دوازدهمین کنفرانس بین‌المللی انجمن کامپیوتر ایران
دانشگاه شهید بهشتی، دانشکده مهندسی برق و کامپیوتر ،تهران، ایران، ۱ تا ۳ اسفند ۱۳۸۵

یک معماری کارآمد ناهمگون برای واحد عملیاتی قابل بازپیکربندی در یک پردازندهٔ قابل توسعه

بهنام قوامی[۱]، آرش مهدی زاده[۲]، مهدی سعیدی[۳]، مرتضی صاحب الزمانی[۴]

چکیده

جایگزین کردن واحدهای عملیاتی یک پردازندهٔ قابل توسعه با واحدهای عملیاتی قابل بازپیکربندی می‌تواند هزینهٔ تولید و زمان ارائه به بازار را بطور چشمگیری کاهش دهد. بعلاوه افزون بر کاهش هزینهٔ تولید، این پردازنده قادر به اجرای دستورات سفارشی بیشتری نیز خواهد بود. در این مقاله یک معماری ناهمگون برای واحدهای عملیاتی قابل بازپیکربندی در پردازنده‌های قابل توسعه ارائه شده است. نتایج آزمایشات نشان می‌دهد که در مقایسه با معماری پیشین، معماری ارائه شده از تعداد زیادی دستورالعمل سفارشی پشتیبانی می‌کند. بعلاوه این معماری قادر است زمان اجرای یک دستورالعمل سفارشی را بین ۲۰ تا ۴۰ درصد بهبود دهد.

کلمات کلیدی

واحد عملیاتی قابل بازپیکربندی، پردازندهٔ قابل توسعه، گراف جریان داده.

An Efficient Heterogeneous Architecture for the Reconfigurable Functional Unit of Extensible Processors

Behnam Ghavami, Arash Mehdizadeh, Mehdi Saeedi, Morteza Saheb Zamani
Amirkabir University of Technology

Abstract

Replacing the functional units of an extensible processor with reconfigurable functional units can reduce cost as well as time to market significantly. Furthermore, this new processor can execute more custom instructions. In this paper, a heterogeneous architecture for the reconfigurable functional unit of an extensible processor has been proposed. Our experiments show that, our architecture supports a wide range of custom instructions compared with the previous architecture. In addition, using the new architecture can improve the execution time of a custom instruction for about 20% to 40%.

Keywords

Reconfigurable Functional Unit, Extensible Processor, Data Flow Graph.

[1] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی کامپیوتر و فنآوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: ghavamib@aut.ac.ir
[2] دانشجوی کارشناسی ارشد، دانشکدهٔ مهندسی کامپیوتر و فنآوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: a_mehdizadeh@aut.ac.ir
[3] دانشجوی دکترا، دانشکدهٔ مهندسی کامپیوتر و فنآوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: saeedi@ce.aut.ac.ir
[4] دکترا، عضو هیأت علمی، دانشکدهٔ مهندسی کامپیوتر و فنآوری اطلاعات، دانشگاه صنعتی امیرکبیر، پست الکترونیک: szamani@ce.aut.ac.ir

۱- مقدمه

سیستم‌های نهفته[۱] امروزه در بسیاری از عرصه‌ها قابلیت‌های خود را به اثبات رسانیده‌اند که از آن جمله می‌توان به کاربرد سیستمهای نهفته در صنایع نظامی، مخابراتی و نیز کاربردهای روزمره اشاره کرد. با توجه به طیف وسیع کاربرد و نیز نحوهٔ استفاده از این سیستم‌ها، در سالهای اخیر روش‌های متفاوتی نظیر استفاده از پردازه‌های همه‌منظوره[۲] و نیز مدارات مجتمع با کاربرد ویژه[۳] برای پیاده‌سازی آنها ارائه شده است.

استفاده از پردازنده‌های همه‌منظوره که به عنوان یکی از عمومی‌ترین روشهای پیاده‌سازی یک سیستم نهفته شناخته می‌شود، اگرچه قابلیت انعطاف‌پذیری زیادی به همراه دارد ولی به دلیل داشتن معماری عام و مجموعه دستورالعمل‌های پیچیدهٔ پیاده‌سازی شده در واحدهای محاسباتی خود، از نظر سرعت اجرا و نیز مصرف توان کارایی چندان زیادی ندارد. از سوی دیگر و به منظور برطرف کردن این مسائل استفاده از روشهای مبتنی بر مدارات ASIC پیشنهاد شده است. ولی در هر صورت بهبود سرعت اجرا و مصرف توان در تراشه‌های ASIC نیز با هزینهٔ عدم انعطاف‌پذیری سخت‌افزار بدست آمده است و از این‌رو این پیاده‌سازی نیز در بسیاری از کاربردها قابل استفاده نیست.

راه‌حل دیگری که برای پیاده‌سازی سیستمهای نهفته ارائه شده و در حقیقت برای پر کردن شکاف موجود بین پیاده‌سازی مبتنی بر پردازنده و پیاده‌سازی مبتنی بر ASIC بوجود آمده است، بکارگیری سخت‌افزارهای سفارشی در کاربردهای خاص است که در این مورد حتی می‌توان دستورالعملهای آنها را نیز سفارشی نمود. بر این اساس در سالهای اخیر پردازنده‌هایی با مجموعه دستورالعملهای خاص منظوره (ASIP[4]) معرفی شده است که در آنها معماری مجموعه دستورالعمل‌ها (ISA[5]) بر اساس کاربرد هدف ایجاد می‌شود. اما در هر حال در این سیستم‌ها بزرگترین نقطهٔ ضعف، زمان طولانی فرآیند طراحی معماری مناسب یک کاربرد خاص می‌باشد.

