مهندسی مکانیک شریف، (بهار ۱۳۹۷)
دوره‌ی ۳ - ۳۴، شماره‌ی ۱، ص. ۹۷-۱۰۴

Original Article

استفاده از سخت‌افزار FPGA برای تسریع محاسبات عددی جریان‌های پایا و ناپایا

عباس ابراهیمی * **(استادیار)**
محمد زندسلیمی (کارشناس ارشد)
دانشکده‌ی مهندسی هوافضا، دانشگاه صنعتی شریف

تسریع محاسبات عددی حل معادلات دیفرانسیل حاکم بر مسائل سیالاتی ــ به‌ویژه در جریان‌های ناپایا ــ از چالش‌های اصلی شبیه‌سازی این مسائل هستند. محاسبات عددی با استفاده از سخت‌افزارهای سنتی مانند CPU به‌صورت سری انجام می‌شوند و زمان‌برند. روش جدیدی که در این مقاله برای حل عددی معادلات دیفرانسیل پیشنهاد شده، به‌کمک بستر سخت‌افزاری FPGA، حل را به‌صورت سخت‌افزاری موازی‌سازی می‌کند. FPGA یک مدار تجمعی از بلوک‌های منطقی با قابلیت پیکربندی دوباره است. در این پژوهش از این سخت‌افزار جهت افزایش سرعت حل عددی معادله‌ی لاپلاس و جریان ناپایای کوئت یک‌بعدی به‌عنوان نمونه‌هایی از مسائل پایه‌ی دینامیک سیالات محاسباتی استفاده شده است. مسائل مورد نظر روی یک FPGA از نوع Zynq-۷۰۲۰ پیاده‌سازی شده و نتایج حل عددی آن‌ها با نتایج حاصل از برنامه‌ی رایانه‌یی روی CPU مقایسه شده است. محاسبات در برخی حالت‌ها تا ده برابر سریع‌تر از حل روی سخت‌افزار CPU انجام شده و از دقت حل نیز کاسته نشده است.

واژگان کلیدی: سخت‌افزار قابل برنامه‌ریزی، دینامیک سیالات محاسباتی، زبان توصیف سخت‌افزار، تسریع محاسبات عددی.

ebrahimi_a@sharif.ir
mohammad.zandsalimy@ae.sharif.ir

۱. مقدمه

طراحی وسایل پرنده و زیرمجموعه‌های آن به‌صورت سنتی براساس آزمون‌های تجربی زمان‌بر و هزینه‌بر در تونل باد انجام می‌شود. شبیه‌سازی جریان در رایانه به‌کمک روش‌های دینامیک سیالات محاسباتی (CFD) می‌تواند این آزمون‌ها را با صرف زمان و هزینه‌ی بسیار کم‌تری جبران کند. هزینه‌های محاسباتی بالا ناشی از حل معادلات دیفرانسیل حاکم بر حرکت جریان سیال شامل بقای جرم، مومنتوم و انرژی باعث شده که همچنان نتوان از دانش CFD به عنوان ابزار عهده‌دار کل فاز طراحی استفاده کرد. حل عددی این معادلات در اکثر مسائل واقعی به شبکه‌های محاسباتی ریز و با تعداد نقاط زیاد یا تعداد تکرارهای حل بالا نیاز دارد. به‌همین دلیل تلاش زیادی برای تسریع حل الگوریتم‌های معادلات دیفرانسیل حاکم بر جریان انجام شده است. نتایج پژوهش‌های انجام شده در این زمینه به محاسبات موازی[۱]، استفاده از تراشه‌های چندهسته‌یی، محاسبات با استفاده از تراشه‌های ویژه مانند GPU [۲] و غیره منجر شده است.[۱و۲] براساس بررسی‌های جدید،[۳-۷] محاسبات با استفاده از تراشه‌ی FPGA [۳]، که مبتنی بر پیاده‌سازی سخت‌افزاری معادلات است، نیز برای افزایش سرعت بسیار مناسب به نظر می‌رسد. طبیعت موازی‌سازی محاسبات عددی در سطح سیستم (استفاده از چند واحد سخت‌افزاری) و در سطح سخت‌افزار (پیکربندی یک واحد سخت‌افزاری)، هر دو درمورد FPGA قابل پیاده‌سازی است.

درگذشته از FPGAها معمولاً برای نمونه‌سازی سخت‌افزار یا منطق پیونددهنده‌ی سخت‌افزارهای دیگر استفاده می‌شد. اما امروزه با افزایش فرکانس سیگنال زمانی، می‌توان از آنها به‌عنوان پردازنده‌هایی با قابلیت انعطاف بالا و به‌تنهایی استفاده کرد. درگذشته بیشتر تحقیقاتی که در زمینه‌ی محاسبات ریاضی با استفاده از FPGA انجام‌شده، مربوط به کاربردهایی مانند ذخیره‌ی ماتریس‌ها و ضرب آنها، تبدیل فوریه‌ی عددی، پردازش سیگنال و کاربردهای دیجیتال دیگر بوده است. در زمینه‌ی CFD فعالیت‌های پژوهشی اندکی وجود دارد که از جمله آن‌ها می‌توان به حل یک میدان یک‌بعدی تراکم‌پذیر با استفاده از FPGA [۸] اشاره کرد. همچنین المان‌های اصلی ــ نظیر ماتریس‌ها، بردارها و عملیات ریاضی روی آنها ــ در یک حل عددی از جمله ضرب داخلی، ضرب ماتریسی و غیره روی سخت‌افزار پیاده‌سازی شده و سپس از این خواص برای ساخت یک حل‌گر جریان یک‌بعدی تراکم‌پذیر صریح استفاده شده است.[۹] سه سطح موازی‌سازی ممکن در محاسبات روی FPGA، شامل «موازی‌سازی در پایین‌ترین سطح محاسبات» (عملیات اصلی ریاضی)، «موازی‌سازی محاسبات توابع پیچیده‌تر»، و «نحوه‌ی شبکه‌بندی میدان و بهینه‌سازی الگوریتم حل» شرح داده شده است. در مطالعه‌یی دیگر،[۱۰] با برنامه‌ریزی یک سخت‌افزار FPGA به‌صورت واحدهایی با قابلیت انجام ضرب و جمع، مسئله‌ی

* نویسنده مسئول
تاریخ: دریافت ۱۳۹۵/۶/۲۴، اصلاحیه ۱۳۹۵/۹/۳، پذیرش ۱۳۹۵/۱۰/۴.
DOI:10.24200/J40.2018.6391

