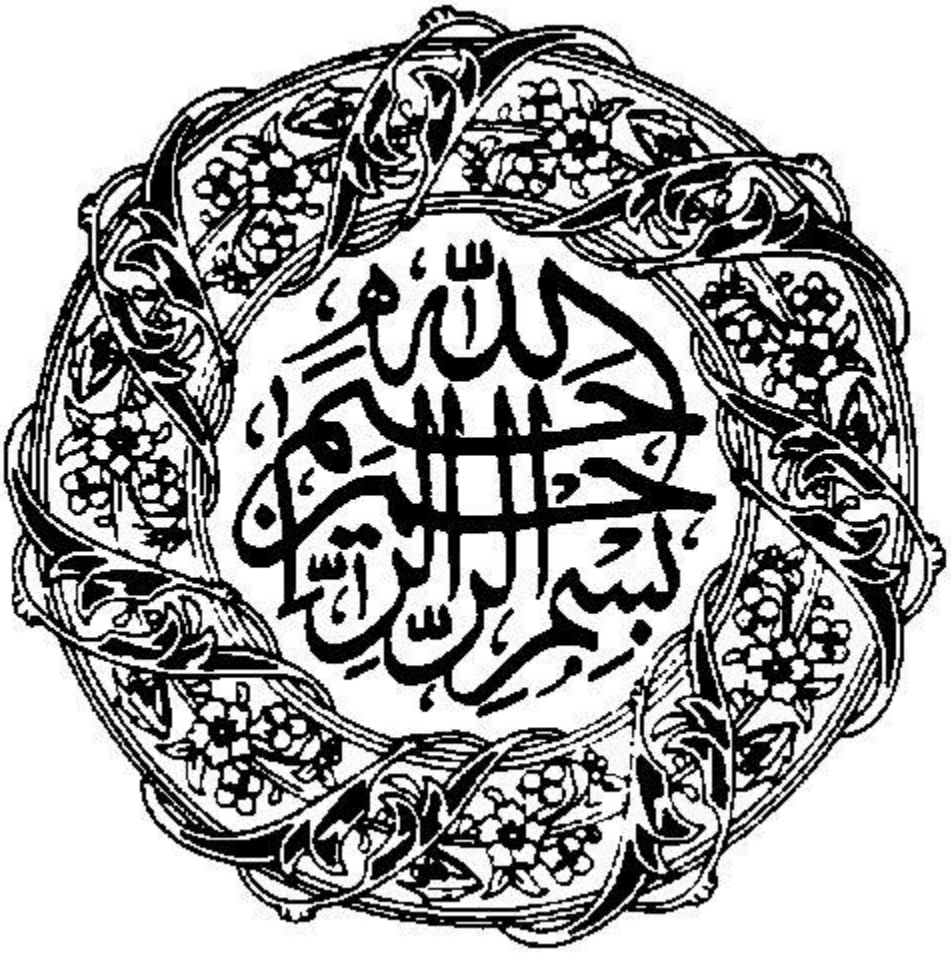

بسم الله الرحمن الرحيم

**دانشکده مهندسی برق و رباتیک**
**رشته مهندسی برق گرایش الکترونیک**

**پایان‌نامه کارشناسی ارشد**

# طراحی و پیاده‌سازی سیستم رفع نویز برای تصاویر دیجیتالی بر مبنای FPGA

**نگارنده: مسلم خانه‌بابائی**

استاد راهنما
**دکتر علی سلیمانی‌ایوری**

استاد مشاور
**دکتر محمدرضا اشرف**

**شهریور ۱۳۹۵**

**این مجموعه را با سپاس بی‌کران تقدیم می نمایم به:**

**محضر ارزشمند پدر و مادر عزیزم، به خاطر همه‌ی تلاش‌های محبت‌آمیزی که در دوران تحصیلم انجام داده‌اند و با مهربانی چگونه زیستن را به من آموخته‌اند. باکمال افتخار دستشان را می‌بوسم.**

**به برادر عزیزم که همواره در طول تحصیل متحمل زحماتم بود و وجودش مایه دلگرمی من هست.**

**به خواهران عزیزم، به‌پاس عاطفه سرشار و گرمای امیدبخش وجودشان که در این روزگاران بهترین پشتیبانم هستند.**

## تشکر و قدردانی

**از زحمات بی‌دریغ استاد ارجمند جناب آقای دکتر علی سلیمانی‌ایوری، دانشیار دانشکده مهندسی برق و رباتیک که راهنمایی این پایان‌نامه را بر عهده داشتند، کمال تشکر را دارم و توفیقات روز افزونشان را از خداوند متعال خواستارم.**

**از جناب آقای دکتر محمدرضا اشرف، استادیار دانشکده مهندسی برق و رباتیک که در این پروژه کمال همکاری را با اینجانب داشتند، صمیمانه تشکر و قدردانی می‌کنم.**

**از کلیه اساتید دانشکده مهندسی برق و رباتیک به خاطر تمامی زحماتشان در طول دو سال تحصیلم در دانشگاه صنعتی شاهرود، کمال تشکر را دارم.**

**از شرکت توسعه و طراحی رایان صنعت (گروه پازج) واقع در شهرک علمی-تحقیقاتی اصفهان به خاطر تمامی راهنمایی‌هایشان تشکر می‌کنم.**

**سپاس ویژه از پدر و مادر عزیز و مهربانم که دعای خیر آن‌ها همیشه بدرقه راهم بوده است.**

# تعهد نامه

اینجانب **مسلم خانه‌بابائی** دانشجوی مقطع کارشناسی ارشد رشته **مهندسی برق-الکترونیک سیستم** دانشکده **مهندسی برق و رباتیک** دانشگاه صنعتی شاهرود نویسنده پایان‌نامه **طراحی و پیاده‌سازی سیستم رفع نویز برای تصاویر دیجیتالی بر مبنای FPGA** تحت راهنمایی **دکتر علی سلیمانی‌ایوری** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های محققان دیگر به مرجع مورد استفاده استناد شده است.
- مطالب مندرج در پایان‌نامه تاکنون توسط خود یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ جا ارائه نشده است .
- کلیه حقوق معنوی این اثر متعلق به دانشگاه صنعتی شاهرود می‌باشد و مقالات مستخرج با نام «دانشگاه صنعتی شاهرود» و یا «Shahrood University of Technology» به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به دست آمدن نتایج اصلی پایان‌نامه تأثیرگذار بوده‌اند در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده ( یا بافت‌های آن‌ها ) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته یا استفاده شده است اصل رازداری، ضوابط و اصول اخلاق انسانی رعایت شده است.

**تاریخ: ۱۳۹۵/۰۶/۱۸**

**مسلم خانه‌بابائی**

## مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده است ) متعلق به دانشگاه صنعتی شاهرود می‌باشد. این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان نامه بدون ذکر مرجع مجاز نمی‌باشد.

## چکیده

یکی از مباحث اصلی در پردازش تصویر، حذف نویز ضربه‌ای فلفل و نمک از تصاویر دیجیتال است. این نوع نویز به‌طور تصادفی نقاطی با رنگ‌های سفید (نمک) و سیاه (فلفل) بر روی تصویر ایجاد می‌کند و باعث از بین بردن اطلاعات و جزئیات در تصاویر می‌شود. هدف این پایان‌نامه طراحی الگوریتمی با استفاده از ترکیب فیلترهای میانه و میانگین برای حذف نویز ضربه‌ای از تصاویر خاکستری و رنگی است که علاوه بر بازدهی بالا در رفع نویز، دارای حجم و محاسبات پیچیده‌ای نیز نباشد. روش پیشنهادی از دو مرحله شناسایی نویز و رفع نویز تشکیل شده است. در ابتدا با در نظر گرفتن مقادیر آستانه پایین و بالا به شناسایی پیکسل نویزی پرداخته می‌شود، در مرحله دوم با استفاده از فیلترهای میانه و میانگین تطبیقی و بر اساس اطلاعات همسایگی مرتبه اول و همسایگی قطری، برای پیکسل‌های نویزی تصمیم‌گیری می‌شود. در ادامه، الگوریتم پیشنهادی با چند کار گذشته در چگالی‌های مختلف نویز ضربه با استفاده از شاخص‌های ارزیابی نسبت بیشینه سیگنال به نویز (PSNR)، خطای میانگین مربعات (MSE)، خطای میانگین مطلق (MAE) و تشابه ساختاری (SSIM) مقایسه می‌گردد. نتایج بصری و عددی نشان از عملکرد کارآمد روش پیشنهادی در حذف نویز ضربه‌ای است. در پایان با توجه به ویژگی‌های پردازش موازی و بلادرنگ تراشه FPGA، الگوریتم ارائه شده با استفاده از زبان توصیف سخت‌افزار VHDL بر روی تراشه Xilinx Spartan-6 سری XC6SLX9 نوشته، سنتز و پیاده‌سازی می‌شود. میزان توان مصرفی کل، بیشینه فرکانس خروجی و متوسط زمان عملیات رفع نویز ضربه در تراشه FPGA برای یک تصویر رنگی با چگالی نویز ۹۰ درصد و اندازه ابعاد ۲۵۶×۲۵۶، به ترتیب مقادیر ۱۴ میلی‌وات، ۴۲/۱۸۶ مگاهرتز و ۱/۲۹۹ ثانیه به دست آمده است که نشان دهنده‌ی عملکرد بهینه و سریع سیستم طراحی شده است.



## واژه‌های کلیدی

نویز ضربه‌ای فلفل و نمک، رفع نویز، فیلتر میانه، فیلتر میانگین، اطلاعات همسایگی، VHDL، FPGA.

**لیست مقالات استخراج شده از این پایان‌نامه**

۱- خانه‌بابائی مسلم و سلیمانی‌ایوری علی، "طراحی و پیاده‌سازی سیستم رفع نویز ضربه برای تصاویر با استفاده از فیلتر میانه-میانگین تطبیقی بر پایه FPGA"، هشتمین کنفرانس بین‌المللی فناوری اطلاعات و دانش، دانشگاه بوعلی سینا همدان، دانشکده مهندسی، شهریور ۱۳۹۵.

# فهرست مطالب

**عنوان** **صفحه**

# فهرست شکل‌ها

**عنوان** **صفحه**

# فهرست جدول‌ها

**عنوان** **صفحه**

# فهرست علائم اختصاری

| Term | Explanation |
|---|---|
| ABEL | Advanced Boolean Expression Language |
| ABMF | Adaptive Based Median Filter |
| ACWMF | Adaptive Center-Weighted Median Filter |
| AMF | Adaptive Median Filter |
| ASIC | Application-Specific Integrated Circuit |
| CE | Chip Enable |
| CLB | Configurable Logic Block |
| CMT | Clock Management Tile |
| CPLD | Complex Programmable Logic Device |
| CWMF | Center-Weighted Median Filter |
| DBA | Decision-Based Algorithm |
| DCM | Digital Clock Manager |
| DRAM | Dinamic Random Access Memory |
| DSP | Digital Signal Processing |
| DVI | Digital Visual Interface |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| EPLD | Erasable Programmable Logic Device |
| EPROM | Erasable Programmable Read-Only Memory |
| FIFO | Frist-in Frist-out |
| FPGA | Field-Programmable Gate Array |
| GAL | Generic Array Logic |
| GP- Processors | General Purpose Processors |
| HDL | Hardware Description Language |
| HDMI | High-Definition Multimedia Interface |
| HIS | Hue, Saturation and Intensity |
| IC | Integrated Circuit |
| IOB | Input-Output Block |
| ISIS | Integrated Silicon Solution Inc |
| ITMMF | Improved Trimmed Mean Median Filter |
| JTAG | Join Test Action Group |
| LUT | Look-Up Table |
| MAE | Mean Absolute Error |
| MDBUTMF | Modified Decision Based Unsymmetric Trimmed Median Filter |

| Term | Explanation |
|---|---|
| MMACS | Millions of Multiply-Accumulate Operations per Second |
| MNLF | Modified Non-Linear Filter |
| MOSFET | Metal–Oxide-Semiconductor Field-Effect Transistor |
| MPGA | Mask-Programmable Gate Array |
| MSE | Mean Squared Error |
| NASM | Noise Adaptive Soft-switching Median |
| NIDF | New Impulse Detection and Filtering |
| NMF | Novel Median Filter |
| OE | Output Enable |
| OTP | One Time Programmable |
| PAL | Programmable Array Logic |
| PLA | Programmable Logic Array |
| PLD | Programmable Logic Device |
| PLL | Phase Locked Loop |
| PROM | Programmable Read-only Memory |
| PSNR | Peak Signal-to-Noise Ratio |
| RAM | Random Access Memory |
| RGB | Red, Green and Blue |
| ROAD | Rank Ordered Absolute Difference |
| ROM | Read only Memory |
| RP | Reprogrammable |
| RTL | Register Transfer Level |
| SLR | Single-Lens Reflex |
| SMF | Standard Median Filter |
| SOC | System On Chip |
| SPLD | Simple Programmable Logic Device |
| SRAM | Static Random Access Memory |
| SSIM | Structural SIMilarity |
| TMDS | Transition-Minimized Differential Signaling |
| UART | Universal Asynchronous Receiver and Transmitter |
| UV | Ultra Violet |
| VGA | Video Graphics Array |
| VHSIC | Very High Speed Integrated Circuits |
| VLSI | Very-Large-Scale Integration |
| WE | Write Enable |
| WMF | Weighted Median Filter |

# فصل اول

## مقدمه و کلیات

## ۱-۱ مقدمه

پردازش سریع اطلاعات از دیرباز مورد نظر بوده و سعی شده است با ارائه سخت‌افزارها، الگوریتم‌های مناسب و روش‌های برنامه‌نویسی خاص تا اندازه‌ای به این مهم نائل آیند. یکی از سخت‌افزارهایی که جدیداً با پیشرفت تکنولوژی استفاده عملی پیدا کرده است، پردازنده‌های FPGA[1] می‌باشد. با توجه به ماهیّت پردازش موازی و خط‌لوله‌ای[2] بسیاری از الگوریتم‌ها در FPGA، این تراشه به عنوان یک بستر مناسب برای پیاده‌سازی شناخته می‌شود. برای پیاده‌سازی الگوریتم‌ها، بایستی آن‌ها را به صورت موازی، پیکربندی و حل کرد. یکی از این مسائل، پردازش بلادرنگ سیگنال‌های تصویر و ویدئو است که شامل پیاده‌سازی الگوریتم‌های پردازش تصویر می‌باشد. در زمینه پردازش تصویر، با افزایش اندازه تصویر و پهنای بیت اطلاعات، نرم‌افزار با کاهش کارایی مواجه می‌شود. الگوریتم‌های پردازش تصویر، عموماً الگوریتم‌های پیچیده و حجیمی هستند و با توجه به اینکه غالباً تمامی داده‌های تصویر در الگوریتم‌ها مورد استفاده قرار می‌گیرند، حجم عملیات زیاد می‌شود. از آنجا که سیستم‌های بهنگام به صورت مستقیم با کاربر در ارتباط هستند، نیاز به افزایش سرعت در پردازش تصویر امری ضروری است. از این‌رو استفاده از تراشه‌های FPGA به عنوان یک شتاب‌دهنده محاسباتی، انتخاب ایده‌آلی برای پیاده‌سازی الگوریتم‌های پردازش تصویر می‌باشند. روش‌های حذف نویز از جمله زمینه‌های مهم در علم پردازش تصویر است و از دیرباز مورد توجه پژوهشگران این علم بوده است که می‌توان به صورت سخت‌افزاری پیاده‌سازی شوند و با سرعت بالایی جهت پاک‌سازی تصاویر تخریب شده، استفاده گردد. به فرآیند تخمین داده بدون نویز از داده نویزی، حذف نویز گفته می‌شود. با توجه به ماهیت مختلف انواع نویزهای موجود در تصاویر، روش‌های حذف این نوع نویزها متفاوت بوده و به ندرت روشی یافت می‌شود که همزمان در حذف دو نوع مختلف نویز نتایج عالی به همراه داشته باشد. از این رو عمدتاً حذف نویزهای مختلف تصاویر مقوله‌های جدا از هم درنظر گرفته می‌شود.

[1] Field-Programmable Gate Array (FPGA)
[2] Pipeline

تصاویر ممکن است در سه مرحله‌ی اخذ، انتقال و نگهداری دچار نویز شوند. در مرحله اخذ، سنسورهای معیوب دوربین یا خود ماهیت صحنه می‌تواند باعث نویز شود. به‌عنوان مثال در تصویربرداری از یک ناو جنگی در دریا، اشعه خورشید به سطح دریا می‌تابد و از آنجا به لنز دوربین منعکس می‌شود، چون شدت نور منعکس شده به دلیل وجود امواج در سطح آب تغییر می‌کند، نویز اخذ تصویر ایجاد می‌شود. در مرحله‌ی انتقال، به دلیل وجود کانال‌های نویزی، تصویر دچار نویز می‌شود. در مرحله‌ی ذخیره‌سازی نیز به دلیل وجود بیت‌های معیوب حافظه یا اغتشاشات خارجی، نویز ایجاد می‌شود. از سایر منشأهای نویز در تصاویر می‌توان به خطاهای زمان‌بندی در تبدیل آنالوگ به دیجیتال و خطای سنکرون در ذخیره دیجیتال نام برد.

در بسیاری از سامانه‌های بینایی ماشین، برای تصمیم‌گیری به استخراج ویژگی از تصویر نیاز است. نویز طبیعتی نوسانی دارد و بنابراین در فرکانس‌هایی که ویژگی‌های تصویر مؤلفه دارند نیز مؤلفه دارد. بنابراین تشخیص و حذف آن از تصویر در کنار حفظ ویژگی‌های تصویر، کار مشکلی است. اگر نویز از تصویر حذف نشود، به عنوان ویژگی به مراحل بالاتر بینایی ماشین منتقل شده و عملکرد سیستم را مختل می‌کند. حتی در بسیاری کاربردها که دقت زیادی مورد نیاز است، وجود یک مقدار جزئی نویز ممکن است زیان‌بار باشد. بنابراین حذف نویز یکی از مهم‌ترین قسمت‌ها در مرحله پیش پردازش تصویر می‌باشد. هدف اصلی کاهش نویز از سیگنال این است که سیگنال بازسازی شده تا حد امکان به سیگنال اصلی نزدیک بوده و در عین حال ویژگی‌های اصلی سیگنال باقی بماند. این اطلاعات اصلی می‌تواند اطلاعات لبه‌ها در تصویر یا اطلاعات فرکانس بالای سیگنال باشد. قابل توجه است که هیچ کدام از روش‌های کاهش نویز، ادعا در حذف کامل نویز ندارند. با این وجود، معمولاً واژه‌ی حذف نویز بجای کاهش نویز در این زمینه استفاده می‌شود.

## ۱-۲ بیان مسئله و چالش‌ها

یکی از نویزهای رایجی که در تصاویر ایجاد می‌شوند، نویزهای ضربه‌ای هستند. نویزهای ضربه‌ای به دو دسته کلی نویز ضربه‌ای فلفل-نمک و نویز ضربه‌ای با مقدار تصادفی تقسیم می‌شوند. در نویزهای فلفل و نمک مقادیر پیکسل‌های نویزی، مقادیر کران شدت روشنایی تصویر می‌باشد. به عنوان مثال برای تصاویر هشت بیتی، مقادیر نویزهای فلفل و نمک به ترتیب صفر یا ۲۵۵ است. در نویزهای ضربه‌ای با مقدار تصادفی همانطور که از اسم آن‌ها بر می‌آید، مقادیر نویز هر عدد دلخواه در بازه صفر تا ۲۵۵ می‌باشد. نویز ضربه اغلب بر اثر خطاهای موجود در مبدل‌های آنالوگ به دیجیتال، انتقال بیت‌ها در تصاویر، شرایط جوی و شدت نور محیط به وجود می‌آید [۶].

حضور نویز از لحاظ ظاهری آزار دهنده است و به علاوه انجام پردازش‌های گوناگون تصویر مانند بخش‌بندی، تشخیص و تفسیر را با مشکل مواجه می‌کند؛ بنابراین افزایش کیفیت تصویر و حذف نویزهای ایجاد شده در تصویر یک مرحله اساسی قبل از هر عملیات پردازشی است. اما نکته مهم در طول روند حذف نویز این است که تصویر اصلی و به خصوص جزئیات آن تا حد امکان آسیبی نبیند و ساختار تصویر اصلی حفظ شود. در این پایان‌نامه علاوه بر معرفی روش پیشنهادی در حذف نویز ضربه‌ای فلفل و نمک، فیلترهای پایه در حذف نویز ضربه‌ای فلفل و نمک نیز مورد بررسی قرار می‌گیرند. فیلترهای میانگین‌گیر و میانه دو نمونه از مهم‌ترین فیلترهای مورد استفاده در حذف نویز ضربه‌ای فلفل و نمک می‌باشند. در هر دو مورد به ازای هر پیکسل، پیکسل‌های همسایه در قالب یک پنجره با اندازه ابعاد ۳×۳ و یا ۵×۵ به طور معمول در نظر گرفته می‌شوند. در فیلتر میانگین‌گیر، میانگین پیکسل‌های درون پنجره جایگزین پیکسل مرکزی می‌شود. مهم‌ترین مشکلات فیلتر میانگین‌گیر، از بین بردن درخشندگی تصویر (تاری تصویر)، هموار کردن لبه‌های تصویر و تغییر شدت روشنایی پیکسل‌های غیر نویزی است. مشکل تغییر شدت روشنایی پیکسل‌های غیر نویزی با استفاده از فیلتر میانگین‌گیر تطبیقی قابل رفع است، اما تاری تصویر و تخریب لبه‌ها همچنان از مهم‌ترین عیب‌های فیلتر میانگین‌گیر می‌باشد. روش دیگر برای حذف نویز ضربه از تصاویر دیجیتال، استفاده از

فیلتر میانه ساده است. در این نوع فیلتر، عنصر میانی پنجره جایگزین عنصر مرکزی پنجره می‌شود. فیلتر میانه دارای دو برتری اساسی نسبت به فیلتر میانگین‌گیر می‌باشد. اول اینکه مقدار میانی بسیار دقیق‌تر از مقدار میانگین است و تغییرات مقادیر همسایه تاثیری در مقدار جایگزین شده نخواهد داشت. دوماً از آنجایی که مقدار میانی در واقع یکی از پیکسل‌های موجود درون پنجره می‌باشد، فیلتر میانه مقدار غیر واقعی تولید نمی‌کند که موجب حفظ درخشندگی تصویر و عملکرد بهتر در پیمایش لبه‌ها می‌شود. فیلتر میانه کاربرد تخصصی در حذف نویز ضربه‌ای دارد. در مقادیر کم نویز، پنجره فیلتر شامل مقادیر کمی نویز می‌باشد و با میانه‌گیری از مقادیر پیکسل‌های تصویر، مقدار نویز حذف می‌شود. در مقادیر نویز بالا، میانه‌گیری از پنجره، منجر به حذف نویز نمی‌شود و اکثر نویزهای تصویر باقی می‌مانند. از معایب این روش نیز هموار شدن لبه‌ها و تغییر پیکسل‌های غیر نویزی است. برای رفع این نقص از فیلتر میانه تطبیقی استفاده می‌شود. فیلتر میانه تطبیقی نیز مشکلاتی دارد، از جمله مهم‌ترین این مشکلات می‌توان عدم کیفیت تصویر بازیابی شده در چگالی‌های بالای نویز اشاره کرد. روش‌های جدید حذف نویز ضربه‌ای، عمدتاً به صورت دو مرحله‌ای هستند. بدین معنی که ابتدا در مرحله‌ی آشکارسازی محل نویزهای ضربه‌ای شناسایی شده و سپس در مرحله‌ی فیلتر، محل نویزها توسط پیکسل‌های سالم اطراف تخمین زده می‌شود. در فیلترهای ذکر شده، علاوه بر اینکه عملیات میانگین‌گیری و مرتب‌سازی، فرآیندی پیچیده و وقت‌گیر است و در چگالی‌های نویز متوسط و بالا نتایج قابل قبولی ندارند، افزایش توان مصرفی در پیاده‌سازی‌های سخت‌افزاری را نیز به دنبال دارد.

### ۱-۳ اهداف و انگیزه

هدف این پایان‌نامه طراحی و پیاده‌سازی سخت‌افزاری سیستم حذف نویز ضربه‌ای فلفل و نمک بر پایه FPGA است. اهداف این پروژه در نگاه کلی به دو گونه بیان می‌شود، افزایش کیفیت تصویر رفع نویز شده و پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی بر روی پردازنده‌ی FPGA. بدین ترتیب در ابتدا، ارائه روش آشکارسازی با دقت و سرعت بالا برای نویز ضربه‌ای فلفل و نمک مدنظر است، به

صورتی که بتواند حداکثر تعداد پیکسل نویزی را بدون خطا آشکارسازی کرده و عملیات رفع نویز را برای آن‌ها به درستی انجام دهد. در مرحله‌ی فیلتر نویز فلفل و نمک نیز روشی که بتواند به صورت مؤثر پیکسل‌های نویزی را تخمین زده و حذف نماید، مطلوب و مورد انتظار است. همچنین بررسی مسأله فیلترهای پایه در رفع نویز ضربه‌ای فلفل و نمک از زوایای مختلف و مقایسه کارایی آن‌ها از لحاظ عملکرد و سرعت نیز یکی از اهداف رساله می‌باشد.

در این پایان‌نامه، طراحی الگوریتمی با استفاده از ترکیب فیلترهای میانه و میانگین و همچنین اطلاعات همسایگی مرتبه اول و همسایگی قطری برای حذف نویز ضربه‌ای از تصاویر خاکستری و رنگی مدنظر است، بطوری که علاوه بر بازدهی بالا در رفع نویز، دارای حجم و محاسبات پیچیده‌ای نیز نباشد. روش پیشنهادی بصورت هوشمندانه عمل می‌کند، به عبارتی دیگر بسته به میزان شدت روشنایی پیکسل و اطلاعات همسایگی آن، در مورد میزان روشنایی نهایی پیکسل مدنظر تصمیم‌گیری خواهد کرد. در مرحله بعد با توجه به ویژگی‌های پردازش موازی و بلادرنگ پردازنده FPGA، الگوریتم ارائه شده با استفاده از زبان توصیف سخت‌افزار VHDL بر روی Xilinx Spartan-6 سری XC6SLX9 نوشته، سنتز و پیاده‌سازی می‌شود. در ادامه الگوریتم‌های فیلتر میانگین‌گیر ساده، فیلتر میانگین‌گیر تطبیقی، فیلتر میانه ساده و فیلتر میانه تطبیقی نیز به صورت سخت‌افزاری پیاده‌سازی می‌شوند و سپس نتایج پیاده‌سازی سخت‌افزاری روش پیشنهادی با سایر الگوریتم‌های پیاده‌سازی شده و شبیه‌سازی شده، مقایسه می‌گردد.

برای مقایسه نتایج رفع نویز از شاخص‌های ارزیابی نسبت بیشینه سیگنال به نویز (PSNR)[1]، میانگین مربعات خطا (MSE)[2]، میانگین مطلق خطا (MAE)[3] و تشابه ساختاری (SSIM)[4] استفاده می‌شود.

[1] Peak Signal-to-Noise Ratio (PSNR)
[2] Mean Squared Error (MSE)
[3] Mean Absolute Error (MAE)
[4] Structural SIMilarity (SSIM)

### ۱-۴ ساختار پایان‌نامه

در فصل اول این پایان‌نامه، مقدمه و کلیات پروژه مورد اشاره قرار گرفت، بطوری که پس از ارائه مقدمه به بیان مسئله و چالش‌های موجود پرداخته شد. در ادامه اهداف و انگیزه این پایان نامه مورد اشاره قرار گرفت.

در فصل دوم به معرفی آرایه دروازه‌های برنامه‌پذیر میدانی (FPGA) پرداخته می‌شود، در این فصل ابتدا انواع مدارات مجتمع برنامه‌پذیر معرفی می‌گردد و سپس به روند تکاملی افزاره‌های برنامه‌پذیر منطقی اشاره می‌شود. در پایان نیز مزایای استفاده از تراشه FPGA و همچنین مقایسه‌ای بین تراشه‌های قابل‌برنامه‌ریزی انجام می‌شود.

در فصل سوم به معرفی زبان توصیف سخت‌افزار VHDL و همچنین آشنایی با ساختار و اصول موجود در این زبان پرداخته می‌شود. در پایان این فصل نیز مراحل طراحی یک برنامه دیجیتال با استفاده از زبان توصیف سخت‌افزار VHDL مورد بحث قرار می‌گیرد.

در فصل چهارم آشنایی پس از معرفی علم پردازش تصویر، با علل ایجاد نویز و انواع نویز در تصاویر بخصوص نویز ضربه‌ای فلفل و نمک مورد اشاره قرار می‌گیرد، در ادامه به مروری از کارهای گذشته در زمینه رفع نویز ضربه‌ای فلفل و نمک و همچنین معرفی شاخص‌های ارزیابی همانندی تصاویر پرداخته می‌شود. در پایان نیز مقایسه‌ای بین بسترهای مختلف برای پیاده‌سازی الگوریتم‌ها انجام می‌شود.

در فصل پنجم روش پیشنهادی برای رفع نویز ضربه از تصاویر دیجیتالی خاکستری و رنگی ارائه می‌شود و سپس ساختار سیستم پیشنهادی جهت عملیات رفع نویز معرفی می‌گردد. در این سیستم جدید و با کاربردهای متنوع، ارتباطی مبتنی بر FPGA-MATLAB طراحی شده است که برای پیاده‌سازی الگوریتم‌های پردازش تصویر مورد استفاده قرار می‌گیرد. از جمله ویژگی‌های این سیستم، سرعت پردازش بالای آن است که در اجرای این الگوریتم‌ها ضروری می‌باشد. در پایان این فصل نیز نتایج پیاده‌سازی سخت‌افزاری روش ارائه شده ذکر خواهد شد و با سایر کارهای گذشته مقایسه

می‌شود. در فصل شش نیز نتیجه‌گیری و پیشنهادات حاصل از این تحقیق بیان می‌شود و با توجه به نتایج به دست آمده از الگوریتم پیشنهادی، نقاط قوت و ضعف این الگوریتم بیان می‌شود. در پایان برای بهبود روش پیشنهادی و همچنین ارتقاء ساختار سیستم معرفی شده، پیشنهاداتی ارائه می‌گردد.

# فصل دوم

## آرایه دروازه‌های برنامه‌پذیر میدانی

## (FPGA)

## ۲-۱ مقدمه

در این فصل افزاره‌های برنامه‌پذیر و روند تکاملی آن‌ها مورد اشاره قرار می‌گیرند. قبل از آن لازم است با تعریف مدارات مجتمع و همچنین انواع آن آشنا شد. مدارات مجتمع[1] به مجموعه‌ای از مدارات الکترونیکی اطلاق می‌گردد که با استفاده از مواد نیمه‌رسانا (عموماً سیلیکون همراه با میزان کنترل شده‌ای ناخالصی) در ابعادی کوچک (معمولاً کمتر از یک سانتی‌متر مربع) ساخته می‌شود. اگر هزاران ترانزیستور در یک ریزتراشه ساخته شود، به آن مدارات مجتمع خیلی فشرده[2](VLSI) می‌گویند. مدارات الکتریکی عموماً شامل المان‌های مداری مانند مقاومت، خازن، سلف و ترانزیستور می‌باشد. در ساخت ICها طراحان سعی می‌کنند تا حد امکان از ترانزیستور استفاده کنند، مثلاً به جای خازن از ترانزیستور در بایاس معکوس استفاده می‌کنند و یا در جایی که مقاومت بزرگی نیاز دارند، باز از ترانزیستور استفاده می‌کنند. چون در حجمی که مقاومت می‌گیرد، می‌توان چند ترانزیستور جای داد. هر تراشه معمولاً حاوی تعداد بسیار زیادی ترانزیستور می‌باشد که با استفاده از فناوری پیچیده‌ای در داخل لایه‌ای از ماده نیمه‌هادی، مانند سیلیکون همگون با پروسه‌های ساخت مدارات مجتمع ساخته می‌شوند. امروزه تراشه‌ها در اکثر دستگاه‌های الکترونیکی و به ویژه رایانه‌ها در ابعادی گسترده بکار می‌روند. وجود تراشه‌ها مرهون کشفیات بشر درباره نیمه رساناها و پیشرفت‌های سریع پیرامون آن‌ها در میانه‌های سده بیستم می‌باشد. مهم‌ترین المان مداری که در تکنولوژی‌های مدار مجتمع ساخته می‌شود، ترانزیستور ماسفت[3] می‌باشد. شرکت Intel یکی از مهم‌ترین و بزرگترین تولید کننده‌های مدارات مجتمع در جهان شناخته شده است. مدارهای مجتمع به دو خانواده بزرگ مدارهای مجتمع با کاربرد خاص[4](ASIC) و مدارهای مجتمع برنامه‌پذیر تقسیم‌بندی می‌شوند. در ادامه به شرح این دو دسته می‌پردازیم.

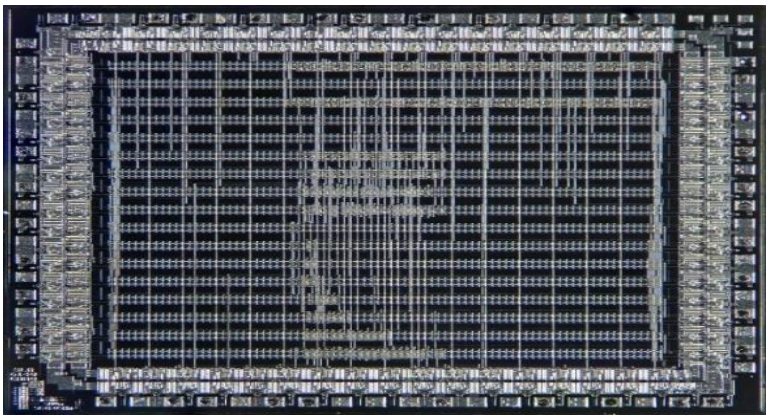
[1] Integrated Circuit (IC)
[2] Very-Large-Scale Integration (VLSI)
[3] Metal–Oxide-Semiconductor Field-Effect Transistor (MOSFET)
[4] Application-Specific Integrated Circuit (ASIC)

## ۲-۲ مدارهای مجتمع با کاربرد خاص (ASIC)

مدارهای مجتمع با کاربرد خاص، مدارهای مجتمعی هستند که به منظور انجام عملیات خاص، طراحی و بهینه‌سازی شده‌اند و اغلب آن‌ها سیستم بر روی چیپ (SOC)[1] می‌نامند. به عنوان مثال یک پردازنده ویژه که در گوشی موبایل مورد استفاده قرار می‌گیرد، نمونه‌ای از این نوع تراشه‌ها می‌باشند. استفاده از تراشه‌های ASIC به افزایش کارایی سیستم منتهی می‌شود، اما طرح ایجاد شده از انعطاف‌پذیری لازم برخوردار نیست. در مقابل، پردازنده‌ها با وجود انعطاف‌پذیری از قابلیت‌های لازم برخوردار نیستند. FPGAها راه حلی برای ایجاد یک سیستم با انعطاف‌پذیری بالا و کارایی مورد نیاز می‌باشند که در ادامه به مورد بحث قرار می‌گیرد. در ASICهای اولیه از فناوری آرایه گیت استفاده می‌شد. اولین برنامه‌ی تجاری موفق در سال‌های ۱۹۸۱ و ۱۹۸۲ میلادی در رایانه‌های هشت بیتی سری ZX اتفاق افتاد. در یک ASIC از ۵۰۰۰ گیت تا ۱۰۰ میلیون گیت بسته به کاربرد می‌تواند وجود داشته باشد. ASICهای مدرن اغلب شامل کل ریزپردازنده، بلوک‌های حافظه از جمله ROM[2]، RAM[3]، EEPROM[4]، حافظه‌های فلش و دیگر بلوک‌ها هستند. تراشه‌های ASIC بهینه‌ترین راه حل موجود می‌باشند، اما استفاده از آن‌ها دارای عیوب عمده‌ای می‌باشد. نخست آن که هزینه اولیه لازم برای تولید این تراشه‌ها بسیار بالاست به طوری که اگر تعداد تراشه‌های مورد نیاز از یک حد خاص کمتر باشد، تولید آن‌ها از دیدگاه اقتصادی قابل توجیه نیست. دوم اینکه زمان اولیه لازم برای طراحی و تولید آن‌ها نسبتاً طولانی است که این موضوع به افزایش زمان ارائه محصول به بازار منجر می‌شود. مشکل دیگر این تراشه‌ها آن است که در صورت نیاز به ایجاد یک تغییر کوچک در طراحی، باید پروسه زمان‌بر و پرهزینه فوق مجدداً طی شود. FPGAها با ارائه یک معماری برنامه‌پذیر، قیمت پایین و زمان راه‌اندازی اندک، مشکلات فوق را مرتفع نمودند. تا آنجا که یکی از کاربردهای تراشه‌های FPGA در انجام مراحل تست و راه‌اندازی ASICها می‌باشد. به این ترتیب که طرح مورد نظر توسط

[1] System On Chip (SOC)
[2] Read only Memory (ROM)
[3] Random Access Memory (RAM)
[4] Electrically Erasable Programmable Read-Only Memory (EEPROM)

تراشه FPGA تست می‌شود و پس از رفع اشکالات و انجام تصحیحات لازم، به روی تراشه ASIC پیاده‌سازی می‌شود. مزیت این روش آن است که از تکرار پروسه زمان‌بر و پرهزینه تولید تراشه‌های ASIC جلوگیری می‌کند. در شکل (۲-۱) یک آرایه ASIC نشان داده شده است [۳۱].



شکل (۲-۱): عکس میکروسکوپ از آرایه گیت‌های ASIC

## ۲-۳ مدارهای برنامه‌پذیر

مدارهای برنامه‌پذیر نقش کلیدی و مهمی در طراحی سخت‌افزارهای دیجیتال ایفا می‌کنند. امروزه مدارهای برنامه‌پذیر متنوعی توسط شرکت‌های مختلف به بازار عرضه می‌شود. هر کدام از این محصولات ویژگی‌های خاص خود را دارند، از جمله این ویژگی‌ها، ظرفیت منطقی، سرعت، ساختار داخلی، کاربرد و قابلیت اطمینان است. مدارهای برنامه‌پذیر از ابتدای به وجود آمدنشان تا به حال مسیر پیشرفت را طی کرده‌اند. به طور کلی مدارهای برنامه‌پذیر را می‌توان به سه دسته کلی زیر تقسیم کرد [۱].