در ادامهٔ بهبود عملکرد معماری معرفی شدهٔ ASIP، پردازنده‌هایی با قابلیت توسعهٔ مجموعه دستورالعمل معرفی شدند که در آنها یک هستهٔ مرکزی به کمک یک هستهٔ با قابلیت بازپیکربندی[۶] (به صورت دانه ریز[۷] و یا دانه درشت[۸]) به اجرای یک برنامهٔ کاربردی می‌پردازند. در این سیستم‌ها حتی پس از طراحی مجموعهٔ دستورات سخت‌افزاری، سیستم قابلیت افزودن دستورالعمل‌های سفارشی (CI[9]) را خواهد داشت. این دستورالعمل‌ها بر اساس بلوکهای پایهٔ بحرانی (HBB)[۱۰] یک برنامه که زمان اجرای زیادی را به خود اختصاص می‌دهد، استخراج می‌گردند. یک بلوک پایه دنباله‌ای از دستورالعمل‌ها است که با یک دستورالعمل کنترلی خاتمه می‌یابد. بلوک‌های پایهٔ بحرانی به بلوکهای پایه‌ای اطلاق می‌شود که در طول اجرای برنامه از یک حد آستانهٔ معین بیشتر تکرار می‌شوند. با این توصیف در این سیستم‌ها، بخشهای بحرانی به صورت گرافهای جریان داده[۱۱] (DFG) از برنامه استخراج شده و بر روی یک واحد محاسباتی در هستهٔ اصلی و یا در سخت‌افزار شتاب‌دهنده (بخش بازپیکربند) نگاشت و سپس اجرا می‌شوند.

این مقاله به بررسی معماری یک واحد عملیاتی قابل بازپیکربندی در پردازنده‌های قابل توسعه می‌پردازد. ادامهٔ این مقاله به این صورت سازماندهی شده است: ابتدا در بخش ۲ پیشینهٔ کار بررسی خواهد شد. سپس در بخش ۳ پردازندهٔ AMBER که به عنوان یک بستر از آن استفاده شده است، معرفی می‌شود. در ادامه و در بخش ۴ معماری پیشنهادی برای یک واحد عملیاتی قابل بازپیکربندی ارائه شده است. معماری ارائه شده با بحث در خصوص خوشه‌بندی گرافهای جریان داده و نیز نحوهٔ نگاشت آنها بر روی واحد عملیاتی قابل باز پیکربندی در بخش ۵ ادامه خواهد یافت. در بخش ۶ نحوهٔ اتصال واحد عملیاتی قابل بازپیکربندی با پردازندهٔ اصلی بحث می‌گردد. ارزیابی معماری ارائه شده و بررسی نتایج آزمایشات در بخش ۷ صورت خواهد پذیرفت و در نهایت این مقاله با نتیجه‌گیری و ارائهٔ پیشنهاد برای کارهای آتی در بخش ۸ خاتمه خواهد یافت.

۲- پیشینهٔ کار

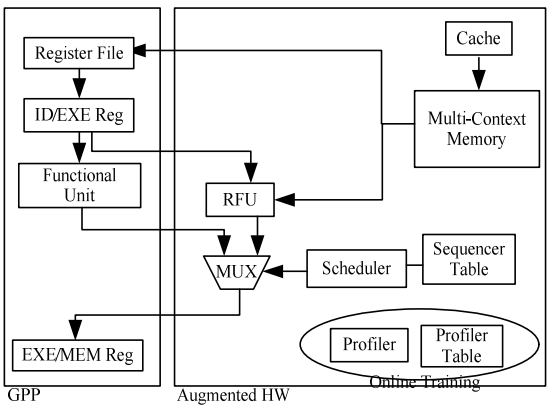
طراحی واحدهای عملیاتی در پردازنده‌های قابل توسعه در چندین مقاله بررسی شده است که از آن جمله می‌توان به کار انجام شده در مقالات [3]، [4]، [5] و [6] اشاره کرد. همچنین در پاره‌ای از موارد تلاشهایی در جهت طراحی واحد عملیاتی با قابلیت بازپیکربندی انجام شده است که از آن جمله می‌توان به PRISC [7]، Chimeara [8]، OneChip [9] اشاره نمود.

واحدهای عملیاتی قابل بازپیکربندی به دو دستهٔ تزویج سست[۱۲] و تزویج محکم[۱۳] تقسیم می‌شوند. در نوع تزویج سست، واحد عملیاتی قابل بازپیکربندی به عنوان یک کمک پردازنده عمل می‌کند و در نتیجه سربار انتقال اطلاعات بین پردازندهٔ اصلی و واحد عملیاتی قابل بازپیکربندی وجود دارد. بعلاوه در این حالت به کامپایلر مجزا و اصلاح دستورالعملها نیاز می‌باشد Garp [14]، MorphoSys [13]. اما در مقابل واحد عملیاتی قابل بازپیکربندی به صورت تزویج محکم هیچ نوع تغییر در کامپایلر و اصلاح دستورات و نیز سربار انتقال اطلاعات را به همراه نخواهد داشت. واحدهای عملیاتی قابل بازپیکربندی به صورت تزویج محکم نیز از نظر بزرگی واحدهای عملیاتی خود به دو دستهٔ دانه ریز و دانه درشت تقسیم‌بندی می‌شوند. در حالی‌که شتاب‌دهنده‌های دانه ریز برای انجام محاسبات در این پردازنده‌ها با قابلیت انعطاف زیادی همراه هستند، اما حجم بسیار زیاد حافظهٔ مصرفی که برای ذخیرهٔ بیتهای پیکربندی بکار می‌رود، استفاده از این شتاب‌دهنده‌ها را با محدودیت رو به رو می‌کند. به عنوان نمونه‌هایی از شتاب‌دهنده‌های دانه ریز می‌توان به کارهای انجام شده درمقاله‌های [7]، [8] و [9] اشاره کرد. به عنوان یک شتاب‌دهندهٔ دانه درشت نیز می‌توان از ADRES [11] نام برد.

