JIPET

Journal of Intelligent Procedures in Electrical Technology
Vol. 16/ No. 63/ Autumn 2025
P-ISSN: 2322-3871, E-ISSN: 2345-5594, http://jipet.iaun.ac.ir/

https://dorl.net/dor/20.....
Research Article

Converting RGB Images to Gray-Scale by the Weighted Average Method, based on Shift-and-Add Technique on the Combination of Color Components for Reducing the Computational Units and Errors in FPGA

Mahdi Ajamin Hamedani[1], *M.Sc.*, Payam Sanaee[1,2], *Assistant Professor*

[1]Department of Electrical Engineering, Najafabad Branch, Islamic Azad University, Najafabad, Iran
[2]Digital Processing and Machine Vision Research Center, Najafabad Branch, Islamic Azad University, Najafabad, Iran
mahdi.ajamin@gmail.com, p.sanaee@pel.iaun.ac.ir

Abstract
Converting RGB (red-green-blue) images to gray-scale is one of the important and fundamental issues in the field of image processing, so many algorithms have been developed to achieve this purpose. These algorithms are used in the most of the machine vision applications pre-processing unit to recognize face, target and objects. In the nearly all image recognition processes, the frame rate of the images which are applied to the system through the digital cameras, is too high; Therefore, in order to achieve real-time processing, the speed of algorithm calculations must be increased. Field programmable gate arrays (FPGA) chips are one of the choices to process algorithms by hardware rapidly. The advantages of these chips are the possibility of hardware implementation of procedures by concurrent and parallel processing algorithms and fully combinational logic circuits. In this research to reduce the calculation error, we used a fixed-point system, and we have a tradeoff between the accuracy and the used logic-blocks. This will help us to manage the hardware resources perfectly. In this article, we design and compare different methods to convert color image to gray on inexpensive FPGA chip. By using the combining color components method with fixed-point calculations (the fractional part of the image components coefficients are 8/15 bits, and in the calculations, the fractional part of numbers 8 bits), the mean square error (MSE) index for the Lenna 512×512 grayscale image was 0.0184 and 105 logic block (LB) units were used to implement the corresponding hardware.

Keywords: computational complexity, fixed-point, field programmable gate arrays, image processing, RGB (red-green-blue) to gray-scale

Received: 4 December 2022
Revised: 5 February 2023
Accepted: 1 April 2023

Corresponding Author: Dr. Payam Sanaei

Citation: M. Ajamin-Hamedani, P. Sanaee, "Converting RGB images to gray-scale by the weighted average method, based on shift-and-add technique on the combination of color components for reducing the computational units and errors in FPGA", Journal of Intelligent Procedures in Electrical Technology, vol. 16, no. 63, pp. 61-82, December 2025 (in Persian).

https://dorl.net/dor/20.....
مقاله پژوهشی

تبدیل تصاویر رنگی RGB به خاکستری به روش میانگین‌گیری وزن‌دهی شده با استفاده از جابه‌جایی و جمعِ ترکیب مولفه‌های رنگی جهت کاهش واحدهای محاسباتی و خطا در تراشه‌های FPGA

مهدی عجمین‌همدانی[۱]، دانشجوی کارشناسی‌ارشد، پیام سنائی[۱،۲]، استادیار

۱- دانشکده مهندسی برق، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد، ایران
۲- مرکز تحقیقات پردازش دیجیتال و بینایی ماشین، واحد نجف‌آباد، دانشگاه آزاد اسلامی، نجف‌آباد، ایران
mahdi.ajamin@gmail.com, p.sanaee@pel.iaun.ac.ir

چکیده: تبدیل تصاویر رنگی RGB به خاکستری یکی از مسائل مهم و اساسی در حوزهٔ پردازش تصویر بوده، از این‌رو روش‌های متعددی برای پیاده‌سازی سخت‌افزاری الگوریتم‌های مربوطه ارائه گردیده است. الگوریتم‌های تبدیل تصاویر رنگی به خاکستری در بخش پیش‌پردازش اکثر برنامه‌های بینایی ماشین، جهت تشخیص چهره و اشیاء به‌کار برده می‌شوند. در اکثر مسائل مرتبط با بازشناسی تصاویر، اطلاعات از طریق دوربین‌های تصویربرداری دیجیتال دریافت شده، از این‌رو نرخ داده‌های ورودی به سامانهٔ سخت‌افزاری بسیار بالا است. بنابراین برای پردازش بی‌درنگ و محاسبهٔ در لحظهٔ الگوریتم‌های به‌کار برده شده، نیازمند سرعت بخشیدن به محاسبات هستیم. یکی از راه‌کارهای سخت‌افزاری برای انجام سریع این محاسبات، استفاده از تراشه‌های مجتمع منطقی برنامه‌پذیر (FPGA) است. از مزایای این تراشه‌ها، امکان پیاده‌سازی سخت‌افزاری الگوریتم‌های محاسباتی به‌صورت پردازش موازی، همروند و مدارهای منطقی تمام ترکیبی است. در این مقاله برای کاهش خطای محاسباتی از سیستم عددی ممیز ثابت استفاده شده و مصالحه‌ای بین دقت و تعداد بلوک‌های منطقی به‌کار رفته ایجاد شده است. این امر به مدیریت منابع سخت‌افزاری کمک بسزایی می‌کند. همچنین روش‌های مختلفی برای تبدیل تصاویر رنگی به خاکستری روی تراشه‌های FPGA ارزان قیمت طراحی شده، و نتایج با یکدیگر مقایسه شده‌اند. با استفاده از روش ترکیب مولفه‌های رنگی در محاسبات اعشاری ممیز ثابت (ضرایب مولفه‌ها ۸ یا ۱۵بیت اعشار و محاسبات ۸ بیت اعشار) شاخص خطای میانگین مربعات (MSE) در تصویر خاکستری شدهٔ لنا ۵۱۲×۵۱۲ برابر با ۰/۰۱۸۴ گشت و برای پیاده‌سازی سخت‌افزار متناظر، ۱۰۵ بلوک منطقی (LB) به‌کار گرفته شده است.

کلمات کلیدی: اعداد ممیز ثابت، پردازش تصویر، تبدیل تصاویر رنگی به تصاویر خاکستری، تراشه‌های مجتمع منطقی برنامه‌پذیر، الگوریتم‌های محاسباتی

تاریخ ارسال مقاله: ۱۴۰۱/۹/۱۳
تاریخ بازنگری مقاله: ۱۴۰۱/۱۱/۱۶
تاریخ پذیرش مقاله: ۱۴۰۲/۱/۱۲

نام نویسنده‌ی مسئول: دکتر پیام سنائی
نشانی نویسنده‌ی مسئول: نجف‌آباد- بلوار دانشگاه- دانشگاه آزاد اسلامی واحد نجف‌آباد- دانشکده مهندسی برق

۱- مقدمه

علم پردازش تصویر در چند دههٔ اخیر از هر دو جنبهٔ نظری و عملی پیشرفت‌های چشم‌گیری داشته است. سرعت این پیشرفت‌ها به اندازه‌ای بوده است که هم اکنون، به راحتی می‌توان ردپای علم پردازش تصویر را در بسیاری از علوم و صنایع مشاهده نمود. بعضی از این کاربردها آنچنان به پردازش تصویر وابسته هستند که بدون آن، اساساً قابل استفاده نیستند. الگوریتم‌های پردازش تصویر بخش مهم و اصلی را در سامانه‌های پردازشی و برنامه‌های کاربردی ایفا می‌کنند. کاربرد این سامانه‌ها در زمینه‌هایی مانند تصویربرداری پزشکی، سنجش از راه دور، تصویربرداری میکروسکوپی، سامانه‌های نظامی مانند کنترل موشک‌ها و پهپادها است. همچنین شناسائی و تشخیص اشیاء، یکی از مهم‌ترین کاربردهای علم پردازش تصویر است. در طراحی الگوریتم‌های پردازش تصویر، پیاده‌سازی سخت‌افزاری این الگوریتم‌ها به‌صورت بی‌درنگ و بر خط از اهمیت و محبوبیت بیشتری برخوردار است. در اغلب مواقع تصاویر ورودی به سامانه‌ها به‌صورت RGB بوده که بر پایهٔ سه رنگ اصلی قرمز، سبز و آبی هستند. بدیهی است که سایر رنگ‌ها با ترکیب شدت‌های مختلفی از این سه رنگ تولید می‌شوند. از آنجایی که هر پیکسل (عنصر تصویری) در تصاویر رنگی شامل سه مولفه بوده، حجم داده‌های ورودی افزایش می‌یابد و در نتیجه محاسبات مورد نیاز الگوریتم‌های پردازش تصویر و بینایی ماشین بسیار پیچیده، حجیم و زمان‌بر خواهند بود. علاوه بر این بسیاری از الگوریتم‌های محاسباتی در علم پردازش تصویر فقط روی تصاویر خاکستری تعریف شده‌اند [۱] . برای حل چنین مسائلی در ابتدا تصاویر RGB به تصاویر خاکستری تبدیل شده و سپس پردازش‌های مورد نیاز روی تصاویر خاکستری انجام می‌شوند. در تصاویر خاکستری هر پیکسل منحصراً از سایه‌های خاکستری تشکیل شده و عموماً شدت روشنایی آن توسط یک عدد هشت بیتی شمارشی بدون علامت بیان می‌گردد. از این‌رو شدت روشنایی هر پیکسل در این تصاویر می‌تواند از مشکی (۰) تا سفید (۲۵۵) تغییر نماید. در مرجع [۲]، راه‌کارهای مختلف تبدیل تصاویر RGB به خاکستری بیان شده‌اند که مهم‌ترین آنها تک کاناله[۱]، روشنایی[۲]/اشباع زدایی[۳]، تجزیه[۴]، میانگین‌گیری[۵] و میانگین‌گیری وزن‌دهی شده[۶]/ درخشندگی[۷] هستند. البته روش‌های دیگری نیز برای تبدیل تصاویر RGB به خاکستری وجود دارد که نه تنها از اطلاعات مولفه‌های رنگی بلکه از اطلاعات مکانی و همسایگی هر پیکسل نیز استفاده می‌کنند [۵-۳]. کیفیت بصری تصاویر خاکستری شده به روش میانگین‌گیری وزن‌دهی شده بهتر از سایر روش‌ها بوده و برای ادراک انسان مناسب‌تر است [۲]. از این‌رو اکثر پژوهشگران این روش را برای تبدیل تصاویر رنگی به خاکستری انتخاب می‌کنند. در بخش دوم این مقاله تعدادی از این روش‌ها را به‌صورت کامل توضیح داده شده‌اند.

در بسیاری از مسائل با حجم زیادی از اطلاعات بُرداری مواجه هستیم که باید به‌صورت بی‌درنگ و برخط پردازش شوند. در اغلب موارد برای حل این‌گونه مسائل، یکی از چهار رَوند سخت‌افزاری (الف) پردازشگرهای چند هسته‌ای[۸] مانند پردازندهٔ مبتنی بر معماری ماشین ریسک پیشرفته[۹] (ARM) چند هسته‌ای مانند Apple M1 Ultra، (ب) پردازشگرهای ویژهٔ سیگنال‌های دیجیتال[۱۰] مانند پردازشگر تصویر TMS 320C6000، (ج) پردازشگرهای گرافیکی[۱۱] مانند GeForce RTX 3090 Ti و (د) تراشه‌های مدار مجتمع با کاربرد خاص[۱۲] (ASIC) و مدارهای مجتمع منطقی برنامه‌پذیر[۱۳] (FPGA) مانند Artix-7 توصیه می‌شود.

از میان چهار راه‌کار فوق، مدارهای مجتمع منطقی برنامه‌پذیر از محبوبیت بیشتری در بین پژوهشگران برخوردار هستند. این امر به دلیل امکان حل مساله، توسط پیاده‌سازی انواع معماری‌های سخت‌افزاری مختلف جهت شتاب بخشیدن به امر محاسبات، مانند خط لوله، پردازش موازی یک دستورالعمل، چند داده[۱۴] (SIMD)، پیاده‌سازی سامانه‌های دیجیتال چند هسته‌ای، ترکیبی و همروند است [۶-۱۷]. سایر موارد منجر به راه‌کارهای نرم‌افزاری شده که مبتنی بر اجرای برنامه (واکشی، رمزگشایی و اجرای دستور) هستند. اجرای چنین برنامه‌هایی نه تنها زمان‌بر بوده بلکه نیاز به سخت‌افزارهایی با توان مصرفی بالا هم دارد. از این‌رو پیاده‌سازی الگوریتم‌ها به‌صورت سخت‌افزاری مبتنی بر مدارهای ASIC/FPGA مناسب‌تر هستند. از این میان، تراشه‌های FPGA به دلیل قابلیت پیکربندی مجدد، سادگی پیکربندی و زمان کوتاه پیکربندی متداول‌تر و محبوب‌تر هستند [۶-۹].

۱-۱- بیشینه تحقیق

در مرجع [۹] الگوریتمی جهت تبدیل تصاویر RGB به خاکستری با رویکرد محاسبات تقریبی و کاهش بهینهٔ توان مصرفی در

تراشهٔ FPGA Virtex5 xc5vlx110t2ff1136 ارائه شده که در راستای مصالحه‌ای بین انرژی مصرفی و کیفیت-دقت نتایج (E-Q) ایجاد شده و طراحی در سطح RTL بهینه‌سازی شده است. روند طراحی مبتنی بر مدارهای منطقی ترتیبی و سیستم عددی ممیز ثابت بنا نهاده شده و اساس طراحی مبتنی بر کاهش توان مصرفی در مدار جمع‌کننده است. براساس تعداد بیت اعشار محاسبات اتخاذ شده در الگوریتم، مدار جمع‌کننده تقریبی ارائه شده است.

