سال چهاردهم، شماره‌های ۵۱ و ۵۲ ، بهار و تابستان ۱۴۰۱

صص: ۲۳۱_۲۴۴

دو فصلنامه علمی

فناوری اطلاعات و ارتباطات ایران

کاهش بار شبکه با نگاشت برنامه کابردی در شبکه روی تراشه با استفاده از الگوریتم شاهین هریس گسسته

الهام حاجبی* وحید ستاری نائینی**

* دانشجوی کارشناسی ارشد، بخش مهندسی کامپیوتر، دانشگاه شهید باهنر کرمان، کرمان، ایران

** دانشیار، بخش مهندسی کامپیوتر، دانشگاه شهید باهنر کرمان، کرمان، ایران

تاریخ دریافت: ۱۳۹۹/۱۲/۰۴ تاریخ پذیرش: ۱۴۰۰/۰۶/۰۳

نوع مقاله: پژوهشی

چکیده

کاهش بار و مصرف انرژی در سیستم‌های شبکه روی تراشه از اهمیت بسیاری برخوردار است و یکی از مهم‌ترین مباحثی که برای افزایش کارایی شبکه روی تراشه مطرح است، موضوع نگاشت یک برنامه کاربردی در شبکه روی تراشه است. حل مسئله نگاشت برنامه کاربردی برای یافتن بهترین نگاشت، یک موضوع پیچیده و زمانبر است و تاثیر بسیار زیادی بر تأخیر و انرژی مصرفی شبکه دارد. در این مقاله با استفاده از الگوریتم شاهین هریس توانسته‌ایم روشی را برای نگاشت هسته‌های پردازشی به روی شبکه روی تراشه ارائه کنیم تا بار روی شبکه و در نتیجه ازدحام در لینک‌ها را کاهش داده و عملکرد شبکه بهبود ببخشیم. نتایج شبیه‌سازی نشان می‌دهد که این الگوریتم عملکرد بهتری در مقایسه با الگوریتم‌های پایه دارد.

واژگان کلیدی: شبکه روی تراشه- نگاشت هسته پردازشی - تأخیر شبکه - بهره‌وری از لینک - الگوریتم شاهین هریس گسسته

۱. مقدمه

در سیستم روی تراشه[۱] برای ارتباط میان اجزای سیستم، از تکنولوژی گذرگاه مشترک استفاده می‌کردند، با افزایش تعداد هسته-های پردازشی روی تراشه و کوچک‌تر شدن اندازه نیمه‌هادی‌ها، تکنولوژی گذرگاه مشترک نمی‌توانست از پهنای باند ارتباطی میان هسته‌ها پشتیبانی کند. شبکه روی تراشه[۲] به عنوان یک بستر ارتباطی قابل استفاده مجدد و مقیاس‌پذیر برای سیستم‌های روی

نویسنده مسئول: وحید ستاری نائینی vsnaeini@uk.ac.ir

[۱] System-on-Chip (SoC)

[۲] Network-on-Chip (NoC)

تراشه پیشنهاد شد. NoC اجازه می‌دهد تعداد زیادی از هسته‌ها در توپولوژی‌های مختلف شبکه متصل شوند و بسته‌ها را از طریق مسیریاب‌ها و لینک‌ها تحت توپولوژی به کار رفته، در شبکه منتقل کنند و همچنین امکان ارتباطات موازی میان هسته‌ها در NoC میسر می‌شود. توپولوژی‌های محبوب NoC که توسط محققان مورد استفاده قرار می‌گیرند عبارتند از: حلقه، مش، توروس، درخت و مکعب است.

شبکه روی تراشه جزئی جدایی ناپذیر از سیستم‌های چند پردازنده-ای است، بنابراین عملکرد شبکه روی تراشه مستقیماً بر عملکرد پردازنده‌ها تأثیر می گذارد[۱]. در شبکه روی تراشه چالش‌هایی مطرح شده است به طور مثال، پهنای باند ارتباطی بالا در شبکه‌های روی تراشه به معنی واقعی به ارتباط میان هسته‌های پردازشی مربوط می‌شود. افزایش یافتن پهنای باند ارتباطی، مصرف توان و تأخیر انتقال به عنوان یک گلوگاه در شبکه روی تراشه مطرح شد. یک راه حل آن استفاده از شبکه‌های روی تراشه نوری است[۲] اما راه حل دیگر، کم کردن حجم ارتباطی میان هسته‌های پردازشی است و این راه حل با نگاشت وظایف برنامه‌کاربردی روی هسته‌های پردازشی انجام می‌گیرد و خود نگاشت نیز یکی از چالش‌های مطرح در سیستم‌های چند پردازنده‌ای مبتنی بر NoC است در نتیجه، عملکرد و کارایی سیستم را تحت تأثیر قرار می‌دهد. نگاشت وظایف به دو صورت استاتیک یا پویا انجام می‌شود. نگاشت پویا در زمان اجرا انجام می‌شود و زمان‌بر می‌باشد. نگاشت استاتیک در زمان طراحی انجام و برای سیستم‌های چندپردازنده‌ای این نوع نگاشت توصیه می‌شود [۳]. نگاشت به عنوان یک مشکل رده سخت شناخته شده است. روش‌های مختلفی برای نگاشت برنامه کاربردی روی NoC ارائه شده است که تعدادی از این روش‌ها به صورت دسته-بندی شده در مرجع[۳] ذکر شده و هر کدام از این روش‌ها معیارهای متفاوتی در نظر گرفته‌اند. بیشتر روش‌هایی که انجام شده و در ادامه به آن‌ها می‌پردازیم، هدف کاهش سربار ارتباطی و مصرف انرژی سیستم روی تراشه بوده است. برای مثال، در مرجع [۴]، برای کاهش سربار ارتباطی، مصرف انرژی ارتباطی و ازدحام شبکه برای MPSoCs مبتنی بر NoC، یک نگاشت پویا اکتشافی وظایف را پیشنهاد می‌کند و تلاش می‌کند تا چندین وظیفه‌ی ارتباطی را به همان و یا نزدیک‌ترین عنصرهای پردازشی نگاشت کند. بااین‌وجود، روش پیشنهادی بار را در بین عنصرهای پردازشی موجود متعادل نمی‌کند. علاوه بر این روش پیشنهادی، بهینه‌سازی کلی سربارهای ارتباطی را تضمین نمی‌کند، زیرا تنها گره‌های همسایه اصلی و فرعی برای قرار دادن در همان عنصر پردازشی در نظر گرفته می‌شود. این تجزیه‌وتحلیل بر روی شبکه روی تراشه با توپولوژی مش ۸ × ۸ و عناصر پردازشی همگن انجام‌شده است. برای غلبه بر مشکلات فوق، در مرجع [۵] یک روش پیش‌پردازشی پیشنهاد دادند که تلاش می-کند تا سربارهای ارتباطی را کاهش دهد و بار را در بین عنصرهای پردازشی موجود به‌طور مساوی متعادل کند. روش پیشنهادی، زمان اجرای برنامه کاربردی، مصرف انرژی و استفاده از منابع را بهبود می‌بخشد. با این حال، روش پیش‌پردازش منجر به محاسبات اضافی می‌شود و نتایج را در زمان اجرای الگوریتم افزایش می‌دهد. علاوه بر این، تجزیه‌وتحلیل با تعداد محدودی از وظایف بر روی شبکه‌های روی تراشه کوچک‌تر انجام‌شده است.

در مرجع [۶] با ترکیب روش‌های جستجوی ممنوعه[۳] و مبادله هسته‌ها بر اساس حجم ارتباطی و بهینه‌سازی ازدحام ذرات گسسته[۴] (DPSO) یک نگاشت برای هسته‌ها روی کاشی‌های NoC ارائه شده است. هدف الگوریتم تاخیر ارتباطی حداقل شود. فضای جستجو نشان‌دهنده تمام ترکیبات هسته‌ای است که به کاشی‌های مختلف NoC اختصاص داده شده است. DPSO به‌عنوان روش اصلی بهینه‌سازی استفاده می‌شود و در آن حرکت ازدحامی ذرات تحت تاثیر بهترین مکان محلی[۵]، بهترین مکان سراسری[۶] و بهینه‌سازی حجم ارتباطات است. در این روش از یک لیست جهت ممنوع کردن نقاط جستجویی که ذرات قبلا بازدید کرده‌اند، استفاده می‌شود. این روش هزینه ارتباطی، زمان تاخیر و انرژی مصرفی را کاهش می‌دهد. عملکرد این روش ترکیبی بهتر از روش DPSO است.