- حافظه‌های فقط خواندنی برنامه‌پذیر[1] (PROM)
- افزاره‌های منطقی برنامه‌پذیر[2] (PLD)
- آرایه دروازه‌های برنامه‌پذیر میدانی[3] (FPGA)

[1] Programmable Read-only Memory (PROM)
[2] Programmable Logic Device (PLD)
[3] Field-Programmable Gate Array (FPGA)

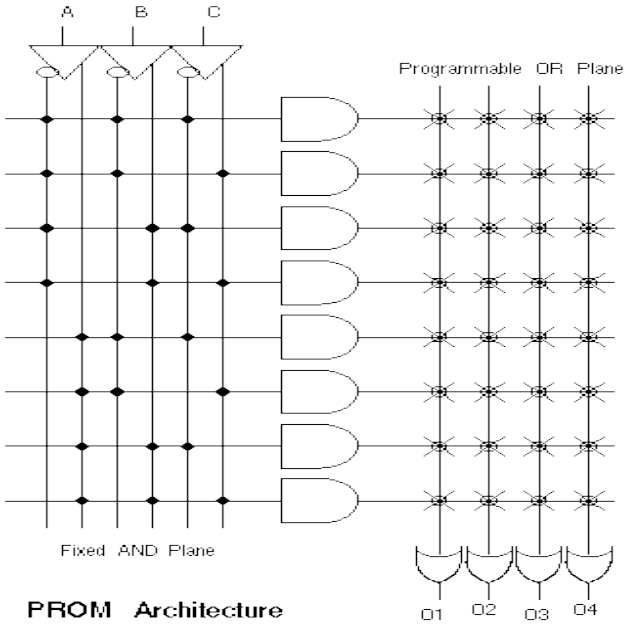
### ۲-۳-۱ حافظه‌های فقط خواندنی برنامه‌پذیر (PROM)

اولین تراشه قابل برنامه‌ریزی توسط کاربر که برای پیاده‌سازی مدارات منطقی نیز مورد استفاده واقع شده است، حافظه فقط خواندنی برنامه‌پذیر است که خطوط آدرس آن بعنوان ورودی و خطوط داده آن به عنوان خروجی مدار منطقی عمل می‌نمایند. در سیستم‌های میکروپروسسوری کوچک معمولاً PROM به عنوان حافظه برنامه استفاده است و کل برنامه‌ی سیستم داخل آن قرار می‌گیرد. در سیستم‌های بزرگتر بخشی از برنامه که وظیفه راه‌اندازی سیستم را دارد، در آن قرار می‌گیرد. حافظه‌های PROM شامل دسته‌ای از گیت‌های AND ثابت شده (غیر قابل برنامه‌ریزی) که به صورت رمز گشا بسته شده‌اند و نیز یک آرایه منطقی OR قابل برنامه‌ریزی است. حافظه‌ی PROM بطور خاص تنها یک بار قابل برنامه‌ریزی است، یا به عبارتی دیگر OTP[1] است. انواع دیگری از آن وجود دارد که قابل برنامه‌ریزی مجدد (RP)[2] است. از جمله‌ی آن‌ها EPROM[3] و EEPROM می‌باشد. EPROM با تابش نور ماورای بنفش (UV)[4] به مدت ۵ تا ۱۵ دقیقه پاک می‌شود، به همین دلیل به آن UVROM نیز گفته می‌شود و جهت برنامه‌ریزی آن از یک دستگاه خاص استفاده می‌شود. خصوصیت EEPROM این است که به وسیله همان دستگاه، برنامه‌ریزی و با سیگنال الکتریکی پاک می‌گردد. لازم به ذکر است شکل (۲-۲) شیوه‌های مختلف اتصال بین خطوط را در آرایه‌های منطقی نشان می‌دهد. شکل (۲-۳) نیز ساختار حافظه‌ی PROM با سه ورودی و چهار خروجی را نشان می‌دهد [۳۱].

(الف) (ب) (پ)

شکل (۲-۲): الف) عدم برقراری اتصال بین خطوط، ب) برقراری اتصال بین خطوط از نوع اتصال قابل برنامه‌ریزی، پ) برقراری اتصال بین خطوط از نوع اتصال ثابت

[1] One Time Programmable (OTP)
[2] Reprogrammable (RP)
[3] Erasable Programmable Read-Only Memory (EPROM)
[4] Ultra Violet (UV)




شکل (۲-۳): ساختار حافظه‌ی PROM با سه ورودی و چهار خروجی

### ۲-۳-۲ افزاره‌های منطقی برنامه‌پذیر (PLD)

توسعه مدارت مجتمع قابل برنامه‌ریزی در حوزه کار به میزان قابل توجهی طراحی سخت‌افزارهای دیجیتال را دگرگون نموده است. بر خلاف نسل‌های گذشته تکنولوژی سخت‌افزار، که در آن طراحی‌های در سطح برد شامل تعداد زیادی از تراشه‌های حاوی گیت‌های پایه بود، در حال حاضر هر طرح دیجیتال تولید شده شامل قطعات با ظرفیت بالاست. این موضوع نه تنها برای قطعات الکترونیکی نظیر پردازنده‌ها و حافظه‌ها صادق است، بلکه در مورد مدارهای منطقی نظیر شمارنده‌ها، ثبات‌ها و رمزگشاها نیز اعتبار دارد. هنگامی که طراحی سیستم‌های حجیم مد نظر باشد، طراحان این سیستم‌ها را به صورت مجتمع درون آرایه‌های گیتی با گنجایش بالا پیاده می‌نمایند، اما هزینه مهندسی غیر قابل برگشت بالا و زمان طولانی، سبب شده تا آرایه‌های متشکل از گیت برای اهداف تهیه نسخه اولیه و سایرکاربردهای با تولید محدود مناسب نباشد. به همین دلیل بسیاری از نسخه‌های اولیه مدار و طرح‌های تولیدی از PLDها استفاده می‌نمایند. از جمله فواید چشمگیر PLDها می‌توان به هزینه پایین طراحی، ریسک اقتصادی کم و بدلیل برنامه‌ریزی شدن نهایی قطعه توسط کاربر، مدت

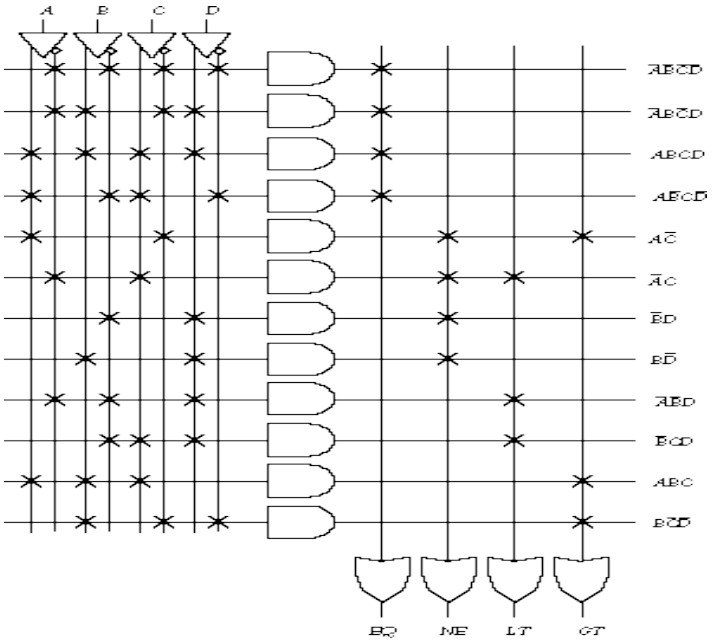
زمان کم تولید محصول و سادگی اعمال تغییرات بعدی روی طرح اشاره نمود. PLDها از نظر ظرفیت منطقی و سطح پیچیدگی مدار داخلی به دو دسته زیر تقسیم می‌شوند:

- افزاره‌های منطقی برنامه‌پذیر ساده (SPLD)[1]
- افزاره‌های منطقی برنامه‌پذیر پیچیده (CPLD)[2]

افزاره‌های SPLD ارزان‌ترین و کوچکترین قطعه برنامه‌پذیر است که می‌تواند جایگزین حداکثر ۲۰۰ گیت NAND از سری ۷۴۰۰ گردد. در ادامه به روند تکاملی PLDهای ساده پرداخته می‌شود [۳].

### ۲-۳-۲-۱ افزاره‌های منطقی برنامه‌پذیر ساده PLA

یک PLA دارای دو صفحه منطقی قابل برنامه‌ریزی است. یک صفحه قابل برنامه‌ریزی با استفاده از تکنیک AND سیم‌بندی شده قرار گرفته است. ساختار PLA این امکان را فراهم می‌سازد که هر کدام از ورودی‌های مربوط به صفحه AND یا مکمل آن‌ها با یکدیگر OR شده و به این ترتیب هر خروجی این صفحه با یک ترم از ورودی‌ها متناظر خواهد بود. در شکل (۲-۴) یک نمونه PLA نشان داده شده است.




شکل (۲-۴): ساختار تراشه قابل برنامه‌ریزی PLAبا چهار ورودی و چهار خروجی

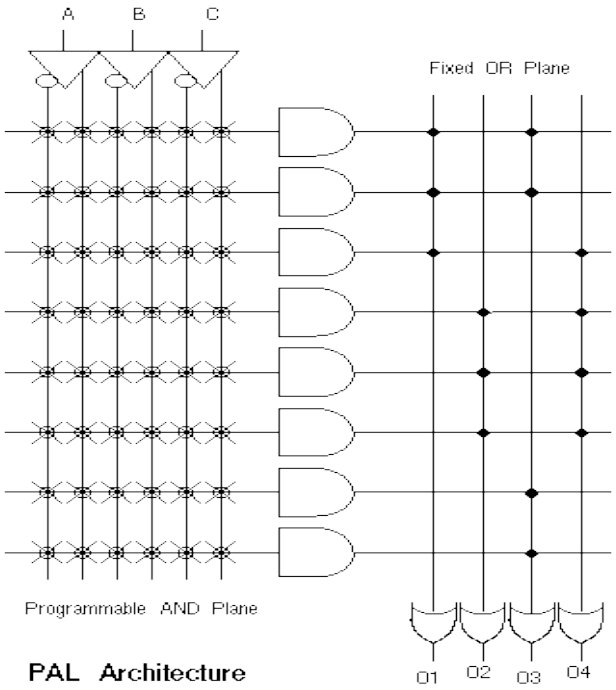
[1] Simple Programmable Logic Device (SPLD)
[2] Complex Programmable Logic Device (CPLD)

به همین صورت کاربر قادر است هر خروجی صفحه OR را بنحوی تنظیم نماید که مجموع تعدادی از خروجی‌های صفحه AND را شامل باشد. به این ترتیب PLAها برای پیاده‌سازی مدارها به شکل مجموع حاصلضرب‌ها مناسب هستند. صفحات AND و OR می‌توانند تعداد زیادی ورودی داشته باشند. این ساختار دارای معایب هزینه بالای ساخت و تا حدی بازدهی و سرعتی کم است. این عیب‌ها ناشی از برنامه‌پذیر بودن دو سطح منطقی موجود در PLA است، چرا که هم ساخت دو صفحه قابل برنامه‌ریزی مشکل است و هم وجود کلیدهای قابل برنامه‌ریزی در هر دو صفحه، تأخیر انتشار قابل توجهی را تحمیل می‌کند. شرکت Memories Monolithic برای پوشش دادن اشکالات PLA، ساختار آرایه قابل برنامه‌ریزی منطقی PAL[1] را به بازار عرضه کرد.

### ۲-۳-۲-۲ افزاره‌های منطقی برنامه‌پذیر ساده PAL

PAL شامل یک آرایه AND قابل برنامه‌ریزی و یک OR تثبیت شده است. در واقع PAL نوع خاصی از PLA است. اولین PAL در اواخر سال ۱۹۷۰میلادی معرفی شد و خصوصیت جدید آن داشتن پایه‌های دوطرفی ورودی-خروجی، هزینه ساخت پایین‌تر و سرعت بالاتر بود. PALهای استاندارد، آرایش‌های متفاوتی دارند که هر یک از آن‌ها توسط عددی یکتا مشخص می‌شوند. این عدد همیشه با پیشوند PAL شروع می‌شود. دو رقم بعد از PAL تعداد ورودی‌ها را نشان می‌دهد که شامل خروجی‌هایی است که به صورت ورودی به کار روند. حرف بعد از تعداد ورودی‌ها، نوع خروجی را نشان می‌دهد، به طور مثال L یعنی فعال پایین، H یعنی فعال بالا و P یعنی قابل برنامه‌ریزی. یک یا دو عدد بعدی که بعد از نوع خروجی قرار می‌گیرد، تعداد خروجی‌هاست. به عنوان مثال PAL10L8 دارای ۱۰ورودی و ۸ خروجی فعال پایین است. علاوه بر این شماره PAL می‌تواند پسوندهایی برای تعیین سرعت، نوع بسته‌بندی و حوزه حرارتی داشته باشد. در شکل (۲-۵) یک نمونه PAL با سه ورودی و چهار خروجی نشان داده شده است.

[1] Programmable Array Logic (PAL)




شکل (۲-۵): ساختار تراشه قابل برنامه‌ریزی PALبا سه ورودی و چهار خروجی

### ۲-۳-۲-۳ افزاره‌های منطقی برنامه‌پذیر ساده GAL

علاوه بر تراشه‌های PAL، تراشه دیگری به نام GAL[1] نیز موجود است. این تراشه بهبود یافته PAL می‌باشد که در سال ۱۹۸۵ میلادی توسط شرکت Lattice Semiconductor اختراع شد. آرایه GAL خصوصیات مشترکی با تراشه PAL دارد، با این تفاوت که قابل پاک شدن و برنامه‌ریزی مجدد است. این تراشه در نمونه‌سازی مراحل یک طراحی بسیار مفید است. GALها را با استفاده از PALها و یا تکنولوژی برنامه‌ریزی داخل مدار می‌توان مجدداً برنامه‌ریزی کرد. معمولاً تمام PLDهای ساده را که شامل PAL، PLA، GAL و قطعات مشابه آن می‌باشند، در مقوله‌ای به نام SPLD قرار می‌دهند که مشخصه اصلی آن‌ها، قیمت کم و بازدهی سرعتی بسیار بالاست.

### ۲-۳-۲-۴ افزاره‌های منطقی برنامه‌پذیر پیچیده (CPLD)

پیشرفت تکنولوژی، عرضه قطعاتی را که گنجایش بالاتری از SPLD دارند ممکن ساخته است. مشکل افزایش گنجایش افزاره‌های SPLD آن است که با افزایش تعداد ورودی‌های صفحه منطقی قابل

[1] Generic Array Logic (GAL)

برنامه‌ریزی، وسعت ساختمان آن به سرعت رو به رشد می‌گذارد. تنها راه حل تهیه نمودن قطعات پر گنجایش مبتنی برSPLD، اتصال قابل برنامه‌ریزی چند SPLD روی یک تراشه منفرد به یکدیگر است. چنین ساختاری CPLD نامیده می‌شود. این ایده اولین بار توسط شرکت Altera در اولین تراشه‌های Classic EPLD[1] استفاده شد. بخاطر بازار رو به رشد PLDهای بزرگ، شرکت‌های دیگری نیز افزاره‌های CPLD را معرفی نمودند. CPLDها گنجایش منطقی در حدود ۵۰ برابر یک SPLD نوعی را ارائه می‌نمایند، لیکن افزایش گنجایش این معماری‌ها به دلیل پایه قرار دادن تحقق توابع منطقی دو سطحی مشکل است.

پرظرفیت‌ترین تراشه‌های منطقی همه منظوره که اکنون در دسترس هستند، آرایه‌های متشکل از گیت‌ها هستند که عموماً با نام آرایه‌ی دروازه‌های برنامه‌پذیر نقابی یا MPGA[2] نامیده می‌شوند. این آرایه‌ها جهت پیاده‌سازی مدارهای منطقی بزرگ به وجود آمده‌اند. از این نوع تراشه‌ها در روش نیمه سفارشی استفاده می‌شود، در این روش عملیات ساخت تراشه تا آنجایی که مستقل از طرح نهایی است، انجام شده و تراشه آماده است که در صورت سفارش یک طرح، مراحل نهایی پیاده‌سازی انجام شود. یک MPGA نوعی، دارای آرایه‌ای از ترانزیستورهاست که می‌تواند جهت پیاده‌سازی هر مدار مطلوب به هم متصل شوند. آرایش ترانزیستورها را می‌توان به صورت سطری کنار هم در نظر گرفت. این امر باعث می‌شود که بتوان دروازه‌های منطقی را به وجود آورد و این دروازه‌ها را به هم متصل نمود. از آنجا که این اتصال‌ها بایستی به وسیله لایه فلزی در کارخانه ایجاد شود، عملیات نیازمند زمان و هزینه است. پیشرفت در ابزارهای طراحی و تراشه‌های قابل برنامه‌ریزی منجر به عرضه FPGA شد. امروزه FPGAها از نظر تکنولوژی در زمره بزرگ‌ترین مدارهای مجتمع موجود در بازار هستند.

[1] Erasable Programmable Logic Device (EPLD)
[2] Mask-Programmable Gate Array (MPGA)

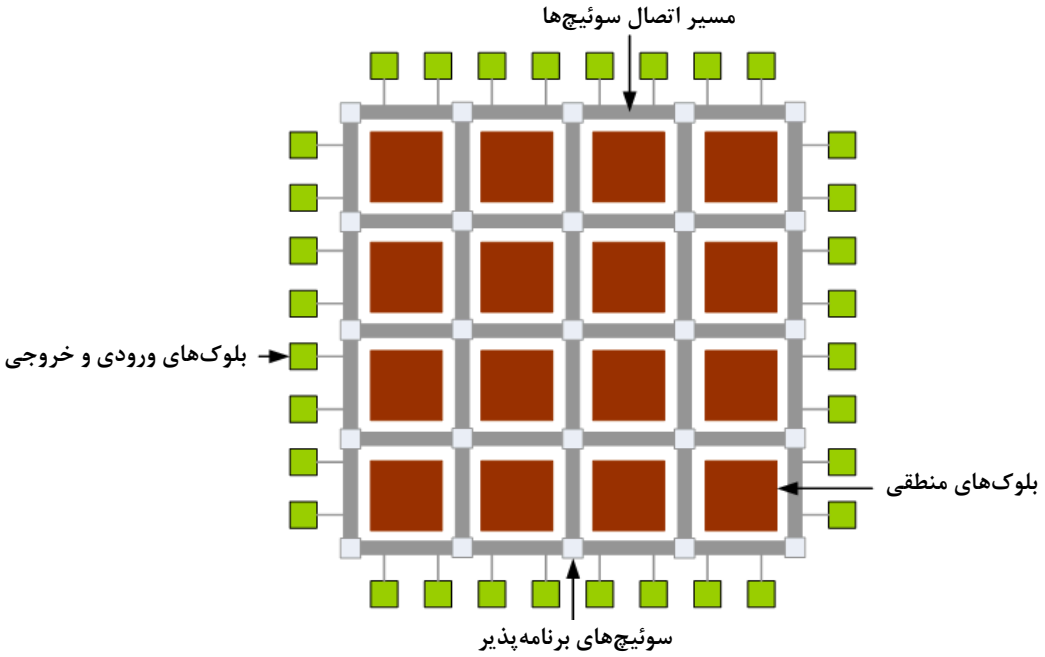
## ۲-۳-۳ آرایه دروازه‌های برنامه‌پذیر میدانی (FPGA)

FPGAها تراشه‌های برنامه‌پذیری هستند که استفاده کننده نهایی می‌تواند آن را بدون داشتن تسهیلات ساخت، پیکربندی نماید. اولین FPGA در سال ۱۹۸۵ میلادی به وسیله شرکت Xilinx ارائه شد. از آن زمان به بعد این تراشه توسط شرکت‌های مختلفی از جمله ALTERA، ACTEL، ATMEL و AT&T تولید می‌شود. بیشتر FPGAهای مورد استفاده ظرفیتی بالغ بر ۸۰۰۰ گیت دارند. این امکان نتیجه پیشرفت در نرم‌افزارهای طراحی است که می‌توانند مستقل از تراشه نهایی، طراحی را انجام دهند و در نهایت طراح می‌تواند تصمیم بگیرد که طرح با FPGA یا MPGA پیاده‌سازی گردد. FPGAها مانند MPGAها تشکیل شده‌اند از یک سری عناصر منطقی که برای کار خاصی محدود شده‌اند و نیز مانند PAL دارای اتصالات قابل برنامه‌ریزی است. بنابراین هر دو جزء اصلی تشکیل دهنده یک مدار یعنی بلوک‌های منطقی و همچنین اتصالات بین آن‌ها، قابل برنامه‌ریزی است. شکل (۲-۶) ساختار اصلی یک FPGA را نشان می‌دهد. همانطور که در شکل (۲-۶) مشخص است، سه جزء اصلی یک FPGA عبارتند از بلوک‌های منطقی (CLB)[1]، عناصری که برای اتصالات بین بلوک‌های منطقی و بلوک‌های ورودی-خروجی به کار می‌روند (Switch Matrix) و بلوک‌های ورودی-خروجی (IOB)[2]. ساختار و محتویات بلوک‌های منطقی می‌تواند خیلی ساده (در حد گیت NAND و یا خیلی پیچیده نظیر چند MUX یا LUT[3] به همراه یک فلیپ فلاپ) باشد. منظور از LUT، هر یک از حافظه‌های مورد استفاده است [۳۱].

بلوک‌های منطقی در حقیقت جایی هستند که قسمت‌های اصلی مدار قرار می‌گیرند. البته ابتدا باید مداری که قرار است روی FPGA قرار بگیرد به اجزای یکسانی که همان محتویات بلوک‌های منطقی هستند تقسیم شود و بعد از این عمل است که می‌توان با اتصال بلوک‌های پایه به هم، مدار واقعی را به دست آورد. عناصری که برای اتصالات به کار می‌روند، معمولاً بین بلوک‌های منطقی قرار

[1] Configurable Logic Block (CLB)
[2] Input-Output Block (IOB)
[3] Look-Up Table (LUT)

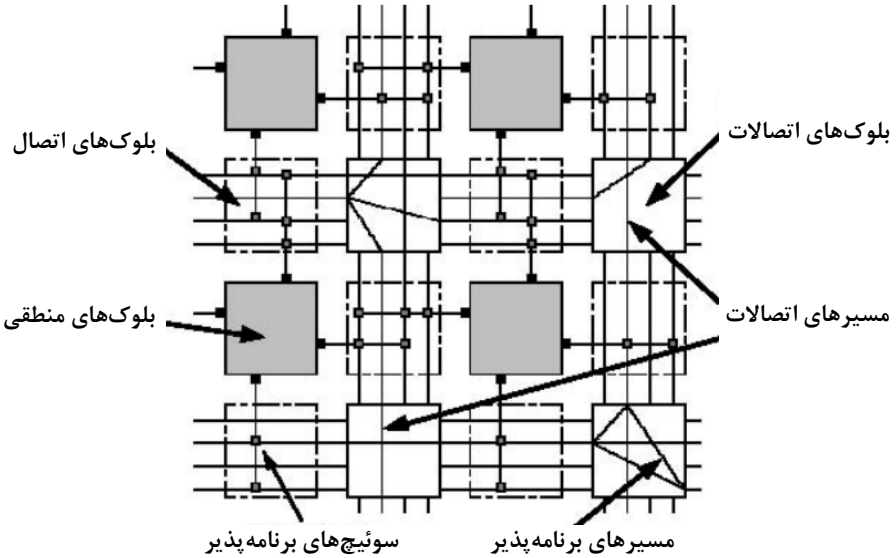
می‌گیرند و از قطعات فلزی که می‌توانند به هم یا به بلوک‌های منطقی متصل شوند، تشکیل شده‌اند. برای متصل کردن این قطعات از سوئیچ‌های قابل برنامه‌ریزی استفاده می‌شود. این قطعات می‌توانند طول‌های متفاوتی داشته باشند .




شکل (۲-۶): ساختار داخلی تراشه FPGA

بلوک‌های ورودی- خروجی برای اینکه پین‌های FPGA را بتوان در مدهای مختلف ۳/۳ یا ۵ ولت برنامه‌ریزی کرد، به کار می‌روند. طراحی بلوک‌های منطقی و عناصر اتصالی مهم‌ترین قسمت طراحی یک FPGA می‌باشد، زیرا طراحی این دو با هم باید به گونه‌ای باشد که پیاده‌سازی مدارات منطقی مختلف را روی FPGA ممکن سازد. معمولاً بین پیچیدگی و انعطاف‌پذیری هر دو بلوک‌های منطقی و منابع اتصالی یک نسبت معکوس وجود دارد. یعنی با زیاد شدن یکی دیگری کم می‌شود و بر عکس. در ضمن، معماری یک بلوک منطقی و همچنین منابع اتصالی بر کل مساحت تراشه و سرعت تراشه اثر دارد. در شکل (۲-۷) نمای داخلی FPGA با جزئیات بیشتری نمایش داده شده است [۲]. ساختمان و محتوای تجهیزات در یک FPGA به معماری مسیرنمایی موسوم است. معماری مسیر نمایی حاوی قطعات سیم و کلیدهای قابل برنامه‌ریزی است. کلیدهای قابل برنامه‌ریزی به طریق

مختلفی ساخته می‌شوند. از جمله توسط ترانزیستورگذر[1] کنترل شونده با SRAM، آنتی‌فیوز و ترانزیستورهای EEPROM. امروزه تحقیقات در زمینه بلوک منطقی FPGA، بیشتر بر روی ارائه معماری‌های غیرهمزمان و FPGAهای ممیز شناور متمرکز شده است. همان طور که قبلاً ذکر شد، سه جزء اصلی یک FPGA عبارت‌اند از بلوک‌های منطقی (CLB)، عناصری که برای اتصال‌های بین بلوک‌های منطقی و بلوک‌های ورودی-خروجی به کار می‌روند و بلوک‌های ورودی-خروجی(IOB). در ادامه به شرح این اجزا پرداخته می‌شود.



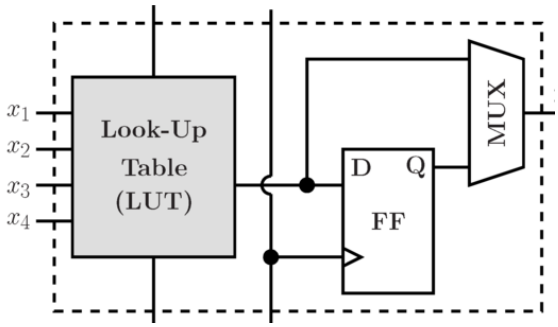

شکل (۲-۷): نمای داخلی یک FPGA معمول

### ۲-۳-۳-۱ بلوک‌های منطقی برنامه‌پذیر (CLB)

واحد اصلی تشکیل دهنده بلوک منطقی در FPGAهای مبتنی بر SRAM، معمولاً LUT است. منظور از بکار بردن واژه واحد اصلی آن است که واحدهای فرعی دیگری نیز داخل CLB وجود دارند، از آن جمله می‌توان به فلیپ فلاپ‌ها، مالتی پلکسرها و گیت‌های ساده اشاره کرد. هدف از بکار بردن فلیپ فلاپ‌ها در داخل بلوک منطقی آن است که امکان تحقق مدارات ترتیبی بدون آنکه از خود CLBها جهت تحقق عناصر فیدبک‌دار استفاده شود، وجود داشته باشد [۳۱].

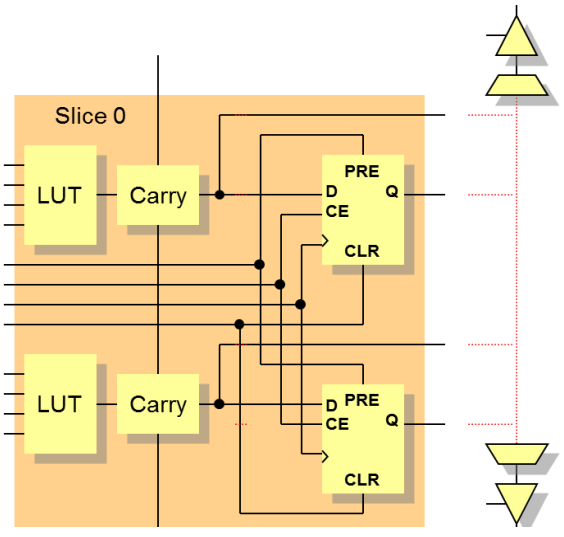
[1] Pass Transistor

از مالتی پلکسرها نیز برای تولید خروجی‌های متنوع استفاده شده است. به عنوان مثال چنانچه لازم باشد خروجی LUT بدون آنکه از میان فلیپ فلاپ‌ها بگذرد، مستقیماً به خروجی CLB متصل می‌شود و یا حتی به یکی از ورودی‌های خود LUT برگردانده می‌شود. استفاده از گیت‌های ساده نیز عمدتاً بخاطر سهولت بخشیدن به تهیه مکمل سیگنال‌های میانی در داخل بلوک منطقی می‌باشد. در شکل (۲-۸) ساختار داخلی یک سلول منطقی نشان داده شده است، هر سلول منطقی از یک LUT، فلیپ فلاپ و مالتی پلکسر تشکیل شده است.



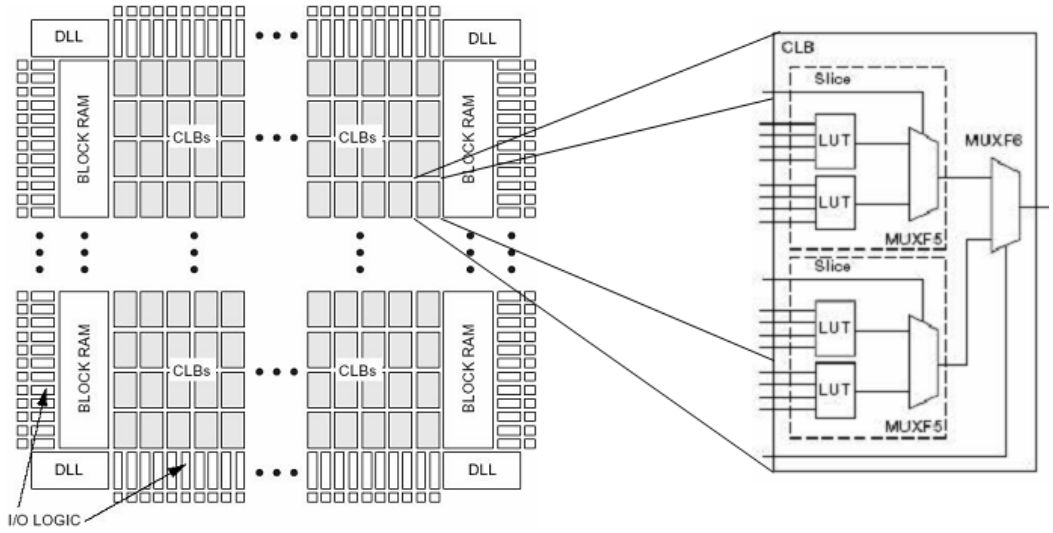

شکل (۲-۸): ساختار داخلی سلول منطقی

در سلول منطقی، خروجی LUT به صورت مستقیم و همچنین از طریق یک فلیپ فلاپ نوع D به یک مالتی‌پلکسر وارد می‌شود. بسته به ورودی خط کنترل MUX، مدار ترکیبی یا ترتیبی خواهد بود. همانند شکل (۲-۹) از ترکیب دو سلول منطقی، یک Slice به وجود می‌آید.




شکل (۲-۹): ساختار داخلی یک نمونه Slice

از ترکیب دو Slice یک بلوک منطقی (CLB) ساخته می‌شود. در واقع CLB اصلی‌ترین بخش از هر FPGA است که توابع منطقی را پیاده‌سازی می‌کند. شکل (۲-۱۰) یک CLB را نشان می‌دهد.

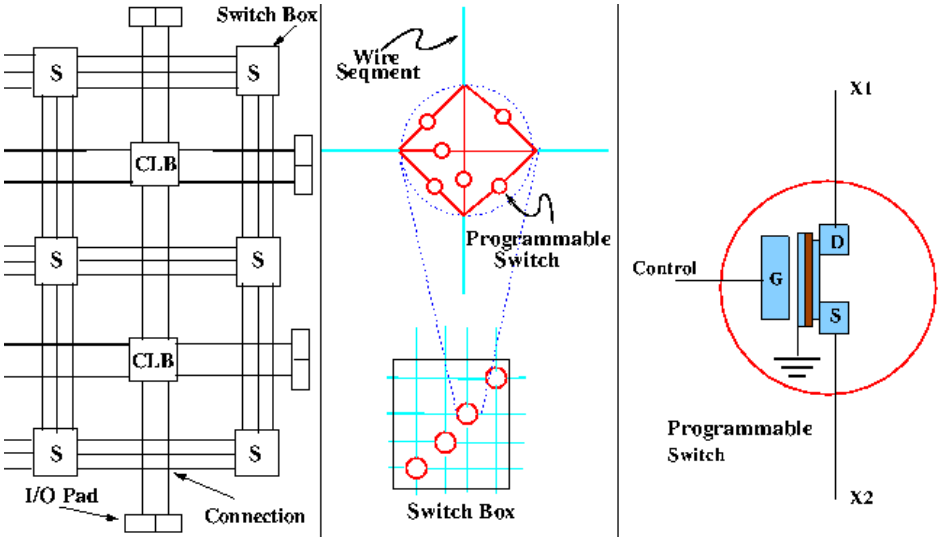



شکل (۲-۱۰): ساختار داخلی بلوک‌های قابل برنامه‌ریزی (CLB)

### ۲-۳-۳-۲ اتصال‌های قابل برنامه‌ریزی

در FPGAهای مبتنی بر SRAM در محل تقاطع کانال‌های افقی و عمودی، یک جعبه حاوی ماتریسی از سوئیچ‌ها قرار دارد که امکان برقراری اتصال بین شیارهای افقی و عمودی را فراهم می‌سازد. در اطراف بلوک‌های منطقی نیز جعبه دیگری با عنوان جعبه اتصال تعبیه شده که توسط سوئیچ‌های داخل آن، شیارهای انتقال سیگنال به ورودی‌ها و خروجی‌های بلوک منطقی متصل می‌گردند. از آن جهت که وجود سوئیچ‌های قابل برنامه‌ریزی به تأخیر انتشار سیگنال اضافه می‌کند و گذشته از آن نیازی وجود ندارد که تمام خطوط انتقال سیگنال به طور کامل به یکدیگر متصل گردند، تعداد این سوئیچ‌ها در هر دو نوع از جعبه‌ها محدود است. شکل (۲-۱۱) خواننده را جهت درک بهتر وضعیت و ساختار معماری مسیر نمایی یاری خواهد کرد. در مورد FPGAهایی که در آن‌ها از آنتی‌فیوز استفاده شده، جعبه‌های مذکور بدلیل ساختار خاص این گونه از تراشه‌ها وجود ندارند. شیارهای عمودی و افقی سیگنال در سراسر تراشه گسترده شده‌اند. شیارهای افقی انتقال سیگنال بین طبقات

مختلف بلوک‌های منطقی قرار گرفته‌اند. شیارهای عمودی با شیارهای افقی مذکور تلاقی داشته و روی ردیف‌های بلوک‌های منطقی با کانال‌های افقی انتقال سیگنال را فراهم می‌سازد [۳۱].



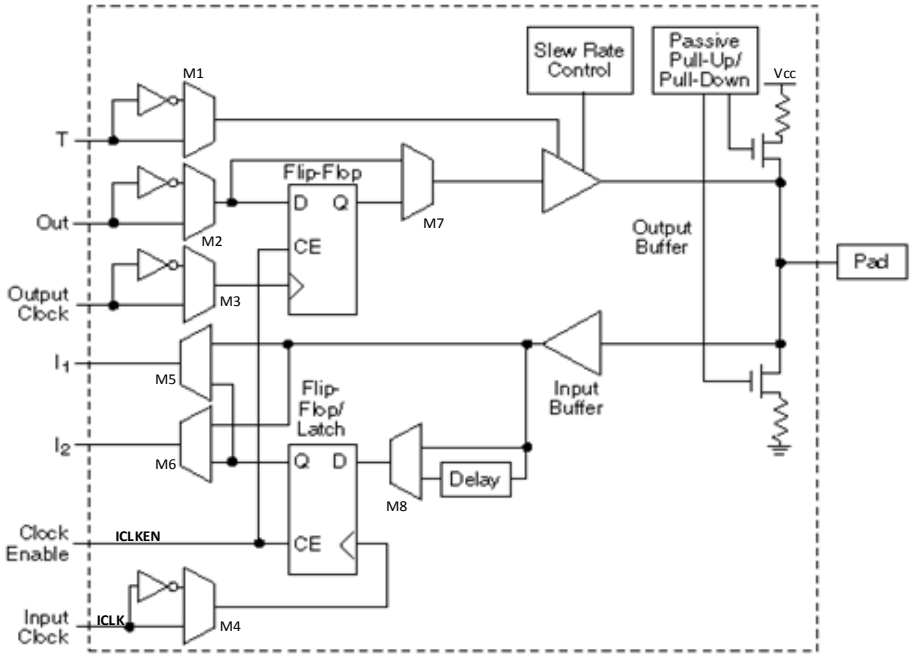

شکل (۲-۱۱): معماری کلی سوئیچ‌های قابل برنامه‌ریزی در FPGA

### ۲-۳-۳-۳ بلوک‌های ورودی-خروجی (IOB)

به منظور آنکه امکان برقراری ارتباط دو طرفه با جهان خارج از FPGA و داخل آن وجود داشته باشد، بلوک‌های I/O با استانداردهای مختلفی در تمام FPGAها وجود دارند. شکل (۲-۱۲) یک ساختار ورودی-خروجی نوعی را نشان می‌دهد. این ساختار مربوط به پردازنده سری XC4000 است. در سر راه ورودی و خروجی این I/O فلیپ فلاپی از نوع D قرار داده شده است که با مالتی پلکسرهای $M_5$ تا $M_7$ قابل انتخاب است. وجود این فلیپ‌فلاپ‌ها مفید است، مثلاً در مواردی که تاخیر نسبتاً زیاد از خروجی فلیپ‌فلاپ داخلی CLB تا IOB مشکل‌ساز شود، این فلیپ فلاپ‌ها باعث همزمانی سیگنال‌های خروجی با کلاک سیستم می‌شود. وجود المان تاخیر در ورودی فلیپ فلاپ شماره دو که با استفاده از $M_8$ قابل انتخاب است، زمان نگه داشتن[1] لازم برای کلاک داخلی FPGA که کپی کلاک سراسری سیستم می‌باشد را ضمانت می‌کند. سایر ورودی‌های IOB که از لحاظ پلاریته به وسیله $M_1$ تا $M_4$ قابل انتخاب هستند، از آرایه CLB و از میان اتصالات قابل برنامه‌ریزی می‌آیند. این ورودی‌ها به

[1] Hold Time

نام‌های OUT، T، OCLK و ICLK به ترتیب بیت خروجی، کنترل فعال‌ساز سه حالته‌ی آن، کلاک برای فلیپ فلاپ خروجی و کلاک برای فلیپ فلاپ ورودی می‌باشد. ورودی ICLKEN فعال‌ساز کلاک‌های ورودی مربوط به هر دو فلیپ فلاپ است. این ساختار IOB کنترل آنالوگ نیز دارند و سرعت راه‌اندازهای خروجی قابل برنامه‌ریزی می‌باشد و همچنین مقاومت Pull-up می‌تواند بین پایه‌های I/O متصل گردد [۳۱].