نویسندگان مقالهٔ [1] و [3] و [5] یک پردازندهٔ قابل توسعه به نام AMBER را معرفی کرده و به بررسی ساختار واحد عملیاتی قابل بازپیکربندی تزویج محکم برای آن پرداخته‌اند. در این مقالات هدف ارائهٔ معماری جهت پوشش درصد بیشتری از دستورات سفارشی بوده است. با توجه به اینکه در مقالهٔ حاضر نیز از این پردازنده استفاده شده است، در ادامه به بررسی آن خواهیم پرداخت.

۳- مرورکلی بر ساختار پردازندهٔ AMBER

AMBER یک پردازندهٔ قابل توسعه می‌باشد که می‌تواند در کاربردهای مختلفی از سیستم‌های نهفته بکار گرفته شود. این پردازنده شامل یک ریزپردازنده، یک پروفایلر[۱۴]، یک واحد عملیاتی برنامه‌پذیر[۱۵] (RFU) و یک زمان‌بند[۱۶] می‌باشد. بعلاوه ریزپردازندهٔ مورد استفاده در AMBER یک ریزپردازنده RISC مبتنی بر مجموعه دستورالعمل‌های MIPS می‌باشد. شکل ۱ اجزای تشکیل‌دهندهٔ این پردازنده را نشان می‌دهد که در ادامه به تشریح هر یک از این اجزا می‌پردازیم.



شکل ۱- اجزای تشکیل دهنده پردازنده AMBER

۳-۱- پروفایلر

در پردازندهٔ AMBER، در زمان اجرای برنامه‌های کاربردی عملیات پروفایل‌کردن به منظور شناسایی دستورالعملهای سفارشی و تشکیل بیتهای بازپیکربندی صورت می‌گیرد. این عملیات شامل مشاهده و ثبت مقادیر شمارنده برنامه[۱۷] (PC) و نیز دستورالعمل در حال اجرا است. برای هر یک از این دو عملیات در این پردازنده یک جدول در نظر گرفته شده است و به این ترتیب با بازبینی مقادیر PC و نگهداری آنها در این جدول، می‌توان ابتدای هر HBB را مشخص کرد. بعلاوه فرکانس اجرای HBB و نقطهٔ شروع آن نیز در جدول مذکور نگهداری خواهد شد. افزون بر این، پروفایلر با مشاهدهٔ دستورالعمل‌های اجرا شده می‌تواند به جستجوی دستورات انشعاب، فرکانس اجرا و نیز تعداد دفعات پذیرفته[۱۸] شدن آنها بپردازد. جدول دیگری موسوم به جدول دستورالعمل، فیلدهای مذکور را در خود جای می‌دهد.

۳-۲- واحد زمان‌بند

انتخاب ترتیب اجرای برنامه از بین واحد برنامه‌پذیر و ریزپردازنده بر عهدهٔ واحد زمان‌بند می‌باشد. این واحد دارای جدولی است که آدرس شروع هر دستورالعمل سفارشی را در حافظهٔ قابل بازپیکربندی و تعداد سیکل‌های لازم برای اجرای کامل آن روی واحد برنامه‌پذیر را در خود نگه می‌دارد. این جدول بر اساس محل شروع یک دستور سفارشی در کد object مقداردهی می‌شود. هنگامی که واحد زمان‌بند در زمان اجرای برنامه مقدار PC را مساوی با یکی از ورودی‌های جدول می‌یابد، واحد عملیاتی ریزپردازنده از کار افتاده و واحد برنامه‌پذیر اجرای برنامه را بر عهده می‌گیرد. در این حالت واحد زمان‌بند به اندازهٔ سیکل‌های اجرای CI صبر کرده و پس از آن مقدار PC را بر اساس طول دستورالعمل سفارشی سپری شده، مقداردهی می‌کند.

۳-۳- واحد برنامه پذیر

در پردازندهٔ AMBER قسمتهایی از برنامهٔ کاربردی را که نیازمند شتاب‌دهی و یا تکرار زیاد می‌باشند، می‌توان بر روی هسته‌ای با قابلیت بازپیکربندی اجرا نمود. به این ترتیب سرعت اجرای برنامه می‌تواند به صورت قابل توجهی افزایش یابد [1] لذا هر چه دستورالعملهای سفارشی بیشتری قابل نگاشت بر روی این واحد باشند، سرعت اجرای برنامه نیز افزایش می‌یابد.

پردازنده AMBER دارای دو حالت عملیاتی یادگیری و عادی می‌باشد که در حالت یادگیری، برنامه‌های کاربردی به منظور شناسایی بلوکهای پایه پروفایل می‌شوند. با استفاده از این اطلاعات، HBB ها شناسایی و از کد object برای استخراج اطلاعات پیکربندی استفاده می‌شود. در این پردازنده، حالت یادگیری به دو صورت در زمان اجرا و قبل از زمان اجرا انجام‌پذیر است. در صورت اول، یک سخت‌افزار برای پروفایلر موردنیاز می‌باشد. بعلاوه تمامی مراحل لازم نظیر شناسایی HBB و یا تولید CI بر روی ریزپردازندهٔ پایه انجام می‌شوند. در دیگر سو و در حالت پیش از زمان اجرا، یک شبیه‌ساز به همراه تعدادی ابزار ویژه برای اجرای برنامه‌های کاربردی و پروفایل کردن آنها موردنیاز می‌باشد. در این حالت تمامی مراحل بر روی یک ماشین میزبان قابل اجرا بوده و مستقل از ریزپردازندهٔ پایه عمل می‌کند.