---

۳ Tabu-search

۴ Discrete Particle Swarm Optimization

۵ local best

۶ global best

در مرجع [۷] یک الگوریتم آنلاین انرژی آگاه برای نگاشت وظایف روی NoC پیشنهاد می‌کند. هدف الگوریتم کاهش مصرف انرژی ارتباطی است. اولین گام الگوریتم این است که وضعیت ارتباطات برنامه‌های کاربردی را در زمان اجرا تجزیه‌وتحلیل کند و وظیفه‌ی با بالاترین میزان ارتباط انتخاب می‌کند. وظیفه انتخاب‌شده برای اولین بار نگاشت می‌شود. در مراحل بعدی، وظایف همسایه‌ی وظیفه انتخاب‌شده، بر اساس حجم ارتباطی مرتب شده و به نزدیک‌ترین عناصر پردازشی نگاشت می‌شوند. به عنوان یک نتیجه از نگاشت انجام‌شده توسط این الگوریتم، وظایف با حجم ارتباطی زیاد نزدیک‌تر به یکدیگر قرار می‌گیرند، اما وظایفی که حجم ارتباطی کمی دارند ممکن است نادیده گرفته و از هم جدا شوند.

در مرجع [۸] یک الگوریتم نگاشت برنامه کاربردی با توجه به رقابت NoC پیشنهاد کردند. الگوریتم پیشنهاد شده در دو مرحله کار می‌کند. در مرحله اول، یک منطقه مستطیلی از اندازه مورد نیاز برای نگاشت یک برنامه کاربردی ورودی شناسایی شده است. در مرحله بعدی، وظایف برنامه کاربردی به عنصرهای پردازشی در منطقه مشخص‌شده، با توجه به حجم ارتباطی لبه‌های متصل به وظایف وابسته اختصاص داده می‌شود. نتایج تجربی نشان‌دهنده برتر بودن الگوریتم پیشنهاد شده در مقایسه با الگوریتم‌های FF و NN است. هدف ما در این مقاله، با کاهش حجم ارتباطی روی لینک‌های میان مسیریاب‌ها در شبکه روی تراشه، ازدحام شبکه تحت تأثیر قرار می‌گیرد. هرچه حجم ارتباطی میان هسته‌ها کمتر باشد، احتمال ازدحام در شبکه کمتر خواهد شد. به همین دلیل ما از الگوریتم VEBAP [۷] که از روش بسته‌بندی[۸] استفاده می‌کنیم و در مرجع [۹] ارائه شده است، برای نگاشت وظایف روی هسته‌های پردازشی استفاده کرده‌ایم تا بیشترین بهره‌وری را از منابع داشته باشیم. وظایفی که حجم ارتباطی میان آن‌ها زیاد است با قرار دادن در یک bin (اصطلاح bin در این روش به جای هسته‌های پردازشی استفاده می‌شود.)، می‌توان حجم ارتباطی روی لینک‌ها را کاهش داد. سپس با استفاده از الگوریتم شاهین هریس[۱۰] و تابع هدفی که برای این الگوریتم در نظر گرفتیم، قصد داریم به نگاشتی با انرژی مصرفی کمتر و همچنین پراکندگی بار کمتر برسیم. انحراف معیار از جمله معیارهای پراکندگی که در این کار استفاده شده است. انحراف معیار میزان پراکندگی در تمام داده‌ها اما دامنه چارکی میزان پراکندگی در نیمی از داده‌های میانی محاسبه می‌کند.

بخش‌بندی این مقاله به شرح زیر است. بخش ۲ کارهای انجام شده در خصوص نگاشت آورده شده است. در بخش ۳ مدل شبکه روی تراشه و گراف وظایف توصیف کردیم. و در بخش ۴ به روش پیشنهادی ارائه شده می‌پردازیم و در نهایت در بخش ۵ محیط شبیه‌سازی و نتایج حاصل از این مقاله شرح می‌دهیم و نتیجه حاصل از کار ارائه شده در بخش ۶ آورده‌ایم.

۲. پیش زمینه

بسیاری از مسائل دنیای واقعی در حوزه یادگیری ماشین و هوش مصنوعی به طور کلی ماهیت پیوسته، گسسته، محدود یا نامحدود دارند[۱۱, ۱۲]. با توجه به این ویژگی‌ها، نمی‌توان با استفاده از روش‌های گرادیان مزدوج، برنامه‌ریزی درجه دوم متوالی و روش‌های شبه نیوتن، به حل بعضی از این مسائل پرداخت[۱۳]. زیرا این روش‌ها به اندازه کافی کارآمد نبودند. بر این اساس الگوریتم‌های فراابتکاری به عنوان یک روشی برای حل این مسائل، طراحی و مورد استفاده قرار گرفته‌اند. مزایای این الگوریتم‌ها، سادگی و روند اجرای آن‌ها در حل مسائل بهینه‌سازی است و معایب برخی از این الگوریتم‌ها، نسبت به تنظیم پارامترها حساس می‌باشند و ممکن است همیشه به جواب بهینه سراسری همگرا نباشند [۱۴].

به طور کلی، دو نوع الگوریتم فراابتکاری داریم [۱۵]؛ نوع اول مبتنی بر یک راه حل مانند شبیه‌سازی تبریدی [۱۶]، نوع دوم مبتنی بر جمعیت مانند الگوریتم ژنتیک است [۱۷]. در نوع اول تنها یک راه حل در مرحله بهینه‌سازی پردازش می‌شود، در حالی‌که در نوع دوم، شامل مجموعه‌ای از راه حل‌ها (یعنی جمعیت) در هر تکرار از روند بهینه‌سازی می‌شود. روش‌های مبتنی بر جمعیت اغلب یک راه حل بهینه یا نزدیک به آن را پیدا می‌کنند که یا همان جواب دقیق مسئله می‌باشد که به آن جواب بهینه سراسری و یا جوابی نزدیک به آن،

[۷] Variable bEnefit Bin pAcking Problem

[۸] bin packing

بهینه محلی گویند. الگوریتم‌های فراابتکاری مبتنی بر جمعیت عمدتاً از پدیده‌های طبیعی تقلید می‌کنند[۱۸, ۱۹]. این الگوریتم‌ها فرآیند بهینه‌سازی را با تولید مجموعه‌ای از افراد (جمعیت) شروع می‌کنند، به طوری که هر یک از افراد در جامعه نماینده یک راه حل انتخابی برای مسئله بهینه‌سازی هستند. با جایگزینی جمعیت فعلی با جمعیت تازه تولید شده با استفاده از برخی از عملگرهای غالباً تصادفی، این جمعیت به طور تکراری تکامل می‌یابد[۲۰]. فرآیند بهینه‌سازی تا رسیدن به یک معیار توقف (یعنی حداکثر تعداد تکرارها) انجام می‌شود[۲۱].

تمام الگوریتم‌های فرااکتشافی دارای یک ویژگی مشترک هستند: مراحل جستجو دارای دو مرحله اکتشاف و بهره‌برداری است [۲۲]. هر الگوریتم تکاملی جدیدی که معرفی می‌شود برای اثبات قدرت خود در بهینه‌سازی باید بر روی دو مرحله مهم یعنی اکتشاف (Exploration) و بهره‌برداری (Exploitation) تمرکز خوبی داشته باشد و بتواند تعادل خوبی بین این دو مرحله ایجاد کند. سپس بر روی مسائل معیار مختلف آزمایش شود و نتایج آن با الگوریتم‌های قدیمی‌تر مقایسه شود.