شکل (۲-۱۲): ساختار ورودی-خروجی نوعی FPGA سری XC4000

## ۲-۴ انواع FPGA بر اساس ساختار منابع اتصالی

ساختار منابع اتصالی در FPGAها متفاوت است. منابع اتصالی از یک سری قطعه سیم با طول متفاوت و یک سری سوئیچ برنامه‌پذیر تشکیل شده است. سوئیچ‌های برنامه‌پذیر بطور کلی به روش‌های مختلف ساخته می‌شود که از آن جمله می‌توان به روش‌های زیر اشاره کرد.

- ❖ با استفاده از فیوز
- ❖ با استفاده از آنتی‌فیوز
- ❖ یک سری ترانزیستور که به وسیله EPROM کنترل می‌شود.
- ❖ یک سری ترانزیستور که به وسیله EEPROM کنترل می‌شود.

❖ یک سری ترانزیستور که به وسیله SRAM کنترل می‌شود.

معماری مسیردهی بین بلوک‌های منطقی نیز به روش‌های گوناگونی انجام می‌شود. برخی سازنده‌ها از اتصالات ساده بین بلوکی استفاده می‌کنند و برخی دیگر بیشتر از اتصالات عمومی و پیچیده‌تر استفاده می‌کنند. در ادامه به شرح انواع FPGA بر اساس تکنولوژی ساخت سوئیچ می‌پردازیم.

### ۲-۴-۱ ساختار مبتنی بر فیوز

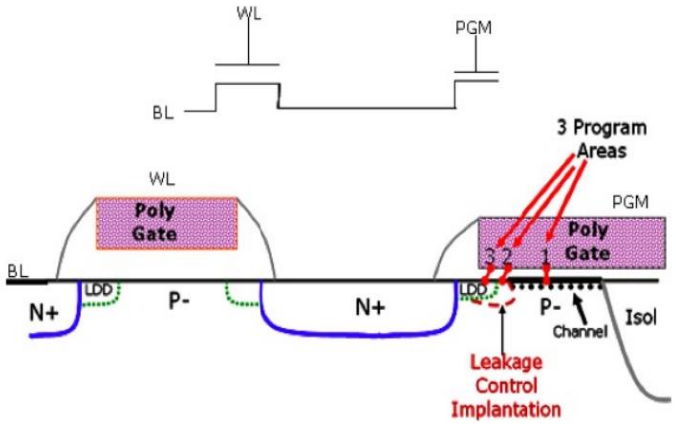
انواع اولیه سوئیچ‌های قابل برنامه‌ریزی نوعی سوئیچ بودند که در PLA و PROM از آن‌ها استفاده می‌شد. فیوز یک اتصال فلزی نازک است که در اثر عبور جریان ذوب شده و از بین می‌رود. فیوز در حالت عادی اتصال کوتاه بوده و پس از برنامه‌ریزی اتصال باز می‌شود. اگر چه هنوز هم فیوز در برخی افزاره‌های برنامه‌پذیر کوچک استفاده می‌شود، ولی به سرعت در حال جایگزینی با تکنولوژی جدیدتر است. در افزاره‌های با ظرفیت بالاتر و در جاهایی که از تکنولوژی CMOS برای ساخت IC استفاده می‌شود، روش‌های دیگری به کار می‌رود، مثلاً تکنولوژی اصلی مورد استفاده در CPLDها، ترانزیستور گیت شناور بوده و در FPGA از تکنولوژی SRAM و Anti-Fuse استفاده می‌شود.

### ۲-۴-۲ ساختار مبتنی آنتی‌فیوز

آنتی‌فیوز نوعی فیوز است که بر عکس فیوز معمولی در حالت عادی اتصال باز است، لذا در حالت عادی قطع است و هنگامی وصل می‌باشد که یک جریان نسبتاً زیادی از دو سر ترمینال آن عبور داده شود. دو نوع آنتی‌فیوز در FPGA مورد استفاده است. یک نوع آن‌ها که تحت عنوان PLICE شناخته شده توسط شرکت ACTEL استفاده می‌شود و نوع دیگر که تحت عنوان VIA LINK شناخته می‌شود، توسط شرکت Quick logic استفاده می‌شود.

FPGAهایی که از قابلیت آنتی‌فیوز استفاده می‌کنند، چگالی گیتی بیشتری را نسبت به انواع دیگر که از المان‌های حافظه استفاده می‌کنند، در اختیار طراح قرار می‌دهند. مزیت عمده استفاده از آنتی‌فیوزها، بالا بودن فرکانس کاری آن‌ها می‌باشد، چرا که ظرفیت خازنی و اثر مقاومتی کمی دارند.

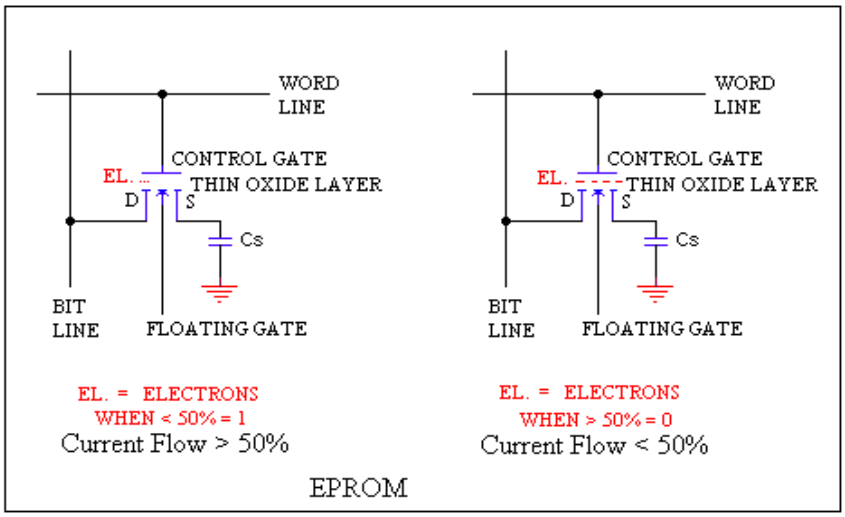
عیب اصلی این روش این است که قابلیت برنامه‌ریزی مجدد ندارند و وقتی که یک آنتی‌فیوز برنامه‌ریزی شد، دیگر به حالت قطع بر نمی‌گردد. شکل (۲-۱۳) نمای داخلی یک آنتی‌فیوز را نشان می‌دهد.




شکل (۲-۱۳): نمای داخلی یک آنتی‌فیوز

### ۲-۴-۳ ساختار مبتنی بر گیت شناور (EPROM و EEPROM)

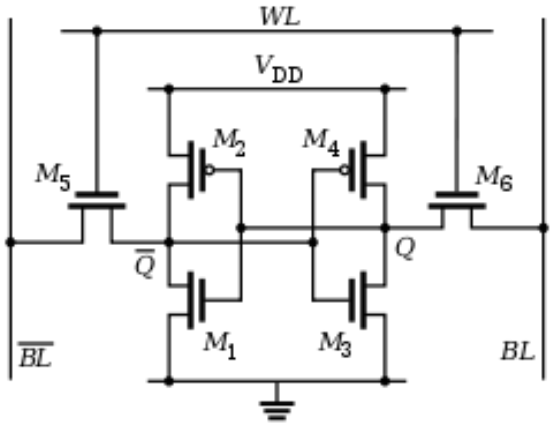
در تکنولوژی MOSFET ترانزیستورها خود به عنوان سوئیچی عمل می‌کنند که با اعمال سطح High یا Low به گیتشان می‌توان باعث قطع و وصل شدن این سوئیچ‌ها شد. در این روش از ترانزیستوری استفاده می‌شود که دارای گیت شناور است. با استفاده از این گیت شناور می‌توان سطح آستانه ترانزیستور را با شارژ و دشارژ کردن گیت دوم تغییر داد و ترانزیستور را به صورت دائم خاموش کرد. به این ترتیب که با گذراندن یک جریان زیاد بین گیت اصلی و درین ترانزیستور، مقداری بار روی گیت شناور قرار می‌گیرد که باعث بالا رفتن ولتاژ آستانه ترانزیستور از حد معمول می‌شود و در نتیجه در ولتاژهای کاری ترانزیستور همیشه خاموش است. برای از بین بردن این بار در EPROM از اشعه ماورای بنفش و در EEPROM از جریان الکتریکی استفاده می‌شود. در شکل (۲-۱۴) ساختار منابع اتصالی مبتنی بر گیت شناور نشان داده شده است.




شکل (۲-۱۴): تکنولوژی گیت شناور

### ۲-۴-۴ ساختار مبتنی SRAM

از سلول حافظه به دو گونه استفاده می‌شود. در روش اول از یک سلول حافظه موقت برای کنترل روشن یا خاموش بودن یک ترانزیستور استفاده می‌شود. به عبارتی خروجی سلول حافظه را به گیت این ترانزیستور متصل می‌کنند. در روش دوم با اتصال خروجی‌های سلول‌های حافظه به ورودی‌های انتخاب یک مالتی پلکسر مشخص می‌کنند که کدام یک از خط‌های ورودی مالتی پلکسر به خروجی متصل می‌شود. همانطور که اشاره شد این روش دارای این ضعف است که باید هر بار که برق سیستم قطع می‌شود، مجدداً محتوای سلول‌های حافظه نوشته شود که این کار نیاز به این دارد که یک حافظه دائمی PROM یا EPROM یا یک دیسک مغناطیسی در کنار FPGA گذاشته شود تا هر بار موقع روشن شدن اطلاعات خودش را از روی آن بخواند. مشکل دیگر این روش حجم زیادی است که سلول حافظه به خود اختصاص می‌دهد. زیرا یک سلول حافظه به ۵ یا ۶ ترانزیستور نیاز دارد. ولی در عین حال مزیت استفاده از حافظه‌های موقت، قابلیت برنامه‌ریزی مجدد آن حتی در داخل مدار می‌باشد. شرکت Xilinx برای برنامه‌ریزی FPGAهای ساخت خود از این نوع استفاده می‌کند. شکل (۲-۱۵) یک نمونه SRAM را نمایش می‌دهد.




شکل (۲-۱۵): سلول SRAM تشکیل شده از ترانزیستور

## ۲-۵ انواع FPGA بر اساس آرایش بلوک‌های منطقی برنامه‌پذیر

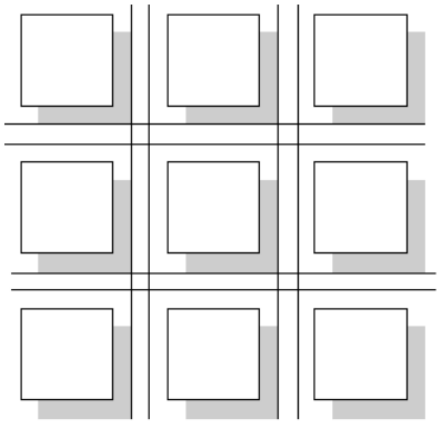
تراشه‌های FPGA را می‌توان از نظر سازماندهی بلوک‌های منطقی به چهار دسته زیر تقسیم‌بندی کرد، این چهار دسته عبارت‌اند از:

- ❖ آرایش آرایه متقارن
- ❖ آرایش سطری
- ❖ آرایش سلسه‌مراتبی
- ❖ آرایش انبوه دروازه‌ها

در ادامه‌ی بحث به شرح این نوع آرایش‌ها پرداخته می‌شود.

### ۲-۵-۱ آرایش آرایه متقارن

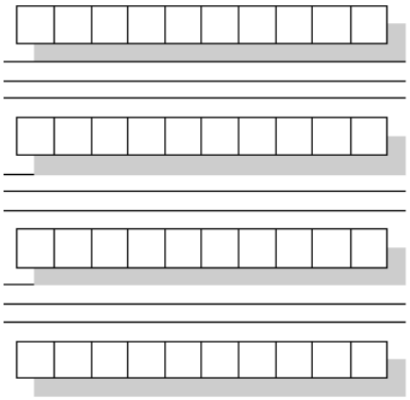
در این روش بلوک‌های منطقی برنامه‌پذیر به صورت یک آرایه دو بعدی سازماندهی شده و خطوط اتصال سیمی به صورت افقی و عمودی از بین آن‌ها عبور می‌کند و سوئیچ‌های برنامه‌پذیر امکان برقراری اتصال بین خطوط سیمی و بین بلوک‌ها و خطوط را فراهم می‌کند. شکل (۲-۱۶) شمای کلی این روش را نشان می‌دهد. شرکت Xilinx در محصولات خود از این آرایش استفاده می‌کند [۳۱].



شکل (۲-۱۶): نمایی از معماری آرایه دو بعدی متقارن در بلوک‌های منطقی برنامه‌پذیر

### ۲-۵-۲ آرایش سطری

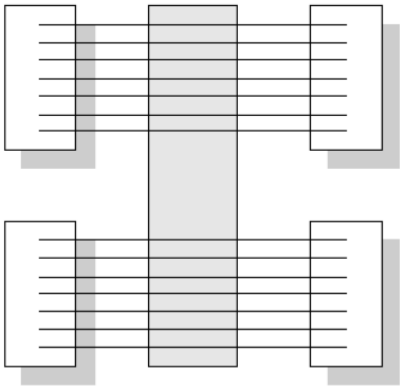
در این روش بلوک‌های منطقی برنامه‌پذیر به صورت سطری چیده شده‌اند و یک سری اتصالی افقی بین سطرهای مجاور وجود دارد. شرکت ACTEL در محصولات خود از این نوع آرایش استفاده می‌کند. شکل (۲-۱۷) ساختار سطری را بطور شماتیک نشان می‌دهد [۳۱].



شکل (۲-۱۷): نمایی از معماری سطری در بلوک‌های منطقی برنامه‌پذیر

### ۲-۵-۳ آرایش سلسله مراتبی

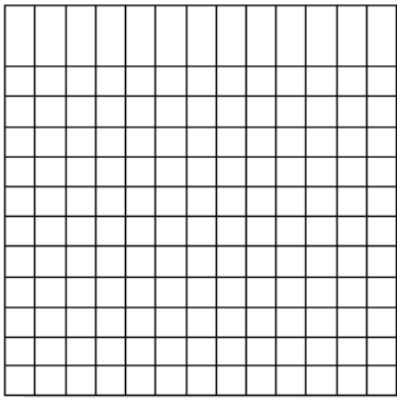
این ساختار شبیه ساختار CPLDهاست. تراشه‌های FPGA شرکت Altera بیشتر از این ساختار پیروی می‌کنند. در شکل (۲-۱۸) نمایی از این نوع معماری نمایش داده شده است [۳۱].



شکل (۲-۱۸): نمایی از معماری سلسله مراتبی در بلوک‌های منطقی برنامه‌پذیر

### ۲-۵-۴ آرایش انبوه دروازه‌ها

این نوع ساختار شامل یک سری دروازه منطقی با قابلیت اتصال به یکدیگر که در یک سطح قرار گرفته و اتصالات قابل برنامه‌ریزی به صورت یک لایه روی آن را پوشانده است. این نوع معماری شبیه به نسل‌های بعدی MPGAها می‌باشد که آن هم توسط شرکت ACTEL در معماری‌های جدیدش استفاده می‌شود. شکل (۲-۱۹) شمایی از این آرایش را نشان می‌دهد [۳۱].



شکل (۲-۱۹): نمایی از معماری دریایی از گیت در بلوک‌های منطقی برنامه‌پذیر

## ۲-۶ مزایای استفاده از FPGAها

FPGAها در پیاده‌سازی توابع نسبتاً پیچیده و دیجیتال به کار می‌روند که نیاز به سرعت پردازش بالایی دارند. علاوه بر این کاهش سخت‌افزار مورد نیاز و همچنین برنامه‌نویسی ساده و استاندارد نیز از

دیگر مزیت‌های استفاده از FPGA است. آنچه که قابلیت و توانایی FPGAها را بالا برده است توانایی‌هایی است که پاره‌ای از آن‌ها در زیر آمده است:

- کاربرد اصلی FPGA در ایجاد هسته‌های پردازشی می‌باشد.
- مدارهای دیجیتال پیچیده به آسانی در آن‌ها پیاده‌سازی می‌شوند و تست مدار سریع است.
- برای تولیدات با تیراژ پایین ارزان تمام می‌شود.
- متناسب با نیاز، تغییرات لازم را در طراحی می‌توان انجام داد و مجداً FPGA را با ساختار جدید برنامه‌ریزی نمود.
- قابل برنامه‌ریزی توسط کاربر است.
- می‌توان چند تا هسته پردازشی داخل یک FPGA تعریف کرد تا در یک زمان واحد چند تا کار را با هم انجام بدهد.
- میکروکنترلرها، DSPها و میکروپروسسورها به صورت سریال دستورالعمل اجرا می‌کنند و قابلیت پردازش بصورت موازی در آن‌ها وجود ندارد، اما در FPGA ذاتاً قابلیت پردازش موازی و انجام عملیات بصورت همزمان را دارد.
- کاهش سخت‌افزار مورد نیاز و همچنین برنامه‌نویسی ساده و استاندارد نیز از دیگر مزیت‌های استفاده از FPGA است.
- سرعت اجرای توابع منطقی درFPGAها بسیار بالا و در حد نانو ثانیه است.
- امکان تعریف هر یک از پایه‌های IC به صورت ورودی یا خروجی یا هر دو.
- مکان برنامه‌ریزی در مدار که این قابلیت را به وجود می‌آورد تا بدون اعمال تغییراتی که سخت‌افزاری هستند و تنها از طریق درگاه برنامه‌ریزی JTAG[1]، برنامه داخلی IC را تغییر داد.

[1] Join Test Action Group (JTAG)

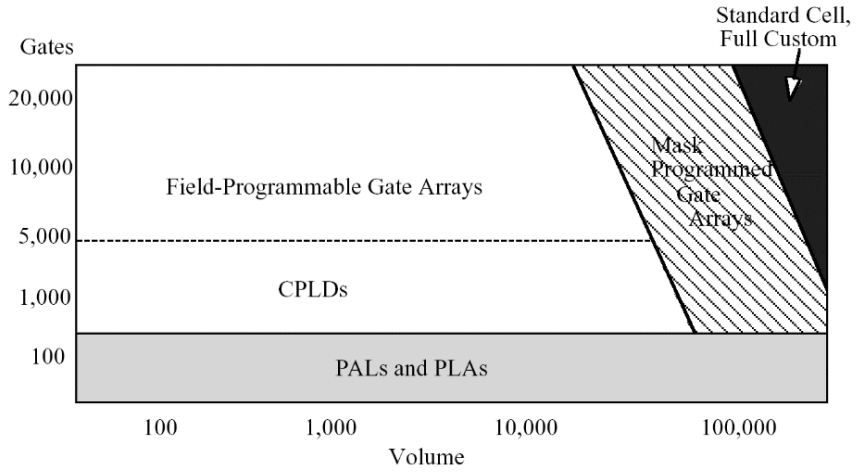
- کاهش حیرت‌انگیز حجم مدار و مجتمع‌سازی در ابعادی تنها به مساحت چند سانتی‌متر مربع.
- کاهش یکسان‌سازی عناصر طراحی و از میان بردن تمامی مشکلات ناشی از عدم تطابق استانداردهای مختلف.
- از میان بردن تمامی نویزهای ناشی از وجود قطعات مختلف و مجزا در مدار.
- کاهش چشمگیر توان مصرفی و اتلاف توان.
- ضریب ایمنی صد در صد به دلیل عدم امکان دستیابی به محتوای داخلی و عدم توان توصیف محتوای داخلی به دلیل انجام ساده‌سازی و فشرده‌سازی بسیار پیچیده.

و بسیاری از قابلیت‌های حیرت‌انگیز دیگر که امکان انجام یک طراحی مجتمع، کم حجم، بهینه و سریع را فراهم می‌آورد.

### ۲-۷ مقایسه FPGA و MPGAها

مزایای عمده FPGA در مقابل MPGA عبارت است از قابلیت برنامه‌ریزی آن توسط کاربر، قابلیت برنامه‌ریزی مجدد و انعطاف‌پذیری که آن را برای ایجاد نمونه اولیه طرح‌ها مناسب نموده است، ضمن اینکه برای تولید با تعداد کم نیز مقرون به صرفه‌تر است. دو عیب اصلی FPGA در مقابل MPGA، سرعت و چگالی کمتر آن می‌باشد. وجود سوئیچ‌های برنامه‌پذیر زمان، تاخیر انتشار را بالا می‌برد و در نتیجه سرعت پایین می‌آید. یک FPGA در مقایسه با MPGA مشابه، حدود سه برابر کندتر است. از طرفی سوئیچ‌ها و عناصر برنامه‌پذیر داخل بلوک‌ها، حجم مدار را افزایش می‌دهد. در نتیجه در شرایط مشابه، چگالی منطقی یک FPGA در مقایسه با یک MPGA حدود ۸ تا ۱۲ برابر کم‌تر است، به همین دلیل یک FPGA با ظرفیت یکسان خیلی گران‌تر است. شکل (۲-۲۰) متناسب با تعداد گیت مورد نیاز برای یک طرح و حجم تولید، تراشه برنامه‌پذیر مناسب را پیشنهاد می‌دهد. محور افقی حجم تولید و محور عمودی بزرگی طرح بر حسب تعداد دروازه مورد نیاز را نشان می‌دهد. با توجه به این نمودار،

FPGA برای کاربردهایی با ۵۰۰۰ تا ۲۰۰۰۰ دروازه و حجم تولید تا ۵۰۰۰۰ عدد عالیست و حتی برای کاربردهای با کمتر از ۵۰۰۰ گیت مناسب است [۴].



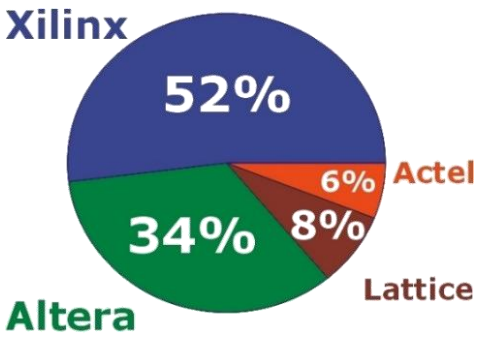

شکل (۲-۲۰): نمودار انتخاب تراشه بر حسب حجم تولید و حجم مدار

## ۲-۸ شرکت‌های سازنده تراشه FPGA

شرکت‌های مهم در زمینه ساخت تراشه‌های FPGA عبارت‌اند از Xilinx، Altera، Lattice و Actel. طبق نمودار شکل (۲-۲۱) شرکت‌های Xilinx و Altera بیشترین سهم تولید را در بازار جهانی دارند. شرکت Xilinx در سال۱۹۸۴ میلادی تاسیس شد، دفتر مرکزی این شرکت در شهر سان‌خوزه در ایالت کالیفرنیای آمریکا می‌باشد. این شرکت بیش از نیمی از محصولات FPGA مورد نیاز در دنیا را تامین می‌کند و همچنین با سازندگان مهمی چون IBM، UMC و Seiko مشارکت دارد. مهم‌ترین نرم‌افزار شبیه‌سازی، سنتز و پیاده‌سازی این شرکت ISE Design Suite نام دارد.

شرکت Altera نیز در سال ۱۹۸۳ میلادی تاسیس شد. دفتر مرکزی این شرکت نیز در شهر سان‌خوزه در ایالت کالیفرنیای آمریکا می‌باشد. ساختار جدول جستجو یا LUT در سال ۱۹۲۲ میلادی توسط این شرکت معرفی شد و همچنین با TSMC در جهت تولید محصولات برنامه‌پذیر مشارکت دارد. مهم‌ترین نرم‌افزار شبیه‌سازی، سنتز و پیاده‌سازی در این شرکت Quartus است. در جدول (۲-۱) تاریخچه ساخت تراشه‌های دو شرکت Xilinx و Altera از سال ۱۹۹۷ تا ۲۰۱۳ میلادی ثبت شده

است. شکل (۲-۲۲) نیز چند نمونه از محصولات FPGA شرکت‌های معروف Xilinx و Altera را نشان می‌دهد [۲۸][۲۷].

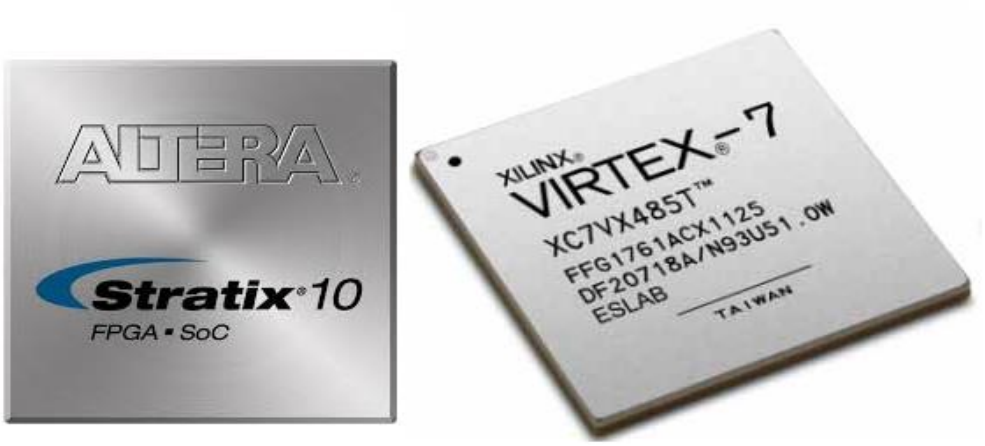



شکل (۲-۲۱): سهم شرکت‌های سازنده از بازار جهانی تولید تراشه FPGA

جدول (۲-۱): تاریخچه تولید محصولات FPGA در دو شرکت Xilinx و Altera

| Year/Company | Xilinx | Altera |
|---|---|---|
| 1997 | - | APEX |
| 1998 | Spartan & Virtex | - |
| 1999 | Virtex-E | - |
| 2000 | Spartan-II & Virtex-EM | - |
| 2001 | Virtex-II | APEX-II |
| 2002 | Virtex-II Pro | Startix & Cyclone |
| 2003 | Virtex-II Pro X | Stratix GX |
| 2004 | - | Startix-II & Cyclone-II |
| 2005 | Virtex-IV & Spartan-3E | Stratix II GX |
| 2006 | Virtex-5 | Stratix III |
| 2007 | Spartan-3A | Arria GX & Cyclone-III |
| 2008 | Virtex-5 FXT | Stratix-IV |
| 2009 | Spartan-6 & Virtex-6 | Arria-II GX & Cyclone-IV |
| 2010 | Virtex-7, Artix-7, Kintex-7 | Stratix-V & Arria-II GZ |
| 2011 | - | Arria-V GX, GT, SX & Cyclone-V |
| 2012 | - | Arria-V GZ |
| 2013 | - | Stratix-10 & Arria-10 GX, GT, SX |


(ب) (الف)




(ت) (پ)

شکل (۲-۲۲): الف) تراشه Xilinx Spartan-6 سری XC6SLX9، ب) تراشه Altera Startix-IV، پ) تراشه Virtex-7، ت) تراشه Startix-10

# فصل سوم

## زبان توصیف سخت‌افزار VHDL

## ۳-۱ مقدمه

VHDL یکی از زبان‌های توصیف سخت‌افزار است که برای طراحی مدارات دیجیتال به کار می‌رود، اصطلاح VHDL از دو بخش VHSIC[1] و HDL[2] تشکیل شده است. در سال ۱۹۸۰ میلادی، وزارت دفاع آمریکا و IEEE (انجمن جهانی مهندسین برق و الکترونیک) با هدف توسعه در مدارات مجتمع پرسرعت، حمایت از این زبان توصیف سخت‌افزار را بر عهده گرفتند. در سال ۱۹۸۶ میلادی نسخه استاندارد IEEE برای VHDL مطرح شد. پس از اعمال بازنگری‌ها و اصلاحات، سرانجام در دسامبر سال ۱۹۸۷ میلادی به نام استاندارد IEEE 1076-1987 به ثبت رسید. پس از آن در سال ۱۹۹۳ میلادی در راستای رفع کاستی‌ها و افزایش قابلیت‌های این زبان، استاندارد IEEE 1076-1993 به وجود آمد. زبان VHDL در حال حاضر یک استاندارد صنعتی برای توصیف سیستم‌های دیجیتال می‌باشد. Verilog زبان توصیف سخت‌افزار دیگری است که بطور گسترده مورد استفاده قرار می‌گیرد. این دو زبان برای توصیف و شبیه‌سازی سیستم‌های پیچیده دیجیتال از توانایی بالایی برخوردار هستند [۳۲].

زبان دیگری که استفاده می‌شود، زبان ABEL[3] است. این زبان بطور خاص برای افزاره‌های منطقی PLD طراحی شده است. ABEL نسبت به دو زبان ذکر شده ضعیف‌تر بوده و در صنعت از محبوبیت کمتری برخوردار است. در این فصل زبان VHDL مورد بررسی قرار می‌گیرد و نویسنده سعی دارد با بیان ساده و قابل فهم، خواننده را در یادگیری سریع این زبان یاری نماید.

گرچه VHDL یک زبان برنامه‌نویسی است و دارای یک مجموعه‌ای از دستور العمل‌هاست، ولی تفاوت‌هایی با سایر زبان‌های برنامه‌نویسی دارد. یکی از مهم‌ترین آن‌ها این است که این زبان ذاتاً موازی است. بطور مثال فرض کنید یک مدار که شامل چند دروازه منطقی است، به زبان VHDL توصیف شده است و معادل هر دروازه یک دستور توصیف شده است. در اجرای برنامه ترتیب نوشتن

[1] Very High Speed Integrated Circuits (VHSIC)
[2] Hardware Description Language (HDL)
[3] Advanced Boolean Equation Language (ABEL)

دستورات مهم نیست و نرم‌افزار شبیه‌ساز تمام دستورات را به صورت موازی می‌بیند. هر دستوری که تغییری در مقدار ورودی آن رخ دهد، اجرا شده و مقدار خروجی به دست می‌آید. از این رو می‌توان گفت که زبان VHDL یک زبان مبتنی بر رخداد است و همروندی دستورات یکی از خصوصیات مهم آن است. البته علاوه بر دستورات همروند، امکان استفاده از دستورات ترتیبی را نیز دارد که در ادامه به آن‌ها اشاره‌ای خواهیم کرد. نکته‌ی دیگر اینکه زبان VHDL امکان مدل کردن تاخیر برنامه‌ها را دارد و این تاخیرها در سنتز مدار هیچ تاثیری ندارد و همچنین این زبان حساس به حروف نیست، یعنی حروف بزرگ و کوچک را بدون تفاوت در نظر می‌گیرد.

آشنایی با این زبان به سه هدف مهم مستندسازی، شبیه‌سازی و سنتز انجام می‌شود. مستندسازی جهت تبادل و انتقال طرح بین طراحان، ارائه در محافل علمی و نگهداری طرح جهت مراجعه و استفاده مجدد می‌باشد. شبیه‌سازی طرح به منظور بررسی نتایج و ارزیابی آن انجام می‌گیرد. فرآیند سنتز به معنی انتقال طرح از حوزه رفتاری به حوزه ساختاری است که معمولاً با هدف پیاده‌سازی انجام می‌شود. مدار طراحی شده با زبان VHDL را می‌توان به صورت مدارهای با کاربرد خاص (ASIC) پیاده‌سازی نمود، اما غالباً مدارات دیجیتال با هدف پیاده‌سازی در FPGA طراحی می‌شوند. برای یادگیری زبان VHDL حداقل دانستن جبر بول و آشنایی با مدارات منطقی ضروری است [۳].

## ۳-۲ ساختار کلی یک برنامه VHDL

در هر برنامه‌ی VHDL سه بخش اساسی وجود دارد که عبارت‌اند از:

- بخش کتابخانه و بسته‌ها
- بخش Entity
- بخش Architecture

در ادامه به شرح این بخش‌ها پرداخته می‌شود [۴].

### ۳-۲-۱ بخش کتابخانه و بسته‌ها

یک کتابخانه[1] را می‌توان محلی در نظر گرفت که کامپایلر در آن اطلاعات مربوط به طراحی را نگهداری می‌کند. یک بسته‌ی VHDL یک فایل یا پیمانه[2] شامل اشیا[3]، انواع داده‌ها[4]، تعریف مولفه‌ها[5]، سیگنال‌ها، توابع و روال‌هایی که معمولاً استفاده شده و ممکن است بین مدل‌های مختلف VHDL بطور مشترک استفاده شود، می‌باشد. کتابخانه استاندارد این زبان را IEEE می‌نامند. هر کتابخانه دارای بسته‌هایی می‌باشد که هر بسته[6] می‌تواند وظیفه خاصی را در صورت فراخوانی انجام دهد. کتابخانه IEEE دارای بسته‌های متعددی است که در ادامه مورد بررسی قرار می‌گیرند، این بسته‌ها عبارت‌اند از:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_textio.all;
use IEEE.std_logic_arith.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_signed.all;
use IEEE.std_logic_unsigned.all;
use IEEE.math_real.all;
use IEEE.math_complex.all;
```

❖ **بسته‌ی std_logic_1164**

با فراخوانی این بسته می‌توان از اعداد تک بیتی، اعداد صحیح و اعداد حقیقی استفاده کرد. معمولاً این بسته در ابتدای هر برنامه‌ی VHDL نوشته می‌شود.

❖ **بسته‌ی std_logic_textio**

زمانی که قصد داریم بر روی داده‌های موجود در یک فایل متنی (.txt) عملیاتی را انجام دهیم، این بسته را باید فراخوانی کرد. این بسته قابلیت خواندن و نوشتن را در یک فایل متنی مقدور می‌سازد.

[1] Library
[2] Module
[3] Objects
[4] Data Type
[5] Component Declaration
[6] Package

❖ **بسته‌ی std_logic_arith**

این بسته محاسبات عددی را در برنامه VHDL فراهم می‌کند.

❖ **بسته‌ی numeric_std**

این بسته نیز همانند بسته‌ی std_logic_arith محاسبات عددی را در فراهم می‌کند.

❖ **بسته‌ی std_logic_signed**

با فراخوانی کردن این بسته، محاسبات عددی برای اعداد علامت‌دار نیز قابل اجرا می‌شود.

❖ **بسته‌ی std_logic_usigned**

با فراخوانی کردن این بسته، محاسبات عددی برای اعداد بدون علامت نیز انجام می‌شود.

❖ **بسته‌ی math_real**

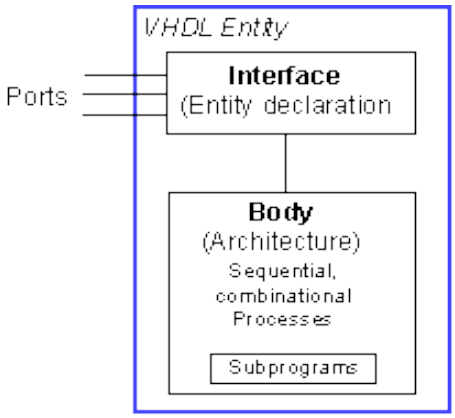
با کمک این بسته محاسبات عددی برای اعداد حقیقی مقدور می‌باشد.

❖ **بسته‌ی math_complex**

با کمک این بسته محاسبات عددی برای اعداد مختلط نیز مقدور قابل اجرا می‌شود.

### ۳-۲-۲ بخش Entity

طرحی که به زبان VHDL صورت می‌گیرد، شامل یک Entity و یک Architecture است که در شکل (۳-۱) نشان داده شده است.




شکل (۳-۱): شماتیک کلی یک توصیف VHDL

در زبان VHDL، یک سیستم در بالاترین سطح شامل یک Entity است که البته می‌تواند در سطوح پایین‌تر شامل Entityهای دیگری که اجزای تشکیل دهنده‌ی Entity در بالاترین سطح هستند، باشند. تعریف Entity شامل یک نام برای Entity و لیست ورودی‌ها و خروجی‌هاست. در زبان VHDL مانند زبان C، برای مشخص کردن انتهای هر دستور از ";" استفاده می‌شود، بنابراین توصیف یک Entity به صورت زیر خواهد بود. لازم به ذکر است که کلماتی که به رنگ سیاه نشان داده شده است، کلمات کلیدی (رزرو شده) هستند.

```
Entity NAME_OF_ENTITY IS
   Port (signal_names: Mode type;
         signal_names: Mode type;
          :
         signal_names: Mode type);
End [NAME_OF_ENTITY];
```

در هر ساختار Entity باید حالت پورت و نوع پورت را مشخص کرد، در ادامه این موارد معرفی می‌شود.

### ۳-۲-۲-۱ پورت‌ها

هر سیگنالی که در قسمت Entity تعریف می‌شود، در قسمت پورت قرار می‌گیرد که در حقیقت به عنوان یک پین در نمای شماتیکی است. هر پورتی که در این قسمت تعریف می‌شود، ابتدا باید مد یا حالت آن سیگنال و سپس نوع داده (data) را مشخص کنیم. حالت یا مد، جهت data که روی آن خط انتقال قرار می‌گیرد را مشخص می‌سازد. مد می‌تواند چهار نوع مختلف به نام‌های In، Out، Inout، Buffer باشد. اگر برای یک پورتی حالت آن را مشخص نسازیم، بصورت پیش فرض حالت آن in قرار می‌گیرد. حال به توضیح این حالت‌ها می‌پردازیم.

#### حالتIn

در این حالت جهت Data به سمت داخل Entity است. مقداردهی یا راه‌اندازی این پورت در

خارج از Entity یا خارج از مدار مورد طراحی انجام می‌شود. معمولاً از این حالت برای ورودی کلاک، ورودی‌های کنترلی و dataهایی که فقط به عنوان ورودی هستند، استفاده می‌گردد.

**حالت OUT**

در این حالت، جهت data به سمت خارج از Entity است و راه‌اندازی این پورت توسط خود Entity صورت می‌گیرد. باید توجه داشت که این پورت را اجازه نداریم به عنوان فیدبک استفاده کنیم. از این حالت برای کلیه سیگنال‌ها خروجی، نظیر خروجی یک کانتر یا یک مقایسه کننده استفاده می‌گردد.

**حالت Buffer**

این حالت شبیه حالت out خروجی بوده و مانند آن راه‌اندازی می‌شود، فقط با این تفاوت که در این حالت می‌توان از این پورت به عنوان فیدبک نیز در مدار استفاده کرد.

**حالت Inout**

از این حالت برای پورت‌های دو جهته نظیر data bus استفاده می‌گردد. بنابراین راه‌اندازی این پورت هم می‌تواند از داخل Entity و هم از خارج آن صورت گیرد. حالت inout اجازه گرفتن فیدبک در مدار را می‌دهد. این حالت می‌تواند جایگزین دیگر حالت‌های buffer و out بشود، چون تمامی خواص آن را دارد.

### ۳-۲-۲-۲ نوع (Type)

بعد از تعیین حالت پورت، باید نوع آن را نیز تعیین کرد. در زیر نوع‌ها را بطور کوتاه شرح می‌دهیم.

**نوع Bit**

با انتخاب این نوع، سیگنال فقط می‌تواند مقدار صفر و یک داشته باشد.

**نوع Bit_vector**

برداری از مقادیر بیتی را می‌توان با انتخاب این نوع برای سیگنال در نظر گرفت.

**نوع Boolean**

سیگنال با این نوع می‌تواند مقدار True یا False را بگیرد.

**نوع real**

سیگنال می‌تواند یک عدد در محدوده‌ی اعداد حقیقی را نگهداری کند.

**نوع Integer**

سیگنال با این نوع یک عدد در محدوده‌ی اعداد صحیح را می‌تواند دریافت کند.

**نوع Time**

این نوع برای نمایش زمان می‌باشد.

### ۳-۲-۳ بخش Architecture

این بخش مشخص می‌کند که عملکرد مدار چیست و یا چگونه ساخته می‌شود. بطور کلی توصیف یک مدار به دو شکل رفتاری و ساختاری ممکن است انجام گیرد، توصیف رفتاری عملکرد و رفتار مدار را بیان می‌کند، به عبارتی دیگر در مدل رفتاری مستقل از ساختار داخلی، ارتباط بین ورودی‌ها و خروجی‌ها به صورت عملیاتی توصیف می‌شود، در توصیف ساختاری دو چیز مشخص می‌شود، اولاً اینکه مدار با چه دروازه‌هایی ساخته می‌شود و ثانیاً اینکه این دروازه‌ها چگونه به هم وصل می‌شوند. در ادامه شیوه کلی نوشتن یک Architecture نشان داده شده است. مستقل از اینکه توصیف رفتاری یا ساختاری است، نام Architecture_Name هر شناسه درستی می‌تواند باشد، گرچه استفاده از شناسه Behavioral و Structural به ترتیب برای مدل‌های رفتاری و ساختاری مرسوم است. بدنه اصلی برنامه VHDL با کلمه کلیدی Begin شروع شده و با End پایان می‌یابد.

