Available in:
Journal.isrc.ac.ir

Journal of Space Science, Technology& Applications (Persian)

Vol. 5, No. 2, pp.: 89-102
2026

DOI:
10.22034/jssta.2025.493057.1220

Article Info

Received: 2024-12-03
Accepted: 2025-01-31

Keywords

Block Adaptive Quantization, Data Compression, FPGA

How to Cite this article

A. Dastmozd, M. R. Mousavi, K. Hassanli, S. Abdi, M. Hatam, "FPGA Based Block Adaptive Quantization Design and Implementation", *Journal of Space Science, Technology& Applications*, Vol. 5, No. 2, pp. 89-102, 2026.

Original Article

FPGA Based Block Adaptive Quantization Design and Implementation

Abbas Dastmozd[1], Mohammad Reza Mousavi[2], Kourosh Hassanli[3*], Sahar Abdi[4], Mahdi Hatam[5]

[1] School of Electrical Engineerig, Shiraz University of Technology, Shiraz, Iran
a.dastmozd@sutech.ac.ir

[2] School of Electrical Engineerig, Shiraz University of Technology, Shiraz, Iran
m.mousavi@sutech.ac.ir

[3] School of Electrical Engineerig, Shiraz University of Technology, Shiraz, Iran
hassanli@sutech.ac.ir

[4] Institute of Mechanics, Shiraz, Iran
s.abdi@isrc.ac.ir

[5] Institute of Mechanics, Shiraz, Iran
ma.hatam@isrc.ac.ir

Abstract

This paper investigates the implementation of the Block Adaptive Quantization (BAQ) algorithm on FPGA hardware, specifically the Xilinx Virtex-7 FPGA, which is an ideal platform for real-time applications due to its flexibility and parallel processing capabilities. In this study, the BAQ algorithm was optimized using the principles of non-uniform Lloyd-Max quantization. The hardware design consists of adaptive blocks enhanced by techniques such as parallelization and pipelining. The implementation results demonstrate significant reductions in FPGA resource utilization while maintaining high performance and low latency. Evaluations indicate that the proposed method achieves less than 2% difference compared to the reference approach in metrics such as MSE, PSNR, SSIM, and CC, highlighting the algorithm's high accuracy and consistency. Furthermore, the implementation is resource-efficient, utilizing only 10% of DSP48 resources and 30% of LUTs on the Virtex-7 FPGA. This approach provides an effective solution for applications such as data compression in imaging systems and signal processing.

دسترس‌پذیر در نشانی:
Journal.isrc.ac.ir

دوفصلنامه علوم، فناوری و کاربردهای فضایی

سال پنجم، شماره ۲، صفحه ۱۰۲-۸۹
پاییز و زمستان ۱۴۰۴

DOI:
10.22034/jssta.2025.493057.1220

مقاله پژوهشی

طراحی و پیاده‌سازی چندی‌سازی بلوکی تطبیق‌پذیر برروی FPGA**

عباس دست‌مزد[۱]، محمدرضا موسوی[۲]، کورش حسنلی*[۳]، سحر عبدی[۴] و مهدی حاتم[۵]

۱- دانشکده مهندسی برق، دانشگاه صنعتی شیراز، شیراز، ایران- a.dastmozd@sutech.ac.ir
۲- دانشکده مهندسی برق، دانشگاه صنعتی شیراز، شیراز، ایران- m.mousavi@sutech.ac.ir
۳- دانشکده مهندسی برق، دانشگاه صنعتی شیراز، شیراز، ایران- hassanli@sutech.ac.ir
۴- پژوهشکده مکانیک، پژوهشگاه فضایی ایران، شیراز، ایران- s.abdi@isrc.ac.ir
۵- پژوهشکده مکانیک، پژوهشگاه فضایی ایران، شیراز، ایران- ma.hatam@isrc.ac.ir
نویسنده مسئول*

**این تحقیق توسط پژوهشگاه فضایی ایران در قالب طرح پژوهشی مشترک با دانشگاه مورد حمایت قرار گرفته است.

چکیده

این مقاله به پیاده‌سازی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر (BAQ) بر روی سخت‌افزار FPGA، به‌ویژه تراشه Xilinx Virtex-۷، می‌پردازد. این تراشه به دلیل انعطاف‌پذیری و توانایی پردازش موازی، گزینه‌ای مناسب برای کاربردهای بلادرنگ محسوب می‌شود. در این پژوهش، الگوریتم BAQ با استفاده از اصول چندی‌سازی غیر یکنواخت Lloyd-Max بهینه‌سازی شده است. طراحی سخت‌افزاری شامل بلوک‌های تطبیق‌پذیری است که با تکنیک‌هایی مانند موازی‌سازی و پردازش مرحله‌ای بهبود یافته‌اند. نتایج پیاده‌سازی نشان می‌دهد که این طراحی، استفاده از منابع FPGA را به‌طور قابل توجهی کاهش داده و در عین حال، کارایی بالا و تأخیر کم را حفظ می‌کند. ارزیابی‌ها نشان می‌دهد که روش پیشنهادی در معیارهایی همچون SSIM، PSNR، MSE و CC نسبت به روش مرجع، اختلافی کمتر از ۲ درصد دارد که دقت و سازگاری بالای الگوریتم را تأیید می‌کند. علاوه بر این، پیاده‌سازی سخت‌افزاری با مصرف تنها ۱۰ درصد از منابع DSP۴۸ و ۳۰ درصد از LUT های تراشه Virtex-۷، کارایی بالایی را ارائه می‌دهد. این روش، راهکاری مؤثر برای فشرده‌سازی داده‌ها در سیستم‌های تصویربرداری و پردازش سیگنال ارائه می‌دهد.

تاریخچه داوری

دریافت: ۱۴۰۳/۰۹/۱۹
پذیرش: ۱۴۰۳/۱۱/۱۸

واژه‌های کلیدی

چندی سازی بلوکی تطبیق پذیر، فشرده سازی داده،آرایه گیت های قابل برنامه ریزی

نحوه استناد به این مقاله

عباس دست‌مزد، محمدرضا موسوی، کورش حسنلی، سحر عبدی و مهدی حاتم، "طراحی و پیاده‌سازی چندی‌سازی بلوکی تطبیق‌پذیر برروی FPGA "، *دوفصلنامه علوم، فناوری و کاربردهای فضایی*، جلد پنجم، شماره دوم، صفحات ۱۰۲-۸۹، ۱۴۰۴.

۱- مقدمه

در دنیای امروز، با پیشرفت سریع فناوری و نیاز فزاینده به پردازش داده‌های حجیم و پیچیده در زمان واقعی، تقاضا برای روش‌های بهینه‌سازی پردازش سیگنال و داده‌ها به‌طور چشمگیری افزایش یافته است. چندی‌سازی، به‌عنوان یکی از تکنیک‌های مهم در کاهش حجم داده‌ها و پیچیدگی محاسباتی، نقشی کلیدی در سیستم‌های پردازشی مدرن ایفا می‌کند.

حجم محدود حافظه موقت و پهنای باند محدود لینک انتقال داده در سامانه‌های مخابراتی و تصویربرداری، مستلزم کاهش حجم داده‌ای است که می‌بایست ذخیره شود. لذا با توجه به موارد مطرح‌شده نیازمند الگوریتم‌هایی با نرخ فشرده‌سازی مناسب هستیم که بتوان داده خام را به طور مطلوب فشرده نمود به شکلی که کیفیت داده یا تصویر حاصله از تشکیل داده یا تصویر فشرده‌شده، تخریب نگردد [۱].

بسیاری از رویکردهای چندی‌سازی داده خام در دو دهه گذشته توسعه یافته‌اند و هم اکنون نیز در حال تکامل می‌باشند [۲]. گرچه محدودیت‌های موجود در سامانه‌های مخابراتی و تصویربرداری از نظر پهنای باند لینک انتقال داده، انتخاب الگوریتم فشرده‌سازی جهت پیاده‌سازی را تحت‌الشعاع قرار می‌دهند [۲، ۳]. این محدودیت‌ها موجب شده است که در سامانه‌های مخابراتی و تصویربرداری عملیاتی از رویکردهای ساده و بهینه چندی‌سازی استفاده شود [۱، ۴]. چندی‌سازی بلوکی تطبیق‌پذیر (BAQ[1]) یکی از روش‌های پیشرفته در این زمینه است که با اعمال چندی‌سازی متغیر بر اساس ویژگی‌های داده‌ها، بهینه‌سازی بیشتری را در مصرف منابع و حفظ کیفیت خروجی فراهم می‌آورد.

رویکرد BAQ، یک رویکرد متداول جهت فشرده‌سازی داده‌های خام سیستم‌های مخابراتی و تصویربرداری محسوب می‌شود. در سامانه‌های تصویربرداری، به دلیل حجم بالای داده اخذشده برای هر ناحیه، ذخیره‌سازی یا ارسال داده خام از طریق لینک داده چالش بزرگی به حساب می‌آید. لذا در این سامانه‌ها با فشرده‌سازی داده خام، پهنای باند لینک داده یا حجم حافظه موردنیاز برای ذخیره‌سازی آن به صورت موثر کاهش می‌یابد. این رویکرد شامل یک چندی‌ساز بهینه است که سطوح چندی‌سازی را به توزیع داده‌های بلوک‌بندی شده مرتبط می‌سازد.