ایده اصلی الگوریتم HHO، رفتار مشارکتی و سبک تعقیب شاهین هریس در طبیعت است، که به نام حمله غافلگیرانه شناخته می‌شود. در این راه‌کار هوشمندانه، چندین شاهین هریس به صورت مشارکتی و با همکاری همدیگر تلاش می‌کنند تا یک طعمه را از جهت‌های مختلف غافلگیر کنند. شاهین هریس، می‌تواند الگوهای تعقیب و گریز متنوعی را براساس ماهیت پویای سناریوها و الگوهای شکار از خود نشان دهد. این الگوریتم به خوبی تعادل بین مراحل اکتشاف و بهره‌برداری را ایجاد می‌کند[۱۰]. از دیگر مزایای این الگوریتم، نیاز به تنظیم پارامتر همانند الگوریتم‌هایی چون PSO، TLBO و ... ندارد و همچنین، نتایج این الگوریتم نسبت به الگوریتم‌های قدیمی روی مسائل معیار بهتر بوده است. به همین دلیل ما از الگوریتم HHO استفاده کرده‌ایم.

شبه کد الگوریتم شاهین هریس در شکل ۱ آورده شده است.

الگوریتم شاهین هریس (HHO)

```
Inputs: The population size N and maximum number of iterations T

Outputs: The location of rabbit and its fitness value

Initialize the random population Xi(i=۱،۲،۳,…,N)
While (stopping condition is not met) do
        Calculate the fitness values of hawks
        Set Xrabbit as the location of rabbit (best location)
        For (each hawk (Xi)) do
            Update the initial energy E۰ and jump strength J▷ E۰=۲rand()-۱, J=۲(۱-rand())
        Update the E using Eq.۲
        If (|E|<۱) then▷ Exploration phase
           Update the location vector using Eq.۱
        If (|E|≥۱) then ▷ Exploration phase
          If (r≥ ۰،۵ and |E|≥۰،۵) then▷ Soft besiege
                Update the location vector
          else If (r≥ ۰،۵ and |E|<۰،۵) ⊳hen Hard besiege
                Update the location vector
        else If (r< ۰،۵ and |E|≥۰،۵) then▷ Soft besiege with progressive rapid dives
                Update the location vector
           else If (r< ۰،۵ and |E|< ۰،۵) then▷ Hard besiege with progressive rapid dives
                Update the location vector
Return Xrabbit
```

شکل ۱. شبه کد الگویتم شاهین هریس[۱۰]

۳. مدل سیستم ارائه شده

۱،۳ مدل NoC

یک گرافی است که با NG=(P,L) نشان داده می‌شود. P شامل مجموعه‌ای از کاشی[۹]ها که هر کاشی دارای یک شناسه است و هر کدام شامل یک عنصر پردازشی[۱۰] (PE) که از طریق یک رابط شبکه[۱۱] (NI) به یک مسیریاب (R) متصل است. L شامل مجموعه-

[۹] tile

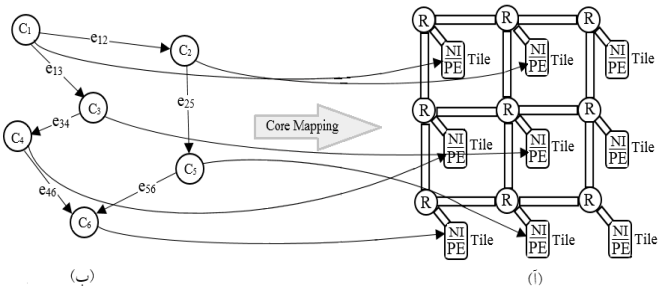
[۱۰] Processing Element

[۱۱] Network interface

مجموعه‌ای از لینک‌های $l_{i,j} \in L$ که دارای یک پهنای‌باند ارتباطی برای عبور داده‌ها میان کاشی‌های $tile_i$ و $tile_j$ است.

۲,۳ گراف هسته

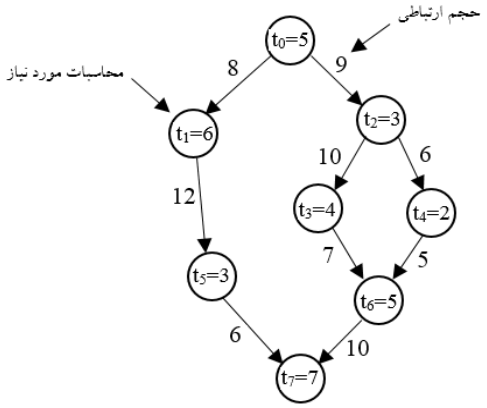
گراف هسته، یک گراف جهت‌دار است و با CG=(C,E) نشان داده می‌شود، C شامل مجموعه‌ای از هسته‌های پردازشی که با C_iدر شکل ۳ قسمت ب نشان داده‌ایم هر هسته‌های پردازشی شامل $(id, cap_{id}, position_{id})$ است که id شناسه، cap_{id} ظرفیت محاسباتی و $position_{id}$ موقعیت هسته پردازشی (در کدام کاشی قرار دارد)، نشان می‌دهد. E شامل مجموعه‌ای از یال‌ها است و $e_{i,j} \in E$نشان‌دهنده مقدار داده ارتباطی میان دو هسته پردازشی C_i و C_j است. در این مدل هر هسته پردازشی شامل یک یا چند وظیفه است.



شکل ۲.(آ) مدل NoC (ب) گراف هسته - نگاشت هسته‌های پردازشی به کاشی‌های NoC

۳,۳ گراف برنامه کاربردی

از مدل برنامه کاربردی در مرجع [۹] که گراف جهت‌دار بدون سیکل است و با AG=(T,R) نشان داده می‌شود، استفاده کردیم. T شامل مجموعه‌ای از وظایف t_i که هر t_i شامل (id, req_i) است. req_i نیاز محاسباتی برای هر وظیفه نشان می‌دهد که باید برای اجرا روی هسته پردازشی، ظرفیت باقی‌مانده آن به اندازه نیاز محاسباتی وظیفه مورد نظر است، قرار بگیرد. R شامل مجموعه‌ای از یال‌ها است $R \square$ $r_{i,j}$نشان‌دهنده حجم ارتباطی میان t_i و t_j است. در شکل ۴، یک گراف وظایف یا جریان کاری (workflow)- ترتیب منطقی زیر وظایف (یا وظایف ساده) از یک وظیفه بزرگ را جریان کار می-نامند [۲۳]- نشان می‌دهد. هر کدام از گره‌های این گراف یک وظیفه را نشان می‌دهد و عدد داخل هر گره برابر با نیاز محاسباتی وظیفه مربوطه است و مقدار روی یال‌های بین دو وظیفه برابر با حجم ارتباطی میان دو وظیفه است.



شکل ۳. گراف وظایف (جریان کاری)

۴,۳ مدل انرژی

انرژی مصرفی شامل انرژی ارتباطی و انرژی محاسباتی است. انرژی ارتباطی (E_{NoC}) مقدار انرژی مصرف شده توسط مولفه‌های شبکه برای انتقال مقدار داده مورد نیاز میان دو هسته پردازشی است[۲۴]. معادله ۱ مقدار انرژی ارتباطی به صورت زیر تعریف می‌کند:

$$E_{NoC}=\sum_{i,j\in B}^{n}\left[e_{i,j}\times\left((N_h+1)\times E_R+N_h\times E_L+2\times E_C\right)\right] \quad (1)$$

E_R انرژی مصرف شده توسط مسیریاب برای انتقال یک بیت است. E_L و E_C به ترتیب انرژی مصرف شده برای انتقال یک بیت توسط لینک میان دو مسیریاب و لینک میان مسیریاب و هسته پردازشی است.$e_{i,j}$ مقدار داده‌ای که میان دو هسته پردازشی C_i و C_j که در دو کاشی t_i و t_j نگاشته شده‌اند.N_h، فاصله منهتن، نشان‌دهنده تعداد گام‌ها یا لینک‌ها برای عبور یک بیت میان دو کاشی $tile_i$ و $tile_j$ است که به صورت زیر محاسبه می‌شود.

$$N_h=|x_i-x_j|+|y_i-y_j| \quad (2)$$

(x, y) موقعیت کاشی در مدل NoC نشان می‌دهد.