```
Architecture Architecture_Name of NAME_OF_ENTITY IS
    -- components declarations
    -- signal declarations
    -- constant declarations
    -- function declarations
  Begin
    -- Statements
  End Architecture_Name;
```

## ۳-۳ انواع ساختارها در زبان VHDL

همانطور که در بخش‌های قبل گفته شد، برنامه‌های VHDL بطور ذاتی موازی هستند، به عبارتی موقع اجرای برنامه همه‌ی دستورات همزمان و موازی بررسی می‌شوند. در مقابل این ساختارهایی از نوع متوالی نیز وجود دارد که می‌توان از آن‌ها نیز استفاده کرد. دانستن چگونگی و محل به کارگیری دستورات و موازی یا متوالی بودن بخش‌های گوناگون یک برنامه VHDL برای یک طراح بسیار ضروری است. در جدول (۳-۱) این نوع ساختارها نمایش داده شده است.

جدول (۳-۱): انواع ساختارهای موازی و متوالی در زبان VHDL

| ساختارهای موازی | ساختارهای متوالی |
|---|---|
| Process | If then else |
| When else | Case statement |
| With select | Variable declaration |
| Signal declaration | Loop statement |
| Block statement | Wait statement |

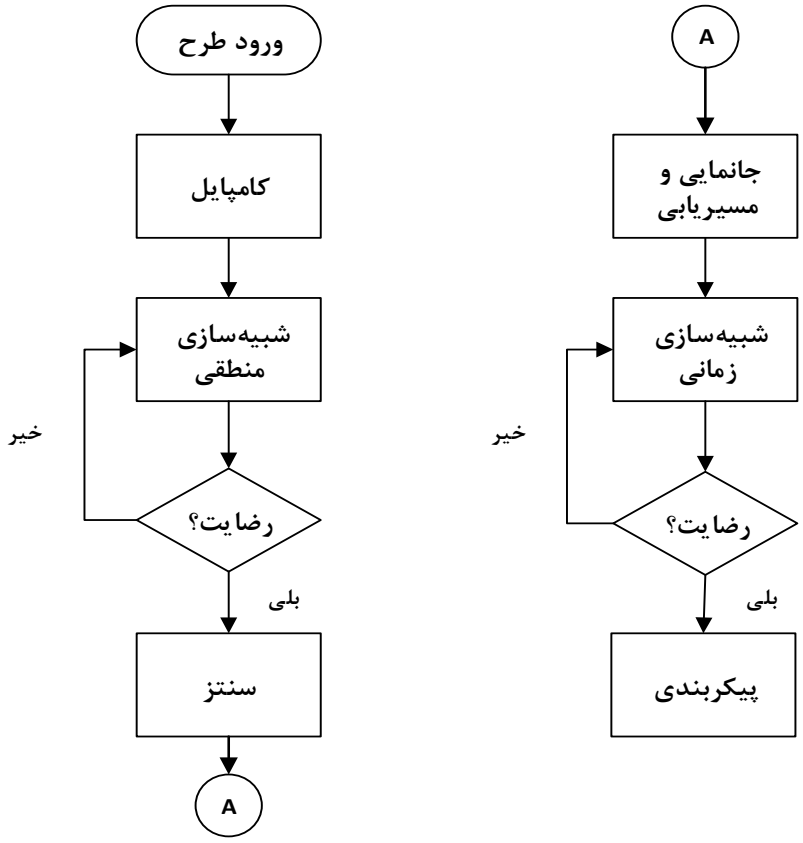
## ۴-۳ پردازش موازی در FPGA

پردازش موازی، اجرای یک فرآیند بطور همزمان روی چندین پردازنده، برای کوتاه کردن زمان رسیدن به جواب است. ایده‌ی این روش بر این مبنا می‌باشد که هر مسئله بطور معمولی قابل قسمت به چندین مسئله به اندازه‌ی کوچک‌تر است که این مسئله‌های کوچک‌تر می‌توانند بصورت همزمان حل شده و در نهایت با هم ترکیب شوند تا نتیجه‌ی نهایی سریع‌تر بدست آید. استفاده از چندین

پردازنده و بکاربستن آن‌ها به بصورت همزمان و در واقع تقسیم برنامه اصلی بین پردازنده‌ها در یک طرح کلی یا استفاده از پردازنده‌هایی که قابلیت اجرای موازی طرح‌ها را دارند، برای این امر راه‌گشا است. توجه به این موضوع که بسیاری از الگوریتم‌های سطح پایین در پردازش سیگنال، دارای عملیات‌های تکراری هستند و ذاتاً برای موازی‌سازی مناسب هستند، می‌توان از موازی‌سازی سخت‌افزاری عملیات پردازش سیگنال، برای بهبود زمان محاسباتی بهره جست. بعنوان مثال برای اعمال موازی‌سازی در پردازش‌های تصویر، تقسیم‌بندی یک تصویر به قسمت‌های جداگانه و توزیع آن‌ها روی شبکه‌ای از پردازنده‌ها این کار مهیا می‌شود. این عمل به موازی‌سازی داده معروف است و یک روش عمومی و مؤثر در پردازش‌های سطح پایین برای موازی‌سازی می‌باشد [۵].

## ۳-۵ مراحل طراحی دیجیتال با تراشه FPGA

شکل (۳-۲) مراحل طراحی با FPGA را در قالب یک فلوچارت نشان می‌دهد [۳].




شکل (۳-۲): فلوچارت مراحل طراحی با زبان توصیف سخت‌افزار VHDL

برای پیاده‌سازی یک طرح دیجیتال به کمک FPGA، مراحل مختلفی وجود دارد که باید به دقت از طرف طراح انجام شوند. البته بخش عمده‌ای از این مراحل به کمک نرم‌افزارهای مخصوصی که برای این کار طراحی شده‌اند انجام می‌شوند. بسته به شرکت سازنده تراشه FPGA باید از نرم‌افزار موردنظر استفاده کرد. در ادامه طبق فلوچارت ارائه شده در صفحه قبل، به ترتیب مراحل طراحی مورد بررسی قرار می‌گیرند.

### ۳-۵-۱ ورود طرح و کامپایل

اولین مرحله طراحی، ورود طرح به ابزار نرم‌افزاری طراحی است. به عبارت دیگر، باید به نحوی طرح را برای نرم‌افزاری که به کمک آن قصد انجام مراحل پیاده‌سازی را داریم، توصیف کرد. برای توصیف طرح، همانطور که در بخش قبل نیز اشاره شد، می‌توان از زبان‌های توصیف سخت‌افزاری مثل VHDL و یا زبان Verilog استفاده کرد. البته معمولاً نرم‌افزارهای ورود طرح، قابلیت توصیف طرح به صورت شماتیکی را نیز دارند که برای توصیف طرح‌های متوسط و بزرگ امکان بکارگیری آن‌ها وجود ندارد. پس از وارد کردن طرح و بررسی خطاهای احتمالی در نوشتن آن (Syntax Error)، باید آن را کامپایل کرد. کامپایل یک توصیف سخت‌افزاری از طرح در اینجا به معنی تبدیل توصیف HDL به مجموعه‌ای از معادلات منطقی است.

### ۳-۵-۲ شبیه‌ساز منطقی

برای ارزیابی صحت طرح معمولاً در این مرحله آن را شبیه‌سازی می‌کنند. شبیه‌سازی در این مرحله را شبیه‌سازی منطقی[1] می‌نامند؛ زیرا آنچه را در این مرحله می‌توان شبیه‌سازی کرد فقط منطق طرح است و مواردی مثل تأخیر گیت‌ها و مسیرها هنوز برای ابزار طراحی شناخته شده نیست.

[1] Functional Simulation

### ۳-۵-۳ سنتز طرح

بعد از اینکه به کمک شبیه‌سازی منطقی از صحت عملکرد طرح اطمینان حاصل شد، باید آن را سنتز نمود. سنتز به طور کلی یعنی تبدیل یک توصیف سطح بالا به توصیف سطح پایین‌تر. سنتز در یک FPGA، یعنی پیاده‌سازی معادلات و روابط منطقی حاصل از مرحله کامپایل به کمک منابع موجود در آن FPGA. به عبارت دیگر، ابزار طراحی باید بتواند به کمک منابعی منطقی که در یک FPGA خاص وجود دارند، مثلLUTها، مالتی‌پلکسرها، ضرب‌کننده‌ها و غیره توصیف سخت‌افزاری مورد نظر را پیاده‌سازی کند. برای این منظور، ابزار طراحی باید نسبت به منابع سخت‌افزاری موجود در FPGA مورد نظر آگاهی کامل داشته باشد. بنابراین باتوجه به اینکه چه نوع FPGA توسط طراح انتخاب می‌شود، نتیجه حاصل از مرحله سنتز می‌تواند متفاوت باشد. در مرحله سنتز، تأخیر گیت‌ها و منابع سخت‌افزاری دیگر که در طراحی بکار می‌روند محاسبه و در نظر گرفته می‌شود. ابزارهای سنتز به دو نوع سنتز RTL[1] و سنتز رفتاری تقسیم می‌شوند. ابزارهای سنتز RTL (سطح انتقال ثبات) یک طراحی VHDL را در قالب ثبات‌ها، ماشین‌های حالت و توابع ترکیبی می‌گیرند و شرح شبکه‌ای از گیت‌ها و سلول‌های کتابخانه‌ای ایجاد می‌کنند. از طرف دیگر، ابزارهای سنتز رفتاری مدل‌های الگوریتمی VHDL را می‌گیرند و به گیت‌ها و سلول‌ها تبدیل می‌کنند. کاربر یک سیستم سنتز رفتاری مثلاً نباید ورودی‌های ساعت را مشخص کند، بلکه فقط باید تعیین کند که عملیاتی مشخص در یک فاصله زمانی معین کامل گردد. ابزارهای سنتر RTL محبوبیت بیشتری نسبت به سنتز رفتاری دارند، بطور مثال نوع ابزار سنتز در نرم‌افزار Xilinx ISE Design به صورت RTL است [۳۳].

### ۳-۵-۴ جانمایی و مسیریابی طرح

مرحله مهم بعدی در طراحی دیجیتال به کمک FPGA، جانمایی[2] است. با توجه به اینکه منابع سخت‌افزاری موجود در هر تراشه FPGA در مکان‌های مشخص و ثابتی درون FPGA قرار دارند، باید

[1] Register Transfer Level (RTL)
[2] Place

مشخص شود که منابع بکار گرفته شده در مرحله سنتز، دقیقاً در چه مکان‌هایی درون FPGA قرار دارند. نرم‌افزار طراحی این کار را با توجه به معیارهای مختلفی انجام می‌دهد. یکی از مهم‌ترین این معیارها این است که منابع را در نقاطی از FPGA قرار دهد که بتواند با مسیرهای کوتاه‌تری آن‌ها را به هم متصل کند. مرحله بسیار مهم دیگر در پیاده‌سازی طرح، مسیریابی[1] است. در این مرحله، ابزار طراحی به کمک الگوریتم‌های پیچیده‌ای سعی در یافتن بهترین مسیرها برای اتصال منابع بکار گرفته شده در طرح مورد نظر می‌کند. با توجه به اینکه خود مسیرها (سیم‌ها) دارای تأخیر هستند، می‌توان در این مرحله نیز مجدداً طرح را شبیه‌سازی زمانی کرد. در این شبیه‌سازی که به آن شبیه‌سازی بعد از جانمایی و مسیریابی می‌گویند، نه تنها تأخیر گیت‌ها و منابع دیجیتالی دیگر، بلکه تاخیر مربوط به سیم‌های متصل کننده منابع دیجیتالی به یکدیگر نیز در شبیه‌سازی در نظر گرفته می‌شود.

### ۳-۵-۵ شبیه‌سازی زمانی

بعد از مرحله سنتز می‌توان مجدداً طرح را شبیه‌سازی نمود. تفاوت این شبیه‌سازی که به آن شبیه‌سازی زمانی[2] می‌شود، با شبیه‌سازی منطقی در این است که تأخیرهای انتشار[3] منابع بکار گرفته شده در طرح نیز در نظر می‌شوند و شبیه‌سازی به آنچه در عمل رخ می‌دهد، نزدیک‌تر است.

### ۳-۵-۶ ساخت فایل پیکربندی و پیکربندی تراشه FPGA

مرحله بعدی طراحی، ساخت فایل پیکربندی[4] است. این فایل که یک فایل باینری (Bit File) است، حاصل تمام مراحل طراحی را به کمک مجموعه‌ای از صفرها و یک‌ها برای ارسال به FPGA در خود دارد. اینکه کدام یک از منابع سخت‌افزاری در FPGA استفاده می‌شود، داخلLUTها چگونه پر می‌شوند و مسیرها چگونه به یکدیگر متصل می‌شوند، همگی درون این فایل باینری قرار دارند. آخرین مرحله از طراحی، پیکربندی FPGA است. در این مرحله، به کمک یک پروگرامر و رابط JTAG موجود

[1] Routing
[2] Timing Simulation
[3] Propagation Delays
[4] Configuration File

در FPGA، فایل باینری پیکربندی به داخل تراشه FPGA بارگذاری می‌شود. پس از بارگذاری این فایل، FPGA به سخت‌افزاری که به کمک زبان HDL توصیف شده بود، تبدیل می‌شود. به جز نرم‌افزارهایی که برای مراحل مختلف کامپایل و پیاده‌سازی توسط شرکت‌های مختلف طراحی شده‌اند، شرکت‌های سازنده تراشه FPGA نیز نرم‌افزارهای بسیار کاملی را ارائه می‌دهند که به کمک آن‌ها می‌توان تمام مراحل طراحی از ورود طرح تا پیکربندی FPGA را انجام داد. باید به این نکته توجه کرد که انجام مراحل پایانی پیاده‌سازی مثل ساخت فایل پیکربندی و بارگذاری آن در FPGA، فقط توسط نرم‌افزارهای شرکت سازنده FPGA مورد نظر قابل انجام است. معمولاً طراحان حرفه‌ای، استفاده از نرم‌افزار ساخت شرکت سازنده تراشه FPGA را ترجیح داده و ممکن است در بعضی مراحل طراحی از نرم‌افزارهای شرکت جانبی دیگر به صورت کمکی استفاده کنند. به عنوان مثال، افرادی که با FPGAهای شرکت Xilinx کار می‌کنند، از نرم‌افزار این شرکت به نام ISE Design Suit استفاده می‌کنند. اما در کنار آن ممکن است برای انجام شبیه‌سازی از نرم‌افزار معروف ModelSim از شرکت Mentor Graphics استفاده کنند. طراحانی که از تراشه‌های FPGA شرکت Altera استفاده می‌کنند، از نرم‌افزار ساخت این شرکت یعنی Quartus استفاده می‌کنند. این طراحان نیز ممکن است برای بعضی از مراحل طراحی مثل شبیه‌سازی یا سنتز، از نرم‌افزارهای دیگر شرکت‌ها نیز کمک بطلبند.

# فصل چهارم

## نویز ضربه‌ای فلفل و نمک

## ۴-۱ مقدمه

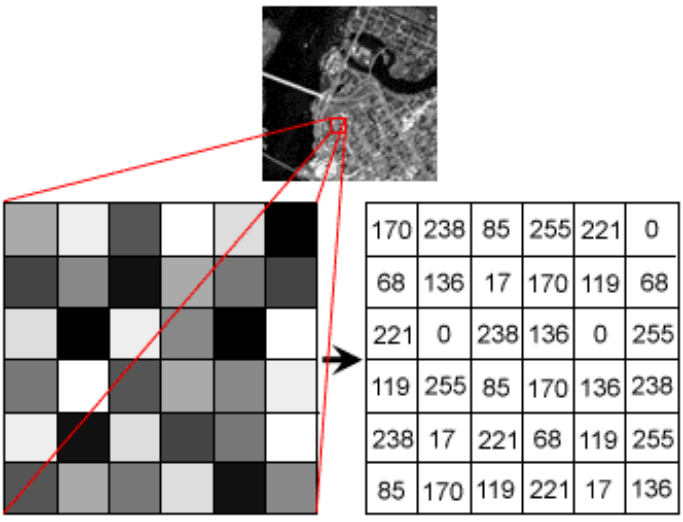
پردازش تصویر روشی برای تبدیل یک تصویر به صورت دیجیتال و انجام برخی از عملیات بر روی آن، به منظور دریافت یک تصویر بهبود یافته و یا برای استخراج برخی از اطلاعات مفید از آن است. در اوایل دهه ۶۰ میلادی، سفینه فضایی رنجر هفت متعلق به ناسا، شروع به ارسال تصاویر تلویزیونی مبهمی از سطح ماه به زمین کرد. استخراج جزئیات تصویر برای یافتن محلی برای فرود سفینه آپولو نیازمند اعمال تصمیماتی روی تصاویر بود. این کار مهم به عهده آزمایشگاه Jet Propulsion قرار داده شد. بدین ترتیب زمینه تخصصی پردازش تصاویر رقومی آغاز گردید و مثل تمام تکنولوژی‌های دیگر سریعاً استفاده‌های متعدد پیدا کرد.

از سال ۱۹۶۴ میلادی تاکنون، موضوع پردازش تصویر رشد فراوانی کرده است. علاوه بر برنامه تحقیقات فضایی، اکنون از فنون پردازش تصویر در موارد متعددی استفاده می‌شود. برای نمونه شیوه‌های رایانه‌ای در پزشکی، تباین[1] تصویر را ارتقا می‌دهند، یا اینکه تصاویر اشعه ایکس یا سایر تصاویر پزشکی، سطوح شدت روشنایی را با رنگ نشانه‌گذاری می‌کنند. متخصصان جغرافیایی نیز از این روش‌ها یا روش‌های مشابه برای مطالعه الگوهای آلودگی هوا که با تصویر برداری هوایی و ماهواره‌ای بدست آمده است، استفاده می‌کنند. در باستان‌شناسی برای تصویر برداری سه بعدی از اجسام مورد استفاده قرار می‌گیرد. در موزه‌ها نیز روش‌های پردازش تصویر برای بازیابی عکس‌های مات شده‌ای که تنها باقی مانده آثار هنری نادر هستند، مورد استفاده قرار می‌گیرد. کاربردهای موفق دیگری از پردازش تصویر را نیز می‌توان در نجوم، زیست شناسی، پزشکی هسته‌ای و صنعت بیان کرد. پردازش تصویر در صنایع مختلف از جمله صنایع دفاعی، صنایع بسته‌بندی و چاپ، صنایع خودرو، داروسازی و پزشکی، صنایع غذایی و صنایع فولاد نیز کاربردهای فراوانی دارد.

[1] Contrast

## ۴-۲ ماهیت پردازش تصویر

در معنای خاص، پردازش تصویر عبارت است از هر نوع پردازش سیگنال که ورودی آن یک تصویر است، مثل عکس یا صحنه‌ای از یک فیلم. خروجی پردازشگر تصویر می‌تواند یک تصویر یا یک مجموعه از نشان‌های ویژه یا متغیرهای مربوط به تصویر باشد. اغلب تکنیک‌های پردازش تصویر شامل برخورد با تصویر به عنوان یک سیگنال دو بعدی و بکار بستن تکنیک‌های استاندارد پردازش سیگنال روی آن‌ها می‌شود. پردازش تصویر اغلب به پردازش دیجیتالی تصویر اشاره می‌کند، ولی پردازش نوری و آنالوگ تصویر هم وجود دارند. تصاویر دیجیتالی از تعداد زیادی مربعات کوچک (پیکسل) تشکیل شده‌اند. هر پیکسل دارای یک شماره رقمی (دیجیتال) می‌باشد که بیانگر مقدار روشنایی آن پیکسل است و می‌تواند مقداری بین صفر (رنگ سیاه) و ۲۵۵ (رنگ سفید) داشته باشد، مقادیر موجود در این بازه را رنگ‌های خاکستری تشکیل می دهند. در شکل (۴-۱) این موضوع به تصویر کشیده شده است.




شکل (۴-۱): نحوه ایجاد یک تصویر خاکستری نمونه

برای مشخص کردن یک پیکسل روش‌های مختلفی استفاده می‌شود، آنچه که متداول‌تر است فضای RGB[1] است، این مدل رنگ، برای ایجاد تصویر در تلویزیون و مانیتورها به کار گرفته می‌شود. در این مدل، تمام رنگ‌ها از ترکیب سه رنگ تشکیل می‌شود. این سه رنگ عبارت هستند از قرمز،

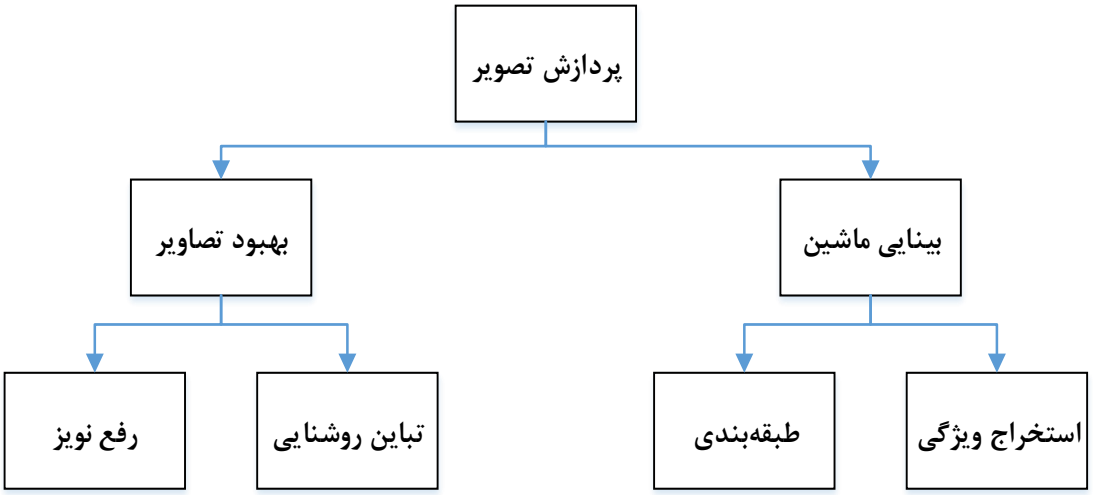
[1] Red, Green and Blue (RGB)

سبز و آبی که به آن RGB اطلاق می‌شود. در صورتی که در ایجاد یک پیکسل از سه بخش قرمز، سبز و آبی استفاده شود و برای هر بخش یک بایت در نظر گرفته شود، هر بخش دارای ۲۵۶ حالت (۲ به توان ۸) خواهد بود. در نتیجه هر پیکسل ۱۶۷۷۷۲۱۶ (۲۵۶ به توان ۳) رنگ مختلف را می‌تواند نشان دهد. در حقیقت این مدل دارای کانال‌های رنگ نیست. شالوده آن نور است و به جای سه رنگ قرمز، سبز و آبی سه نور رنگی داریم. اگر بدانید که نمایشگرها چگونه تصویر را نمایش می‌دهند، خواهید فهمید که غیر از این هم نمی‌تواند باشد. زیرا در مانیتور، تغییر ولتاژ ارسالی به پیکسل‌ها، باعث ایجاد نور رنگی می‌شود. این نورها که با هم ترکیب می‌شوند، در نهایت رنگ پیکسل را می‌سازند. زمانی که به تابش یک نور نیاز نداریم، مانند این است که جلوی آن نور یک تلق مشکی قرار دهیم تا از تابش آن نور جلوگیری کنیم و اصطلاحاً آن را ببندیم. این سیستم رنگ که به عنوان متداول‌ترین سیستم رنگ شناخته شده است، در وسایل گوناگون به صورت‌های مختلفی ظاهر می‌شود. در پردازش تصویر از فضای رنگی دیگری بنام HSI[1] نیز استفاده می‌شود. فضای رنگ HSI شامل سه مولفه رنگ، اشباع و روشنایی است. از آنجا که تعریف رنگ در این فضا بسیار ساده و قابل فهم است، در پردازش تصویر عموماً از این فضای رنگ استفاده می‌شود [۶].

## ۴-۳ حوزه‌های کاری در پردازش تصویر

پردازش تصاویر دارای دو شاخه عمده‌ی بهبود تصاویر و بینایی ماشین است. بهبود تصاویر در برگیرنده‌ی روش‌هایی همچون استفاده از فیلتر محوکننده و افزایش تضاد برای بهتر کردن کیفیت دیداری تصاویر و اطمینان از نمایش درست آن‌ها در محیط مقصد (مانند چاپگر یا نمایشگر رایانه) است، در حالی که بینایی ماشین به روش‌هایی می‌پردازد که به کمک آن‌ها می‌توان معنی و محتوای تصاویر را درک کرد تا از آن‌ها در کارهایی چون رباتیک و محور تصاویر استفاده شود. شکل (۴-۲) این موضوع را به تصویر کشیده است.

[1] Hue, Saturation and Intensity (HSI)




شکل (۴-۲): حوزه‌های کاری در پردازش تصویر

از آنجا که هدف این پروژه رفع نویز ضربه از تصاویر دیجیتالی است، بنابراین در ادامه بحث به حوزه کاری رفع نویز پرداخته می‌شود.

## ۴-۴ نویز در تصاویر دیجیتالی

شاید بتوان به زبان ساده، هدف اصلی علم پردازش تصویر را بدست آوردن اطلاعات مورد نیاز از تصاویر یا قابل استفاده کردن آن‌ها در کاربردی خاص دانست. به عنوان مثال می‌توان به جدا کردن عنبیه از تصویر چشم برای شناسایی هویت افراد اشاره کرد. کیفیت پایین تصویر، بازده فرآیند پردازش را به صورت قابل توجهی می‌کاهد و حتی ممکن است منجر به تصمیم‌گیری نادرست شود. افزایش کیفیت تصاویر شامل دو دسته‌ی کلی ارتقا یا بهسازی تصویر[۱] و بازسازی تصویر[۲] است. ارتقای تصویر شامل روش‌هایی شهودی[۳] است که بدون دانستن مدل خرابی تصویر، سعی در افزایش کیفیت آن برای کاربرد موردنظر دارد. بازسازی تصویر شامل روش‌های عینی[۴] است که با دانستن مدل خرابی و اعمال عکس آن، تصویر را بهبود می‌بخشد. یکی از زمینه‌های مهم افزایش کیفیت تصاویر، حذف نویز تصویر است. نویز به سیگنال‌های ناخواسته‌ای اطلاق می‌شود که باعث تداخل در اطلاعات اصلی شده و

[1] Image Enhancement
[2] Image Restoration
[3] Subjective
[4] Objective

آن‌ها را تحت تأثیر قرار می‌دهد. این زمینه را می‌توان از هر دو دیدگاه بهسازی و بازسازی تصویر بررسی کرد. تفاوت حذف نویز با روش‌های کلاسیک بازسازی تصویر در این است که با وجود دانستن مدل نویز، امکان اعمال عکس مدل خرابی وجود ندارد و از این رو با وجود داشتن مدل نویز در بیشتر مواقع باید روشی شهودی برای افزایش کیفیت تصویر ارائه کرد.

## ۴-۵ علل ایجاد نویز در تصاویر دیجیتالی

با پیشرفت روزافزون دانش مخابرات، نرخ تبادل اطلاعات به صورت چشمگیری افزایش یافته است. با در نظر گرفتن این نکته که حجم بالایی از این اطلاعات به شکل تصاویری می‌باشند که کاربران مورد استفاده قرار می‌دهند، اهمیت پردازش تصویر مشخص می‌شود. به علاوه، در کاربردهای مختلف پردازش تصویر هدف بهبود کیفیت تصویر برای بیننده و یا آماده کردن تصویر برای استخراج اطلاعات نهفته در آن شامل ترکیب‌ها، خصوصیات و ساختارهای مشخص است. برای هر دو کاربرد فوق افزایش کیفیت تصویر جهت بهبود خواص ظاهری تصویر و یا پر کردن جنبه‌ی خاصی از اطلاعات تصویر ضروری است. اما به خاطر طبیعت فیزیکی تصادفی موجود در سیستم‌های تصویر برداری، وجود نویز در تصویر اجتناب ناپذیر است. به عنوان مثال میزان روشنایی و دمای حسگرهای تصویربرداری از مهم‌ترین موارد موثر در میزان نویز تصویر هستند. همچنین از آنجا که حسگرهای تصویر تعداد فوتون‌های دریافتی را، که یک کمیت تصادفی است، می‌شمارند، تصاویر عموماً نویز شمارش فوتون دارند. به علاوه به دلایل مختلف در حین تبدیل تصویر از یک قالب به قالب دیگر مثلاً تصویر برداری، کپی کردن، اسکن کردن، دیجیتال کردن، انتقال در کانال، نمایش دادن، چاپ و یا فشرده‌سازی تصویر، همواره انواع گوناگونی نویز به تصویر افزوده می‌شود.

هر یک از پیکسل‌های سنسور شامل یک یا تعداد بیشتری دیود حساس به نور بوده که نور ورودی به دوربین (فوتون) را به سیگنال‌های الکتریکی تبدیل کرده که این سیگنال‌ها در تصویر نهایی به صورت پیکسل‌های رنگی ظاهر خواهد شد. اگر یک پیکسل مشخص به دفعات با همان میزان نور ثابت در معرض نوردهی قرار گیرد، ارزش‌های رنگی تغییر نخواهد کرد اما تغییراتی به نام نویز در

تصویر ایجاد خواهد شد. حتی بدون در نظر گرفتن نور ورودی، با فعالیت‌های الکتریکی سنسور، سیگنال‌های مزاحمی تولید خواهد شد که با بالا رفتن دمای رنگ میزان آن نیز افزایش پیدا می‌کند. میزان رنگ نهایی پیکسل‌ها می‌بایست بیشتر از گستره نویز بوده تا قابل تشخیص باشد.

نویز تصاویر دیجیتالی همانند دانه‌های برفک در فیلم‌های آنالوگ بیشتر قابل مشاهده خواهد بود. همانطور که قبلاً اشاره شد، با افزایش دمای رنگ، میزان نویز نیز افزایش پیدا می‌کند. با افزایش حساسیت سنسور نیز نویز تصویر به خصوص نویز رنگی در دوربین‌های دیجیتال افزایش پیدا خواهد کرد. یکی دیگر از عوامل بالا رفتن میزان نویز تصویر، کاهش اندازه پیکسل‌هاست که به همین دلیل میزان نویز تصویر ثبت شده توسط دوربین‌های دیجیتالی از نوع کامپکت بیشتر از دوربین‌های دیجیتال SLR[1] می‌باشد. دوربین‌های دیجیتال پیشرفته که مجهز به پردازشگرهای قدرتمند هستند با بکارگیری الگوریتم‌های خاصی جهت حذف نویز، تصویری بدون نویز را به خصوص در حساسیت پایین سنسور در اختیار کاربر قرار می‌دهند. نویز تصویر بیشتر در سطوح آبی و یا قرمز رنگ نسبت به سطوح سبز رنگ قابل رویت می‌باشد. یکی دیگر از علل ایجاد نویز قرار دادن سنسور به مدت زمان طولانی در معرض نور است (بیشتر از ۱ یا ۲ ثانیه)، در این حالت نویز به صورت دانه‌های رنگی در سطح تصویر ایجاد خواهد شد.

