مهندسی مکانیک شریف، (بهار ۱۳۹۶)
دوره‌ی ۳ - ۳۳، شماره‌ی ۱، ص. ۹۳-۹۹

Original Article

پیاده‌سازی سخت‌افزاری حل عددی معادلات دیفرانسیل روی FPGA

فؤاد فرحانی بغلانی * **(دانشیار)**
پژوهشکده‌ی مکانیک، سازمان پژوهش‌های علمی و صنعتی ایران
عباس ابراهیمی چمگردانی (استادیار)
دانشکده‌ی مهندسی هوافضا، دانشگاه صنعتی شریف
ایوب نیکروان شلمانی (دانشجوی دکتری)
پژوهشکده‌ی مکانیک، سازمان پژوهش‌های علمی و صنعتی ایران

حل عددی معادلات دیفرانسیل با استفاده از بسترهای CPU و GPU مبتنی بر پیاده‌سازی نرم‌افزاری است. در سال‌های اخیر، راهکار جدیدی مبتنی بر پیاده‌سازی سخت‌افزاری معادلات با استفاده از بستر FPGA، به‌دلیل افزایش سرعت حل و کاهش توان مصرفی، مورد توجه جدی قرارگرفته است. در این پژوهش با حل چند مسئله‌ی نوعی، شامل سیستم جرم و فنر و معادله‌ی موج، روش پیاده‌سازی سخت‌افزاری برای حل معادلات دیفرانسیل بر روی FPGA، مزایا و چالش‌های این پیاده‌سازی و روش‌های حل آن ارائه شده است. نتایج سرعت پردازش برای حل سیستم تک جرم و فنر نشان می‌دهد که سرعت CPU تقریباً برابر FPGA است ولی برای سیستم ۶ جرم و فنر سرعت FPGA ۸ برابر CPU است. همچنین نتایج سرعت پردازش حل معادله‌ی موج نشان‌دهنده‌ی افزایش ۳/۶ برابری سرعت FPGA نسبت به CPU است. این نتایج نشان‌گر افزایش کارایی FPGA با افزایش تعداد المان‌های محاسباتی است.

واژگان کلیدی: محاسبات قابل بازپیکربندی، افزایش سرعت حل، معادلات دیفرانسیل عادی و پاره‌یی.

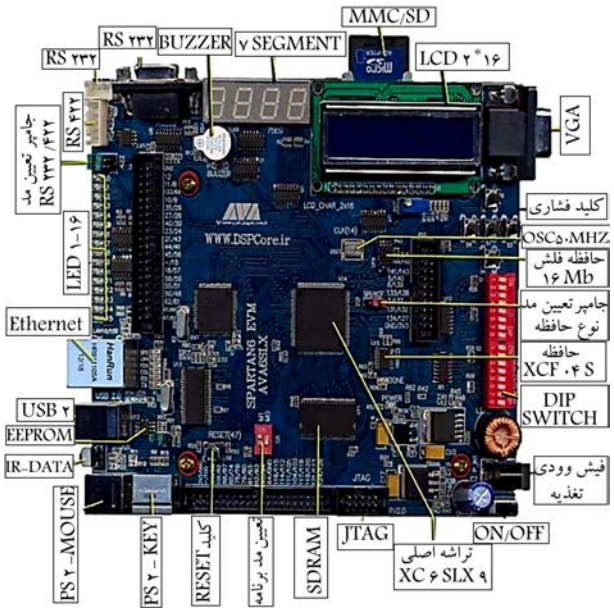
f.farhani@irost.ir
ebrahimi_a@ae.sharif.edu
nikravan@irost.ir

۱. مقدمه

امروزه محاسبات رایانه‌یی ــ شامل مدل‌سازی، شبیه‌سازی و بهینه‌سازی ــ به‌عنوان ابزاری مفید و توان‌مند در مهندسی و علوم کاربرد گسترده‌یی دارد. در مهندسی عمدتاً از روش‌های محاسبات ریاضی برای شبیه‌سازی سیستم‌های پیچیده استفاده می‌شود. افزایش سرعت حل عددی معادلات با مصرف حافظه‌ی مناسب، همواره یکی از چالش‌های اصلی پژوهش‌گران علوم مهندسی بوده است. بدین منظور، تحقیقات زیادی در زمینه‌ی پیاده‌سازی محاسبات با کارایی بالا روی تراشه‌های چندهسته‌یی، سخت‌افزارهای ویژه و سیستم‌های تلفیقی غیرهمگن ــ شامل یک یا چند پردازنده‌ی معمولی و پردازنده‌های ویژه ــ انجام شده است. ازجمله‌ی این پردازنده‌ها می‌توان به تراشه‌های GPU اشاره کرد، که در کنار پردازنده‌های چندهسته‌یی به‌خوبی کارایی خود را در حل مسائل عددی نشان داده‌اند.[۱] FPGA [۱] یکی دیگر از تراشه‌هایی است که می‌تواند با فراهم آوردن بستر سخت‌افزاری قابل تغییر، راهکار جدیدی با عنوان «محاسبات قابل بازپیکربندی[۲]» بر پایه‌ی پیاده‌سازی سخت‌افزاری معادلات ارائه کند.[۲] پیاده‌سازی سخت‌افزاری معادلات با استفاده از بستر FPGA، به دلیل افزایش سرعت حل و کاهش توان مصرفی، به‌عنوان راهکاری جدید مورد توجه جدی پژوهش‌گران قرار گرفته است.