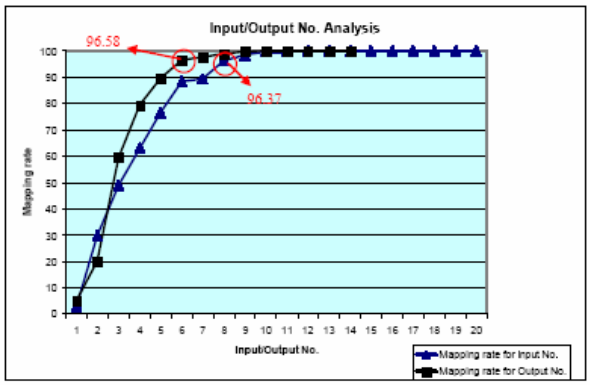
در هر صورت زمانی که عملیات پروفایل در زمان اجرا انجام شد، می‌توان برای مواردی که در آنها بین اجرای برنامه‌های کاربردی فاصله می‌افتد از حالت عملکرد عادی به حالت عملکرد یادگیری تغییر حالت داد و به تولید CI ها پرداخت.

۴- ارائهٔ یک معماری برای واحدهای عملیاتی قابل بازپیکربندی

در این بخش ابتدا معماری پیشین بررسی شده و سپس معماری مورد نظر ارائه خواهد شد.

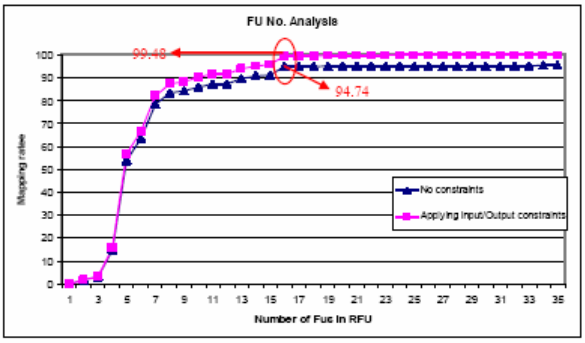
۴-۱- معماری پیشین با واحدهای عملیاتی یکسان[2]

برای تعیین تعداد ورودیها و خروجی‌های واحد عملیاتی قابل بازپیکربندی در ابتدا فرض می‌شود که هر نوع ساختار DFG قابل نگاشت بر روی این واحد باشد. سپس تحلیل‌هایی بر روی دستورالعملهای سفارشی به دست آمده از ۲۰ عدد از برنامه‌های آزمون Mibench [15] صورت گرفته است [2]. مطابق با نتایج به دست آمده که در شکل ۲ نیز نشان داده شده است، تعداد ورودی‌ها و خروجی‌های مناسب که علاوه بر داشتن نرخ نگاشت[۱۹] بالا، بتوانند منابع مصرفی را نیز کاهش دهد به ترتیب ۸ و ۶ خواهد بود. این امر بدین معنا است که با داشتن واحد عملیاتی قابل باز پیکربندی با ۶ ورودی و ۸ خروجی، DFG های با تعداد ورودی ۶ و تعداد خروجی ۸ و یا کمتر قابل نگاشت بر روی این واحد هستند [2].



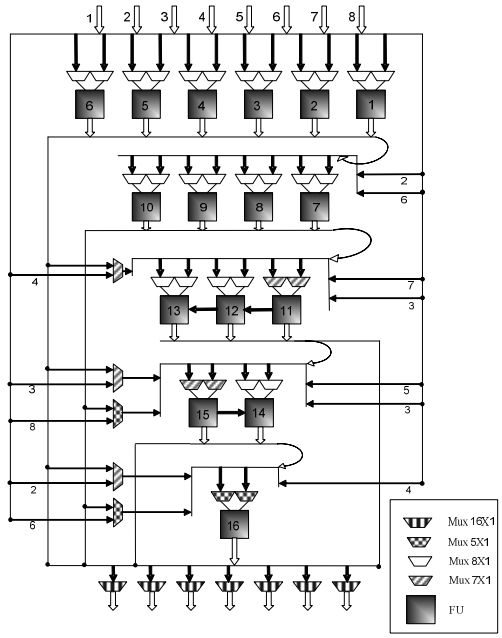
شکل ۲- نسبت نگاشت برای مقادیر مختلف تعداد ورودی و خروجی واحد عملیاتی قابل بازپیکربندی [2]

در مرحلهٔ بعدی باید تعداد واحدهای عملیاتی قابل بازپیکربندی در کل طرح بررسی و تحلیل شود. هدف از این تحلیل رسیدن به شرایطی است که بتوان در صورت امکان ضمن داشتن نرخ نگاشت بالا، از تعداد واحدهای عملیاتی کمتری نیز استفاده کرد. مطابق با بررسی‌های انجام شده که در شکل ۳ نیز نشان داده شده است، چنانچه محدودیت تعداد ورودی‌ها و خروجی‌ها در نظر گرفته شود، منحنی نرخ نگاشت به تعداد واحدهای عملیاتی تقریباً در عدد ۱۶ به حالت یکنواخت می‌رسد. به این ترتیب تعداد واحد عملیاتی مناسب برابر ۱۶ می باشد [2].



