بهبود کارایی تبدیل موجک گسسته دوبعدی با استفاده از تکنیک موازی‌سازی در سطح داده

عبدالبصیر تیباش[۱]، دانشجوی کارشناسی ارشد؛ اسدالله شاه‌بهرامی[۲]، دانشیار

۱- گروه مهندسی کامپیوتر – دانشکده فنی - دانشگاه گیلان - رشت - ایران – BasirTibash@msc.guilan.ac.ir
۲- گروه مهندسی کامپیوتر – دانشکده فنی - دانشگاه گیلان - رشت - ایران – shahbahrami@guilan.ac.ir

چکیده: تبدیل موجک گسسته دوبعدی (2D-DWT) به‌صورت گسترده‌ای در کاربردهای مختلف پردازش داده‌های چندرسانه‌ای ازجمله استانداردهای فشرده‌سازی تصاویر و ویدئو مورداستفاده قرار می‌گیرد. بااین‌وجود، این تبدیل دارای پیچیدگی محاسباتی بالاتری نسبت به تبدیل‌های مرسوم مانند تبدیل گسسته کسینوسی و دیگر توابع موجود در استانداردهای فشرده‌سازی است و بیشترین درصد از زمان اجرا را به خود اختصاص می‌دهد. در این مقاله، برای بهبود کارایی 2D-DWT از مجموع دستورات فنّاوری‌های توسعه برداری پیشرفته AVX/AVX2 و جمع ضرب ترکیبی (FMA) که قابلیت پردازش ۲۵۶ بیت داده با استفاده از معماری یک دستورالعمل و چندین داده (SIMD) که توسط اکثر پردازشگرهای همه‌منظوره (GPP) پشتیبانی می‌گردد، پیشنهادشده است. با استفاده از این فنّاوری‌ها قابلیت پردازش هشت داده ۳۲ بیتی برای اعداد اعشاری و شانزده داده ۱۶ بیتی برای اعداد صحیح شانزده بیتی در ثبات‌های SIMD یک GPP فراهم می‌گردد. بعلاوه نحوی نگاشت تبدیل‌های مختلف موجک به روش پردازش‌های سطری-ستونی که پردازش‌های سطری و ستونی را جداگانه انجام می‌دهد و مبتنی بر خط که هر دو، سطرها و ستون‌های تصویر را در یک حلقه پردازش می‌کند، استفاده‌شده است. نتایج پیاده‌سازی موازی‌سازی تبدیل‌های مختلف بر روی یک پلتفرم GPP نشان داد که کارایی، 2D-DWT به ازای اندازه تصاویر مختلف را می‌توان تا ۲۸/۸ برابر نسبت به پیاده‌سازی سریال بالا برد. همچنین نگاشت مبتنی بر خط که باعث استفاده بهتر از ساختار سلسله مراتبی حافظه می‌گردد، کارایی را نسبت به نگاشت سطری – ستونی بیشتر بهبود می‌دهد.

واژه‌های کلیدی: پردازشگرهای همه‌منظوره، پردازش موازی، تبدیل موجک گسسته دوبعدی، موازی‌سازی سطح داده، یک دستورالعمل چندین داده.

Performance Improvement of 2D Discrete Wavelet Transform using Data-Level Parallelism Technique

A. Tibash[1], MSc. student; A. Shahbahrami[2], Associate Professor

1- Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran, Email: BasirTibash@msc.guilan.ac.ir
2- Department of Computer Engineering, Faculty of Engineering, University of Guilan, Rasht, Iran, Email: Shahbahrami@guilan.ac.ir

Abstract: The two-Dimensional Discrete Wavelet Transform (2D-DWT) is widely used in various applications for multimedia data processing, including image and video compression standards. However, this transform is computational intensive than conventional conversions, such as the discrete cosine transform. In this paper, in order to improve the performance of 2D-DWT, we use Single Instruction, Multiple Data (SIMD) set instructions including Advanced Vector Extensions (AVX), Fused Multiply-Add (FMA), and AVX2 supported by most General-Purpose Processors (GPP). These technologies capable to process 256-bit data located in SIMD registers. The AVX technology can process eight 32-bit floating point numbers, while AVX2 processes sixteen 16-bit fixed-point numbers. In other words, it is possible to exploit 8- and 16-way data-level parallelism. In addition, two different way of parallelism, Row Column Wavelet Transform (RCWT) which processes rows and columns separately and Line-Based Wavelet Transform (LBWT) that processes both rows and columns in a single loop are used. Experimental results of different wavelet transform with different image sizes on a GPP show that the speedups of up to 28.8x yield. Furthermore, LBWT approach improves performance more than RCWT. This is because it uses memory hierarchy structure more efficiently than RCWT approach.

Keywords: Data-Level Parallelism, Discrete Wavelet Transform, General-Purpose Processor, Parallel processing, Single Instruction, Multiple Data.

تاریخ ارسال مقاله: ۱۳۹۶/۱۰/۱۹
تاریخ اصلاح مقاله: ۱۳۹۷/۰۴/۰۲و ۱۳۹۷/۰۶/۱۰
تاریخ پذیرش مقاله: ۱۳۹۷/۱۰/۰۳
نام نویسنده مسئول: اسدالله شاه بهرامی
نشانی نویسنده مسئول: گروه مهندسی کامپیوتر – دانشکده فنی - دانشگاه گیلان، رشت، ایران.

۱- مقدمه

تبدیل موجک گسسته (DWT)[۱] یک ابزار ریاضی است که امکان تجزیه‌وتحلیل زمان-فرکانسی سیگنال را فراهم می‌سازد. امروزه این تبدیل در حوزه پردازش تصویر، مخصوصاً در الگوریتم‌های فشرده‌سازی، استخراج ویژگی‌های یک تصویر، نشانه‌گذاری رقمی و تشخیص الگو به‌صورت گسترده‌ای مورداستفاده قرار می‌گیرد. به‌عنوان‌مثال، در استانداردهای فشرده‌سازی JPEG-2000 و MPEG-4 از DWT استفاده‌شده است. درنتیجه، مزیت تبدیل موجک نسبت به تبدیل‌های مرسوم مانند تبدیل فوریه در حال حاضر به‌خوبی شناخته‌شده است [۳-۱]. بااین‌وجود DWT دوبعدی ازنظر مدت‌زمان موردنیاز برای اجرا، می تواند بسیار ناکارآمد باشد. به‌عنوان‌مثال، نشان داده‌شده که در استاندارد JPEG-2000 بیشترین درصد زمان اجرای پردازنده مربوط به تابع DWT دوبعدی است. همچنین، نشان داده‌شده که DWT به‌طور میانگین ۴۶ درصد از زمان کدگذاری فشرده‌سازی‌های بدون‌اتلاف و ۶۸ درصد از زمان کدگذاری فشرده‌سازی‌های بااتلاف را به خود اختصاص داده و پیچیدگی محاسباتی بالاتری نسبت به تبدیل کسینوسی گسسته (DCT)[۲] دوبعدی دارد [۴، ۵].

معماری‌های مختلفی جهت بهبود کارایی 2D-DWT ارائه‌شده و بر روی سخت‌افزاری مختلفی ازجمله پردازشگرهای همه‌منظوره (GPP)[۳] با بهره‌برداری از توسعه یک دستورالعمل و چندین داده (SIMD)[۴] پیاده‌سازی شده‌اند. بردارسازی‌های انجام‌شده از DWT دوبعدی بر روی GPPها با استفاده از مجموعه دستورالعمل‌های SIMD را می‌توان بر اساس معیارهای مختلفی ازجمله الگوریتم‌های پیاده‌سازی دوبعدی، نوع پیاده‌سازی و مجموعه دستورالعمل‌های SIMD استفاده‌شده به دسته‌های مختلفی تقسیم کرد. ازنظر الگوریتم پیاده‌سازی دوبعدی تبدیل موجک گسسته می‌توان آن‌ها را به دودسته بردارسازی SIMD از الگوریتم تبدیل موجک سطری - ستونی (RCWT)[۵] و الگوریتم موجک مبتنی بر خط (LBWT)[۶] تقسیم کرد. در [۵، ۶] بردارسازی SIMD از تبدیل دابچیز-۴ (Daub-4)[۷] و کوهان-دابچیز-فوو-۵/۳ (CDF-5/3)[۸] با استفاده از این دو الگوریتم ارائه و نتایج حاصل از آن‌ها مقایسه شده‌اند. نتایج حاصل از پیاده‌سازی‌ها نشان می‌دهد اگرچه بردارسازی SIMD از الگوریتم RCWT آسان‌تر از الگوریتم LBWT است، اما بهبود کارایی بردارسازی SIMD با استفاده از الگوریتم LBWT بیشتر است. بر اساس نوع پیاده‌سازی نیز می‌توان آن‌ها را به دودسته پیاده‌سازی صحیح و اعشاری تقسیم کرد. در [۷] نشان داده‌شده که می‌توان ضرایب پالایش اعشاری تبدیل‌های مختلف را به‌صورت صحیح درآورده و محاسبات را به‌صورت صحیح و با استفاده از مجموعه دستورالعمل‌های صحیح انجام داد. آن‌ها نشان داده‌اند که پیاده‌سازی صحیح به ازای خطایی ناچیز می‌تواند کارایی را بهبود بخشد. بااین‌وجود در اکثر پیاده‌سازی‌های انجام‌شده از مجموعه دستورالعمل‌های توسعه‌های چندرسانه‌ای (MMX)[۹] و جریان‌سازی توسعه SIMD (SSE)[۱۰] به‌منظور بردارسازی DWT دوبعدی استفاده‌شده و از مجموعه دستورالعمل‌های توسعه برداری پیشرفته (AVX)[۱۱]، AVX2 و جمع ضرب ترکیبی (FMA)[۱۲] که امکان قابلیت پردازش بردارهای ۲۵۶ بیتی را فراهم می‌کنند بهره نبرده‌اند.

در این مقاله، جهت بهبود کارایی 2D-DWT از مفهوم موازی‌سازی سطح داده (DLP)[۱۳] با استفاده از فنّاوری‌های AVX/AVX2 و جمع ضرب ترکیبی (FMA) در SIMD توسعه‌یافته در GPPs پیشنهادشده است. بدین منظور دو بردارسازی[۱۴] SIMD برای پنج تبدیل پرکاربرد هار (Haar) و Daub-4 با رویکرد پیچش[۱۵] و هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی[۱۶] بر اساس دو الگوریتم پیاده‌سازی RCWT و LBWT ارائه می‌شود. نتایج ارزیابی کارایی حاکی از آن است که بردارسازی از الگوریتم RCWT میزان کارایی تبدیل‌های مختلف را به ازای ابعاد مختلف تصاویر بین ۲/۷ الی ۲۳/۷ برابر بهبود می‌بخشد درحالی‌که میزان این افزایش کارایی برای بردارسازی SIMD از الگوریتم LBWT بیشتر و بین ۳/۳ الی ۲۸/۸ برابر است.