در [۲] یک رویکرد با پیچیدگی محاسباتی کم جهت متمرکزکردن داده و فشرده‌سازی داده خام به وسیله الگوریتمی مبتنی بر چندی‌سازی برداری تطبیق‌پذیر ارائه شده است که موجب بهبود سیگنال به نویز داده متمرکزشده به مقداری کمتر از ۱dB می‌گردد. در [۳] رویکرد BAQ به وسیله کدگذاری هافمن ارتقا یافته است. در [۵] از یک چندی‌ساز غیریکنواخت تحت عنوان Lloyd Max جهت کمینه‌کردن میانگین مربعات خطا برای یک تعداد مشخص از سطوح چندی‌سازی استفاده شده است. در [۶] از یک رویکرد مبتنی بر موجک جهت کدگشایی داده خام سار استفاده شده است که در نرخ bit/sample ۳ موجب ۳dB بهبود SNR می‌گردد.

درعین‌حال، FPGA[2]ها (آرایه‌های دروازه‌ای قابل‌برنامه‌ریزی) به‌عنوان یک پلتفرم سخت‌افزاری قابل‌پیکربندی، توانایی بالایی در اجرای موازی و بلادرنگ الگوریتم‌های پیچیده دارند. این ویژگی‌ها FPGA را به یکی از بهترین گزینه‌ها برای پیاده‌سازی الگوریتم‌هایی نظیر BAQ تبدیل کرده است. FPGAها با ارائه انعطاف‌پذیری در طراحی و قابلیت پیکربندی مجدد، امکان پیاده‌سازی بهینه‌سازی‌های سخت‌افزاری را برای کاربردهای خاص فراهم کرده و به طور هم‌زمان، کارایی و سرعت پردازش را افزایش می‌دهد [۷].

پیاده‌سازی الگوریتم BAQ بر روی FPGA، نیازمند درنظرگرفتن ملاحظات مختلف از جمله محدودیت‌های منابع سخت‌افزاری، نیازمندی‌های زمان‌بندی و بهینه‌سازی مصرف انرژی است [۸]. با این حال، ترکیب قدرت محاسباتی FPGA با انعطاف‌پذیری الگوریتم BAQ می‌تواند منجر به بهبود چشمگیری در عملکرد و کارایی سیستم‌های پردازشی شود [۹]. در [۹-۱۳] به بررسی روش‌های مختلف برای بهینه‌سازی و پیاده‌سازی BAQ بر روی FPGA پرداخته شده است و تلاش شده با استفاده از تکنیک‌های پیشرفته نظیر Pipeline، موازی‌سازی و بهینه‌سازی معماری، به یک پیاده‌سازی بهینه و کارآمد رسید.

در [۱۴] به بررسی پیاده‌سازی یک فشرده‌ساز داده پرداخته شده است. این فشرده‌ساز به‌عنوان یک ماژول جانبی بر روی بورد FPGA از خانواده Virtex-7 طراحی و پیاده‌سازی شده و از الگوریتم BAQ بهره می‌گیرد. هدف اصلی این پژوهش، کاهش حجم داده‌های خام به‌منظور تسهیل ارسال آنها است. با این‌حال در کاربردهای بسیار دقیق عملکرد ضعیفی دارد و همچنین به تاخیر طرح پیاده‌سازی‌شده

[1] Block Adaptive Quantization

[2] Field Programmable Gate Array

توجه نشده است. در [۱۵] الگوریتم BAQ روی FPGA Stratix EP1S30F1020I6 پیاده‌سازی شده است. در این پژوهش پیاده‌سازی عملی برروی سخت‌افزار انجام نشده و صرفا پیاده‌سازی طرح در سطح شبیه‌سازی است. همچنین، ارزیابی جامعی جهت تحلیل نتایج بدست‌آمده انجام نشده است و تحلیل تأثیر تأخیر زمانی یا مقیاس‌پذیری سیستم در پروژه‌های بزرگ‌تر نادیده گرفته شده است. در [۱۶] به بررسی طراحی و پیاده‌سازی الگوریتم BAQ بر اساس معماری FPGA پرداخته شده است. در این پژوهش، از FPGAمدل XILINX Virtex 5 XC5VLX110T استفاده شده است و به تأخیر طرح پیاده‌سازی‌شده پرداخته نشده است. همچنین تمرکز بیشتر این پژوهش در سطح شبیه‌سازی بوده و در طرح پیاده‌سازی‌شده Mode های مختلف BAQ در نظر گرفته نشده است.

در بررسی مقالات در زمینه پیاده‌سازی سخت‌افزار الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر به میزان منابع مصرفی استفاده شده، میزان تاخیر موجود در طراحی سخت‌افزار، تنوع Mode های الگوریتم BAQ و کاربرد عملی و پیاده‌سازی سخت‌افزاری برروی FPGA توجه نشده است.

هدف اصلی این پژوهش، توسعه یک راهکار سخت‌افزاری است که بتواند به‌صورت تطبیق‌پذیر و با بهره‌وری بالا، عملیات چندی‌سازی را بر روی داده‌های ورودی با کمترین میزان تاخیر انجام دهد و درعین‌حال، منابع سخت‌افزاری را به بهترین نحو ممکن مدیریت کند. نتایج حاصل از این پیاده‌سازی می‌تواند در کاربردهای مختلف از جمله‌پردازش سیگنال، سیستم‌های نهفته و تصویربرداری و شبکه‌های عصبی مصنوعی به کار گرفته شود و بهبود قابل‌توجهی در عملکرد و بهره‌وری این سیستم‌ها ایجاد کند.

۲- مواد و روش ها

۲-۱- اصول رویکرد چندی‌سازی بلوکی تطبیق‌پذیر

الگوریتم فشرده‌سازی چندی‌سازی بلوکی تطبیق‌پذیر یک الگوریتم با عملکرد بلوکی مبتنی بر چندی‌سازی غیریکنواخت Lloyd-max است که در شکل (۱ عملکرد کدگذاری این الگوریتم آمده است.

شکل (۱): عملکرد کدگذاری BAQ

در تحلیل سیگنال‌ها، به‌ویژه پردازش سیگنال‌های مخابراتی، مؤلفه‌های همفاز (I)[3] و ربعی[4] (Q) نقش بسیار مهمی دارد. مؤلفه I به بخش حقیقی و مؤلفه Q به بخش موهومی سیگنال خروجی مدولاتور I/Q در گیرنده اطلاق می‌شود. این دو مؤلفه در واقع نشان‌دهنده روش تجزیه سیگنال به دو مؤلفه متعامد هستند که در بسیاری از کاربردهای مخابراتی و پردازشی مورد استفاده قرار می‌گیرند.

در کدگذاری الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر در ابتدا داده‌های ورودی الگوریتم را باید بررسی کرد. داده‌ها بصورت I و Q و دارای توزیع نرمال با میانگین صفر و واریانس مشخص بوده و واریانس داده‌های I و Q یکسان است.

با توجه به داده‌های I و Q هر پالس در ابتدا داده‌ها به بلوک‌های N تایی تبدیل می‌شود و برای هر بلوک N تایی انحراف معیار محاسبه می‌شود.

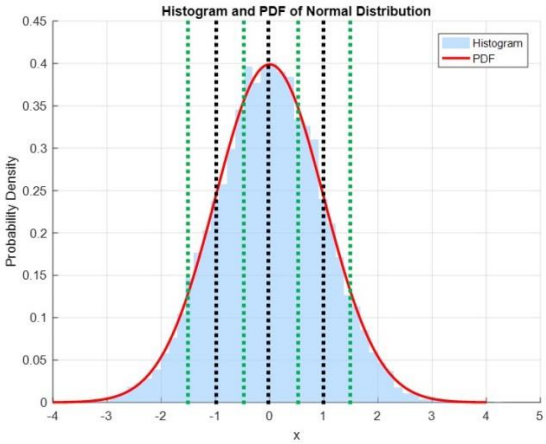
جهت چندی‌سازی داده‌های هر بلوک از رویکرد چندی‌سازی Lloyd-Max استفاده می‌شود. این چندی‌ساز، یک چندی‌ساز اسکالر غیریکنواخت است که بر اساس توزیع یک مجموعه داده x با تابع چگالی احتمال $f_x(x)$ سطوح بازسازی و مرزهای چندی‌سازی را به صورت مرحله به مرحله و با هدف کمینه‌کردن میانگین مربع خطا، تغییر می‌دهد. در نهایت سطوح و مرزهای چندی‌سازی به مقادیر بهینه همگرا می‌شود. عملکرد این الگوریتم برای یک مجموعه داده در شکل (۲ نمایش داده شده است.