در مرجع [۱۰] تبدیل تصاویر رنگی RGB به تصاویر خاکستری با استفاده از ۳ روش میانگین‌گیری، اشباع زدایی و میانگین‌گیری وزن‌دهی شده به کمک محاسبات شمارشی ۸ بیتی و روند جابه‌جایی و جمع به زبان وریلاگ[۱۵] روی تراشه‌های FPGA سری اسپارتان-۶ [۱۶] پیاده‌سازی شد. این پژوهش مبتنی بر بیت‌های منطقی برگشت‌پذیر بوده و در پیاده‌سازی این طرح از ۳۵ بلوک منطقی[۱۷] (LB) استفاده شده است. در مرجع [۱۱] برای تبدیل تصاویر RGB به خاکستری از روش میانگین‌گیری وزن‌دهی شده استفاده شد و طرح پیشنهادی مبتنی بر محاسبات شمارشی ۸ بیتی جمع و جابه‌جایی به‌صورت سخت‌افزاری روی تراشهٔ FPGA Artix-7 پیاده‌سازی شد. این طراحی مبتنی بر حداقل سخت‌افزار بوده و تعداد بلوک‌های منطقی به کار برده شده در این طراحی ۴۲ واحد است. اما در قبال کاهش حجم سخت‌افزار، خطای محاسباتی بسیار بالایی ایجاد شد. در مرجع [۱۲] یک الگوریتم سریع را برای تبدیل تصاویر RGB به خاکستری با استفاده از روش ضرب ۸ بیتی شمارشی و جمع در سیستم عددی ممیز ثابت با ۸ بیت اعشاری ارائه شد و این الگوریتم توسط تراشهٔ FPGA سری اسپارتان-۶ مدل xc6slx45-3csg484 پیاده‌سازی شد. این طراحی مبتنی بر ۳ ضرب‌کننده و ۲ جمع‌کننده بود و جهت کاهش توان مصرفی از ۳ تقریب در پیاده‌سازی ضرب‌کننده‌ها و از ۲ تقریب در پیاده‌سازی جمع‌کننده‌ها استفاده شده‌است. تعداد بلوک‌های منطقی به کار برده شده ۶۵۶ واحد است. بسیاری از پژوهشگران، راه‌کارهای مختلفی را برای تبدیل تصاویر RGB به خاکستری به روش میانگین‌گیری وزن‌دهی شده پیشنهاد کردند، اما تعداد کمی از آنها به‌صورت دقیق و با جزییات کامل نحوهٔ پیاده‌سازی الگوریتم‌های پیشنهادی را در سطح سخت‌افزار و VLSI مورد بحث قرار داده‌اند. این الگوریتم‌ها عموماً توسط نرم‌افزارهای سطح بالایی مانند متلب[۱۸] و ابزارهای سیمولینک[۱۹] طراحی شده و سپس توسط ابزار سیستم ژنراتور زایلینکس (کد کننده زبان‌های توصیف سخت‌افزار)[۲۰] کدها، مستقیماً بر روی تراشهٔ FPGA بارگزاری می‌شوند [۱۳-۱۷]. بدین ترتیب نرم‌افزار سنتزکننده بصورت مستقیم الگوریتم سطح بالا را در تراشهٔ FPGA قرار داده از این‌رو جزئیات هر واحد مشخص نیست. در مرجع [۱۷] الگوریتم میانگین‌گیری وزن‌دهی شده را با استفاده از نرم‌افزار سیمولینک متلب بر روی تراشهٔ FPGA به شمارهٔ Virtex-5 XUPV5-LX110T پیاده‌سازی شد. در این طرح پیشنهادی برای تبدیل مولفه‌های RGB هر نقطه به معادل خاکستری بصورت صریح از سه ضرب‌کننده و دو جمع‌کننده استفاده شده است.

در روش‌های تبدیل تصاویر RGB به خاکستری که نیازمند انجام عمل ضرب هستند، عموماً یکی از دو روند زیر استفاده می‌شود:

- استفاده از الگوریتم جابه‌جایی و جمع[۲۱]
- استفاده از عملیات ضرب (ضرب‌کننده مجتمع داخلی مانند DSP48 یا پیاده‌سازی ضرب‌کننده به‌صورت گسسته)

به‌طور معمول در تراشه‌های مجتمع منطقی برنامه‌پذیر ارزان قیمت، ضرب‌کننده بصورت مجتمع وجود ندارد و پیاده‌سازی آن به‌صورت گسسته با عملوندهایی با ابعاد بزرگ به حجم زیادی از بلوک‌های منطقی نیاز دارد که این امر خود باعث کاهش منابع محاسباتی آزاد و افزایش تاخیر انتقال می‌شود.

۲-۱- نوآوری تحقیق

در این مقاله برای تبدیل تصاویر RGB به خاکستری به روش میانگین‌گیری وزن‌دهی شده چهار الگوریتم اصلی به همراه سه الگوریتم بهینه شده در سیستم عددی ممیز ثابت ارائه شده است. به‌طور خلاصه برجستگی‌های مقاله را می‌توان به شرح زیر بیان کرد:

- استفاده از سیستم عددی ممیز ثابت جهت افزایش دقت محاسبات و کاهش خطا و بررسی تاثیر تعداد بیت‌های بخش اعشاری
- استفاده از روش جابه‌جایی و جمع برای پیاده‌سازی عمل ضرب مبتنی بر الگوریتم بوث[۲۲] جهت کاهش تعداد بلوک‌های منطقی
- استفاده از ترکیب مولفه‌های رنگی جهت کاهش تعداد محاسبات تکراری روی مولفه‌های مستقل رنگی

۳-۱- ساختار مقاله

ساختار مقاله در ادامه به این شرح است. در بخش دوم به بررسی چهار الگوریتم اصلی و تاثیر تعداد بیت‌های اعشاری اتخاذ شده در دقت محاسبات و حجم سخت‌افزار اشاره شده است. در بخش سوم به بررسی و مقایسهٔ نتایج خواهیم پرداخت. در بخش چهار مقایسه روش‌ها ارائه شده است. در نهایت در بخش پنجم نتیجه‌گیری بیان شده است. شایان ذکر است که برای شبیه‌سازی و سنجش عملکرد سخت‌افزار پیاده‌سازی شده بر روی تراشهٔ FPGA سری اسپارتان-۳ مدل xc3s50 از تصویر رنگی لنا ۵۱۲×۵۱۲ و زبان طراحی سخت‌افزار وریلاگ در بستر VIVADO استفاده شده است. کلیه کُدهای به‌کار برده شده موجود هستند[۲۳].

۲- روندهای متداول تبدیل تصاویر رنگی به خاکستری

در تمامی این روندها هدف تبدیل تصاویر رنگی RGB ۲۴ بیتی به تصاویر خاکستری ۸ بیتی بوده که به معنی نگاشت داده‌های واقع در فضای اطلاعاتی سه بعدی ۸ بیتی شمارشی مثبت (فضای رنگی R,G,B) به اطلاعات ۸ بیتی شمارشی مثبت (فضای خاکستری)، است. در تمامی این الگوریتم‌ها از یک فرآیند اصلی سه مرحله‌ای استفاده می‌شود:

۱- دریافت مقادیر مولفه‌های قرمز، سبز و آبی هر پیکسل

۲- استفاده از توابع ریاضی مربوطه برای تبدیل مقادیر مولفه‌های قرمز، سبز و آبی هر پیکسل به مقدار خاکستری متناظر

۳- جایگزینی مقادیر اصلی قرمز، سبز و آبی با مقدار خاکستری

روش تک کاناله ساده‌ترین و سریع‌ترین روش محاسباتی برای تبدیل عکس‌های رنگی به خاکستری است. در این روش، نیازی به محاسبه نیست. تنها کاری که باید انجام شود این است که یکی از کانال‌های رنگی انتخاب گردد و به عنوان تصویر خاکستری معرفی شود. یکی از رابطه‌های (۱) الی (۳) برای تبدیل تصویر رنگی به خاکستری استفاده می‌شود.

$$Gray = R \quad (۱)$$

$$Gray = G \quad (۲)$$

$$Gray = B \quad (۳)$$

روش دیگر تجزیه است که در این روش مقدار حداقل و یا حداکثر مولفه‌های قرمز، سبز و آبی برای به‌دست آوردن مقدار متناظر خاکستری برای هر پیکسل لحاظ می‌گردند. یکی از رابطه‌های (۴) یا (۵) برای تبدیل تصویر رنگی به خاکستری استفاده می‌شود.

$$Gray = Max(R,G,B) \quad (۴)$$

$$Gray = Min(R,G,B) \quad (۵)$$

روش بعدی، الگوریتم روشنایی/اشباع زدایی است. در این روش برای دستیابی به تصویر خاکستری از کنترل اشباع شدن رنگ استفاده شده، به این‌صورت که میانگین حداکثر و حداقل مقدار مولفه‌ها به عنوان مقدار خاکستری متناظر محاسبه می‌گردد. این روش غیرخطی بوده و مبتنی بر رابطهٔ (۶) است.

$$Gray = (R*0.21+G*0.71+B*0.07) \quad (۶)$$

متداول‌ترین الگوریتم تبدیل تصاویر RGB به خاکستری روش میانگین‌گیری است. در این شیوه مقدار متوسط مولفه‌های R، G و B هر پیکسل به عنوان مقدار خاکستری متناظر لحاظ می‌گردد [رابطهٔ (۷)]. این الگوریتم یک روش بسیار ساده و سریع بوده اما در نمایش سایه‌های خاکستری دچار ضعف است [۲،۱۱].

$$Gray = \frac{R + G + B}{3} \quad (۷)$$

روش بعدی الگوریتم میانگین‌گیری وزن‌دهی شده یا درخشندگی بوده که روش پیچیده‌تری نسبت به روش متوسط‌گیری است. در این الگوریتم میانگین مولفه‌های رنگی هر پیکسل با ضرایب غیریکنواختی محاسبه می‌شود. این روش برای ادراک انسان مناسب‌تر است. این الگوریتم بر اساس این واقعیت استوار است که تراکم سلول‌های مخروطی در شبکیهٔ چشم انسان یکسان نبوده و در چشم انسان حساسیت سلول‌های بینایی نسبت به رنگ سبز بیشتر از رنگ قرمز و رنگ قرمز بیشتر از رنگ آبی است. بنابراین الگوریتم میانگین‌گیری جهت تبدیل تصاویر رنگی به خاکستری مناسب نیست. در الگوریتم میانگین‌گیری وزن‌دهی شده به جای برخورد یکسان با مولفه‌های قرمز، سبز، آبی و انتخاب ضریب ۰/۳۳۳ برای هر مولفه، بر اساس میزان حساسیت چشم انسان،

ضرایب مناسبی انتخاب می‌شود [رابطه‌های (۸) و (۹)]. این رابطه‌ها مبتنی بر دانش رنگ‌سنجی[۲۴] بوده و توسط موسسهٔ کمیته استانداردهای تلویزیون ملی[۲۵] (NTSC) معرفی شده‌اند. در ابتدا رابطهٔ (۸) ابداع شده و سپس پس از اصلاحاتی رابطهٔ (۹) معرفی شد. در حال حاضر از رابطهٔ (۹) برای تبدیل تصاویر رنگی به خاکستری استفاده می‌شود.

$$Gray = (0.21*R+0.71*G+0.07*B) \quad (۸)$$

$$Gray = (0.30*R+0.59*G+0.11*B) \quad (۹)$$

با توجه به دقت و عملکرد مناسب الگوریتم میانگین‌گیری وزن‌دهی شده در این مقاله این الگوریتم انتخاب شده و از رابطهٔ (۹) استفاده می‌شود. از این‌رو به بررسی و طراحی سخت‌افزار بهینه و مناسب جهت پیاده‌سازی این الگوریتم اشاره می‌شود. روش‌های متعددی برای انجام محاسبات الگوریتم میانگین‌گیری وزن‌دهی شده وجود دارند که در این مقاله از روش تمام ترکیبی جابه‌جایی و جمع جهت رسیدن به حداقل تاخیر زمانی و هزینه استفاده می‌شود.

۳- روش‌های پیشنهادی پیاده‌سازی الگوریتم میانگین‌گیری وزن‌دهی شده مبتنی بر جابه‌جایی و جمع

در این روش‌ها، محاسبات ضرب اعشاری توسط عملیات جابه‌جایی و جمع پیاده‌سازی می‌شوند. نکتهٔ مهم در خصوص طراحی سخت‌افزاری این نوع از الگوریتم‌ها، بررسی میزان دقت محاسبات و تعداد واحدهای سخت‌افزاری به کار برده شده است. در این مقاله دقت محاسبات توسط سه معیار خطای میانگین مربعات[۲۶] (MSE)، حداکثر قدر مطلق خطا[۲۷] (MAX-ABSER) و درصد پیکسل‌های مغشوش[۲۸] سنجیده شده است. میزان دقت محاسبات و تعداد واحدهای سخت‌افزاری به کار برده شده اکیداً به ابعاد بیتی عملوندهای ورودی جمع‌کننده‌ها و تعداد واحدهای جمع‌کننده وابسته است.