ازجمله پژوهش‌های انجام شده در این زمینه می‌توان به تحقیقات به عمل آمده در آژانس تحقیقات فضایی ژاپن اشاره کرد که با هدف پیاده‌سازی سخت‌افزاری حل عددی مسائل دینامیک سیالات بر روی FPGA صورت گرفت و در آن بخش‌هایی از کد FASTAR پیاده‌سازی سخت‌افزاری شد، و درنتیجه سرعت محاسبات مربوط به عبارت جابه‌جایی این کد، که در حدود ۳۰٪ کل زمان محاسبات است، به شش برابر افزایش یافت.[۳] در پژوهشی دیگر، محققین با پیاده‌سازی بخش‌های انتگرال زمانی و مکانی رمزینه‌ی موجود CFD برای بررسی جریان دائم حول یک ایرفویل به زبان C++ بر روی FPGA، سرعت محاسبات را در مقایسه با حل روی CPU به میزان ۱۶ برابر افزایش دادند.[۴] در تحقیقات مختلف صورت‌گرفته برای یافتن جنس لایه‌های زمین، بستر FPGA برای حل معادله‌ی موج مورد استفاده قرار گرفته است.[۵و۶] همچنین بسترهای سخت‌افزاری FPGA برای پیاده‌سازی روش‌های غیرکلاسیک از قبیل شبکه‌های عصبی[۷] یا اتومتای سلولی[۸] کاربرد داشته است. در قابلیت‌های ارائه شده برای بستر سخت‌افزاری FPGA در حل معادلات ساده،[۹] معادلات جبری

* نویسنده مسئول
تاریخ: دریافت ۱۳۹۴/۱۲/۱۵، اصلاحیه ۱۳۹۵/۴/۱۲، پذیرش ۱۳۹۵/۴/۲۹.



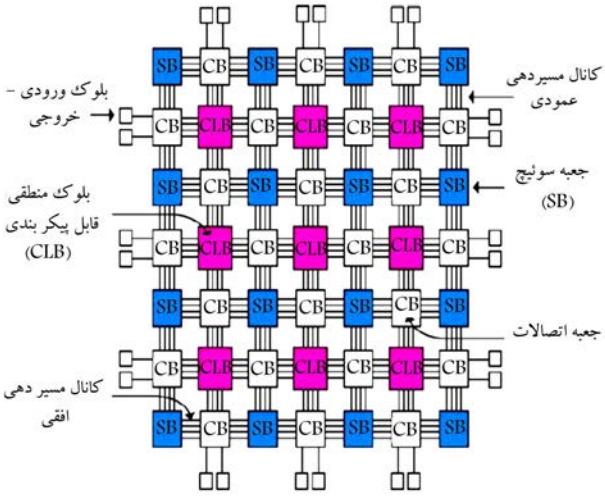
شکل ۱. بورد **FPGA** مورد استفاده.

همچون حل دستگاه دو معادله و دو مجهول بر بستر FPGA پیاده‌سازی شده است.

در این پژوهش با حل چند مسئله‌ی نوعی، شامل سیستم جرم و فنر و معادله‌ی موج، روش پیاده‌سازی سخت‌افزاری برای حل معادلات دیفرانسیل روی FPGA، مزایا و چالش‌های این پیاده‌سازی ارائه شده است. برای آزمایش و پیاده‌سازی FPGA از نوع Spartan با حدود ۹۰۰۰ بلوک قابل برنامه‌ریزی روی بورد ساخت داخل استفاده شده است (شکل ۱).

۲. بستر سخت‌افزاری FPGA

بستر سخت‌افزاری FPGA نسل جدید مدارهای مجتمع دیجیتال قابل برنامه‌ریزی‌اند که سرعت اجرای توابع منطقی در آن‌ها بسیار بالا و در حد نانوثانیه است. یک بستر سخت‌افزاری FPGA (شکل ۲) تراشه‌یی است متشکل از تعداد زیادی بلوک‌های منطقی، خطوط ارتباطی (برای ارتباط بین بلوک‌های منطقی) و پایه‌های ورودی/خروجی که به‌صورت آرایه‌یی درکنار یکدیگر قرارگرفته‌اند. تراشه‌ی FPGA، برخلاف تراشه‌های CPU و GPU، که در آنها سخت‌افزار از قبل طراحی و ساخته شده، همانند یک بوم نقاشی سفید است که به طراح دیجیتال امکان طراحی معماری سخت‌افزاری با هر حجم و پیچیدگی را می‌دهد. در این بستر سخت‌افزاری، طراح می‌تواند با استفاده از زبان‌های توصیف سخت‌افزاری، نظیر VHDL و VERILOG، معماری مورد نظر را روی این بستر خام پیاده‌سازی کند. این معماری ویژه امکان پردازش حل مسائل مختلف از جمله معادلات دیفرانسیل را به‌صورت سخت‌افزاری فراهم می‌کند؛ این کار به‌نوبه‌ی خود باعث کاهش توان مصرفی و افزایش سرعت محاسبات می‌شود. به‌عنوان نمونه، در شبیه‌سازی سیستم‌های زیست‌شناسی با استفاده از ابررایانه‌ی Novo-G شامل ۱۹۲ تراشه‌ی FPGA، افزایش سرعت محاسبه شده در حدود ۵۵۰ هزار برابر یک CPU ۲/۴ گیگاهرتز و توان مصرفی ۸ کیلووات در مقایسه با توان مصرفی حدود ۷ مگاوات برای پردازنده‌های موازی به دست آمده است.[۱۰] مصرف توان کم و سرعت محاسبات بالا باعث شده تا FPGA در ساخت ابررایانه‌های قابل حمل مورد استفاده قرار گیرد.[۱۱و۱۲] همچنین شرکت



شکل ۲. ساختار داخلی **FPGA**.

اینتل اخیراً (اواخر سال ۲۰۱۵) یکی از شرکت‌های بزرگ تولیدکننده‌ی FPGA به‌نام ALTERA را خریداری کرده که به نظر می‌رسد کم‌کم شاهد تولید سیستم‌های جدید CPU-FPGA خواهیم بود که در آن یک هسته‌ی FPGA درکنار CPU بر روی یک تراشه ارائه خواهد شد. همه‌ی این روندها نشان می‌دهد که در آینده سیستم‌های محاسباتی FPGA به‌عنوان یک کمک‌پردازنده نقش بسیار مهمی در فرایند محاسبات خواهد داشت.