[3] In-Phase

[4] Quadrature

شکل (۲): عملکرد چندی‌سازی Lloyd-Max

عملکرد این چندی‌ساز مبتنی بر داده و براساس هیستوگرام آن است. برای مثال اگر توزیع داده به صورت نرمال با میانگین صفر و واریانس ۱ باشد، مرزهای چندی‌سازی و سطوح بازسازی برای یک چندی‌ساز دو بیتی همانند شکل (۳) است.



شکل (۳): چندی‌ساز Lloyd-Max دوبیتی برای یک مجموعه داده با توزیع نرمال با میانگین صفر و واریانس یک

چندی‌ساز Lloyd-max مبتنی بر توزیع داده است و سطوح بازسازی و مرزهای چندی‌سازی طی یک‌رویه بازگشتی برای یک توزیع داده خاص به مقادیر بهینه همگرا می‌گردند. ازآنجاکه اساس این رویکرد بر مبنای رویه تکرارپذیر و بازگشتی استوار است، همگرایی این الگوریتم زمان‌بر است. لیکن اگر سطوح بازسازی و مرزهای چندی‌سازی برای یک توزیع با میانگین صفر و واریانس واحد در یک LUT[5] ذخیره گردد، آنگاه سطوح بازسازی و مرزهای چندی‌سازی برای یک توزیع نرمال با میانگین صفر و واریانس σ^2 از رابطه‌های (۱) و (۲) بدست می‌آید:

$$y_i|_{var=\sigma^2} = \sigma \times y_i|_{var=1} \quad (۱)$$

$$b_i|_{var=\sigma^2} = \sigma \times b_i|_{var=1} \quad (۲)$$

سطوح بازسازی را با y_i و مرزهای چندی‌سازی را با b_i نشان می‌دهیم و σ انحراف معیار مرتبط به بلوک داده می‌باشد.

با استفاده از انحراف معیار محاسبه‌شده برای هر بلوک Nنمونه‌ای و با کمک سطوح بازسازی و مرزهای چندی‌سازی بدست‌آمده از Lloyd-Max، مرزهای چندی‌سازی و سطوح بازسازی را باکمک رابطه‌های (۱) و (۲) بروزرسانی کرده و برروی داده‌های موجود در بلوک N نمونه‌ای چندی‌سازی اعمال و به هر سطح یک کد باینری تخصیص داده می‌شود. عملیات کدگشایی در شکل (۴ آمده است.

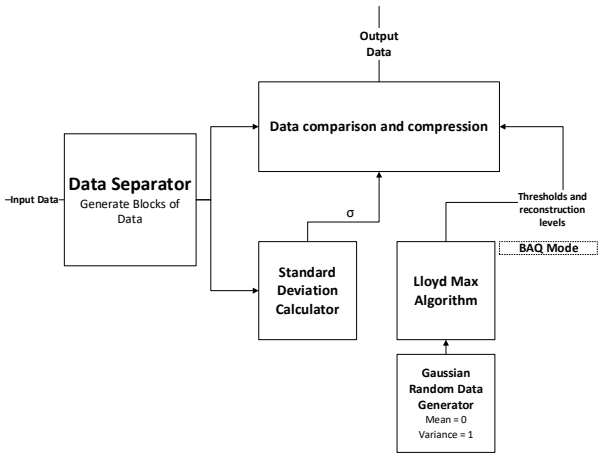
شکل (۴): عملکرد کدگشایی BAQ

در روند کدگشایی باتوجه به مقدار انحراف معیار هر بلوک N نمونه‌ای مقادیر چندی‌سازی شده به مقدار تقریبی اولیه قبل از چندی‌سازی برگشت داده می‌شود.

[5] Look Up Table

۲-۲- طراحی سیستمی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر

بلوک دیاگرام طراحی سیستمی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر در شکل (۵ آمده است.



شکل (۵): بلوک دیاگرام طراحی سیستمی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر

این طراحی از ۴ زیربلوک تشکیل شده است که توضیحات هر بخش در ادامه آمده است

۲-۲-۱- زیربلوک Data separator

این بخش وظیفه دارد داده‌های ورودی را به بلوک‌های ۱۲۸نمونه‌ای تقسیم کند و هر بلوک ۱۲۸نمونه‌ای را برای عملیات فشرده‌سازی به زیر بلوک‌های دیگر تحویل دهد.

علت آنکه طول بلوک‌ها برابر ۱۲۸ نمونه در نظر گرفته شده این است که درصورتی که تعداد نمونه‌ها کمتر از ۱۲۸ انتخاب شود خطای تخمین انحراف معیار افزایش می‌یابد (روابط (۳) و (۴) و توضیحات بعد از آن را ببینید). درصورتی که تعداد نمونه‌های هر بلوک را بیشتر از ۱۲۸ انتخاب کنیم سبب کاهش خطا می‌شود اما مصرف منابع پردازشی از جمله LUT ها افزایش پیدا می‌کند. با توجه به میزان خطا و مصرف منابع پردازشی موجود در FPGA باید مصالحه‌ای صورت پذیرد که با توجه به شبیه‌سازی‌ها و پیاده‌سازی‌های انجام شده مقدار ۱۲۸ انتخاب گردید.

ورودی زیربلوک Data separator با Input Data نمایش داده شده است که داده‌های ورودی سیستم طراحی‌شده می‌باشد.

۲-۲-۲- زیربلوک Standard Deviation Calculation

این بخش وظیفه دارد هر بلوک ۱۲۸نمونه‌ای را دریافت کرده و انحراف معیار آن را محاسبه کند و برای تصمیم‌گیری نهایی، انحراف معیار محاسبه‌شده را به زیربلوک بعدی گزارش دهد.

۲-۲-۳- زیربلوک Lloyd-Max Algorithm

این بخش وظیفه دارد با اجرای الگوریتم Lloyd-Max برروی داده‌های نرمال با میانگین صفر و واریانس واحد، سطوح بازسازی و مرزهای چندی‌سازی را محاسبه کند و در نهایت به زیربلوک بعد گزارش دهد. وظیفه تولید داده‌های ورودی الگوریتم Lloyd-Max برای محاسبه سطوح بازسازی و مرزهای چندی‌سازی را بلوک Gaussian Random Data Generator بر عهده دارد. و خروجی زیربلوک Lloyd-max شامل اعدادی است که سطوح بازسازی و مرزهای چندی‌سازی را نشان می‌دهد و این اعداد برای پیاده‌سازی سخت‌افزاری در حافظه‌ای ذخیره می‌شود. زیربلوک Gaussian Random Data Generator یک تولیدکننده داده می‌باشد که صرفا به عنوان ورودی الگوریتم Lloyd-Max بوده و ارتباطی با ورودی اصلی سیستم که Input Data می‌باشد ندارد.

۲-۲-۴- زیربلوک Data Comparison and Compression

این بخش به عنوان ورودی، داده‌های هر بلوک ۱۲۸نمونه‌ای به همراه انحراف معیارآن‌ها را دریافت کرده و به کمک سطوح بازسازی و مرزهای چندی‌سازی عملیات فشرده‌سازی را انجام می‌دهد.

جهت پیاده‌سازی الگوریتم BAQ با توجه به چندی‌سازی Lloyd-max در ابتدا داده مورد نظر به بلوک‌های Nنمونه‌ای تقسیم می‌شود و برای هر بلوک انحراف معیار را محاسبه کرده و با کمک انحراف معیار محاسبه‌شده، سطوح و مرزهای چندی‌سازی بروزرسانی می‌شود و در نهایت N نمونه موجود در هر بلوک را با توجه به بازه‌های مرزهای چندی‌سازی به یکی از سطوح چندی‌سازی تخصیص و سپس به هر کدام از سطوح چندی‌سازی یک کد باینری اختصاص داده می‌شود و عملیات کدگذاری انجام می‌شود.

جهت محاسبه انحراف معیار در این تحقیق از ایده پیشنهادی که در ادامه آمده استفاده شده است.

اگر متغیر تصادفی x که دارای توزیع گوسی با میانگین صفر است را در نظر بگیریم امید ریاضی اندازه این متغیر تصادفی در رابطه (۳) نشان داده شده است.

$$E\{|x|\} = 2\int_0^{\infty} x f_x(x)\,dx = \frac{2}{\sigma\sqrt{2\pi}}\int_0^{\infty} x\exp\left(\frac{-x^2}{2\sigma^2}\right)dx = \sqrt{\frac{2}{\pi}}\,\sigma \approx 0.8\,\sigma \quad (۳)$$

یک متغیر گوسی با میانگین و واریانس آن مشخص می‌شود و چون میانگین متغیر تصادفی x صفر می‌باشد و امید ریاضی اندازه x با انحراف معیار این متغیر تصادفی رابطه دارد پس می‌توان نتیجه گرفت که رابطه (۳) به طور کامل متغیر تصادفی x را توصیف می‌کند و می‌توان میانگین و انحراف معیار این متغیر تصادفی را مشخص کرد. برای محاسبه امید ریاضی اندازه x از رابطه (۴) استفاده می‌شود که به کمک آن می‌توان انحراف معیار هر بلوک Lنمونه‌ای از I و Q را محاسبه کرد.

$$\alpha = \frac{1}{L}\sum_{i=1}^{L}|x_i| \quad (۴)$$

رابطه (۴) تخمین $E\{|x|\}$ برای یک بلوک از داده‌های عملی به طول L می‌باشد و در نهایت از ارتباط α با نتیجه بدست‌آمده در رابطه (۳) که $0.8\,\sigma$ می‌باشد می‌توان انحراف معیار بلوک مورد نظر را محاسبه کرد. به عبارت دیگر خواهیم داشت $\sigma = \alpha/0.8$. L تعداد نمونه‌های بلوکی است که می‌خواهیم انحراف معیار آن را محاسبه کنیم. لازم به ذکر است که هرچه تعداد داده‌ها بیشتر باشد دقت محاسبه انحراف معیار نیز بیشتر می‌شود.