دوبعدی حفره در یک شبکه‌ی ۱۲ × ۸ حل شد و در فرکانس کارکرد ۶۰ مگاهرتز، به سرعت نهایی حل ۷٫۱۴ برابر سرعت حل در یک CPU ۳٫۲ گیگاهرتزی رسید. محققین ضمن امکان‌سنجیِ توسعه‌ی الگوریتم برای افزایش سرعت محاسبات در FPGA،[۱۱] روشی کاربردی برای حل معادلات شبه یک‌بعدی اویلر در مسئله‌ی لوله‌ی موج ضربه‌یی ارائه کرده‌اند. اندرس و همکاران[۱۲] در پژوهشی فرایندهای مختلف حل معادلات RANS از نظر زمان مورد نیاز را بررسی کردند و سپس با بهینه‌سازی کد، موازی‌کردن محاسبات و استفاده از سخت‌افزارهای جدید مانند GPU و FPGA، حل با سرعت بالاتری انجام شد. همچنین معادلات اویلر به روش حجم محدود با استفاده از یک زبان سطح بالا (زبان برنامه‌نویسی C) روی FPGA پیاده‌سازی شده[۱۳] و تا ۱۳٫۲۵ برابر، افزایش سرعت مشاهده شده است. در پژوهش دیگری، اثر استفاده از FPGA در افزایش سرعت محاسبات یک مسئله‌ی CFD خطی که به‌شکل تکراری حل می‌شد، بررسی شده است.[۱۴] با افزایش تعداد FPGAهایی که هم‌زمان مورد استفاده قرارگرفته‌اند، مشاهده شد که سرعت حل به‌صورت خطی افزایش می‌یابد. همچنین در زمینه‌ی احتراق، حل عددی جریان یک‌بعدی احتراقی داخل لوله‌های موتور بنزینی با استفاده از FPGA [۱۵] انجام شده و ساختار غیرهمگنی برای صرفه‌جویی بیشتر در استفاده از سخت‌افزار به کار گرفته شده است.

در نوشتار حاضر، توانایی سخت‌افزاری FPGA به‌عنوان روشی جدید و مورد توجه در آینده، برای افزایش سرعت حل مسائل مختلف دینامیک سیالات محاسباتی بدون کاهش دقت حل عددی مسئله‌ی بررسی شده است. بدین‌منظور، از این سخت‌افزار برای حل معادله‌ی لاپلاس و جریان ناپایای کوئت به‌عنوان مسائل نمونه‌ی دینامیک سیالات محاسباتی استفاده شده است. روش‌ها و تکنیک‌های برنامه‌نویسی مختلفی برای بهبود عملکرد محاسبات (تسریع حل) روی FPGA اعمال شده، و نتایج زمانی حل و دقت آن با حل روی رایانه‌های سنتی (استفاده از CPU) مقایسه شده است. تلاش اصلی در اینجا افزایش سرعت حل معادلات جریان با هر دو روش سخت‌افزاری و نرم‌افزاری است.

۲. سخت‌افزارهای با قابلیت پیکربندی مجدد

تراشه‌های الکترونیکی به دو دسته تقسیم می‌شوند: قابل‌پیکربندی (مانند FPGAها) و غیرقابل پیکربندی (مانند CPU و GPUها). سخت‌افزارهایی با قابلیت تغییر در پیکربندی[۴]، مدارهای مجتمعی هستند که از تعدادی گیت منطقی و اجزایی دیگر ساخته شده‌اند. عملکرد گیت‌ها و روابط و کلیدهای بین اجزای داخل این سخت‌افزار تغییرپذیرند و می‌توان پیکربندی و ساختار این سخت‌افزار را در حین اجرای برنامه تغییر داد. امروزه چنین سخت‌افزارهایی اجزای مختلفی دارند که از مهم‌ترین آنها سلول حافظه[۵] است. از این اجزای حافظه‌دار به‌عنوان جداول مراجعه[۶] استفاده می‌شود تا به‌کمک آنها کلیدهای کنترلی اتصالات بین اجزای مختلف، تنظیم و کنترل شود. برنامه‌یی که عملکرد هر گیت منطقی و حالت کلید را تعیین می‌کند، یک ساختار[۷] نامیده می‌شود. فرق اساسی این سخت‌افزارها در مقایسه با میکروپردازنده‌های معمولی، توانایی تغییر در مسیر انتقال اطلاعات و کنترل این فرایند است. علاوه‌بر آن می‌توان در هنگام اجرای برنامه، ساختار سخت‌افزار را با بارگذاری یک طراحی جدید، تغییر داد. یکی از معروف‌ترین انواع سخت‌افزار با قابلیت تغییر در پیکربندی، با نام FPGA موجود است. FPGA معمولاً با استفاده از زبان توصیف سخت‌افزاری[۸] مانند VHDL و VERILOG پیکربندی می‌شود. این معماری امکان پردازش و حل مسائل مختلف ازجمله معادلات دیفرانسیل را به‌صورت سخت‌افزاری فراهم می‌کند. CPU به‌عنوان شناخته‌شده‌ترین پردازنده‌ی مورد استفاده، سخت‌افزاری است که پس از ساخت، تغییر در ساختار آن ممکن نیست؛ به این معنی که این سخت‌افزار تنها کاری را انجام می‌دهد که در زمان طراحی مد نظر بوده، و تغییر در پیکربندی آن دیگر ممکن نخواهد بود.

۳. انواع معماری FPGA و CPU در کنار یکدیگر

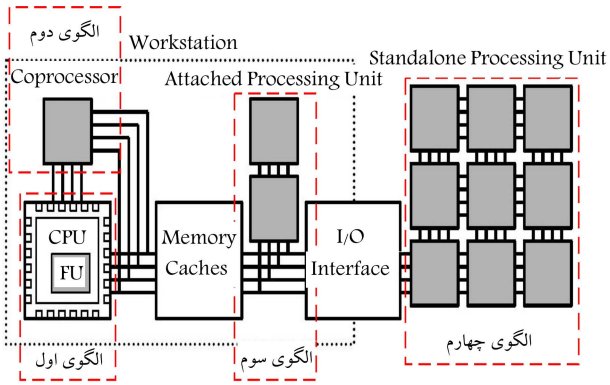
در سیستم حاوی FPGA و میکروپردازنده، روش‌های مختلفی برای اتصال آنها به هم وجود دارد. در به شکل ۱، چهار الگو برای برقراری ارتباط بین FPGA و CPU پیشنهاد شده است (در این شکل FPGAها با رنگ خاکستری مشخص شده‌اند).

در الگوی اول، سخت‌افزار قابل‌پیکربندی را می‌توان به تنهایی برای ایجاد واحدهای عملیاتی درون یک پردازنده میزبانی کرد. به این ترتیب یک محیط برنامه‌نویسی سنتی به وجود می‌آید که درون آن یک سری عملیات با قابلیت تغییر در طول زمان وجود دارد. واحد قابل پیکربندی در اینجا به‌عنوان واحد تابع پردازنده عمل می‌کند و در ورودی و خروجی‌های آن ثبت‌کننده‌هایی برای کنترل ورود و خروج داده قرار دارد.