انرژی محاسباتی (E_{comp}) از انرژی مصرف شده در هسته‌ها بدست می‌آید[۹]. انرژی مصرفی هر هسته به صورت زیر بدست می‌آید:

$$E_{corej}=\left(n_{idlej}\times E_{idle}\right)+\left(n_{utilj}\times E_{util}\right) \quad (3)$$

E_{corej} انرژی مصرفی توسط هسته پردازشی j را نشان می‌دهد. هسته پردازشی در دو حالت در حال پردازش یا بیکار قرار می‌گیرد. n_{idlej} تعداد سیکل‌هایی که هسته پردازشی j بیکار است و n_{utilj} تعداد سیکل‌هایی که هسته پردازشی j در حال پردازش داده می‌باشد. E_{idle} و E_{util} به ترتیب مقدار مصرف انرژی یک هسته پردازشی در حالت‌های بیکار و پردازش داده را نشان می‌دهد[۹].

$$E_{comp}=\sum_{j}^{n} E_{corej} \qquad (٤)$$

انرژی مصرفی کل با استفاده از معادله ۵ بدست می‌آید:

$$E_{total}=E_{NoC}+E_{comp} \qquad (٥)$$

مقدار کل داده‌ای است که میان دو کاشی t_i و t_j که با فاصله منهتن ۱ مبادله می‌شود با W_{t_i,t_j} نشان می‌دهیم و مقدار آن طبق معادله زیر بدست می‌آید:

$$W_{t_i,t_j}= \sum_{\forall m,k \in B} (e_{k,m} \times g) \qquad (٦)$$

$$g=\begin{cases} ١ & \text{if } l_{i,j} \in Path_{k,m} \\ ٠ & \text{otherwise} \end{cases}$$

مسیر بین دو هسته پردازشی C_i و C_j که در t_m و t_k نگاشته شده‌اند، شامل مجموعه‌ای از لینک‌ها است که بین این دو کاشی در مسیریابی X-Y قرار گرفته‌اند. به عبارتی دیگر هر لینک ممکن است بین چند جفت هسته پردازشی مشترک باشد. W_{t_i,t_j} برابر است با جمع مقدار داده‌هایی که از لینک میان $(l_{i,j})$ t_i و t_j می‌گذرد. مقدار g برابر با ۱ است اگر $e_{k,m}$ از لینک میان t_i و t_j بگذرد (اگر $l_{i,j}$ عضو مسیر دو هسته پردازشی m و k $(Path_{k,m})$ باشد.)، در غیر این صورت مقدار آن برابر با ۰ می‌باشد.

۴. روش پیشنهادی

۴_۱ الگوریتم شاهین هریس گسسته

الگوریتم شاهین هریس یک الگوریتم پیوسته است و مسئله نگاشت ماهیت گسسته دارد و برای حل مسئله نگاشت سعی بر آن شد الگوریتم شاهین هریس را به روش زیر به صورت گسسته تعریف کنیم. در این روش، هر شاهین موقعیتی دارد و در هر تکرار بهترین موقعیت بر اساس تابع هدف مشخص شده و موقعیت جدید طعمه (خرگوش) بدست می‌آید. موقعیت هر شاهین یک جواب برای مسئله نگاشت هسته‌های پردازشی روی NoC است و با بدست آوردن جایگشت‌های مختلف نگاشت، موقعیت شاهین‌ها تغییر می‌یابد و در نهایت جواب بهینه یا نزدیک به بهینه که در واقع موقعیت طعمه را مشخص می‌کند، بدست می‌آید.

Algorithm: Proposed Discrete HHO Algorithm Pseudo-Code

```
Inputs: number of bins(dim), population size N (# haris), #iteration=T
Output: core mapping in NoC
Initialize:
Foreach haris random pos (۰, dim)
Rabbit.pos= random pos (۰, dim)

While(T) {
Rabbit.pos = pos of minimum fitness of haris
r= random (۰, ۱)
E=۲E. (۱- t/T), -۱<E.<۱
If(|E|>=۱)
        Update location of haris using Eq. (۷)
else
        If(r>= ۰,۵ && |E|>= ۰,۵)
            Soft besiege
        else if (r>= ۰,۵ && |E|>= ۰,۵)
            Hard besiege
        else if (r>= ۰,۵ && |E|>= ۰,۵)
            Soft besiege with progressive rapid dives
        else if (r>= ۰,۵ && |E|>= ۰,۵)
            Hard besiege with progressive rapid dives
Return Rabbit.pos
```

شکل ۴. شبه کد الگوریتم شاهین هریس گسسته

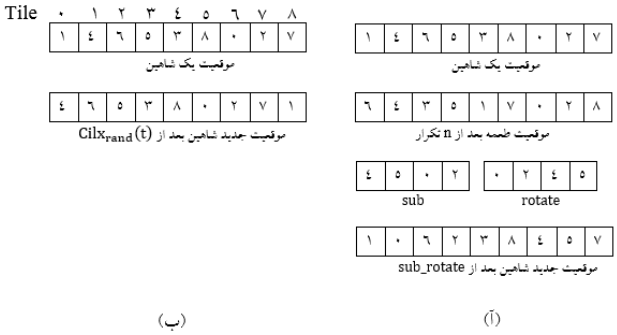
در ادامه هر یک از مراحل را توضیح می‌دهیم.

۴_۱_۱ مرحله اکتشاف

در این گام الگوریتم به دنبال کشف فضای جستجو می‌باشد و به‌روزرسانی موقعیت شاهین‌ها براساس عملگر تصادفی و یا براساس موقعیت سایر شاهین‌ها صورت می‌گیرد.

$$x(t+1)=\begin{cases} Cilx_{rand}(t), & q\geq 0.5 \\ sub\_rotate\left(x_{rabbit}(t)-x_m(t)\right), & q < 0.5 \end{cases} \quad (7)$$

$Cilx_{rand}(t)$، یک شاهین که به صورت تصادفی انتخاب شده و موقعیت آن به صورت شیفت چرخشی به چپ تغییر داده و موقعیت جدید شاهین فعلی می‌شود. $x_m(t)$، ابتدا میانگین برآورد موقعیت شاهین‌ها بدست آورده سپس اولین شاهینی که برآورد موقعیت آن کمتر از میانگین است، انتخاب می‌شود. $x_{rabbit}(t)$، موقعیت طعمه می‌باشد. با استفاده از تابع $sub\_rotate\left(x_{rabbit}(t)-x_m(t)\right)$ اعداد مشابه در درایه‌های با اندیس یکسان از $x_m(t)$ و $x_{rabbit}(t)$ مشخص و به صورت تصادفی شیفت چرخشی می‌دهیم. موقعیت جدید شاهین فعلی بدست می‌آید. شکل ۵ نمونه مثالی از بدست آوردن موقعیت جدید شاهین با استفاده از معادله ۷ نشان می‌دهد.



شکل ۵. مثالی از بدست آوردن موقعیت جدید شاهین با انجام عملیات sub_rotate (آ) و $Cilx_{rand}(t)$ (ب)

۴_۱_۲ مرحله بهره‌برداری

در این گام، الگوریتم به دنبال بهبود جواب‌های پیدا شده می‌باشد و از عملگرهای محاصره سخت[۱۲] و محاصره نرم[۱۳] استفاده می‌کند.

فرض کنید که r شانس طعمه در فرار موفقیت‌آمیز یا فرار ناموفق قبل از حمله غافلگیرانه باشد. در حالی که طعمه فرار می‌کند، شاهین‌ها ، محاصره سخت یا نرم را برای گرفتن طعمه شکل می‌دهند.

به این معنی که آن‌ها بر اساس انرژی باقیمانده طعمه، حلقه محاصره را به صورت سخت یا نرم تنگ‌تر می‌کنند.

۴_۱_۲_۱ محاصره نرم:

jump_strength، یک عدد تصادفی بین ۰ تا اندازه جمعیت شاهین‌ها است. در این گام، اگر $E\times jump_{strength}>r\times dim$ باشد، موقعیت جدید شاهین طبق موقعیت طعمه بدست می‌آید. به این صورت عدد موجود در درایه با اندیس jump_strength از موقعیت طعمه، در اندیس ۰ از موقعیت جدید شاهین قرار بگیرد و باقی درایه‌ها از موقعیت طعمه نیز به درایه‌های موقعیت جدید انتقال یابند. در غیر این صورت عدد موجود در درایه با اندیس jump_strength از موقعیت طعمه در اندیس آخر از موقعیت جدید شاهین قرار بگیرد.