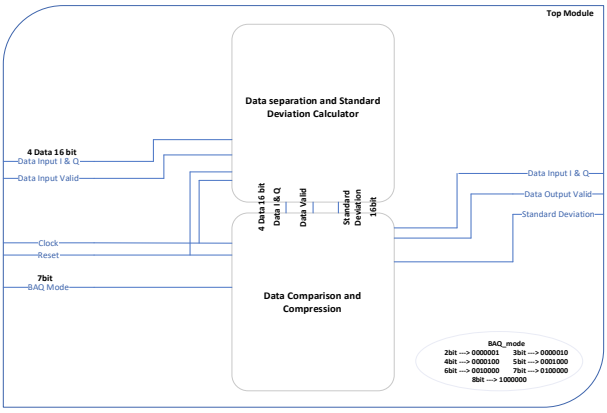
برای محاسبه انحراف معیار بصورت کلی از رابطه (۵) استفاده می‌شود.. اما برای کاهش پیچیدگی پیاده‌سازی، برای محاسبه انحراف معیار از روابط (۳) و (۴) استفاده شده است که این روش دارای خطای کمی می‌باشد (نتایج مقایسه دو روش در بخش ۳ و جدول ۱ آمده است).

$$\sigma = \sqrt{\frac{\sum(x-\mu)^2}{n}} \quad (۵)$$

۲-۳- طراحی و پیاده‌سازی سخت‌افزاری الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر

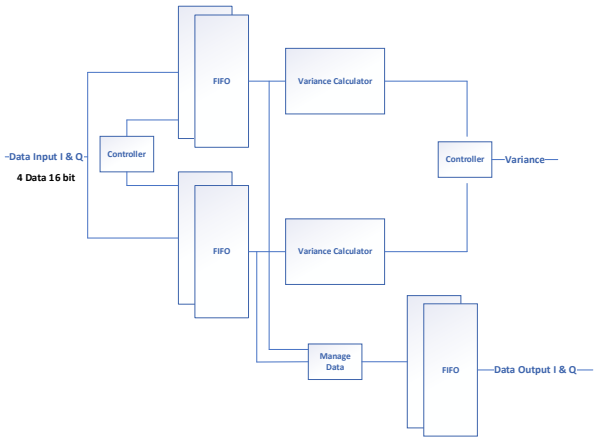
در بخش طراحی سیستمی پس از شبیه‌سازی‌های انجام‌شده در نرم‌افزار متلب و تحلیل آن‌ها، پیاده‌سازی سخت‌افزاری الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر بررسی می‌شود. برای پیاده‌سازی از سخت‌افزار قابل برنامه‌ریزی FPGA که دارای تراشه Virtex7 از شرکت Xilinx است، استفاده شده است. پیاده‌سازی در نرم‌افزار Vivado و به کمک زبان توصیف سخت‌افزار VHDL انجام شده است. در بخش سخت‌افزار فرض بر این است که از خروجی‌های ADC، ۴ داده همزمان I و Q دریافت می‌شود. همچنین هر ADC یک ساعت[۶] خروجی تحویل می‌دهد که داده I و Q همزمان با این ساعت به عنوان ورودی به تراشه تحویل داده می‌شود. نرخ ساعت دریافتی ۲۵۰ مگاهرتز است و به دلیل دریافت ۴ داده همزمان با هر ساعت، نرخ داده و پردازش روی آنها به اندازه ۱ گیگاهرتز است. هدف از طراحی، سخت‌افزار عملیات فشرده‌سازی داده به روش چندی‌سازی بلوکی تطبیق‌پذیر است. در این طراحی چندی‌سازی با عرض بیت‌های ۲ بیت تا ۸ بیت طراحی شده است که متناسب با نیاز کاربر توسط یک پورت ورودی قابل انتخاب است. ماژول اصلی طراحی سخت‌افزاری از دو زیرماژول تشکیل شده است. در ماژول اول داده‌ها به صورت بلوک‌های ۱۲۸ تایی جداسازی می‌شود و واریانس آن‌ها محاسبه می‌گردد و در ماژول دوم متناسب با عرض بیت فشرده‌سازی به کمک الگوریتم Lloyd_Max، مقادیر آستانه مقایسه و سطح‌های چندی‌سازی ایجاد می‌شوند و فشرده‌سازی انجام می‌شود. شکل (۶ بلوک دیاگرام ماژول اصلی طراحی‌شده را نشان می‌دهد. در این شکل، داده‌های ورودی به صورت ۴ داده همزمان I و ۴ داده همزمان Q در هر ساعت است. ورودی BAQMode مشخص‌کننده تعداد بیت‌های خروجی فشرده‌شده است که طبق شکل (۶ نوع انتخاب عرض بیت‌های مختلف مشخص شده است. در بخش پورت‌های خروجی نیز داده‌های فشرده‌سازی شده به صورت ۴ داده همزمان I و Q در هر ساعت به همراه واریانس مشخص شده است.

[۶] Clock



شکل (۶): طراحی سخت‌افزاری الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر

در بخش جداسازی N نمونه‌ای داده‌ها که در اینجا N برابر ۱۲۸ می‌باشد باید داده‌ها به صورت ۱۲۸تایی جداسازی شود و عملیات محاسبه واریانس روی آن‌ها انجام شود. در هر ساعت ۴ داده همزمان به سیستم وارد می‌شود، برای محاسبه واریانس به همه ۱۲۸ داده نیاز است تا بتوان مجموع قدر مطلق این داده‌ها را محاسبه کرد و با تقسیم حاصل جمع بر عدد ۱۲۸ واریانس محاسبه می‌شود. چالش موجود در پیاده‌سازی این است که داده‌ها به صورت زمان حقیقی و پشت سرهم وارد FPGA می‌شوند و بنابراین در حین جداسازی و محاسبه واریانس، داده‌های جدید وارد می‌شوند و دسترسی به داده‌های قبلی امکان‌پذیر نیست و به دلیل اینکه نیاز هست هر مجموعه ۱۲۸ تایی داده و واریانس مربوط به آن به صورت همزمان به ماژول بعدی ارسال شود، بنابراین باید از حافظه FIFO جانبی برای نگه‌داری داده‌ها استفاده کرد تا در حین محاسبه واریانس این داده‌ها ذخیره شوند و بعد از محاسبه واریانس، این داده‌ها به همراه واریانس مربوطه به ماژول بعدی منتقل شود و بعد از انتقال مجموعه داده مربوطه دوباره مقدار واریانس جدید متناسب با مجموعه داده ۱۲۸تایی جدید به بخش بعد منتقل شود. همچنین جمع‌کردن داده‌ها برای محاسبه واریانس به صورت accumulator انجام می‌شود، زیرا همه داده‌ها به صورت یکجا وارد سیستم نمی‌شود و در هر ساعت تنها ۴ داده در دسترس خواهد بود. برای اینکه داده‌ها به صورت ۱۲۸تایی جداسازی شود و همینطور در هنگام محاسبه واریانس هیچ داده جدیدی از دست نرود از دو FIFO موازی برای مدیریت جداسازی و نگه‌داری داده‌ها قبل از محاسبه واریانس نیز استفاده می‌شود. برای پیاده‌سازی FIFO از IP Core تعبیه‌شده در نرم‌افزار Vivado استفاده شده است. هر FIFO دارای عرض بیت ۶۴ و عمق ۳۲ خانه است که به اندازه ۱۲۸ داده ۱۶ بیتی در نظر گرفته شده است. همچنین برای داده‌های I و Q، FIFO های جداگانه در نظر گرفته شده است. ۴ داده همزمان که در هر لحظه قرار است به FIFO وارد شوند با یک‌دیگر در یک رشته ۶۴ بیتی قرار می‌گیرند و سپس آماده ورود به FIFO ها می‌شوند. این کار باعث می‌شود بهینه‌سازی سخت‌افزاری انجام شود و همینطور جانمایی مناسبی برای آن در سخت‌افزار توسط نرم‌افزار Vivado انجام شود. زیرا اگر برای هر لاین از ۴ داده همزمان، یک FIFO در نظر گرفته شود، هر کدام از FIFO ها ممکن است در جاهایی دور از یکدیگر جانمایی شوند. بنابراین هر ۴ داده همزمان، به صورت یک رشته وارد FIFOها می‌شود. شکل نشان‌دهنده بلوک دیاگرام پیاده‌سازی‌شده برای ماژول بخش جداساز و محاسبه واریانس است. مدیریت FIFO های موازی توسط یک بخش کنترلی انجام می‌شود که وظیفه آن این است که به مدت ۳۲ ساعت داده‌ها را دریکی از شاخه‌های FIFO های موازی وارد کند که به این کار عملیات نوشتن داده گفته می‌شود. در حین نوشتن در FIFO شاخه بالا برای اولین بار، FIFO شاخه پایین غیرفعال است و نه در حال نوشتن و نه در حال خواندن است. بعد از اینکه ۱۲۸ داده بعد از ۳۲ ساعت در FIFO شاخه بالا نوشته شد، عملیات نوشتن درون این شاخه غیرفعال شده و عملیات نوشتن در FIFO شاخه پایین فعال می‌شود و داده‌ها که به صورت زمان حقیقی وارد سیستم می‌شوند به مدت ۳۲ ساعت به شاخه پایین منتقل می‌شوند تا زمانی که ۱۲۸ داده به این حافظه وارد شود. زمانی که عملیات نوشتن در FIFO شاخه بالا غیر فعال است، می‌توان عملیات خواندن از آن را آغاز کرد. برای اینکه تداخلی در بخش کنترلی طراحی‌شده برای مدیریت جداسازی داده‌ها ایجاد نشود، عملیات خواندن از FIFO شاخه بالایی با سه ساعت تاخیر انجام می‌شود و همچنین تاخیر ذاتی خود FIFO از لحظه فعال‌سازی عملیات خواندن تا زمانی که اولین داده به خروجی آن منتقل شود، به اندازه ۳ ساعت است. مجموعه این تاخیر اولیه یک حاشیه امنیتی ایجاد می‌کند که مقدار واریانس برای ۱۲۸ داده که از شاخه بالا خوانده می‌شود، به درستی دریافت شود.