در این پژوهش برای تبدیل تصاویر رنگی RGB به خاکستری از رابطهٔ (۹) استفاده می‌شود. در این رابطه، دامنهٔ مولفهٔ قرمز در ۰/۳۰، مولفهٔ سبز در ۰/۵۹ و مولفهٔ آبی در ۰/۱۱ ضرب شده و سپس مجموع حاصل ضرب‌ها محاسبه می‌شوند. در روش جابه‌جایی و جمع در ابتدا هرکدام از ضرایب را توسط رابطه‌های (۱۰) الی (۱۲) محاسبه می‌کنیم. در این رابطه‌ها پارامتر n بیانگر تعداد بیت اعشاری اتخاذ شده در سیستم عددی ممیز ثابت[۲۹] است. در جدول (۱) مقادیر باینری معادل ضرایب به ازای ۱ تا ۷ بیت اعشاری نمایش داده شده‌اند. در اینجا برای کاهش خطا در محاسبات اعشاری از روش گردکردن[۳۰] استفاده کردیم. بدیهی است با افزایش تعداد بیت‌های بخش اعشار، دقت بیان ضرایب افزایش یافته و در نتیجه حاصل محاسبات خطای کمتری خواهد داشت.

$$0.30 = \text{Round}\left(\frac{\text{int}(2^n * 0.30)}{2^n}\right) \quad (۱۰)$$

$$0.59 = \text{Round}\left(\frac{\text{int}(2^n * 0.59)}{2^n}\right) \quad (۱۱)$$

$$0.11 = \text{Round}\left(\frac{\text{int}(2^n * 0.11)}{2^n}\right) \quad (۱۲)$$

Table (1): The coefficients of the triple components of equ. (9) in fixed point form by rounding off n bits of fractional part

جدول (۱): ضرایب مولفه‌های سه‌گانه در رابطهٔ ۹ به‌صورت ممیز ثابت با n بیت اعشارگرد شده

تعداد ارقام اعشار	ضریب ۰/۳۰		ضریب ۰/۵۹		ضریب ۰/۱۱		مجموع ضرایب
	معادل باینری	مقدار واقعی	معادل باینری	مقدار واقعی	معادل باینری	مقدار واقعی	
۱	0.1B	۰/۵	0.1B	۰/۵	0.0B	۰	۱
۲	0.01B	۰/۲۵	0.10B	۰/۵	0.00B	۰	۰/۷۵
۳	0.010B	۰/۲۵	0.101B	۰/۶۲۵	0.001B	۰/۱۲۵	۱
۴	0.0101B	۰/۳۱۲۵	0.1001B	۰/۵۶۲۵	0.0010B	۰/۱۲۵	۱
۵	0.01010B	۰/۳۱۲۵	0.10011B	۰/۵۹۳۷۵	0.00100B	۰/۱۲۵	۱/۰۳۱۲۵
۶	0.010011B	۰/۲۹۶۸۷۵	0.100110B	۰/۵۹۳۷۵	0.000111B	۰/۱۰۹۳۷۵	۱
۷	0.0100110B	۰/۲۹۶۸۷۵	0.1001100B	۰/۵۹۳۷۵	0.0001110B	۰/۱۰۹۳۷۵	۱

۱-۳- روش پیشنهادی اول

در این روش، محاسبات بدون در نظر گرفتن بیت‌های اعشار برای مولفه‌های RGB انجام می‌شود. مولفه‌های RGB عناصر تصویری توسط اعداد شمارشی بدون علامت ۸ بیتی بیان می‌شوند؛ در نتیجه مولفه‌ها پس از ۸ بیت جابه‌جایی به سمت راست صفر می‌گردند. از این‌رو برای رسیدن به حداکثر دقت، ضرایب با ۷ رقم اعشار بیان می‌شود. در این راه‌کار ضرایب، اعداد اعشاری ۷ بیتی بوده اما اطلاعات شمارشی و بدون علامت هستند و محاسبات به‌صورت صحیح انجام می‌شوند [رابطه‌های (۱۳) و (۱۴)].

$$Gray=(0.30*R+0.59*G+0.11*B) \quad (۱۳)$$

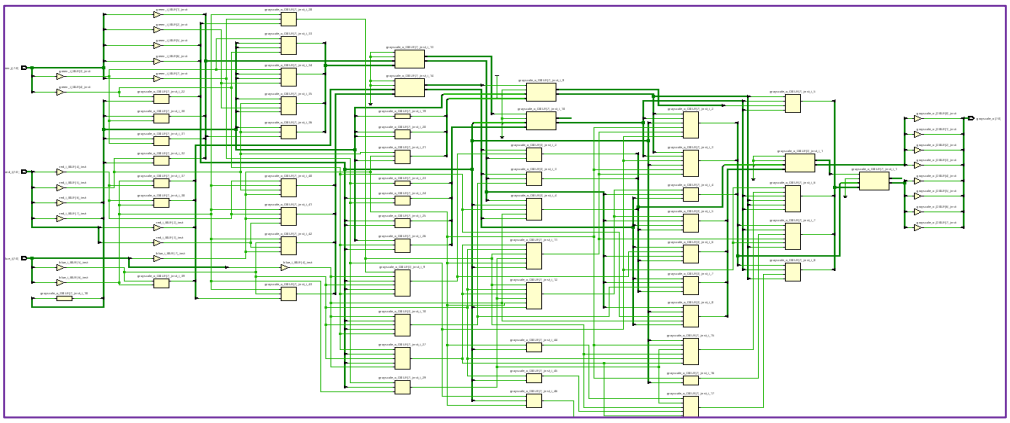
$$\text{if } n=7 \Rightarrow Gray = 0.0100110B*R+0.1001100B*G+0.0001110B*B$$

$$Gray = shr(R,2)+shr(R,5)+shr(R,6)+shr(G,1)+shr(G,4)+shr(G,5)+shr(B,4)+shr(B,5)+shr(B,6) \quad (۱۴)$$

در برنامهٔ وریلاگ نظیر الگوریتم فوق، برای جابه‌جایی به سمت راست، از جابه‌جا کننده‌های بشکه‌ای[۳۱] مبتنی بر مدارهای ترکیبی (مالتی پلکسر یا بافر سه حالته) استفاده شده و به هنگام جابه‌جایی، بیت‌های با ارزش کمتر مولفه‌های RGB دور ریخته می‌شوند. از آنجایی که ضرایب مولفه‌های RGB تصویری بصورت اعداد اعشاری ۷ بیتی بیان شده‌اند و از روش گردکردن استفاده شده است؛ این احتمال وجود دارد که به هنگام جمع‌کردن مولفه‌های جابه‌جا شدهٔ گرد شده، حاصل محاسبات بیشتر از عدد ۲۵۵ شود. این امر هنگامی روی می‌دهد که حاصل جمع ضرایب بیشتر از یک شود. به عنوان مثال هنگامی که ضرایب مولفه‌های رنگی ۵ بیتی اتخاذ شوند این مسئله مشاهده می‌شود [جدول (۱)]. از این‌رو قالب اطلاعات جابه‌جا شده را ۹ بیتی در نظر می‌گیریم. سپس در انتها حاصل محاسبات را که ۹ بیتی است به‌صورت ۸ بیتی در می‌آوریم. این امر، با مقایسهٔ حاصل محاسبات با عدد ۲۵۵ انجام می‌پذیرد. برای ساده‌تر شدن سخت‌افزار مربوطه، بیت با ارزش بیشتر حاصل محاسبات ۹ بیتی بررسی می‌شود. در صورتی که صفر باشد ۸ بیت با ارزش کمتر محاسبات ۹ بیتی به عنوان خروجی در نظر گرفته شده و در صورتی که یک باشد عدد ۲۵۵ به عنوان خروجی در نظر گرفته می‌شود (فرآیند اشباع شدن). کد وریلاگ نظیر به ازای n برابر با ۷ در ادامه آورده شده است. همچنین سخت‌افزار متناظر در شکل (۱) نمایش داده شده است.

```
// Methode 1
// RGB INETGER
// Coeficent fractional part 7 bit
module ShiftAndAdd_Mod2(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [7:0] R_N, G_N, B_N;
   wire [8:0] Sum_RGB;
   assign R_N = red_i;
   assign G_N = green_i;
   assign B_N = blue_i;
   assign Sum_RGB = {3'b000,   R_N[7:2]} + {6'b000000, R_N[7:5]} + {7'b0000000, R_N[7:6]} +
                    {2'b00,    G_N[7:1]} + {5'b00000,  G_N[7:4]} + {6'b000000,  G_N[7:5]}  +
                    {5'b00000, B_N[7:4]} + {6'b000000, B_N[7:5]} + {7'b0000000, B_N[7:6]} ;
   assign grayscale_o = (Sum_RGB[8] == 1) ? 8'b11111111 : Sum_RGB[7:0];
endmodule
```

برای سنجش عملکرد سخت‌افزارهای سنتز[۳۲] شدهٔ پیشنهادی، از معیارهای کمی خطای میانگین مربعات، حداکثر قدرمطلق خطا و درصد نقاط مغشوش استفاده شده است. این امر به دو صورت انجام می‌گیرد. در ابتدا تصویر معیار لنا ۵۱۲×۵۱۲ رنگی را توسط محاسبات ممیز شناور با استفاده از برنامهٔ نوشته شده به زبان پایتون[۳۳] به تصویر خاکستری تبدیل کرده و سپس این تصویر را با تصاویر خروجی ماژول‌های سخت‌افزاری پیشنهادی مقایسه می‌کنیم. در مرحلهٔ دوم کلیهٔ مقادیر مختلف مولفهٔ‌های RGB را به‌صورت جداگانه توسط سه حلقهٔ تو در تو به ماژول‌های سخت‌افزاری اعمال کرده و معادل خاکستری آنها را محاسبه می‌نماییم. سپس شاخص‌های ذکر شده را بین مقادیر محاسبه شده و واقعی به‌دست می‌آوریم. در جدول (۲) تعداد بلوک‌های منطقی به‌کار برده شده به‌ازای ۱ تا ۷ بیت اعشاری و مقادیر فوق نمایش داده شده‌اند. همچنین در شکل (آ-۲) تصاویر لنا اصلی رنگی و تبدیل شدهٔ آن به خاکستری توسط نرم‌افزار پایتون و خروجی واحدهای سخت‌افزاری به‌ازای ۱ تا ۷ بیت اعشاری نشان داده شده‌اند.



شکل (۱): سخت‌افزار متناظر با روش پیشنهادی اول به ازای ۷رقم اعشار

Figure (1): The hardware of the first proposed method for n=7

Table (2): The number of LB used in the first proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۲): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی اول و معیارهای خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد ارقام اعشار	تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
		خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش
۱	۷	۲۵۹/۵۱۲	۳۲	٪۹۹/۹۸	۳۲۹/۲۰۹	۵۲	٪۹۹/۰۰
۲	۶	۱۰۱۷/۵۸۴	۶۰	٪۱۰۰/۰۰	۱۱۸۰/۶۸۸	۶۵	٪۹۹/۹۰
۳	۱۴	۳۲/۱۰۲	۱۰	٪۹۹/۹۹	۲۴/۲۰۷	۱۵	٪۹۲/۹۵
۴	۲۸	۱/۶۱۰	۵	٪۷۱/۳۰	۱۰/۷۶۷	۱۱	٪۸۸/۴۰
۵	۲۷	۵/۱۷۰	۷	٪۸۵/۹۰	۵/۰۰۸	۸	٪۸۳/۸۰
۶	۳۴	۱۹/۵۶۰	۸	٪۹۹/۹۰	۱۷/۴۲۲	۹	٪۹۹/۹۰
۷	۳۴	۱۹/۵۶۰	۸	٪۹۹/۹۰	۱۷/۴۲۲	۹	٪۹۹/۹۰

شکل (۲): مقایسهٔ تصویر اصلی لنا با تصویر خاکستری شده توسط محاسبات ممیز شناور بر روی کامپیوتر و تصویر خاکستری خروجی ماژول سخت‌افزاری پیشنهادی اول به ازای بیت‌های اعشاری مختلف، آ) تصویر رنگی اصلی، ب) تصویر خاکستری شده توسط محاسبات ممیز شناور، پ) ۱ بیت اعشار، ت) ۲ بیت اعشار، ث) ۳ بیت اعشار، ج) ۴ بیت اعشار، چ) ۵ بیت اعشار، ح) ۶ بیت اعشار، خ) ۷ بیت اعشار

Figure (2): Comparison of original Lena image with the grayed image by floating point operation on the computer and output the first proposed hardware for different fractional bits, a) Original color image "Lenna", b) Grayscaled "Lenna"image using floating point operation, c) 1 bit for fractional part, d) 2 bit for fractional part, e) 3 bit for fractional part, f) 4 bit for fractional part, g) 5 bit for fractional part, h) 6 bit for fractional part, i) 7 bit for fractional part

در این روش با دو خطا مواجه هستیم. خطای اول ناشی از جابه‌جایی اطلاعات مولفه‌های RGB به سمت راست است. این امر باعث حذف شدن بیت‌های معنادار با ارزش کمتر این مولفه‌ها می‌شود. از این‌رو این خطا باعث کاهش مقدار حاصل ضرب خواهد شد. خطای دوم ناشی از تقریبی است که برای ضرایب مولفه‌های RGB داریم. اما از دلایل این که به ازای ۵ بیت اعشار معیار خطای میانگین مربعات کمینه است، تقریب رو به بالای ضرایب RGB است. این افزایش ضرایب جبران کاهش مقدار مولفه‌های RGB را می‌نماید. از آنجایی که به ازای ۲ بیت اعشار جمع ضرایب ۰/۷۵ است معیار خطای میانگین مربعات نسبت به هنگامی که ۱ بیت اعشار است به جای کاهش دچار افزایش چشم‌گیری می‌شوند. نکتهٔ مهم در خصوص طراحی به روش جابه‌جایی و جمع این است که باید دقت را از ۲ دید افزایش دهیم که عبارتند از: افزایش دقت محاسبات و افزایش دقت ضرایب. از آنجایی که در این روش به ازای هر یک بیت جابه‌جایی به سمت راست مقادیر مولفه‌های قرمز، سبز و آبی عناصر تصویری، بیت‌های با ارزش کمتر مولفه‌های تصویری دور ریخته می‌شوند. می‌توان نتیجه گرفت که معایب این روش عبارتند از:

۱ - از آنجایی که اعداد شمارشی بدون علامت ۸ بیتی، پس از ۸ بیت جابه‌جایی به سمت راست صفر می‌شوند؛ پس انتخاب بیش از ۷ بیت برای بخش اعشاری ضرایب مولفه‌ها تاثیری در دقت محاسبات نخواهد داشت.