شکل ۳- نرخ نگاشت برای تعداد متفاوت واحدهای عملیاتی [2]

بعد از مشخص شدن تعداد ورودی‌ها، خروجی‌ها و واحدهای عملیاتی در معماری پیشنهاد شده، برای حفظ نرخ نگاشت بالا فرض بر این است که تمامی واحدهای عملیاتی یکسان و تنها قادر به اجرای یک دستورالعمل هستند. بر این اساس، تحلیل‌هایی مشابه با مواردی که ذکر شد برای تعیین عمق و نیز تعداد واحدهای هر ردیف انجام شده که حاصل این بررسی پیشنهاد ساختاری مشابه شکل ۴ می‌باشد. در این ساختار ۱۶ واحد عملیاتی یکسان وجود دارد که بر اساس بیتهای پیکربندی که به ورودی آنها اعمال می‌شود، دستورالعمل‌های تعیین شده را اجرا می‌کنند. اتصالات مابین این ۱۶ واحد از طریق بیتهای پیکربندی و به واسطه مالتی پلکسرها تعین می‌گردد.



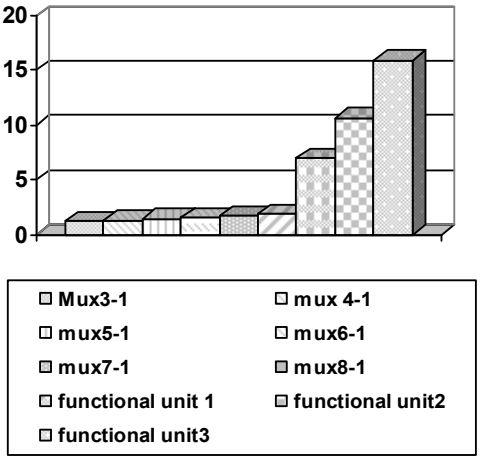
شکل ۴- معماری پیشنهادی برای واحد عملیاتی قابل بازپیکربندی با واحدهای یکسان

۴-۲- معماری پیشنهادی با واحدهای عملیاتی ناهمگون

یکی از مسائل مهم در تعیین ساختار واحد عملیاتی که در مقالات مورد توجه قرار نگرفته است، در نظر گرفتن مسیر بحرانی[۲۰] برای دستورالعمل‌های سفارشی نگاشت شده بر روی واحد عملیاتی است. به عنوان مثال فرض کنید یک دستورالعمل سفارشی برای اجرا بر روی واحد عملیاتی قابل بازپیکربندی به ۲ پالس ساعت زمان نیاز داشته باشد و این دستور در مجموع برای یک برنامهٔ مشخص ۲۰۰۰ بار در زمان اجرا تکرار شود. همچنین فرض کنید که بتوان با کاهش مسیر بحرانی، مدت زمان لازم برای اجرای این دستورالعمل را به یک پالس ساعت تقلیل داد. در این صورت مدت زمان اجرای برنامه به اندازهٔ ۲۰۰۰ پالس ساعت بهبود یافته که بسیار قابل‌توجه است.

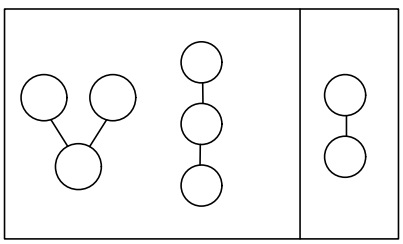
بر اساس موارد ذکر شده و برای کاهش مسیر بحرانی، تحلیل‌هایی بر روی نسبت مسیر بحرانی اجزای تشکیل‌دهندهٔ واحد عملیاتی قابل بازپیکربندی صورت گرفته است. نتایج بررسی‌ها نشان می‌دهد که تعداد مالتی‌پلکسرهایی که بر روی مسیر بحرانی گراف جریان دادهٔ

نگاشت شده قرار می‌گیرند، سهم قابل توجهی از مسیر بحرانی را به خود اختصاص می‌دهند که این موضوع در نتایج به دست آمده از سنتز مدار در تکنولوژی TSMC 0.18µ که در شکل ۵ نیز آورده شده است قابل مشاهده است. بنابراین با کاهش تعداد مالتی پلکسرها در طول مسیر بحرانی می‌توان قدم بزرگی برای بهبود این مسأله برداشت.



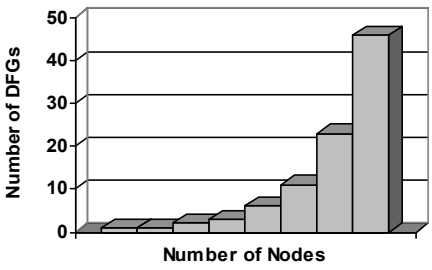
شکل ۵- طول مسیر بحرانی به دست آمده در تکنولوژی TSMC 0.18 µ برای مؤلفه‌های مختلف (به ترتیب از چپ به راست)

بر این اساس و به منظور کاهش تعداد مالتی‌پلکسرها در طول مسیر بحرانی، در این مقاله به ارائهٔ یک معماری برای واحد عملیاتی قابل بازپیکربند پرداخته شده است که واحدهای عملیاتی آن ناهمگون و بسته به نوع آن قادر به اجرای یک یا دو و یا سه دستورالعمل با هر ترتیب ممکن هستند. واحدهای عملیاتی دو تایی و سه تایی بر حسب بیتهای پیکربندی که به ورودی آنها اعمال می‌شود می‌توانند دو و یا سه دستورالعمل پشت سر هم را که نمایانگر زیرگرافهای جریان دادهٔ نشان داده شده در شکل ۶ هستد اجرا کنند.