شکل (۷): زیر ماژول مدیریت داده و محاسبه انحراف معیار

بعد از شروع خواندن از FIFO بالایی، رشته ۶۴ بیتی داده تفکیک شده و به ۴ داده ۱۶ بیتی بازگردانده می‌شود. از داده‌ها قدر مطلق گرفته می‌شود و با یک‌دیگر جمع می‌شوند و این فرآیند تا ۳۲ ساعت که معادل خواندن ۱۲۸ داده است تکرار می‌شود، بعد از اینکه مجموع قدر مطلق کل داده‌ها محاسبه شد، عملیات تقسیم بر عدد ۱۲۸ انجام می‌شود تا مقدار واریانس برای این مجموعه ۱۲۸ تایی از داده‌ها محاسبه شود. بعد از ۳۲ ساعت خواندن از FIFO شاخه بالا و محاسبه واریانس آن، FIFO شاخه پایین نیز که داده‌های زمان حقیقی در آن نوشته شده است، آماده خواندن و محاسبه واریانس مجموعه ۱۲۸ تایی دوم است و FIFO شاخه بالا دوباره به حالت نوشتن تغییر کرده است. عملیات محاسبه واریانس برای شاخه پایین نیز تکرار می‌شود و این عملیات به صورت مداوم بین شاخه‌های بالا و پایین تکرار می‌شود.

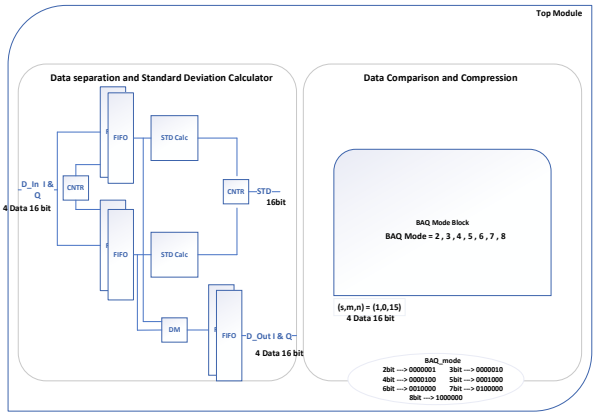
FIFO دیگری برای نگهداری داده‌های اصلی نیز در نظر گرفته شده است. وظیفه این FIFO نگهداری داده‌ها تا زمان آماده شدن واریانس مربوط به مجموعه ۱۲۸ تایی داده‌ها است. بعد از آماده‌شدن واریانس یک بیت فعال‌ساز خواندن برای FIFO نگهدار داده ارسال می‌شود و ۱۲۸ داده و واریانس آن‌ها در ۳۲ ساعت به ماژول بخش بعد منتقل می‌شود. این FIFO دارای عرض بیت ۶۴ و عمق ۶۴ خانه است که به اندازه ۲۵۶ داده ۱۶ بیتی در نظر گرفته شده است. بعد از ۳۲ ساعت مقدار جدید واریانس متناسب با ۱۲۸ داده بعدی به ماژول بخش دو ارسال می‌شود و این فرآیند ادامه خواهد داشت. برای عملیات نوشتن در FIFO نگهدارنده داده یک بخش کنترلی طراحی شده که در آن متناسب با فعال‌بودن عملیات خواندن هرکدام از شاخه‌های بالا یا پایین، رشته ۶۴ بیتی خروجی این شاخه‌ها بدون هیچ عملیاتی به صورت مستقیم به FIFO نگهدارنده ارسال می‌شوند و بعد از آماده‌شدن اولین واریانس و دریافت دستور خواندن، داده‌ها از FIFO به ماژول بخش دوم ارسال می‌شوند. عملیات انجام‌شده برای جداسازی مجموعه داده‌های ۱۲۸ تایی و همچنین محاسبه واریانس با نرخ ۴ برابر نرخ ساعت انجام می‌شود که این پیاده‌سازی از نرخ ساعت ۲۵۰ مگاهرتز استفاده می‌کند و بنابراین نرخ کاری کل سخت‌افزار پیاده‌سازی شده برابر ۱ گیگا هرتز است.

در زیر سیستم مدیریت داده و محاسبه انحراف معیار واریانس داده حساب می‌شود. واریانس محاسبه‌شده به همراه ۴ داده همزمان I و ۴ داده همزمان Q به ماژول جدید وارد می‌شوند. در این ماژول هدف این است که ابتدا مرزهای متناسب با عرض بیت فشرده‌سازی انتخاب‌شده محاسبه شوند که این کار به کمک واریانس به دست‌آمده در مرحله قبل و ضرایب ثابتی از جدول Lloyd-Max انجام می‌شود. در مرحله بعد عملیات مقایسه داده اصلی با مرزهای محاسبه‌شده انجام می‌شود. برای هر نوع فشرده‌سازی با عرض بیت‌های مختلف تعداد مرزها متفاوت است. تعداد مرزها در هر نوع فشرده‌سازی از رابطه (۶) محاسبه می‌شود.

$$thershold\_num = 2^{compression\_bit\_Width} - 1 \quad (۶)$$

در رابطه (۶) مقدار compression_bit_width برابر با تعداد بیت خروجی عملیات فشرده‌سازی است که توسط کاربر انتخاب می‌شود و قابلیت انتخاب از دو بیت تا هشت بیت را دارد. مقدار threshold_num، خروجی به‌دست‌آمده از رابطه (۶) است که به معنی تعداد مرزهای ایجادشده متناسب با تعداد بیت خروجی فشرده‌سازی شده است. برای فشرده‌سازی با عرض ۲ بیت، ۳ عدد ترشولد وجود دارد که داده‌های روی نمودار را به ۴ سطح تقسیم می‌کند. سپس پایین‌ترین سطح به عدد (۲-) باینری، سطح داده بعدی به عدد (۱-) باینری، سطح سوم به عدد (۰) باینری و درنهایت سطح آخر نیز به عدد (۱) باینری کد می‌شوند و به این ترتیب هر داده متناسب با سطحی که قرار می‌گیرد، در دو بیت فشرده‌سازی می‌شود. برای فشرده‌سازی با عرض بیت‌های دیگر نیز همین روال وجود دارد. به عنوان مثال برای فشرده‌سازی به صورت ۸ بیتی، ۲۵۵ مرز وجود خواهد داشت که با توجه به ۱۶ بیتی بودن داده‌های I و Q، مرزها در محدوده [۳۲۷۶۷ , ۳۲۷۶۸-] قرار خواهد گرفت و با توجه به مقادیر واریانس برای هر مجموعه ۱۲۸ تایی و جدول Lloyd-Max، مقادیر داده‌های قرارگرفته بین مرزها به اعداد [۱۲۷،

۱۲۸-] فشرده‌سازی خواهد شد. در بخش پورت‌های خارجی این طراحی داده‌های فشرده‌سازی شده I و Q به همراه اطلاعات مربوط به واریانس وجود خواهد داشت تا برای عملیات عکس فشرده‌سازی در گیرنده بتوان از آن استفاده کرد. شکل (۷ طراحی نهایی سخت‌افزار الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر را نشان می‌دهد.