۲ - با حذف شدن بیت‌های با ارزش کمتر ناشی از جابه‌جایی به سمت راست مولفه‌های تصویری خطایی در محاسبات ایجاد خواهد شد و این خطاها با یکدیگر جمع شده و افزایش می‌یابند.

از این‌رو برای برای بهینه‌سازی این الگوریتم جهت افزایش دقت محاسبات، پیشنهاد می‌شود تا از رابطهٔ (۱۵) به جای رابطهٔ (۱۴) استفاده شود. بدین ترتیب که در ابتدا، مولفه‌های تصویری به سمت چپ جابه‌جا شده آنگاه پس از انجام عمل جمع، حاصل جمع به سمت راست جابه‌جا شده و در انتها حاصل محاسبات گرد شود. از این‌رو در این روش بهبود یافته، خطای ایجاد شده ناشی از حذف شدن بیت‌های با ارزش کمتر مولفه‌های تصویری به دلیل جابه‌جایی اطلاعات به سمت راست، در سیستم عددی شمارشی ۸ بیتی مرتفع می‌شود. رابطهٔ (۱۷) جهت گردکردن عدد اعشاری است.

$$Gray = (0.30*R+0.59*G+0.11*B)$$

$$\text{if} \quad n = 7 \Rightarrow Gray = 0.0100110B*R+0.1001100B*G+0.0001110B*B$$

$$Gray = \left(0.0100110B*R+0.1001100B*G+0.0001110B*B\right)*2^{7}/2^{7}$$

$$Gray = \frac{\left(0100110B*R+1001100B*G+0001110B*B\right)}{2^{7}}$$

$$Gray = \text{shr}\left(\left(0100110B*R+1001100B*G+0001110B*B\right),7\right) \qquad (۱۵)$$

$$Temp = \text{shl}(R,1)+\text{shl}(R,2)+\text{shl}(R,5)+\text{shl}(G,2)+\text{shl}(G,3)+\text{shl}(G,6)+\text{shl}(B,1)+\text{shl}(B,2)+\text{shl}(B,3) \qquad (۱۶)$$

$$Gray = \text{shr}\left(Temp,7\right)+Temp_{6} \qquad (۱۷)$$

کُد وریلاگ مانند سخت‌افزار متناظر با روش پیشنهادی اول بهینه شده و به ازای ۷ بیت اعشار در ادامه آورده شده است. همچنین سخت‌افزار متناظر در شکل (۳) نمایش داده شده است. همچنین در جدول (۳) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش نشان داده شده است. در انتها در شکل (۴) تصویر خروجی لنا برای بررسی کیفی نمایش داده شده است.

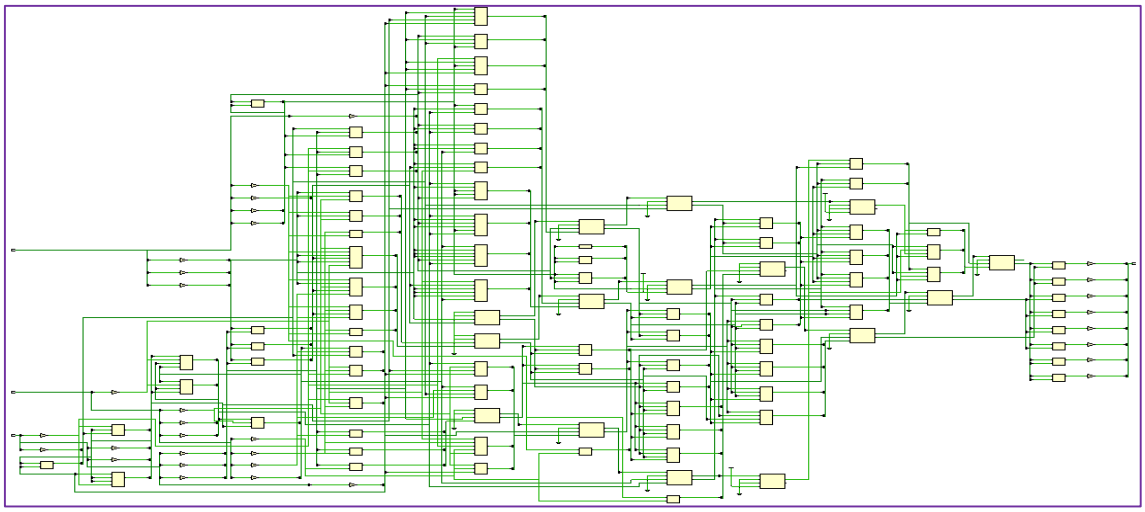
```
//  Methode 1 optimized
//  RGB INTEGER
//  Coeficent fractional part 7 bit
module ShiftAndAdd_ModRGB_7bit_V2(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [7:0] R_N, G_N, B_N;
   wire [8:0] Sum_RGB;
   wire [15:0] Temp;
   assign R_N  = red_i;
   assign G_N  = green_i;
   assign B_N  = blue_i;
```

```
    assign Temp = {7'b0000000, R_N, 1'b0}  + {6'b000000, R_N, 2'b00}  + {3'b000, R_N, 5'b00000} +
                  {6'b000000,  G_N, 2'b00} + {5'b00000,  G_N, 3'b000} + {2'b00,  G_N, 6'b000000}+
                  {7'b0000000, B_N, 1'b0}  + {6'b000000, B_N, 2'b00}  + {5'b00000, B_N, 3'b000};
    assign Sum_RGB = Temp[15:7];
    assign grayscale_o = (Sum_RGB[8] == 1) ? 8'b11111111 : Sum_RGB[7:0];
endmodule
```

با مقایسهٔ جدول‌های (۲) و (۳) درمی‌یابیم که با درنظر گرفتن بیت‌های با ارزش کمتر مولفه‌های RGB به ازای ۷ بیت اعشار، در الگوریتم بهینه شده، شاخص‌های خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های معیوب درصد پیکسل‌های مغشوش به‌صورت چشم‌گیری ارتقا می‌یابند.

۲-۳- روش پیشنهادی دوم

اما راه‌کار دیگر برای ارتقای نتایج، استفاده از سیستم عددی ممیز ثابت برای بیان مولفه‌های RGB عناصر تصویری است. در اینجا با افزودن بیت‌های اعشاری به مولفه‌های RGB عناصر تصویری، دیگر اطلاعات ۸ بیتی بخش شمارشی مولفه‌ها پس از جابه‌جایی به سمت راست دور ریخته نمی‌شوند بلکه وارد قسمت اعشاری خواهند شد و در نتیجه خطای محاسبات کاهش پیدا می‌نماید.



شکل (۳): سخت‌افزار متناظر با روش پیشنهادی اول بهینه شده و ۷ رقم اعشار

Figure (3): The hardware of the optimized first proposed method for n=7

Table (3): The number of LB used in the optimized first proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۳): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی اول بهینه شده و خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد ارقام اعشار	تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
		خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش
۷	۵۴	۰/۷۵۲	۱	٪۷۵/۲۰	۰/۵۴۵	۲	٪۵۰/۹۰

شکل (۴): تصویر خاکستری خروجی ماژول سخت‌افزاری نظیر روش پیشنهادی اول بهینه شده به ازای ۷ رقم اعشار

Figure (4): The output grayscale image of the optimized first proposed hardware module for n=7

بدیهی است برای حفظ اطلاعات تصویری، مناسب است به گونه‌ای تعداد بیت‌های بخش اعشاری را انتخاب نماییم تا پس از جابه‌جایی به سمت راست هیچ بیت ارزشمندی حذف نشود. از این‌رو تعداد بیت‌های بخش اعشاری ضرایب بیانگر تعداد بیت بخش اعشار خود مولفه‌ها خواهند بود. به این ترتیب باعث افزایش ابعاد عملوندهای ورودی تمام جمع‌کننده‌ها خواهیم شد و در نتیجه تعداد بلوک‌های منطقی در تراشهٔ FPGA افزایش می‌یابد. اکنون تاثیر تعداد بیت‌های قسمت اعشاری (سیستم عددی ممیز ثابت اتخاذ شده مولفه‌های تصویری) در میزان دقت و مقدار سخت‌افزار به کار برده شده را برای رابطهٔ (۱۸) بررسی می‌نماییم. از آن جایی که تعداد بیت‌های اعشاری را در سیستم‌های ممیز ثابت انتخابی از ۱ تا ۸ تغییر می‌دهیم؛ این امکان وجود دارد که بدون صفر شدن بتوانیم مقدار مولفه‌های RGB عناصر تصویری را تا ۸ بیت هم به سمت راست جابه‌جا نماییم. از این‌رو برای رسیدن به دقت بالا و مقایسه با روش پیشنهادی اول مقدار ضرایب مولفه‌های تصویر را بصورت ۸ بیت اعشار در رابطهٔ (۱۸) در نظر می‌گیریم.

$$Gray = (0.30*R+0.59*G+0.11*B)$$

$$\text{if} \quad n = 8 \quad \Rightarrow \quad Gray = 0.01001101B*R+0.10010111B*G+0.00011100B*B \qquad (۱۸)$$

برای پیاده‌سازی رابطهٔ (۱۸) با قالب ممیز ثابت ۸ بیت صحیح و ۸ بیت اعشار برای مولفه‌های RGB عناصر تصویری از برنامه وریلاگ زیر استفاده نمودیم.

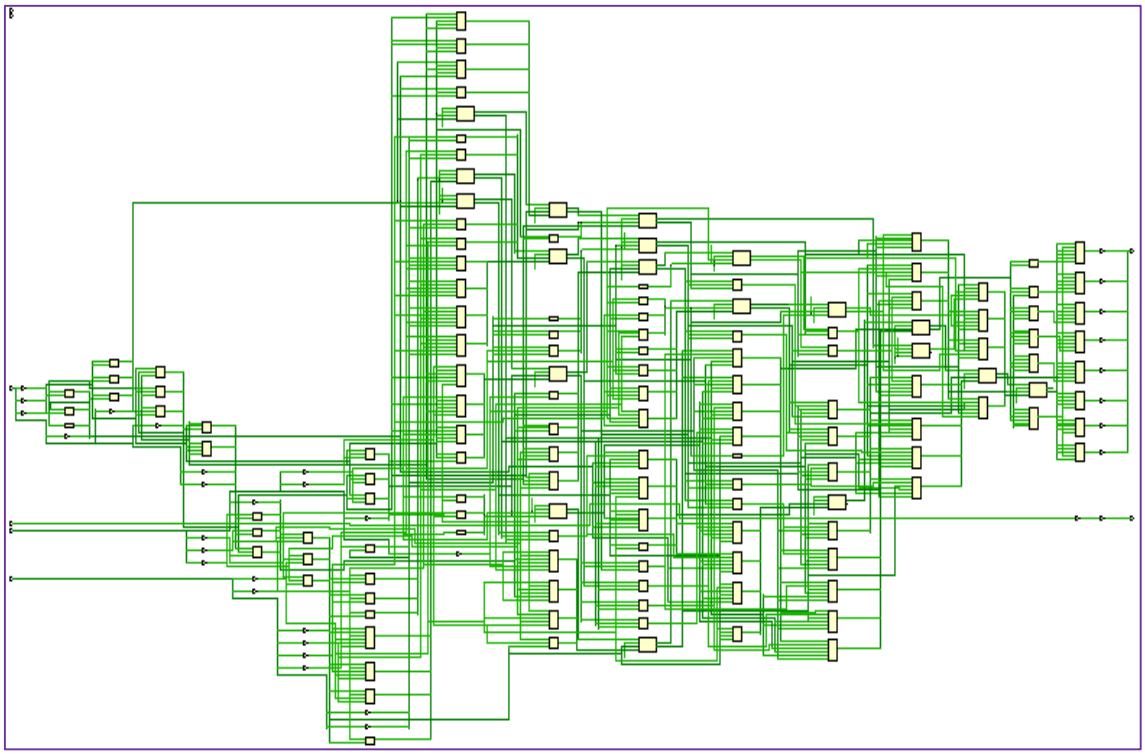
```
// Methode 2
// RGB INTEGER
// Components and Coefficent fractional part 8 bit
module ShiftAndAdd_ModRGB_8bit_V3(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [15:0] R_N, G_N, B_N;
   wire [16:0] Sum_RGB;
   assign R_N ={red_i, 8'b00000000};
   assign G_N ={green_i, 8'b00000000};
   assign B_N ={blue_i, 8'b00000000};
   assign Sum_RGB = {3'b000,R_N[15:2]}+ {6'b000000,R_N[15:5]}+ {7'b0000000,R_N[15:6]}          +
                    {9'b000000000,R_N[15:8]}+{2'b00,G_N[15:1]}+{5'b00000,G_N[15:4]}          +
                    {7'b0000000,G_N[15:6]}+{8'b00000000,G_N[15:7]}+{9'b000000000,G_N[15:8]}  +
                    {5'b00000,B_N[15:4]}+{6'b000000,B_N[15:5]}+{7'b0000000,B_N[15:6]};
endmodule
```

در شکل (۵) سخت‌افزار معادل برنامهٔ بالا نمایش داده شده است. همچنین در جدول (۴) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی دوم و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش با در نظر گرفتن ۰ تا ۸ بیت اعشار برای مولفه‌های RGB نشان داده شده است. در انتها در شکل‌های (پ-۶) الی (ذ-۶) تصاویر خروجی لنا به ازای تعداد بیت اعشاری مختلف برای بررسی کیفی نمایش داده شده است.