۴_۱_۲_۲ محاصره سخت:

ابتدا مقدار $sub\_rotate\left(x_{rabbit}(t)-x(t)\right)$ محاسبه می‌کنیم سپس موقعیت جدید شاهین را طبق موقعیت طعمه و sub_rotate محاسبه می‌کنیم.

۴_۱_۲_۳ محاصره نرم با شیرجه‌های پیشرو[۱۴]:

در این گام، اگر $E\times jump_{strength}>r\times dim$ باشد، موقعیت جدید شاهین طبق موقعیت طعمه بدست می‌آید به این صورت عدد موجود در درایه با اندیس ۰ از موقعیت طعمه، در اندیس jump_strength از موقعیت جدید شاهین قرار بگیرد و باقی درایه‌ها از موقعیت طعمه نیز به درایه‌های موقعیت جدید انتقال یابند. در غیر این صورت عدد موجود در درایه با اندیس (dim - jump_strength) از موقعیت طعمه در اندیس jump_strength از موقعیت جدید شاهین قرار بگیرد. اگر برآورد موقعیت جدید شاهین نسبت به برآورد موقعیت فعلی بهتر باشد، موقعیت جدید را می‌پذیریم. در غیر این صورت $sub\_rotate\left(x_{rabbit}(t)-x(t)\right)$ محاسبه می‌کنیم سپس موقعیت

[۱۲] Hard besiege
[۱۳] Soft besiege
[۱۴] Soft besiege with progressive rapid dives

جدید شاهین را طبق موقعیت طعمه و sub_rotate محاسبه می-کنیم.

۴_۱_۲_۴ محاصره سخت با شیرجه‌های پیشرو[۱۵]:

در این گام، $sub\_rotate(x_{rabbit}(t)-x_m(t))$ محاسبه و موقعیت جدید شاهین را طبق موقعیت طعمه و sub_rotate محاسبه می-کنیم. اگر برآورد موقعیت جدید شاهین نسبت به برآورد موقعیت فعلی بهتر باشد، موقعیت جدید را می‌پذیریم. در غیر این صورت از تابع $swap(x_m,dim)$ – دو عدد در بازه ۰ تا اندازه جمعیت به صورت تصادفی بدست آورده و درایه‌های آن‌ها را تعویض می‌کند- برای بدست آوردن نگاشت‌های بهتر استفاده می‌کنیم.

۴_۱_۳ انتقال از مرحله اکتشاف به مرحله بهره‌برداری

برای انتقال از مرحله اکتشاف به مرحله بهره‌برداری با توجه به معادله زیر انجام می‌شود. به طور خلاصه، اکتشاف هنگامی اتفاق می‌افتد که $|E|\geq ۱$، در حالی که بهره‌برداری زمانی که $|E|<۱$ باشد، اتفاق می‌افتد. E_0 انرژی اولیه طعمه است.

$$E=E_0\left(۱-\frac{t}{T}\right) \qquad (۸)$$

E انرژی فرار طعمه را نشان می‌دهد، E_0 انرژی اولیه و مقدار آن بین ۱ و ۱- می‌باشد. T برابر با تعداد کل تکرارها است[۱۰].

۴_۲ تعیین تابع هدف

الگوریتم HHO یک تکنیک بهینه‌سازی مستقل از گرادیان و مبتنی بر جمعیت می‌باشد، بنابراین می‌توان از این الگوریتم برای حل مسائل بهینه‌سازی به شرط وجود یک تابع هدف مناسب، استفاده کرد. تعیین تابع هدف بستگی به مسئله مورد نظر دارد. در این مقاله هدف کاهش انرژی مصرفی و بهتر شدن عملکرد سیستم است و با توجه به اینکه با کاهش فاصله میان وظایف مرتبط می‌توان انرژی ارتباطی را کاهش داد. در تابع هدف مقدار E_{NoC} را محاسبه و با نگاشت اولیه که به صورت تصادفی تعیین می‌شود، مقایسه می‌کنیم و بعد از T تکرار الگوریتم شاهین هریس، مقدار E_{NoC} بهینه‌تر می-شود.

با توجه به معادله ۱ که برای محاسبه انرژی ارتباطی شبکه به کار می‌رود می‌توان به صورت زیر به ساده کرد:

$$E_{NoC}=\sum_{i,j\in B}^{n}\left[e_{i,j}\times\left((N_h+۱)\times E_R+N_h\times E_L+۲\times E_C\right)\right]\approx \sum_{i,j\in B}^{n}(e_{i,j}\times N_h)=\sum_{li,j=۰}^{|L|}w_{i,j} \qquad (۹)$$

در محاسبه انرژی ارتباطی می‌توان انرژی مسیریاب، لینک و هسته را ناچیز در نظر گرفت به این دلیل که مقدار حجم ارتباطی و تعداد گام تأثیر بیشتری در محاسبه بار شبکه دارند و از آنجا که می‌توان وزن هر لینک با استفاده از رابطه ۳ محاسبه کرد. در نتیجه برای محاسبه بار شبکه در تابع هدف می‌توان از ساده شده معادله ۱ یعنی معادله ۹ استفاده کرد.

تعادل بار لینک نه تنها می‌تواند ازدحام شبکه و تأخیر در صف را کاهش دهد بلکه از ایجاد نقاط داغ نیز جلوگیری می‌کند [۲۴]. ما در نظر داریم تا حد امکان وظایف ارتباطی را در لینک‌های مختلف توزیع کنیم. اگر بیشتر وظایف ارتباطی از تعداد محدودی از لینک‌ها استفاده کنند، ممکن است ازدحام شبکه و نقاط داغ افزایش یابد. بنابراین با محاسبه جذر واریانس (انحراف معیار) بار هر لینک طبق معادله زیر، تعادل بار را ارزیابی می‌کنیم.

$$Load=\sqrt{\sum_{i=۱}^{L}\frac{\left(load_i-\sum_{i=۱}^{L}\frac{load_i}{L}\right)^۲}{L}} \qquad (۱۰)$$

در این معادله $load_i$ بار لینک i و L تعداد لینک‌ها را مشخص می‌کند. هرچه مقدار انحراف معیار بار لینک کمتر باشد یعنی بار لینک متعادل‌تر است. زیرا انحراف معیار از جمله معیارهای پراکندگی می‌باشد.

ما در این قسمت توابع هدف را مطابق با اهداف گفته شده، تعریف می‌کنیم. هدف ما در این قسمت، تعیین تابع هدف درست می‌باشد.

۴_۲_۱ VARYANCE

[۱۵] Hard besiege with progressive rapid dives

در الگوریتمی که VARYANCE نام گذاری کرده‌ایم، تابع هدف را به صورت زیر قرار دادیم و نتایج آن را در بخش بعدی می‌توان مشاهده کرد.

$(E_{NoC} < E_{NoCrandom} \ || \ load < load_{random})$

CHARAK ۲_۲_۴

ما از دامنه میان چارکی که از جمله معیارهای پراکندگی می‌باشد، استفاده کردیم.

$$IQR=Q_3-Q_1 \qquad (11)$$

Q_1 چارک اول و Q_3 چارک سوم است. در الگوریتم CHARAK تابع هدف را به صورت زیر قرار داده‌ایم.

$(E_{NoC} < E_{NoCrandom} \ || \ IQR < IQR_{random})$

OR ۳_۲_۴

در الگوریتم OR تابع هدف به صورت زیر تعریف کرده‌ایم.

$(E_{NoC} < E_{NoCrandom} \ || \ load < load_{random} \ || \ IQR < IQR_{random})$

&OR ۴_۲_۴

در الگوریتم &OR تابع هدف به صورت زیر تعریف کرده‌ایم.