ادامه مقاله به‌صورت زیر ساختاربندی می‌شود. در بخش دوم ابتدا مروری کوتاه از مبانی نظری تبدیل موجک گسسته و SIMD ارائه‌شده و سپس پیشینه تحقیق موردبررسی قرارگرفته است. در بخش سوم بردارسازی تبدیل موجک گسسته بر اساس دو روش پیاده‌سازی RCWT و LBWT ارائه‌شده است. بخش چهارم به ارزیابی نتایج آن‌ها بر روی یک پردازشگر با ریز معماری Haswell پرداخته است. درنهایت بخش آخر به نتیجه‌گیری اختصاص‌یافته است.

۲- مفاهیم اولیه و پیشینه تحقیق

۲-۱- تبدیل موجک گسسته

تبدیل موجک گسسته می‌تواند به‌عنوان یک روش مناسب جهت تجزیه یک سیگنال به اجزای فرکانسی بالا گذر و پایین گذر درک شود. این تبدیل ویژگی‌های بسیار مناسبی را ارائه می‌دهد که ازجمله آن‌ها می توان
به موارد زیر اشاره نمود [۸]:

- به‌اندازه کافی اطلاعات موردنیاز برای تحلیل و بررسی موج اصلی ارائه می‌دهد.
- می‌توان توسط آن سیگنال اصلی را در فرکانس‌های مختلف و با وضوح‌های کاملاً متفاوت بررسی و تحلیل نمود.

رویکردهای مختلفی برای اعمال DWT بر روی سیگنال ازجمله رویکرد پیچش یا کانولوشن و پلکانی ارائه‌شده است. همچنین، به‌منظور پیاده‌سازی دوبعدی این تبدیل دو الگوریتم تبدیل موجک سطری-ستونی و تبدیل موجک مبتنی بر خط ارائه گردیده است. در این بخش به بیان این رویکردها و الگوریتم‌ها پرداخته‌شده است. در زیر بخش‌های ۲-۱-۱ و ۲-۱-۲ به ترتیب دو رویکرد مختلف پیچش و پلکانی برای پیاده‌سازی DWT تشریح شده‌اند. در زیر بخش ۲-۱-۳ الگوریتم‌های پیاده‌سازی DWT دوبعدی با استفاده DWT یک‌بعدی ارائه‌شده است.

۲-۱-۱- رویکرد پیچش

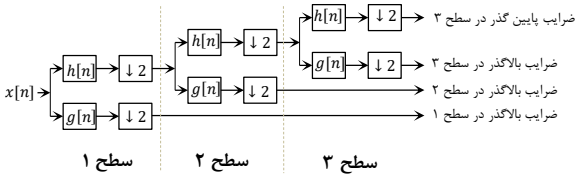
درروش تبدیل موجک از یک تابع پایه به نام موجک مادر و توابع دیگری که از بسط و گسترش موجک مادر تولید می‌شوند، استفاده می‌گردد [۹]. در رویکرد پیچش ضرایب موجک رزولوشن‌های پایین از هر مرحله تبدیل موجک گسسته به‌صورت بازگشتی بر طبق معادلات (۱) و (۲) محاسبه می‌گردد:

$$c_{j-1,k} = \sum_{m} c_{j,m} \cdot h[m-2k] \qquad (۱)$$

$$d_{j-1,k} = \sum_{m} c_{j,m} \cdot g[m-2k] \qquad (۲)$$

که در آن $c_{j,m}$ نشان‌دهنده jمین ضریب پایین‌گذر در سطح تبدیل m و $d_{j,m}$ نشان‌دهنده jمین ضریب بالاگذر در سطح تبدیل m است.

شکل ۱ نمودار بلوکی برای DWT یک‌بعدی با استفاده از رویکرد پیچش را برای سه سطح نشان می‌دهد. در این شکل $h[n]$ و $g[n]$ به ترتیب ضرایب پالایش بالا گذر و پایین گذر، ۲↓ نمونه کاهی با مقیاس ۲ را نشان می‌دهند [۱۰، ۱۱].



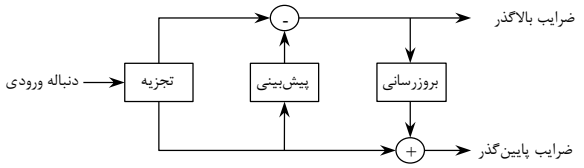
شکل ۱: نمودار بلوکی تحلیل DWT یک‌بعدی با استفاده از رویکرد پیچش برای سه سطح تجزیه [۱۰].

پالایه‌ها و فیلترهای مختلفی برای پیاده‌سازی تبدیل موجک گسسته ازجمله پالایه‌های دابچیز و CDF که دو تا از مهم‌ترین این پالایه‌ها هستند، ارائه گردیده‌اند. همچنین، در استانداردهای فشرده‌سازی JPEG-2000 و MPEG-4 از این پالایه‌ها استفاده‌شده است.

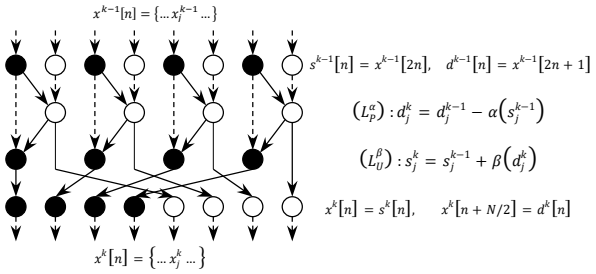
۲-۱-۲- رویکرد پلکانی

در این بخش به‌صورت خلاصه ایده اصلی رویکرد پلکانی ارائه گردیده و سه روش پیاده‌سازی افقی، عمودی و مورب آن بیان‌شده‌اند. رویکرد پلکانی شامل سه مرحله تجزیه، پیش‌بینی و به‌روزرسانی است که نمودار کلی آن در شکل ۲ نشان داده‌شده است [۶].

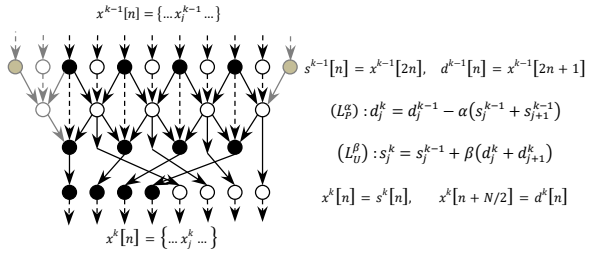
شکل ۳ توضیح مصوری از DWT یک‌بعدی با رویکرد پلکانی بر روی تبدیل‌های هار، CDF-5/3 و CDF-9/7 را نشان می‌دهد. همان‌طور که در این شکل مشاهده می‌گردد رویکرد پلکانی DWT شامل چندین مرحله پلکان (دو مرحله برای هار و CDF-5/3 و چهار مرحله برای CDF-9/7) و یک مرحله نرمال‌سازی اختیاری در پایان است [۱۲].



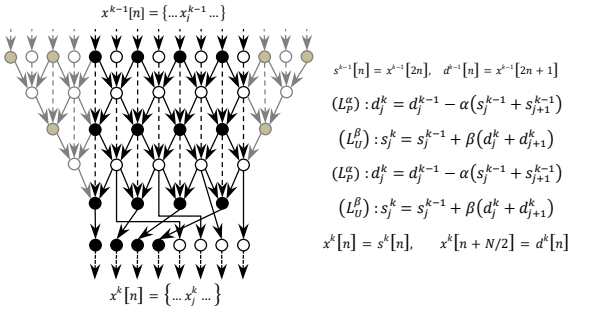
شکل ۲: رویکرد پلکانی برای پیاده‌سازی DWT دوبعدی [۶].



الف) تبدیل هار



ب) تبدیل CDF-5/3



ج) تبدیل CDF-9/7

شکل ۳: رویکرد پلکانی برای پیاده‌سازی سه تبدیل هار، CDF-5/3 و CDF-9/7 [۱۲].

ابتدا سیگنال ورودی به دو جریان از ضرایب تقسیم می‌شوند: آرایه‌های زوج و فرد، به‌طورمعمول سری‌های تقریب و جزئیات نامیده شده و فرکانس‌های پایین و بالای طیف موجک را نشان می‌دهند. به‌عبارت‌دیگر، سیگنال ورودی چندگانه در داخل یک نمایش چندمرحله‌ای به‌صورت زیر تجزیه‌شده است:

$$s^{k-1}[n] = x^{k-1}[2n], \quad d^{k-1}[n] = x^{k-1}[2n+1] \qquad (۳)$$

سپس، قسمت جزئیات با استفاده از مقادیر فعلی هر دو جریان پیش‌بینی (معادله (۴)) و در مرحله بعد قسمت تقریب طبق معادله (۵) به‌صورت زیر به‌روزرسانی می‌گردد ($s^k[n]$ و $d^k[n]$ به ترتیب ضرایب تقریب و جزئیات سطح k از سطح $k-1$ هستند):

$$d^k[n] = d^{k-1}[n] - L_P(s^{k-1}[n]) \qquad (۴)$$

$$s^k[n] = s^{k-1}[n] + L_U(d^k[n]) \qquad (۵)$$

به زبان ساده، L_P و L_U (با پیکان مشکی در شکل ۲ نشان داده‌شده‌اند) که عملگرهای پلکان پیش‌بینی و به‌روزرسانی در معادله (۴) و معادله (۵) هستند، می‌توانند به‌عنوان مجموع وزنی محلی ضرایب پلکان بیان

شوند. رویکرد پلکانی دو موجک هار و CDF-5/3 تنها از یک جفت وزن (α وβ) استفاده می‌کنند درحالی‌که موجک CDF-9/7 از دو جفت وزن ((α و β) و (γ و δ)) که در جدول ۱ لیست شده‌اند استفاده می‌کنند. درنهایت، ضرایب نهایی در مرحله ادغام (معادله (۶) که N نشان‌دهنده اندازه ابعاد تصویر است) به مکان‌های مناسب، با یا بدون نرمال‌سازی، فرستاده‌شده و یک سطح از تبدیل تکمیل می‌گردد. معکوس تبدیل را می‌توان با استفاده از اعمال مراحل بیان‌شده در جهت عکس و وزن علامت مخالف به دست آورد [۱۲].