همان‌گونه که در جدول (۴) مشاهده می‌شود، با افزایش تعداد بیت‌های بخش اعشاری، میزان خطا کاهش یافته اما تعداد واحدهای سخت‌افزاری مورد استفاده نیز بیشتر شده است. همچنین مشاهده می‌شود هنگامی که تعداد بیت‌های بخش اعشاری بیش از ۳ بیت شود، مقدار حداکثر مقدار خطا ۱ واحد می‌گردد. بنابراین با افزایش تعداد بیت‌های اعشار به بیش از ۳ بیت، معیار خطای میانگین مربعات کاهش یافته و این خود به معنای کاهش تعداد نقاطی است که دارای خطا در محاسبات هستند.

نکتهٔ بعدی این که انتظار می‌رفت با افزایش تعداد بیت‌های بخش اعشاری در مولفه‌های تصویری میزان خطا در محاسبات توسط سخت‌افزار کاهش بیابد که این امر هنگامی که متوسط خطا برای کلیه تصاویر اندازه‌گیری شده مشاهده می‌شود؛ اما در تصویر لنا ۵۱۲×۵۱۲ رنگی به دلیل عدم توزیع یکنواخت اطلاعات تصویری و عدم وجود برخی از رنگ‌ها این روند مشاهده نمی‌شود. برای کاهش سخت‌افزار به کار برده شده در این روش، پیشنهاد می‌شود که ابعاد عملوندهای ورودی تمام جمع‌کننده‌های به‌کار برده شده را ۸ بیتی درنظر بگیریم. بدین ترتیب که محاسبات را از بیت‌های با ارزش کمتر ضرایب مولفه‌ها آغاز کنیم. این مسئله در رابطهٔ (۱۹) و جدول (۵) نشان داده شده است.

$$2*x+y = shl(x,1)+y=\{shl(x+shr(y,1)),y_0\} \qquad (۱۹)$$



شکل (۵): سخت‌افزار متناظر با روش پیشنهادی دوم با در نظر گرفتن ۸ بیت اعشار برای مولفه‌های RGB

Figure (5): The hardware of the second proposed method, considering 8 bits for fractional part of RGB components

Table (4): The number of LB used in the second proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۴): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی دوم و معیارهای خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد ارقام صحیح	تعداد ارقام اعشار	تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
			خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش
۸	۰	۳۸	۳۱/۷۷۶۰	۱۰	٪۱۰۰/۰۰	۳۲/۰۰۹۳	۱۱	٪۹۹/۹۰
۸	۱	۴۲	۵/۵۰۵۰	۵	٪۹۹/۷۰	۵/۶۰۸۳	۵	٪۹۹/۶۰
۸	۲	۶۹	۱/۱۱۶۰	۲	٪۸۴/۸۰	۱/۲۴۵۶	۳	٪۸۸/۱۰
۸	۳	۷۶	۰/۳۸۲۰	۱	٪۳۸/۲۰	۰/۴۴۰۵	۵	٪۴۴/۱۰
۸	۴	۸۶	۰/۱۰۴۰	۱	٪۱۰/۴۰	۰/۱۶۲۴	۱	٪۱۶/۲۰
۸	۵	۹۳	۰/۰۲۹۶	۱	٪۲/۹۶	۰/۰۷۵۷	۱	٪۷/۶۰
۸	۶	۹۹	۰/۰۴۹۵	۱	٪۴/۹۵	۰/۰۶۲۵	۱	٪۶/۲۰
۸	۷	۱۰۳	۰/۰۵۶۰	۱	٪۵/۶۹	۰/۰۶۱۷	۱	٪۶/۲۰
۸	۸	۱۰۴	۰/۰۵۸۷	۱	٪۵/۸۷	۰/۰۶۱۷	۱	٪۶/۲۰

Table (5): Reducing the bit size of full adders input oprands

جدول (۵): نحوه کاهش ابعاد عملوند جمع‌کننده‌ها

9bit-FullAdder

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	0	+
0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0	
K_8	K_7	K_6	K_5	K_4	K_3	K_2	K_1	K_0	

8bit-FullAdder

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	+
0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	
S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0	Y_0

شکل (۶): مقایسهٔ تصویر اصلی لنا با تصویر خاکستری شده توسط محاسبات ممیز شناور بر روی کامپیوتر و تصویر خاکستری خروجی ماژول سخت‌افزاری پیشنهادی دوم به ازای بیت‌های اعشاری مختلف، آ) تصویر رنگی اصلی، ب) تصویر خاکستری شده توسط محاسبات ممیز شناور، پ) ۱ بیت اعشار، ت) ۲ بیت اعشار، ث) ۳ بیت اعشار، ج) ۴ بیت اعشار، چ) ۵ بیت اعشار، ح) ۶ بیت اعشار، خ) ۷ بیت اعشار، د) ۸ بیت اعشار

Figure (6): Comparison of original Lena image with the grayed image by floating point operation on the computer and output the second proposed hardware for different fractional bits, a) Original color image "Lenna", b) Grayscaled "Lenna"image using floating point operation, c) 1 bit for fractional part, d) 2 bit for fractional part, e) 3 bit for fractional part, f) 4 bit for fractional part, g) 5 bit for fractional part, h) 6 bit for fractional part, i) 7 bit for fractional part, j) 8 bit for fractional part

برنامه وریلاگ روش پیشنهادی دوم بهینه شده به شرح زیر بوده و سخت‌افزار معادل در شکل (۷) نمایش داده شده است. همچنین در جدول (۶) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی سوم و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش نشان داده شده است.

```
//  Methode 2 optimized
//  RGB INTEGER
//  Components and Coefficent fractional part 8 bit
module ShiftAndAdd_ModRGB_8bit_V4(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [8:0] t1, t2, t3, t4, t5, t6, t7;
   wire [9:0] s1, s2, s3;
   wire [16:0] tempx;
   wire [10:0] tempy;
   assign t1            = {1'b0, red_i} + {1'b0, green_i};
   assign tempx[0]      = t1[0];
   assign t2            = {1'b0, green_i} + {1'b0, t1[8:1]};
   assign tempx[1]      = t2[0];
   assign t3            = {1'b0, blue_i} + {1'b0, t2[8:1]};
   assign s1            = {1'b0, t1} + {1'b0, t3};
   assign tempx[2]      = s1[0];
   assign t4            = {1'b0, red_i} + {1'b0, blue_i};
   assign s2            = {1'b0, t4} + {1'b0, s1[9:1]};
   assign tempx[3]      = s2[0];
   assign t5            = {1'b0, green_i} + {1'b0, blue_i};
   assign s3            = {1'b0, t5} + {1'b0, s2[9:1]};
```

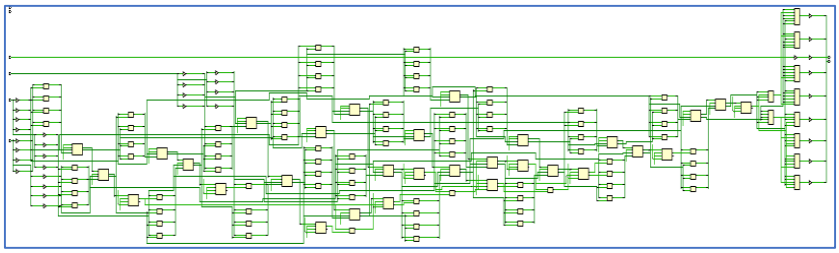
```
  assign tempx[4]     = s3[0];
  assign tempx[5]     = s3[1];
  assign t6           = {1'b0, red_i} + {1'b0, s3[9:2]};
  assign tempx[6]     = t6[0];
  assign t7           = {1'b0, green_i} + {1'b0, t6[8:1]};
  assign tempx[15:7]  = t7;
  assign tempx[16]    = 1'b0;
endmodule
```

۳-۳- روش پیشنهادی سوم

در این روش، فرض بر این است که حاصل ضرب هر مولفهٔ RGB در ضریب متناظر توسط سیستم عددی ممیز ثابت با ۸ بیت صحیح و ۸ بیت اعشاری بیان شده باشد. از این‌رو می‌توان مولفه‌های RGB را حداکثر تا ۱۵ بیت به سمت راست جابه‌جا نمود. از این‌رو برای بهبود دقت محاسبات می‌توان ضرایب مولفه‌های RGB را تا ۱۵ رقم اعشار درنظر گرفت. در اینجا باید خاطر نشان شد که با توجه به اینکه مقدار مولفه‌های RGB ۸ بیتی و شمارشی است. از این‌رو تا ۸ بیت جابه‌جایی به سمت راست مولفه‌ها هیچ بیت با ارزشی دور ریخته نمی‌شود؛ اما هنگامی که مقدار جابه‌جایی به سمت راست بیش از ۹ بیت شود بیت‌های با ارزش کمتر بخش شمارشی دور ریخته شده و این امر خود باعث ایجاد خطای محاسباتی خواهد شد [شکل (۸)]. به این ترتیب محاسبات توسط رابطهٔ (۲۰) انجام می‌پذیرد. در این روش پس از جابه‌جا کردن بیتی اطلاعات مولفه‌های رنگی به سمت راست در اثر ضرب در بیت‌های ضرایب با وزن $۲^{-۹}$ تا $۲^{-۱۶}$ تعدادی از بیت‌های با ارزش کمتر مولفه‌های تصویری دور ریخته می‌شوند.

$$Gray = (0.30*R+0.59*G+0.11*B)$$

$$\text{if} \quad n = 15 \quad \Rightarrow \quad Gray = 0.010011001100110B*R+0.100101110000101B*G+0.000111000010100B*B \qquad (۲۰)$$



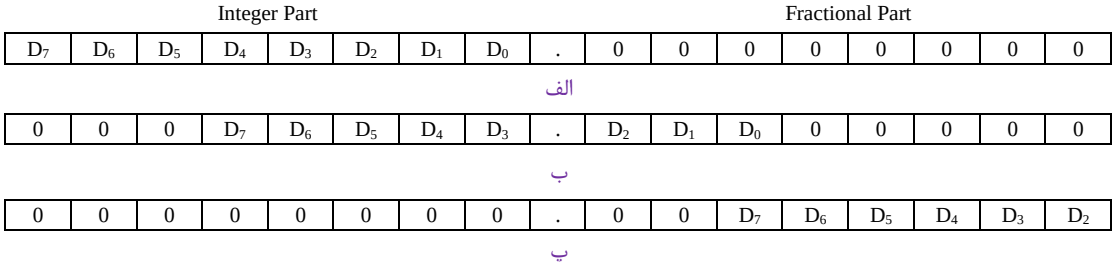
شکل (۷): سخت‌افزار متناظر با روش پیشنهادی دوم بهینه شده

Figure (7): The hardware of the optimized second proposed method

Table (6): The number of LB used in the optimized second proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۶): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی دوم بهینه شده و خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
	درصد پیکسل‌های مغشوش	حداکثر قدرمطلق خطا	خطای میانگین مربعات	درصد پیکسل‌های مغشوش	حداکثر قدرمطلق خطا	خطای میانگین مربعات
۹۱	٪۵/۹۰	۱	۰/۰۵۹	٪۶/۲۰	۱	۰/۰۶۲

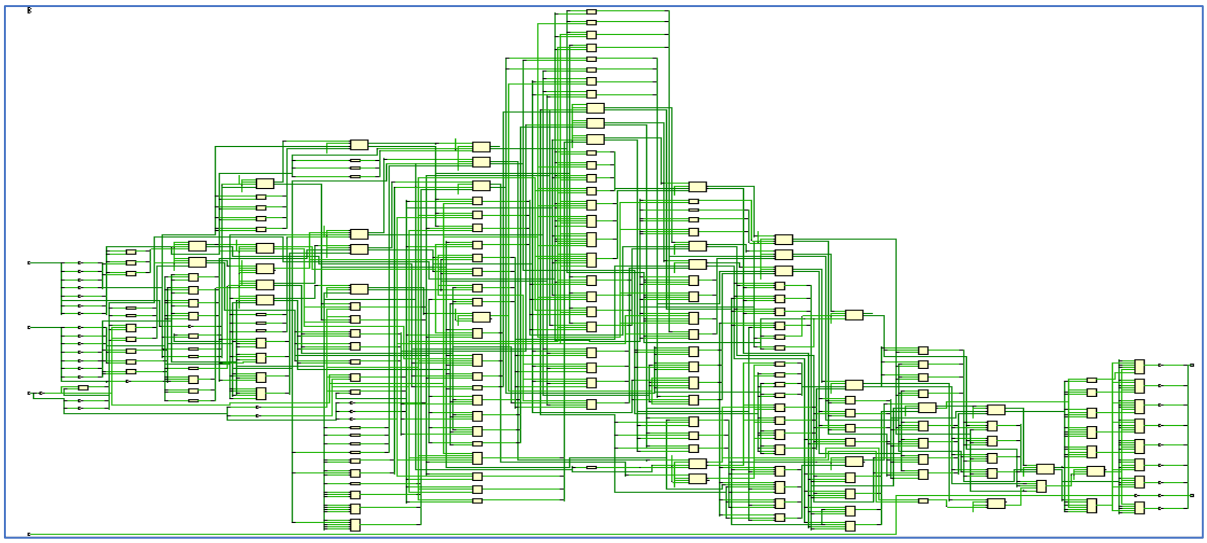


شکل (۸) جابه‌جایی اطلاعات اعشاری ممیز ثابت به سمت راست، الف) مولفه اصلی، ب) ۳ بیت جابه‌جایی مولفه به سمت راست، پ) ۱۰ بیت جابه‌جایی مولفه به سمت راست