## ۴-۶ انواع نویز در تصاویر دیجیتالی

برای حذف نویز، ابتدا باید نویز را شناخت تا بتوان آن را حذف نمود. نویز با توجه به نوع ترکیب با اطلاعات اصلی، به سه نوع ضرب شونده و جمع شونده و نویز ضربه‌ای تقسیم می‌شود. در نویز جمع شونده، همانگونه که از نامش برمی‌آید، اطلاعات ناخواسته با اطلاعات اصلی جمع شده و اطلاعات نویزی مجموعی از هر دو است. این امر در نویز ضرب شونده به صورت ضرب دو عامل نویز و تصویر درنظر گرفته می‌شود. از انواع مهم این نویزها در تصویر، می‌توان به نویز جمع‌شونده‌ی گوسی که معمولاً در تصاویر طبیعی مشاهده می‌شود، اشاره کرد. همچنین مدل‌سازی نویزهای حرارتی نیز با

[1] Single-Lens Reflex (SLR)

استفاده از این نوع نویز انجام می‌شود. نویز ضرب شونده‌ی اسپکل، بیشتر در تصاویر آلتراسوند پزشکی و تصاویر هوایی راداری SAR وجود دارد. دسته آخر انواع نویزها، نویز ضربه‌ای است و ماهیتی متفاوت از دو دسته دیگر دارد. در این نوع نویز، برخی پیکسل‌های تصویر اصلی حذف شده و به جای آن پیکسل‌های نویزی قرار می‌گیرد. همچنین برخلاف نویزهای جمع شونده و ضرب شونده که هرپیکسل تصویر ترکیبی از نویز و تصویر اصلی است، در محل پیکسل‌های نویزی هیچگونه اطلاعاتی از تصویر اصلی در دسترس نیست. نویزهای ضربه‌ای به دو دسته کلی، نویزهای نمک-فلفل و نویزهای ضربه‌ای با مقدار متغیر تقسیم می‌شوند. در نویزهای فلفل و نمک مقادیر پیکسل‌های نویزی، مقادیرکران شدت روشنایی تصویر می‌باشد. به عنوان مثال برای تصاویر هشت بیتی، مقادیر نویزهای فلفل و نمک به ترتیب صفر یا ۲۵۵ است. در نویزهای ضربه‌ای با مقدار متغیر همانطور که از اسم آن‌ها بر می‌آید، مقادیر نویز هر عدد دلخواه در بازه صفر تا ۲۵۵ می‌باشد. در این قسمت به معرفی برخی از نویزهای تصویری معروف پرداخته می‌شود.

### ۴-۶-۱ نویز ضربه

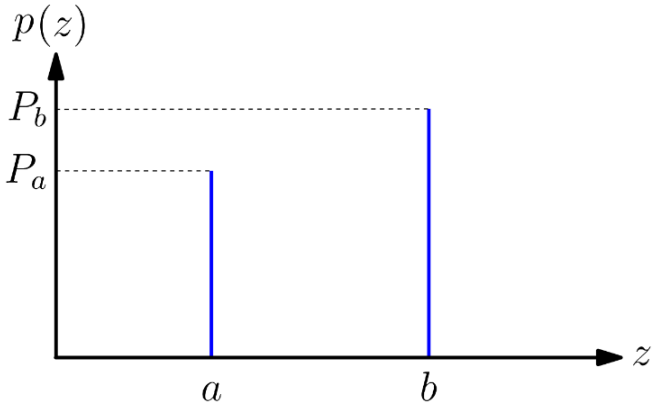
یکی از مباحث مهم در پردازش تصویر، حذف نویز ضربه[1] از تصاویر دیجیتال است. تابع چگالی احتمال نویز ضربه‌ای (دو قطبی) به صورت رابطه (۴-۱) می‌باشد.

$$p(z) = \begin{cases} P_a & z = a \\ P_b & z = b \\ 0 & else \end{cases} \qquad (۱-۴)$$

اگر $a > b$ باشد، شدت روشنایی b به صورت یک نقطه روشن و سطح a به صورت یک نقطه تاریک در تصویر دیده می‌شود. اگر هر یک از $P_a$ یا $P_b$ صفر باشد، نویز ضربه‌ای حاصل یک قطبی نامیده می‌شود. اگر هیچ یک صفر نباشند (و به خصوص اگر هر دو تقریباً مساوی باشند) نویز حاصل شبیه پراکندگی ذرات فلفل و نمک بر روی تصویر خواهد بود. به همین علت نویز ضربه‌ای دو قطبی را نویز فلفل-نمکی نیز می‌خوانند. همچنین نوع تصویر پالس سوزنی (نوک تیز) نیز خوانده می‌شود. ضربه‌های نویز

[1] Impulse

می‌توانند مقدار مثبت یا منفی داشته باشند. معمولاً مقیاس‌گذاری بخشی از فرآیند دیجیتال‌سازی است. از آنجا که آلودگی ضربه‌ای معمولاً نسبت به قدرت سیگنال تصویر بیشتر است، لذا نقاط نویز ضربه‌ای بالاترین مقدار (سیاه و سفید) را پس از دیجیتالی شدن پیدا می‌کنند. بنابراین مقدارهای a و b را با توجه به مقدار حداقل و حداکثر مجاز در تصویر اشباع شده فرض می‌کنیم، پس ضربه‌های منفی به صورت نقاط سیاه (فلفل) و ضربه‌های مثبت به صورت نقاط سفید (نمک) در تصویر مشاهده می‌شوند. در یک تصویر هشت بیتی، این امر نوعاً به معنای a برابر با میزان شدت روشنایی صفر و b برابر با میزان شدت روشنایی ۲۵۵ می‌باشد. شکل (۴-۳) این موضوع را به تصویر کشیده است [۶].




شکل (۴-۳): تابع چگالی احتمال نویز ضربه

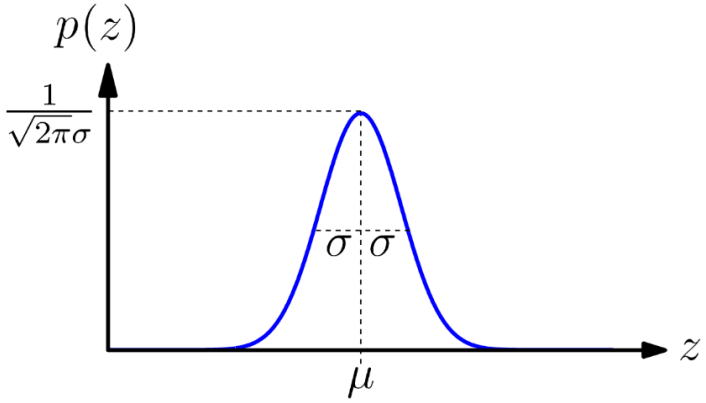
### ۴-۶-۲ نویز گوسی

به خاطر قابلیت ردگیری ریاضی در حوزه‌های زمان و فرکانس، از نویز گوسی (که به آن نرمال هم گفته می‌شود)، در عمل استفاده می‌گردد. در حقیقت این مهارپذیری آن قدر قابل اطمینان است که از این مدل حتی در شرایطی که تنها به طور مرزی قابل اعمال است، نیز استفاده می‌شود .تابع چگالی احتمال یک متغیر تصادفی گوسی z از رابطه (۴-۲) به دست می‌آید.

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} \qquad (۴-۲)$$

که در آن پارامتر μ میزان روشنایی میانگین وپارامتر σ انحراف معیار می‌باشد. در پردازش سیگنال، نویز سفید به سیگنالی گفته می‌شود که در تابع چگالی توان آن، توان بطور یکنواخت در همه‌ی

فرکانس‌ها توزیع شده باشد. چنین نویزی هیچ شباهتی با نسخه‌ی تاخیر یافته‌اش ندارد و تابع خود همبستگی آن به صورت تابع دلتای دیراک است. بطور خاص اگر نمونه‌ها دارای یک توزیع نرمال با میانگین صفر و انحراف معیار متناهی باشند به آن نویز سفید گوسی گفته می‌شود. در شکل (۴-۴) تابع چگالی احتمال نویز گوسی رسم شده است.




شکل (۴-۴): تابع چگالی احتمال نویز گوسی

### ۴-۶-۳ نویز رایلی

تابع چگالی احتمال نویز رایلی[1] توسط معادله (۴-۳) داده می‌شود.

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{\frac{-(z-a)^2}{2}} & for\ z \geq a \\ 0 & for\ z \leq a \end{cases} \qquad (۳-۴)$$

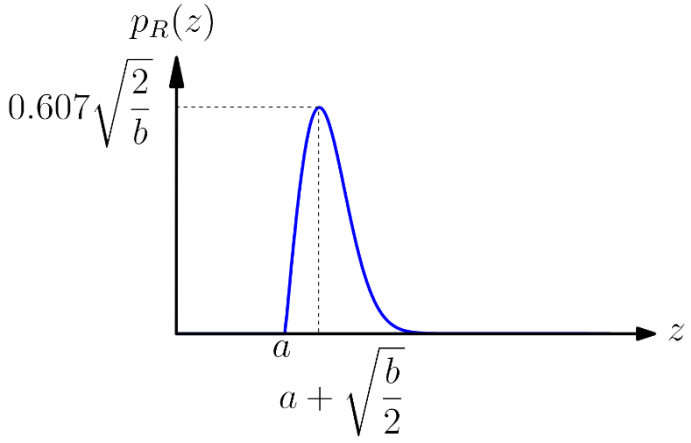
مقدار متوسط و انحراف معیار به ترتیب از رابطه (۴-۴) و (۴-۵) به دست می‌آید.

$$\bar{z} = a + \sqrt{\pi b/4} \qquad (۴-۴)$$

$$\sigma^2 = \frac{b(4-\pi)}{4} \qquad (۵-۴)$$

تابع رایلی برای تقریب زدن هیستوگرام‌های دارای برآمدگی بسیار مناسب است. شکل (۴-۵) تابع چگالی احتمال نویز رایلی را نمایش می‌دهد.

[1] Rayleigh




شکل (۴-۵): تابع چگالی احتمال نویز رایلی

### ۴-۶-۴ نویز ارلانگ (گاما)

تابع چگالی احتمال نویز ارلانگ[1] از تابع (۴-۶) به دست می‌آید.

$$p(z)=\begin{cases}\frac{a^b z^{b-1}}{(b-1)!}e^{-az} & for\ z\geq 0\\ 0 & for\ z<0\end{cases} \qquad (۴-۶)$$

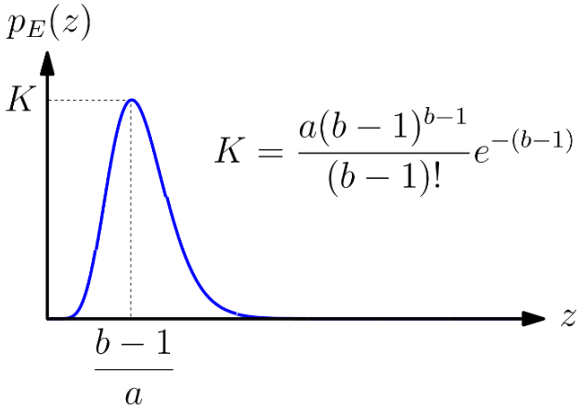
که در آن a و b اعدادی بزرگتر از صفر هستند و "!" علامت فاکتوریل است. میانگین و انحراف معیار این چگالی به ترتیب از روابط (۴-۷) و (۴-۸) به دست می‌آید.

$$\bar{z}=\frac{b}{a} \qquad (۴-۷)$$

$$\sigma^2=\frac{b}{a^2} \qquad (۴-۸)$$

تابع چگالی احتمال نویزگاما در شکل (۴-۶) نشان داده شده است.

[1] Erlang




شکل (۴-۶): تابع چگالی احتمال نویز رایلی

### ۴-۶-۵ نویز نمایی

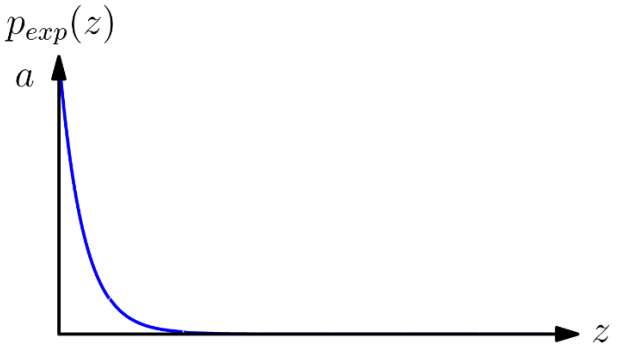
تابع چگالی احتمال نویز نمایی از رابطه (۴-۹) به دست می‌آید و همانند شکل (۴-۷) می‌باشد.

$$p(z) = \begin{cases} ae^{-az} & for\ z \geq 0 \\ 0 & for\ z < 0 \end{cases} \quad (۴-۹)$$

که در آن a بزرگتر از صفر است. میانگین و انحراف معیار این چگالی به ترتیب از رابطه‌های (۴-۱۰) و (۴-۱۱) به دست می‌آید. این تابع چگالی حالت خاصی از تابع چگالی احتمال ارلانگ با b=1 است.

$$\bar{z} = \frac{1}{a} \quad (۴-۱۰)$$

$$\sigma^2 = \frac{1}{a^2} \quad (۴-۱۱)$$




شکل (۴-۷): تابع چگالی احتمال نویز نمایی

### ۴-۶-۶ نویز یکنواخت

تابع چگالی احتمال نویز یکنواخت[1] با رابطه (۴-۱۲) زیر داده می‌شود.

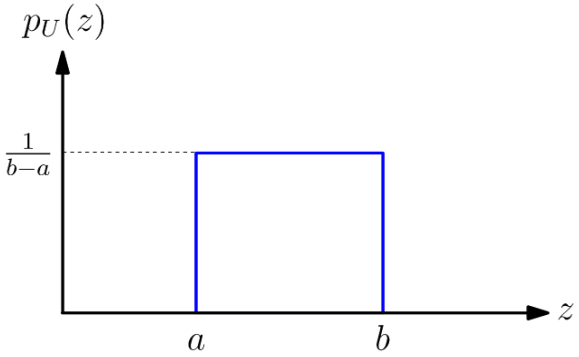
$$p(z) = \begin{cases} \frac{1}{b-a} & for \quad a \le z \le b \\ 0 & else \end{cases} \qquad (۱۲-۴)$$

میانگین و انحراف معیار این چگالی احتمال به ترتیب از روابط (۴-۱۳) و (۴-۱۴) به دست می‌آیند.

$$\bar{z} = \frac{a+b}{2} \qquad (۱۳-۴)$$

$$\sigma^2 = \frac{(b-a)^2}{12} \qquad (۱۴-۴)$$

در شکل (۴-۸) شکل تابع چگالی احتمال نویز یکنواخت نشان داده شده است.




شکل (۴-۸): تابع چگالی احتمال نویز یکنواخت

### ۴-۷ حذف نویز ضربه‌ای فلفل و نمک

با توجه به این که حذف نویز ضربه‌ای فلفل و نمک تصاویر به دلیل رایج‌تر بودن این نویز دارای اهمیت بیشتری است، در این پایان‌نامه علاوه بر ارائه روش پیشنهادی برای رفع این نویز، چند نمونه از فیلترهای حذف نویز ضربه‌ای فلفل و نمک نیز مورد بررسی قرار می‌گیرد. در ادامه به مروری از کارهای گذشته در زمینه حذف نویز ضربه‌ای فلفل و نمک پرداخته می‌شود.

[1] Uniform

### ۴-۷-۱ مروری بر الگوریتم‌های رفع نویز ضربه‌ای

از اولین روش‌های حذف نویز ضربه‌ای، می‌توان به روش فیلترهای خطی و غیرخطی اشاره کرد. از انواع فیلترهای خطی می‌توان فیلتر میانگین استاندارد[1] را نام برد. در این روش، با عبور پنجره‌ای از روی تصویر و میانگین‌گیری از مقادیر و قرار دادن آن بجای پیکسل وسط در پنجره، حذف نویز انجام می‌شود. البته این حذف نویز به قیمت هموارشدن لبه‌ها (مات شدن تصویر) و از دست رفتن فرکانس‌های بالای تصویر می‌باشد [۷]. از آنجایی که نویزهای گوسی معمولاً با میانگین صفر و دارای مقادیر مستقل در نظر گرفته می‌شوند، میانگین‌گیری از آن‌ها می‌تواند تا حدودی نویز گوسی را حذف کند. رابطه شماره (۴-۱۵) نحوه اعمال فیلتر میانگین‌گیر در پنجره همسایگی $S_{xy}$ با ابعاد M×N را نشان می‌دهد.

$$f(x.y) = \frac{1}{MN}\sum_{(s.t)\epsilon Sxy} \quad g(s.t) \qquad (۱۵-۴)$$

خروجی f پیکسل بازیابی شده در مختصات (x,y) و g پنجره همسایگی از تصویر نویزی است. در شکل (۴-۹) روند انجام عملیات میانگین‌گیری در یک پنجره همسایگی با اندازه ابعاد ۳×۳ نشان داده شده است.

| | | |
|---|---|---|
| 10 | 20 | 37 |
| 25 | 255 | 18 |
| 26 | 35 | 15 |

Mean Pixel = 49

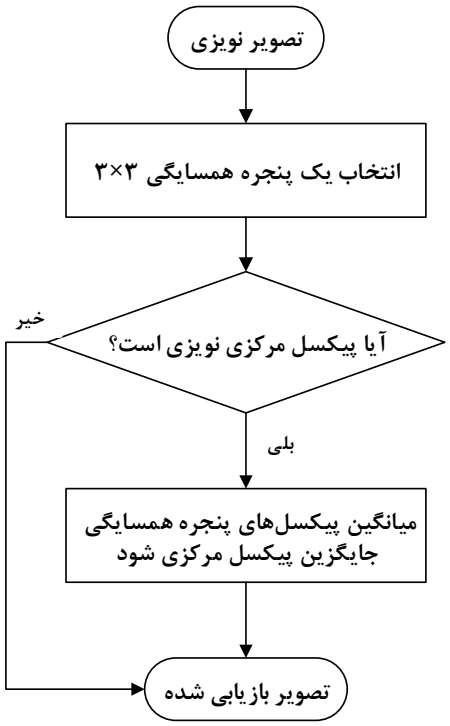
| | | |
|---|---|---|
| 10 | 20 | 37 |
| 25 | 49 | 18 |
| 26 | 35 | 15 |

شکل (۴-۹): روند حذف نویز ضربه‌ای فلفل و نمک در فیلتر میانگین استاندارد

از آنجا که این فیلتر به ازای تمام پیکسل‌ها عملیات میانگین‌گیری را انجام می‌دهد، باعث می‌شود که پیکسل‌های مفید و غیر نویزی تصویر نیز دچار تغییر در میزان شدت روشنایی شوند. برای رفع این

[1] Standard Mean Filter

مشکل از فیلتر میانگین‌گیر تطبیقی[1] استفاده می‌شود. در این فیلتر دو مرحله‌ای، ابتدا عملیات تشخیص نویز صورت می‌گیرد و سپس مرحله رفع نویز انجام می‌شود. با این حال این فیلتر برای چگالی‌های متوسط و بالای نویز فلفل و نمک اصلاً مناسب نمی‌باشد و همچنان مشکلات تاری و از بین بردن درخشندگی تصویر را دارد، پس این موضوع ما را به سمت استفاده از فیلترهای غیر خطی مانند فیلتر میانه سوق می‌دهد. فلوچارت الگوریتم فیلتر میانه تطبیقی در شکل (۴-۱۰) نمایش داده شده است.




شکل (۴-۱۰): فلوچارت الگوریتم فیلتر میانگین تطبیقی

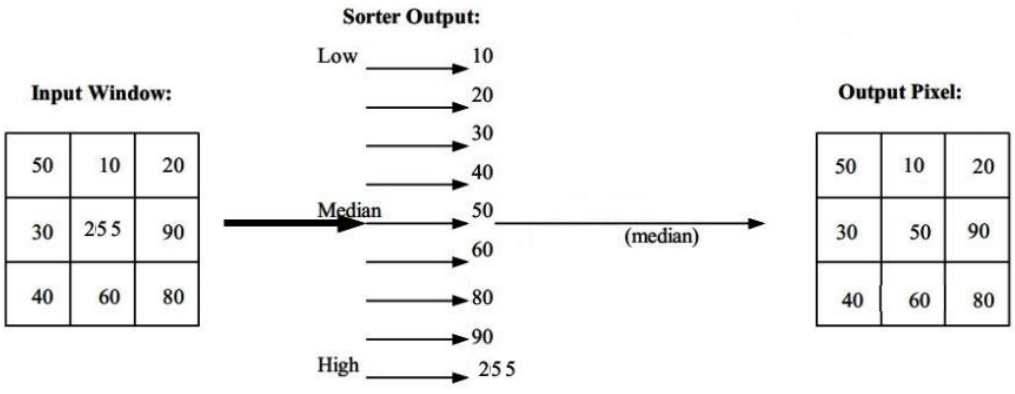
فیلتر میانه استاندارد[2] از بهترین فیلترهای آماری غیرخطی است که برای حذف نویز فلفل و نمک استفاده می‌شود و این تفاوت را با فیلتر میانگین دارد که بجای میانگین‌گیری، میانه‌ی پیکسل‌ها را در خانه‌ی مرکزی پنجره همسایگی قرار می‌دهد [۸]. فیلتر میانه نیز کاربرد تخصصی در حذف نویز ضربه‌ای دارد. در مقادیر کم نویز، پنجره فیلتر شامل مقادیر کمی نویز می‌باشد و با میانه‌گیری از

[1] Adaptive Mean Filter
[2] Standard Median Filter (SMF)

مقادیر پیکسل‌های تصویر، مقدار نویز حذف می‌شود. البته از معایب این روش نیز هموارشدن لبه‌ها است. در مقادیر نویز بالا، میانه‌گیری از پنجره، منجر به حذف نویز نمی‌شود و اکثر نویزهای تصویر باقی می‌مانند. از دیگر مشکلات فیلتر میانه استاندارد این است که عملیات میانه‌گیری برای پیکسل‌های غیر نویزی تصویر نیز انجام می‌شود و این باعث از بین رفتن اطلاعات مفید در تصویر می‌شود. برای رفع این نقص، فیلتر میانه به صورت دو مرحله‌ای استفاده می‌شود. بدین معنی که ابتدا در مرحله‌ی آشکارسازی محل نویزهای ضربه‌ای شناسایی شده و سپس در مرحله‌ی فیلتر، محل نویزها توسط پیکسل‌های سالم اطراف تخمین زده می‌شود. این فیلتر دو مرحله‌ای، فیلتر میانه تطبیقی[1] نام دارد [۹]. این فیلتر در نویزهای با چگالی کم و چگالی متوسط نتیجه‌ای مطلوب دارد با این حال در چگالی‌های بالای نویز فلفل و نمک پاسخ قابل قبولی را ندارد و تصویر خروجی غیر واضح و بی‌کیفیت است. رابطه (۴-۱۶) نحوه اعمال فیلتر میانه در پنجره همسایگی $S_{xy}$ با ابعاد M×N را نشان می‌دهد.

$$f(x.y) = Median\{g(s.t)\} \ , \ (s.t)\epsilon Sxy \qquad (۴-۱۶)$$

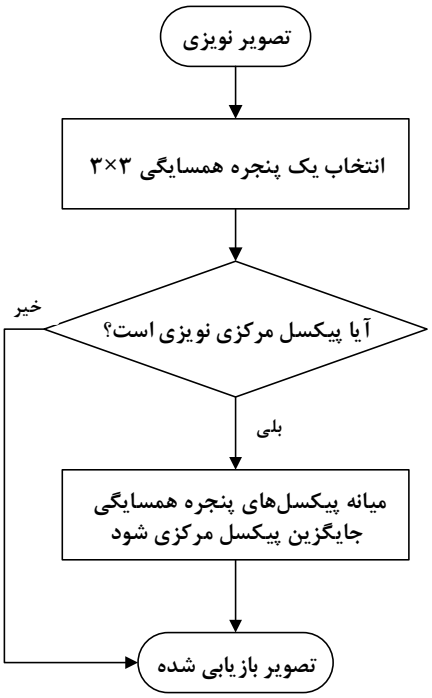
خروجی f پیکسل بازیابی شده در مختصات (x,y) و g پنجره همسایگی از تصویر نویزی است. در شکل (۴-۱۱) روند انجام عملیات میانه‌گیری در یک پنجره همسایگی با اندازه ۳×۳ نشان داده شده است.




شکل (۴-۱۱): روال کار انتخاب پیکسل میانی در فیلتر میانه استاندارد

[1] Adaptive Median Filter (AMF)

در شکل (۴-۱۲) فلوچارت الگوریتم فیلتر میانه تطبیقی نشان داده شده است. این فیلتر دو مرحله‌ای در چگالی‌های نویزی کم و متوسط، از عملکرد نسبتاً خوبی را برخوردار است، در صورتی که در چگالی‌های بالای ۵۰ درصد باعث تخریب شدید لبه‌های موجود در تصویر می‌شود.




شکل (۴-۱۲): فلوچارت الگوریتم فیلتر میانه تطبیقی

در ادامه فیلتر میانه وزن‌دار مرکزی (CWMF)[1] و فیلتر میانه وزن‌دار (WMF)[2] برای بهبود فیلتر میانه ساده با دادن وزنی بیشتر به پیکسل‌های انتخاب شده در همسایگی پیشنهاد شد. اگر چه این دو فیلتر جزئیات بیشتری از تصویر را نسبت به فیلتر میانه حفظ می‌کنند، اما هنوز هم به صورت یکنواخت پیاده‌سازی می‌گردد و باعث می‌شود که پیکسل‌های غیر نویزی نیز دچار تغییر شوند [۱۱][۱۰]. در سال ۱۹۹۹ میلادی، فیلتری با هدف شناسایی پیکسل‌های مخرب و غیر مخرب در یک همسایگی معرفی شد. در این فیلتر پیکسل‌های نویزی شناسایی شده و با مقدار میانه و یا مقداری دیگری جایگزین می‌شود. این فیلتر دارای محاسباتی پیچیده و وقت‌گیر است که پیاده‌سازی

[1] Center-Weighted Median Filter (CWMF)
[2] Weighted Median Filter (WMF)

سخت‌افزاری آن را دشوار می‌کند [۱۲]. دو سال بعد فیلتری وفقی با استفاده از فیلتر میانه مرکزی (ACWMF)[1] برای حذف نویز با چگالی بالا ارائه شد. این فیلتر به بهینه‌سازی یک مقدار آستانه برای نویز ضربه و نویز تصادفی نیاز دارد [۱۳]. در سال ۲۰۰۱ میلادی فیلتر NASM[2] ارائه شد. در این فیلتر عملیات آشکارسازی پیکسل نویزی شامل سه مرحله است. فیلتر NASM در چگالی‌های نویزی ۱۰ تا ۵۰ درصد عملکرد خوبی در حفظ جزئیات تصویر دارد، اما در چگالی‌های ۵۰ درصد به بالا، کیفیف تصویر بازیابی شده به شدت کاهش می‌یابد [۱۴]. چند سال بعد فیلتر Trilatrral برای حذف نویز ضربه‌ای فلفل-نمک و تصادفی پیشنهاد شد. این فیلتر بر اساس اختلاف پیکسل مرکزی با پیکسل‌های همسایگی و همچنین یک محاسبات آماری، اقدام به حذف نویز ضربه‌ای می‌نماید [۱۵].

در سال ۲۰۰۷ میلادی فیلتر تصمیم‌گیری بر اساس الگوریتم (DBA)[3] برای حذف نویز ضربه در چگالی‌های بالا ارائه شد. در این فیلتر پیکسل مخرب با پیکسل میانه موجود در همسایگی جایگزین می‌شود. از آنجا که در نویز با چگالی بالا شاید پیکسل میانه نیز نویز باشد، در این حالت آخرین پیکسل پردازش شده جایگزین پیکسل مخرب خواهد شد [۱۶]. در ادامه فیلتر NIDF[4] ارائه شد. این فیلتر بر اساس قدرمطلق کمترین مقدار از چهار کانولوشن بدست آمده از عملگر لاپلاسین یک بعدی برای پیکسل نویزی تصمیم‌گیری می‌کند. این فیلتر در چگالی‌های بالای نویز عملکرد نسبتاً خوبی دارد [۱۷]. در ادامه و در سال ۲۰۱۰ میلادی، فیلتر ROAD[5] پیشنهاد شد، تصمیم‌گیری این فیلتر بر اساس فیلتر میانه و اختلاف پیکسل مرکزی با سایر پیکسل‌های همسایگی است. با توجه به حجم عملیات، این فیلتر برای پیاده‌سازی‌های سخت‌افزاری مناسب نیست [۱۸]. در همین سال الگوریتم جدیدی با استفاده از یک پیشگویی خطی و فیلتر میانه برای حذف نویز ضربه‌ای فلفل و نمک ارائه شد. این فیلتر عملکرد خوبی در رفع نویز ضربه در چگالی‌های بالا دارد [۱۹].

[1] Adaptive Center-Weighted Median Filter (ACWMF)
[2] Noise Adaptive Soft-Switching Median (NASM)
[3] Decision-Based Algorithm (DBA)
[4] New Impulse Detection and Filtering (NIDF)
[5] Rank Ordered Absolute Difference (ROAD)

در سال ۲۰۱۱ میلادی فیلتر MDBUTMF[1] معرفی شد. عملکرد این فیلتر بر اساس مرتب کردن غیر متقارن در فیلتر میانه است و در چگالی‌های بالای نویز، میانگین تمام پیکسل‌های موجود در همسایگی جایگزین پیکسل مخرب می‌شود [۲۰]. در سال‌های ۲۰۱۳ و ۲۰۱۴ میلادی به ترتیب فیلترهای MNLF[2] و ITMMF[3] ارائه شد که بر اساس ترکیب فیلتر میانه و میانگین برای حذف نویز ضربه عمل می‌کنند. پنجره همسایگیMNLF ابعادی برابر با ۳×۳ دارد. در این الگوریتم اگر تمام پیکسل‌های همسایگی نویزی بود، اندازه ابعاد افزایش می‌یابد، در غیر این صورت پیکسل‌های مخرب حذف می‌گردد و میانه پیکسل‌های باقیمانده جایگزین پیکسل مورد پردازش می‌شود. روش ITMMF دارای سه حالت کلی است، اول اینکه پیکسل‌های پنجره همسایگی همه مخرب هستند، دوم اینکه پیکسل‌های همسایگی ترکیبی از پیکسل‌های نویزی و غیر نویزی هستند و در آخر اینکه پیکسل‌های همسایگی همگی دارای شدت روشنایی صفر یا ۲۵۵ هستند. در این روش طبق وضعیت پیکسل‌های همسایگی یکی از این حالات انتخاب شده و سپس با استفاده از فیلترهای میانه و میانگین برای پیکسل مورد پردازش تصمیم‌گیری می‌شود. این دو فیلتر در نویز بالا عملکرد نسبتاً خوبی دارند [۲۲][۲۱]. در ادامه فیلتر ABMF[4] پیشنهاد شد که بر اساس فیلتر میانه ساده عمل می‌کند و زمانی که پیکسل میانه نویزی بود، پیکسل پردازش شده قبلی جایگزین پیکسل پردازشی می‌شود [۲۳]. در سال ۲۰۱۵ میلادی روش NMF[5] ارائه شد که بر اساس روش ROAD و فیلتر میانه ساده برای حذف پیکسل نویزی تصمیم‌گیری می‌شود [۲۴].

## ۴-۸ شاخص‌های ارزیابی همانندی تصاویر دیجیتالی

در طول استفاده از تصویر، طیف گسترده‌ای از تحریفات شامل پردازش و عملیات‌ها، فشرده‌سازی، ذخیره‌سازی، انتقال و تکثیر ممکن است موجب تخریب کیفیت بصری تصویر شود. برای برنامه‌های

[1] Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF)
[2] Modified Non-Linear Filter (MNLF)
[3] Improved Trimmed Mean Median Filter (ITMMF)
[4] Adaptive Based Median Filter (ABMF)
[5] Novel Median Filter (NMF)

کاربردی که تصاویر در آن نهایتاً توسط انسان مشاهده می‌شود، تنها روش ارزیابی کیفیت تصویر بصری، از طریق ارزیابی ذهنی است. با این حال عمل ارزیابی ذهنی معمولاً ناخوشایند، وقت‌گیر و گران است. در مقابل شاخص‌های ارزیابی عددی زیادی وجود دارد که در کاربردهای مختلف مورد استفاده قرار می‌گیرند. شاخص‌های ارزیابی عددی کیفیت تصویر را با توجه به در دسترس بودن تصویر اصلی می‌توان طبقه‌بندی کرد. در مسائل مربوط به تشخیص کیفیت تصاویر دو شاخه اصلی وجود دارد: تشخیص کیفیت با داشتن تصویر مرجع (FR)[1] یا تشخیص کیفیت بدون داشتن تصویر مرجع (NR)[2]. در نوع اخیر (بدون مرجع) کیفیت یک تصویر بدون داشتن تصویر مرجع تخمین زده می‌شود، در حالی که در نوع اول امکان مقایسه با تصویر اصلی امکان‌پذیر است. کیفیت بدون مرجع به مراتب پیچیده‌تر و مشکل‌تر از روش با مرجع است، به گونه‌ای که تقریباً امروزه تمامی مدل‌های تشخیص کیفیت از نوع همراه با مرجع می‌باشند و مدل‌های بدون مرجع هنوز در مراحل مقدماتی قرار دارند. دسته دیگری از الگوریتم‌های مقایسه به مدل مرجع کاهش یافته معروفند (RR)[3] که در آن تصویر اصلی نه به صورت کامل بلکه تنها مشخصه‌هایی از تصویر مرجع هنگام مقایسه موجود است. این بخش بر روی ارزیابی کیفیت با داشتن تصویر مرجع[4] تمرکز دارد. در ادامه برخی از شاخص‌های معروف و پرکاربرد ارزیابی تصاویر مورد بررسی قرار می‌گیرد.