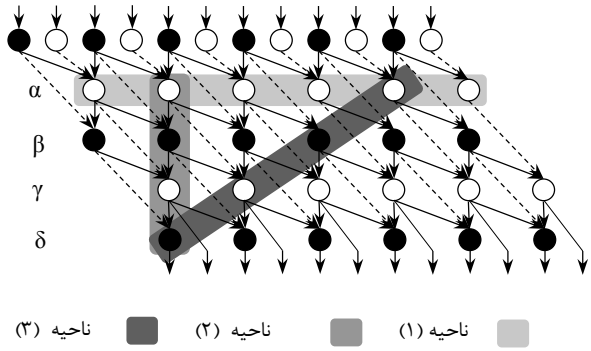
$$x^k[n] = s^k[n], \qquad x^k[n+N/2] = d^k[n] \qquad (۶)$$

جدول ۱: ضرایب پلکان از خانواده‌های موجک هار، CDF-5/3 و CDF-9/7 [۱۲].

	هار	CDF-5/3	CDF-9/7
α	۱/۰	۰/۵	۱/۵۸۶۱۳۴۳۴
β	۰/۵	۰/۲۵	-۰/۰۵۲۹۸۰۱۲
γ	---	---	-۰/۸۸۲۹۱۱۰۸
δ	---	---	۰/۴۴۳۵۰۶۸۵

باید توجه داشت که تبدیل CDF-9/7 بااتلاف است، بدین معنا که دنباله ورودی به تبدیل CDF-9/7 و دنباله حاصل از معکوس آن به دلیل ضرایب موجک غیردقیق نشان داده‌شده در جدول ۱ دقیقاً یکسان نیستند. علاوه بر این، CDF-9/7 به دلیل داشتن چهار مرحله پلکانی دارای هزینه محاسباتی بیشتری در مقایسه با دو موجک هار و CDF-5/3 با دو مرحله پلکانی است. شکل ۳ همچنین نشان می‌دهد که L_p و L_U تبدیل موجک هار می‌تواند به‌صورت محلی و بدون خواندن مقادیر همسایه در طول یک مرز بلاک اعمال شود چراکه هر دو مرحله پیش‌بینی و به‌روزرسانی فقط از مقادیر یک سمت استفاده می‌کنند؛ اما CDF-5/3 و CDF-9/7 نیاز به پیش‌بینی و به‌روزرسانی عنصر فعلی با استفاده از مقادیر همسایه هر دو طرف دارند [۱۲].

به‌منظور پیاده‌سازی رویکرد پلکانی سه روش مبتنی بر بردارسازی افقی، عمودی و مورب وجود دارد. در شکل ۴ این سه روش بردارسازی به ترتیب با ناحیه‌های (۱)، (۲) و (۳) نشان داده‌شده‌اند.



شکل ۴: رویکرد پلکانی با سه روش مبتنی بر بردارسازی افقی ناحیه ۱، عمودی ناحیه ۲ و مورب ناحیه ۳.

رویکرد پلکانی مبتنی بر بردارسازی افقی به چندین عمل خواندن و نوشتن نتایج میانی جهت محاسبه یک جفت ضرایب پلکانیs_n و d_n نیازمند است. با توجه به پیاده‌سازی GPPها به ازای تصاویر بزرگ، این نتایج میانی چندین بار از حافظه نهان واحد پردازشگر مرکزی اخراج خواهند شد. درنتیجه، فقدان‌های حافظه نهان زیادی در چنین محاسباتی رخ خواهد داد [۱۳].

بردارسازی عمودی روشی دیگر جهت بردارسازی گراف جریان داده رویکرد پلکانی است که روش تک حلقه نیز نامیده شده است. گراف داده بردارسازی عمودی برای تبدیل CDF-9/7 با رویکرد پلکانی در شکل ۴ با ناحیه (۲) نشان داده‌شده است. در این روش، محاسبات تبدیل با رویکرد پلکانی به‌جای استفاده از حلقه‌های متعدد بر روی همه ضرایب، در داخل یک حلقه انجام می‌شود؛ بنابراین، در هر تکرار حلقه یک جفت از ضرایب پلکان s_n و d_n محاسبه می‌گردند. محاسبات در درون ناحیه (۲) نشان داده‌شده در این شکل، به دلیل وجود وابستگی بین داده‌ها، نمی‌تواند به‌طوری موازی پیاده‌سازی شود. بااین‌وجود، این روش به دلیل اینکه ضرایب فقط یک‌بار خوانده و نوشته می‌شوند بسیار سودمند است. چون این کار مانع از رخ دادن فقدان‌های حافظه نهان غیرضروری می شود. در بردارسازی‌های DWT یک‌بعدی با استفاده SIMD، می‌توان چندین ناحیه از بردارسازی عمودی را به‌صورت موازی هم‌زمان پردازش کرد [۱۳].

در [۱۴]، بردارسازی مورب از رویکرد پلکانی ارائه‌شده است. گراف جریان داده بردارسازی مورب برای تبدیل CDF-9/7 با رویکرد پلکانی در شکل ۴ با ناحیه (۳) داده‌شده است. در این روش نیز مانند بردارسازی عمودی، محاسبات تبدیل پلکانی در داخل یک حلقه انجام‌شده و در هر تکرار حلقه یک جفت از ضرایب پلکانی s_n و d_n محاسبه گردند. بااین تفاوت که در بردارسازی مورب برخلاف بردارسازی عمودی، وابستگی بین داده‌ها در ناحیه (۳) حذف‌شده است؛ بنابراین عملیات نشان داده‌شده در داخل این ناحیه می‌توانند به‌صورت موازی انجام شوند.

۲-۱-۳- الگوریتم‌های پیاده‌سازی DWT دوبعدی

تبدیل موجک گسسته می‌تواند به دوبعدی یا چندبعدی توسعه یابد. الگوریتم‌های مختلفی ازجمله تبدیل موجک سطری_ستونی و تبدیل موجک مبتنی بر خط برای پیاده‌سازی DWT به‌صورت دوبعدی وجود دارد.

در پیاده‌سازی مبتنی بر سطری – ستونی (RCWT)، تبدیل موجک گسسته دوبعدی به دو DWT یک‌بعدی که پالایش افقی و پالایش عمودی نامیده شده‌اند، تقسیم‌شده است. پالایش افقی سطرهای تصویر ورودی را پردازش و ضرایب موجک را در ماتریس کمکی ذخیره می‌کند. سپس، پالایش عمودی ستون‌های ماتریس کمکی را، پردازش و نتایج را در ماتریس اصلی ذخیره می‌کند. به‌عبارت‌دیگر، در این الگوریتم همه سطرها قبل از شروع فرایند پالایش عمودی، پالایش می‌شوند. پیچیدگی محاسباتی پالایش افقی و عمودی یکسان و برابر $o\left(n^{۲}\right)$ است. در این

الگوریتم پالایش‌های افقی و عمودی به‌صورت جداگانه‌ای و به ترتیب به تصویر ورودی اعمال‌شده‌اند. هر یک از ماتریس‌های $N \times M$ نیازمند NMc بایت از حافظه هستند. جایی که c نشان‌دهنده تعداد بایت‌های موردنیاز برای بیان کردن یک ضریب موجک است [۱۵].

درروش تبدیل موجک مبتنی بر خط، پالایش عمودی بلافاصله پس‌ازاینکه تعداد سطرهای کافی که در رویکرد پیچش توسط طول ضرایب پالایش مشخص می‌شود، به‌صورت افقی پالایش شدند شروع می‌گردد. الگوریتم LBWT از یک حلقه برای پردازش سطرها و ستون‌ها باهم استفاده می‌کند. این الگوریتم، DWT دوبعدی برای یک تصویر $N \times M$ را توسط مراحل زیر انجام می‌دهد. ابتدا پالایش L سطر از تصویر را پردازش و نتایج مقادیر بالا گذر و پایین گذر را به‌صورت بازه_آمیخته در یک بافر $L \times M$ ذخیره می‌کند. سپس، ستون‌های این بافر کوچک توسط پالایش عمودی پردازش و نتایج در ماتریس نهایی ذخیره می‌گردد. درنهایت این مراحل جهت پردازش همه سطرها و ستون‌ها تکرار می‌گردد [۱۵].

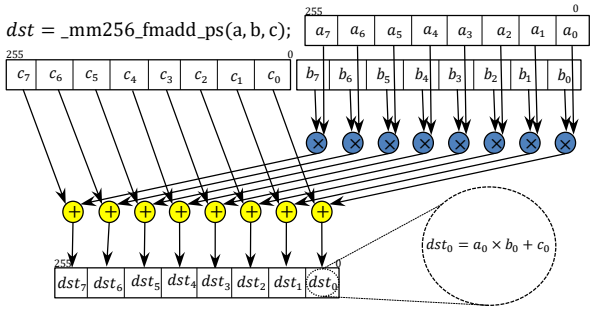
۲-۲- یک دستورالعمل و چندین داده

در چند سال اخیر، پردازشگرهای همه‌منظوره به‌منظور افزایش کارایی برنامه‌های کاربردی چندرسانه‌ای، معماری مجموعه دستورالعمل‌های خودشان را با توسعه SIMD ازجمله مجموعه دستورالعمل‌های توسعه‌های چندرسانه‌ای، جریان‌سازی توسعه SIMD، توسعه برداری پیشرفته، AVX2 و جمع ضرب ترکیبی گسترش داده‌اند. هر یک از آن‌ها مجموعه‌ای از دستورالعمل‌ها را جهت انجام عملیات SIMD بر روی عناصر داده صحیح و یا اعشاری بسته‌بندی‌شده فراهم می‌کنند.

توسعه MMX قابلیت پردازش بردارهای ۶۴ بیتی در دامنه صحیح و توسعه SSE قابلیت پردازش بردارهای ۶۴ و ۱۲۸ بیتی در دامنه صحیح و اعشاری را فراهم می‌کنند. درحالی‌که توسعه AVX، AVX2 و FMA قابلیت پردازش بردارهای ۲۵۶ بیتی را فراهم و ویژگی‌های پیشرفته‌تری نسبت به نسل‌های پیشین از توسعه SIMD فراهم می‌کنند.

توسعه FMA، توسعه AVX را بخصوص در محاسبات عددی اعشاری پیشرفت می‌دهد. به‌عنوان‌مثال در شکل ۵ ساختار دستورالعمل _mm256_fmadd_ps که یک دستورالعمل FMA است نشان داده‌شده است. این دستورالعمل عناصر ۳۲ بیتی اعشاری بسته‌بندی‌شده در a و b را ضرب، سپس نتایج میانی را با عناصر بسته‌بندی‌شده در c جمع و نتایج نهایی را در dst ذخیره می‌کند. درواقع امکان انجام شانزده عمل ضرب و جمع توسط یک دستورالعمل فراهم می‌سازد. مجموعه دستورالعمل‌های AVX2 انجام محاسبات در دامنه صحیح و اعشاری با استفاده از ثبات‌های برداری ۲۵۶ بیتی را فراهم می‌کنند [۱۶].