شکل (۷): طراحی نهایی سخت‌افزار الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر

۳- نتایج و تحلیل

شبیه‌سازی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر در نرم‌افزار متلب جهت طراحی سیستمی الگوریتم و صحت‌سنجی انجام شده است. داده‌های ورودی سیستم طراحی‌شده در نرم‌افزار متلب بصورت داده‌هایی با توزیع نرمال با میانگین صفر و واریانس ۱ می‌باشد. در این شبیه‌سازی چندی‌سازی Lloyd-Max مورد بررسی قرارگرفته است و خروجی شبیه‌سازی شامل مرزهای چندی‌سازی و سطوح بازسازی با توجه به مقدار Mode انتخاب‌شده می‌باشد. نتایج شبیه‌سازی محاسبه انحراف معیار از روش پیشنهادی در رابطه (۳) و (۴) و مقایسه آن با روش اصلی محاسبه انحراف معیار در رابطه (۵) در

جدول (۱ آمده است. داده استفاده شده به عنوان ورودی سیستم، داده با توزیع نرمال با میانگین صفر و واریانس ۴ است. تعداد نمونه‌های موجود در این شبیه‌سازی ۲۰۴۸ نمونه می‌باشد. عملیات scaling برروی داده‌های ورودی انجام شده است تا بین ۱ و ۱- قرار بگیرند. نمونه‌های داده ورودی (۲۰۴۸ نمونه) در ابتدا به بلوک‌های ۱۲۸ نمونه‌ای تقسیم شده و هر بلوک وارد بخش محاسبه انحراف معیار شده و مقدار آن محاسبه می‌شود. بنابراین، ۱۶ بلوک ۱۲۸ نمونه‌ای خواهیم داشت که نتایج محاسبه انحراف معیار در ادامه آمده است. با بررسی اعداد بدست‌آمده میزان اختلاف رابطه پیشنهادی محاسبه انحراف معیار و رابطه اصلی برای هر ۱۲۸ نمونه حداکثر ۰/۰۱۲ می‌باشد. در نتیجه، حداکثر اختلاف محاسبه انحراف معیار با استفاده از رابطه‌های پیشنهادی و اصلی کمتر از ۲ درصد خواهد بود.

جدول (۱): محاسبه انحراف معیار با استفاده از رابطه (۳) ، (۴) و (۵)

اختلاف روش اصلی و پیشنهادی	انحراف معیار (روش پیشنهادی)	انحراف معیار (روش اصلی)
۰/۰۰۳۰	۰/۳۱۵۷	۰/۳۱۸۷
۰/۰۰۷۸	۰/۲۴۲۲	۰/۲۵۰۰
۰/۰۰۶۳	۰/۲۷۸۰	۰/۲۸۴۳
۰/۰۰۰۲	۰/۲۵۱۰	۰/۲۵۰۸
۰/۰۰۰۷	۰/۲۵۷۵	۰/۲۵۸۳
۰/۰۰۳۶	۰/۲۷۵۸	۰/۲۷۲۱
۰/۰۰۶۰	۰/۲۸۲۴	۰/۲۷۶۳
۰/۰۰۵۲	۰/۲۶۷۳	۰/۲۷۲۶
۰/۰۰۵۶	۰/۲۹۸۰	۰/۲۹۲۳
۰/۰۰۰۰	۰/۲۸۸۳	۰/۲۸۸۳
۰/۰۱۱۱	۰/۲۳۷۵	۰/۲۴۸۶
۰/۰۰۲۲	۰/۲۶۳۷	۰/۲۶۵۹
۰/۰۱۲۰	۰/۲۷۹۶	۰/۲۹۱۶
۰/۰۰۴۹	۰/۲۷۴۹	۰/۲۷۹۹
۰/۰۰۳۱	۰/۲۶۵۲	۰/۲۶۲۱
۰/۰۰۱۷	۰/۲۷۷۷	۰/۲۷۵۹

اختلاف در محاسبه انحراف معیار سبب ایجاد خطا در فشرده‌سازی می‌شود. جهت بررسی این خطا الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر را به شرح زیر با دو روش شبیه‌سازی می‌کنیم:

- روش اول: شبیه‌سازی الگوریتم با استفاده از محاسبه انحراف معیار با رابطه اصلی
- روش دوم: شبیه‌سازی الگوریتم با استفاده از محاسبه انحراف معیار با رابطه پیشنهادی

معیار های ارزیابی مورد بررسی MSE[۷]، PSNR[۸]، SSIM[۹] و CC[۱۰] می‌باشد که یکبار برای داده فشرده‌سازی‌شده با استفاده از روش اول محاسبه می‌شودکه با MSE1، PSNR1، SSIM1 و CC1 نمایش می‌دهیم و یکبار برای داده فشرده‌سازی شده با استفاده از

[7] **Mean Square Error**

[8] **Peak Signal-to-Noise Ratio**

[9] **Structural Similarity Index Measure**

[10] **Cross-Correlation**

روش دوم محاسبه می‌شود که با MSE2، PSNR2، SSIM2 وCC2 نمایش می‌دهیم. لازم به ذکر است که این مقادیر خطا می‌بایست برای مقادیر مختلف Mode فشرده‌سازی محاسبه گردد. نتایج درجدول (۲، جدول (۳، جدول (۴ و

جدول (۵ آمده است. باید توجه کرد هنگام محاسبه MSE از ۲۰۴۸ نمونه فشرده‌شده و ۲۰۴۸ نمونه اصلی استفاده شده است.

جدول (۲): معیار MSE با توجه به Mode های مختلف با توجه به دو رویکرد محاسبه انحراف معیار

Mode	MSE1	MSE2
۲	۰/۰۴۲۱	۰/۰۴۲۳
۳	۰/۰۳۲۳	۰/۰۳۲۵
۴	۰/۰۲۵۱	۰/۰۲۵۳
۵	۰/۰۱۹۴	۰/۰۱۹۶
۶	۰/۰۱۲۵	۰/۰۱۲۷
۷	۰/۰۱۲۰	۰/۰۱۲۲
۸	۰/۰۱۲۰	۰/۰۱۲۲

جدول (۳): معیار PSNR با توجه به Mode های مختلف با توجه به دو رویکرد محاسبه انحراف معیار

Mode	PSNR1	PSNR2
۲	۱۳/۷۵۵۰	۱۳/۷۳۶۴
۳	۱۴/۹۰۵۷	۱۴/۸۷۷۵
۴	۱۵/۹۹۸۹	۱۵/۹۶۰۹
۵	۱۷/۱۲۱۸	۱۷/۰۷۲۹
۶	۱۸/۹۹۹۹	۱۸/۹۳۰۸
۷	۱۹/۱۷۲۷	۱۹/۱۰۱۳
۸	۱۹/۱۷۳۰	۱۹/۱۰۱۷

جدول (۴): معیار SSIM با توجه به mode های مختلف با توجه به دو رویکرد محاسبه انحراف معیار

Mode	SSIM1	SSIM2
۲	۰/۲۷۵۴	۰/۲۷۳۳
۳	۰/۴۰۷۱	۰/۴۰۴۸
۴	۰/۵۰۵۳	۰/۵۰۰۶
۵	۰/۵۹۸۹	۰/۵۹۶۷
۶	۰/۷۱۶۲	۰/۷۱۳۲
۷	۰/۷۲۴۳	۰/۷۲۱۸
۸	۰/۷۲۴۹	۰/۷۲۱۷

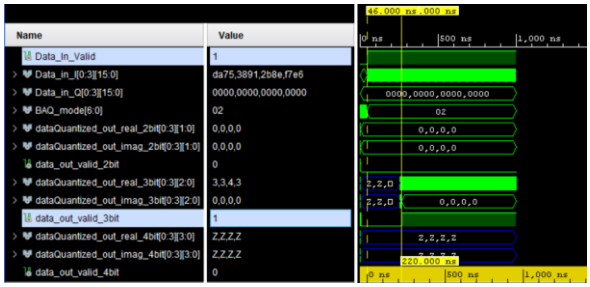
جدول (۵): معیار CC با توجه به Mode های مختلف با توجه به دو رویکرد محاسبه انحراف معیار

Mode	CC1	CC2
۲	۰/۸۴۸۸	۰/۸۴۸۵
۳	۰/۸۸۱۲	۰/۸۸۰۳
۴	۰/۹۰۲۹	۰/۹۰۱۶
۵	۰/۹۲۰۹	۰/۹۱۹۹
۶	۰/۹۴۳۷	۰/۹۴۲۹
۷	۰/۹۴۵۷	۰/۹۴۴۸
۸	۰/۹۴۵۷	۰/۹۴۴۸

بعد از شبیه‌سازی الگوریتم در نرم‌افزار متلب می‌توان الگوریتم را در نرم‌افزار Vivado با زبان VHDL بصورت سخت‌افزاری پیاده‌سازی کرد.