### ۴-۸-۱ میانگین مربعات خطا

ساده‌ترین و پر استفاده‌ترین روش اندازه‌گیری کیفیت با داشتن تصویر مرجع، میانگین مربع خطا (MSE)[5] می‌باشد. این روش به میانگین‌گیری از مجموع مربع اختلاف روشنایی هر پیکسل تصویر ثانویه از تصویر اصلی (مرجع) می‌پردازد. در رابطه (۴-۱۷) تابع ریاضی این خطا نمایش داده شده است.

[1] Fully Reference (FR)
[2] No Reference (NR)
[3] Reduced Reference (RR)
[4] Full-Reference
[5] Mean Squared Error (MSE)

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[F(i,j) - G(i,j)]^{2} \qquad (۱۷-۴)$$

F(i,j) تصویر اصلی، G(i,j) تصویر بازیابی شده و M×N اندازه ابعاد تصویر اصلی می‌باشد. همانطور که گفته شد در این روش افت کیفیت تصویر بر اساس میزان مقدار متوسط تغییر مقادیر سطوح خاکستری پیکسل‌های مختلف نسبت به مقادیر اصلی به دست می‌آید. این روش به دلیل محاسبات ساده، معانی فیزیکی روشن و ریاضیات مناسبی در زمینه بهینه‌سازی، بسیار پر طرفدارند، اما به خوبی با کیفیت بصری ادراکی منطبق نمی‌شود. در این معیار به واسطه آنکه ساختار تصاویر در چشم انسان مورد توجه قرار نمی‌گیرد، عملاً تمامی پیکسل‌ها نقش یکسانی پیدا می‌کنند، در حالی که مسلماً مقادیر پیکسل‌ها بسته به موقعیت آن‌ها در تصویر تاثیرات متفاوتی را بر روی چشم انسان ایجاد می‌کنند [۲۵].

### ۴-۸-۲ نسبت بیشینه سیگنال به نویز

معیار دیگری که در ارزیابی همانندی تصاویر استفاده می‌شود، نسبت بیشینه سیگنال به نویز (PSNR)[1] است که بر حسب دسی بل با رابطه (۴-۱۸) محاسبه می‌شود. این معیار نیز برگرفته از خطای مربعات میانگین است و همان مشکلات را نیز دارد، با این تفاوت که مقادیر را فشرده می‌کند و مشکل بزرگ بودن اعداد در مقایسات را برطرف می‌کند. پارامتر MAX بیشینه مقدار شدت روشنایی در تصویر است، مثلاً در یک تصویر هشت بیتی، مقدار آن ۲۵۵ می‌باشد [۲۵].

$$PSNR = 10\,Log_{10}(\frac{MAX^{2}}{MSE}) \qquad (۱۸-۴)$$

### ۴-۸-۳ میانگین مطلق خطا

معیار میانگین مطلق خطا (MAE)[2] نیز همانند خطای میانگین مربعات است، با این تفاوت که به جای مربع اختلاف پیکسل‌ها، قدر مطلق اختلاف پیکسل‌ها در نظر گرفته می‌شود و باعث می‌شود که

[1] Peak Signal-to-Noise Ratio (PSNR)
[2] Mean Absolute Error (MAE)

اعداد نهایی این خطا کوچکتر باشند. تابع ریاضی این شاخص ارزیابی در رابطه (۴-۱۹) نشان داده شده است. F(i,j) تصویر اصلی، G(i,j) تصویر بازیابی شده و M×N اندازه ابعاد تصویر مرجع می‌باشد.

$$MAE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}|F(i,j)-G(i,j)| \qquad (۱۹-۴)$$

### ۴-۸-۴ اندازه‌گیری تشابه ساختاری

روش تشابه ساختاری (SSIM)[1] متدی برای ارزیابی همانندی دو تصویر می‌باشد. در ابتدا فرض کنید که x و y دو قطعه از تصاویر مورد مقایسه با مکانی یکسان باشند. شاخص SSIM سه عامل نور I(x,y)، تباین c(x,y) و ساختار s(x,y) را در این دو قطعه مورد بررسی و اندازه‌گیری قرار می‌دهد. نکته قابل توجه این است که این سه جزء نسبتاً مستقل هستند. به عنوان مثال تغییر در نور یا تباین بر روی ساختار تصویر تاثیری ندارد. این شباهت‌های محلی با استفاده از روابط آماری بیان می‌شوند و به صورت رابطه شماره (۴-۲۰) و (۴-۲۱) با هم ترکیب می‌شوند تا شاخص ارزیابی SSIM را ایجاد نمایند.

$$SSIM(x,y) = l(x,y).c(x,y).s(x,y) \qquad (۲۰-۴)$$

$$SSIM(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^{\,2} + \mu_y^{\,2} + C_1}.\frac{2\sigma_x\sigma_y + C_2}{\sigma_x^{\,2} + \sigma_y^{\,2} + C_2}.\frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \qquad (۲۱-۴)$$

که در آن $\mu_x$ و $\mu_y$ به ترتیب میانگین پیکسل‌های واقع در قطعه‌های x و y است. $\sigma_x$ و $\sigma_y$ نیز به ترتیب انحراف معیار پیکسل‌های واقع در قطعه‌های x و y هستند. $\sigma_{xy}$ نیز همبستگی متقابل بین پیکسل‌های موجود در قطعات x و y می‌باشد. مقادیر $C_1$، $C_2$ و $C_3$ اعداد کوچک و مثبتی هستند که از ناپایداری محاسباتی مقادیر کوچک و صفر میانگین و انحراف معیار در روابط بالا جلوگیری می‌کنند.

### ۴-۹ استفاده از FPGA به عنوان بستر پیاده‌سازی در پردازش تصویر

پیشرفت چشمگیر نرم‌افزارهای طراحی اتوماتیک مدارات دیجیتال، طراحی کم هزینه، توان کم و فشردگی مدارات دیجیتال را به دنبال داشته است. پردازش سیگنال‌های دیجیتال اکنون کاربردهای

[1] Structural SIMilarity (SSIM)

زیادی در سیستم‌های مرتبط با پردازش تصویر، پردازش گفتار، مخابرات، مهندسی پزشکی و غیره دارد. طراحی و توسعه‌ی پردازنده‌های DSP[1] قابل برنامه‌ریزی و همچنین راه‌حل‌های اختصاص سیستم روی تراشه (SOC) در طول سه دهه گذشته یک بحث فعال جهت تحقیق و طراحی است. سه فاکتور اساسی که به نظر می‌رسد در پیشرفت سیستم‌های پردازش سیگنال دیجیتال موثر بوده‌اند، عبارت‌اند از یافتن روشی کارآمد برای محاسبه تبدیلات فوریه در دهه ۶۰ میلادی، تجاری شدن ساخت پردازشگرهای دیجیتال که در آن‌ها انجام عملیات ریاضی اعشاری در یک پالس ساعت امکان‌پذیر شد و در پایان، تجاری شدن FPGAهای جدید که پیاده‌سازی‌های پردازش سیگنال کم هزینه و سریع با استفاده از آن‌ها امکان‌پذیر شد. در بخش‌های مختلف پردازش سیگنال برای دستیابی به اهداف مختلف طراحی، پیاده‌سازی‌های متنوعی به صورت نرم‌افزاری و سخت‌افزاری ارائه شده است. با اینکه پیاده‌سازی‌های نرم‌افزاری به کمک پردازنده‌های با هدف عام[2]، چند پردازنده‌ها[3]، میکروکنترلرها[4] و یا حتی پردازنده‌های خاص سیگنال‌های دیجیتال قابل انجام هستند، ساختار ترتیبی اجرای دستورات در آن‌ها (هر چند فرکانس پردازش زیاد باشد) با پردازش‌های سریع مورد نیاز برای کاربردهای زیادی از جمله فشرده‌سازی با رزلوشن بالا، روش‌های رفع نویز، سیستم‌های مخابراتی و ماهواره‌ای متناسب نمی‌باشند. با اینکه تکنولوژی‌های متنوعی وجود دارند، عملاً انتخاب نوع بستر پیاده‌سازی وابسته به نوع کاربرد است.

بسترهای پیاده‌سازی بطور عام در چهار دسته قرار می‌گیرند که ویژگی‌های آن‌ها در جدول شماره (۴-۱) بصورت مقایسه‌ای ارزیابی شده است. با مقایسه کیفی پارامترهای جدول می‌توان دریافت که FPGA راه حل مناسبی برای پیاده‌سازی‌های سخت‌افزاری الگوریتم‌های پردازش تصویر می‌باشد. مدارات مجتمع با کاربرد خاص (ASIC) از نظر توان، سطح و هزینه بسیار قابل توجه هستند. اما این نوع از پیاده‌سازی‌ها به علت هزینه‌ی زیاد، ساخت نمونه‌ی اولیه و طولانی بودن زمان کل فرآیند

[1] Digital Signal Processor (DSP)
[2] General Purpose Processors (GP- Processors)
[3] Multiprocessors
[4] Microcontrollers

طراحی تا تجاری کردن، در بسیاری از شرایط راه حل مناسبی نمی‌باشد. در مقایسه با آن‌ها، FPGAها به علت ساختار موازی، هزینه نسبی کمتر و معماری قابل برنامه‌ریزی، بستر مناسی برای پیاده‌سازی بسیاری از سیستم‌های پردازشی و مخابراتی است. هنگامی که یک سیستم با هزینه و کارایی متوسط مورد نیاز است. می‌توان از پردازنده‌های DSP استفاده کرد. اما کارایی متوسط و ساختار خاص آن‌ها، برای پیاده‌سازی همه‌ی الگوریتم‌های پردازش سیگنال مناسب نمی‌باشد. FPGAها یک مزیت تکنیکی نسبت به این پردازنده‌ها دارند و آن هم در ساخت است که معمولاً از نظر تکنولوژی بسیار پیشرفته‌تر هستند. به علاوه در FPGAهای جدید، بلوک‌های سیگنال دیجیتال خاص[1] تعبیه شده است که کارایی آن‌ها قابل مقایسه با پیاده‌سازی‌های ASIC هستند. یک پردازنده DSP، سری TMS320C6411 که با پالس ساعت ۳۰۰ مگاهرتز کار می‌کند، قابلیت انجام دو عملیات ضرب-جمع هشت بیتی را در یک دوره تناوب دارد. نتیجه اینکه، این پردازنده کارایی برابر با ۶۰۰ MMACS[2] (۳۰۰Meg×۲=) را خواهد داشت [۲۷]. اما استفاده از ۵۸ بلوک اختصاصی DSP48A1 برای انجام عملیات ضرب و جمع ۱۸ بیتی در یک قطعه Xilinx Spartan-6 سری XC6SLX45 که با فرکانس ۲۵۰ مگاهرتز کار می‌کند، کارایی برابر با ۱۴۵۰۰ MMACS (۲۵۰Meg×۵۸=) حاصل خواهد شد که ناشی از ساختار ذاتاً موازی و اختصاصی شده‌ی آن‌ها می‌باشد. در بسیاری از موارد نوع کاربرد در انتخاب بین این دو تعیین کننده خواهد بود [۲۹].

جدول (۴-۱): مقایسه بسترهای مختلف پیاده‌سازی

| Platform | ASIC | DSP | GP-Processor | FPGA |
|---|---|---|---|---|
| **کارایی** | خیلی زیاد | متوسط | خیلی کم | زیاد |
| **هزینه‌ی واحد** | خیلی زیاد | متوسط | خیلی کم | متوسط |
| **توان** | خیلی کم | متوسط | خیلی زیاد | متوسط |
| **انعطاف‌پذیری** | خیلی کم | متوسط | خیلی زیاد | زیاد |
| **پیچیدگی(طراحی)** | خیلی زیاد | متوسط | خیلی کم | زیاد |
| **زمان ساخت (طراحی)** | خیلی زیاد | متوسط | کم | متوسط |
| **چگالی** | زیاد | کم | خیلی زیاد | متوسط |

[1] Dedicated DSP Blocks
[2] Millions of Multiply-Accumulate operations per Second (MMACS)

پردازنده‌های با کاربرد عام، ساده‌ترین نوع بسترسازی هستند و هنگامی که پردازش بلادرنگ (سرعت پردازش) در اولویت قرار نداشته باشد، از آن‌ها استفاده می‌شود. با اینکه این سیستم‌ها کارایی نسبتاً کمتر و توان بیشتری دارند، با توجه به سادگی روند طراحی و هزینه کم، معمولاً مرسوم‌ترین راه برای طراحی اولیه‌ی یک سیستم هستند.

در کاربردهایی که سرعت بالا یکی از مشخصه‌های مهم سیستم باشد، استفاده از تراشه FPGA یکی از بهترین انتخاب‌هاست. مثلاً از FPGAها اغلب برای ساخت دستگاه‌های مخابراتی پرسرعت، دستگاه‌های صنعتی و تجاری خیلی حساس و سریع، دستگاه‌های نظامی و مصارف این چنین بهره می‌برند. FPGAها دارای چند هزار تا چند میلیون گیت در داخل خود هستند، که تعداد گیت‌های یک FPGA بر قیمت آن تاثیر اساسی می‌گذارد. در هر پروژه‌ای شما باید بدانید از چه تراشه‌ای و با چه تعداد گیت استفاده کنید تا بهینه‌ترین حالت ممکن را بوجود آورید.

# فصل پنجم

## روش پیشنهادی برای حذف نویز

## ضربه‌ای فلفل و نمک

## ۵-۱ مقدمه

در این فصل به معرفی روش پیشنهادی برای حذف نویز ضربه‌ای فلفل و نمک از تصاویر خاکستری و رنگی پرداخته می‌شود. در ابتدای فصل به شرح مراحل الگوریتم ارائه شده برای حذف نویز فلفل و نمک اشاره می‌شود. در ادامه ساختار سیستم پیشنهادی مورد بررسی قرار می‌گیرد و در پایان نتایج پیاده‌سازی سخت‌افزاری حذف نویز برای الگوریتم پیشنهادی ارائه می‌شود و سپس با سایر کارهای ذکر شده در فصل چهار، بصورت عددی (با استفاده از شاخص‌های ارزیابی) و بصری (چشمی) مقایسه می‌گردد.

## ۵-۲ برخی روابط پایه‌ای بین پیکسل‌های تصویر

در این بخش به معرفی بعضی از روابط مهم و کاربردی بین پیکسل‌های در یک تصویر دیجیتال پرداخته می‌شود. این روابط در بسیاری از الگوریتم‌های پردازش تصویر از جمله حذف نویز و تباین تصویر استفاده می‌شود. در این بخش تصویر را با $f(x,y)$ نشان داده می‌شود. یک پیکسل P با مختصات $(x,y)$، چهار همسایه افقی و عمودی دارد که مختصاتشان را به صورت زیر است:

$$(x+1,y),(x-1,y),(x,y+1),(x,y-1)$$

این مجموعه از پیکسل‌ها، ۴ همسایگان P و یا پیکسل‌های مرتبه اول P نامیده می‌شود، که به صورت $N_4(p)$ نمایش داده می‌شود. مختصات چهار همسایه‌ی قطری P به شکل زیر است:

$$(x+1,y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1)$$

که با $N_D(p)$ نشان داده می‌شود. این نقاط به همراه ۴-همسایه، ۸-همسایه‌ی پیکسل P نامیده شده و با $N_8(p)$ نشان داده می‌شود. اگر $(x,y)$ در مرز تصویر باشد، برخی از همسایه‌های P در خارج از تصویر دیجیتال قرار می‌گیرند که برای رفع این مشکل اطراف تصویر را صفرگذاری می‌کنند. در شکل (۵-۱) موقعیت همسایگان چهارتایی و همسایگان قطری پیکسل P نشان داده شده است [۶].

| | | |
|---|---|---|
| $P_{i-1,j-1}$ | $P_{i-1,j}$ | $P_{i-1,j+1}$ |
| $P_{i,j-1}$ | $P_{i,j}$ | $P_{i,j+1}$ |
| $P_{i+1,j-1}$ | $P_{i+1,j}$ | $P_{i+1,j+1}$ |

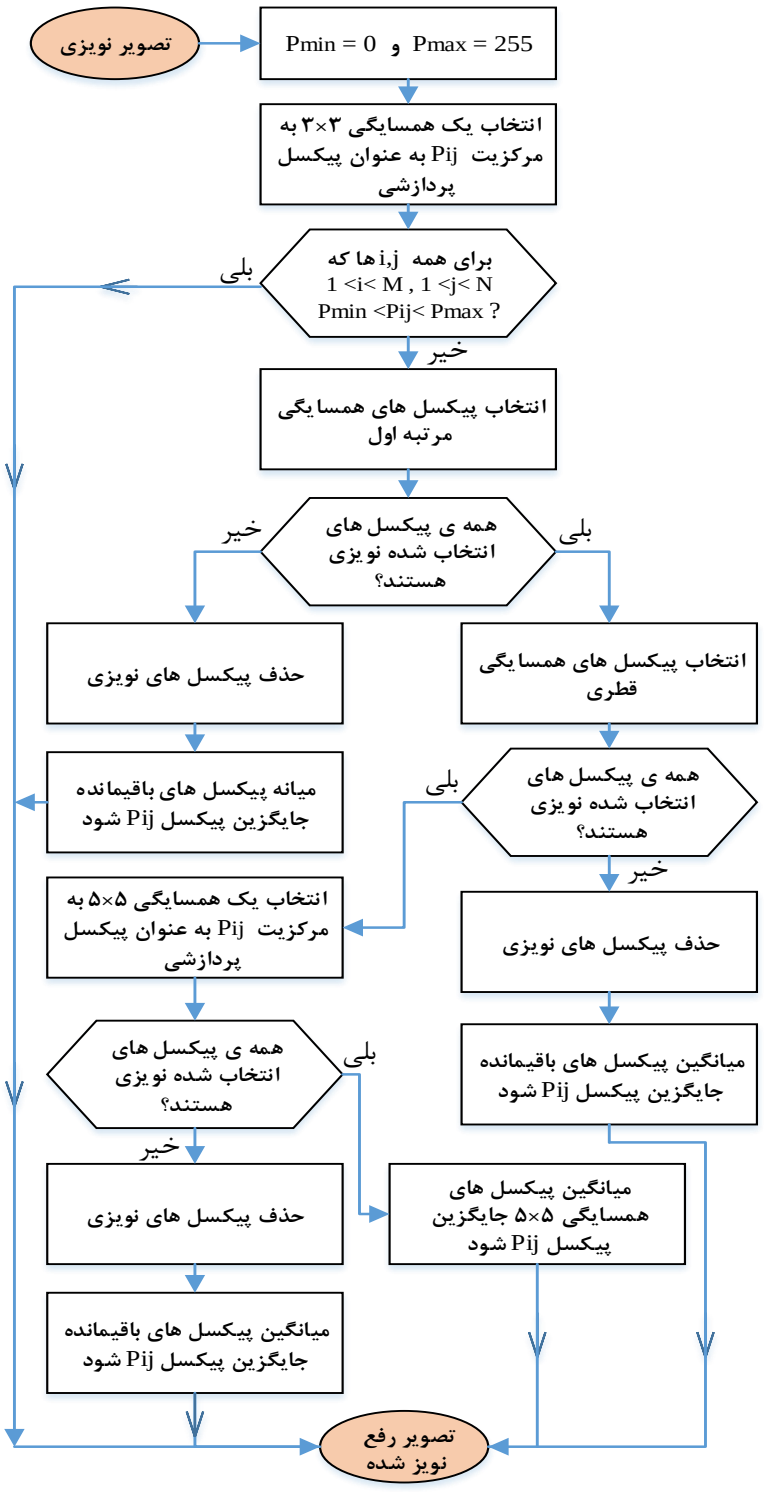
شکل (۵-۱): همسایگی مرتبه اول (آبی رنگ) و همسایگی قطری (سبز رنگ)

از آنجا که در یک پنجره همسایگی ۳×۳ مقدار واقعی پیکسل مرکزی (در صورتی که در محل لبه قرار نداشته باشد) همبستگی زیادی با مقادیر پیکسل‌های همسایگی مرتبه اول و همسایگی قطری دارد، بنابراین با استفاده از این اطلاعات همسایگی می‌توان برای پیکسل نویزی تصمیم‌گیری کرد. در ادامه به معرفی و شرح مراحل مختلف الگوریتم پیشنهادی پرداخته می‌شود.

## ۵-۳ الگوریتم پیشنهادی برای حذف نویز ضربه‌ای فلفل ونمک

در این بخش برای حذف نویز ضربه‌ای فلفل و نمک از تصاویر خاکستری و رنگی، روشی با استفاده از اطلاعات آماری میانه، میانگین و همچنین اطلاعات همسایگی مرتبه اول و اطلاعات همسایگی قطری ارائه می‌شود. در شکل (۵-۲) فلوچارت روش پیشنهادی نشان داده شده است.

الگوریتم پیشنهادی از دو مرحله شناسایی نویز و عملیات رفع نویز تشکیل شده است. در مرحله اول ابتدا تصویر نویزی با اندازه ابعاد M×N دریافت می‌گردد. همانطور که در فصل چهار گفته شد، نویز ضربه‌ای فلفل و نمک به طور تصادفی نقاطی با رنگ‌های سفید (مقدار صفر) و سیاه (مقدار ۲۵۵) بر روی تصویر ایجاد می‌کند که باعث از بین بردن اطلاعات و جزئیات تصویر می‌شود. پس بنابراین برای حذف این نوع نویز، آستانه پایین $P_{min}$ و آستانه بالای $P_{max}$ را برای شناسایی پیکسل‌های نویزی به ترتیب برابر با صفر و ۲۵۵ قرار می‌دهیم.




شکل (۵-۲): فلوچارت الگوریتم پیشنهادی برای حذف نویز ضربه‌ای فلفل و نمک

در ادامه پنجره‌ای مانند شکل (۵-۱) به عنوان پیکسل‌های همسایگی برای تصمیم‌گیری در مورد پیکسل نویزی، در نظر گرفته می‌شود. همانطور که گفته شد، الگوریتم پیشنهادی دارای دو مرحله

شناسایی نویز و عملیات رفع نویز می‌باشد. در ادامه به معرفی مراحل این الگوریتم پرداخته خواهد شد. برای تمام پیکسل‌های $P_{ij}$ که در آن $1<i<M$ و $1<j<N$ روال زیر انجام می‌شود.

### ۵-۳-۱ مرحله شناسایی نویز ضربه

اگر $0<P_{ij}<255$ بود، پس بنابراین پیکسل مرکزی $P_{ij}$ غیر نویزی است و از آنجا که الگوریتم به صورت تطبیقی عمل می‌کند، پیکسل $P_{ij}$ بدون تغییر می‌ماند و پنجره همسایگی یک واحد جابه‌جا می‌شود.

### ۵-۳-۲ مرحله رفع نویز ضربه

در صورتی که پیکسل مرکزی $P_{ij}$ در بازه مورد نظر نبود، پس به احتمال زیاد $P_{ij}$ پیکسلی مخرب است و باید مقداری مناسب برای آن تعیین گردد. مقادیر واقعی شدت روشنایی برخی از پیکسل‌های تصویر می‌تواند اعداد صفر و ۲۵۵ باشند که البته تعداد آن‌ها در یک تصویر با روشنایی متوسط، کم است. از آنجا که در یک پنجره همسایگی ۳×۳ مقدار واقعی پیکسل مرکزی (در صورتی که در محل لبه قرار نداشته باشد) همبستگی زیادی با مقادیر پیکسل‌های همسایگی مرتبه اول و همسایگی قطری دارد، بنابراین با استفاده از این اطلاعات همسایگی، برای پیکسل نویزی تصمیم‌گیری می‌شود.

### ۵-۳-۳ اطلاعات همسایگی مرتبه اول

در ادامه پیکسل‌های همسایگی مرتبه اول که طبق شکل شماره (۵-۱) شامل پیکسل‌های $P_{i-1,j}$، $P_{i,j-1}$، $P_{i+1,j}$ و $P_{i,j+1}$ است، انتخاب می‌شود و سپس پیکسل‌های مفید و غیر نویزی از آن‌ها استخرج می‌شود. اگر تعداد پیکسل‌های غیر نویزی در همسایگی مرتبه اول بزرگتر از صفر بود، میانه آن‌ها محاسبه شده و جایگزین پیکسل مرکزی $P_{ij}$ می‌شود. در صورتی که همه‌ی پیکسل‌های همسایگی مرتبه اول نویزی باشد، از اطلاعات موجود در همسایگی قطری استفاده می‌شود.

### ۵-۳-۴ اطلاعات همسایگی قطری

در این قسمت ابتدا پیکسل‌های همسایگی قطری که شامل پیکسل‌های $P_{i-1,j-1}$، $P_{i-1,j+1}$، $P_{i+1,j-1}$ و $P_{i+1,j+1}$ است، انتخاب می‌شود و سپس پیکسل‌های غیر نویزی را از آن‌ها استخراج کرده و میانگین آن‌ها جایگزین پیکسل مرکزی $P_{ij}$ می‌شود. در صورتی که همه‌ی پیکسل‌های موجود در همسایگی قطری نیز نویزی بودند، اندازه ابعاد پنجره همسایگی افزایش داده خواهد شد.

### ۵-۳-۵ افزایش اندازه ابعاد پنجره همسایگی

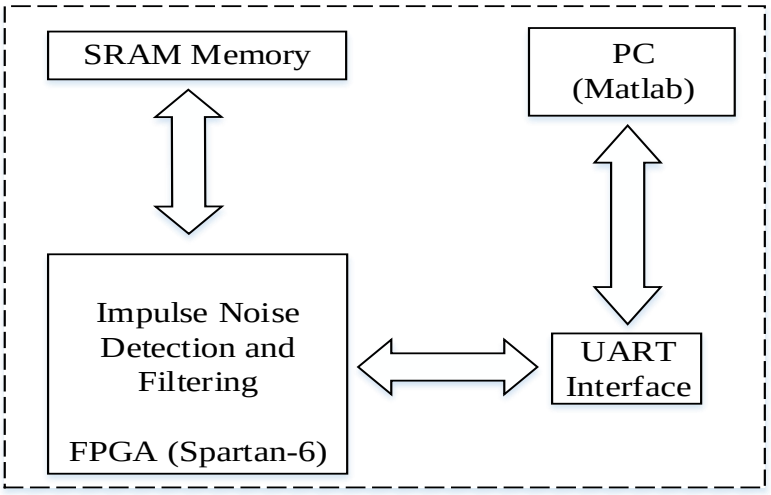
با توجه به این که کلیه پیکسل‌های همسایگی ۳×۳ نویزی بود، اندازه ابعاد پنجره به ۵×۵ افزایش داده می‌شود. در ادامه پیکسل‌های غیر نویزی موجود در پنجره ۵×۵ را استخراج کرده و میانگین آن‌ها را جایگزین پیکسل مرکزی $P_{ij}$ می‌شود. در صورتی که همه‌ی پیکسل‌های همسایگی نویزی بود، میانگین کل پیکسل‌های موجود در پنجره ۵×۵ محاسبه شده و سپس جایگزین پیکسل مرکزی $P_{ij}$ می‌شود. این روند برای کل پیکسل‌های تصویر نویزی انجام می‌شود تا تصویر رفع نویز شده بازیابی شود.

## ۵-۴ ساختار سیستم پیشنهادی

ساختار سیستم پیشنهادی برای رفع نویز ضربه‌ای فلفل و نمک در شکل (۵-۳) نشان داده شده است، این سیستم از چهار بخش اصلی کامپیوتر، ارتباط سریال (UART)[1]، تراشه Xilinx Spartan-6 و حافظه خارجی SRAM[2] تشکیل شده است. از آنجا که نرم‌افزار متلب ارتباط سریال را پشتیبانی می‌کند، با استفاده از متلب نسخه 2015b، اطلاعات تصویر نویزی از طریق ارتباط سریال به پردازنده FPGA ارسال می‌شود و سپس پردازنده آن‌ها را به صورت بایت به بایت در حافظه خارجی SRAM ذخیره می‌کند. در ادامه بر اساس الگوریتم پیشنهادی، پیکسل‌های مدنظر از حافظه خارجی فراخوانی شده و پردازش می‌شوند و مجدداً پیکسل‌های رفع نویز شده در حافظه خارجی ذخیره می‌گردد. در

[1] Universal Asynchronous Receiver and Transmitter (UART)
[2] Static Random Access Memory (SRAM)

نهایت نیز تصویر بازیابی شده با استفاده از ارتباط سریال برای نرم‌افزار متلب جهت نمایش ارسال خواهد شد. میزان پالس ساعت مورد استفاده برای پیاده‌سازی این سیستم، ۲۴ مگاهرتز است که این مقدار از روی بورد FPGA قابل تامین می‌باشد. در ادامه به جزئیات این سیستم پرداخته می‌شود.




شکل (۵-۳): ساختار سیستم پیشنهادی برای رفع نویز ضربه‌ای فلفل و نمک

### ۵-۴-۱ کامپیوتر

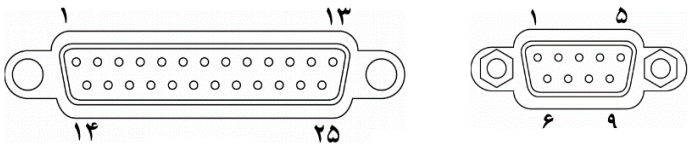
در سیستم پیشنهادی، کامپیوتر نقش ارسال تصویر نویزی و دریافت تصویر رفع نویز شده را بر عهده دارد. همانطور که گفته شده نرم‌افزار متلب ارتباط سریال را پشتیبانی می‌کند، پس بنابراین در اینجا با استفاده از متلب نسخه R2015b در یک کامپیوتر۶۴ بیتی با پردازنده Core(TM)i5-3337U و فرکانس کاری ۱/۸ گیگاهرتز که میزان حافظه RAM آن شش گیگابایت است، اطلاعات تصویر نویزی برای تراشه FPGA از طریق ارتباط سریال ارسال می‌شود و در پایان نیز اطلاعات تصویر بازیابی شده جهت نمایش دریافت می‌گردد. لازم به ذکر است که برای ایجاد یک ارتباط سریال صحیح، ابتدا باید پارامترهایی از قبیل تعداد بیت‌های داده، بیت شروع، تعداد بیت توقف، نرخ ارسال یا باودریت و بیت خطایاب (توازن) را بین فرستنده و گیرنده تعیین کرد. این موضوع در ادامه بیشتر شرح داده شده است.

### ۵-۴-۲ ارتباط سریال

درگاه سریال استاندارد RS-232، یکی از عمومی‌ترین ارتباطات خارجی کامپیوترها تا چند سال اخیر بود و توانایی اتصال دستگاه‌های مختلفی از جمله مودم، اسکنر، پرینتر و غیره را دارا می‌باشد. ولی به علت نیاز به سرعت‌های بالاتر در کاربرد کنونی، امروزه پورت‌های دیگری در حال جایگزین شدن با این پورت می‌باشند. به عنوان مثال پورت موازی که می‌تواند هشت خط ارتباطی را به طور همزمان ارسال کند و یا پورت‌های USB-2 و USB-3 که توانایی ارسال فوق‌العاده سریع اطلاعات را دارند، جایگزین‌های مناسبی برای این پورت می‌باشند. با این حال استفاده از این پورت نه تنها هنوز منسوخ نشده، بلکه در پاره‌ای از مواقع به دلایل اقتصادی و فنی، کاربرد این پورت بر پورت‌های دیگر از ارجحیت بیشتری برخوردار است. UART به معنای فرستنده و گیرنده جامع غیرهمزمان می‌باشد. کار آن در عمل تبدیل بیت‌های دریافتی از درگاه سریال (و یا هر نوع اطلاعات سریال دیگر) به اطلاعات موازی می‌باشد. می‌دانیم که اطلاعات در پورت سریال بـه صورت بیت‌های پشت سرهم دریافت می‌شوند و نیز می‌دانیم که برای کامپیوتر یک بیت به تنهایی معنایی ندارد و بایت‌ها هستند که با معنی هستند. UART نیز بایت‌های ورودی از پورت سریال را بافر کرده و هنگامی که تعداد آن‌ها به یک بایت رسید، آن‌ها را جهت پردازش ارسال می‌کند. در شکل (۵-۴) طرز کار UART نشان داده شده است.