در الگوی دوم، ممکن است یک واحد قابل پیکربندی به‌عنوان پردازنده مورد استفاده قرار بگیرد. در کل واحد قابل پیکربندی در این حالت بزرگ‌تر از حالت اول است و قابلیت انجام محاسبات را بدون نظارت پردازنده‌ی اصلی دارد. پردازنده‌ی اصلی واحد قابل‌پیکربندی را مقداردهی می‌کند و اطلاعات لازم برای شروع به کار را به آن می‌دهد یا نشانی قسمتی از حافظه را که این اطلاعات در آن ذخیره شده، به واحد قابل پیکربندی صادر می‌کند.

در الگوی سوم، ممکن است واحد قابل برنامه‌ریزی به‌عنوان پردازنده‌یی از یک سیستم چندپردازنده عمل کند. اطلاعات کش[۹] پردازنده‌ی اصلی برای واحد قابل برنامه‌نویسی، رویت‌ناپذیر است. به‌همین دلیل ارتباط بین این اجزا (مانند ارتباط برای انتقال اطلاعات پیکربندی، اطلاعات ورودی و نتایج) با سرعت پایین‌تر صورت می‌گیرد. این ساختار از این نظر که واحد قابل‌پیکربندی توانایی انجام محاسبات روی مقادیر زیاد اطلاعات را به‌صورت هم‌زمان دارد، مناسب است.

الگوی چهارم، سخت‌افزار قابل پیکربندی با یک پردازنده‌ی خارجی و مستقل



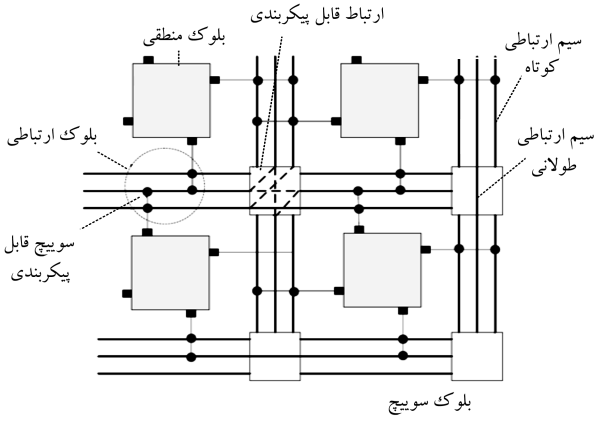
شکل ۱. پیکربندی‌های مختلف برای اتصال FPGA و CPU.[۱۶]

مرتبط است. در این ساختار انتقال اطلاعات (اگر وجود داشته باشد) بین این دو جزء به‌ندرت صورت می‌گیرد. این مدل مشابه حالتی است که ایستگاه‌های مختلف، پردازش اطلاعات را برای مدت بسیار طولانی انجام می‌دهند، بدون این که انتقال اطلاعات لازم باشد.

هرکدام از این ساختارها مزایا و معایبی دارند. هرچه مکان قرارگیری واحد قابل‌برنامه‌ریزی به پردازنده نزدیک‌تر باشد، برنامه‌های مختلف به‌دلیل زمان کوتاه انتقال اطلاعات، می‌توانند به تعداد دفعات بیشتر با آن ارتباط برقرار کنند. در این حالت سخت‌افزار نمی‌تواند بدون دخالت پردازنده برای مدت طولانی به انجام عملیات بپردازد و معمولاً مقدار سخت‌افزار در دسترس نیز محدود است. فاصله‌ی ارتباط طولانی‌تر به‌معنی سطح موازی‌سازی بالاتر است و به‌همین سبب سرعت و تعداد دفعات ارتباط کاهش خواهد یافت. در کاربردهایی که به دفعات ارتباط زیاد و با سرعت بالا لازم است، چنین ساختاری احتمالاً موجب کاهش زمان اجرای برنامه نخواهد شد. در پژوهش حاضر FPGA و CPU دو واحد جدا از هم هستند که با ارتباطات بسیار سریع و بدون واسطه به‌هم متصل شده‌اند؛ یعنی از الگوی دوم برای اتصال دو سخت‌افزار استفاده شده است.

۴. بلوک‌های ارتباطی درون FPGA

درون هر تراشه‌ی FPGA بلوک‌های منطقی زیادی وجود دارد که برای برقراری ارتباط بین آنها از بلوک‌ها و سوئیچ‌های ارتباطی استفاده می‌شود. ساختار این اتصالات به تراکم و تعداد بلوک‌های منطقی درون سخت‌افزار بستگی دارد. با توجه به شکل ۲، دو روش اصلی برای تعیین ساختار سیم‌کشی‌های محلی و کلی در یک سخت‌افزار قابل‌پیکربندی وجود دارد. یکی از این روش‌ها مربوط به استفاده از اتصالات کوتاه[۱۰] است که در آن، از یک سری سیم کوتاه برای انتقال اطلاعات محلی استفاده می‌شود. این اتصالات را می‌توان توسط جعبه‌سوئیچ‌ها به‌یکدیگر متصل کرد تا به‌عنوان سیم‌های طولانی‌تر مورد استفاده قرار گیرند. سیم‌های طولانی‌تر را می‌توان برای اتصال محلی و کلی استفاده کرد. در اتصال سلسله‌مراتبی[۱۱]، بعد از اتصال اجزای داخلی گروه‌های کوچک توسط سیم‌های کوتاه، برای اتصال این گروه‌ها از سیم‌های طولانی‌تر در مرز این گروه‌ها استفاده می‌شود. معمولاً اتصال سلسله‌مراتبی را برای حالتی که قرار است بیشتر ارتباطات بین اجزای داخلی گروه‌ها و قسمت کم‌تری از ارتباطات بین گروه‌ها باشد، بهینه‌سازی می‌کنند.



شکل ۲. شماتیک سیم‌کشی درون یک FPGA.[۱۷]

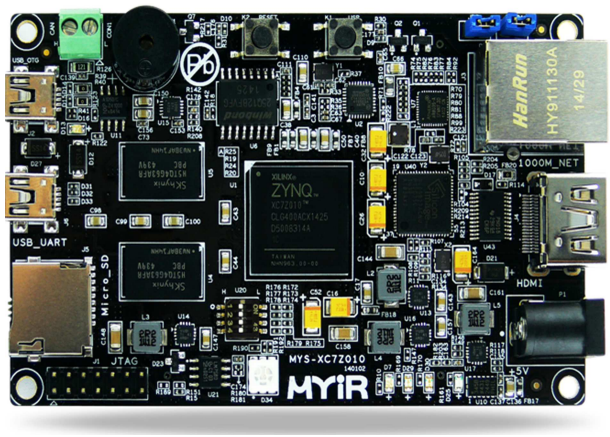
۵. معرفی سخت‌افزار مورد استفاده

سخت‌افزار مورد استفاده در این پژوهش از نوع SoC FPGA است، بدین معنا که داخل تراشه‌ی اصلی پردازشی هر دو سخت‌افزار قابل برنامه‌ریزی (FPGA) و پردازنده‌ی (CPU) وجود دارد. همچنین تراشه‌ی اصلی مورد استفاده، مدل Zynq-۷۰۲۰ از خانواده Zynq-۷۰۲۰ ساخت شرکت Xilinx است.[۵] این تراشه درکاربردهایی با حجم محاسبات بالا و مصرف توان کم، بهینه است. تراشه‌ی مورد نظر به‌تنهایی کاربرد خاصی نخواهد داشت، زیرا ارتباطات خارجی آن برای ریختن اطلاعات پیکربندی و استخراج نتایج هنوز قابل استفاده نیست. بهترین راه حل ممکن برای انجام این کار، نصب تراشه روی بردی است که دارای ارتباطات خارجی استاندارد است. برد مورد استفاده در این پژوهش با نام Z-turn شناخته شده و ساخت شرکت MYiR است. روی این برد ارتباطات خارجی USB، شبکه، حافظه‌ی فلش و غیره تعبیه شده است. تصویری از این برد و اجزای آن در شکل ۳ نشان داده شده است.