۱.۲. قابلیت بستر سخت‌افزاری FPGA برای حل عددی معادلات دیفرانسیل

حل سخت‌افزاری معادلات دیفرانسیل روی بستر سخت‌افزاری FPGA معمولاً در مسائلی خود را بیشتر نمایان می‌سازد که از دو ویژگی برخوردار باشند:

۱. مقدار بسیار زیادی از زمان محاسبه روی بخش کوچکی ازکد انجام شود و این کار چندین بار روی داده‌های بسیار زیاد تکرار شود؛

۲. زمان انتقال اطلاعات نسبت به کل زمان محاسبات کوچک باشد.

معمولاً حل عددی مسائل مهندسی بر پایه‌ی انجام عملیات یکسان و تکراری روی داده انجام می‌شود. به‌عنوان مثال حل مسئله‌ی عددی معادلات جریان سیال، برانتقال شار بین حجم‌های مختلف که بیانگر مرزهای فیزیکی‌اند، پایه‌ریزی می‌شود. محاسبه‌ی عددی شار برای هر حجم مجزا نیازمند صدها عملیات اعشاری است که چندین میلیون بار تکرار می‌شوند (مطابقت با ویژگی اول). همچنین در این‌گونه مسائل انتقال اطلاعات بین داده‌های محلی خلاصه شده و این یعنی محاسبات اجرا شده در محل‌های مختلف فقط وابسته به انتقال اطلاعات از همسایه‌های آن محل است. این خاصیت برای پیاده‌سازی سیستم‌های موازی بسیار جذاب است زیرا کل دامنه‌ی تحلیل را می‌توان به بخش‌ها و زیرمجموعه‌های کوچک‌تر تقسیم کرد و در نهایت برای هر زیرمجموعه یک پردازش‌گر در نظر گرفت. ضرورت ارتباط بین این پردازش‌گرها، وابستگی حل این زیرمجموعه‌ها به داده‌های بقیه‌ی المان‌ها و گره‌هاست. اما خوشبختانه این ارتباط فقط در سطح محلی آن زیرمجموعه‌ها بوده و این دلیلی است برای هزینه کم زمان ارتباط با حافظه نسبت به کل زمان حل مسئله (مطابقت با ویژگی دوم).

در ادامه به تعدادی از خاصیت‌های بستر FPGA، که این سخت‌افزار را برای حل عددی مناسب می‌گرداند، اشاره می‌شود:

۱. دستورات مربوط به جریان انتقالِ داده و کنترل آن را، که به‌صورت دستورات مجزا روی CPU انجام می‌شود، می‌توان به‌صورت پایپلاین[۳] روی بسترهای سخت‌افزاری FPGA پیاده‌سازی کرد و درنهایت با تلفیق FPGA با یک رابط پهنای باند بالا با حافظه‌ی خارجی، امکان دست‌یابی به کارایی‌ها در حد چندین گیگافلاپس و حتی ترافلاپس فراهم می‌شود.

۲. روش‌های عددی که برای حل معادلات دیفرانسیل مورد استفاده قرار می‌گیرند فقط ضرب‌کننده‌ها، جمع‌کننده‌ها، تفریق‌کننده‌ها و تقسیم‌کننده‌ها و چند عملگر ساده‌ی دیگر را شامل می‌شود (حتی در حالت سیستم‌های پیچیده).

۳. برخلاف بسترهای CPU که حافظه‌ی Cash آنها ثابت است، در بسترهای سخت‌افزاری FPGA می‌توان با توجه به الگوریتم موجود، حافظه‌ی مورد نیاز را تولید کرد که این موضوع به کاربران اجازه می‌دهد تا با توجه به پایپلاین طراحی شده در داخل FPGA، بهترین اندازه‌ی این حافظه‌های واسط را برای انجام مداوم عملیات محاسباتی انتخاب کنند.

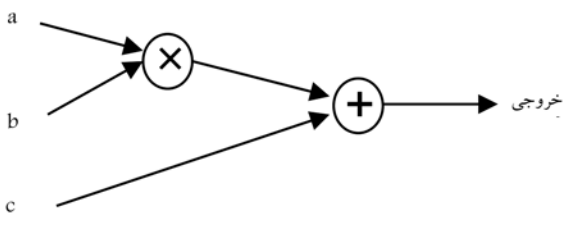
۴. پهنای باند حافظه و مقدار آن را می‌توان با توجه به نیاز مسئله تعیین و پیاده‌سازی کرد. مثلاً اگر پیاده‌سازی بهینه لازم باشد می‌توان مداری با آن میزان از حافظه تولید کرد.

۵. معمولاً کدهایی که در حلقه‌های داخلی یک روش عددی وجود دارد شبیه به هم هستند و اگر یک FPGA با آن سیستم پیکربندی شد، می‌توان از آن برای حل مسائل دیگر فراتر از مسئله اولیه استفاده کرد.

۶. معمولاً گسسته‌سازی مکانی میدان حل در مسائل سازه‌یی و سیالاتی با شبکه‌ی محاسباتی بی‌سازمان[۴] انجام می‌شود. بنابراین برای دسترسی به داده‌های موجود در حافظه‌ی اصلی دستگاه باید یک ساختار مناسب اطلاعاتی تولید کرد که داده‌ها درکم‌ترین زمان در دسترس قرار گیرد. اما در FPGA می‌توان گره‌ها و المان‌های همسایه را کنار یکدیگر قرار داد تا انتقال اطلاعات به‌صورت محلی انجام گیرد.

۲.۲. ناپایداری یک روش عددی پایدار به‌دلیل خطای اعوجاج

«گلیچ[۵]» یکی از چالش‌های محاسبات با قابلیت باز پیکربندی به‌شمار می‌رود. گاهی به‌دلیل بروز گلیچ، خروجی یک عملگر محاسباتی موقتاً با خروجی مورد انتظار متفاوت می‌شود. گلیچ زمانی رخ می‌دهد که داده‌های ورودی به یک عملگر از مسیرهای متفاوت عبور کند، به‌طوری که بخشی از داده‌ها در زمان طولانی‌تری به ورودی آن عملگر برسد. مثلاً در شکل ۳ خروجی مورد انتظار $a \times b + c$ است، حال آن که در چند نانوثانیه‌ی اول به‌دلیل عدم وجود خروجی در عملگر ضرب، خروجی سیستم برابر c است و پس از رسیدن نتیجه‌ی $a \times b$، خروجی برابر $a \times b + c$ خواهد شد.