Figure (8): Shifting right the fixed-point number, a) Component value, b) Shifting right component 3 bit, c) Shifting right component 10 bit

حال با توجه به این‌که در روش پیشنهادی دوم با اتخاذ سیستم عددی ممیز ثابت ۸ بیت صحیح و ۸ بیت اعشار برای مولفه‌های RGB عناصر تصویری به نتایج مناسبی دست پیدا کردیم؛ در ادامه می‌خواهیم تا با افزایش تعداد بیت‌های اعشاری ضرایب مولفه‌ها به میزان ۱۵ بیت دقت محاسبات را بهبود ببخشیم. با انجام این کار دقت محاسبات بالا رفته و مقدار خطای میانگین مربعات کاهش پیدا خواهد کرد. برنامهٔ وریلاگ این روش به‌صورت زیر است. در شکل (۹) سخت‌افزار معادل نمایش داده شده است. همچنین در جدول (۷) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی سوم و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش نشان داده شده است. در انتها در شکل (۱۰) تصویر خروجی لنا برای بررسی کیفی نمایش داده شده است.

```
//  Methode 3
//  RGB INTEGRE
//  Components fractional part 8 bit and Coefficent fractional part 15 bit
module ShiftAndAdd_ModRGB_7bit_V5(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [15:0] R_N, G_N, B_N;
   wire [16:0] Sum_RGB;
   wire [10:0] Result_Sum;
   //
   assign R_N ={red_i, 8'b00000000};
   assign G_N ={green_i, 8'b00000000};
   assign B_N ={blue_i, 8'b00000000};
   assign Sum_RGB = {3'b000, R_N[15:2]} + {6'b000000, R_N[15:5]} + {7'b0000000, R_N[15:6]}    +
                    {10'b0000000000, R_N[15:9]} + {11'b00000000000, R_N[15:10]}              +
                    {14'b00000000000000, R_N[15:13]} + {15'b000000000000000, R_N[15:14]}     +
                    {2'b00, G_N[15:1]} + {5'b00000, G_N[15:4]} + {7'b0000000, G_N[15:6]}     +
                    {8'b00000000, G_N[15:7]} + {9'b000000000, G_N[15:8]}                     +
                    {14'b00000000000000, G_N[15:13]} + {16'b0000000000000000, G_N[15:15]}    +
                    {5'b00000, B_N[15:4]} + {6'b00000, B_N[15:5]}                            +
                    {7'b00000, B_N[15:6]} +  {12'b000000000000, B_N[15:11]}                  +
                   {14'b00000000000000, B_N[15:13]};
endmodule
```



شکل (۹): سخت‌افزار متناظر با روش پیشنهادی سوم با در نظر گرفتن ۱۵ بیت اعشاری برای ضرایب مولفه‌های RGB

Figure (9):The hardware of the third proposed method, considering 15 bits for fractional part of RGB components coefficents

Table (7): The number of LB used in the third proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۷): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی سوم و خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵			لنا ۵۱۲×۵۱۲			تعداد بلوک‌های منطقی
درصد پیکسل‌های مغشوش	حداکثر قدرمطلق خطا	خطای میانگین مربعات	درصد پیکسل‌های مغشوش	حداکثر قدرمطلق خطا	خطای میانگین مربعات	
٪۱/۸۵	۱	۰/۰۱۸۵	٪۱/۸۴	۱	۰/۰۱۸۴	۱۱۲

شکل (۱۰): تصویر خاکستری خروجی ماژول سخت‌افزاری روش سوم

Figure (10): The output grayscale image of the third proposed hardware module

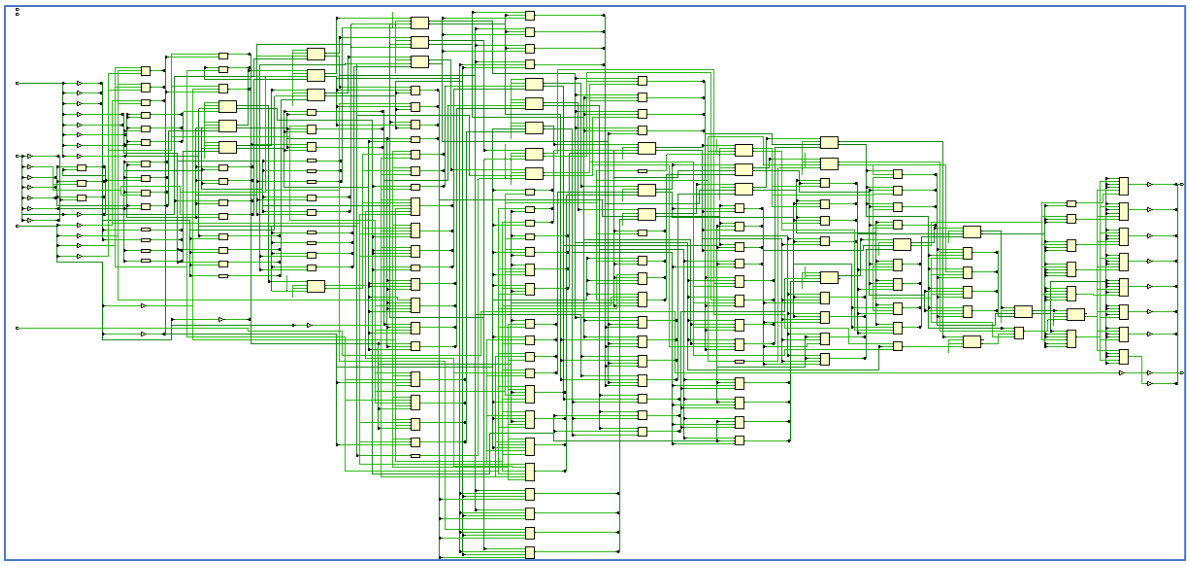
شایان ذکر است که اگر چه در هر دو روش دوم (به ازای ۸ بیت اعشار برای مولفه‌های رنگی) و سوم برای انجام عمل ضرب اعداد باینری از روند جابه‌جایی و جمع استفاده شده است. اما ابعاد اطلاعاتی ضرایب و دیدگاه به کار رفته جهت انجام محاسبات با یکدیگر متفاوت بوده از این‌رو همان‌گونه که در نتایج نشان داده شده حجم سخت‌افزار به کار رفته و دقت محاسبات نیز با یکدیگر متفاوت هستند. روش دوم بر این اساس استوار است که در اثر جابه‌جایی به سمت راست هیچ بیتی با معنایی از بخش شمارشی مولفه‌های رنگی حذف نشوند. اما در روش سوم با پذیرش حذف بیت‌های با ارزش کمتر و فقط ثابت بودن ابعاد اطلاعاتی، مشاهده می‌شود که در ازای ۸ واحد بلوک منطقی افزوده شده به دقت بالاتری می‌رسیم. در الگوریتم دوم و سوم ابعاد اطلاعاتی مولفه‌های تصویری یکسان بوده اما در الگوریتم دوم هدف غایی عدم حذف اطلاعات ناشی از جابه‌جایی به سمت راست بوده اما در الگوریتم سوم رسیدن به حداکثر دقت در ضرایب هدف اصلی است. حال به دلیل وجود یک‌های متوالی در ضرایب مولفه‌های متناظر با بخش‌های سبز و آبی امکان بهینه‌سازی الگوریتم پیشنهادی سوم با استفاده از روش بوث وجود دارد. این روش در رابطهٔ (۲۱) بیان شده است. در این روش n واحد جمع‌کننده به یک جمع‌کننده و یک تفریق‌کننده تبدیل شده حجم سخت‌افزار به کار برده کاهش می‌یابد. در واقع در اینجا با استفاده همبستگی اطلاعاتی موجود در ضرایب مولفه‌های تصویری (وجود ۱ های متوالی در ضرایب) تعداد n واحد جمع‌کننده را به یک جمع‌کننده و یک تفریق‌کننده کاهش دادیم.

$$a*\overset{n}{1...1}B = a*(2^{n+1}-1) \qquad (۲۱)$$

برنامه وریلاگ روش پیشنهادی سوم بهینه‌سازی شده با استفاده از الگوریتم بوث به شرح زیر بوده و سخت‌افزار معادل در شکل (۱۱) نمایش داده شده است. همچنین در جدول (۸) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی سوم به روش بوث و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش نشان داده شده است.

```
// Methode 3 optimized booth
// RGB INTEGER
// Components fractional part 8 bit and Coefficent fractional part 15 bit
module ShiftAndAdd_ModRGB_7bit_V5(
  input [7:0] red_i,green_i,blue_i,
  output wire [7:0] grayscale_o );
  wire [15:0] R_N, G_N, B_N;
  wire [16:0] Sum_RGB;
  wire [10:0] Result_Sum;
  assign R_N ={red_i, 8'b00000000};
  assign G_N ={green_i, 8'b00000000};
  assign B_N ={blue_i, 8'b00000000};
```

```
assign Sum_RGB = {3'b000, R_N[15:2]}  + {6'b000000, R_N[15:5]} + {7'b0000000, R_N[15:6]} +
        {10'b0000000000, R_N[15:9]} + {11'b00000000000, R_N[15:10]}                      +
        {14'b00000000000000, R_N[15:13]} + {15'b000000000000000, R_N[15:14]}             +
        {2'b00, G_N[15:1]} + {5'b00000, G_N[15:4]}                                       +
        {6'b000000, G_N[15:5]} – {9'b000000000, G_N[15:8]}                               +
        {14'b00000000000000, G_N[15:13]} + {16'b0000000000000000, G_N[15:15]}            +
        {4'b0000, B_N[15:3]} – {7'b00000, B_N[15:6]}                                     +
        {12'b000000000000, B_N[15:11]} +{14'b00000000000000, B_N[15:13]};
endmodule
```



شکل (۱۱): سخت‌افزار متناظر با روش پیشنهادی سوم بهینه شده به روش بوث

Figure (11): The hardware of the third proposed method optimized by Booth method

Table (8): The number of LB used in the third proposed hardware optimized by Booth method and the MSE, MAX-ABSER indexes with the faulty pixels percent

جدول (۸): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی سوم به روش بوث و خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش
۱۰۵	۰/۰۱۸۴	۱	٪۱/۸۴	۰/۰۱۸۵	۱	٪۱/۸۵

۴-۳- روش پیشنهادی چهارم

روش پیشنهادی چهارم موسوم به ترکیب مولفه‌های رنگی است. در این روش از همبستگی بین ضرایب مختلف مولفه‌های تصویری (وجود دو یا سه ۱ در ارزش مکانی یکسان بین ضرایب مختلف) استفاده شده است. حال فرض کنیم که تعداد بیت‌های اعشاری زیادی را برای ضرایب مولفه‌های تصویری در نظر گرفته‌ایم؛ اگر هر یک از مولفه‌های تصویر را بصورت جداگانه به سمت راست جابه‌جا کرده و سپس آنها را با یکدیگر جمع نماییم تعداد بلوک‌های منطقی به‌صورت چشم‌گیری افزایش پیدا خواهند کرد. برای جبران این موضوع می‌توان هنگامی که بیت‌های هم ارزشی از ضرایب مولفه‌ها ۱ باشند؛ به‌جای این که هر مولفه را جداگانه به سمت راست جابه‌جا نموده و سپس با یکدیگر جمع نماییم؛ در ابتدا مولفه‌ها را با یکدیگر جمع نموده و سپس حاصل جمع (ترکیب مولفه‌ها) را جابه‌جا نماییم. یعنی در مکان‌هایی که امکان جابه‌جایی مولفه‌ها بصورت مشترک وجود دارد (یعنی در یک ارزش مکانی امکان جابه‌جایی دو یا سه مولفه بصورت همزمان وجود دارد) می‌توان ابتدا آن مولفه‌ها را با یکدیگر جمع نموده و سپس نتیجه را به سمت راست جابه‌جا کرد. به این‌صورت کاهش تعداد بلوک‌های منطقی مشاهده خواهد شد [رابطه‌های (۲۲) و (۲۴)]. در جدول (۹) ترکیب رنگ‌های به‌کار برده شده در روش پیشنهادی چهارم، نمایش داده شده است.