CPU مورد استفاده برای مقایسه‌ی نتایج، مدل Core i۷ Q۷۴۰، ساخت شرکت اینتل با بیشینه فرکانس عملکرد ۱٫۷۳ گیگاهرتز است. این CPU دارای چهار هسته‌ی پردازنده‌ی فیزیکی است. هرکدام از این هسته‌های فیزیکی می‌توانند به‌صورت هم‌زمان دو عملیات جداگانه را انجام دهند، پس در کل هشت هسته‌ی پردازشی مجازی دارد. بیشینه توان محاسبات این پردازنده برابر ۱۳٫۸۴ گیگافلاپ (یعنی ۱۳٫۸۴ میلیارد عملیات اعشاری در هر ثانیه!) است. البته باید توجه داشت که این مقدار اسمی، بیشینه توان محاسباتی CPU است و در عمل هیچ وقت امکان دست‌یابی به این عدد امکان‌پذیر نیست.

۶. روش معماری سخت‌افزار

انجام محاسبات با استفاده از سخت‌افزار FPGA نیازمند برنامه‌ریزی و طراحی آن به‌کمک زبان توصیف سخت‌افزار (HDL) است که در بین آنها دو زبان توصیف سخت‌افزار VHDL و Verilog بیشترین کاربرد را دارند. کدنویسی با زبان توصیف سخت‌افزار، حتی برای افراد متخصص، می‌تواند (بسته به نوع مسئله) بسیار زمان‌بر باشد. خوشبختانه تکنولوژی FPGA در زمینه‌ی ایجاد رابط کاربری و ابزارهای توصیف سخت‌افزار سطح بالا (HLL)[۱۲] در حال توسعه است.[۱۸] طراحی سخت‌افزار FPGA با استفاده از این نرم‌افزارها عموماً براساس زبان برنامه‌نویسی C نوشته



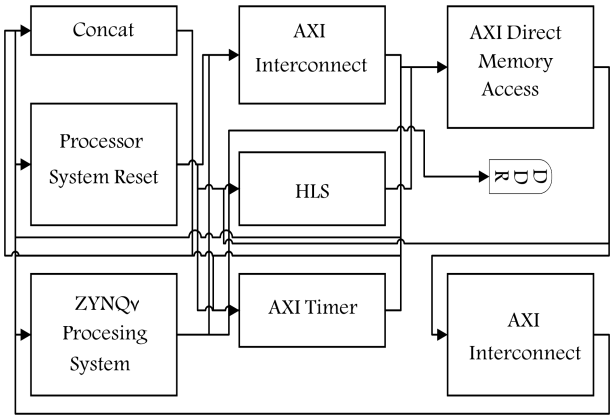
شکل ۳. برد Z-turn ساخت شرکت MYiR.

می‌شود و بنابراین استفاده از آن برای افراد غیرمتخصص در برنامه‌نویسی ساده‌تر است.

HLLها زمان توسعه‌ی نرم‌افزار را کاهش می‌دهند اما برنامه‌ی نوشته شده با این زبان‌ها برای اجرای بهتر روی سخت‌افزار نیازمند بهینه‌سازی دستی است. خانواده‌ی جدید FPGAها استفاده از اعداد اعشاری و انجام اعمال ریاضی روی آنها را ساده‌تر می‌کنند و به این ترتیب بهینه‌سازی در درون آنها نهفته است.

مراحل مختلفی در فرایند طراحی یک مدار وجود دارد. مشخص‌کردن مدار[۱۳]، فرایند تعریف توابعی است که قرار است روی سخت‌افزار قابل‌پیکربندی قرار بگیرند. این کار را می‌توان با نوشتن برنامه‌یی به زبان C که تابعیت الگوریتم را توضیح می‌دهد، انجام داد. از طرف دیگر ممکن است همین‌کار با مشخص کردن ورودی و خروجی‌ها و عملیات یک‌یک بلوک‌های منطقی درون سخت‌افزار انجام شود که دشوارتر از حالت قبل است. ساخت مدار با استفاده از واحدهای از پیش ساخته‌شده مانند جمع‌کننده‌ها و ضرب‌کننده‌ها نیز روشی است که از نظر دشوار بودن، بین دو روش بالا قرار می‌گیرد.

فرایند اصلی استفاده شده برای طراحی سخت‌افزار در این پژوهش مخصوص تراشه‌های Zynq-۷۰۰۰ است که توسط شرکت زایلینکس پیشنهاد شده است. سه نرم‌افزار اصلی در طول این فرایند استفاده می‌شوند که هر سه توسط مهندسین شرکت زایلینکس و مخصوص محصولات همین شرکت توسعه داده شده‌اند. این سه نرم‌افزار شامل Vivado HLS، Vivado و Xilinx SDK هستند. فرایند طراحی در اینجا با استفاده از زبان توصیف سخت‌افزار سطح بالا انجام می‌شود. طراحی سخت‌افزار با بهره‌گیری از یک سری بلوک‌های پیش‌ساخته با نام IP [۱۴] و برقراری اتصالات بین آنها به‌صورت گرافیکی در نرم‌افزار Vivado انجام می‌شود. در شکل ۴ نمونه ساختار طراحی‌شده برای حل مسئله‌ی لاپلاس و IPهای استفاده شده نشان داده شده است. هرکدام از این بلوک‌ها کاربرد خاصی دارند که در ادامه توضیح آنها ارائه شده است. بلوک HLS حل‌گر معادله‌ی لاپلاس است که با استفاده از یک برنامه نوشته شده با زبان C++ در نرم‌افزار Vivado HLS برای پژوهش حاضر توسعه داده شده است. این IP در ورودی خود داده‌های اولیه‌ی مسئله‌ی لاپلاس را دریافت می‌کند و بعد از انجام فرایند حل، نتایج را به بیرون از بلوک می‌فرستد. بلوک AXI Timer یک IP توسعه یافته توسط شرکت زایلینکس برای اندازه‌گیری تعداد دقیق ارتعاش تراشه (سخت‌افزار طراحی شده) در هنگام کارکرد آن است. از این بلوک برای محاسبه‌ی دقیق زمان طی شده در حل هر مسئله استفاده می‌شود. همچنین از بلوک ZYNQ۷ Processing System برای راه‌اندازی میکروپردازنده‌های موجود



شکل ۴. طرح بلوکی برای حل مساله لاپلاس.