در مقالات زیادی از بردارسازی به‌منظور بهبود کارایی در الگوریتم‌های مختلف استفاده‌شده است. در [۱۷] بررسی و تحلیل‌ها نشان داده که استراتژی‌های برنامه‌نویسی، الگوریتم‌ها و ساختار داده‌ها مورداستفاده در اکثر برنامه‌های کامپیوتری مانع از بهره‌برداری کامپایلرها از بردارسازی خودکار می‌گردد. در [۱۸] دو روش به‌منظور کاهش فاصله[۱۷] میان بردارسازی خودکار و بردارسازی صریح ارائه‌شده است. بااین‌وجود، بردارسازی صریح کارایی را به میزان بیشتری نسبت به بردارسازی خودکار بهبود می‌بخشد و برای همین برخی از الگوریتم‌ها به روش صریح بردار سازی می‌شوند. در [۱۹] از بردارسازی صریح SIMD به‌منظور کاهش پیچیدگی و افزایش کارایی در الگوریتم هش و در [۲۰] از مجموعه دستورالعمل‌های SSE4.2، AVX و AVX2 به‌منظور بردارسازی صریح الگوریتم‌های پردازش تصویر استفاده‌شده که در آن‌ها کارایی را به میزان چشمگیری بهبود بخشیده‌اند. در [۲۱] نیز معماری یک FMA سه سطحی طراحی‌شده که در عین حفظ کارایی میزان ناحیه و انرژی مصرفی را به‌اندازه ۵۰ درصد کاهش داده است. به همین دلیل از بردارسازی صریح به‌عنوان روشی برای بهبود کارایی الگوریتم‌های مختلفی ازجمله 2D-DWT استفاده‌شده که در بخش بعدی پیشینه تحقیق بردارسازی DWT با استفاده از دستورالعمل‌های SIMD موردبررسی قرارگرفته است.



شکل ۵: ساختار دستورالعمل _mm256_fmadd_ps.

۲-۳- پیشینه تحقیق

در [۱۶، ۲۲] تبدیل CDF-9/7 با رویکرد پلکانی را با بهره‌برداری از مجموعه دستورالعمل‌های SIMD پیاده‌سازی کرده‌اند. آن‌ها از بردارسازی خودکار با استفاده از دستورالعمل‌های SSE بهره برده‌اند. درواقع در این کار، بر روی سلسله‌مراتب حافظه تمرکز شده و چندین تکنیک ازجمله کاشی‌کاری جهت بهبود تمرکز موضوعی در زمان و مکان مراجعات در نظر گرفته‌شده‌اند. همچنین، به‌منظور پیاده‌سازی دوبعدی DWT از الگوریتم LBWT استفاده‌شده است.

در [۲۳] بر روی بردارسازی تبدیل CDF-9/7 با رویکرد پلکانی تمرکز شده است. در این کار، تبدیل CDF-9/7 با استفاده از روش بردارسازی عمودی و دستورالعمل‌های SSE بردارسازی شده است. در این پیاده‌سازی پالایش افقی و عمودی تبدیل دوبعدی در یک حلقه منفرد ترکیب‌شده‌اند. به همین دلیل، در این پیاده‌سازی بافری با ابعاد $16 \times M$ جهت ذخیره‌سازی نتایج میانی نیاز است. جایی که M برابر با تعداد ستون‌های تصویر ورودی است.

در کارهای زیادی دو تبدیل Daub-4 اعشاری با رویکرد پیچش و CDF-5/3 صحیح با رویکرد پلکانی را به ترتیب با استفاده از دستورالعمل‌های SSE و MMX پیاده‌سازی کرده‌اند. در [۲۴، ۲۵] بردارسازی SIMD از DWT دوبعدی را با استفاده از الگوریتم تبدیل موجک سطری_ستونی ارائه کرده‌اند. در این پیاده‌سازی‌ها نشان داده‌شده که بردارسازی SIMD از پالایش عمودی به‌مراتب سرراست از پالایش افقی است. چراکه، در پالایش عمودی عناصری که باید پردازش شوند به‌صورت متوالی در حافظه ذخیره‌شده‌اند، درحالی‌که بردارسازی SIMD از پالایش افقی نیازمند بازآرایی[۱۸] عناصر ورودی است. همچنین در [۵، ۲۶] بردارسازی SIMD از این دو تبدیل با استفاده از الگوریتم تبدیل موجک مبتنی بر خط ارائه و نتایج حاصل از آن با بردارسازی با استفاده از الگوریتم RCWT مقایسه شده است. نتایج حاصل از پیاده‌سازی‌ها نشان می‌دهد اگرچه بردارسازی SIMD از الگوریتم RCWT آسان‌تر از الگوریتم LBWT است، اما بهبود کارایی بردارسازی SIMD از تبدیل موجک گسسته دوبعدی با استفاده از الگوریتم LBWT بیشتر است.

در [۲۷] تبدیل CDF-9/7 با رویکرد پلکانی را با بهره‌برداری از خصوصیات مجموعه دستورالعمل‌های SIMD پیاده‌سازی کرده‌اند. این پیاده‌سازی‌ها بر مبنای بردارسازی افقی، عمودی و مورب ارائه‌شده‌اند. نتایج ارزیابی کارایی نشان می‌دهد که بردارسازی SIMD از تبدیل CDF-9/7 با رویکرد پلکانی بر مبنای بردارسازی عمودی به‌مراتب کاراتر از بردارسازی آن بر مبنای بردارسازی افقی و مورب است.

جدول ۲ یک دید کلی از پیاده‌سازی‌های مختلف DWT دوبعدی بر روی پردازشگرهای همه‌منظوره باقابلیت بهره‌برداری از توسعه SIMD بر اساس معیارهای مختلف ازجمله رویکرد پیاده‌سازی (پیچش یا پلکانی)، الگوریتم پیاده‌سازی دوبعدی (RCWT یا LBWT)، تبدیل‌های موجک استفاده‌شده، نوع پیاده‌سازی (صحیح یا اعشاری)، مجموعه دستورالعمل‌های استفاده‌شده، سایر خصوصیات این پیاده‌سازی‌ها را نشان می‌دهد. همان‌طور که در این جدول مشاهده می‌شود در اکثر کارها از مجموعه دستورالعمل‌های MMX و SSE به‌منظور پیاده‌سازی 2D-DWT استفاده‌شده است.

۳- روش پیشنهادی

در این بخش بردارسازی SIMD از پنج موجک پرکاربرد ازجمله تبدیل‌های، هار و Daub-4 با رویکرد پیچش و تبدیل‌های هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی با استفاده از فنّاوری‌های AVX/AVX2 و جمع ضرب ترکیبی (FMA) پیشنهادشده است. این بردارسازی‌ها بر اساس نوع الگوریتم پیاده‌سازی تبدیل موجک گسسته دوبعدی به دودسته بردارسازی الگوریتم RCWT و بردارسازی الگوریتم LBWT تقسیم‌شده است. این دو بردارسازی در ادامه این بخش به‌صورت جداگانه‌ای بحث شده‌اند.

۳-۱- بردارسازی الگوریتم تبدیل موجک سطری_ستونی

در این بخش پیاده‌سازی‌های AVX/AVX2 از الگوریتم RCWT برای دو تبدیل هار و Daub-4 با رویکرد پیچش و سه تبدیل هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی ارائه‌شده و نتایج کارایی آن‌ها بحث شده است. جهت ساده‌سازی بردارسازی SIMD الگوریتم RCWT تا انتهای بخش ۳-۱، تنها بردارسازی SIMD نامیده شده است. این بخش به‌صورت زیر سازمان‌دهی شده است.

جدول ۲: یک دید کلی از پیاده‌سازی‌های مختلف DWT دوبعدی بر روی پردازشگرهای همه‌منظوره.

منبع	رویکرد	الگوریتم پیاده‌سازی دوبعدی	تبدیل	نوع پیاده‌سازی	مجموعه دستورالعمل	سایر خصوصیات
[۲۸]	پلکانی	LBWT	CDF-9/7	اعشاری	SSE	برای ذخیره نتایج میانی نیازمند یک بافر $16 \times M$ است. استفاده از تکنیک خط لوله.
[۲۶]	پلکانی	LBWT	CDF-9/7	اعشاری	SSE	برای ذخیره نتایج میانی نیازمند دو بافر است. استفاده از تکنیک بلاک بندی.
[۱۶، ۲۳]	پیچش و پلکانی	RCWT	Daub-4	اعشاری	SSE	استفاده از تکنیک موازی‌سازی سطح داده. بازآرایی عناصر ورودی برای پالایش افقی.
			CDF-5/3	صحیح	MMX	
[۵، ۲۴]	پیچش و پلکانی	LBWT	Daub-4	اعشاری	SSE	برای ذخیره نتایجی نیازمند یک بافر $L \times M$ است. استفاده از تکنیک موازی‌سازی سطح داده.
			CDF-5/3	صحیح	MMX	
[۲۹]	پلکانی	LBWT	CDF-9/7	اعشاری	AVX	برای ذخیره نتایجی میانی نیازمند یک بافر $4 \times M$ است. استفاده از تکنیک بلاک بندی.

پیاده‌سازی‌های SIMD از دو تبدیل هار و Daub-4 با رویکرد پیچش در زیر بخش ۳-۱-۱ ارائه‌شده است. سپس، در زیر بخش ۳-۱-۲ پیاده‌سازی‌های SIMD از تبدیل‌های هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی موردبحث قرارگرفته است.

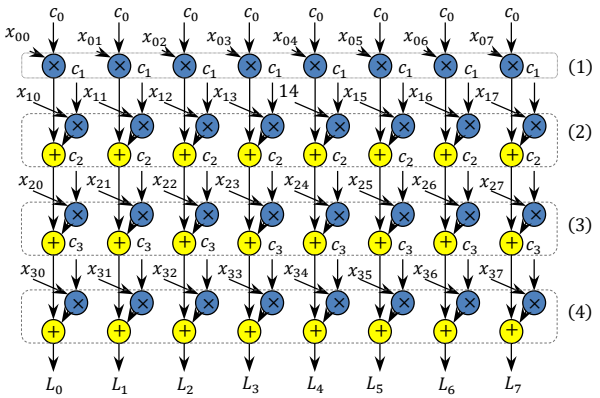
۳-۱-۱- پیاده‌سازی SIMD رویکرد پیچش

هر دو تبدیل هار و Daub-4 با رویکرد پیچش برای محاسبه ضرایب بالا گذر و پایین گذر از ضرایب پالایش اعشاری ۳۲ بیتی استفاده می‌کنند؛ درنتیجه، پیاده‌سازی SIMD این دو تبدیل بسیار مشابه‌اند. به همین دلیل، در ادامه این بخش پیاده‌سازی‌های SIMD این دو تبدیل باهم موردبحث قرار می‌گیرند.