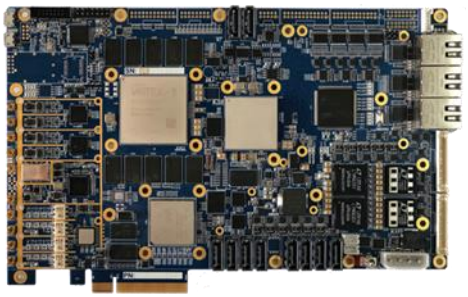
با مقایسه خروجی شبیه‌ساز نرم‌افزار Vivado و خروجی مدل Fixed Point نرم‌افزار متلب نتایج یکسان است. پس الگوریتم پیاده‌سازی‌شده با زبان سخت‌افزار نتایج مشابهی با مدل متلب دارد.

در شبیه‌ساز نرم‌افزار Vivado مقدار تاخیر اولیه سیستم اندازه‌گیری شده که نتیجه در شکل (۸ ارائه شده است. ساعت در نظر گرفته‌شده ۲۵۰ مگاهرتز بوده و Mode فشرده‌سازی ۳ در نظر گرفته شده است. تاخیر اولیه سیستم با توجه به ساعت ۲۵۰ مگاهرتز ۲۲۰ نانوثانیه است وکه این تاخیر اولیه فاصله بین یک‌شدن سیگنال data in valid و data out valid است.



شکل (۸): محیط شبیه‌سازی در محیط نرم افزار VIVADO

الگوریتم پیاده‌سازی به زبان سخت‌افزار با استفاده از نرم‌افزار Vivado برروی کارت پردازشی قرار می‌گیرد و می‌توان الگوریتم را بصورت عملی برروی FPGA موردنظر پیاده و تست کرد. کارت پردازشی مورداستفاده در این پروژه AVA7V19 است که از FPGA با مشخصات xc7vx690tffg1157-2 استفاده می‌کند که در شکل (۹ آمده است.

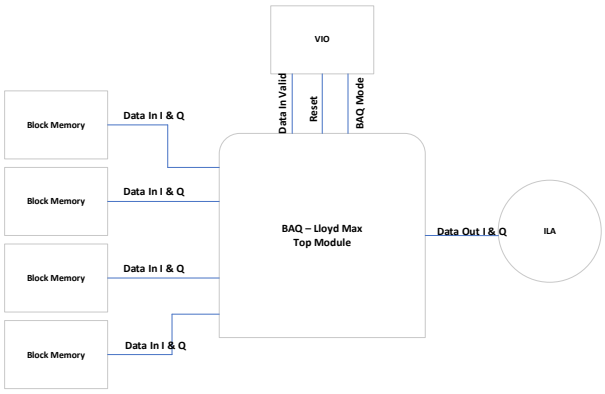


شکل (۹): کارت پردازشی AVA7V19

جهت تست الگوریتم باید ابتدا یک سناریو تست طراحی کنیم. بلوک دیاگرام سناریو تست در شکل (۱۰ آمده است. در این سناریو داده‌ها بصورت ۴ لاین وارد سیستم می‌شوند. بنابراین، از ۴ بلوک حافظه استفاده شده است که در ساعت یک داده را می‌خوانند. به این ترتیب، در هر ساعت ۴ داده وارد سیستم طراحی‌شده می‌شود.

نرم‌افزار Vivado که برای طراحی و پیاده‌سازی FPGA استفاده می‌شود، ابزارهای ILA[11] و VIO[12] برای مشاهده و تجزیه و تحلیل دقیق‌تر عملکرد مدار به کار می‌روند. ILA به عنوان یک آنالیزور منطق داخلی، این امکان را به طراحان می‌دهد که سیگنال‌های داخلی مدار FPGA را در زمان واقعی مشاهده کنند. این ابزار مشابه یک اسیلوسکوپ دیجیتال عمل کرده و به شما کمک می‌کند تا وضعیت سیگنال‌ها و تغییرات آن‌ها را بررسی کرده و مشکلات یا ایرادات طراحی را شناسایی کنید. از سوی دیگر، VIO به عنوان یک ابزار مجازی برای ورودی/خروجی، این امکان را فراهم می‌آورد که سیگنال‌های ورودی و خروجی FPGA را در زمان واقعی تغییر داده و مانیتور کنید. همچنین VIO به طراحان این آزادی را می‌دهد که بدون نیاز به توقف یا بازسازی طراحی، مقادیر سیگنال‌ها را در حین اجرای سیستم تغییر دهند. این دو ابزار مکمل یکدیگر بوده و در کنار هم به طراحان کمک می‌کنند تا با دقت بیشتر و سرعت بالاتر مشکلات طراحی را شناسایی و اصلاح کنند.

جهت مقداردادن به ورودی‌های دیگر سیستم از VIO و برای ذخیره خروجی‌های سیستم از ILA استفاده شده است.



شکل (۱۰): بلوک دیاگرام تست عملیاتی بر روی کارت پردازشی

در این سناریو از همان داده‌های ورودی استفاده‌شده در شبیه‌سازی بهره گرفته شده است. با بررسی خروجی‌ها مشاهده می‌شود که خروجی تست عملی سیستم با شبیه‌سازی نرم‌افزار Vivado و مدل Fixed point نرم‌افزار متلب یکسان است.

لازم به ذکر است که پیاده‌سازی سخت‌افزاری فوق بصورتی است که کمترین منابع پردازشی را مصرف کرده است و طرح بصورت پرسرعت پیاده‌سازی شده است. با توجه به سناریو تست عملی، منابع مصرفی این طرح برروی کارت پردازشی در جدول (۶ ارائه شده است.

در FPGA ، منابع مختلفی برای انجام عملیات‌های مختلف وجود دارد که هرکدام نقش ویژه‌ای در عملکرد این تراشه‌ها ایفا می‌کنند. از جمله این منابع می‌توان به DSP48 ، LUT[13]، BRAM و Flip-Flop اشاره کرد.

منابع DSP48 بلوک‌های پردازش دیجیتال سیگنال هستند که برای انجام عملیات‌های ریاضی پیچیده مانند ضرب و جمع و نیز پردازش‌های مربوط به سیگنال‌های دیجیتال طراحی شده‌اند. این واحدها معمولاً شامل ضرب‌کننده‌های ۱۸ در ۱۸ و جمع‌کننده‌ها می‌باشند و می‌توانند عملیات‌های مختلف مانند فیلتر کردن و پردازش سیگنال‌های پیچیده را با سرعت بالا انجام دهند.

LUT یا جدول جست‌وجو، به‌عنوان ابزاری برای پیاده‌سازی توابع منطقی درFPGA ها استفاده می‌شود. این جداول می‌توانند هر تابع منطقی دلخواه را بر اساس ورودی‌های آن پیاده‌سازی کنند. در حقیقت، LUTها به‌عنوان بلوک‌های اصلی برای پیاده‌سازی

[11] Integrated Logic Analyzer
[12] Virtual Input/Output
[13] Look-Up-Table

گیت‌های منطقی ترکیبی مانند AND ، OR، XOR و غیره در FPGAها عمل می‌کنند.

BRAM یا حافظه بلوکی، واحدهای حافظه هستند که برای ذخیره‌سازی داده‌ها و برنامه‌ها به‌طور موقت استفاده می‌شوند. این منابع حافظه معمولاً به‌صورت بلوک‌هایی با اندازه‌های مختلف در FPGAها موجود هستند و برای ذخیره داده‌های محلی و پشتیبانی از پردازش‌هایی که نیاز به حافظه زیادی دارند، بسیار کارآمد هستند.

در نهایت، Flip-Flopها واحدهای ذخیره‌سازی دیجیتال هستند که می‌توانند یک بیت داده را ذخیره کنند. این منابع به‌طور گسترده برای ذخیره وضعیت‌ها و اطلاعات در سیستم‌های دیجیتال استفاده می‌شوند و نقش حیاتی در سنکرون‌سازی عملیات و مدیریت سیگنال‌های کنترلی دارند.

استفاده بهینه از این منابع در FPGA ها موجب افزایش کارایی و سرعت پردازش در سیستم‌های پیچیده می‌شود و این تراشه‌ها را به ابزاری مناسب برای پیاده‌سازی انواع الگوریتم‌ها و پردازش‌های موازی تبدیل می‌کند.