روی تراشه‌ی Zynq-۷۰۲۰ استفاده می‌شود. با استفاده از این IP می‌توان از طریق بلوک ارتباطی پرسرعت AXI Interconnect با بلوک HLS در ارتباط بود و از این طریق منطق قابل پیکربندی را کنترل کرد. برای بررسی کاربرد سایر IPها منابعی در دسترس است.[۱۹]

۷. مسائل مورد بررسی

در این پژوهش، معادله‌ی لاپلاس و معادله‌ی ناپایای جریان کوئت یک‌بعدی به‌عنوان نمونه‌هایی از مسائل پایه‌ی دینامیک سیالات محاسباتی برای پیاده‌سازی روی سخت‌افزار FPGA و ارزیابی سرعت حل عددی استفاده شده است. یکی از مسائل اولیه و پرکاربرد در زمینه‌ی دینامیک سیالات محاسباتی، مسئله‌ی لاپلاس است. معادله‌ی حاکم بر جریان سیال تراکم‌ناپذیر غیرلزج (جریان پتانسیل) و همچنین معادله‌ی حاکم بر یک مسئله‌ی ساده‌ی انتقال حرارت پایای دوبعدی بدون جمله‌ی چشمه‌ی گرما، معادله‌ی لاپلاس است. در مسئله‌ی جریان پتانسیل با معلوم بودن شرایط مرزی و حل معادله‌ی لاپلاس، مقادیر تابع جریان محاسبه می‌شود که با مشتق‌گیری از این پارامتر می‌توان میدان سرعت در جریان را به‌دست آورد. درمورد مسئله‌ی انتقال حرارت پایای دوبعدی، بعد از حل معادله‌ی لاپلاس، میدان دما به‌صورت مستقیم محاسبه می‌شود. این مسئله به روش‌های متفاوتی قابل حل است و با توجه به این که حل تحلیلی نیز برای آن وجود دارد، دقت نتایج را می‌توان اعتبارسنجی کرد. معادله‌ی لاپلاس در مختصات دکارتی و دوبعدی به‌صورت رابطه‌ی ۱ نوشته می‌شود.

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = 0 \qquad (1)$$

شرایط مرزی برای این مساله به‌صورت، $\psi(x,0) = x$، $\psi(x,\pi) = 0$، $\psi_x(0,y) = 0$ و $\psi_x(\pi,y) = 0$ درنظر گرفته شده است. با استفاده از روش جداسازی متغیرها، حل تحلیلی و دقیق این مساله به‌صورت رابطه‌ی ۲ نوشته می‌شود:[۲۰]

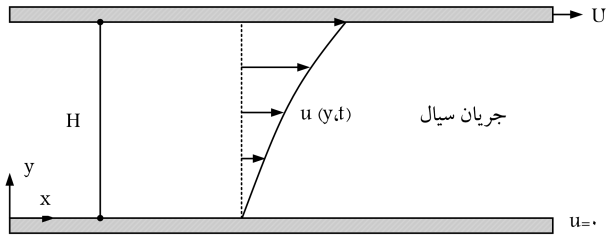
$$\psi(x,y) = \frac{1}{2}(\pi - y) + \sum_{n=1}^{\infty} \frac{2}{\pi n^2}[(-1)^2 - 1] \frac{\sinh(n(\pi-1))}{\sinh(n\pi)} \cos(nx) \qquad (2)$$

برای انجام حل عددی این مسئله از روش نقطه‌یی گوس سایدل[۱۵] استفاده می‌شود. معادله‌ی ۱ بعد از گسسته‌سازی با این روش به رابطه‌ی ۳ تغییر می‌یابد[۲۱] که از آن برای حل عددی مسئله‌ی لاپلاس استفاده خواهد شد:

$$\psi_{i,j}^{k+1} = \frac{1}{2(1+\beta^2)} \left(\psi_{i-1,j}^{k+1} + \psi_{i+1,j}^{k} + \beta^2(\psi_{i,j-1}^{k+1} + \psi_{i,j+1}^{k})\right) \qquad (3)$$

که در آن، k گام زمانی حل، پسوندهای i و j نماینده‌ی نقطه‌یی از شبکه‌ی عددی و β نشان‌دهنده‌ی نسبت اندازه شبکه در راستای x به راستای y است.

مسئله‌ی دیگری که در این پژوهش مورد بررسی قرار می‌گیرد، شبیه‌سازی جریان ناپایای کوئت یک‌بعدی است. این مسئله شامل سیالی با لزجت مشخص است که بین دو صفحه‌ی بی‌نهایت طولانی قرار داده شده است. در لحظه‌ی صفر، صفحه‌ی بالایی با سرعت ثابت U به حرکت درمی‌آید. به‌علت وجود لزجت سیال، اثر شرط



شکل ۵. کانتور ψ در حل عددی مساله لاپلاس.

مرزی متحرک به مرور زمان در محیط نفوذ کرده و پروفیل سرعت تغییر می‌کند (شکل ۵). این مسئله جریان یک بعدی ناپایای سیال لزج تراکم‌ناپذیر را نشان می‌دهد که با نام جریان کوئت شناخته می‌شود. معادله‌ی حاکم بر مسئله از ساده‌سازی معادله‌ی بقای مومنتوم در جهت x به‌دست می‌آید.[۲۲] این معادله به‌شکل بی‌بعد در رابطه‌ی ۴ آمده است:

$$\frac{\partial u(y,t)}{\partial t} = \frac{1}{\mathrm{Re}} \frac{\partial^2 u(y,t)}{\partial y^2} \qquad (۴)$$

که در آن شرایط مرزی و اولیه چنین تعریف می‌شود:

$$\begin{cases} u(y<0,0)=0, & u(0,0)=0 \\ u(0,t)=0, & u(0,t)=1 \end{cases} \qquad (۵)$$

همچنین در معادله‌ی ۴ عدد رینولدز با استفاده از رابطه‌ی ۶ محاسبه می‌شود:

$$\mathrm{Re} = \frac{HU}{\nu} \qquad (۶)$$

معادله‌ی ۴ با روش اویلر صریح گسسته‌سازی می‌شود. برای این کار، مشتق زمانی به‌صورت تقریب مرتبه‌ی اول پیشرو و مشتق مکانی با تقریب اختلاف محدود مرکزی گسسته‌سازی می‌شود (معادله‌ی ۷):

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\mathrm{Re}(\Delta y)^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) \qquad (۷)$$

که در آن n نمایانگرگام زمانی حل، j نمایانگر شماره‌ی نقطه روی شبکه‌ی عددی، ΔT اندازه‌ی هرگام زمانی و Δy اندازه‌ی هر واحد از شبکه‌ی عددی است. حل تحلیلی (پروفیل سرعت بی‌بعد) جریان ناپایای کوئت به‌صورت معادله‌ی ۸ نوشته می‌شود.[۲۱] در شرایط حدی با گذشت زمان و پیشروی حل به مقداری که t خیلی بزرگ شود (شرایط پایا)، جواب مسئله به‌شکل $u(y,t)=y$ خلاصه می‌شود که نشان‌دهنده‌ی یک رابطه‌ی خطی بین y و u است.