شکل (۵-۴): طرز کار ارتباط سریال UART

خروجی پورت سریال در پشت کامپیوتر تعبیه شده تا بتوان به سهولت دستگاه‌های خارجی را بدون باز کردن نقاب کامپیوتر به آن متصل کرد. این پورت از نوع نر بوده و معمولاً دارای ۹ پین و یا ۲۵ است. در شکل (۵-۵) نمایی از این پورت نمایش داده شده است.




(ب) (الف)

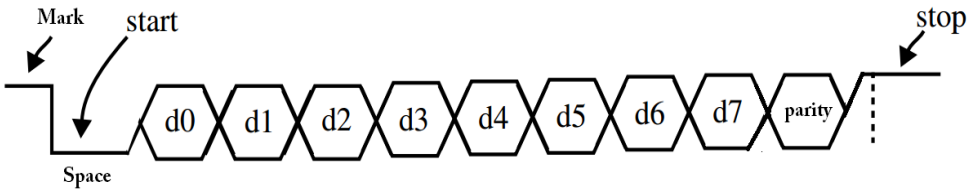
شکل (۵-۵): الف) پورت سریال ۹ پایه-نوع D ،ب) پورت سریال ۲۵ پایه-نوع D

ارتباطات سریال به دو دسته همزمان[1] و یا غیرهمزمان[2] تقسیم می‌شوند. ارتباط همزمان با استفاده از یک پالس ساعت[3] زمان‌بندی انتقال هشت بیت اطلاعات را انجام می‌دهد. این نوع انتقال، نسبت به روش غیرهمزمان از سرعت بیشتری برخوردار است. با این حال بـه خاطر مدارات پیچیده و کابل‌کشی بیشتری که در این نوع انتقال مـورد نیاز است، در عمل از ارتباط غیرهمزمان استفاده می‌شود. ارتباط غیرهمزمان که در صنعت کامپیوتر کاربرد بیشتری دارد، به علت فقدان یک ساعت هماهنگ کننده بین فرستنده و گیرنده، می‌بایست با سرعت یکسانی بین هر دو پایانه ارتباطی برقرار گردد. این سرعت با نام نرخ ارسال[4] شناخته می‌شود. نرخ انتقال با سرعت انتقال بیت‌ها متفاوت است. در این ارتباط فرستنده و گیرنده هر دو بسته به ساعت داخلی سیستم‌های خود، سیگنال‌هایی بـا نرخ انتقال انتخابی ارسال می‌کنند. نرخ انتقال فرستنده و گیرنده می‌تواند از ۳ تا ۵ درصد متفاوت باشند. انتقال اطلاعات در ارتباط غیرهمزمان با یک سیگنال ممتد بالا (Mark)که ولتاژ منفی بین ۳- تا ۲۵- می‌باشد آغاز می‌شود. در این حالت هنوز هیچ اطلاعاتی منتقل نشده‌است. شروع ارسال با تغییر وضعیت از ولتاژ منفی به مثبت می‌باشد. این تغییر، خط را برای مدت زمان یک بیت در حالت پایین

[1] Synchronous
[2] Asynchronous
[3] Clock Pulse
[4] Baud Rate

(Space) که ولتاژ مثبتی بین ۳+ تا ۲۵+ می‌باشد، نگه می‌دارد، این بیت به بیت شروع[1] نیز مشهور است و نشان می‌دهد که هفت یا هشت بیت اطلاعات[2] در حال دریافت شدن می‌باشند. تعداد بیت‌ها که هفت یا هشت می‌باشند را می‌توان در تنظیمات نرم‌افزاری تغییر داد. پس از اتمام این هفت یا هشت بیت، یک بیت اختیاری به عنوان بیت خطایاب[3] جهت اطمینان از صحیح ارسال شدن اطلاعات بیت‌ها فرستاده می‌شود. پس از منتقل شدن بیت خطایاب یک یا چند بیت توقف[4] ارسال می‌شود تا اطمینان حاصل شود که خط در وضعیت Mark قرار دارد و برای انتقال بایت بعدی آمادگی دارد.

در حین انتقال اگر بیتی می‌بایست صفر باشد با ولتاژ مثبت یا Space و در صورت نیاز به بیت از ولتاژ منفی یا Mark استفاده می‌شود. همانطور که در شکل (۵-۶) نشان داده شده است، هشت بیت بلافاصله پس از بیت شروع ارسال شده‌اند و یک بیت خطایاب نیز پس از آن ارسال گردیده و سپس خط به حالت اولیه خود بازگشته است.




شکل (۵-۶): ترتیب ارسال اطلاعات در ارتباط سریال

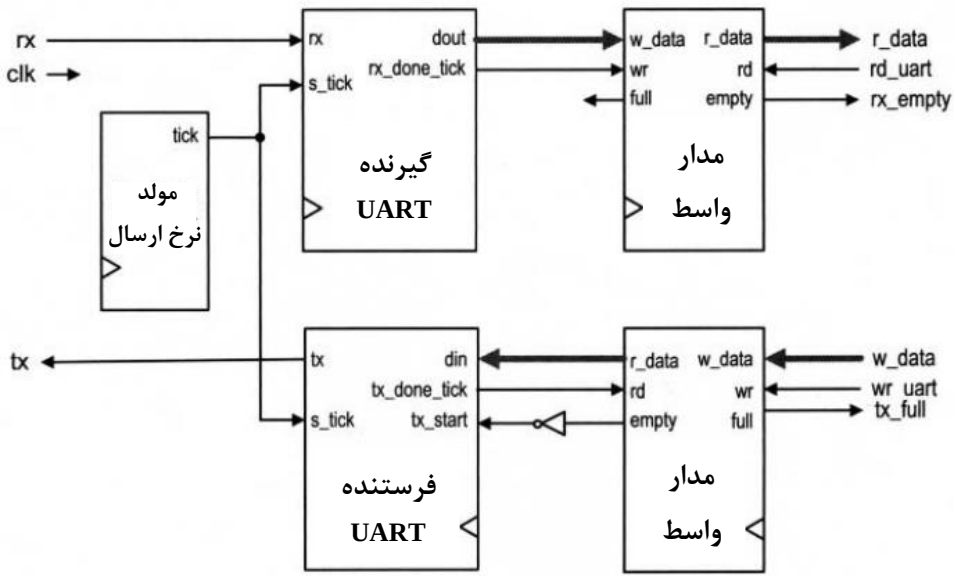
برای ایجاد یک ارتباط سریال صحیح، ابتدا باید پارامترهایی از قبیل تعداد بیت‌های داده، بیت شروع، بیت توقف، نرخ ارسال یا میزان باودریت[5] و بیت توازن را بین فرستنده و گیرنده تعیین کرد. در اینجا فرستنده و گیرنده می‌تواند پردازنده FPGA و یا نرم‌افزار متلب باشد. در این پروژه از آنجا که پیکسل‌های تصویر هشت بیتی هستند، ارتباط سریال ما از هشت بیت داده، یک بیت شروع، یک بیت توقف، نرخ ارسال ۹۲۱۶۰۰ بیت بر ثانیه و بدون هیچ بیت توازنی استفاده شده است. میزان نرخ ارسال به نحوی تعیین شده است که هیچگونه خطایی در ارسال و دریافت اطلاعات تصاویر رخ ندهد. در

[1] Start Bit
[2] Data
[3] Parity
[4] Stop Bit
[5] Baudrate

جدول (۵-۱) نام و کاربرد کانکتورهای ۹ پایه و ۲۵ پایه برای ایجاد یک ارتباط سریال قرار گرفته‌است. در شکل (۵-۷) شماتیک یک ارتباط سریال نیز ترسیم شده است.

جدول (۵-۱): نام و کاربرد پایه‌های پورت سریال در نوع ۹ پایه و ۲۵ پایه

| نام پایه | کاربرد | شماره پایه (نوع ۹ پایه) | شماره پایه (نوع ۲۵ پایه) |
|---|---|---|---|
| Transmit Data (TD) | خروجی اطلاعات سریال | پایه ۳ | پایه ۹ |
| Receive Data (RD) | ورودی اطلاعات سریال | پایه ۲ | پایه ۳ |
| Signal Ground | اتصال زمین | پایه ۵ | پایه ۷ |




شکل (۵-۷): بلوک دیاگرام کلی یک ارتباط سریال

مدار مولد نرخ ارسال در حقیقت شمارنده‌ای هست که سیگنال نمونه‌گیری را تولید می‌کند. بطور مثال برای تولید نرخ ارسال ۹۲۱۶۰۰ بیت بر ثانیه در پالس ساعت ۲۴ مگاهرتز، میزان نمونه‌گیری باید ۲۶ (نسبت پالس ساعت به نرخ ارسال) باشد. گیرنده و فرستنده ارتباط سریال مداری است که کلمه داده را از طریق نمونه‌برداری به دست می‌آورد. مدار واسط نیز بافر و وضعیت بین گیرنده-فرستنده ارتباط سریال و سیستمی که از ارتباط سریال استفاده می‌کند را تشکیل می‌دهد [۲۶].

### ۵-۴-۳ تراشه‌ی Xilinx Spartan-6

پس از اینکه اطلاعات تصویر نویزی توسط نرم‌افزار متلب از طریق ارتباط سریال برای پردازنده FPGA ارسال شد، این پردازنده اطلاعات را به صورت بایت به بایت در حافظه‌ی خارجی SRAM ذخیره می‌نماید. در مرحله بعد طبق روال الگوریتم پیشنهادی پیکسل‌ها از حافظه جانبی فراخوانی می‌شوند و عملیات رفع نویز روی آن‌ها اعمال می‌گردد و مجدداً در حافظه‌ی جانبی ذخیره می‌شوند. تراشه استفاده شده در این پروژه Xilinx Spartan-6 سری XC6SLX9 می‌باشد. این تراشه از خانواده Spartan و از شرکت معروف Xilinx است که نسبت به نسل قبلی خود (Spartan-3) دارای برتری‌های زیادی دارد که می‌توان موارد زیر را بیان کرد:

- ❖ مصرف توان پایین‌تر (حدود ۵۰٪) به لطف تکنولوژی ساخت ۴۵ نانومتر.
- ❖ افزایش دو برابری ظرفیت لاجیک FPGA بواسطه‌ی جایگزینیLUTهای چهار ورودی نسل قبلی با LUTهای شش ورودی در Spartan-6.
- ❖ افزایش ظرفیت بلوک‌هایDSP.
- ❖ پشتیبانی از استاندارد ارتباطی TMDS[1] با امکان برقراری ارتباط ویدئویی HDMI[2] و DVI[3] بدون نیاز به هیچ‌گونه سخت‌افزار اضافی.
- ❖ بهبود منابع کلاک با استفاده از واحدهای CMT[4] که علاوه بر دو DCM[5] حاوی یک واحد PLL[6] نیز می‌باشند (برخلاف Spartan-3 که تنها دارای واحد DCM است).
- ❖ دارای ۱۶ عدد DSP48A1 Slices، این Slicesها شامل ضرب کننده-جمع کننده هستند.

در جدول (۵-۲) مشخصات بعضی از تراشه‌های خانواده Spartan-6 با جزئیات بیشتری ذکر شده است.

[1] Transition-Minimized Defferential Signaling (TMDS)
[2] High-Definition Multimedia Interface (HDMI)
[3] Digital Visual Interface (DVI)
[4] Clock Management Tile (CMT)
[5] Digital Clock Manager (DCM)
[6] Phase Locked Loop (PLL)

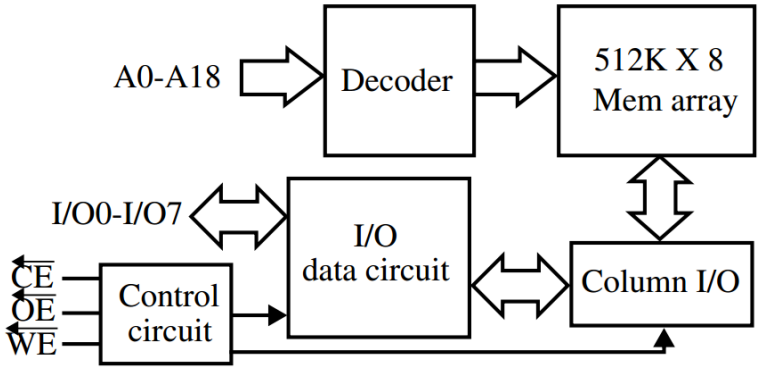
جدول (۵-۲): مشخصات برخی از تراشه‌های خانواده Xilinx Spartan-6

| Part Namber | XC6SLX4 | XC6SLX9 | XC6SLX16 | XC6SLX25 |
|---|---|---|---|---|
| Slice | 600 | 1430 | 2278 | 3758 |
| Logic Cell | 3840 | 9152 | 14579 | 24051 |
| CLB-Flip Flop | 4800 | 11440 | 18224 | 30064 |
| Total Block RAM | 216 kbits | 576 kbits | 576 kbits | 936 kbits |
| Digital Clock Manager | 2 | 2 | 2 | 2 |
| Available User Pins | 120 | 200 | 232 | 266 |
| DSP48A1 Slices | 8 | 16 | 32 | 38 |
| Configuration Memory | 2.7 Mbits | 2.7 Mbits | 2.7 Mbits | 4.4 Mbits |

### ۵-۴-۴ حافظه جانبی SRAM

حافظه با دستیابی تصادفی (RAM) برای ذخیره حجیم در یک سیستم دیجیتال به کار می‌رود. یک سلول RAM خیلی کوچکتر از یک سلول فلیپ فلاپ است، نوع رایج RAM، نوع استاتیک غیر همزمان یا آسنکرون(SRAM) می‌باشد. هر چند در یک قیمت مساوی SRAM در مقایسه با نوع حافظه‌ی DRAM[1] از ظرفیت کمتری برخوردار است، اما مزیت مهم SRAMها سادگی استفاده از آن‌هاست. بر خلاف یک ثبات که در آن داده در یک سیگنال ساعت نمونه‌گیری و ذخیره می‌شود، دسترسی به یک داده از یک SRAM همزمان بسیار پیچیده‌تر است. عمل خواندن و نوشتن، لازم می‌دارد که سیگنال‌های داده، آدرس و کنترل با ترتیب خاصی تولید شوند و این سیگنال‌ها باید برای مدت معینی از عمل پایدار باشند. ما معمولاً یک کنترلگر حافظه را به عنوان واسط به کار می‌بریم که فرمان‌ها را بطور پیوسته از سیستم اصلی دریافت کرده و سپس سیگنال‌های زمانی مناسب تولید می‌نماید تا به SRAM دست یابد. کنترلگر سیستم اصلی را از زمان‌بندی پیچیده دور نگه می‌دارد. حافظه‌ای که ما در این پروژه استفاده می‌کنیم، 512K×8 بیت حافظه‌ی SRAM است که توسط ISIS[2] ساخته شده است. نمودار بلوکی ساده حافظه جانبی SRAM در شکل (۵-۸) نمایش داده شده است.

[1] Dinamic Random Access Memory (DRAM)
[2] Integrated Silicon Solution Inc (ISIS)




شکل (۵-۸): نمودار بلوکی حافظه خارجی SRAM

در شکل بالا A0-A18 نمایانگر خط آدرس ۱۹ بیتی حافظه، I/O0-I/O7 نشان دهنده خط داده هشت بیتی، فعال‌ساز تراشه (CE)[1]، فعال‌ساز خروجی حافظه (OE)[2] و فعال‌ساز عمل نوشتن (WE)[3] در حافظه جانبی می‌باشد. لازم به ذکر است که خطوط کنترل حافظه همگی فعال پایین هستند. مشخصه زمان‌بندی یک SRAM غیر همزمان بسیار پیچیده و بیش از دو جین پارامتر دارد. ما فقط بر چند پارامتر کلیدی که مرتبط با طراحی ما هستند، توجه می‌کنیم [۲۶]. نمودارهای زمان‌بندی ساده برای عمل خواندن در شکل (۵-۹) نشان داده شده است. پارامترهای زمان‌بندی مرتبط عبارت‌اند از:

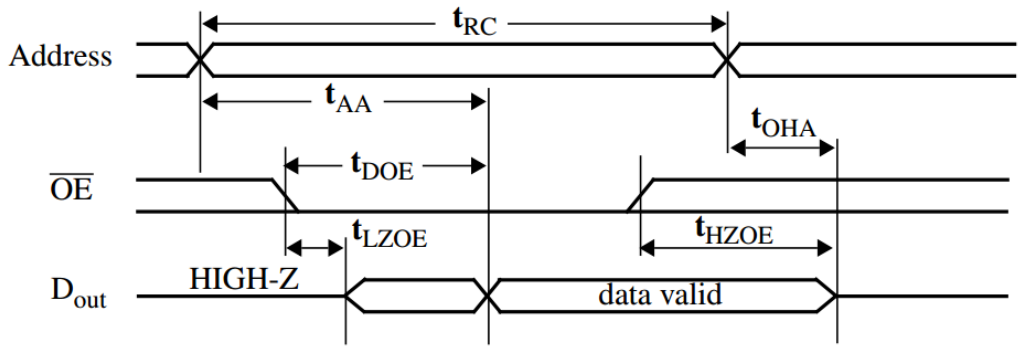
- $T_{RC}$: زمان سیکل خواندن، حداقل زمان سپری شده بین دو عمل خواندن است و این زمان تقریباً دو برابر زمان $T_{AA}$ برای SRAM برابر است.
- $T_{AA}$: زمان دستیابی آدرس، زمانی است برای به دست آوردن یک داده خروجی پایدار که پس از یک تغییر آدرس لازم است.
- $T_{OHA}$: زمان نگهداری خروجی، زمانی که داده خروجی پس از تغییر آدرس معتبر می‌ماند.
- $T_{DOE}$: زمان دستیابی فعال‌ساز خروجی، زمان لازم برای به دست آوردن داده معتبر پس از فعال شدن خروجی حافظه.

[1] Chip Enable (CE)
[2] Output Enable (OE)
[3] Write Enable (WE)

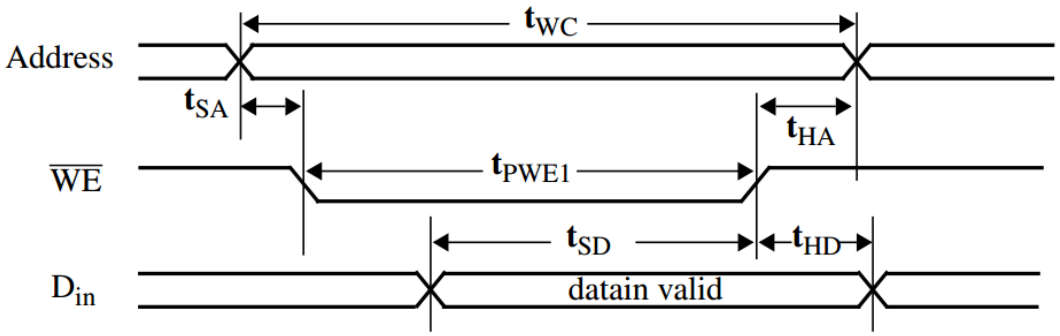
- $T_{HZOE}$: فعال‌ساز خروجی تا زمان امپدانس بالا، زمان لازم برای بافر سه حالته جهت ورود به حالت امپدانس بالا پس از غیر فعال شدن OE است.
- $T_{LZOE}$: فعال‌ساز خروجی تا زمان امپدانس پایین، زمان لازم برای بافر سه حالته تا حالت امپدانس بالا را ترک کند، داده هنوز نامعتبر است.

نمودار زمان‌بندی ساده برای یک عمل نوشتن در شکل (۵-۱۰) نشان داده شده است. پارامترهای مربوطه عبارت‌اند از:

- $T_{WC}$: زمان سیکل نوشتن، حداقل زمان سپری شده بین دو نوشتن است.
- $T_{SA}$: زمان برپایی آدرس، حداقل زمانی است که آدرس باید قبل از فعال شدن WE پایدار باشد.
- $T_{HA}$: زمان نگهداری آدرس، حداقل زمانی که آدرس قبل از غیر فعال شدن WE باید پایدار بماند.
- $T_{PWE1}$: عرض پالس WE، حداقل زمانی که WE باید فعال بماند.
- $T_{SD}$: زمان برپایی داده، حداقل زمانی است که داده باید قبل از لبه قفل کردن پایدار بماند (لبه‌ای که در آن WE از مقدار صفر به یک می‌رود).
- $T_{HD}$: زمان نگهداری داده، حداقل زمانی است که داده پس از لبه لچ کردن باید پایدار باشد.




شکل (۵-۹): نمودار زمان‌بندی یک سیکل خواندن در حافظه جانبی SRAM با OE کنترل شده




شکل (۵-۱۰): نمودار زمان‌بندی یک سیکل نوشتن در حافظه جانبی SRAM با WE کنترل شده

در جدول (۵-۳) مقدار زمان‌بندی‌های لازم برای پارامترهای ذکر شده در بالا، ثبت شده است.

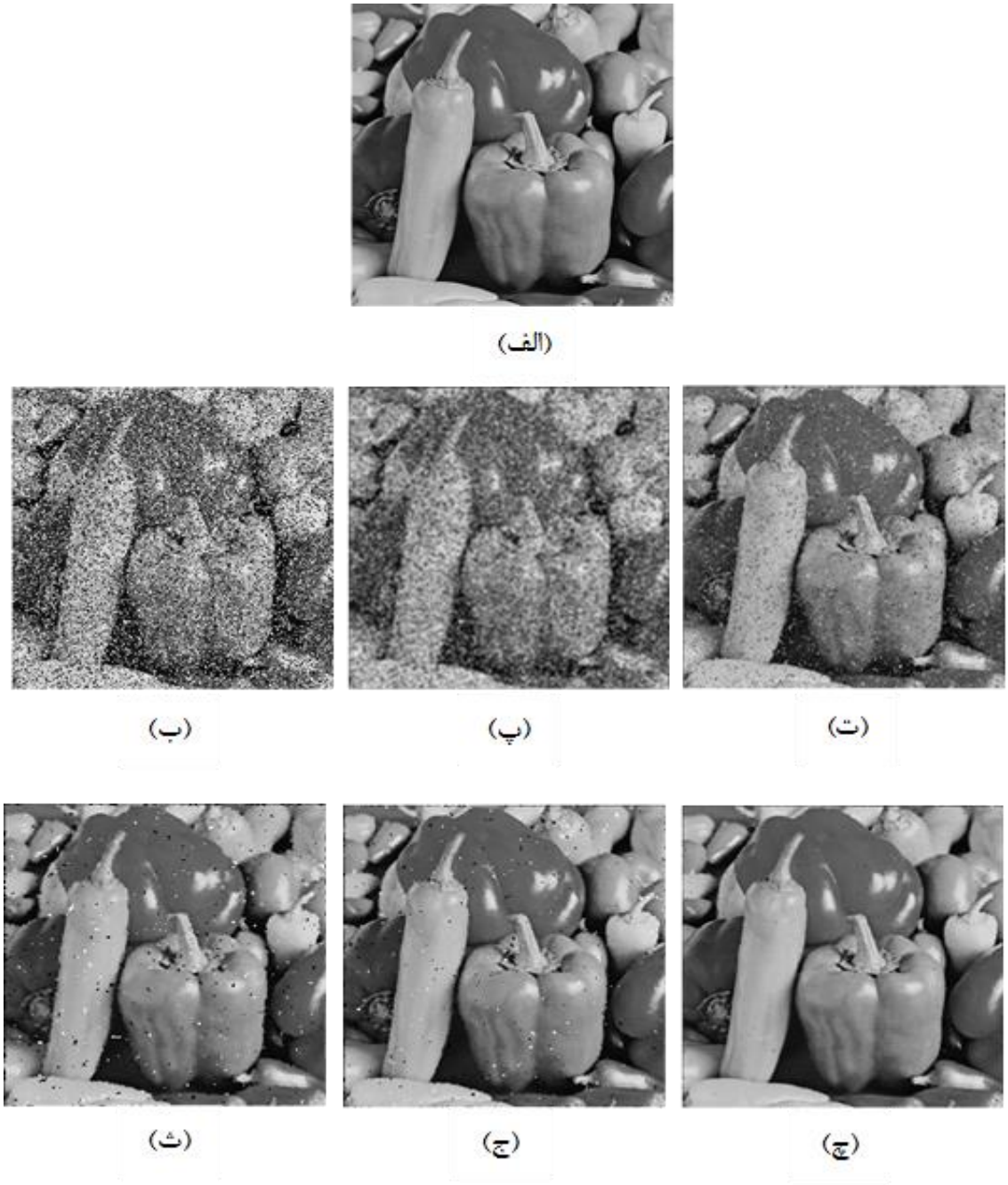
جدول (۵-۳): پارامترهای زمان‌بندی درحافظه جانبی SRAM بر حسب نانوثانیه

| Parameter | Min(nsec) | Max(nsec) |
|---|---|---|
| Read Cycle Time ($T_{RC}$) | 10 | - |
| Address Access Time ($T_{AA}$) | - | 10 |
| Output Hold Time ($T_{OHA}$) | 2 | - |
| Output Enable Access Time ($T_{DOE}$) | - | 4 |
| Output Enable to High-Z Time ($T_{HZOE}$) | - | 4 |
| Output Enable to Low-Z Time ($T_{LZOE}$) | 0 | - |
| Write Cycle Time ($T_{WC}$) | 10 | - |
| Address Setup Time ($T_{SA}$) | 0 | - |
| Address Hold Time ($T_{HA}$) | 0 | - |
| W_en Pulse Width ($T_{PWE1}$) | 8 | - |
| Data Setup Time ($T_{SD}$) | 6 | - |
| Data Hold Time ($T_{HD}$) | 0 | - |

## ۵-۵ نتایج پیاده‌سازی سخت‌افزاری

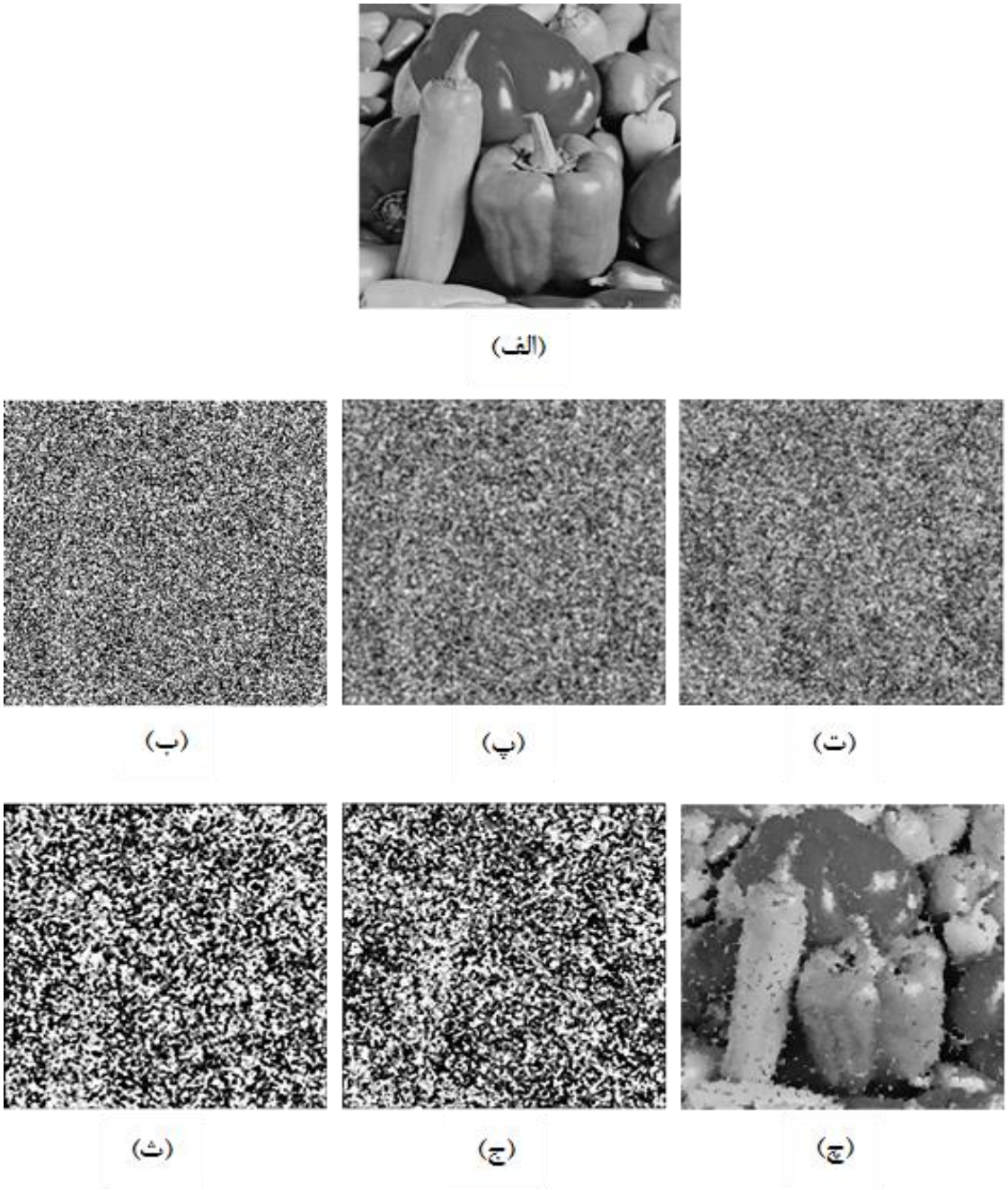
در این بخش نتایج پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی رفع نویز ضربه‌ای فلفل و نمک گنجانده شده است. لازم به ذکر است، علاوه بر روش پیشنهادی، فیلترهای میانگین استاندارد، میانگین تطبیقی، میانه استاندارد و میانه تطبیقی نیز به صورت سخت‌افزاری پیاده‌سازی شده است تا در پایان مقایسه‌ای بین این فیلترها صورت گیرد. در این قسمت الگوریتم پیشنهادی و سایر فیلترهای پایه در رفع نویز ضربه‌ای فلفل و نمک به صورت سخت‌افزاری بر روی تصویر خاکستری و استاندارد peppers با اندازه ابعاد ۲۵۶×۲۵۶ اعمال شده و سپس نتایج با هم مقایسه می‌گردد. نتایج بصری این

پیاده‌سازی در چگالی‌های نویز اعمالی ۳۰ درصد، ۶۰ درصد و ۹۰ درصد به ترتیب در شکل‌های شماره (۵-۱۱)، (۵-۱۲) و (۵-۱۳) ارائه شده است.




شکل (۵-۱۱): الف) تصویر اصلی peppers، ب) تصویر (الف) با چگالی نویز اعمالی ۳۰ درصد، پ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین استاندارد، ت) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین تطبیقی، ث) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه استاندارد، ج) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه تطبیقی، چ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی


شکل (۵-۱۲): الف) تصویر اصلی peppers، ب) تصویر (الف) با چگالی نویز اعمالی ۶۰ درصد، پ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین استاندارد، ت) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین تطبیقی، ث) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه استاندارد، ج) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه تطبیقی، چ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی




شکل (۵-۱۳): الف) تصویر اصلی peppers، ب) تصویر (الف) با چگالی نویز اعمالی ۹۰ درصد، پ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین استاندارد، ت) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین تطبیقی، ث) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه استاندارد، ج) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه تطبیقی، چ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی

برای مقایسه عددی نتایج الگوریتم از شاخص‌های ارزیابی PSNR، MSE، MAE و SSIM استفاده شده است. مقادیر این شاخص‌های ارزیابی برای تصویر خاکستری peppers با اندازه ابعاد ۲۵۶×۲۵۶ به ترتیب در جدول‌های (۵-۴)، (۵-۵)، (۵-۶) و (۵-۷) ثبت شده است.

جدول (۵-۴): مقادیر شاخص ارزیابی PSNR برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 23.20 | 30.08 | 31.10 | 37.15 | 41.57 |
| 20 | 20.17 | 25.43 | 26.87 | 30.64 | 36.95 |
| 30 | 18.32 | 22.29 | 22.53 | 24.97 | 34.06 |
| 40 | 16.87 | 20.08 | 18.52 | 20.51 | 31.91 |
| 50 | 15.55 | 17.88 | 15.01 | 16.65 | 29.97 |
| 60 | 14.58 | 16.23 | 12.19 | 13.39 | 28.63 |
| 70 | 13.62 | 14.90 | 9.73 | 10.88 | 27.12 |
| 80 | 12.83 | 13.50 | 8.08 | 8.67 | 25.13 |
| **90** | **12.09** | **12.43** | **6.59** | **6.87** | **20.83** |

جدول (۵-۵): مقادیر شاخص ارزیابی MSE برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 310.62 | 63.99 | 50.38 | 11.58 | 4.52 |
| 20 | 624.62 | 186.24 | 133.65 | 56.01 | 13.09 |
| 30 | 955.98 | 383.77 | 362.31 | 206.6 | 25.49 |
| 40 | 1336.5 | 637.37 | 912.58 | 577.21 | 41.87 |
| 50 | 1810.2 | 1058.5 | 2048.54 | 1404.15 | 65.45 |
| 60 | 2262.59 | 1546.94 | 3926.43 | 2975.6 | 89.13 |
| 70 | 2821.26 | 2100.74 | 6908.5 | 5297.9 | 125.91 |
| 80 | 3385.1 | 2901.22 | 10111 | 8820.88 | 199.46 |
| **90** | **4017.17** | **3714.29** | **14236** | **13360** | **536.43** |

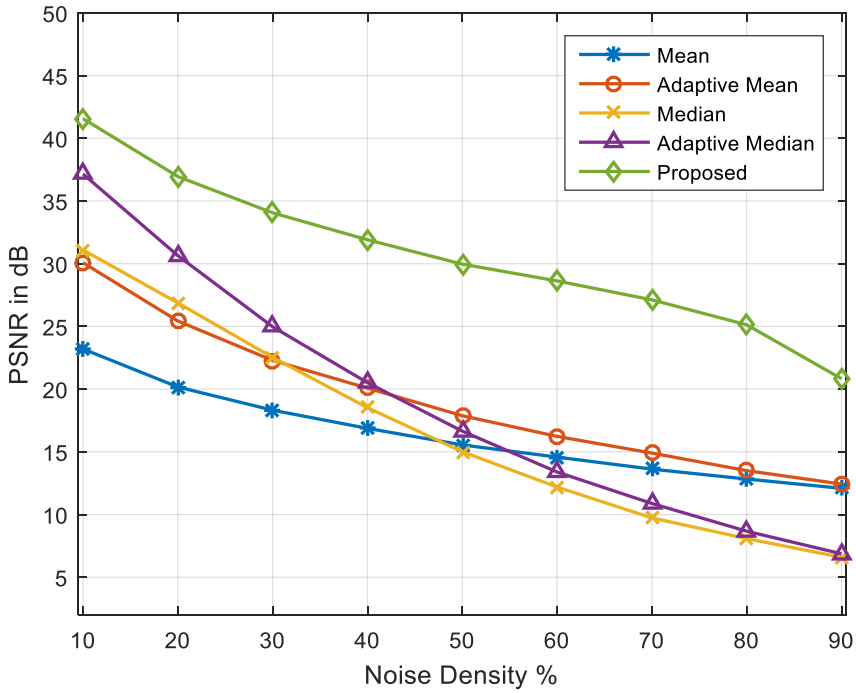
جدول (۵-۶): مقادیر شاخص ارزیابی MAE برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 12.29 | 2.04 | 2.89 | 0.476 | 0.32 |
| 20 | 18.83 | 4.75 | 4.01 | 1.28 | 0.75 |
| 30 | 23.84 | 8.46 | 6.05 | 2.86 | 1.28 |
| 40 | 28.35 | 12.59 | 10.15 | 5.88 | 1.92 |
| 50 | 33.44 | 18.31 | 18.04 | 11.84 | 2.67 |
| 60 | 37.67 | 24.28 | 30.38 | 22.57 | 3.50 |
| 70 | 42.3 | 30.59 | 49.51 | 38.33 | 4.56 |
| 80 | 46.59 | 38.70 | 70.54 | 61.09 | 6.16 |
| **90** | **51.16** | **46.56** | **97.14** | **90.44** | **11.17** |

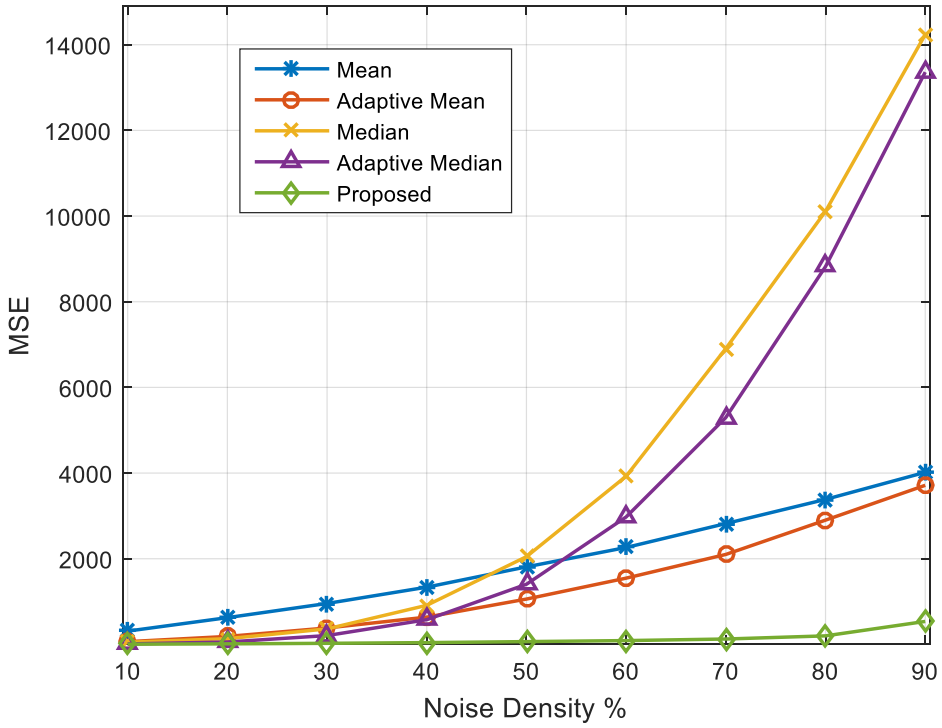
جدول (۵-۷): مقادیر شاخص ارزیابی SSIM برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 0.409 | 0.692 | 0.854 | 0.981 | 0.987 |
| 20 | 0.290 | 0.529 | 0.782 | 0.926 | 0.969 |
| 30 | 0.224 | 0.402 | 0.633 | 0.783 | 0.942 |
| 40 | 0.180 | 0.323 | 0.413 | 0.557 | 0.914 |
| 50 | 0.140 | 0.243 | 0.237 | 0.324 | 0.879 |
| 60 | 0.110 | 0.182 | 0.131 | 0.174 | 0.842 |
| 70 | 0.075 | 0.138 | 0.066 | 0.100 | 0.795 |
| 80 | 0.047 | 0.089 | 0.033 | 0.050 | 0.704 |
| **90** | **0.025** | **0.043** | **0.018** | **0.021** | **0.437** |

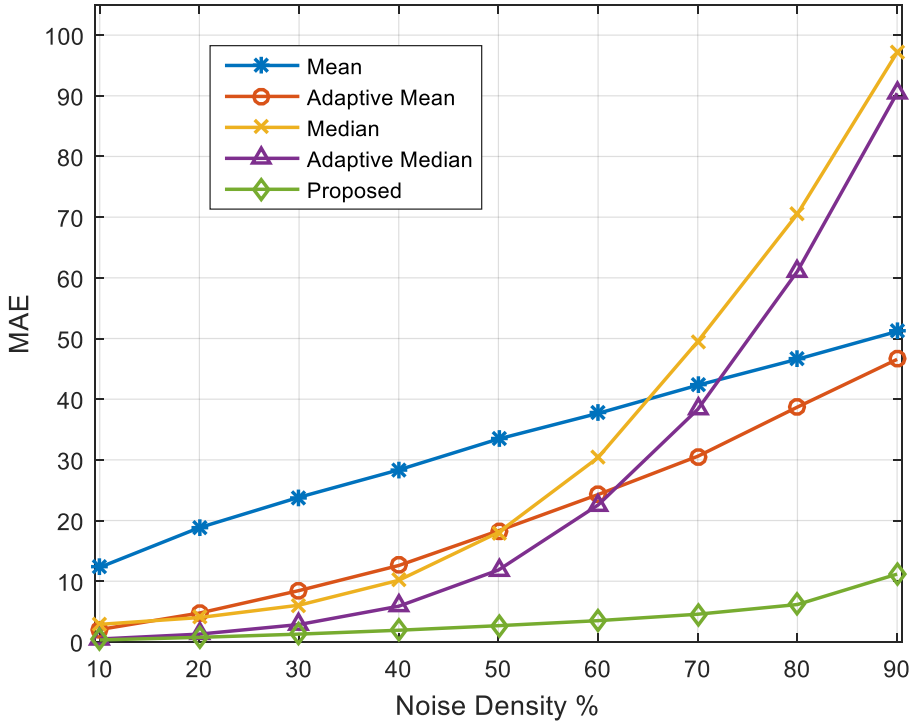
از مقایسه بصری و عددی نتایج حاصل شده در بالا می‌توان دریافت که الگوریتم پیشنهادی عملکرد خوبی در حذف نویز ضربه‌ای فلفل و نمک را دارد. نمودار مقادیر شاخص‌های ارزیابی PSNR، MSE، MAE و SSIM برای تصویر خاکستری peppers با اندازه ابعاد ۲۵۶×۲۵۶ نیز به ترتیب در شکل‌های (۵-۱۴)، (۵-۱۵)، (۵-۱۶) و (۵-۱۷) نشان داده شده است.



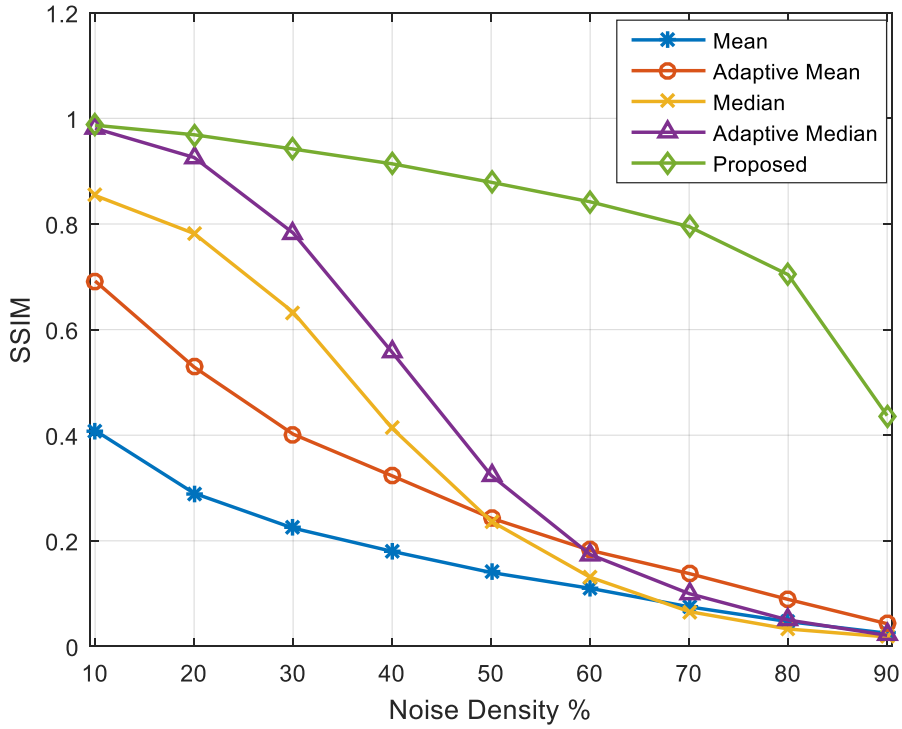

شکل (۵-۱۴): نمودار شاخص ارزیابی PSNR برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف




شکل (۵-۱۵): نمودار شاخص ارزیابی MSE برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

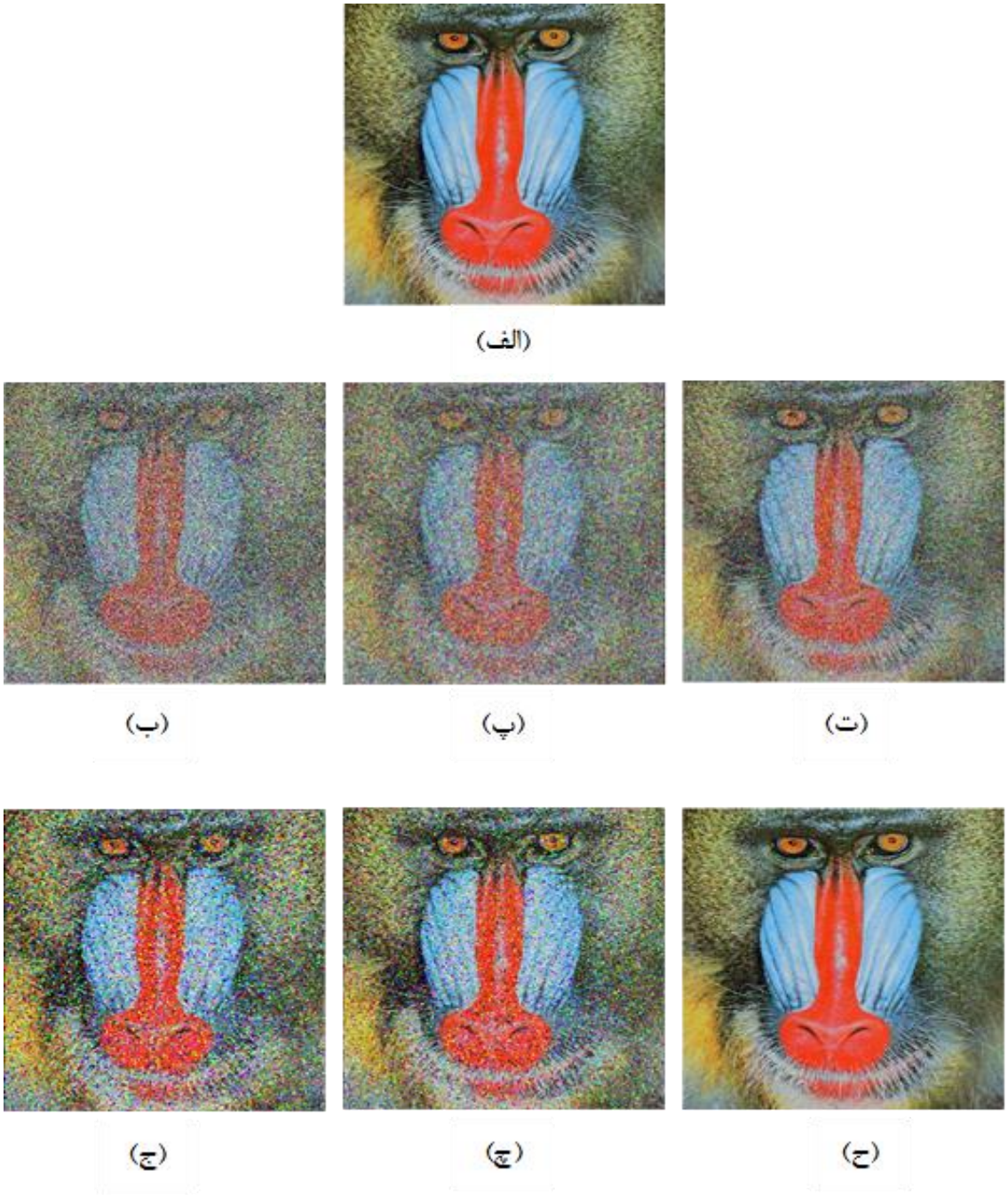



شکل (۵-۱۶): نمودار شاخص ارزیابی MAE برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

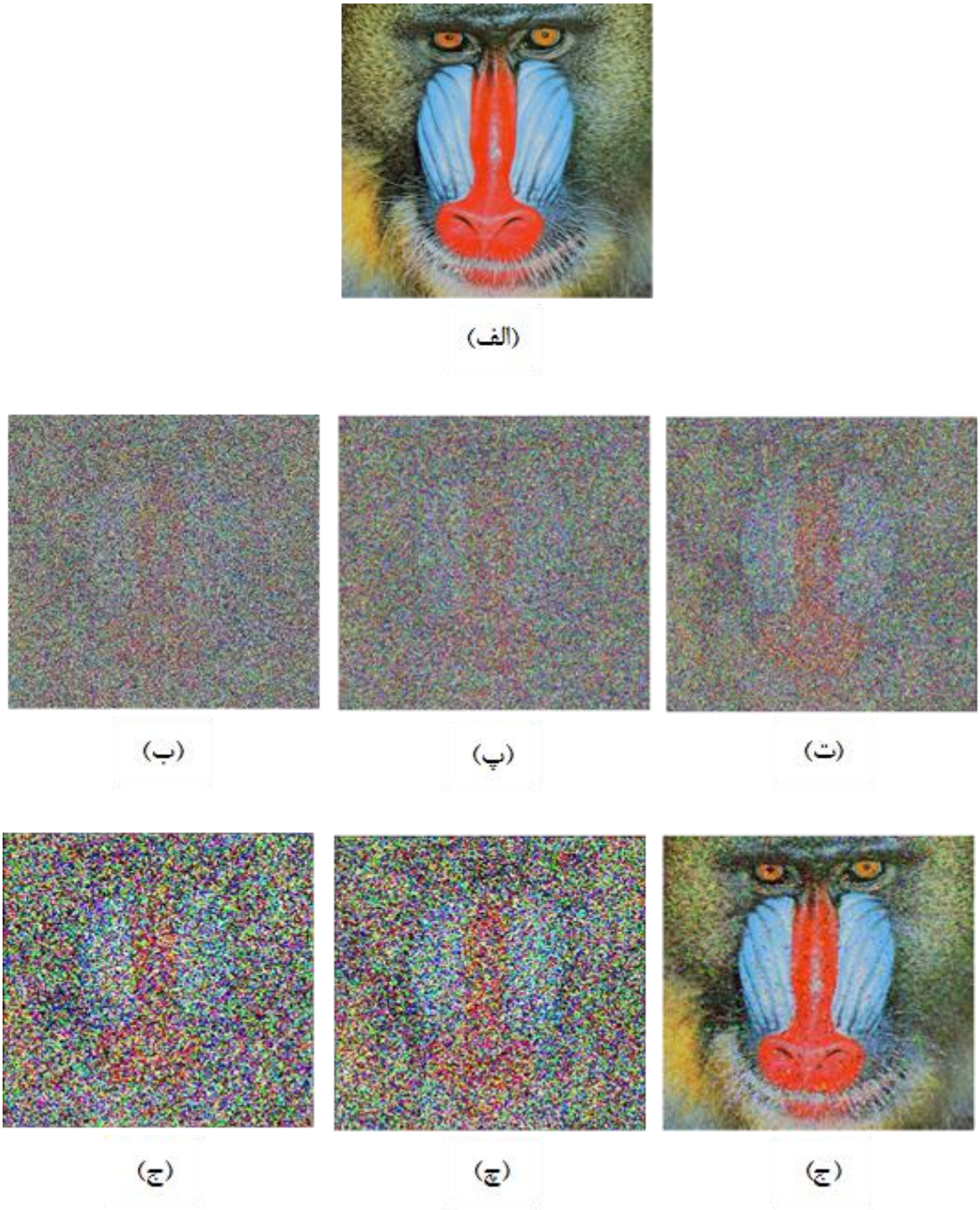



شکل (۵-۱۷): نمودار مقادیر شاخص ارزیابی SSIM برای تصویر خاکستری peppers در چگالی‌های نویزی مختلف

طبق نمودارهای ترسیم شده در بالا متوجه می‌شویم که در چگالی‌های نویزی مختلف روش پیشنهادی نسبت به چهار فیلتر مکانی پایه، دارای PSNR بالا، خطای MSE و MAE کم و همچنین میزان شاخص SSIM بالایی است. هر چقدر میزان شاخص SSIM به یک نزدیکتر باشد، به این معناست که تصویر بازیابی شده به تصویر مرجع نزدیکتر است. در ادامه الگوریتم پیشنهادی و سایر فیلترهای پایه در رفع نویز ضربه‌ای فلفل و نمک به صورت سخت‌افزاری بر روی تصویر رنگی و استاندارد baboon با اندازه ابعاد ۲۵۶×۲۵۶ اعمال شده و سپس نتایج با هم مقایسه می‌گردد. نتایج بصری این پیاده‌سازی در چگالی نویز اعمالی ۶۰ درصد و ۹۰ درصد به ترتیب در شکل‌های (۵-۱۸) و (۵-۱۹) ارائه شده است.