$(E_{NoC} < E_{NoCrandom} \ || \ (load < load_{random} \ \&\& \ IQR < IQR_{random}))$

شبه کد تابع هدف &OR در شکل ۶ آورده شده است.

```
Pseudo-code of fitness function
Fitness (haris, bins) {

Noc[i][j] = haris.pos[k] and Set position x=i, y=j of
            bin [ haris.pos[k]]

Calculator weight of link of NoC //weight of l_{i,j}(W_{i,j})
Calculator sqrt of variance
Calculator Middle quarter
load= E_Noc
if ((variance > v&& Middle quarter>q) || load> l)
        returen ۰
else
returen load
}
```

شکل ۶. شبه کد تابع هدف

۳_۴ محاسبه پیچیدگی الگوریتم

پیچیدگی الگوریتم برابر با $O\ (T\times N\times M_{E_{NoC}})$ است. T تعداد تکرار الگوریتم شاهین هریس و N تعداد جمعیت شاهین‌ها و $M_{E_{NoC}}$ زمان اجرای محاسبه E_{NoC} است که به تعداد هسته‌های پردازشی برای نگاشت بر روی NoC بستگی دارد.

نتایج هر چهار تابع هدف در بخش بعد برای مقدار انرژی مصرفی ۵۰ گراف مختلف نشان داده شده است.

۵. محیط شبیه‌سازی و ارزیابی آزمایش‌ها

۵-۱ محیط شبیه‌سازی

محیط شبکه روی تراشه با مش دو بعدی در چارچوب جاوا و با استفاده از Eclipse IDE را شبیه‌سازی کرده‌ایم. و برای ارزیابی عملکرد روش ارائه شده از آن استفاده می‌کنیم.

HNOCS یک شبیه‌ساز NoC منبع باز و مبتنی بر OMNeT++ است. HNOCS مخفف Heterogeneous Network-on-Chip Simulator است. این شبیه‌ساز یک چارچوب منبع باز، مقیاس پذیر، قابل توسعه و کاملا قابل پارامترسازی برای مدل‌سازی انواع NoC است [۲۵]. شبیه‌ساز HNOCS از توپولوژی مش، سوئیچینگ خزشی[۱۶] با کانال‌های مجازی، کنترل جریان مبتنی بر اعتبار و از الگوریتم مسیریابی XY برای مسیریابی بسته‌ها استفاده می‌کند. HNOCS مجموعه‌ای از اندازه‌گیری‌های آماری، مانند توان عملیاتی و تأخیرهای مختلف را در سطح فلیت و بسته ارائه می‌دهد و ما از این شبیه‌ساز استفاده کردیم، زیرا دارای ساختاری مدولار است که اصلاح و افزودن به آن در مقایسه با سایر شبیه‌سازها آسان‌تر می‌کند.

[۱۶] Wormhole switching

جدول ۱. تنظیم پارامترهای HNOCS

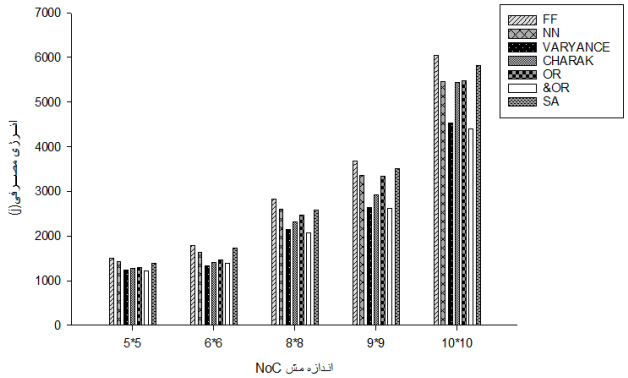
پارامترها	مقدار
پهنای باند لینک	۱۶ Gbps
تأخیر لینک	۰,۰۱ ns
تأخیر مسیریاب	۲ ns

در حالت ساده این شبیه‌ساز برای هسته‌های همگن طراحی شده و می‌توان هسته‌های ناهمگن نیز به آن اضافه کرد اما در این کار همان حالت اولیه را در نظر گرفتیم و فایل خروجی بدست آمده از Eclipse IDE را در این شبیه‌ساز اجرا کردیم.

روش پیشنهادی را با الگوریتم‌های FF، NN [۷] و SA [۲۶] مقایسه کردیم.

۵_۲ جریان‌های کاری ساختگی

کار ارائه شده را برای ۵۰ جریان کاری با اندازه مش NoC از ۵×۵ تا ۱۰×۱۰، تعداد bin از ۲۵ تا ۱۰۰ و تعداد وظایف در گراف برنامه-کاربردی از ۳۰۰ تا ۱۰۰۰، مورد آزمایش قرار دادیم.

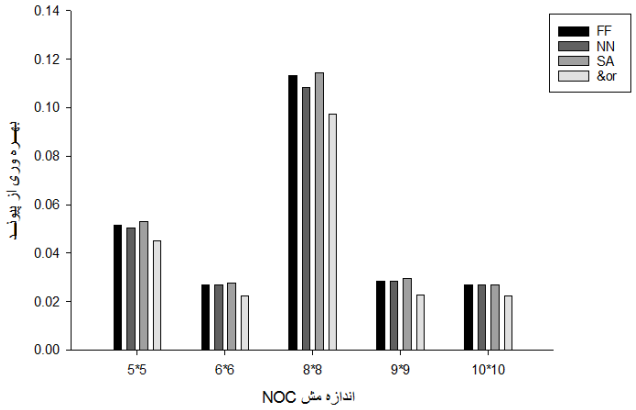


شکل ۷. میانگین انرژی مصرفی برای اندازه‌های مختلف NOC

در شکل ۷، میانگین انرژی مصرفی در اندازه‌های مختلف NoC نشان داده شده است. در روش OR انرژی مصرفی نسبت به روش &OR بیشتر است. در روش OR به دلیل اینکه الگوریتم باید نگاشتی را که هم انرژی مصرفی و هم انحراف معیار بار و دامنه چارکی بار کمتری دارد را بیابد، در نگاشت‌های بهینه محلی اولیه متوقف می‌شود. اما در روش &OR به دلیل اینکه باید هم انرژی مصرفی و هم انحراف معیار بار یا دامنه چارکی بار کمتری دارد را بیابد، نگاشت‌های محلی بهتری را می‌یابد و پراکندگی بار کمتری دارند. همان‌طور که مشاهده می‌شود روش &OR نسبت به روش-های دیگر انرژی مصرفی کمتری دارد.

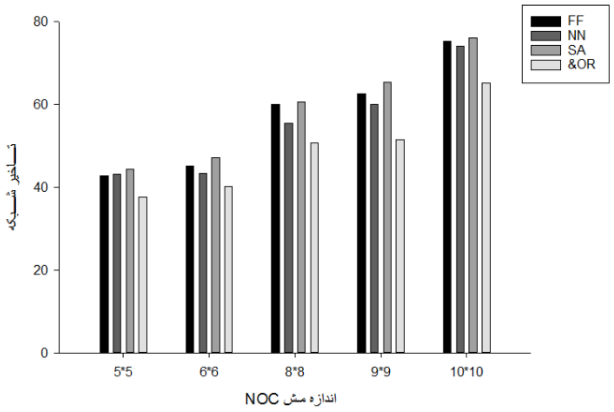
در سه شکل زیر بهره‌وری از لینک، تأخیر انتها به انتها و تأخیر شیکه برای گراف‌های وظایف در چهار روش FF، NN، SA و &OR نشان داده شده است.

هر چه فاصله منهتن کمتر به عبارت دیگر وظایف به یکدیگر نزدیک-تر باشند، انرژی ارتباطی و در نتیجه انرژی مصرفی کل کاهش می-یابد. همچنین برای متعادل کردن بار لینک‌ها از معیارهای پراکندگی استفاده شده است. به همین دلیل در این روش علاوه بر کاهش انرژی ارتباطی، کاهش میزان پراکندگی بار نیز در نظر گرفته شده است تا به یک نگاشت با انرژی مصرفی کم و میزان پراکندگی بار کم برسیم.

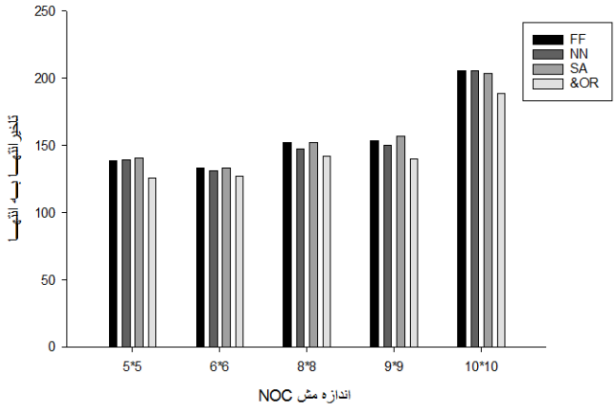


شکل ۸. میانگین بهره‌وری از لینک‌ها در اندازه‌های مختلف NoC

در شکل ۸ میانگین بهره‌وری از لینک‌ها را در چهار روش مختلف نشان می‌دهد که روش &OR نسبت به سه روش دیگر میانگین بهره‌وری کمتری داشته به این دلیل که با کاهش بار ارتباطی در شبکه و تعادل در میزان پراکندگی بار در لینک‌های شبکه بهره‌وری از لینک‌ها کاهش می‌یابد.