شکل ۶- زیرگرافهای قابل اجرا توسط واحدهای عملیاتی دوتایی و سه تایی به ترتیب از چپ به راست

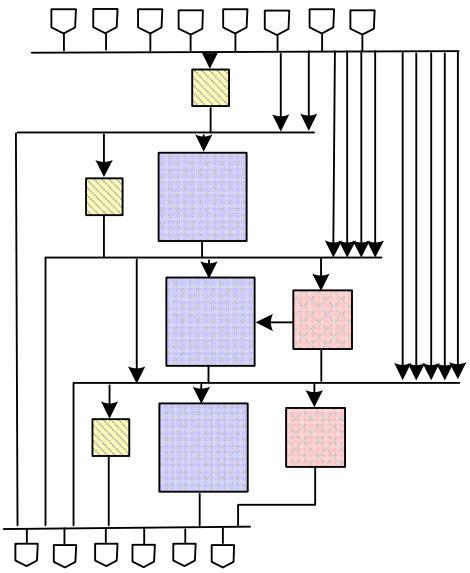
برای تعیین ساختار واحد عملیاتی قابل بازپیکربند پیشنهادی، ابتدا تحلیلی بر روی تعیین تعداد واحدهای سه تایی و دوتایی و نیز تنها صورت پذیرفته است. بر اساس نتایج به دست آمده از شکل ۲، از آنجاییکه با اعمال محدودیت تعداد ورودی و خروجی، ۸۵ درصد دستورالعمل‌های سفارشی کمتر از ۹ گره دارند لذا برای بدست آوردن نتایج دقیق، تمامی گرافهای جریان داده با تعداد گرهٔ کمتر از ۹ و با توپولوژی‌های مختلف تولید و ارزیابی شده‌اند (تعداد این گرافها در شکل ۷ نشان داده شده است). بعلاوه برای تحلیل دستورالعمل‌های با تعداد گرهٔ بیشتراز ۸ از روش آماری بر روی برنامه‌های آزمایشی Mibench استفاده شده است.



شکل ۷- تعداد DFG ها با توپولوژی متفاوت برای تعداد گره‌های کمتر از ۹ (محور افقی تعداد گره، محور عمودی تعداد DFG)

با استفاده از تحلیل‌های صورت گرفته برای تعیین تعداد واحدهای عملیاتی تنها، دوتایی و سه تایی و نیز نحوهٔ اتصال آنها ساختار نشان داده شده در شکل ۸ برای واحد عملیاتی قابل بازپیکربند پیشنهاد می‌شود که با استفاده از آن، تمامی دستورالعمل‌های با کمتر از ۹ گره و ۷۶ درصد دستورالعمل‌های بین ۹ و ۱۶ گره را می‌توان در آن نگاشت کرد که در مجموع نرخ نگاشت به ۹۶،۶٪ (۰،۸۵×۱+۰،۱۵×۰،۷۶) درصد ارتقاء می‌یابد.

علاوه بر این، با توجه به ساختار واحد عملیاتی همگون ارائه شده در [2] که در شکل ۴ نیز مشاهده می‌شود، با لحاظ کردن اتصالات ساختار مزبور، DFG هایی که ارتفاع آنها بیشتر از ۸ گره باشد قابل نگاشت بر روی این واحد نیستند که این محدودیت ارتفاع در معماری پیشنهادی با واحدهای ناهمگون به ۱۲ گره بهبود یافته است.



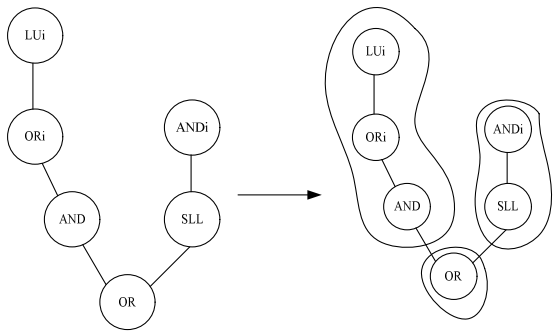
شکل ۸- معماری پیشنهاد شده جهت کاهش مسیر بحرانی با استفاده از واحدهای عملیاتی ناهمگون

۵- خوشه بندی گراف جریان داده

با توجه به آنکه واحد عملیاتی قابل بازپیکربندی دارای واحدهای عملیاتی متفاوتی می‌باشد، جهت نگاشت گراف جران داده بر روی واحد

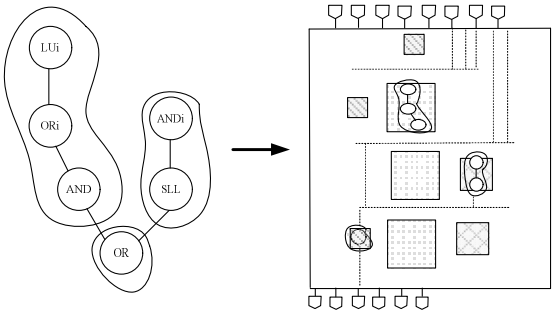
عملیاتی قابل باز پیکربندی، گراف جریان داده باید خوشه‌بندی[21] شود. برای خوشه‌بندی دستورالعمل‌های سفارشی در این مقاله از یک الگوریتم حریصانه[22] استفاده شده است به این ترتیب که ابتدا بزرگترین شاخهٔ گراف خوشه‌بندی می‌شود و سپس زیرگراف جدید تشکیل و این‌کار برای زیر گراف باقیمانده و با در نظر گرفتن تعداد منابع باقیمانده تکرار می‌گردد.