با توجه به اعشاری بودن ضرایب پالایش این دو تبدیل بسیار مشخص است که بردارسازی آن‌ها با استفاده از دستورالعمل‌های AVX و استخراج هشت مسیر موازی‌سازی ۲۵۶ = ۳۲ × ۸ انجام می‌شود. همچنین، همان‌طور که قبلاً اشاره شد بردارسازی SIMD پالایش عمودی نسبت به پالایش افقی سرراست‌تر است. چون در پالایش عمودی عناصری که می‌توانند هم‌زمان پردازش شوند به‌صورت متوالی در حافظه ذخیره‌شده‌اند. به‌عنوان‌مثال، تبدیل Daub-4 را در نظر بگیرید که $x_{i,j}$ نمونه‌های ورودی، c_0، c_1، c_2 و c_3 ضرایب پایین گذر و $L_{i,j}$ مقادیر پایین گذر باشند. آنگاه پالایش عمودی از مقادیر پایین گذر را می‌توان با استفاده از معادله (۷) به دست آورد.

$$
\begin{aligned}
&(L_{i,j}\, L_{i,j+1}\, L_{i,j+2}\, L_{i,j+3}\, L_{i,j+4}\, L_{i,j+5}\, L_{i,j+6}\, L_{i,j+7}) = \\
&\big((c_0\, c_0\, c_0\, c_0\, c_0\, c_0\, c_0\, c_0) \times \\
&(x_{2i,j}\, x_{2i,j+1}\, x_{2i,j+2}\, x_{2i,j+3}\, x_{2i,j+4}\, x_{2i,j+5}\, x_{2i,j+6}\, x)\big) + \\
&\big((c_1\, c_1\, c_1\, c_1\, c_1\, c_1\, c_1\, c_1) \times \\
&(x_{2i+1,j}\, x_{2i+1,j+1}\, x\, x_{2i+1,j+3}\, x_{2i+1,j+4}\, x_{2i+1,j+5}\, x_{2i+1,j+6}\, x_{2i+1,j+7})\big) + \\
&\big((c_2\, c_2\, c_2\, c_2\, c_2\, c_2\, c_2\, c_2) \times \\
&(x_{2i+2,j}\, x_{2i+2,j+1}\, x_{2i+2,j+2}\, x_{2i+2,j+3}\, x_{2i+2,j+4}\, x_{2i+2,j+5}\, x\, x_{2i+2,j+7})\big) + \\
&\big((c_3\, c_3\, c_3\, c_3\, c_3\, c_3\, c_3\, c_3) \\
&\times (x_{2i+3,j}\, x_{2i+3,j+1}\, x_{2i+3,j+2}\, x\, x_{2i+3,j+4}\, x_{2i+3,j+5}\, x_{2i+3,j+6}\, x_{2i+3,j+7})\big) \quad (۷)
\end{aligned}
$$

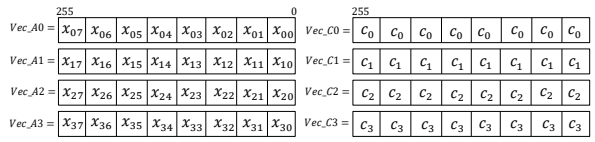
در این معادله، عملگر × نشان‌دهنده ضرب چند عنصری برداری است. چون قابلیت موازی‌سازی هشت داده با استفاده از یک دستور SIMD وجود دارد، هر ضریب به‌صورت یک عدد ۳۲ بیتی در یک ثبات SIMD ۲۵۶ بیتی، هشت بار تکرار شده است. معادله مشابهی نیز می‌توان برای مقادیر بالا گذر رسم کرد. شکل ۶ گراف جریان داده را برای پالایش عمودی نشان می‌دهد. همان‌طور که در شکل نشان داده‌شده، هشت نمونه ورودی از هر سطر به‌صورت هم‌زمان در یک ضریب پالایش (هر ضریب پالایش باید در هر هشت زیر باند یک ثبات رسانه تکرار شود) ضرب شده‌اند. بعد از هشت عملیات ضرب‌در چهار سطر متوالی با ضرایب مختلف، نتایج هر ستون با یکدیگر جمع شده‌اند.



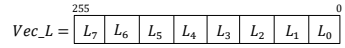
شکل ۶: گراف جریان داده پالایش عمودی برای تبدیل Daub-4.

شکل ۷ نحوه محاسبه هشت ضریب موجک خروجی برای پالایش عمودی با استفاده از دستورالعمل‌های AVX/FMA را نشان می‌دهد. همان‌طور که در شکل دیده می‌شود برای عملیات ضرب و جمع‌های متوالی (کادرهای خط‌چین (۲)، (۳) و (۴)) از دستورالعمل _mm256_fmadd_ps که یک دستورالعمل FMA است، استفاده گردیده است. در جدول ۳ کارایی آن را در مقایسه با کارایی دو دستورالعمل _mm256_mul_ps و _mm256_add_ps نشان داده‌شده است.

بردارسازی پالایش افقی به‌مراتب سخت‌تر از بردارسازی پالایش عمودی است. گراف جریان داده از پالایش افقی در شکل ۸ نشان داده‌شده است. همان‌طور که این شکل نشان می‌دهد چهار نمونه ورودی مختلف با چهار نمونه مختلف از ضرایب ضرب شده و نتایج میانی در یک عملوند میانه انباشته‌شده است.



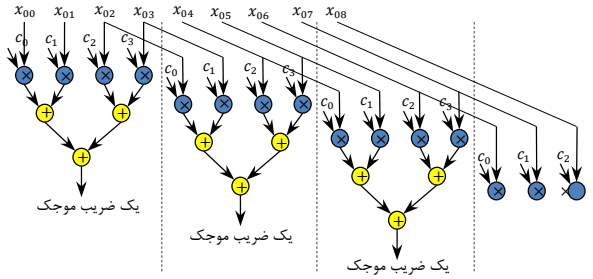
(1) Vec_L = _mm256_mul_ps(Vec_A0, Vec_C0);
(2) Vec_L = _mm256_fmadd_ps(Vec_A1, Vec_C1, Vec_L);
(3) Vec_L = _mm256_fmadd_ps(Vec_A2, Vec_C2, Vec_L);
(4) Vec_L = _mm256_fmadd_ps(Vec_A3, Vec_C3, Vec_L);



شکل ۷: محاسبه هشت ضریب موجک خروجی برای پالایش عمودی با استفاده از دستورالعمل‌های AVX/FMA.

جدول ۳: کارایی دستورالعمل _mm256_fmadd_ps در مقایسه با کارایی دو دستورالعمل _mm256_mul_ps و _mm256_add_ps

دستورالعمل	تأخیر	توان عملیاتی
_mm256_add_ps	۳	۱
_mm256-mul_ps	۵	۰/۵
_mm256_fmadd_ps	۵	۰/۵



شکل ۸: گراف جریان داده از پالایش افقی برای تبدیل Daub-4.

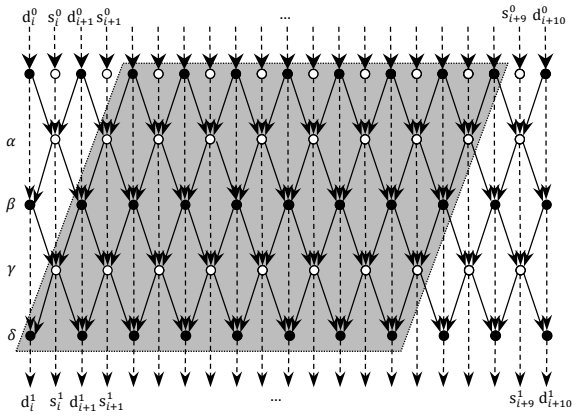
پیاده‌سازی پالایش افقی به روش پیاده‌سازی پالایش عمودی، نیازمند بازآرایی عناصر ورودی است. برای بازآرایی عناصر ورودی از دستورالعمل _mm256_setr_ps (تنظیم عناصر ۳۲ بیتی اعشاری بسته‌بندی‌شده در dst با عناصر عرضه‌شده به ترتیب عکس) استفاده‌شده است.

۳-۱-۲- بردارسازی SIMD رویکرد پلکانی

به دلیل ساختار رویکرد پلکانی تبدیل‌های هار، CDF-5/3 و CDF-9/7 با این رویکرد به روش کاملاً متفاوتی نسبت به تبدیل‌های هار و Daub-4 با رویکرد پیچش بردارسازی شده‌اند. همچنین، پیاده‌سازی‌های SIMD از تبدیل‌های هار و CDF-5/3 با رویکرد پلکانی بسیار متفاوت با پیاده‌سازی SIMD از تبدیل CDF-9/7 با رویکرد پلکانی است. اولاً، تبدیل‌های هار و CDF-5/3 با رویکرد پلکانی از محاسبات صحیح استفاده می‌کنند و به همین دلیل پیاده‌سازی‌های SIMD آن‌ها دستورالعمل‌های AVX2 را بکار می‌برد. درحالی‌که پیاده‌سازی SIMD تبدیل CDF-9/7 با رویکرد پلکانی به دلیل اعشاری بودن محاسبات از دستورالعمل‌های AVX استفاده می‌کند. دوماً، با توجه به ضرایب پالایش تبدیل‌های هار و CDF-5/3 نشان داده‌شده در جدول ۱ می‌توان این ضرایب پالایش را با استفاده عملیات ضرب صحیح و گردش‌به‌راست اعمال کرد [۲۶]. در پیاده‌سازی SIMD تبدیل‌های هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی از بردارسازی عمودی بهره گرفته‌شده است.

در شکل ۹ گراف جریان داده پالایش افقی تبدیل CDF-9/7 نشان داده‌شده است. همان‌طور که در شکل دیده می‌شود این تبدیل دارای چهار مرحله پلکانی است. به‌راحتی می‌توان محاسبات مراحل پلکانی این تبدیل را به‌صورت متوالی با استفاده از دستورالعمل‌های AVX نگاشت کرد و در هر تکرار حلقه هشت ضریب بالا گذر و هشت ضریب پایین گذر (ناحیه خاکستری‌رنگ) را محاسبه کرد؛ اما بردارسازی پالایش افقی تبدیل CDF-9/7 همانند تبدیل‌های هار و CDF-5/3 نیازمند بازآرایی دنباله‌های زوج و فرد است.