شکل ۳. معادل سخت‌افزاری $a \times b + c$.

در الگوریتم‌های عددی به‌دلیل ارتباط گره‌ها با هم، داده‌ها درکل سیستم انتقال می‌یابند و درنتیجه درصورت بروز هرگونه تأخیر زمانی، حتی در حدود چند نانوثانیه‌ی فوق‌الذکر، خروجی گره‌ها نادرست شده که منجر به عدم پایداری کل حل خواهد شد.

برای حل این مشکل می‌توان از رجیسترها یا المان‌های تأخیری بین هریک از گره‌ها استفاده کرد. رجیسترها، المان‌های حافظه‌یی هستند که با کلاک سیستم هماهنگ است و تا زمانی که کلاک را در جهت مناسب دریافت نکنند همان مقدار قبلی را در خروجی خود نشان خواهند داد و به محض دریافت آن، مقدار جدید را در خروجی نمایش می‌دهند. کاربر باید کلاک را طوری تنظیم کند که مقدار نادرست بر روی خروجی قرار نگیرد. به شکستن کل فرایند محاسبات به نقاط کوچک‌تر و قرار دادن ریجستر بین آنها پایپلاین می‌گویند.

با این روش، محاسبات به اجزاء کوچک‌تری تجزیه شده که حل هم‌زمان آنها را امکان‌پذیر می‌کند. با وجود پایداری روش حل، عدم استفاده از این رجیسترها می‌تواند منجر به ناپایداری حل شود. به‌عبارت دیگر، حتی درصورت پایداری معادله‌ی اصلی و معادله‌ی جبری از نظر ریاضی، درصوت عدم تنظیم صحیح معماری سخت‌افزاری، ممکن است نتایج حل ناپایدار شود. در بخش‌های بعدی اثر بروز این پدیده در حل معادله‌ی موج و نقش آن در ناپایداری جواب بررسی شده است.

۳. حل معادلات دیفرانسیل معمولی و پاره‌یی بر FPGA

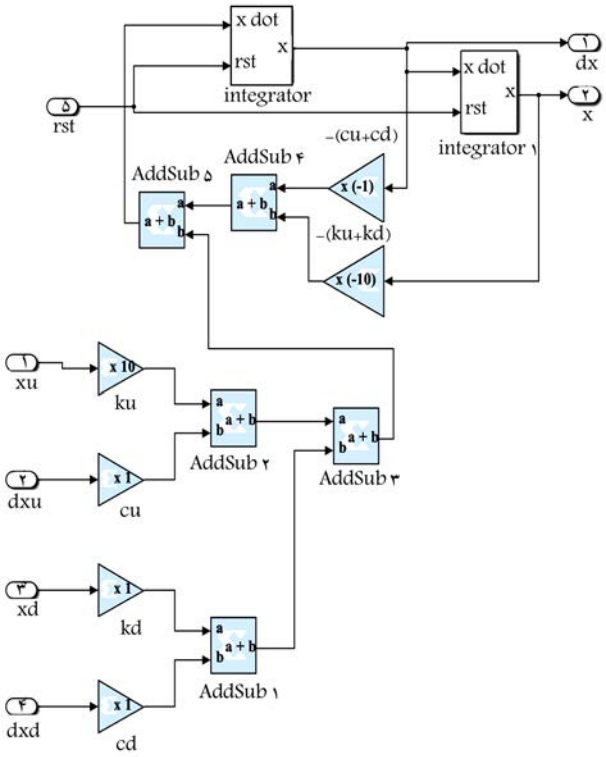
در این بخش، حل سخت‌افزاری معادلات دیفرانسیل حاکم بر سیستم جرم و فنر و معادله‌ی مرتبه‌ی اول موج ارائه شده است. در ادامه، نتایج پیاده‌سازی سخت‌افزاری با نتایج پیاده‌سازی نرم‌افزاری با استفاده از CPU مقایسه شده است. برای این منظور، از CPU اینتل با بسامد ۲٫۴ گیگاهرتز، که به‌لحاظ تکنولوژی ساخت هم‌رده‌ی FPGA به‌کارگرفته شده در این کار تحقیقاتی است. همچنین، در این پژوهش به جای استفاده از زبان‌های توصیف سخت‌افزاری نظیر VHDL و VERILOG، به‌دلیل سادگی کاربری ابزار مولد سیستم[۶] در نرم‌افزار MATLAB به کارگرفته شده است. روند کار چنان است که ابتدا سیستم در مولد سیستم مدل‌سازی شده و به‌کمک آن به VHDL تبدیل می‌شود و سپس به نرم‌افزار Xilinx ISE وارد شده و هسته‌ی Chiposcope به آن اضافه شده و پس از کدگذاری مناسب، درون سخت‌افزار اجرا می‌شود. نتایج نیز با استفاده از Chiposcope Pro از درون سخت‌افزار استخراج می‌شود.