شکل ۹. میانگین تأخیر شبکه در اندازه‌های مختلف NoC



شکل ۱۰. میانگین تأخیر انتها به انتها در اندازه‌های مختلف NoC

در شکل‌های ۹ و ۱۰ نتایج روش &OR نسبت به سه روش دیگر مشاهده می‌شود. تأخیر تحت تأثیر بار روی لینک‌ها در NoC به عبارتی دیگر با کاهش میزان بار ارتباطی روی لینک‌ها می‌توان تا حدودی تأخیر را کاهش داد. در روش &OR هدف کاهش بار ارتباطی شبکه و تعادل در میزان پراکندگی بار می‌باشد به همین دلیل در مقایسه با سه روش دیگر توانسته تأخیر در شبکه نسبت به تأخیر انتها به انتها را بیشتر کاهش دهد.

۵_۳ جریان‌های کاری واقعی

علاوه بر اینکه کار ارائه شده را روی جریان‌های ساختگی بررسی کردیم از چند جریان کاری واقعی همچون Cybershake، Genome، Montage و Ligo [۲۷] برای ارزیابی این روش استفاده می‌کنیم. در جدول زیر اطلاعات مورد نیاز برای اجرای جریان‌های کار آورده شده است.

جدول ۲- اطلاعات مورد نیاز برای اجرای جریان‌های کار

جریان کاری	تعداد وظایف	تعداد bin مورد نیاز برای اجرا	ظرفیت bin
Cybershake	۱۰۰	۱۶	۲۰۰
Genome	۱۰۰	۱۶	۱۰۰۰۰
Montage	۱۰۰	۲۵	۶۰
Ligo	۱۰۰	۳۶	۱۰۰۰

ما برای یک نمونه از اجرای جریان‌های کاری در روش‌های SA و &OR مقدار بهره‌وری از لینک، تأخیر انتها به انتها و تأخیر شبکه را با اجرا در HNOCS محاسبه کردیم. در جداول زیر مقدار مصرف انرژی و انحراف معیار بار نیز آورده شده است.

جدول ۳. مقدار انرژی مصرفی برای هریک از جریان‌های کاری در الگوریتم‌های مختلف

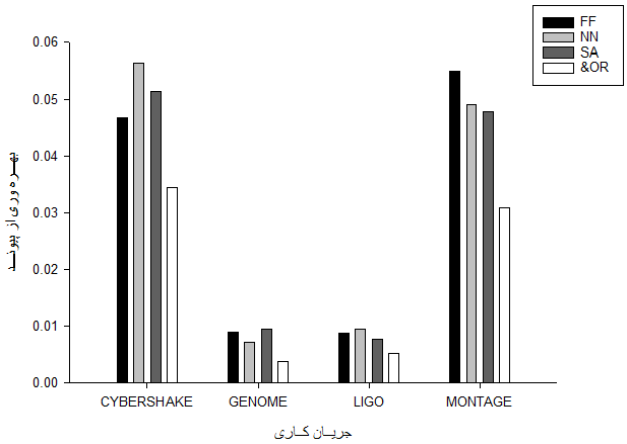
	Cybershake	Genome	Montage	Ligo
FF	۷۱۵۶,۸۹	۱۴۹۲۱۳,۳۴	۴۳۶۵,۳۶	۳۲۳۹۸,۳۸
NN	۹۱۸۲,۷	۱۴۳۱۰۲,۹۷	۳۹۷۱,۷	۳۱۶۹۷,۷۳
SA	۷۹۵۹,۳۳	۱۲۵۶۷۸,۷۹	۴۱۸۶,۶۲	۳۱۳۴۲,۹۵
&OR	۵۳۱۰,۰۹	۹۳۷۱۴,۰۳	۲۵۳۹,۰۸	۱۶۴۴۵,۷

در جدول ۳، مقدار انرژی مصرفی برای هر یک از جریان‌های کار محاسبه شده است. روش &OR مقدار انرژی مصرفی کمتری نسبت به روش‌های FF، NN و SA داشته است.

جدول ۴. مقدار انحراف معیار بار لینک برای هریک از جریان‌های کاری در الگوریتم‌های مختلف

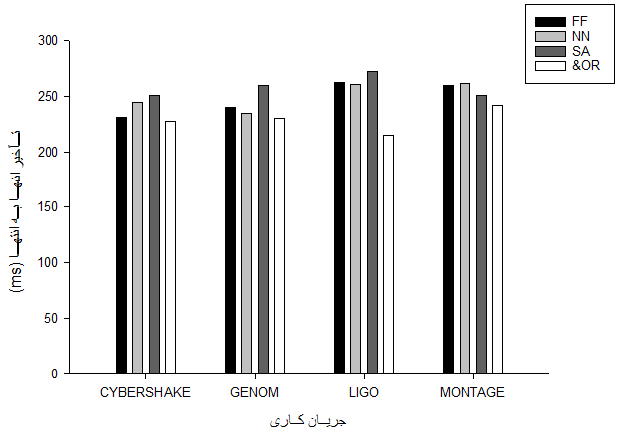
	Cybershake	Genome	Montage	Ligo
FF	۹۶۳,۶۸	۲۶۳۹۰,۷	۲۹۲,۱	۳۲۳۶,۳۵
NN	۱۳۳۳,۹۳	۲۶۴۸۹,۸۷	۲۷۸,۵۴	۲۲۸۳,۹۵
SA	۱۲۰۲,۴۳	۲۳۰۳۰,۳۱	۳۰۸,۶۹	۲۸۱۸,۸۳
&OR	۶۵۶,۸۵	۱۶۸۴۷,۱۱	۱۶۲,۶۶	۱۳۸۴,۵۲

در جدول ۴، مقدار انحراف معیار بار لینک برای هر یک از جریان‌های کار محاسبه شده است. و روش &OR مقدار انحراف معیار کمتری نسبت به روش‌های FF، NN و SA داشته است.



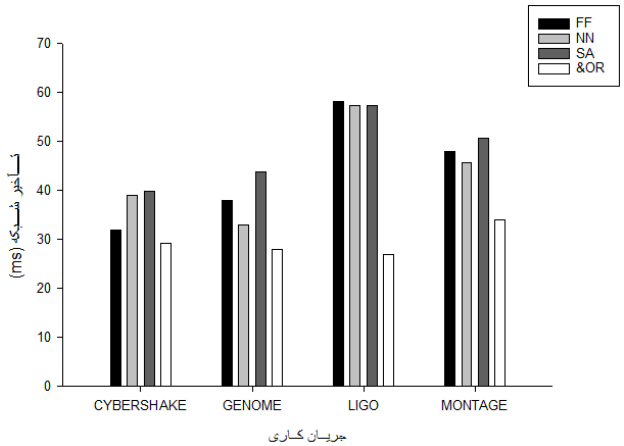
شکل ۱۱. بهره‌وری از لینک برای نمونه‌ای از جریان‌های کاری

با متعادل کردن بار روی لینک‌ها و کاهش انرژی ارتباطی می‌توان بهره‌وری از لینک‌ها را کاهش داد. در شکل ۱۱، روش &OR نسبت به سه روش دیگر میزان بهره‌وری از لینک را کاهش داده است.



شکل ۱۲. تأخیر انتها به انتها برای نمونه‌ای از جریان‌های کاری

در شکل ۱۲، تأخیر انتها به انتها برای جریان‌های کاری نشان داده شده است. روش &OR نسبت به سه روش دیگر میزان تأخیر انتها به انتها را کاهش داده است. در جریان کاری LIGO کاهش تاخیر انتها به انتها نسبت به سه جریان کاری دیگر بیشتر مشاهده می‌شود.