Table (9): Coefficients of combined colors

جدول (۹): ضرایب رنگ‌های ترکیبی

0.	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	ضریب R
0.	1	0	0	1	0	1	1	1	0	0	0	0	1	0	1	ضریب G
0.	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	ضریب B
-	G	R	-	C	M	W	G	G	R	R	B	-	W	R	G	

$$shr(B,2)+shr(G,2) = shr(B+G,2) \quad (۲۲)$$

$$Gray = (0.30*R+0.59*G+0.11*B)$$

$$\text{if} \quad n = 15 \quad \Rightarrow \quad Gray = 0.010011001100110B*R+0.100101110000101B*G+0.000111000010100B*B$$

$$\begin{aligned} Gray = {} & shr(R,2)+shr(R,5)+shr(R,6)+shr(R,9)+shr(R,10)+shr(R,13)+shr(R,14)+ \\ & shr(G,1)+shr(G,4)+shr(G,6)+shr(G,7)+shr(G,8)+shr(G,13)+shr(G,15)+ \\ & shr(B,4)+shr(B,5)+shr(B,6)+shr(B,11)+shr(B,13) \end{aligned} \quad (۲۳)$$

$$\begin{aligned} Gray = {} & shr(R,2)+shr(R+B,5)+shr(R+G+B,6)+shr(R,9)+shr(R,10)+shr(R+G+B,13)+ \\ & shr(R,14)+shr(G,1)+shr(G+B,4)+shr(G,7)+shr(G,8)+shr(G,15)+shr(B,11) \end{aligned}$$

$$M = R+B, \quad C = G+B, \quad W = M+G$$

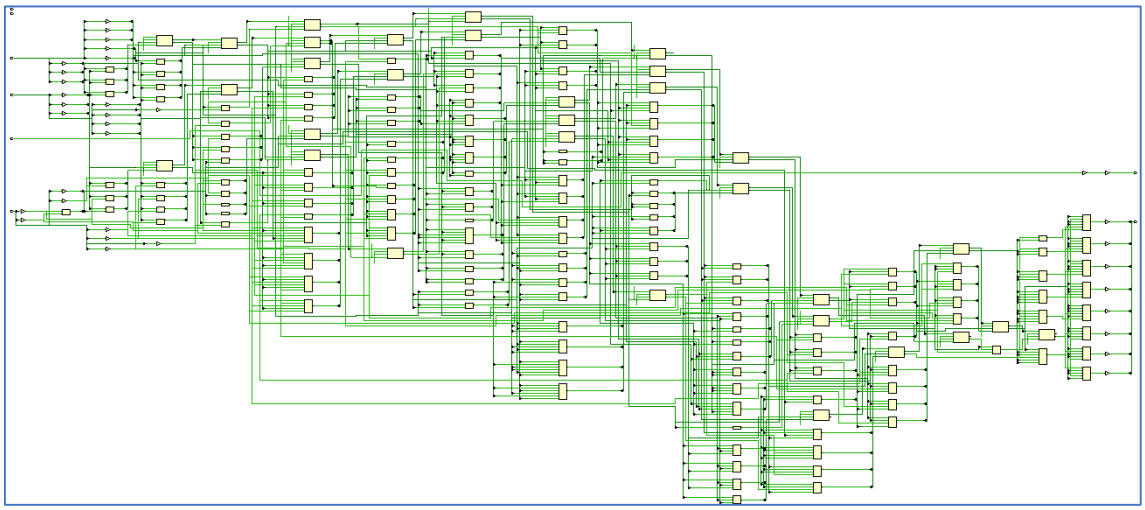
$$\begin{aligned} Gray = {} & shr(R,2)+shr(M,5)+shr(W,6)+shr(R,9)+shr(R,10)+shr(W,13)+ \\ & shr(R,14)+shr(G,1)+shr(C,4)+shr(G,7)+shr(G,8)+shr(G,15)+shr(B,11) \end{aligned} \quad (۲۴)$$

برنامه وریلاگ روش پیشنهادی چهارم به شرح زیر بوده و سخت‌افزار معادل در شکل (۱۲) نمایش داده شده است. همچنین در جدول (۱۰) تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی چهارم و معیارهای خطای میانگین مربعات و حداکثر قدر مطلق خطا و درصد پیکسل‌های مغشوش نشان داده شده است.

```
//  Methode 4
//  RGB INTEGER
//  Components fractional part 8 bit and Coefficent fractional part 16 bit
module ShiftAndAdd_ModRGB_7bit_V5(
   input [7:0] red_i,green_i,blue_i,
   output wire [7:0] grayscale_o  );
   wire [15:0] R_N, G_N, B_N;
   wire [16:0] Con_RB, Con_RG, Con_BG;
   wire [17:0] Con_RB_G;
   wire [16:0] Sum_RGB;
   wire [10:0] Result_Sum;
   assign R_N ={red_i, 8'b0000000};
   assign G_N ={green_i, 8'b0000000};
   assign B_N ={blue_i, 8'b0000000};
   assign Con_RB   = {1'b0, R_N} + {1'b0, B_N};
   assign Con_BG   = {1'b0, B_N} + {1'b0, G_N};
   assign Con_RB_G = {1'b0, Con_RB} + {2'b00, G_N};
   assign Sum_RGB  = {3'b000,R_N[15:2]} +{6'b000000,Con_RB[16:5]} +{7'b0000000,Con_RB_G[17:6]}  +
                     {10'b0000000000,R_N[15:9]} +{11'b00000000000,R_N[15:10]}                   +
                     {14'b00000000000000,Con_RB_G[17:13]}+{15'b000000000000000,R_N[15:14]}     +
                     {2'b00,G_N[15:1]}+{5'b00000,Con_BG[16:4]}+{8'b00000000,G_N[15:7]}          +
                     {9'b000000000,G_N[15:8]}+{16'b0000000000000000,G_N[15:15]}                 +
                     {12'b000000000000,B_N[15:11]};
endmodule
```

از آنجایی که فقط در روش‌های سوم، سوم بهینه شده توسط راه‌کار بوث و روش چهارم ضرایب ۱۶ بیتی لحاظ شده‌اند می‌توان این سه روش را از نظر تعداد بلوک‌های منطقی به‌کار برده شده در طرح‌های سخت‌افزاری (تعداد واحدهای تمام جمع‌کننده و جابه‌جا کنندهٔ بیتی به سمت راست) با یکدیگر مقایسه نمود. در رابطهٔ (۲۳) که در روش پیشنهادی سوم به‌کار گرفته شده است

۱۸ تمام جمع‌کننده و ۱۹ جابه‌جا کننده مشاهده می‌شود. در روش پیشنهادی سوم مبتنی بر راه‌کار بوث به ۱۶ تمام جمع‌کننده و به ۱۷ جابه‌جا کنندهٔ بیتی نیاز خواهیم داشت. اما در رابطهٔ (۲۴) که در روش پیشنهادی چهارم به کار گرفته شده است ۳ تمام جمع‌کننده جهت ایجاد ترکیب دوتایی و سه‌تایی مولفه‌های رنگی به همراه ۱۲ تمام جمع‌کننده دیگر که در مجموع ۱۵ تمام جمع‌کننده شده و ۱۳ جابه‌جا کننده مشاهده می‌شوند.



شکل (۱۲): سخت‌افزار متناظر با روش پیشنهادی چهارم
Figure (12): The hardware of the forth proposed method

Table (10): The number of LB used in the forth proposed hardware and the MSE, MAX-ABSER indexes with the faulty pixels percent
جدول (۱۰): تعداد بلوک‌های منطقی به‌کار برده شده در سخت‌افزار پیشنهادی چهارم و خطای میانگین مربعات ، حداکثر قدرمطلق خطا و درصد پیکسل‌های مغشوش

تعداد بلوک‌های منطقی	لنا ۵۱۲×۵۱۲			کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		
	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش	خطای میانگین مربعات	حداکثر قدرمطلق خطا	درصد پیکسل‌های مغشوش
۱۰۵	۰/۰۱۸۴	۱	٪۱/۸۴	۰/۰۱۸۵	۱	٪۱/۸۵

۴- بررسی و مقایسهٔ روش‌های پیشنهادی

در روش پیشنهادی اول محاسبات ضرب اعشاری توسط عملیات جابه‌جایی و جمع در تراشهٔ FPGA سری اسپارتان-۳ مدل xc3s50 شبیه‌سازی و پیاده‌سازی شده است. این روش مبتنی بر جابه‌جا کردن اعداد شمارشی مولفه‌های تصویری به سمت راست و جمع نمودن آنها است. در این روش محاسبات بصورت شمارشی بوده و به هنگام جابه‌جایی به سمت راست بیت‌های کم ارزش مولفه‌های تصویری حذف می‌شوند. همان‌گونه که در جدول (۲) مشاهده می‌شود در این روش حداکثر دقت هنگامی حاصل می‌گردد که ضرایب مولفه‌ها بصورت اعداد اعشاری ۴ و یا ۵ بیتی بیان شوند. به این ترتیب ساده‌ترین سخت‌افزار به‌دست آمده اما خطا در مقایسه با سایر روش‌های پیشنهادی حداکثر است. در روش پیشنهادی بعدی برای ارتقا روش اول و کاهش خطای محاسباتی در ابتدا اعداد اعشاری نظیر ضرایب مولفه‌های تصویری را در 2^7 ضرب نموده تا تبدیل به اعداد شمارشی شوند و در اثر جابه‌جایی مولفه‌های تصویری به سمت چپ هیچ بیت با ارزشی دور ریخته نشود سپس در انتها حاصل محاسبات را بر 2^7 تقسیم می‌نماییم. همان‌گونه که از جدول (۱۱) بر می‌آید اگرچه ایده اصلی محاسبات در دو روش یکسان است، اما به ازای دو برابر شدن حجم سخت‌افزار، دقت محاسبات بصورت چشم‌گیری افزایش یافته است.

در روش پیشنهادی دوم از سیستم عددی ممیز ثابت برای بیان مولفه‌های RGB عناصر تصویری و ضرایب مربوطه استفاده کردیم. در اینجا با افزودن بیت‌های اعشاری به مولفه‌های RGB عناصر تصویری، دیگر اطلاعات ۸ بیتی بخش شمارشی مولفه‌های تصویری، پس از جابه‌جایی به سمت راست و تقسیم بر توان‌های ۲ شدن دور ریخته نشده بلکه وارد قسمت اعشاری عدد شده و

در نتیجه خطا کاهش پیدا می‌نماید. با افزایش تعداد بیت‌های قسمت اعشاری در سیستم عددی ممیز ثابت میزان دقت و مقدار سخت‌افزار به‌کار برده شده افزایش می‌یابد. برای کاهش حجم سخت‌افزار به‌کار برده شده به ویژه کاهش ابعاد عملوندهای ورودی تمام جمع‌کننده‌های به‌کار برده شده در روش پیشنهادی دوم، روش پیشنهادی دوم بهینه شده ارائه گردید. همان‌گونه که در جدول (۱۱) مشاهده می‌شود که روش پیشنهادی دوم بهینه شده با تعداد سخت‌افزار کمتری نسبت به روش پیشنهادی دوم، به همان شاخص‌های روش پیشنهادی دوم در شبیه‌سازی‌ها دست می‌یابد.

در روش پیشنهادی سوم مولفه‌های RGB عناصر تصویری توسط سیستم عددی ممیز ثابت با ۸ بیت بخش صحیح و ۸ بیت بخش اعشاری و ضرایب مولفه‌ها به‌صورت اعداد اعشاری ۱۵ بیتی بیان شده‌اند. در نتیجه حاصل ضرب هر مولفه در ضریب متناظر با امکان از دست دادن اطلاعات توسط اعداد اعشاری باینری با ۸ رقم اعشار و ۸ رقم صحیح بیان می‌شود و در نهایت گرد شده و به یک عدد ۸ بیتی شمارشی بدون علامت تبدیل می‌گردد. با انجام این روند دقت محاسبات بالا رفت و مقدار خطای میانگین مربعات کاهش پیدا کرد. اگر چه در هر دو روش پیشنهادی دوم و پیشنهادی سوم از روند جابه‌جایی بیتی به سمت راست و جمع که اساس عمل ضرب است استفاده شده و ابعاد اطلاعاتیِ مولفه‌های تصویری یکسان بوده اما ابعاد اطلاعاتیِ ضرایب و دیدگاه به کار رفته جهت انجام محاسبات با یکدیگر متفاوت است. از این‌رو همان‌گونه که در جدول (۱۱) نتایج نشان می‌دهند حجم سخت‌افزار به‌کار رفته و دقت محاسبات کسب شده با یکدیگر متفاوت هستند. روش پیشنهادی دوم بر این اساس استوار است که در اثر جابه‌جایی به سمت راست هیچ بیتی از مولفه‌های رنگی حذف نشوند. اما در روش سوم با پذیرش امکان حذف بیت‌های با ارزش کمتر اطلاعات تصویری و صرفاً ثابت بودن ابعاد اطلاعاتیِ حاصل ضرب (۸ بیت صحیح و ۸ بیت اعشار) و افزایش ابعاد بیت ضرایب مولفه‌ها، به ازای ۸ واحد بلوک منطقی افزوده شده به دقت بالاتری رسیده‌ایم. به عبارت دیگر در روش پیشنهادی دوم هدف نهایی عدم حذف اطلاعات ناشی از جابه‌جایی به سمت راست بوده. اما در روش پیشنهادی سوم رسیدن به حداکثر دقت در ضرایب مولفه‌های تصویری هدف اصلی است. به علاوه به دلیل وجود یک‌های پشت سرهم در ضرایب مولفه‌های متناظر با بخش‌های سبز و آبی امکان پیاده‌سازی الگوریتم پیشنهادی با روش بوث وجود داشت. که در روش پیشنهادی سوم مبتنی بر الگوریتم بوث بکار گرفته شد. در اثر استفاده از راه‌کار بوث تعداد سخت‌افزار بکار برده شده ۱۳ واحد کاهش یافته اما به همان معیارهای روش پیشنهادی سوم دست یافتیم.

در نهایت روش پیشنهادی چهارم را برای زمانی ارائه کردیم که تعداد بیت‌های اعشاری ضرایب مولفه‌های تصویری زیاد باشند. در این روش به‌جای این که هر مولفه را جداگانه به سمت راست جابه‌جا نموده و سپس جمع نماییم؛ در ابتدا مولفه‌ها را جمع نموده و سپس حاصل را جابه‌جا نمودیم. نکتهٔ شایان ذکر این که اگر چه الگوریتم‌های به کار برده شده در روش پیشنهادی سوم بهینه‌سازی شده با استفاده از الگوریتم بوث و روش پیشنهادی چهارم با یکدیگر کاملاً متفاوت هستند. اما نرم‌افزار سنتزکنندهٔ VIVADO با استفاده از هوش مصنوعی به کار گرفته شده برای بهینه‌سازی طرح‌های سخت‌افزارهای، تعداد بلوک‌های منطقی مورد نیاز برای دو الگوریتم متفاوت را یکسان (۱۰۵ بلوک منطقی) به‌دست آورده است اما لزوماً سخت‌افزار این دو الگوریتم یکسان نیست. این امر بدلیل تفاوت مدارهای منطقی شکل‌های (۱۱) و (۱۲) است. به عبارت دیگر محتویات بلوک‌های منطقی یکسان نیست. با این وجود نتایج شبیه‌سازی‌ها نیز یکسان است.