۱.۳. حل معادله‌ی دیفرانسیل معمولی: معادله‌ی جرم و فنر

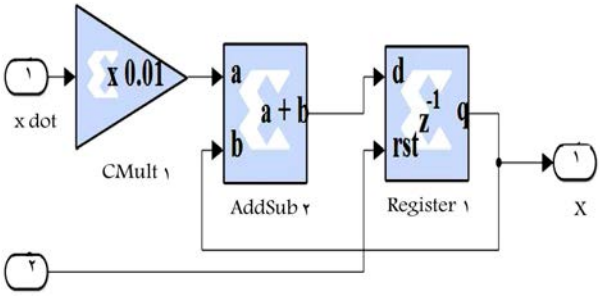
ابتدا به‌منظور بررسی روش پیاده‌سازی سخت‌افزاری حل معادلات دیفرانسیل معمولی بر FPGA، یک مسئله‌ی جرم و فنر و دمپر مورد مطالعه قرارگرفته است. معادله‌ی دیفرانسیل جرم و فنر از رابطه‌ی ۱ به دست می‌آید که در آن m جرم فنر، c ضریب میرایی و k ضریب کشسان فنر است.

$$m\ddot{x} + c\dot{x} + kx = f(t) \qquad (۱)$$

در این مسئله، شرط اولیه برای مکان جرم، در فاصله‌ی ۵ متری نقطه‌ی تعادل تعریف شده است. همچنین ارتعاش جرم بدون سرعت اولیه و به‌صورت آزاد فرض شده



شکل ۴. بلوک نمودار مسئله سیستم یک جرم و فنر.

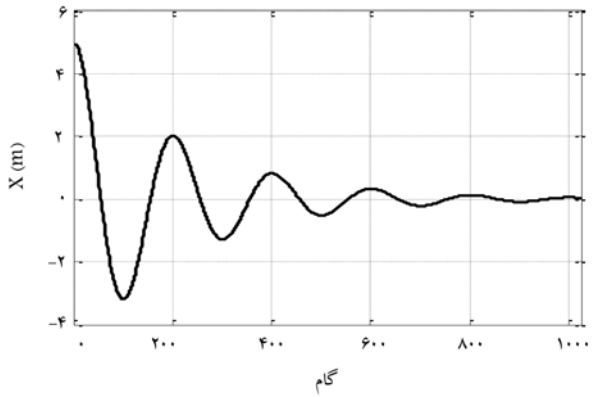


شکل ۵. بلوک نمودار انتگرال‌گیر.

است. مشخصات جرمی این مسئله به‌صورت رابطه‌ی ۲ ارائه شده است.

$$\begin{cases} m = ۱\ \mathrm{kg} \\ k = ۱۰\ \mathrm{N/m} \\ c = ۱\ \mathrm{Ns/m} \end{cases} \qquad (۲)$$

برای حل این مسئله از روش تکراری رونگ ـ کوتا استفاده شده است. در شکل ۴ بلوک نمودار معادل پیاده‌سازی سخت‌افزاری این مسئله نشان داده شده است. این بلوک نمودار شامل دو انتگرال‌گیر[۷] است. ورودی انتگرال‌گیر اول، شتاب و خروجی آن سرعت بوده و به ورودی انتگرال‌گیر دوم متصل می‌شود، به‌طوری که خروجی این انتگرال‌گیر مکان جرم را نشان می‌دهد. چنان که در شکل ۵ مشاهده می‌شود برای پیاده‌سازی سخت‌افزاری هریک از انتگرال‌گیرها از یک رجیستر و یک ضرب‌کننده و یک جمع‌کننده استفاده شده است که در آن گام زمانی انتگرال‌گیری برابر ۰٫۰۱ ثانیه انتخاب شده است. این مقدار با توجه به تجربه‌ی عددی و استقلال حل از گام زمانی انتخاب شده است. در این مسئله



شکل ۶. نتایج خروجی برای مسئله سیستم یک جرم و فنر بر روی FPGA و CPU.

جدول ۱. محاسبات لازم برای حل سیستم تک جرم و فنر.

نوع تراشه	بسامد (MHz)	تعداد کلاک	زمان (μs)
CPU	۲۴۰۰	۶۵۵۳۶	۲۷٫۳
FPGA	۵۰	۱۰۲۴	۲۰٫۵

تعداد گام‌های حل ۱۰۲۴ است که در هر گام ۸ عدد عملیات اعشاری انجام می‌شود.

نتایج حاصل از پیاده‌سازی سیستم جرم و فنر بر FPGA و CPU در شکل ۶ نشان داده شده است. مطابق انتظار نوسان جرم از فاصله‌ی ۵ متری نقطه‌ی تعادل شروع و پس از گذشت حدوداً ۱۰ ثانیه (۱۰۰۰ گام زمانی)، این نوسان میرا می‌شود.

چنان که مشاهده می‌شود این نوع معادله‌سازی بسیار شبیه رایانه‌های آنالوگ دهه ۵۰ و ۶۰ است که در آنها یک دستگاه بزرگ به‌همراه سیم‌های زیاد برای معادل‌سازی الکتریکی معادلات استفاده می‌شد، اما در اینجا دو فرق اساسی وجود دارد، اول این که این سیستم‌ها دیجیتال‌اند و بنابراین محدودیت‌های آنالوگ را ندارند، و دوم این که کل آن سیستم حجیم، داخل یک تراشه‌ی چند سانتی‌متری با میلیون‌ها المان محاسباتی قابل پیاده‌سازی است.

مقایسه‌ی نتایج پیاده‌سازی سخت‌افزاری روی FPGA با نتایج پیاده‌سازی نرم‌افزاری روی CPU با استفاده از مفهوم کلاک[۸] (دوره زمانی کاری) قابل ارائه است. در مورد اول، FPGA قادر است کل عملیات یک گام زمانی حل معادله‌ی جرم و فنر، شامل ۸ عملیات اعشاری را در یک کلاک انجام دهد در حالی که CPU باید هر عملیات اعشاری شامل فراخوانی از حافظه و ذخیره‌سازی در آن را در هشت کلاک و مجموعاً کل عملیات را در ۶۴ کلاک انجام دهد.

بررسی جدول ۱ نشان می‌دهد که مدت زمان مورد نیاز برای حل مسئله‌ی جرم و فنر بر بستر FPGA حدود ۲۰٫۵ میکروثانیه و برای CPU در بهترین حالت (بدون در نظر گرفتن تأخیرهای حافظه) این زمان برابر ۲۷٫۳ میکروثانیه است. در این حالت تقریباً سرعت هر دو سیستم تقریباً یکسان است و در نهایت این امر نشان‌دهنده‌ی سرعت ۱٫۳ برابری FPGA نسبت به CPU است.