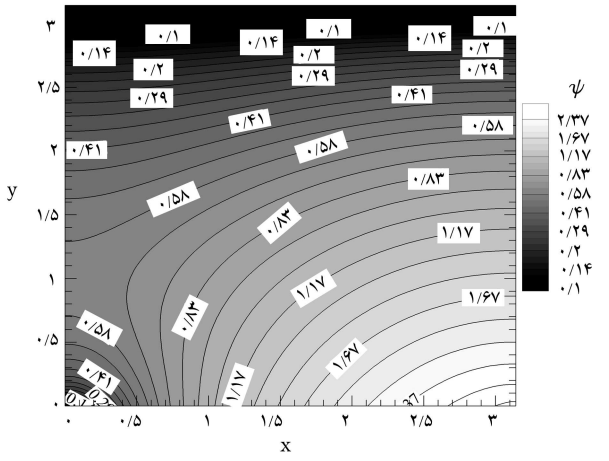
$$u(y,t) = y + \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}\left\{(-1)^n \exp(-n^2\pi^2\frac{tU}{\mathrm{Re}H})\sin(n\pi y)\right\} \qquad (۸)$$

۸. نتایج حل عددی

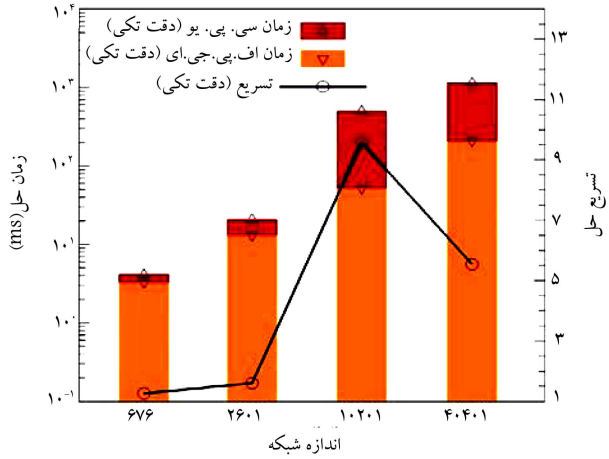
حل عددی معادله‌ی لاپلاس روی FPGA و CPU برای شبکه‌ی محاسباتی با ابعاد ۱۰۱×۱۰۱ انجام شده است. نتایج خطوط هم‌تراز برای تابع جریان ψ از حل معادله‌ی ۳ با دقت مضاعف[۱۶] و اندازه شبکه‌ی $\frac{\pi}{۱۰۰}$، بعد از ۱۰۰۰۰ گام حل در شکل ۶ آمده است. برای حل تحلیلی این مسئله، سری معادله‌ی ۲ تا $n=۱۰۰۰$ محاسبه شده و در جدول ۱ نرم خطای بین داده‌های حل عددی و تحلیلی آورده

جدول ۱. نرم خطای حل تحلیلی و عددی مساله لاپلاس.

اندازه شبکه	گام حل	نرم اول خطا	نرم دوم خطا	نرم بی‌نهایت خطا
	۱۰۰۰	۰٫۳۸۳۶	۰٫۴۳۶۶	۰٫۷۷۴۳
$\pi/۱۰۰$	۵۰۰۰	۰٫۰۵۳۷	۰٫۰۶۰۱	۰٫۰۹۳۷
	۱۰۰۰۰	۰٫۰۰۵	۰٫۰۰۶۳	۰٫۰۱۸



شکل ۶. کانتور ψ در حل عددی مساله لاپلاس.



شکل ۷. نمودار زمان و تسریع حل مساله لاپلاس.

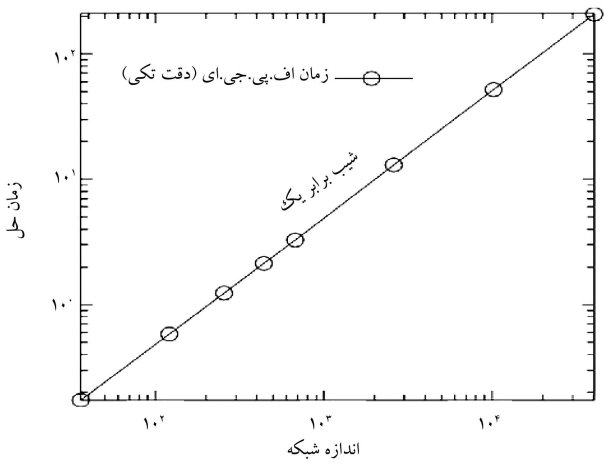
شده است. مقادیرکم نرم خطای موجود در این جدول نشان‌دهنده‌ی صحت حل عددی است. معیار هم‌گرایی این مسئله، نرم خطای بین داده‌های حل تحلیلی و حل عددی است. نرم خطا بین دو ماتریس A و B (هر دو با اندازه $m \times n$) با استفاده از روابط ۹ تا ۱۱ محاسبه شده است.

$$L_1 = \frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n}|A_{ij}-B_{ij}| \qquad (۹)$$

$$L_2 = \left[\frac{1}{nm}\sum_{i=1}^{m}\sum_{j=1}^{n}(A_{ij}-B_{ij})^2\right]^{\frac{1}{2}} \qquad (۱۰)$$

$$L_\infty = \max\left(|A_{ij}-B_{ij}|\right) \qquad (۱۱)$$

در شکل ۷ نمودار میله‌یی زمان حل مسئله (محور عمودی سمت چپ) و میزان تسریع محاسبات (محور عمودی سمت راست) با استفاده از CPU و FPGA برای



شکل ۸. زمان حل برحسب تعداد نقاط شبکه.

دقت تکی برحسب تعداد نقاط شبکه‌ی محاسباتی آمده است. مشاهده می‌شود که در یکی از حالت‌ها تا ۱۰ برابر نیز افزایش سرعت به‌دست آمده است. از آنجا که برای به‌دست آوردن جواب عددی مناسب و دقیق، لازم است حل به تعداد گام‌های زیادی پیش برود (بیشتر از ۱۰۰۰ گام حل)، حل کامل این مسئله روی سخت‌افزار قابل پیکربندی، به‌علت وجود سه حلقه تکرار منطقی تودرتو در برنامه‌ی رایانه‌یی آن، بهینه نخواهد بود. بنابراین بهتر است فقط افزایش سرعت حل یک گام حل روی FPGA بررسی شود. البته چنانچه از نظر قدرت و فرکانس پردازش سخت‌افزار قوی‌تری در اختیار بود، می‌توان مسئله را به‌صورت کامل روی آن حل کرد.