این بردارسازی با استفاده از دستورالعمل _mm256_setr_ps انجام‌شده است. همچنین، چون مقادیر خروجی مرحله پلکان قبلی (به ترتیب مقادیر بالا گذر مرحله پیش‌بینی یک، مقادیر پایین گذر مرحله به‌روزرسانی یک و مقادیر بالا گذر مرحله پیش‌بینی دوم) که برای مرحله پلکان بعدی (به ترتیب مرحله بروز رسانی یک، مرحله پیش‌بینی دوم و مرحله بروز رسانی دوم) نیاز است را نمی‌توان در یک ثبات نگه داشت، دسترسی به آن‌ها نیازمند بارگذاری مجدد مقادیر خروجی مرحله پلکان قبلی است. بردارسازی پالایش عمودی تبدیل CDF-9/7 نسب به پالایش افقی ساده‌تر است. چراکه، به دلیل قرار گرفتن دنباله‌های زوج و فرد در سطرهای کاملاً جداگانه، پیاده‌سازی آن نیازمند بازآرایی دنباله‌های زوج و فرد ورودی نیست. همانند پیاده‌سازی تبدیل‌های با رویکرد پیچش، بردارسازی پالایش عمودی آسان‌تر بردارسازی پالایش افقی است. در این مورد، دنباله‌های زوج و فرد نیاز نیست که از هم جدا شوند چون آن‌ها در سطرهای مختلفی قرار دارند؛ اما همانند پالایش افقی به دلیل انجام شدن عملیات جمع در مرحله بروز رسانی، دو کپی از دنباله فرد که یکی از d_{i+1}^0 و دیگری از d_i^0 شروع می‌شود نیاز است. در شکل ۱۰ پیاده‌سازی پالایش عمودی تبدیل CDF-5/3 با رویکرد پلکانی با استفاده از دستورالعمل‌های AVX2 برای شانزده ضریب پایین گذر و شانزده ضریب بالا گذر نشان داده‌شده است.



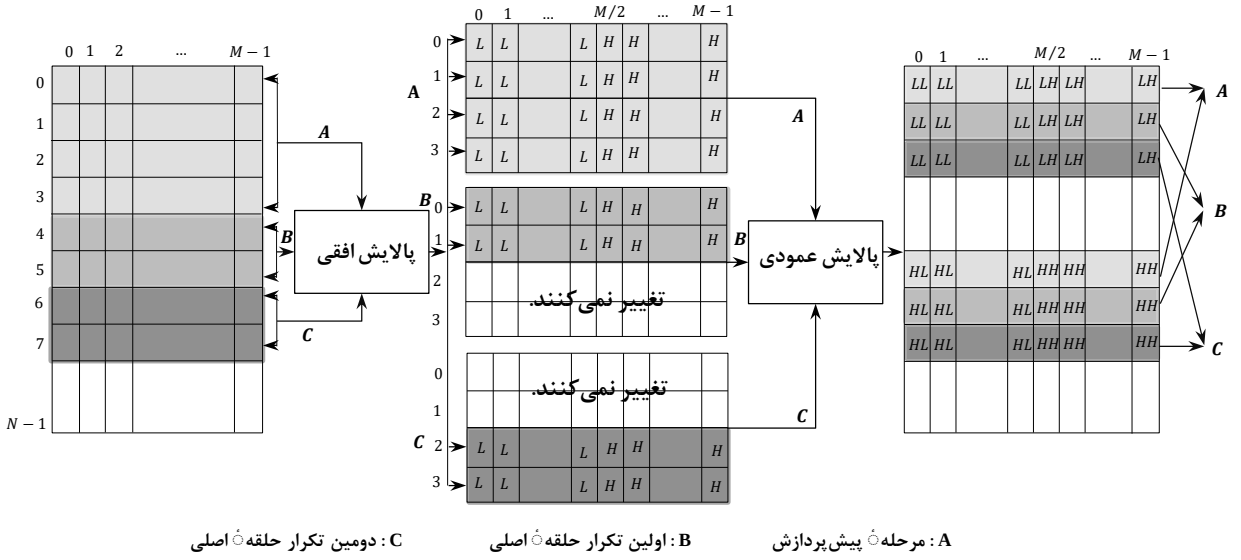
شکل ۹: گراف جریان داده از تبدیل CDF-9/7.

```
Vec_A1 = _mm256_loadu_si256 ((__m256i *)&temp[i][j]);
Vec_A2 = _mm256_loadu_si256 ((__m256i *)&temp[i + 1][j]);
Vec_A3 = _mm256_loadu_si256 ((__m256i *)&temp[i + 2][j]);

//------------Predict 1------------------
Vec_A3 = _mm256_add_epi16(Vec_A1, Vec_A3);
Vec_A3 = _mm256_srai_epi16(Vec_A3, 1);
Vec_A3 = _mm256_sub_epi16(Vec_A2, Vec_A3);
int k = i / 2 + N/2;
_mm256_store_si256((__m256i *)&image[k][j], Vec_A3);

//------------Update 1--------------
Vec_A2 = _mm256_loadu_si256 ((__m256i *)&image[k - 1][j]);
Vec_A2 = _mm256_add_epi16(Vec_A3, Vec_A2);
Vec_A2 = _mm256_srai_epi16(Vec_A2, 2);
Vec_A2 = _mm256_add_epi16(Vec_A2, Vec_A1);
_mm256_store_si256((__m256i *)&image[i / 2][j], Vec_A2);
```

شکل ۱۰: پیاده‌سازی پالایش عمودی تبدیل CDF-5/3 با رویکرد پلکانی با استفاده از دستورالعمل‌های AVX2 برای شانزده ضریب پایین گذر و شانزده ضریب بالا گذر.



شکل ۱۱: مرحله پیش‌پردازش و دو تکرار حلقه اصلی پیاده‌سازی SIMD الگوریتم LBWT برای تبدیل Daub-4 با رویکرد پیچش با استفاده از موازی‌سازی هشت-مسیره.

۳-۲- بردارسازی الگوریتم تبدیل موجک مبتنی بر خط

همان‌طور که در بخش ۲-۱-۳ بیان شد، الگوریتم تبدیل موجک مبتنی بر خط برای پردازش سطرها و ستون‌ها تنها از یک حلقه استفاده می کند. در این بخش ابتدا بردارسازی الگوریتم LBWT برای دو تبدیل هار و Daub-4 با رویکرد پیچش و سه تبدیل هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی در زیر بخش ۳-۲-۱ ارائه‌شده است. سپس، نتایج ارزیابی کارایی برای بردارسازی الگوریتم LBWT در زیر بخش ۳-۲-۲ ارائه گردیده است.

۳-۲-۱- بردارسازی الگوریتم LBWT برای رویکرد پیچش و پلکانی

بردارسازی الگوریتم LBWT برای دو تبدیل هار و Daub-4 با رویکرد پیچش و سه تبدیل هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی بسیار مشابه است. به‌منظور بردارسازی الگوریتم LBWT بافر صف حلقوی به‌اندازه $L \times M$ استفاده می‌شود، جایی که M برابر با عرض تصویر ورودی و L برای تبدیل‌های مختلف متفاوت است. به‌عنوان‌مثال، اندازه L برای Daub-4 با رویکرد پیچش برابر طول ضرایب یعنی چهار است.

در شکل ۱۱ مرحله پیش‌پردازش و دو تکرار حلقه اصلی پیاده‌سازی SIMD الگوریتم LBWT برای تبدیل Daub-4 با رویکرد پیچش با استفاده از موازی‌سازی هشت-مسیره نشان داده‌شده است. در مرحله پیش‌پردازش ابتدا چهار سطر از تصویر ورودی به‌صورت افقی پردازش و نتایج در چهار سطر از بافر ذخیره می‌شود. سپس، بافر پرشده به‌صورت عمودی پالایش‌شده و ضرایب موجک محاسبه‌شده در یک ماتریس $N \times M$ اضافی ذخیره می‌گردد. در اولین تکرار حلقه اصلی، ابتدا دو سطر بعدی از تصویر ورودی به‌صورت افقی پردازش و نتایج دست‌آمده در سطرهای ۰ و ۱ بافر ذخیره می‌شوند. چراکه این دو سطر در مرحله قبلی به‌صورت عمودی پالایش‌شده و دیگر مورداستفاده قرار نمی‌گیرند. سپس، پالایش عمودی به بافر اعمال می‌گردد. در دومین تکرار حلقه اصلی، مراحل موجود در اولین تکرار حلقه با این تفاوت که نتایج حاصل از پالایش افقی در سطرهای ۲ و ۳ از بافر ذخیره می‌گردد، تکرار می‌شوند. درنهایت این دو مرحله به‌صورت متوالی بروی کل تصویر ورودی اعمال می‌گردند.

جدول ۴: خصوصیات بافرهای استفاده‌شده برای بردارسازی الگوریتم LBWT پنج تبدیل مدنظر.

رویکرد	تبدیل	تعداد بافر	اندازه بافر
پیچش	هار	۱	$2 \times M$
	Daub-4	۱	$4 \times M$
پلکانی	هار	۱	$2 \times M$
	CDF-5/3	۱	$3 \times M$
	CDF-9/7	۲	$5 \times M$

جدول ۴ خصوصیات بافرهای استفاده‌شده برای بردارسازی الگوریتم LBWT دو تبدیل هار و Daub-4 با رویکرد پیچش و سه تبدیل هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی را نشان می‌دهد. به این نکته باید توجه کرد که به دلیل وجود چهار مرحله پلکانی در تبدیل CDF-9/7 این تبدیل نیازمند یک بافر اضافی برای ذخیره‌سازی نتایج حاصل از دو مرحله اول پلکانی است. چون این نتایج در تکرار بعدی حلقه مورداستفاده قرار می‌گیرند. همچنین، تبدیل‌های Daub-4 و هار با رویکرد پیچش و CDF-9/7 با رویکرد پلکانی به‌صورت اعشاری و با استفاده از دستورالعمل‌های AVX/FMA پیاده‌سازی شده‌اند. درحالی‌که

بردارسازی تبدیل‌های هار و CDF-5/3 با رویکرد پلکانی به‌صورت صحیح و با استفاده از دستورالعمل‌های AVX2 انجام‌شده است.