با توجه به قابلیت FPGA مورد استفاده در این پژوهش، امکان پیاده‌سازی کمینه یک سیستم متشکل از ۶ جرم و فنر وجود دارد. بنابراین، فرایند فوق برای پیاده‌سازی این سیستم شامل ۶ جرم و فنر تکرار شد که نتایج آن در جدول ۲ ارائه شده است. نکته‌ی شاخص در این پیاده‌سازی سخت‌افزاری آن

است که به‌دلیل امکان‌پذیر بودن پیاده‌سازی موازی عبارت‌های معادلات بر روی FPGA، می‌توان همه‌ی این ۶ جرم و فنر را هم‌زمان تحلیل کرد در حالی که زمان مورد نیاز برای این ۶ عدد کماکان همان ۲۰ میکروثانیه باقی می‌ماند. در این حالت به‌علت استفاده از رجیسترهای میانی تأخیر زمانی اتفاق می‌افتد که نسبت به کل زمان حل قابل اغماض است. به‌دلیل اجرای هم‌زمان (جدول ۲) سرعت FPGA برای حل کامل ۶ عدد جرم و فنر حدود ۸ برابر CPU شده است.

۲.۳. معادله‌ی دیفرانسیل پاره‌یی

معادله‌ی مرتبه‌ی اول موج از نوع هذلولی است و رابطه‌ی یک‌بعدی آن چنین است:

$$\frac{\partial u(x,t)}{\partial t} + a\frac{\partial u(x,t)}{\partial x} = ۰ \qquad (۳)$$

که در آن u دامنه‌ی موج و a سرعت موج و عدد مثبتی است که نشان می‌دهد اطلاعات با سرعت a و به سمت راست حرکت می‌کند. بنابراین اگر شرایط اولیه‌ی موج را داشته باشیم، این شکل موج در زمان‌های مختلف به سمت راست حرکت خواهد کرد.

برای تضمین پایداری رابطه‌ی ۴ لازم است عدد کورانت، $c = \frac{a\Delta t}{\Delta x}$، مساوی یا کم‌تر از ۱ باشد. در حل معادله‌ی موج انجام شده، عدد کورانت برابر ۰٫۲ و سرعت موج برابر ۱ متر بر ثانیه در نظر گرفته شده است. همچنین در این بررسی، از ۶ گره استفاده شده و شرط اولیه‌ی دامنه‌ی موج u برای این گره‌ها به‌صورتی است که همه‌ی گره‌ها به جز گره ۲ در نقطه‌ی صفر بوده و گره ۲ به اندازه‌ی ۱ واحد در جهت مثبت قرار دارد (شکل ۷). شرط مرزی در این گره‌ها چنان است که گره شماره ۱ در مکان صفر نگه داشته شده و بقیه‌ی گره‌ها آزادند.

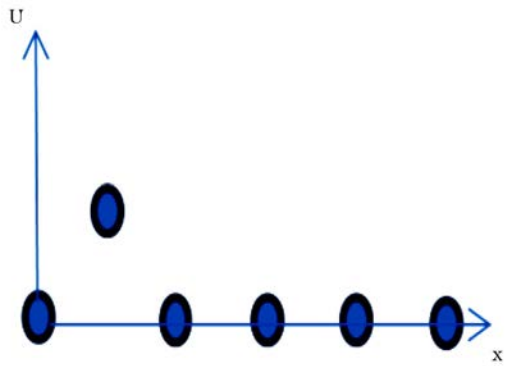
با استفاده از روش تفاضل محدود بالادست مرتبه اول زمانی و پس‌روی مکانی، معادله‌ی جبری گسسته‌ی رابطه‌ی ۳ چنین به دست می‌آید:

$$\frac{u_i^{n+۱} - u_i^n}{\Delta t} = -a\frac{u_i^n - u_{i-۱}^n}{\Delta x} \qquad (۴)$$

برای پیاده‌سازی این سیستم، هریک از ۶ گره را، که رابطه‌ی گسسته شده‌ی ۴ در

جدول ۲. محاسبات لازم برای حل سیستم ۶ جرم و فنر.

نوع تراشه	بسامد (MHz)	تعداد کلاک	زمان (μs)
CPU	۲۴۰۰	۳۹۳۲۱۶	۱۶۳٫۸
FPGA	۵۰	۱۰۲۴	۲۰٫۵



شکل ۷. موقعیت گره‌ها در زمان صفر.

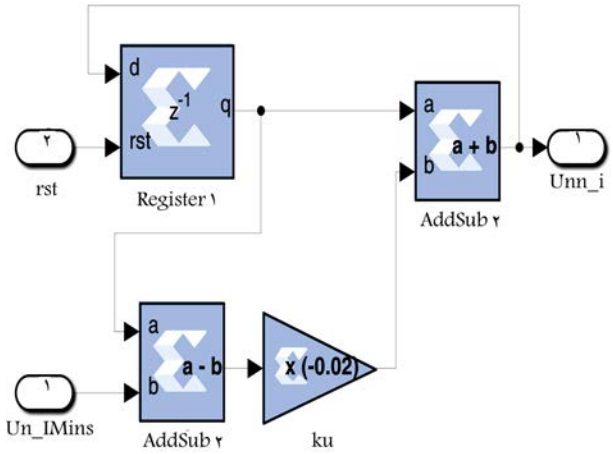
آنها حاکم است، می‌توان با یک ضرب‌کننده و دو جمع‌کننده و یک رجیستر مطابق شکل ۸ پیاده‌سازی کرد. با تکرار فرایند شکل ۸ برای هر گره، مدل سخت‌افزاری رابطه‌ی گسسته شده‌ی ۴ روی FPGA مطابق شکل ۹ خواهد بود، که در آن مستطیل‌های بزرگ نمایان‌گر گره‌ها هستند.