نمودار زمان حل مسئله‌ی لاپلاس روی FPGA با دقت تکی در مقابل تعداد نقاط شبکه‌ی عددی در شکل ۸ آمده است. چنان که مشاهده می‌شود، زمان حل روی سخت‌افزار قابل‌پیکربندی با تعداد نقاط شبکه رابطه‌ی مستقیم و خطی دارد. شیب این خط برابر ۱ بوده که ناشی از خطی بودن معادله‌ی لاپلاس است. به این ترتیب می‌توان زمان حل همین مسئله را در ابعاد شبکه‌ی بزرگ‌تر، در صورت وجود سخت‌افزار با سطح محاسباتی بالاتر، تخمین‌زد.

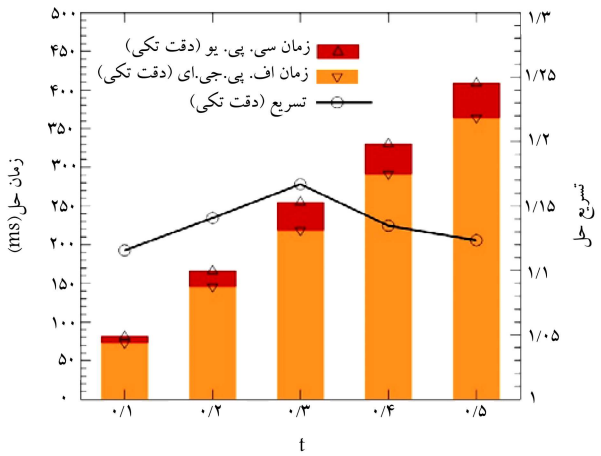
در پژوهش‌های قبلی که در زمینه‌ی تسریع محاسبات عددی با استفاده از FPGA انجام شده‌اند، یکی از مسائل مهم مورد بررسی حل مسئله‌ی لاپلاس بوده است. در این بخش نتایج زمانی به‌دست آمده از حل مسئله‌ی لاپلاس با سایر مراجع مقایسه شده است. حل یک نقطه از شبکه‌ی عددی این مسئله روی هر دو سخت‌افزار CPU و FPGA انجام‌شده است.[۹] سخت‌افزارهای استفاده شده برای این منظور، با سخت‌افزارهای مورد استفاده در پژوهش حاضر متفاوت است. به‌همین دلیل لازم است پارامتری عمومی برای مقایسه‌ی نتایج انتخاب شود. متغیری که می‌تواند توان محاسباتی را صرف‌نظر از سخت‌افزار استفاده شده مشخص کند، تعداد دوره‌های فرکانس زمانی (کلاک[۱۷]) لازم برای حل مسئله است. تعداد کلاک لازم برای تکمیل فرایند حل، یک پارامتر بی‌بعد است و از این نظر نیز نسبت‌به سایر متغیرهای بابعد برتری دارد. در جدول ۲ نتایج زمانی و تعداد کلاک در حل یک نقطه از شبکه‌ی عددی مسئله‌ی لاپلاس آمده است. تمامی آزمایش‌ها با دقت اعشاری معمولی انجام شده است. پژوهش دیگری نیز در این زمینه انجام شده[۱۴] که نتایج آن نیز برای مقایسه ارائه شده است. روش‌های طراحی سخت‌افزار[۹،۱۴] و پژوهش حاضر نیز با هم تفاوت دارند. اما از آنجا که هدف نهایی، تسریع محاسبات عددی مسئله‌ی لاپلاس است، بهترین عملکرد هریک از این پژوهش‌ها مدنظر قرارگرفته و روش طراحی سخت‌افزار با هم مقایسه نمی‌شود.

جدول ۲. نتایج زمانی حل یک نقطه از شبکه لاپلاس.

پژوهش	سخت‌افزار	فرکانس پردازش (MHz)	زمان حل (ns)	تعداد کلاک
[۹]	CPU	۳۰۰۰	۴۳	۱۳۰
	FPGA	۸۲۲٫۳۷	۱٫۲۲	۱
[۱۴]	CPU	۲۹۳۰	۸۹٫۱۱	۲۶۱٫۱
	FPGA	۱۳۳	۷٫۵	۱
پژوهش حاضر	CPU	۱۷۳۰	۶٫۱۲	۱۰٫۵۸۲۲
	FPGA	۲۵۰	۴٫۸۳	۱٫۲۰۷۱

جدول ۳. بررسی دقت حل عددی مساله کوئت.

y	t	حل تحلیلی $u \times ۱۰$	حل عددی $u \times ۱۰$	درصد خطای نسبی
۰٫۳	۰٫۱	۱٫۱۷۳۷	۱٫۱۳۸۷	۳٫۰۷
	۰٫۳	۲٫۷۵۹۵	۲٫۷۳۳۳	۰٫۵۹
	۰٫۵	۲٫۹۶۶۶	۲٫۹۶۲۹	۰٫۱۲
۰٫۷	۰٫۱	۵٫۰۶۶۱	۵٫۰۲۱۹	۰٫۸۸
	۰٫۳	۶٫۷۴۹۴	۶٫۷۳۳۳	۰٫۲۳
	۰٫۵	۶٫۹۶۶۶	۶٫۹۶۲۹	۰٫۰۵



شکل ۹. نمودار مدت زمان محاسبات و میزان تسریع در زمان‌های مختلف حل مساله کوئت.

نتایج حل مسئله‌ی کوئت با دقت مضاعف در $\Delta t = ۱۰^{-۷}$ و همچنین اختلاف آنها با مقادیر حل تحلیلی از رابطه ۸، در جدول ۳ آورده شده است. برای محاسبه مقدار تحلیلی حل مسئله‌ی کوئت، سری معادله‌ی ۸ تا $n = ۱۰۰۰$ محاسبه شده است. از مقادیر خطای محاسبه شده در این جدول می‌توان نتیجه گرفت که حل عددی مسئله‌ی کوئت درست انجام شده است. در شکل ۹ نمودار میله‌یی زمان حل مسئله‌ی کوئت (محور عمودی سمت چپ) با استفاده از CPU و FPGA در دقت تکی آمده است. در این شکل همچنین میزان افزایش سرعت حل (محور عمودی سمت راست) در هر زمان مشخص شده است.