۴- ارزیابی کارایی

۴-۱- مجموعه داده‌ها

به‌منظور ارزیابی کارایی بردارسازی SIMD بر اساس روش پیشنهادی برای پنج موجک پرکاربرد ازجمله تبدیل‌های هار و Daub-4 با رویکرد پیچش که در ادامه به ترتیب با Haar_FB و Daub-4_FB نشان داده می شوند و تبدیل‌های هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی نیز به ترتیب با Haar_LS، CDF-5/3_LS و CDF-9/7_LS نشان می‌شوند ارائه‌شده است. جدول ۵ یک دید کلی از خصوصیات پیاده‌سازی انجام‌شده از این تبدیل‌ها را بر اساس معیارهای مختلف ازجمله رویکرد پیاده‌سازی (پیچش یا پلکانی)، نوع پیاده‌سازی (صحیح یا اعشاری)، مجموعه دستورالعمل‌های استفاده‌شده و سطح موازی‌سازی به‌دست‌آمده را نشان می‌دهد. در ستون آخر این جدول، ماکزیمم تعداد داده‌هایی که به‌طور موازی در هر یک از تبدیل‌ها می‌توانند پردازش شوند را نشان داده‌شده است.

جدول ۵: یک دید کلی از خصوصیات پیاده‌سازی پنج تبدیل موجک.

تبدیل	رویکرد	نوع پیاده‌سازی	مجموعه دستورالعمل‌های SIMD	سطح موازی‌سازی قابل‌استخراج
هار	پیچش	اعشاری	AVX	۸
Duab-4	پیچش	اعشاری	AVX/FMA	۸
هار	پلکانی	صحیح	AVX2	۱۶
CDF-5/3	پلکانی	صحیح	AVX2	۱۶
CDF-9/7	پلکانی	اعشاری	AVX/FMA	۸

۴-۲- محیط ارزیابی کارایی و تصاویر محک‌زن

همه پیاده‌سازی‌ها بر روی پردازشگر Intel® Core™ i7-4500U با ریز معماری Haswell اجراشده و پارامترهای معماری این سیستم در جدول ۶ به‌صورت خلاصه نشان داده‌شده است. همه نسخه‌ها با استفاده از gcc با بهینه‌سازی سطح O2– کامپایل و بر روی یک سیستم با بارکاری سبک اجراشده‌اند. کارایی با استفاده از نرخ تعداد چرخه اجرا محاسبه‌شده است. تعداد کلی چرخه‌ها با استفاده از شمارنده چرخه به‌دست‌آمده است. شمارنده چرخه یک ابزار بسیار دقیق برای اندازه‌گیری زمان مصرف‌شده بین دونقطه مختلف در اجرای برنامه است. برای حذف کردن اثرات تعویض متن و فقدان‌های حافظه نهان اجباری، از رویکرد اندازه‌گیری K-بهترین و گرم کردن حافظه نهان استفاده‌شده است. این بدان معناست که تابع K بار اجراشده و سریع‌ترین زمان در نظر گرفته‌شده است. اجرای تابع حداقل یک‌بار قبل از شروع اندازه‌گیری تأثیر فقدان‌های حافظه نهان داده و دستورالعمل را کاهش می‌دهد [۳۲-۳۰]. برای ارزیابی کارایی پیاده‌سازی‌ها شش تصویر محک‌زن[۱۹] استاندارد در پردازش تصویر که در شکل ۱۲ نشان داده‌شده‌اند، با ابعاد متفاوت از ۱۲۸ × ۱۲۸ تا ۴۰۹۶ × ۴۰۹۶ مورداستفاده قرارگرفته‌اند و متوسط کارایی به ازای هر اندازه تصویر در شکلهای ۱۳ تا ۱۵ نشان داده شده است. تمامی پیاده‌سازی‌ها، کدهای سریال و موازی توسط یک برنامه‌نویس و بر روی یک سیستم مورد آزمون و ارزیابی قرارگرفته است.

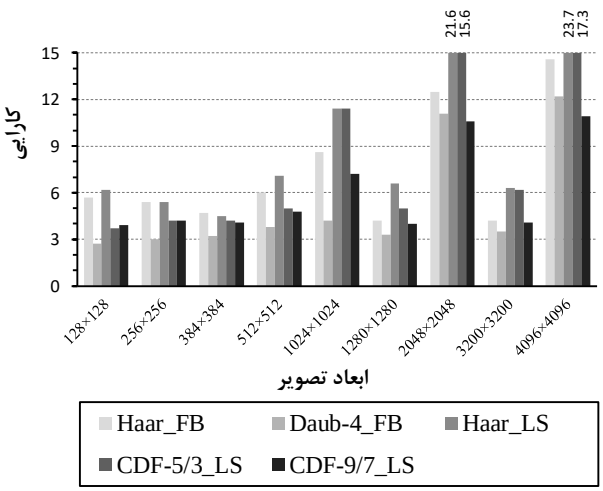
جدول ۶: پارامترهای معماری سکوی پیاده‌سازی.

واحد پردازشگر	Intel® Core™ i7-4500U CPU @1.80 GHz
حافظه نهان L1	128 KB
حافظه نهان L2	512 KB
حافظه نهان L3	4 MB
حافظه اصلی	6 GB

شکل ۱۲: تصاویر محک‌زن استاندارد استفاده‌شده برای ارزیابی کارایی

۴-۳- ارزیابی کارایی بردارسازی الگویتم RCWT

در این بخش پیاده‌سازی‌های سریال تبدیل‌ها به‌عنوان پیاده‌سازی پایه در نظر گرفته‌شده و نتایج پیاده‌سازی‌های SIMD موازی شده برای تمامی تبدیل‌ها با آن‌ها مقایسه شده‌اند. در شکل ۱۳ کارایی پیاده‌سازی‌های SIMD برای تبدیل‌های هار و Daub-4 با رویکرد پیچش و تبدیل‌های هار، CDF-5/3 و CDF-9/7 با رویکرد پلکانی نسبت به پیاده‌سازی سریال الگوریتم‌های متناظر نشان داده‌شده است.

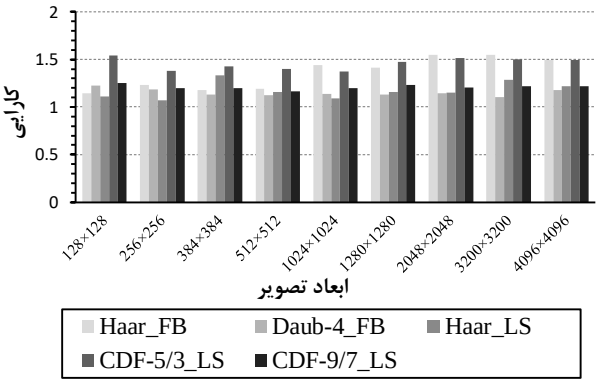


شکل ۱۳: کارایی پیاده‌سازی‌های SIMD تبدیل‌های مختلف نسبت به پیاده‌سازی سریال.

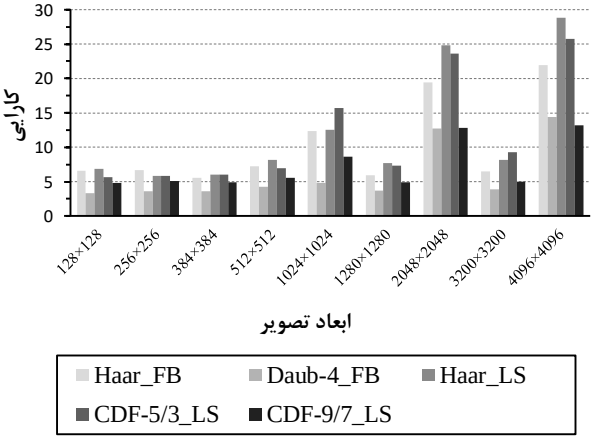
همان‌طور که در شکل ۱۳ مشاهده می‌شود بالاترین میزان افزایش کارایی مربوط به تبدیل هار Haar_LS (23.7x) و CDF-5/3_LS (17.3x) با رویکرد پلکانی است. چراکه در این تبدیل‌ها به دلیل صحیح بودن ضرایب از دستورالعمل‌های AVX2 برای بردارسازی استفاده‌شده است. به‌عبارت‌دیگر تعداد داده‌های که به‌طور موازی پردازش می‌شوند همان‌طوری که در جدول ۵ نشان داده‌شده است ۱۶ است. برای سایر تبدیل‌های دیگر دارای میزان افزایش کارایی بین 2.6x تا 14.6x برای ابعاد مختلف هستند. همچنین با افزایش ابعاد تصاویر میزان افزایش کارایی برای تمامی تبدیل‌های افزایش‌یافته است.

۴-۴- ارزیابی کارایی بردارسازی الگوریتم LBWT

در شکل ۱۴ کارایی بردارسازی الگوریتم LBWT برای تبدیل‌های هار و Daub-4 با رویکرد پیچش و تبدیل‌های هار، CDF-5/3 و CDF-9/7 نسبت به بردارسازی الگوریتم RCWT نشان داده‌شده است. همان‌طور که در شکل دیده می‌شود برای تمامی تبدیل‌ها بردارسازی الگوریتم LBWT کارایی را نسبت به بردارسازی الگوریتم RCWT بهبود بخشیده است. چراکه در تمام نمودارها کارایی بیشتر از یک است.



شکل ۱۴: کارایی بردارسازی الگوریتم LBWT برای تبدیل‌های موردنظر نسبت به بردارسازی الگوریتم RCWT.



شکل ۱۵: کارایی بردارسازی الگوریتم LBWT برای تبدیل‌های مدنظر نسبت به پیاده‌سازی سریال.

همچنین شکل ۱۵ کارایی موازی‌سازی LBWT تبدیل‌ها را نسبت به پیاده‌سازی سریال نشان می‌دهد. با توجه به شکل، بردارسازی LBWT میزان کارایی را برای تبدیل‌های مختلف با ابعاد تصویر مختلف بین ۳/۳ الی ۲۸/۸ برابر بهبود می‌بخشد. در دو موجک هار (Haar_LS) و CDF-5/3_LS کارایی نسبت به سایر تبدیل‌ها بیشتر است. چون در بردارسازی SIMD این دو تبدیل از مجموعه دستورالعمل‌های AVX2، اعداد صحیح باقابلیت موازی‌سازی شانزده داده در یک دستور SIMD استفاده‌شده است. درحالی‌که برای سه تبدیل دیگر از مجموعه دستورالعمل‌های AVX، اعداد اعشاری باقابلیت موازی‌سازی هشت داده در یک دستورالعمل استفاده‌شده است. همچنین میزان بهبود کارایی برنامه نویسی SIMD برای تصاویری که دارای ابعاد توانی از دو، 2^n هستند بدلیل فقدان‌های حافظه نهان کمتر نسبت به سایر ابعاد دیگر بیشتر است. با توجه به نتایج ارائه‌شده در شکل‌های ۱۳ و ۱۵ می‌توان نتیجه گرفت که فناوری‌های جدید SIMD، کارایی تبدیل‌های مختلف 2D-DWT را به‌طور قابل‌ملاحظه‌ای افزایش می‌دهد و هزینه این افزایش فقط روش برنامه‌نویسی و نگاشت از برنامه سریال به برنامه موازی است. همچنین با توجه به شکل ۱۴ می‌توان نتیجه گرفت که کارایی الگوریتم برداری‌سازی LBWT نسبت به RCWT بیشتر است.