در بستر FPGA، با تبادل متقابل و هم‌زمان داده‌ها بین گره‌های همسایه، معادله‌ی ۴ حل و دامنه‌ی موج برای هرگره در زمان‌های مختلف به دست می‌آید. با حل این پیکربندی، منحنی حرکت گره‌ها مطابق شکل ۱۰ به دست می‌آید. در این شکل، منحنی‌های داخل نمودارها بیان‌گر تغییرات جابه‌جایی هرگره نسبت به زمان است. وجود رجیسترها بین هرگره در شکل ۹ به دلیل پایداری جواب‌هاست، در حالی که این رجیسترها عملاً هیچ گونه عملگر ریاضی نیستند اما عدم وجودشان باعث ناپایداری معادله‌ی گسسته شده می‌شود. برای مثال اگر از رجیسترها استفاده نشود منحنی حرکت گره شماره ۵ مطابق شکل ۱۱ کاملاً ناپایدار خواهد شد. لازم به ذکر است این مشکل حتی برای مسائل پایا، همچون معادله‌ی لاپلاس، نیز اتفاق می‌افتد.

مطابق رفتار موج، کاهش دامنه‌ی گره شماره ۲ از $u = ۱$ شروع و پس از مدتی دامنه به صفر می‌رسد. در این مدت، دامنه‌ی گره‌های دیگر با فاصله‌ی زمانی مشخص که متناسب با سرعت a است، ابتدا افزایش و سپس کاهش می‌یابد.

با توجه به گام‌های زمانی تعیین شده در این مسئله (۱۰۲۴ گام)، دفعات انتقال اطلاعات بین گره‌ها برابر ۶×۱۰۲۴ است. لازم به ذکر است که کل زمان فرایند حل، شامل زمان مورد نیاز برای انتقال داده‌ها و زمان انجام محاسبات داخلی گره‌های سخت‌افزاری می‌باشد.

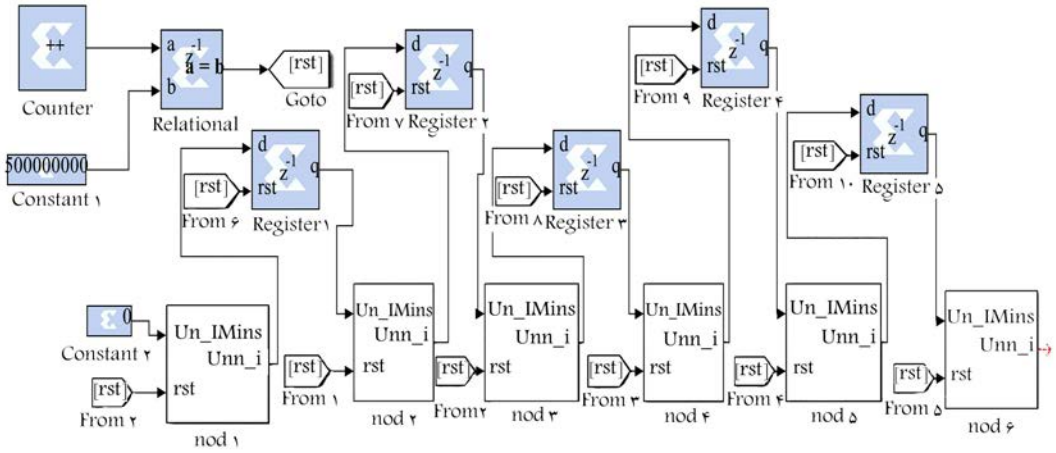
مقایسه‌ی کارایی FPGA و CPU از نظر محاسبات لازم برای حل معادله‌ی موج از نظر تعداد کلاک یا زمان مورد نیاز در جدول ۳ ارائه شده است. نتایج نشان می‌دهد که در این مسئله FPGA سرعت ۳٫۶ برابری نسبت به CPU دارد. این افزایش سرعت FPGA در حالی است که بسامد کاری آن بسیار پایین‌تر از CPU و در حدود یک‌چهلم آن است که به همین میزان می‌تواند باعث کاهش توان مصرفی در محاسبات شود.



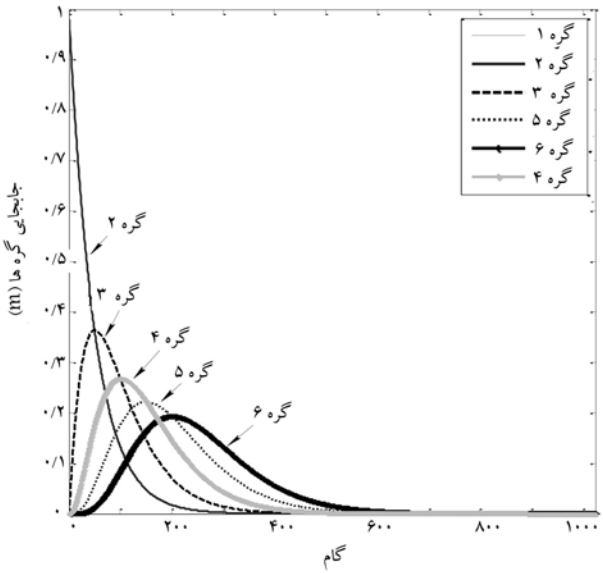
شکل ۸. معادله‌ی نموداری یک گره سخت‌افزاری برای رابطه‌ی گسسته‌ی موج.

جدول ۳. محاسبات لازم برای حل معادله موج با ۶ عدد گره.