نکتهٔ قابل توجه در این الگوریتم، مسألهٔ خوشه‌بندی بهینه است. از آنجا که خوشه‌بندی تحت تأثیر محدودیت منابع اتصالات قرار می‌گیرد، برای بدست آوردن سرعت بالاتر در اجرای برنامه کاربردی، در هنگام طراحی معماری واحد عملیاتی قابل بازپیکربندی، برای ساختارهایی که بیشتر تکرار می‌شوند اولویت قرار داده می‌شود. به عبارت دیگر، در تعیین اتصالات ابتدا ساختارهایی که بیشتر تکرار شده‌اند لحاظ می‌شوند و به این ترتیب می توان آنها را به صورت بهینه خوشه‌بندی کرد. بر این اساس برای ساختارهایی که کمتر تکرار شده‌اند خوشه‌بندی تحت تأثیر محدودیت اتصالات قرار گرفته و بنابراین به صورت بهینه نخواهد بود. این مسأله در شکل ۹ نشان داده شده است.



شکل ۹- نحوهٔ خوشه بندی گراف جریان داده

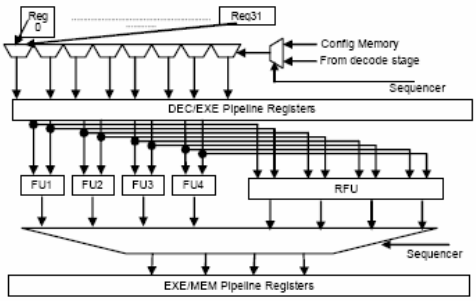
برای بدست آوردن نتایج بهتر، برای نگاشت دستورالعمل سفارشی با تعداد گرهٔ کمتر از ۹، از روشی که آن را "ثبت ساختار" نامیده‌ایم استفاده می‌شود. از آنجایی که در تحلیل‌های صورت گرفته تمامی ساختارهای گراف جریان داده با کمتر از ۹ گره تولید شده‌اند، برای هر DFG با توپولوژی متفاوت با استفاده از پیمایش سطحی پایین به بالا کد ویژه‌ای تولید می‌شود. سپس ساختارهای DFG به صورت دستی و بهینه بر روی واحد عملیاتی قابل بازپیکربندی نگاشت شده و با توجه به نحوهٔ نگاشت بیتهای پیکربندی مربوط به گراف به همراه کد مشخصهٔ توپولوژی، در یک جدول ثبت می‌شوند. به این ترتیب، در هنگام تولید دستورالعمل سفارشی با توجه به ساختار گراف و با استفاده از پیمایش سطحی پایین به بالا، می‌توان کد مربوط به این ساختار را تولید و با کدهای در جدول مقایسه کرد و سپس بیت‌های پیکربندی مربوط به این دستورالعمل سفارشی را از جدول ذکر شده استخراج کرد. برای گراف‌های دارای ۹ گره و بیشتر از یک نگاشت‌کننده که از یک الگوریتم جایابی ساده بهره گرفته شده است استفاده می شود. یک نمونه از نحوهٔ نگاشت یک دستورالعمل سفارشی بر روی واحد عملیاتی قابل بازپیکربندی در شکل ۱۰ نشان داده شده است.



شکل ۱۰- نحوهٔ نگاشت دستورالعمل سفارشی بر روی واحد عملیاتی قابل بازپیکربندی

۶- اتصال واحد عملیاتی به پردازندهٔ اصلی

پردازندهٔ اصلی AMBER یک پردازندهٔ RISC و به صورت 4-way می‌باشد. نحوهٔ اتصال واحد عملیاتی قابل بازپیکربندی با پردازندهٔ اصلی در شکل ۱۱ نشان شده است.



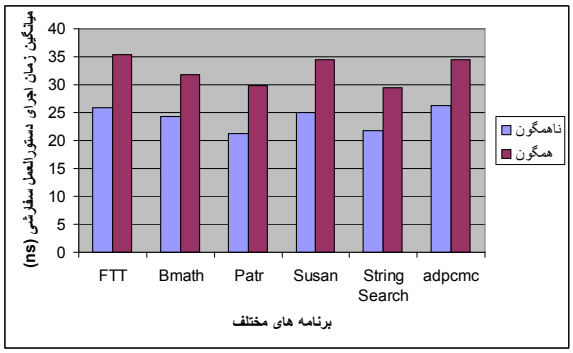
شکل ۱۱- نحوهٔ اتصال واحد عملیاتی قابل بازپیکربندی با پردازندهٔ اصلی

همانطوری که در این شکل نیز دیده می‌شود پورتهای ورودی و خروجی پردازندهٔ اصلی با پورتهای واحد عملیاتی قابل بازپیکربندی مشترک هستند. این نوع اتصال باعث می‌شود که به پورتهای اضافی نیاز نباشد ولی در مقابل امکان موازی سازی نیز از بین می‌رود. از آنجاییکه بخش زیادی از کد برنامه بر روی واحد قابل بازپیکربندی اجرا می‌شود، در عمل امکان موازی سازی کمتر پیش می‌آید. همانطوری که پیش‌تر نیز ذکر شد واحد عملیاتی به ۶ پورت خروجی نیاز دارد که در این صورت برای اضافه نشدن پورت نوشتن اضافی به رجیستر فایل، دو رجیستر به واحد عملیاتی قابل بازپیکربندی اضافه می‌گردد تا خروجیهای بیش از ۴، در آنها ذخیره شوند و در چرخهٔ بعدی در رجیستر فایل نوشته شوند.

۷- نتایج آزمایشات

مدت زمان حاصل از شبیه‌سازی اجرای ۶ برنامه آزمایشی Mibench بر روی دو معماری ارائه شده همگون و ناهمگون اندازه گیری شده است. میانگین مدت زمان اجرا دستورالعملهای سفارشی برنامه‌های مذکور در شکل ۱۲ نشان داده شده‌است. همانگونه که از شکل

پی‌داست، به علت کاهش مسیر بحرانی در معماری ناهمگون در اکثر موارد این معماری نتایج بسیار بهتری ارائه می‌کند. همچنین در معماری ناهمگون به علت کاهش تعداد مالتی‌پلکسرها تعداد بیتهای بازپیکربندی کاهش و در نتیجه منجر به کاهش حافظه مصرفی نیز شده است که این امر موجب کاهش توان مصرفی خواهد شد.