۵- نتیجه‌گیری

تبدیل موجک گسسته دوبعدی (2D-DWT) یک ابزار قوی در پردازش سیگنال‌های چندرسانه‌ای است. ولی این تبدیل ازنظر محاسباتی وقت‌گیر و حجم محاسبات آن زیاد است. به‌منظور بهبود کارایی این تبدیل از موازی‌سازی سطح داده با استفاده از مجموع دستوارت توسعه یافته، یک دستورالعمل و چند داده (SIMD) با استفاده از فنّاوری‌های AVX/AVX2 و FMA که تقریباً در تمامی پردازنده‌های همه‌منظوره جدید موجود هستند و قابلیت پردازش ثبات‌های ۲۵۶ بیتی را دارند پیشنهاد شد. به‌منظور موازی‌سازی از دو روش بردارسازی پردازش سطری-ستونی (RCWT) که سطرها و ستون‌های تصویر جداگانه پردازش می‌شوند و مبتنی بر خط (LBWT) که هردوی سطرها و ستون‌ها در یک حلقه پردازش می‌شوند استفاده گردید. پیچیدگی پیاده‌سازی برداری الگوریتم LBWT به دلیل انجام پالایش افقی و پالایش عمودی در یک حلقه، نسبت به بردارسازی الگوریتم RCWT بیشتر است؛ اما کارایی موازی‌سازی آن نسبت به RCWT بیشتر است. به‌طوری‌که حداکثر کارایی قابل‌استخراج با استفاده از RCWT به ازای تبدیل‌های مختلف موجک و رویکردهای پیچش و پلکانی و اندازه تصاویر متفاوت تا ۴۰۹۶، ۲۳/۷ درحالی‌که با استفاده از LBWT ۲۸/۸ برابر است. کلیه پیاده‌سازی‌ها به روش صریح بوده که توسط یک برنامه‌نویس صورت گرفته است. برای کارهای آتی پیشنهاد می‌گردد که عملیات بردارسازی با استفاده از بهینه‌سازهای کامپایلرها به‌طور خودکار، ضمنی صورت گیرد و میزان کارایی آن را، با بردارسازی صریح مورد مقایسه و ارزیابی قرارداد و محدودیت‌های موجود در کامپایلرها به دلیل عدم موقعیت در برخی موارد برای بردارسازی شناسایی گردد.

مراجع

[1] M. Rabbani, and R. Joshi, "An Overview of the JPEG2000 Still Image Compression Standard," Signal Processing: Image Communication, vol. 17, no. 1, pp. 3–48, 2002.

[2] S. Battista, F. Casalino, and C. Lande, "MPEG-4: A Multimedia Standard for the Third Millenium, part 1," IEEE Multimedia, vol. 6, no. 4, pp.74–83, 1999.

[3] S. Battista, F. Casalino, and C. Lande, "MPEG-4: A Multimedia Standard for the Third Millenium, part 2," IEEE Multimedia, vol. 6, no. 4, pp.76–84, 2000.

[4] M. D. Adams, and R.K. Ward, "JasPer: A Portable Flexible Open-Source Software Tool Kit for Image Coding Processing," In Proc. IEEE, Int. Conf. on Acoustics, Speech, and Signal Processing, vol. 5, pp. 241-244, 2004.

[5] A. Shahbahrami, "Improving the Performance of 2D Discrete Wavelet Transform using Data-Level Parallelism", In IEEE Int. Conf. on High Performance Computing and Simulation, pp. 362-368, 2011.

[6] A. Shahbahrami, "Algorithms and Architectures for 2D Discrete Wavelet Transform," The Journal of Supercomputing, vol. 62, no. 2, pp. 1045-1064, May 2012.

[۷] عبدالبصیر تیباش و اسدالله شاه بهرامی، «مقایسه کارایی پیاده‌سازی صحیح و اعشاری فیلترهای مختلف تبدیل موجک گسسته دوبعدی،» هشتمین کنفرانس فناوری اطلاعات و دانش، دانشگاه بوعلی سینا، ۱۷و۱۸ شهریور ۱۳۹۵.

[۸] فرشته صادقی، ابوالفضل جلیلوند و سیدهادی حسینی، «ارائه یک روش ترکیبی مبتنی بر تبدیل موجک گسسته برای پیشبینی بارالکتریکی با استفاده از یک مدل دوبعدی،» مجله مهندسی برق دانشگاه تبریز، جلد ۴۵، شماره ۳، صفحه ۶۸-۷۸، پاییز ۹۴.

[۹] حسین شایقی و علی قاسمی، «پیشبینی قیمت روزانه برق با شبکه عصبی بهبودیافته مبتنی بر تبدیل موجک و روش آشوبناک جستجوی گرانشی،» مجله مهندسی برق دانشگاه تبریز، جلد ۴۵، شماره ۴، صفحه ۱۰۳-۱۱۳، پاییز ۹۴.

[10] Z. N. Li, M.S. Drew, and J. Liu, Fundamentals of Multimedia, NJ: Prentice Hall, 2014.

[11] H. Liao, M.K. Mandal, and B.F. Cockburn, "Efficient Architectures for 1-D and 2-D Lifting-Based Wavelet Transforms," IEEE Trans. on Signal Processing, vol. 52, no. 5, pp. 1315-1326, 2004.

[12] T. M. Quan, and , W.K. Jeong, "A Fast Discrete Wavelet Transform using Hybrid Parallelism on GPUs," IEEE Trans. on Parallel and Distributed Systems, vol. 27, no. 11, pp. 3088-3100, 2016.

[13] D. Barina, and P. Zemcik, "Minimum Memory Vectorisation of Wavelet Lifting," In Int. Conf. on Advanced Concepts for Intelligent Vision Systems, Springer, pp. 91-101, 2013.

[14] D. Barina, and P. Zemcik, "Diagonal Vectorisation of 2-D Wavelet Lifting," In IEEE Int. Conf. on Image Processing, pp. 2978-2982, 2014.

[15] A. Shahbahrami, and B.H. Juurlink, *SIMD Architectural Enhancements to improve the Performance of the 2D Discrete Wavelet Tranform*, 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, pp. 497-504, 2009.

[16] D. Chaver, C. Tenllado, L. Piñuel, M. Prieto, and F. Tirado, "2-D Wavelet Transform Enhancement on General-Purpose Microprocessors: Memory Hierarchy and SIMD Parallelism Exploitation," In Proc. Int. Conf. on the High Performance Computing, vol. 2552, pp. 9-21, 2002.

[17] F. Yazdanpanah, "An Approach for Analyzing Auto-vectorization Potential of Emerging Workloads", Microprocessors and Microsystems, vol. 49, pp. 139-149, 2017.

[18] H. Saito, S. Preis, N. Panchenko, and X. Tian, *Reducing the Functionality Gap between Auto-vectorization and Explicit Vectorization*, Int. workshop on OpenMP, pp. 173-186, 2016.

[19] T. Behrens, V. Rosenfeld, J. Traub, S. Breb, and V. Markl, "Efficient SIMD Vectorization for Hashing in OpenCl", Int.Conf. On Extending Database Technology, vol. 3, no. 4, 2018.

[20] O. Reiche, C. Kobylko, F. Hanniq, J. Teich, "Auto-vectorization for Image Processing DSLs", Proc. in Int. Conf. on Languages, Compilers, and Tools for Embedded Systems, vol. 52, no. 5, pp. 21-30, 2017.

[21] V. Arunachalam, A.N. J. Raj, N. Hampannavar, C. B. Bidul, "Efficient Dual-precision Floating-point Fused-multiply-add Architecture," Microprocessors and Microsystems, vol. 57, no. 1, pp. 23-31, 2018

[22] D. G. Chaver, C. Tenllado, L. Pinuel, M. Prieto, and F. Tirado, *Vectorization of the 2D Wavelet Lifting Transform using SIMD Extensions*, IEEE Workshop on Parallel and Distributed Image Processing, Video Processing, and Multimedia, pp. 8-12, 2003.

[23] R. Kutil, *A Single-Loop Approach to SIMD Parallelization of 2D Wavelet Lifting*, In IEEE Proc. of the 14th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Processing, pp. 413–420, 2006.

[24] A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "Implementing the 2-D Wavelet Transform on SIMD-Enhanced General-Purpose Processors," IEEE Tran. on Multimedia, vol. 10, no. 1, pp. 43-51, 2008.

[25] A. Shahbahrami, B. Juurlink, and S. Vassiliadis, *Performance Comparison of SIMD Implementations of the Discrete Wavelet Transform*, In 16th IEEE Int. Conf. on Application-Specific Systems, Architecture Processors, pp. 393-398, 2005.

[26] A. Shahbahrami, and B. Juurlink, *A Comparison of Two SIMD Implementations of the 2D Discrete Wavelet Transform*, In Proc. 18th Annual Workshop on Circuits, Systems and Signal Processing, pp. 169-177, 2007.

[27] D. Barina, and P. Zemcik, *Vectorization and Parallelization of 2-D Wavelet Lifting*, Journal of Real-Time Image Processing, pp. 1-13, 2015.

[28] H. B. Prajapati, and S.K. Vij, *Analytical Study of Parallel and Distributed Image Processing*, In IEEE Int. Conf. on Image Information Processing, pp. 1-6, 2011.

[29] P.P. Dang, and P.M. Chau, *Integer Fast Wavelet Transform and its VLSI Implementation for Low Power Applications*, In IEEE Workshop on Signal Processing Systems, pp. 93-98, 2002.

[30] Intel Corporation, Intel® 64 and IA-32 Architectures. Software Developer's Manual, System Programming Guide, vol. 3A, Part 1, 2010.

[31] D.B. Stewart, *Measuring Execution Time and Real-Time Performance*, In Embedded Systems Conf., vol. 141, 2001.

[32] Intel Corporation, Intel® Architecture Code Analyzer, User Guide, Document Number: 321356-001US, 2009.

زیرنویس‌ها

[1] Discrete Wavelet Transform
[2] Discrete Cosine Transform
[3] General-Purpose Processor
[4] Single Instruction, Multiple Data
[5] Row-Column Wavelet Transform
[6] Line-Based Wavelet Transform
[7] Daubechies-4
[8] Cohan-Daubechies-Feauveau-5/3
[9] MultiMedia eXtentions
[10] Streaming SIMD Extensions
[11] Advanced Vector Extensions
[12] Fused Multiply-Add
[13] Data Level Parallelism
[14] Vectorization
[15] Convolution
[16] Lifting scheme
[17] Gap
[18] Rearrangement
[19] Benchmark