شکل ۱۳. تأخیر شبکه برای نمونه‌ای از جریان‌های کاری

در شکل ۱۳، تأخیر شبکه برای جریان‌های کاری نشان داده شده است. روش &OR نسبت به سه روش دیگر تأخیر شبکه را کاهش داده است که نتیجه کاهش انرژی ارتباطی و میزان پراکندگی بار می‌باشد. همان طور که مشاهده می‌شود، کاهش انرزی ارتباطی تأثیر بیشتری در کاهش تأخیر شبکه نسبت به تأخیر انتها به انتها دارد.

۶. نتیجه‌گیری

نگاشت وظایف روی شبکه روی تراشه یک مسئله رده سخت است. در این مقاله سعی بر آن شد که با استفاده از الگوریتم شاهین هریس، بتوان نگاشتی را برای هسته‌های پردازشی بر روی NoC انتخاب کرد که علاوه بر اینکه انرژی مصرفی کمتری داشته باشد، میزان پراکندگی بار کمتری داشته تا بهره‌وری از لینک‌های NoC متعادل و همچنین تأخیر کمتری در شبکه داشته باشد. همان‌طور که نشان داده شد، روش &OR در مقایسه با الگوریتم‌های پایه، عملکرد بهتری دارد.

مراجع

[۱] A. V. Bhaskar and T. Venkatesh, "Performance analysis of network-on-chip in many-core processors," Journal of Parallel and Distributed Computing, vol. ۱۴۷, pp. ۱۹۶-۲۰۸, ۲۰۲۱.

[۲] اسدی بهاره، رشادی میدیا، خادم زاده احمد، کرباسی مصطفی،"مدل های چرخشی تطابقی و الگوهای ترافیکی جهت کاهش اتلاف نوری در شبکه های روی تراشه ی نوری"، فصلنامه فناوری اطلاعات و ارتباطات ایران، ۱۰(۳۶ و ۳۵) ، ۲۶-۱۵، ۱۳۹۷.

[۳] W. Amin et al., "Performance evaluation of application mapping approaches for Network-on-Chip designs," IEEE Access, vol .۸ ,pp. ۶۳۶۰۷-۶۳۶۳۱, ۲۰۲۰.

[٤] A. K. Singh, W. Jigang, A. Kumar, and T. Srikanthan, "Run-time mapping of multiple communicating tasks on MPSoC platforms," Procedia Computer Science, vol. ۱, no. ۱, pp. ۱۰۱۹-۱۰۲٦, ۲۰۱۰.

[٥] S. Kaushik, A. K. Singh, and T. Srikanthan, "Computation and communication aware run-time mapping for NoC-based MPSoC platforms," in ۲۰۱۱ IEEE International SOC Conference, ۲۰۱۱, pp. ۱۸٥-۱۹۰: IEEE.

[٦] M. Obaidullah and G. N. Khan, "Application mapping to mesh NoCs using a Tabu-search based swarm optimization," Microprocessors and Microsystems, vol. ٥٥, pp. ۱۳-۲٥, ۲۰۱۷.

[۷] B. Xie, T. Chen, W. Hu, X. Tang, and D. Wang, "An energy-aware online task mapping algorithm in NoC-based system," The Journal of Supercomputing, vol. ٦٤, no. ۳, pp. ۱۰۲۱-۱۰۳۷, ۲۰۱۳

[۸] C. Wang, Y. Zhu, J. Jiang, X. Liu, and X. Han, "A dynamic contention-aware application allocation algorithm for many-core processor," in ۲۰۱٥ IEEE ۱۷th International Conference on High Performance Computing and Communications, ۲۰۱٥ IEEE ۷th International Symposium on Cyberspace Safety and Security, and ۲۰۱٥ IEEE ۱۲th International Conference on Embedded Software and Systems, ۲۰۱٥, pp. ۳۰۸-۳۱٥: IEEE.

[۹] T. Maqsood et al., "Energy and communication aware task mapping for MPSoCs," Journal of parallel and distributed computing, vol. ۱۲۱, pp. ۷۱-۸۹, ۲۰۱۸.

[۱۰] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," Future Generation Computer Systems, vol. ۹۷, pp. ۸٤۹-۸۷۲, ۲۰۱۹ .

[۱۱] R. Abbassi, A. Abbassi, A. A. Heidari, and S. Mirjalili, "An efficient salp swarm-inspired algorithm for parameters identification of photovoltaic cell models,"

Energy conversion and management, vol. ۱۷۹, pp. ۳٦۲-۳۷۲, ۲۰۱۹.

[۱۲] H. Faris et al., "An intelligent system for spam detection and identification of the most relevant features based on evolutionary random weight networks," Information Fusion, vol. ٤۸, pp. ٦۷-۸۳, ۲۰۱۹.

[۱۳] G. Wu, "Across neighborhood search for numerical optimization," Information Sciences, vol. ۳۲۹, pp. ٥۹۷-٦۱۸, ۲۰۱٦.

[۱٤] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, Metaheuristics for hard optimization: methods and case studies. Springer Science & Business Media, ۲۰۰٦.

[۱٥] E.-G. Talbi, Metaheuristics :from design to implementation. John Wiley & Sons, ۲۰۰۹.

[۱٦] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "others. ۱۹۸۳," Optimization by simulated annealing. science, vol. ۲۲۰, no. ٤٥۹۸, pp. ٦۷۱-٦۸۰, ۱۹۸۳.

[۱۷] H. John, "Holland. genetic algorithms ", Scientific american, vol. ۲٦۷, no. ۱, pp. ٤٤-٥۰, ۱۹۹۲.

[۱۸] J. Luo, H. Chen, Y. Xu, H. Huang, and X. Zhao, "An improved grasshopper optimization algorithm with application to financial stress prediction," Applied Mathematical Modelling, vol. ٦٤, pp. ٦٥٤-٦٦۸, ۲۰۱۸ .

[۱۹] Q. Zhang, H. Chen, J. Luo, Y. Xu, C. Wu, and C. Li, "Chaos enhanced bacterial foraging optimization for global optimization," IEEE Access, vol. ٦, pp. ٦٤۹۰٥-٦٤۹۱۹, ۲۰۱۸.

[۲۰] M. Mafarja et al., "Evolutionary population dynamics and grasshopper optimization approaches for feature selection problems," Knowledge-Based Systems, vol. ۱٤٥, pp. ۲٥-٤٥, ۲۰۱۸.

[۲۱] I. Aljarah, M. Mafarja, A. A. Heidari, H. Faris, Y. Zhang, and S. Mirjalili, "Asynchronous accelerating multi-leader salp chains for feature selection," Applied Soft Computing, vol. ۷۱, pp. ۹٦٤-۹۷۹, ۲۰۱۸.

[۲۲] S. Salcedo-Sanz, "Modern meta-heuristics based on nonlinear physics processes: A review of models and design procedures," Physics Reports, vol. ٦٥٥, pp. ۱-۷۰, ۲۰۱٦.

[۲۳] L. F. Bittencourt, E. R. Madeira, and N. L. Da Fonseca, "Scheduling in hybrid clouds," IEEE Communications Magazine, vol. ٥۰, no. ۹, pp. ٤۲-٤۷, ۲۰۱۲.

[۲٤] Q. Le, G. Yang, W. N. Hung, X. Song, and X. Zhang, "Pareto optimal mapping for tile-based network-on-chip under reliability constraints," International Journal of Computer Mathematics, vol. ۹۲, no. ۱, pp. ٤۱-٥۸, ۲۰۱٥.

[۲٥] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "HNOCS: modular open-source simulator for heterogeneous NoCs," in ۲۰۱۲ international conference on embedded computer systems (SAMOS), ۲۰۱۲, pp. ٥۱-٥۷: IEEE.

[۲٦] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-aware task allocation for network-on-chip based heterogeneous multiprocessor systems," in ۲۰۱۱ ۱۹th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, ۲۰۱۱, pp. ٤٤۷-٤٥٤: IEEE.

[۲۷] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," Future Generation Computer Systems, vol. ۲۹ ,no. ۳, pp. ٦۸۲-٦۹۲, ۲۰۱۳.