همان‌گونه که از جدول (۱۱) بر می‌آید نتایج روش پیشنهادی اول به ازای n برابر با ۴ و ۵ دقیقاً از نتیجه‌های مرجع [۱۱] بهتر است. اگرچه این روش حداقل سخت‌افزار را نیاز دارد اما دارای خطای محاسباتی بسیار زیادی است. که این امر ناشی از محاسبات صحیحی است که در الگوریتم به کار گرفته شده است. در مقایسهٔ روش پیشنهادی اول بهینه شده به ازای n مساوی با ۷ با روش اول به ازای n برابر با ۴ و ۵ در می‌یابیم که برای رسیدن به دقت بالاتر حجم سخت‌افزار افزایش می‌یابد. در مقایسهٔ روش دوم بهینه شده با روش دوم مشاهده می‌گردد که روند بهینه‌سازی کارآمد بوده و همان شاخص‌های روش دوم با سخت‌افزار کمتری به‌دست آمده است. همچنین مشاهده می‌شود که به‌کارگیری الگوریتم بوث در روش پیشنهادی سوم باعث کاهش حجم سخت‌افزار می‌گردد. اما بالاترین دقت هنگامی حاصل می‌گردد هنگامی که یکی از دو روش پیشنهادی سوم مبتنی بر الگوریتم بوث و یا روش چهارم به ازای ۱۵ بیت اعشار اتخاذ گردد. نکتهٔ شایان ذکر این است که در روش مرجع [۱۲] با استفاده از مدار ضرب‌کننده به دقت بالا رسیده‌اند اما حجم سخت‌افزار به کار برده شده بسیار زیاد است. روش مرجع [۹] دقت مناسبی داشته و هم حجم

سخت‌افزار به کار برده شده نسبتا پایین است. اما از ضرب‌کنندهٔ مجتمع DSP48 استفاده کرده است. اگرچه در روش‌های مرجع‌های [۱۰] و [۱۱] حجم سخت‌افزار به‌کار برده شده بسیار کم بوده اما دقت محاسبات بسیار پایین است. روش مرجع [۱۲] دقت مناسبی داشته اما به دلیل پیاده‌سازی عمل ضرب به‌صورت سخت‌افزاری حجم سخت‌افزار به کار برده شده بسیار زیاد است.

Table (11): The number of LB used in the hardwares of all proposed methods and other reffrences and the MSE, MAX-ABSER indexes

جدول (۱۱): تعداد بلوک‌های منطقی به‌کار برده شده در روش‌های پیشنهادی و مراجع توسط تراشهٔ FPGA سری اسپارتان-۳ مدل xc3s50 معیارهای خطای میانگین مربعات ، حداکثر قدرمطلق خطا

روش	کلیهٔ مولفه‌های رنگی از ۰ تا ۲۵۵		لنا ۵۱۲×۵۱۲		تعداد بلوک‌های منطقی
	حداکثر قدرمطلق خطا	خطای میانگین مربعات	حداکثر قدرمطلق خطا	خطای میانگین مربعات	
روش پیشنهادی اول به ازای ۴ بیت اعشار	۱۱	۱۰/۷۶۷	۵	۱/۶۱۰	۲۸
روش پیشنهادی اول به ازای ۵ بیت اعشار	۸	۵/۰۰۸	۷	۵/۱۷۰	۲۷
روش پیشنهادی اول بهینه شده به ازای ۷ بیت اعشار	۲	۰/۵۴۵۰	۱	۰/۷۵۲۰	۵۴
روش پیشنهادی دوم به ازای ۸ بیت اعشار	۱	۰/۰۶۱۷	۱	۰/۰۵۸۷	۱۰۴
روش پیشنهادی دوم بهینه شده به ازای ۸ بیت اعشار	۱	۰/۰۶۱۷	۱	۰/۰۵۸۷	۹۱
روش پیشنهادی سوم	۱	۰/۰۱۸۵	۱	۰/۰۱۸۴	۱۱۲
روش پیشنهادی سوم بهینه شده مبتنی بر روش بوث	۱	۰/۰۱۸۵	۱	۰/۰۱۸۴	۱۰۵
روش پیشنهادی چهارم به ازای ۱۵ بیت اعشار	۱	۰/۰۱۸۵	۱	۰/۰۱۸۴	۱۰۵
[۹]	۳	۰/۰۲۹۶	۱	۰/۰۷۵۷	۴۵
[۱۰]	۴	۵/۵۰۴۵	۱	۷/۳۶۵۰	۳۵
[۱۱]	۹	۱۷/۴۲۲	۸	۱۹/۵۶۰۰	۳۴
[۱۲]	۱	۰/۰۴۹۵	۱	۰/۰۶۲۵	۶۵۶

در مقایسهٔ کلی در می‌یابیم که روش‌های پیشنهادی چهارم و سوم بهینه شده مبتنی بر روش بوث بهترین دقت را به ازای ۱۰۵ بلوک منطقی دارند. شایان ذکر است که نتایج نمایش داده شده در جدول (۱۱) حاصل پیاده‌سازی و شبیه سازی کلیهٔ الگوریتم‌های پیشنهادی و الگوریتم‌های مراجع در تراشهٔ FPGA سری اسپارتان-۳ مدل xc3s50 و داده‌های یکسان است.

۵- نتیجه گیری

محققین روش‌های متعددی برای پیاده‌سازی الگوریتم میانگین‌گیری وزن‌دهی شده جهت تبدیل تصاویر رنگی به خاکستری را در تراشه‌های FPGA ارایه نموده‌اند. ما در این مقاله در مجموع ۴ روش و ۳ روش بهینه شده را پیشنهاد نمودیم. با توجه به امکانات تراشهٔ FPGA سری اسپارتان-۳ مدل xc3s50 از روش تمام ترکیبی جابه‌جایی و جمع جهت رسیدن به حداقل تاخیر زمانی با استفاده از واحدهای محاسبات منطقی ساده در تراشه‌های FPGA ارزان قیمت استفاده کردیم. از میان روش‌های پیشنهادی، روش پیشنهادی سوم مبتنی بر الگوریتم بوث و روش چهارم بهترین دقت را به ازای ۱۰۵ بلوک منطقی دارند. در روش پیشنهادی سوم مبتنی بر الگوریتم بوث از همبستگی اطلاعاتی مابین هر یک از ضرایب مولفه‌های تصویری و در روش پیشنهادی چهارم از همبستگی اطلاعاتی ما بین بیت‌های با ارزش مکانی یکسان در ضرایب مولفه‌های تصویری استفاده شده است. به این ترتیب مشاهده می‌شود که نه تنها حجم سخت‌افزار به کار برده شده در دو شیوه یکسان بوده، بلکه معیارهای یکسانی را نیز در شبیه‌سازی‌ها به‌دست آوردند.

سپاسگزاری

این پژوهش برگرفته شده از طرح تحقیقاتی به شمارهٔ ۳۹۷۱/ص/۱۴۰۰ فی مابین دانشگاه آزاد اسلامی واحد نجف‌آباد و گروه فن‌آوران سپاهان رمزینه است که لازم است از پشتیبانی مرکز تحقیقات پردازش دیجیتال و بینایی ماشین دانشگاه آزاد اسلامی واحد نجف‌آباد کمال تشکر و قدردانی را داشته باشیم. نویسندگان بر خود لازم می‌دانند مراتب تشکر صمیمانه خود را از همکاران حوزه پژوهشی دانشگاه آزاد اسلامی، عوامل اجرایی نشریه و داوران محترم که ما را در انجام و ارتقای کیفی این مقاله یاری نموده‌اند، اعلام نمایند.

References

مراجع

[1] P. Sanaee, P. Moallem, F. Razzazi, "An interpolation filter based on natural neighbor Galerkin method for salt and pepper noise restoration with adaptive size local filtering window", Signal, Image and Video Processing, vol. 13 , pp. 895-903, Jan. 2019 (doi: 10.1007/s11760-019-01426-3).

[2] B. Lee, J. Choi, K. Yun, J.Y. Choi. "Gradient preserving RGB-to-gray conversion using random forest", Proceeding of the IEEE/ICIP, pp. 3170-3174, Quebec City, QC, Canada, Sept. 2015 (doi: 10.1109/ICIP.20-15.7351388).

[3] H.Z. Nafchi, A. Shahkolaei, R. Hedjam, M. Cheriet, "CorrC2G: Color to gray conversion by correlation", IEEE Signal Processing Letters, vol. 24, no. 11, pp. 1651-1655, Nov. 2017 (doi: 10.1109/LSP.2017.2755077).

[4] Y. Kim, J. Cheolhun, J. Demouth, L. Seungyong, "Robust color-to-gray via nonlinear global mapping", ACM Transactions on Graphics, vol. 28, no. 5, pp. 1-4, Dec. 2009 (doi: 10.1145/1661412.1618507).

[5] H.A.A. Alshadoodee, M.Q. Dhahir, Z.H. Rasool, "Digital camera in movement tracking on FPGA board DE2", Proceeding of the IEEE/RusAutoCon, pp. 1-8, Sochi, Russia, Dec. 2019 (doi: 10.1109/RUSAUTOCON.2-019.8867726).

[6] S.H. Keyhomayoon, M. Amoom, "Design and implementation of an intelligent high frequency counter with optimized architecture on a low cost FPGA Chip XC6SLX9-2FTG256C", Journal of Intelligent Procedures in Electrical Technology, vol. 14, no. 54, pp. 147-157, Sept. 2023 (dor: 20.1001.1.23223871.1402.14.54.10.6).

[7] O. Sharifi Tehrani, M. Ashorian, P. Moallem, "Hardware implementation of LMS-based adaptive noise cancellation core with low resource utilization", Journal of Intelligent Procedures in Electrical Technology, vol. 2, no. 7, pp. 68-73, Dec. 2011 (dor: 20.1001.1.23223871.1390.2.7.8.6).

[8] A. Abdolazimi, A.S. Molahosseini, F. Keynia, "Detect and implement facial modes in image on FPGA using multilayer neural network", International Journal of Smart Electrical Engineering, vol. 10, no. 3, pp. 135-140, Sept. 2021 (doi: 10.30495/ijsee.2021.684015).

[9] X. Yang, S. Sha, "Exploiting energy–quality (E–Q) tradeoffs: A case study on color-to-grayscale converters with approximate design on FPGA", Journal of Circuits, Systems and Computers, vol. 30, no. 04, Article Number: 2150062, Aug. 2020 (doi: 10.1142/S0218126621500626).

[10] U. Swathi, U. Smitha, "Design and implementation of efficient RGB to gray scale converter architectures using reversible logic", Proceeding of IEEE/DISCOVER, pp. 194-199, Udupi, India, Oct. 2020 (doi: 10.11-09/DISCOVER50404.2020.9278066).

[11] K. Kumar, R.K. Mishra, D. Nandan, "Efficient hardware of RGB to gray conversion realized on FPGA and ASIC", Procedia Computer Science, no. 171, pp: 2008-2015, Jan. 2020 (doi: 10.1016/j.procs.2020.04.215).

[12] Y. Zhang, X. Yang, L. Wu, J. Lu, K. Sha, A. Gajjar, H. He, "Exploring slice-energy saving on an video processing FPGA platform with approximate computing", Proceeding of the ICACS, pp. 138-143, Beijing, China, Jul 2018 (doi: 10.1145/3242840.3242852).

[13] G. Ravivarma, K. Gavaskar, D. Malathi, K.G. Asha, B. Ashok, S. Aarthi, "Implementation of sobel operator based image edge detection on FPGA", Materials Today: Proceedings, vol. 45, part 2, pp. 2401-2407, Jan. 2021 (doi: 10.1016/j.matpr.2020.10.825).

[14] B.P. Kumar, B.H. Kumar, "FPGA implementation of canny edge detection algorithm for noisy image identification", International Journal of Management, Technology and Engineering, vol. 9, no. 12, pp. 386-395, Dec. 2019.

[15] S. Yaman, M. Yildrim, B. Karmşhoğlu, Y. Erol, H. Kürüm, "Image and video processing applications using Xilinx system generator", Proceeding of the IEEE/ISDFS, pp. 1-5, Barcelos, Portugal, June 2019 (doi: 10.1109/ISDFS.2019.8757540).

[16] C. Li, Y. Bi, F. Marzani, F. Yang, "Fast FPGA prototyping for real-time image processing with very high-level synthesis", Journal of Real-Time Image Processing, vol. 16, pp. 1795–1812, Apr. 2017 (doi: 10.1007/s1-1554-017-0688-1).

[17] G. Wilson, Y. Premson, "FPGA implementation of hardware efficient algorithm for image contrast enhancement using Xilinx System Generator", Procedia Technology, vol. 24, pp. 1141-1148, Jan. 2016 (doi: 10.1016/j.protcy.2016.05.067).

زیرنویس‌ها

1. Single color channel
2. Lightness
3. Desaturation
4. Decomposition
5. Average
6. Weighted mean
7. Luminosity
8. Multi core processors
9. Advanced risc machine
10. Digital signal processors
11. Graphic processor unit
12. Application-specific integrated circuit
13. Field-programmable gate array
14. Single instruction, multiple data
15. Verilog
16. Spartan 6 FPGA family-xilinx
17. Logic Block
18. Matlab
19. Simulink
20. Xilinx system generator hardware description language coder
21. Shift and add
22. Booth
23. https://research.iaun.ac.ir/pd/sanaee/pdfs/UploadFile_9811.zip
24. Photometric
25. National television standards committee
26. Mean squared error
27. Max absolute value of the error
28. Faulty pixels percent
29. Fixed-point
30. Round off
31. Barrel-shifter
32. Synthesize
33. Python