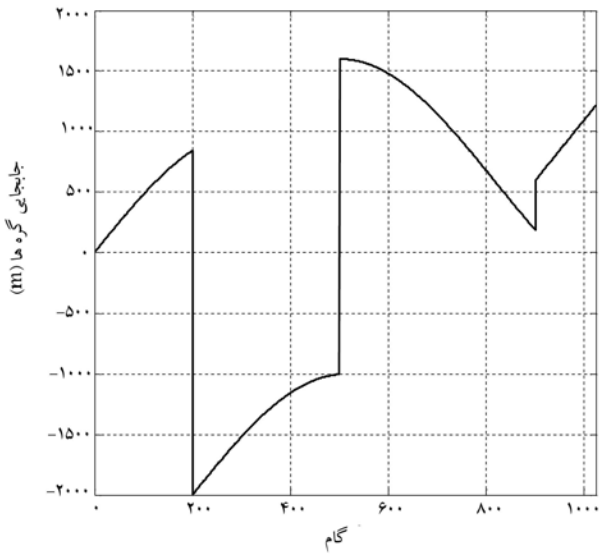
نوع تراشه	بسامد (MHz)	تعداد کلاک	زمان (μs)
CPU	۲۴۰۰	۱۴۷۴۵۶	۶۱٫۴
FPGA	۶۰	۱۰۲۴	۱۷



شکل ۹. بلوک نمودار کل مسئله‌ی موج.



شکل ۱۰. منحنی حرکت گره‌ها.



شکل ۱۱. منحنی حرکت گره ۵ در حالت پیاده‌سازی سخت‌افزاری بدون استفاده از رجیستر.

۴. نتیجه‌گیری

در این نوشتار روشی جدید و در حال توسعه در زمینه‌ی تسریع محاسبات عددی با عنوان محاسبات قابل بازپیکربندی بر پایه‌ی ابزاری جدید به‌نام FPGA معرفی شده است. برای این کار از سخت‌افزارهای موجود در داخل کشور و نرم‌افزارهای در دسترس مانند ISE استفاده شده که به‌صورت پیش‌فرض برای کاربرد محاسبات معادلات دیفرانسیل طراحی نشده است. استفاده از این ابزارها برای حل محاسبات دیفرانسیل برای نخستین بار صورت گرفته که در آن با پیاده‌سازی دسته‌یی از معادلات دیفرانسیل بر بستر تراشه‌ی FPGA از نوع Spartan۶، نحوه‌ی پیاده‌سازی سخت‌افزاری معادلات، حل موازی عبارات و انتقال داده‌ها بین گره‌ها و المان‌های همسایه به‌صورت مستقیم نشان داده شده است.

در این نوشتار روند پیاده‌سازی سخت‌افزاری معادلات و پدیده‌ها و مشکلات جدیدی که با آن روبرو می‌شویم (و در فرایندهای کلاسیک محاسبات دیده نمی‌شوند) و حل آنها ارائه شد که از آن جمله می‌توان به پدیده‌ی خطای ناشی از اعوجاج اشاره کرد، که منجر به ناپایداری حل معادلات پایدار ریاضی می‌شود.

عملکرد روش پیشنهادی در تسریع محاسبات عددی با حل مسائل جرم و فنر و معادله موج نشان داده شده است. نتایجی که مستقیماً از حل معادلات بر FPGA به دست آمد نشان داد که پیاده‌سازی سخت‌افزاری با کاهش تعداد چرخه‌ی کاری برای انجام محاسبات باعث افزایش سرعت حل معادلات خواهد شد، به‌طوری که برای دو مسئله‌ی حل شده در این نوشتار، نتایج سرعت پردازش نشان می‌دهد که:

- سرعت CPU برای حل سیستم تک جرم و فنر تقریباً برابر FPGA بوده ولی برای سیستم ۶ جرم و فنر سرعت FPGA حدوداً ۸ برابر CPU شد که نشان‌دهنده‌ی افزایش کارایی FPGA با افزایش تعداد المان‌های محاسباتی است.
- سرعت پردازش حل معادله‌ی موج نشان‌دهنده‌ی افزایش ۳/۶ برابری سرعت FPGA نسبت به CPU است، در حالی که بسامد کاری FPGA بسیار کم‌تر از CPU است.

پانوشت‌ها

1. field programmable gate array
2. reconfigurable computing
3. Pipeline
4. unstructured grid
5. Glitch
6. system generator
7. integrator
8. Clock

منابع (References)

1. Owens, J., Houston, M., Luebke, D., Green, S., Stone, J. and Phillips, J. "GPU computing", *Proceedings of the IEEE*, **96**(5), pp. 879-899 (May 2008).
2. Sundararajan, P., *High Performance Computing Using FPGAs*, Tech. Rep., Available Online: www. xilinx.com/support/documentation/white papers/ wp 375 HPC Using FPGAs.pdf (2010).
3. AbuTalip, M.S., Akamine, T., Hatto, M. and Amano, H. "Adaptive flux calculation scheme in advection term computation using partial reconfiguration", *International Journal of Networking and Computing*, **3**(2), pp. 289-306 (2013).
4. Sanchez-Roman, D. Sutter, G. Lopez-Buedo, S. and Gonzalez, I. "High-level languages and floating-point arithmetic for FPGA based CFD simulations", *IEEE Design & Test of Computers*, **28**(4), pp. 28-36 (2011).
5. HE, C. "Numerical solutions of differential equations on FPGA-enhanced computers", PhD Thesis, A&M Texas-USA (2007).
6. Hu, J. "Solution of partial differential equations using reconfigurable computing", PhD Thesis, University of Birmingham-England (2010).
7. Nagay, Z. "Implementation of emulated digital CNN-UM on programmable logic device and it's application", PhD Thesis (2010).
8. Murtaza, S. "High performance reconfigurable computing with cellular automata", PhD Thesis, Von Amesterdam-Holand (2010).
9. Singleterry, R.C. and Fithian, W.S. "Field programmable gate array computer in structural analysis: An initial exploration", AIAA/ASME/ASCE/AHS/ ASC Structures, Structural Dynamics, and Materials (2002).
10. George, A., Lam, H. and Stitt, G. "At the forefront of scalable reconfigurable supercomputing", *Computing in Science and Engg*, **13**(1), pp. 82-86 (January 2011).
11. https://www.parallella.org/, Accessed on 20 November (2015).
12. http://zedboard.org/, Accessed on 20 November (2015).