شکل ۱۲- میانگین مدت زمان اجرای دستورالعمل‌های سفارشی تولید شده از ۶ برنامهٔ آزمایشی Mibench بر روی دو معماری ارائه شده متشکل از واحدهای همگون و ناهمگون

۸- نتیجه‌گیری و کارهای آتی

با بکارگیری واحدهای عملیاتی قابل بازپیکربندی ناهمگون می‌توان زمان اجرای برنامه و نیز نرخ نگاشت دستورالعمل‌های سفارشی را به طور قابل ملاحظه‌ای بهبود بخشید. در این مقاله با استفاده از روش تحلیلی و نیز آماری، یک واحد عملیاتی قابل بازپیکربندی ناهمگون برای پردازندهٔ قابل توسعهٔ AMBER ارائه شده است که در آن نرخ نگاشت برابر با ۹۶،۶ درصد و سرعت اجرای دستورالعمل سفارشی نسبت به معماری پیشین بطور چشمگیری افزایش یافته است. این واحد عملیاتی دارای ۶ ورودی، ۸ خروجی، ۳ واحد عملیاتی سه تایی، ۲ واحد عملیاتی دوتایی و ۳ واحد عملیاتی تنها می‌باشد.

به عنوان کارهای آتی می‌توان برای افزایش سرعت اجرای برنامه در این واحد عملیاتی از دستورات ممیز شناور نیز پشتیبانی کرد. بعلاوه برای افزایش سرعت و نیز کاهش توان مصرفی استفاده از تکنیک‌های مختلف کاهش توان مصرفی در سیستمهای نهفته می‌تواند بسیار مفید باشد.

سپاسگزاری

نویسندگان این مقاله به این وسیله از آقای دکتر فرهاد مهدی‌پور بخاطر پیشنهادات سازندهٔ ایشان تشکر می‌کنند.

مراجع

[1] N. Clark, M. Kudlur, H. Park, S. Mahlke and K. Flautner, *"Application-Specific Processing on a General-Purpose Core via Transparent Instruction Set Customization"*, in Proc. MICRO-37, pp. 30-40, 2004.

[2] H. Noori, K. Murakami, and K. Inoue, *"A General Overview of an Adaptive Dynamic Extensible processor"*, in Proc. Workshop on Introspective Architecture, 2006.

[3] T. Miyamori and K. Olukotun, "*A Quantitative Analysis of Reconfigurable Coprocessors for Multimedia Applications,"* Symposium on FPGAs for Custom Computing Machines, pp. 2-11, 1998.

[4] N. Clark, J. Blome, M. Chu, S. Mahlke, S. Biles, and K. Flautner, *"An Architecture Framework for Transparent Instruction Set Customization in Embedded Processors,"* ISCA, pp. 272-283, 2005.

[5] P. Yu and T. Mitra, "*Scalable Custom Instructions Identification for Instruction-Set Extensible Processors,"* CASES, pp. 69-78, 2004.

[6] J. Lee et al., "*Reconfigurable ALU Array Architecture with Conditional Execution*" International SoC Design Conference (ISOCC), 2004.

[7] A. Gordon-Ross and F. Vahid, "*Frequent Loop Detection Using Efficient Non-Intrusive On-Chip Hardware,"* IEEE Transactions on Computers, Vol. 54, Issue 10, pp 1203 - 1215. 2005.

[8] M. H. Lee, H. Singh, G. Lu, N. Bagherzadeh, and F. J. Kurdahi , "*Design and implementation of the MorphoSys Reconfigurable Computing Processor,"* Journal of VLSI and Signal Processing-Systems for Signal, Image and Video Technology, pp. 147-164, Mar. 2000.

[9] C. Bobda, "*Synthesis of dataflow graphs for reconfigurable systems using temporal partitioning and temporal placement*", PhD thesis, University of Pederson, 2003.

[10] F. Mehdipour, M. Saheb Zamani, M. Sedighi, *"Reducing Reconfigurable Time of Reconfigurable Computing Systems in Integrated Temporal Partitioning and Physical Design Framework," IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2006.

[11] B. Mei, S. Vernalde, D. Verkest, and R. Lauwereinsg, "*Design Methodolody for a Tightly Coupled LIW/Reconfigurable Matrix Architecture: A Case Study,"* DATE, 2004.

[12] F. Mehdipour, H. Noori, M. Saheb Zamani, K. Murakami, M. Sedighi, *"An Integrated Temporal Partitioning and Mapping Framework for Handling Custom Instruction on a Reconfigurable Functional Unit,"* ACSA'06, 2006.

[13] http://www.cs.ucr.edu/~vahid/warp/

[14] http://www.Simplescalar.com

[15] Mibench, www.eecs.umich.edu/mibench

---

[1] Embedded Systems
[2] General purpose processors
[3] Application Specific Integrated Circuit (ASIC)
[4] Application Specific Instruction set Processor
[5] Instruction Set Architecture (ISA)
[6] Reconfigurable
[7] Fine Grain
[8] Coarse Grain
[9] Custom Instructions
[10] Hot Basic Block (HBB)
[11] Data Flow Graph
[12] Loosely coupled
[13] Tightly coupled
[14] Profiler

[15] Reconfigurable Functional Unit
[16] Scheduler
[17] Program Counter
[18] Taken
[19] Mapping rate
[20] Critical Path
[21] Clustering
[22] Greedy