جدول (۶): منابع مصرفی در پیاده‌سازی الگوریتم در سخت‌افزار

منبع پردازشی	درصد مصرف منبع پردازشی
DSP48 Block	۱۰
Flip Flop	۱
LUT	۳۰
BRAM	۱

نرم‌افزار Vivado بعد از کدنویسی و پیاده‌سازی سخت‌افزار گزارش‌هایی در مورد تأخیر اولیه سیستم تا زمان رسیدن اولین مقدار خروجی، مقدار منابع مصرفی و نرخ قابل‌دستیابی برای کارکرد صحیح ارائه می‌دهد. باتوجه‌به گزارش نرم‌افزار، مقدار تأخیر اولیه سیستم ۲۲۰ نانوثانیه است، به این معنی که از زمان دریافت اولین داده‌های ورودی تا زمان ارسال اولین داده‌های فشرده‌سازی، تنها ۲۲۰ نانوثانیه زمان لازم است و بعد از این زمان سیستم پیاده‌سازی‌شده به صورت زمان حقیقی و به طور مداوم خروجی‌های فشرده‌سازی‌شده را تحویل می‌دهد. لازم به ذکر است که این تاخیر به دست‌آمده بر واحد پریود ساعت محاسبه شده است که با در نظرگرفتن نرخ ساعت ۲۵۰ مگاهرتز، دوره تناوب ساعت برابر ۴ نانو ثانیه خواهد بود. گزارش مهم دیگر، پیاده‌سازی بدون مشکل با نرخ کاری ۲۵۰ مگاهرتز است که با توجه به این گزارش این نیازمندی پوشش داده شده است و سخت‌افزار با این نرخ کاری به مشکل برنخواهد خورد. منابع مصرفی گزارش‌شده در جدول (۶ آمده است. بر اساس این گزارش منابع DSP48 مصرف‌شده به اندازه ۱۰ درصد است. DSP48 از منابع اصلی و پرسرعت پیاده‌سازی عملیات جمع و ضرب می‌باشد که در سخت‌افزارهای شرکت Xilinx مانند Virtex7 به صورت پیش‌فرض برای عملیات پیچیده ریاضی تعبیه شده است. LUT ها از منبع‌های اصلی تراشه هستند که معمولاً هر سیگنالی که درون سخت‌افزار تعریف می‌شود از جنس LUT خواهد بود و می‌توان با طراحی‌های انجام‌شده، عملیات ریاضی یا عملیات کنترلی موردنظر را روی آن‌ها انجام داد. مقدار مصرف‌شده برای LUT نزدیک به ۳۰ درصد گزارش شده است. برای Flip Flop مقدار ۱ درصد گزارش شده است که به صورت سنکرون با ساعت پیاده‌سازی می‌شود و برای عملیات کنترلی مناسب است. مقدار Block RAM مصرفی نیز به اندازه ۱ درصد است که نشان می‌دهد پیاده‌سازی به شکل بهینه انجام شده است، زیرا برای نگهداری داده‌ها به خصوص در بخش جداسازی و نگهداری برای محاسبه واریانس از حداقل حافظه استفاده شده است.

٤- نتیجه گیری

این پژوهش به طراحی و پیاده‌سازی الگوریتم چندی‌سازی بلوکی تطبیق‌پذیر (BAQ) بر روی سخت‌افزار FPGA پرداخته است و توانسته با بهره‌گیری از تکنیک‌هایی مانند موازی‌سازی و بهینه‌سازی معماری، کارایی و بهره‌وری بالایی ارائه دهد. هدف اصلی این طرح، کاهش مصرف منابع سخت‌افزاری و به حداقل رساندن تأخیر زمانی در کنار دستیابی به سرعت پردازش بالا بوده است. نتایج نشان می‌دهد که سیستم طراحی‌شده توانسته است سرعت پردازش ۱ گیگاهرتز را به همراه تأخیر اولیه ۲۲۰ نانوثانیه ارائه دهد، که این ویژگی آن را برای کاربردهای بلادرنگ ایده‌آل می‌سازد. علاوه بر این، کاهش مصرف منابع FPGA، شامل تنها ۱۰ درصد از منابع DSP48 و ۳۰ درصد از LUTها، نشان‌دهنده بهره‌وری بالای این طرح است. همچنین، معیارهای ارزیابی نظیر MSE، PSNR، SSIM و CC تأیید می‌کنند که الگوریتم پیشنهادی دقت و عملکرد مطلوبی را در فشرده‌سازی داده‌ها حفظ کرده است. این طراحی، با ترکیب سرعت پردازش بالا، مصرف بهینه منابع، و دقت مناسب، می‌تواند در زمینه‌های مختلفی نظیر سامانه‌های تصویربرداری، مخابرات، و

پردازش سیگنال مورد استفاده قرار گیرد و الگویی مؤثر برای توسعه راهکارهای سخت‌افزاری مشابه ارائه کند.

تعارض منافع

"هیچ‌گونه تعارض منافع توسط نویسندگان بیان نشده است."

تشکر و قدردانی

بدین‌وسیله نویسندگان مراتب سپاس و قدردانی صمیمانه خود را از تمامی افرادی و سازمان‌هایی که در انجام این پژوهش و تهیه مقاله حاضر نقش داشته‌اند، اعلام می‌دارند. از دانشکده مهندسی برق دانشگاه صنعتی شیراز به‌دلیل فراهم‌سازی محیط علمی، امکانات پژوهشی و حمایت‌های علمی لازم در مراحل مختلف این پژوهش صمیمانه تشکر می‌کنیم. همچنین، از پژوهشکده مکانیک پژوهشگاه فضایی ایران در شیراز به‌دلیل فراهم‌کردن بستر مناسب برای انجام فعالیت‌های پژوهشی و فنی و حمایت‌های ارزشمند همکاران این مجموعه قدردانی می‌شود.

نویسندگان همچنین از راهنمایی‌ها، پیشنهادهای سازنده و همکاری‌های علمی ارزشمند اساتید، پژوهشگران و همکارانی که به‌طور مستقیم یا غیرمستقیم در شکل‌گیری، توسعه و تکمیل این پژوهش مشارکت داشته‌اند، سپاسگزاری می‌کنند. بدون حمایت‌های علمی، فنی و سازمانی این عزیزان، انجام این پژوهش و دستیابی به نتایج ارائه‌شده امکان‌پذیر نبود.

مراجع

[1] M. Hatam, A. Liaqat, N. Mardaneh, and M. Hatam, “Analysis of the efficiency of adaptive block quantization in raw synthetic aperture radar data compression using ISRCSAR indigenous radar raw data and presentation of an adaptive rate selection approach,” *Radar Journal*, vol. 7, no. 2, pp. 99–109, 2019.

[2] C. D'Elia, G. Poggi, and L. Verdoliva, "Compression of SAR raw data through range focusing and variable-rate trellis-coded quantization," *IEEE transactions on image processing*, vol. 10, no. 9, pp. 1278-1287, 2001.

[3] T. Algra, “Data compression for operational SAR missions using entropy-constrained block adaptive quantisation,” in Proc. *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2, 2002, pp. 1135–1139.

[4] J. Hamkins and K. Zeger, "Gaussian source coding with spherical codes," *IEEE Transactions on Information Theory*, vol. 48, no. 11, pp. 2980-2989, 2002.

[5] R. Kwok and W. T. Johnson, "Block adaptive quantization of Magellan SAR data," *IEEE Transactions on Geoscience and Remote sensing*, vol. 27, no. 4, pp. 375-383, 1989.

[6] V. Pascazio and G. Schirinzi, "SAR raw data compression by subband coding," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 964-976, 2003.

[7] R. C. Huxtable, "SAR raw data compression using FPGA technology," 2008.

[8] T. Jiang, C. Zhang, F. Zhang, Y. Wan, and L. Chen, "A Multichannel, Multipulse, Multiweight Block-Adaptive Quantization (3MBAQ) Algorithm Based on Space-Borne Multichannel SAR Doppler Domain A-BAQ," *Remote Sensing*, vol. 16, no. 3, Art. no. 477, 2024.

[9] W. Ji, X. Qiu, X. Wen, and L. Huang, "An Improved BAQ Encoding and Decoding Method for Improving the Quantized SNR of SAR Raw Data," *Sensors*, vol. 18, no. 12, Art. no. 4221, 2018.

[10] D. A. Devi, "Compression of Sar Raw Data Using Block Adaptive Quantization," 2018.

[11] D. Gleich, P. Planinsic, B. Gergic, and Z. Cucej, "Progressive space frequency quantization for SAR data compression," *IEEE transactions on geoscience and remote sensing*, vol. 40, no. 1, pp. 3-10, 2002.

[12] E. Magli and G. Olmo, "Lossy predictive coding of SAR raw data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 41, no. 5, pp. 977-987, 2003.

[13] S. Mohseni, M. Nayebi, R. Mohseni, and B. Ebrahimi, "Introducing Azimuth FFT Architecture in Range-Doppler Imaging Algorithm in SAR Systems," 2015.

[14] M. Knioła, A. Kawalec, C. Leśnik, and M. Szugajew, “Implementation of block adaptive quantizer as a peripheral module for the FPGA-based SAR system,” in Proc. *18th International Radar Symposium (IRS)*, 2017, pp. 1–9.

[15] L. Sheng, T. Y. Zheng, and X. J. Zhang, “Design and implementation of SAR raw data BAQ based on FPGA,” in Proc. *1st Asian and Pacific Conference on Synthetic Aperture Radar (APSAR)*, 2007, pp. 664–666.

[16] W. Li, L. Chen, and X. Q. Cheng, “Design and implementation of the fixed-point BAQ decompression algorithm based on FPGA,” in Information, *Computer and Application Engineering, CRC Press*, 2018, pp. 21–26.