیص/زیرتوری	میانگین زمان اجرای کار (٪)	میانگین بهره‌وری سیستم (٪)
۱	MT&MPN[7]/TCB[6]/(LTDC&ODC-FC) [7]	36.1	47.4
۲	MT&MPN[7]/TCB[6]/(TRBMA&ODC-FC) [7]	38.1	46.5
۳	MT&MPN[7]/TCB[6]/TRBMA[7]	36.2	48.5
۴	MD[5, 6]/TCB[6]/(LTDC&ODC-FC) [7]	37.5	38.2
۵	MT&MPN[7]/TCB[6]/ LTDC&TRBMA[7]	36.3	48.4
۶	MD[5, 6]/TCB[6]/LTDC&TRBMA[7]	36.4	44.3
۷	MT&MPN[7]/TCB[6]/TSM	36	44.3

با توجه به جدول ۵ ملاحظه می‌گردد که الگوریتم‌های پیشنهادی در این مقاله غالباً نتایج بهتری را نسبت به باقی روش‌های انتخابی کسب نموده‌اند.

۶- نتیجه گیری

در این مقاله جهت نگاشت کارهای موازی بر روی یک تراشه چندپردازنده‌ای، الگوریتم‌های مبتنی بر انتخاب ابعاد زیرتوری، تخصیص کار و مهاجرت کار به‌عنوان مفاهیم مورد استفاده در این راستا مطرح و به‌اجمال بررسی شده‌اند. شایان ذکر است که روش‌های تخصیص پیوسته منجر به تکه‌تکه‌شدن خارجی و الگوریتم‌های مهاجرت پردازنده جهت حل این چالش در شبکهٔ پردازنده‌ها مطرح شده‌اند [۳]. ضرورت مقایسه‌های انجام شده در کارهای انجام شده پیشین این است که به ارائه راهکاری مؤثر جهت بهبود کارایی در شبکه‌های روی تراشه رهنمون گردیم. در این راستا، روش‌های مؤثری جهت مهاجرت کارها بیان نمودیم تا پیچیدگی محاسباتی و سربار زمان اجرای الگوریتم‌های پیشنهادی را در مقایسه با الگوریتم‌های مهاجرت کار پیشین بهبود دهیم.

مراجع

[1] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design", *Journal of Systems Architecture*, Vol. 59, pp. 60–76, 2013.

[2] S. Bani-Mohammad, I. Ababneh, "Improving system performance in non-contiguous processor allocation for mesh interconnection networks", *Simulation Modelling Practice and Theory*, Vol. 80, pp. 19-31, 2018.

[3] R. Zolfaghari, "Efficientand Quick Algorithm for Processor Allocation in Mesh Multi-Computers Network", *International Journal of Engineering and Advanced Technology (IJEAT)*, Vol. 2, No. 5, 2013.

[4] D. D. Sharma, and D. K. Pradhan, "A fast and efficient strategy for submesh allocation in mesh-connected parallel computers", *in Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Processing*, pp. 682–689, 1993.

[5] A. Reza and M. Rafie, "Performance Improvement in Multiprocessors Using Two Row Boundary Allocation Method and Online Dynamic Compaction Algorithm", *International journal of computer applications (IJCA)*, Vol. 123, No.1, pp. 14-20, USA, 2015.

[6] A. Reza and M. Rafie, "Improving Multi Task Running Time in Two Column Boundary Allocation Method in Mesh-based Chip Multiprocessors Using Combined Migration Mechanisms", *International Journal of Computer & Information Technologies (IJOCIT)*, Vol. 3, No. 3, pp. 743-755, 2015.

[7] M. Rafie, A. Khademzadeh, A. Reza, M. Reshadi, "Performance Evaluation of Task Migration in Contiguous Allocation for Mesh Interconnection Topology", *The Journal of Supercomputing*, Vol. 72, No. 4, pp. 1660-1677, 2016.

[8] G. Chmaj, D. Zydek, and L. Koszalka, "Allocation Algorithms Problems in Mesh-Connected Systems", *2nd Student's Science Conference*, 2004.

[9] A. Reza, and R. Faghih Mirzaee, "Non-preemptive offline multi-job mapping for a photonic network on a chip", *Nano Communication Networks*, Vol. 11, pp. 11-23, 2017.

[10] L. K. Goh and B. Veeravalli, "Design and performance evaluation of combined first-fit task allocation and migration strategies in mesh multicomputer systems", *Journal of Parallel Computing*, Vol. 34, No. 9, pp. 508–520, 2008.

[11] W. T. Shen, C. H. Chao, Y. K. Lien, and A.Y. Wu, "A New Binomial Mapping and Optimization Algorithm for Reduced-Complexity Mesh-Based On-Chip Network", *in Proceedings of the First International Symposium on Networks-on-Chip (NOC '7)*, pp. 317-322, 2007.

[12] F. Moosavi, M. rafie, and D. Zeinalabedini, "Dynamic and Static Mapping Methods in Two-Dimensional Network on Chips", *International Journal of Computer & Information Technologies (IJOCIT)*, Vol. 4, No. 2, pp. 23-33, 2016.

[13] S. Bani-Mohammad, "On the Performance of Job Scheduling for Noncontiguous Allocation in 2D Mesh-connected Multicomputers", *in Proceedings of the 16th IEEE Mediterranean Electrotechnical Conference (MELECON)*, pp. 92-96, 2012.

[14] Y. Zhu, "Efficient processor allocation strategies for mesh-connected parallel computers", *Journal of Parallel and Distributed Computing*, Vol. 16, No. 4, pp. 328–337, 1992.

[15] S. Bani-Ahmad, "On Improved Processor Allocation in 2D Mesh-based Multicomputers: Controlled Splitting of Parallel Requests", *in Proceedings of the 2011 International Conference on Communication*, Computing and Security (ICCCS'11), pp. 204-209, 2011.

[16] S. B. Ahmad, "Bounded Gradual-Request-Partitioning-Based Allocation Strategies in 2D-Mesh Multicomputers", *International Journal of Digital Content Technology and its Applications*, Vol. 5, No. 1, 2011.

[17] J. Fang, H. Zong, H. Zhao, and H. Cai, "Intelligent Mapping Method for Power Consumption and Delay Optimization Based on Heterogeneous NoC Platform", *Electronics*, Vol. 9, No. 912, 2019.

پاورقی‌ها

[1] Network on Chip
[2] Offline mapping
[3] Online mapping
[4] External fragmentation
[5] Online offline mapping
[6] Minimum Diameter
[7] Minimum Threshold and Minimum Prime Number
[8] Minimum Prime Number
[9] Improved Stack Based Algorithm
[10] Two Row Boundary
[11] Two Column Boundary
[12] ROW
[13] Online Dynamic Compaction-Four Corner
[14] Two Row Boundary Migration Algorithm
[15] Limited Top Down Compaction
[16] Three Steps Migration