۹. نتیجه‌گیری

در پژوهش‌های قبلی معمولاً از سخت‌افزارهای قابل پیکربندی قوی از نظر قدرت و فرکانس پردازش یا سیستم‌های موازی متشکل از چندین FPGA برای تسریع محاسبات استفاده شده است. سخت‌افزار مورد استفاده در اینجا از نظر تعداد و تراکم اجزای منطقی و محاسباتی و بیشینه فرکانس پردازش از سخت‌افزارهای مورد استفاده در پژوهش‌های گذشته ضعیف‌تر است. با این وجود در حل مسئله‌ی لاپلاس تا ۱۰ برابر تسریع نیز مشاهده شد. روش‌های مختلفی برای افزایش سرعت محاسبات استفاده شده است، به جز تغییر سخت‌افزار محاسباتی مورد استفاده از CPU به FPGA، روش‌های الگوریتمی و بهینه‌سازی‌های نرم‌افزاری زیادی برای رسیدن به این هدف استفاده شد. همچنین دقت انجام حل با استفاده از FPGA در پژوهش حاضر نسبت به CPU کاهش نیافته است. یکی از دلایل عدم افزایش سرعت محاسبات به اندازه زیاد، همین موضوع است. درصورتی که دقت‌های پایین‌تر برای اعداد در حل روی FPGA انتخاب شود، باید به این نکته توجه داشت که میزان اختلاف نتایج نهایی به‌دست آمده از FPGA با حل CPU به اندازه قابل قبولی باشد. در این حالت سطح سخت‌افزاری مورد استفاده برای یک مسئله‌ی خاص کاهش یافته و می‌توان شبکه‌های ریزتر را نیز برای حل انتخاب کرد. همچنین زمان حل نیز به‌علت نیازکم‌تر به محاسبات منطقی، کاهش خواهد یافت.

فهرست علائم

ψ: تابع جریان؛

β: نسبت اندازه‌های شبکه محاسباتی؛

k: گام تکرار حل؛

u: مولفه‌ی افقی سرعت؛

H: ارتفاع؛

U: سرعت بیشینه؛

ν: لزجت سینماتیکی؛

پانوشت‌ها

1. parallel computing
2. graphical processing unit
3. field programmable gate array
4. reconfigurable hardware
5. memory cell
6. lookup table
7. configuration
8. hardware description language
9. cache
10. segmented routing
11. hierarchical routing
12. high level language
13. circuit specification
14. intellectual property
15. point Gauss Seidel
16. double
17. Clock

منابع (References)

1. Liu, X., Zhong, Z. and Xu, K. "A hybrid solution method for CFD applications on GPU-accelerated hybrid HPC platforms", *Future Generation Computer Systems*, **56**, pp. 759-765 (2016).
2. Alevras, P. and Yurchenko, D. "GPU computing for accelerating the numerical Path Integration approach", *Computers & Structures*, **171**, pp. 46-53 2016.
3. Smari, W.W., Bakhouya, M., Fiore, S., Aloisio, G. "New advances in high performance computing and simulation: Parallel and distributed systems, algorithms, and applications", *Councurrency and Computation: Practice and Experience*, **28**, pp. 2024-2030 (2016).
4. Konstantinos, K., Helal, A.E., Verma, A. and Feng, W.C. "Bridging the performance-programmability gap for FPGAs via OpenCL: A case study with OpenDwarfs", Computer Science Technical Reports, Virginia Tech., TR-16-03 (2016).
5. Sugimoto, N., Miyajima, T., Sakai, R., Osana, Y., Fujita, N. and Amano, H. "Zynq cluster for CFD parametric survey", *Proc. of the International Simposium on Applied Reconfigurable Computing*, Springer International Publishing, pp. 287-299 (2016).
6. Raase, S. and Nordstrom, T. "On the use of a many-core processor for computational fluid dynamics simulations", *Procedia Computer Science*, **51**, pp. 1403-1412 (2015).
7. AbuTalip, M.S., Akamine, T., Hatto, M., Amano, H., Osana, Y. and Fujita, N. "Adaptive flux calculation scheme in advection term computation using partial reconfiguration", *International Journal of Networking and Computing*, **3**(2), pp. 289-306 (2013).
8. Hauser, T. "A flow solver for a reconfigurable FPGA-based hypercomputer", *43rd AIAA Aerospace Sciences Meeting and Exhibit,* Reno, Nevada, AIAA 2005-1382 (2005).
9. Nunez, R., Gonzalez, J. and Camberos, J. "Large-scale numerical solution of partial differential equations with reconfigurable computing", *36th AIAA Thermophysics Conference*, AIAA 2007-4085 (2007).
10. Sano, K., Iizuka, T. and Yamamoto, S. "Systolic architecture for computational fluid dynamics on FPGAs", in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '07)*, pp. 107-116 (2007).

11. Andrés, E., Carreras, C., Caffarena, G., Molina, M.D.C. Nieto-Taladriz, O. and Palacios, F. "A methodology for CFD acceleration through reconfigurable hardware", *46th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA 2008-481 (2008).

12. Andres, E., Widhalm, M. and Caloto, A. "Achieving high speed CFD simulations: Optimization, parallelization, and FPGA acceleration for the unstructured DLR TAU code", *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*, Orlando, Florida, AIAA 2009-759 (2009).

13. Sanchez-Roman, D., Sutter, G., Lopez-Buedo, S., Gonzalez, I., Gomez-Arribas, F.J. and Aracil, J. "An euler solver accelerator in FPGA for computational fluid dynamics applications", in *Programmable Logic (SPL), 2011 VII Southern Conference on,* pp. 149-154 (2011).

14. Sano, K., Hatsuda, Y. and Yamamoto, S. "Performance evaluation of FPGA-based custom accelerators for iterative linear-equation solvers", *20th AIAA Computational Fluid Dynamics Conference Honolulu*, Hawaii, AIAA 2011-3223 (2011).

15. Liu, I., Lee, E.A., Viele, M., Wang, G. and Andrade, H. "A heterogeneous architecture for evaluating real-time one-dimensional computational fluid dynamics on FPGAs", *IEEE 20th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM),* pp. 125-132 (2012).

16. Gokhale, M.B. and Graham, P.S., *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*, First Edition, Springer (2005).

17. Singh, P. and Vishvakarma, S.K. "Device/circuit/ architectural techniques for ultra-low power FPGA design", *Microelectronics and Solid State Electronics*, **2**(A), pp. 1-15 (2013).

18. Windh, S., Ma, X., Halstead, R., Budhkar, P., Luna, Z., Hussaini, O. and Najjar, W. "High-level language tools for reconfigurable computing", *Proceedings of the IEEE,* **103**(3), pp. 390-408 (2015).

19. https://www.xilinx.com/products/intellectual-property.html (23 Nov. 2016).

20. Myint-U, T. and Debnath, L., *Linear Partial Differential Equations for Scientists and Engineers*, Fourth Edition, Birkhäuser Boston (2007).

21. Hoffmann, K.A. and Chiang, S.T., *Computational Fluid Dynamics*, Fourth Edition, Engineering Education System (2000).

22. Jordan, P.P.P.M. "Exact solutions for the unsteady plane couette flow of a dipolar fluid", *Proc. Math. Phys. Eng. Sci.,* **458**(2021), pp. 1245-1272 (2002).