شکل (۵-۱۸): الف) تصویر رنگی و اصلی baboon، ب) تصویر (الف) با چگالی نویز اعمالی ۶۰ درصد، پ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین استاندارد، ت) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین تطبیقی، ث) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه استاندارد، ج) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه تطبیقی، چ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی




شکل (۵-۱۹): الف) تصویر رنگی و اصلی baboon، ب) تصویر (الف) با چگالی نویز اعمالی ۹۰ درصد، پ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین استاندارد، ت) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانگین تطبیقی، ث) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه استاندارد، ج) نتیجه حاصل از پیاده‌سازی سخت‌افزاری فیلتر میانه تطبیقی، چ) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی

در ادامه مقادیر شاخص‌های ارزیابی PSNR، MSE، MAE و SSIM برای تصویر رنگی baboon با اندازه ابعاد ۲۵۶×۲۵۶ به ترتیب در جدول‌های (۵-۸)، (۵-۹)، (۵-۱۰) و (۵-۱۱) ثبت شده است.

جدول (۵-۸): مقادیر شاخص ارزیابی PSNR برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 21.32 | 29.20 | 24.37 | 32.44 | 34.32 |
| 20 | 19.24 | 24.75 | 22.93 | 27.99 | 30.93 |
| 30 | 17.72 | 21.72 | 20.68 | 23.60 | 28.59 |
| 40 | 16.41 | 19.63 | 17.56 | 19.52 | 26.92 |
| 50 | 15.30 | 17.76 | 14.53 | 16.08 | 25.59 |
| 60 | 14.41 | 16.24 | 11.83 | 13.23 | 24.31 |
| 70 | 13.58 | 14.77 | 9.75 | 10.77 | 23.10 |
| 80 | 12.79 | 13.55 | 7.98 | 8.61 | 21.70 |
| **90** | **12.04** | **12.41** | **6.53** | **6.68** | **19.11** |

جدول (۵-۹): مقادیر شاخص ارزیابی MSE برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 479.27 | 78 | 237.29 | 37 | 24.03 |
| 20 | 774.02 | 217.43 | 331.29 | 103.3 | 52.43 |
| 30 | 1097.69 | 437.03 | 555.24 | 283.28 | 89.95 |
| 40 | 1485.83 | 707.75 | 1128.40 | 725.82 | 132.12 |
| 50 | 1916.68 | 1088.53 | 2287.31 | 1602.72 | 179.37 |
| 60 | 2345.54 | 1544.61 | 4263.07 | 3085.38 | 240.57 |
| 70 | 2845.54 | 2167.64 | 6875.09 | 5532.54 | 317.77 |
| 80 | 3417.37 | 2869.31 | 10339.43 | 8950.94 | 439.23 |
| **90** | **4062.17** | **3725.05** | **14446** | **13394.03** | **796.35** |

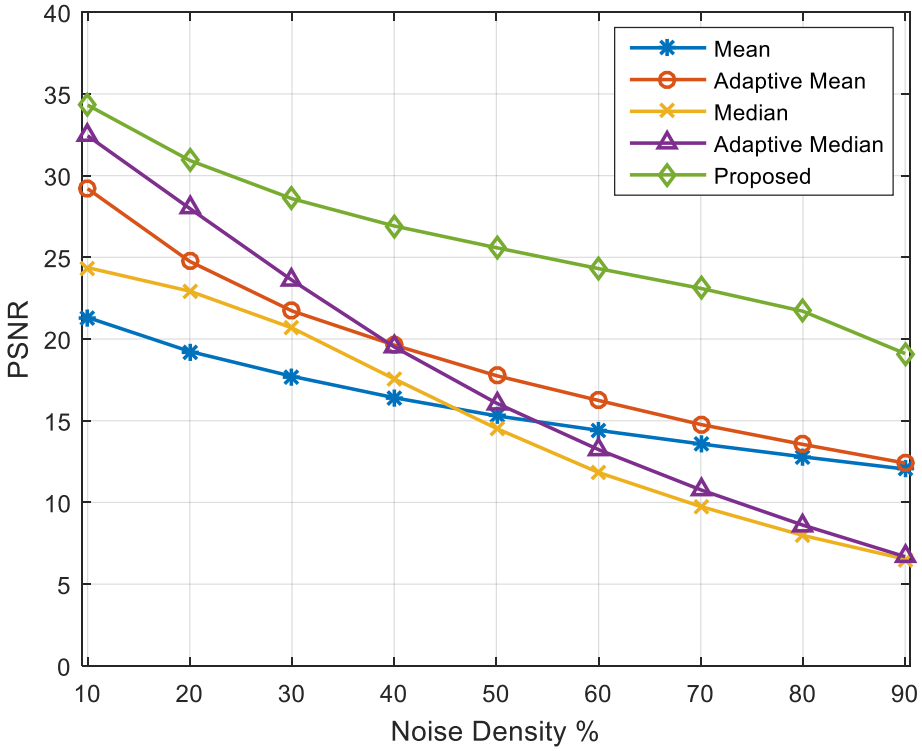
جدول (۵-۱۰): مقادیر شاخص ارزیابی MAE برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 16.41 | 2.20 | 9.54 | 1.28 | 1.06 |
| 20 | 21.48 | 5.24 | 10.66 | 2.84 | 2.22 |
| 30 | 25.92 | 9.06 | 12.67 | 5.15 | 3.52 |
| 40 | 30.40 | 13.36 | 16.92 | 9.16 | 4.92 |
| 50 | 34.75 | 18.60 | 24.61 | 15.91 | 6.44 |
| 60 | 38.67 | 24.41 | 37.11 | 26.25 | 8.20 |
| 70 | 42.78 | 31.30 | 53.32 | 42.39 | 10.19 |
| 80 | 47.04 | 38.56 | 74.52 | 64.34 | 12.80 |
| **90** | **51.09** | **46.88** | **99.56** | **92.26** | **18.17** |

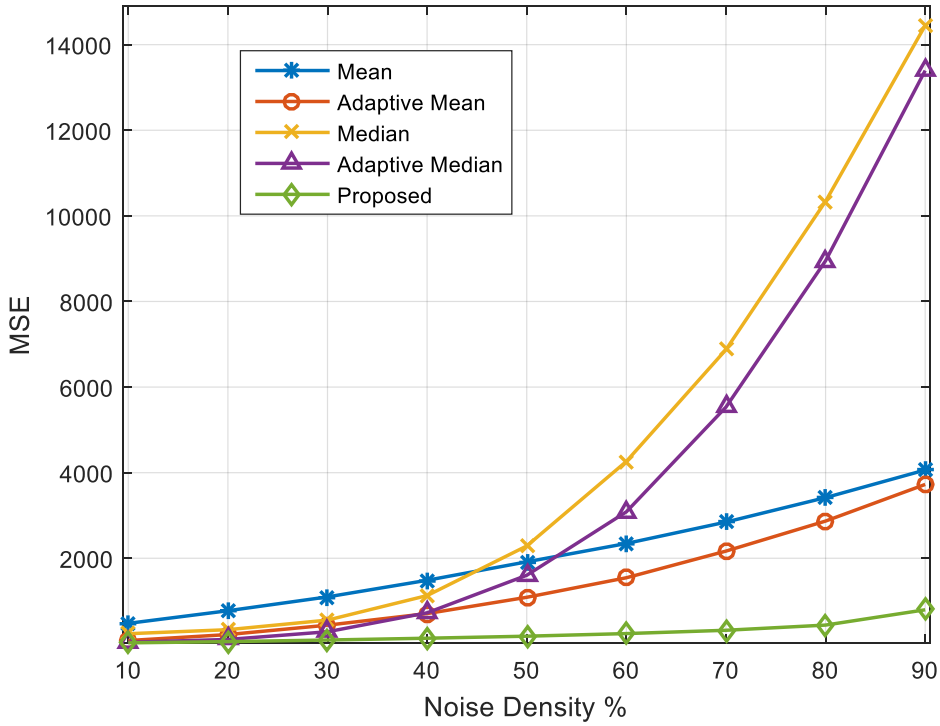
جدول (۵-۱۱): مقادیر شاخص ارزیابی SSIM برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

| Noise Density% | Standard Mean Filter | Adaptive Mean Filter | Standard Median filter | Adaptive Median Filter | This work |
|---|---|---|---|---|---|
| 10 | 0.664 | 0.944 | 0.807 | 0.976 | 0.984 |
| 20 | 0.561 | 0.864 | 0.769 | 0.936 | 0.965 |
| 30 | 0.475 | 0.767 | 0.693 | 0.854 | 0.941 |
| 40 | 0.397 | 0.665 | 0.555 | 0.707 | 0.914 |
| 50 | 0.316 | 0.559 | 0.407 | 0.531 | 0.883 |
| 60 | 0.248 | 0.453 | 0.267 | 0.364 | 0.844 |
| 70 | 0.181 | 0.342 | 0.167 | 0.225 | 0.798 |
| 80 | 0.124 | 0.233 | 0.095 | 0.127 | 0.732 |
| **90** | **0.058** | **0.111** | **0.039** | **0.055** | **0.585** |

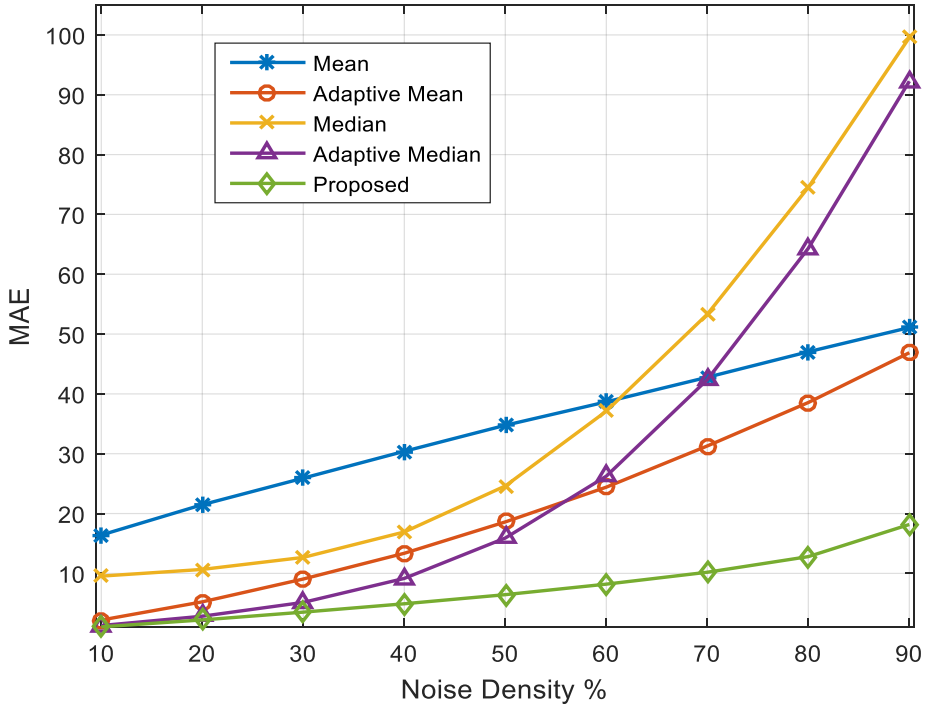
از مقایسه بصری و عددی نتایج حاصل شده در بالا می‌توان دریافت که الگوریتم پیشنهادی عملکرد خوبی در حذف نویز ضربه‌ای فلفل و نمک در تصاویر رنگی دارد. نمودار شاخص‌های ارزیابی PSNR، MSE، MAE و SSIM برای تصویر رنگی استاندارد baboon با اندازه ابعاد ۲۵۶×۲۵۶ نیز به ترتیب در شکل‌های (۵-۲۰)، (۵-۲۱)، (۵-۲۲) و (۵-۲۳) نشان داده شده است.



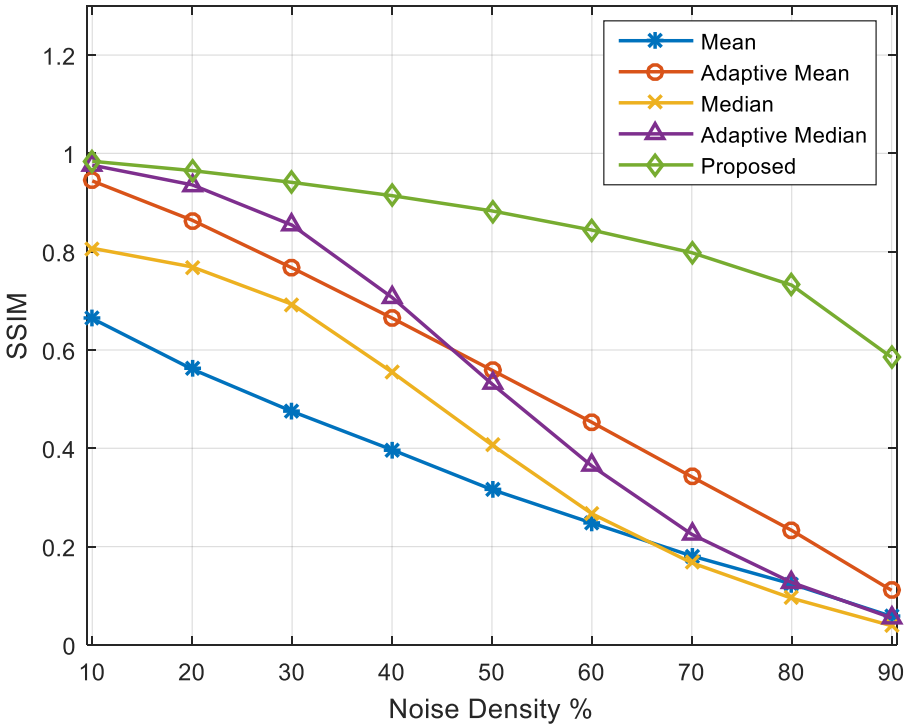

شکل (۵-۲۰): نمودار مقادیر شاخص ارزیابی PSNR برای تصویر رنگی baboon در چگالی‌های نویزی مختلف




شکل (۵-۲۱): نمودار شاخص ارزیابی MSE برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

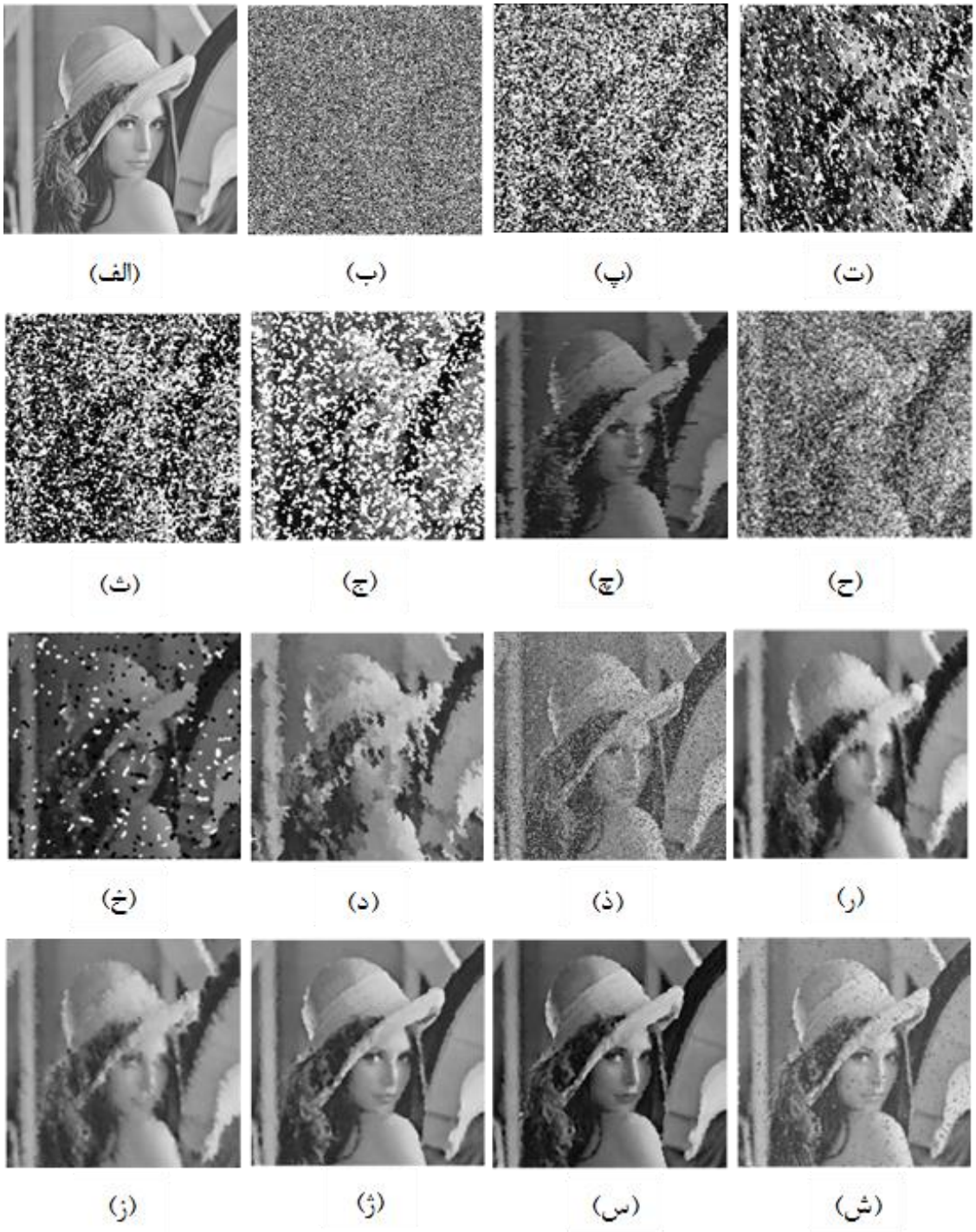



شکل (۵-۲۲): نمودار شاخص ارزیابی MAE برای تصویر رنگی baboon در چگالی‌های نویزی مختلف




شکل (۵-۲۳): نمودار مقادیر شاخص ارزیابی SSIM برای تصویر رنگی baboon در چگالی‌های نویزی مختلف

بر اساس نمودارهای ترسیم شده در بالا متوجه می‌شویم که در چگالی‌های نویزی مختلف، روش پیشنهادی نسبت به چهار فیلتر مکانی پایه برای رفع نویز در یک تصویر رنگی، دارای PSNR بالا، خطای MSE و MAE کم و همچنین میزان شاخص SSIM بالایی است. در پایان نیز الگوریتم پیشنهادی به صورت سخت‌افزاری بر روی تصویر خاکستری Lena با اندازه ابعاد ۵۰۰×۵۰۰ اعمال شده و سپس نتایج با کارهای ارائه شده در گذشته، مقایسه می‌گردد. نتایج بصری پیاده‌سازی و شبیه‌سازی‌ها در چگالی نویز اعمالی ۹۰ درصد در شکل (۵-۲۴) نشان داده شده است. میزان شاخص‌های ارزیابی PSNR و MSE برای تصویر Lena به ترتیب در جدول‌های (۵-۱۲) و (۵-۱۳) ثبت گردیده است. نمودار این شاخص‌های ارزیابی نیز به ترتیب در شکل‌های شماره (۵-۲۵) و (۵-۲۶) نشان داده شده است.




شکل (۵-۲۴): الف) تصویر اصلی Lena، ب) تصویر الف با چگالی نویز ۹۰ درصد، پ) نتیجه حاصل از AMF، ت) نتیجه حاصل از ACWMF، ث) نتیجه حاصل از NASM، ج) نتیجه حاصل از Trilatral، چ) نتیجه حاصل از DBA، ح) نتیجه حاصل از NIDF، خ) نتیجه حاصل از MDBA، د) نتیجه حاصل از ROAD، ذ) نتیجه حاصل از MDBUTMF، ر) نتیجه حاصل از MNLF، ز) نتیجه حاصل از ITMMF، ژ) نتیجه حاصل از ADBMF، س) نتیجه حاصل از NMF، ش) نتیجه حاصل از پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی

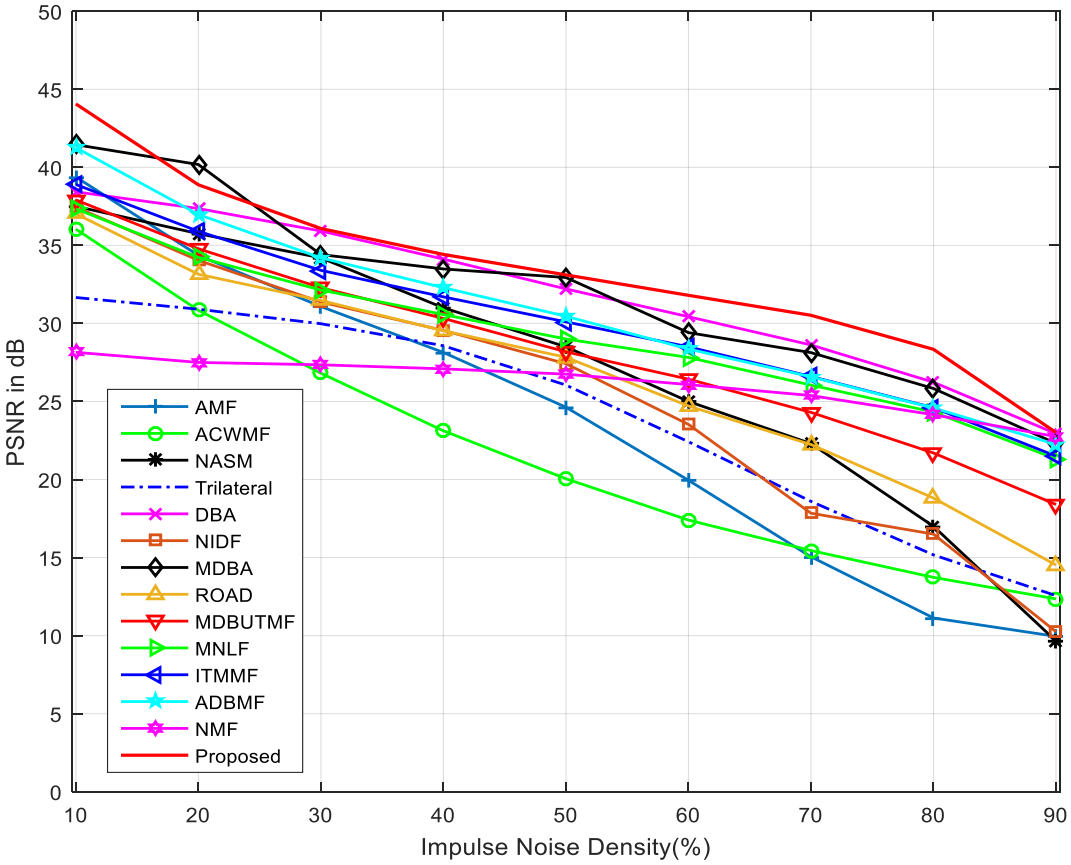
جدول (۵-۱۲): مقادیر شاخص ارزیابی PSNR الگوریتم‌ها برای تصویر خاکستری Lena در چگالی‌های نویزی مختلف

| ALG \ ND% | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|---|---|
| **AMF** | 39.38 | 34.36 | 31.08 | 28.14 | 24.63 | 19.95 | 15.05 | 11.13 | **9.95** |
| **ACWMF** | 36.04 | 30.84 | 26.81 | 23.12 | 20.05 | 17.40 | 15.45 | 13.73 | **12.35** |
| **NASM** | 37.50 | 35.75 | 34.20 | 31 | 28.50 | 25 | 22.30 | 17 | **9.65** |
| **Trilatral** | 31.66 | 30.91 | 29.98 | 28.57 | 26.05 | 22.42 | 18.60 | 15.17 | **12.56** |
| **DBA** | 38.43 | 37.36 | 35.92 | 34.12 | 32.21 | 30.43 | 28.62 | 26.23 | **22.94** |
| **NIDF** | 37.45 | 34.04 | 31.37 | 29.55 | 27.42 | 23.53 | 17.84 | 16.50 | **10.25** |
| **MDBA** | 41.45 | 40.18 | 34.41 | 33.49 | 32.94 | 29.41 | 28.13 | 25.83 | **22.31** |
| **ROAD** | 37.02 | 33.15 | 31.47 | 29.53 | 27.83 | 24.71 | 22.24 | 18.81 | **14.52** |
| **MDBUTMF** | 37.91 | 34.78 | 32.29 | 30.32 | 28.18 | 26.43 | 24.30 | 21.70 | **18.40** |
| **MNLF** | 37.34 | 34.23 | 32.14 | 30.57 | 29.00 | 27.82 | 26.05 | 24.32 | **21.32** |
| **ITMMF** | 38.90 | 35.90 | 33.40 | 31.70 | 30.10 | 28.50 | 26.60 | 24.60 | **21.50** |
| **ADBMF** | 41.26 | 36.98 | 34.20 | 32.29 | 30.46 | 28.36 | 26.58 | 24.58 | **22.25** |
| **NMF** | 28.15 | 27.50 | 27.35 | 27.09 | 26.75 | 26.08 | 25.38 | 24.15 | **22.75** |
| **Proposed** | 44.06 | 38.88 | 36.08 | 34.42 | 33.11 | 31.80 | 30.52 | 28.34 | **23.01** |

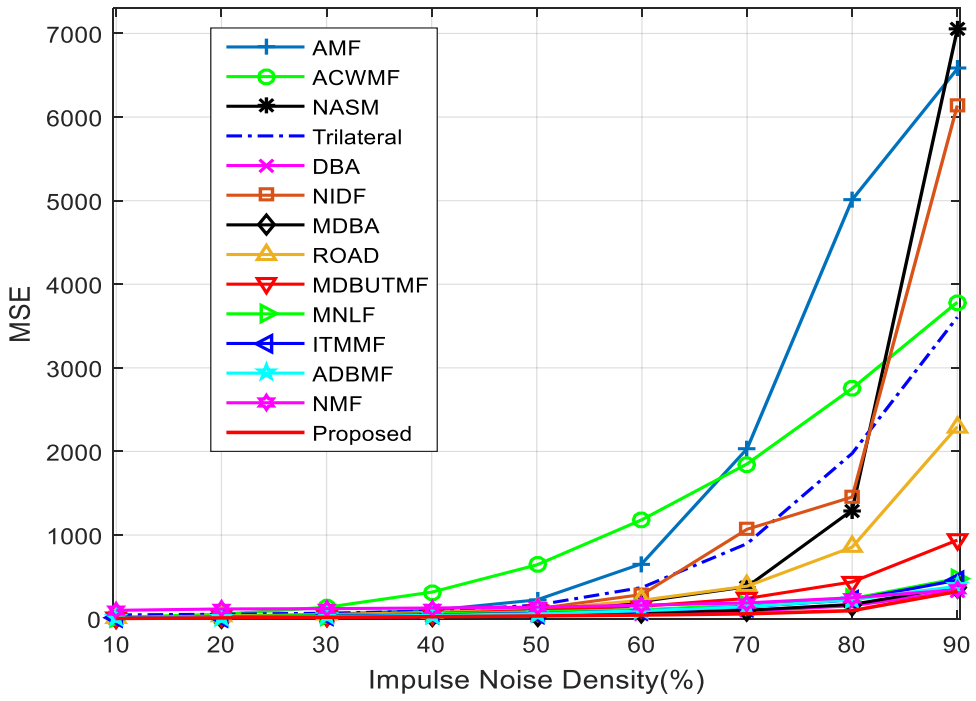
جدول (۵-۱۳): مقادیر شاخص ارزیابی MSE الگوریتم‌ها برای تصویر خاکستری Lena در چگالی‌های نویزی مختلف

| ALG \ ND% | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|---|---|
| **AMF** | 7.5 | 23.8 | 50.7 | 99.7 | 223.9 | 657.7 | 2032 | 5012 | **6577** |
| **ACWMF** | 16.1 | 53.5 | 135.5 | 317.0 | 642.8 | 1183.2 | 1851 | 2754 | **3785** |
| **NASM** | 11.5 | 17.3 | 24.7 | 51.6 | 91.8 | 205.6 | 382 | 1297 | **7048** |
| **Trilatral** | 44.3 | 52.7 | 65.3 | 90.3 | 161.4 | 372.4 | 897 | 1977 | **3606** |
| **DBA** | 9.3 | 11.9 | 16.6 | 25.1 | 39 | 58.8 | 89.3 | 154 | **330** |
| **NIDF** | 11.6 | 25.6 | 47.4 | 72.1 | 117.7 | 288.4 | 1069 | 1455 | **6138** |
| **MDBA** | 4.65 | 6.23 | 23.55 | 29.11 | 33.04 | 74.48 | 100.01 | 169.8 | **382** |
| **ROAD** | 12.9 | 31.4 | 46.3 | 72.4 | 107.1 | 219.8 | 388.2 | 855 | **2296** |
| **MDBUTMF** | 10.5 | 21.6 | 38.3 | 60.4 | 98.8 | 147.9 | 241.5 | 439 | **939** |
| **MNLF** | 11.9 | 24.5 | 39.7 | 57 | 81.8 | 107.4 | 161.4 | 240 | **479** |
| **ITMMF** | 8.3 | 16.7 | 29.7 | 43.9 | 63.5 | 91.8 | 142.2 | 225 | **460** |
| **ADBMF** | 4.86 | 13.03 | 24.72 | 38.37 | 58.48 | 94.58 | 142.91 | 226 | **387** |
| **NMF** | 99.5 | 115.6 | 119.6 | 127.1 | 137.4 | 160.3 | 188.3 | 250 | **345** |
| **Proposed** | 2.5 | 8.4 | 16.03 | 23.5 | 31.7 | 42.9 | 57.6 | 95.2 | **325** |

از مقایسه بصری و عددی نتایج حاصل شده در بالا می‌توان دریافت که الگوریتم پیشنهادی عملکرد خوبی در حذف نویز ضربه‌ای فلفل و نمک در تصاویر خاکستری با حجم بالا را دارد و در چگالی‌های نویزی مختلف نسبت به سایر الگوریتم‌های شبیه‌سازی شده در گذشته، دارای PSNR بالا و همچنین خطای MSE کمتری می‌باشد. این موضوع در نمودارهای صفحه بعد به خوبی قابل مشاهده هست.

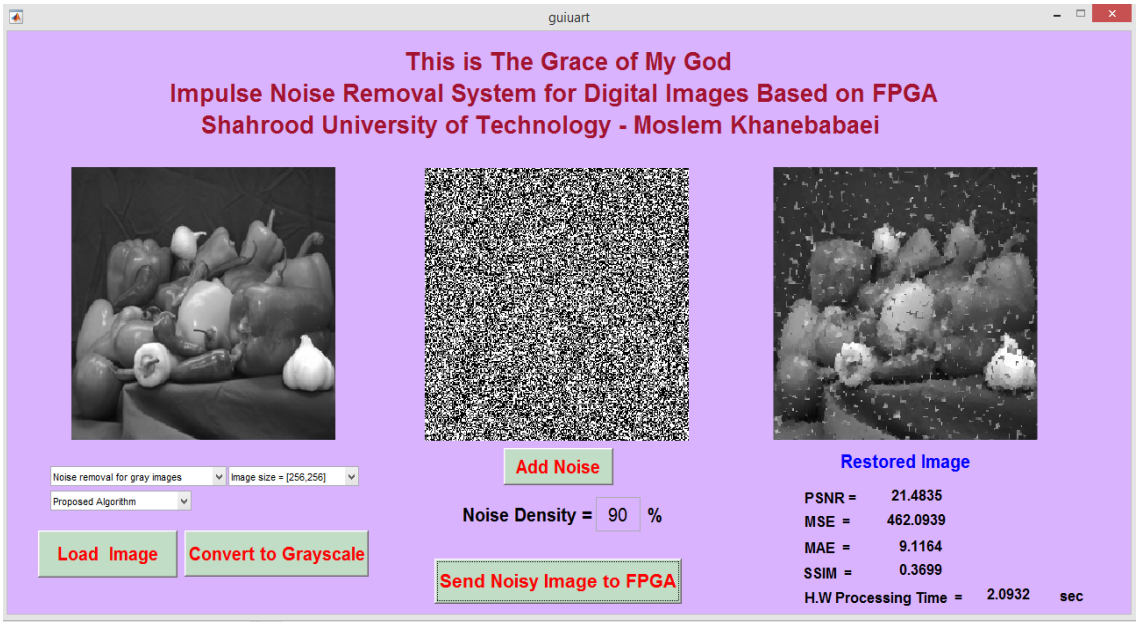



شکل (۵-۲۵): نمودار شاخص ارزیابی PSNR الگوریتم‌ها برای تصویر خاکستری Lena در چگالی‌های نویزی مختلف




شکل (۵-۲۶): نمودار شاخص ارزیابی MSE الگوریتم‌ها برای تصویر خاکستری Lena در چگالی‌های نویزی مختلف

در شکل (۵-۲۷) نمونه‌ای از عملیات رفع نویز ضربه‌ای را توسط الگوریتم ارائه شده در محیط گرافیکی GUI[1] نرم‌افزار متلب مشاهده می‌کنید.




شکل (۵-۲۷): نمایی از اجرا و نتایج الگوریتم پیشنهادی در محیط گرافیکی GUI نرم‌افزار متلب

از مقایسه بصری و عددی نتایج حاصل شده در بالا می‌توان دریافت که الگوریتم پیشنهادی عملکرد خوبی در حذف نویز از تصاویر خاکستری و رنگی با اندازه ابعاد بالا را دارد و میزان درخشندگی و لبه‌های تصویر را بخوبی حفظ و بازیابی می‌کند. در ادامه زمان اجرای عملیات رفع نویز ضربه‌ای فلفل و نمک در تصویر خاکستری Cameraman و تصویر رنگی peppers با اندازه ابعاد ۲۵۶×۲۵۶ با استفاده از الگوریتم پیشنهادی و الگوریتم‌های پایه در محیط شبیه‌سازی نرم‌افزار متلب و پیاده‌سازی سخت‌افزاری مقایسه می‌گردد، نتایج مقایسه زمان پردازش در جدول‌های زیر نمایش داده شده است. لازم به ذکر است که زمان پردازش سخت‌افزار از دو بخش میزان زمان پردازش توسط تراشه FPGA و میزان زمان تبادل اطلاعات تصویر توسط ارتباط سریال تشکیل گردیده است.

[1] Graphical User Interfaces (GUI)

جدول (۵-۱۴): میزان زمان پردازش نرم‌افزاری و سخت‌افزاری در فیلتر میانگین استاندارد

| Filter | **Standard Mean Filter** | | | | | |
|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | |
| Time | Hardware Time processing (in second) | | Software Time processing (in second) | Hardware Time processing (in second) | | Software Time processing (in second) |
| ND% | Serial Interface | FPGA | MATLAB | Serial Interface | FPGA | MATLAB |
| 30 | 1.429 | 154m | 4.929 | 4.267 | 467m | 8.291 |
| 60 | 1.428 | 153m | 4.903 | 4.266 | 466m | 8.369 |
| 90 | 1.428 | 154m | 4.908 | 4.266 | 467m | 8.370 |

جدول (۵-۱۵): میزان زمان پردازش نرم‌افزاری و سخت‌افزاری در فیلتر میانگین تطبیقی

| Filter | **Adaptive Mean Filter** | | | | | |
|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | |
| Time | Hardware Time processing (in second) | | Software Time processing (in second) | Hardware Time processing (in second) | | Software Time processing (in second) |
| ND% | Serial Interface | FPGA | MATLAB | Serial Interface | FPGA | MATLAB |
| 30 | 1.429 | 141m | 3.114 | 4.267 | 413m | 6.050 |
| 60 | 1.428 | 144m | 4.172 | 4.266 | 432m | 7.126 |
| 90 | 1.428 | 149m | 4.807 | 4.266 | 457m | 7.734 |

جدول (۵-۱۶): میزان زمان پردازش نرم‌افزاری و سخت‌افزاری در فیلتر میانه استاندارد

| Filter | **Standard Median Filter** | | | | | |
|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | |
| Time | Hardware Time processing (in second) | | Software Time processing (in second) | Hardware Time processing (in second) | | Software Time processing (in second) |
| ND% | Serial Interface | FPGA | MATLAB | Serial Interface | FPGA | MATLAB |
| 30 | 1.429 | 163m | 6.868 | 4.267 | 498m | 10.323 |
| 60 | 1.428 | 164m | 6.922 | 4.266 | 499m | 10.300 |
| 90 | 1.428 | 164m | 6.920 | 4.266 | 499m | 10.322 |

جدول (۵-۱۷): میزان زمان پردازش نرم‌افزاری و سخت‌افزاری در فیلتر میانه تطبیقی

| Filter | **Adaptive Median Filter** | | | | | |
|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | |
| Time | Hardware Time processing (in second) | | Software Time processing (in second) | Hardware Time processing (in second) | | Software Time processing (in second) |
| ND% | Serial Interface | FPGA | MATLAB | Serial Interface | FPGA | MATLAB |
| 30 | 1.429 | 150m | 3.814 | 4.267 | 463m | 7.297 |
| 60 | 1.428 | 155m | 4.508 | 4.266 | 489m | 8.516 |
| 90 | 1.428 | 159m | 5.409 | 4.266 | 492m | 9.806 |

جدول (۵-۱۸): میزان زمان پردازش نرم‌افزاری و سخت‌افزاری در الگوریتم پیشنهادی

| Filter | **Proposed Filter** | | | | | |
|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | |
| Time | Hardware Time processing (in second) | | Software Time processing (in second) | Hardware Time processing (in second) | | Software Time processing (in second) |
| ND% | Serial Interface | FPGA | MATLAB | Serial Interface | FPGA | MATLAB |
| 30 | 1.429 | 450m | 4.704 | 4.267 | 1.295 | 8.720 |
| 60 | 1.428 | 466m | 6.141 | 4.266 | 1.297 | 12.501 |
| 90 | 1.428 | 493m | 7.720 | 4.266 | 1.299 | 19.925 |

همانطور که در جدول‌ها نشان داده شده است، زمان اجرای الگوریتم پیشنهادی در پیاده‌سازی سخت‌افزاری، کمتر از زمان اجرا در محیط شبیه‌سازی با متلب است. پس بنابراین انتخاب پردازنده FPGA به عنوان بستر پیاده‌سازی برای الگوریتم‌های پردازش تصویر، امری صحیح و قابل قبول می‌باشد. لازم به ذکر است که اکثر زمان پردازش در پیاده‌سازی سخت‌افزاری برای ارسال و دریافت اطلاعات تصویر بین نرم افزار متلب و بورد از طریق ارتباط سریال صرف می‌شود. یکی دیگر از نکات مهم در پیاده‌سازی سخت‌افزاری الگوریتم‌ها، استفاده بهینه از منابع داخلی FPGA است. در این پروژه سعی بر این بوده که الگوریتم پیشنهادی تا حد امکان به صورت پردازش موازی انجام شود تا از منابع داخلی کمتری استفاده گردد. در جدول‌های زیر میزان استفاده از منابع داخلی تراشه برای اجرای روش پیشنهادی و سایر الگوریتم‌های پایه در رفع نویز یک تصویری خاکستری با اندازه ابعاد ۲۵۶×۲۵۶ ثبت شده است. با توجه به جدول (۵-۲۳) میزان استفاده پروژه از منابع داخلی قابل قبول و بهینه می‌باشد.

جدول (۵-۱۹): میزان استفاده از منابع داخلی تراشه در فیلتر میانگین استاندارد

| Filter | **Standard Mean Filter** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | | Gray Scale 500×500 | | |
| Logic | Used | Available | Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Registers | 597 | 11440 | 5% | 599 | 11440 | 5% | 607 | 11440 | 5% |
| Slice LUTs | 661 | 5720 | 11% | 744 | 5720 | 13% | 657 | 5720 | 11% |
| IOBs | 33 | 102 | 32% | 33 | 102 | 32% | 33 | 102 | 32% |
| BUFG | 1 | 16 | 6% | 1 | 16 | 6% | 1 | 16 | 6% |
| DSP48A1 | 2 | 16 | 12% | 2 | 16 | 12% | 2 | 16 | 12% |

جدول (۵-۲۰): میزان استفاده از منابع داخلی تراشه در فیلتر میانگین تطبیقی

| Filter | **Adaptive Mean Filter** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | | Gray Scale 500×500 | | |
| Logic | Used | Available | Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Registers | 589 | 11440 | 5% | 596 | 11440 | 5% | 599 | 11440 | 5% |
| Slice LUTs | 653 | 5720 | 11% | 752 | 5720 | 13% | 650 | 5720 | 11% |
| IOBs | 33 | 102 | 32% | 33 | 102 | 32% | 33 | 102 | 32% |
| BUFG | 1 | 16 | 6% | 1 | 16 | 6% | 1 | 16 | 6% |
| DSP48A1 | 2 | 16 | 12% | 2 | 16 | 12% | 2 | 16 | 12% |

جدول (۵-۲۱): میزان استفاده از منابع داخلی تراشه در فیلتر میانه استاندارد

| Filter | **Standard Median Filter** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | | Gray Scale 500×500 | | |
| Logic | Used | Available | Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Registers | 689 | 11440 | 5% | 691 | 11440 | 6% | 699 | 11440 | 6% |
| Slice LUTs | 991 | 5720 | 11% | 1115 | 5720 | 19% | 992 | 5720 | 17% |
| IOBs | 33 | 102 | 32% | 33 | 102 | 32% | 33 | 102 | 32% |
| BUFG | 4 | 16 | 25% | 4 | 16 | 25% | 4 | 16 | 25% |
| DSP48A1 | 2 | 16 | 12% | 2 | 16 | 12% | 2 | 16 | 12% |

جدول (۵-۲۲): میزان استفاده از منابع داخلی تراشه در فیلتر میانه تطبیقی

| Filter | **Adaptive Median Filter** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | | Gray Scale 500×500 | | |
| Logic | Used | Available | Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Registers | 597 | 11440 | 5% | 632 | 11440 | 5% | 691 | 11440 | 6% |
| Slice LUTs | 661 | 5720 | 11% | 771 | 5720 | 13% | 999 | 5720 | 17% |
| IOBs | 33 | 102 | 32% | 33 | 102 | 32% | 33 | 102 | 32% |
| BUFG | 4 | 16 | 25% | 4 | 16 | 25% | 4 | 16 | 25% |
| DSP48A1 | 2 | 16 | 12% | 2 | 16 | 12% | 2 | 16 | 12% |

جدول (۵-۲۳): میزان استفاده از منابع داخلی تراشه در الگوریتم پیشنهادی

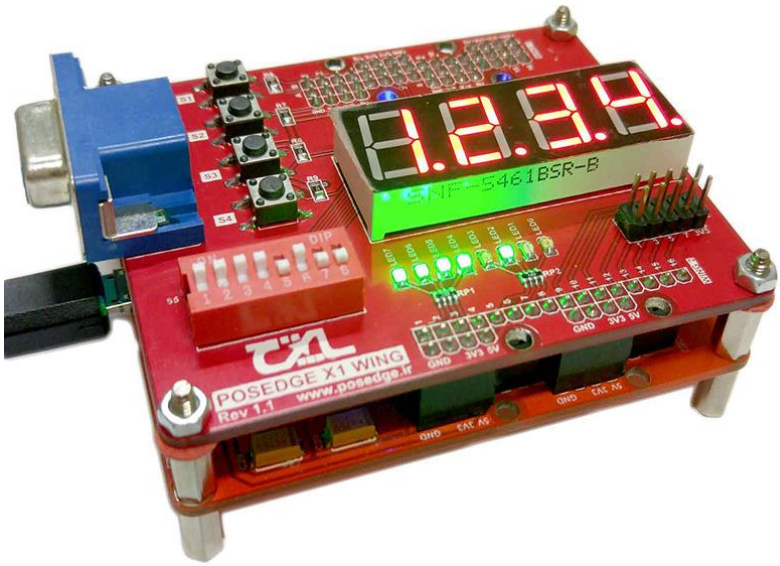
| Filter | **Proposed Filter** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Image and size | Gray Scale 256×256 | | | RGB Scale 256×256 | | | Gray Scale 500×500 | | |
| Logic | Used | Available | Utilization | Used | Available | Utilization | Used | Available | Utilization |
| Registers | 1887 | 11440 | 16% | 1859 | 11440 | 16% | 1909 | 11440 | 16% |
| Slice LUTs | 2115 | 5720 | 36% | 2125 | 5720 | 37% | 2118 | 5720 | 37% |
| IOBs | 33 | 102 | 32% | 33 | 102 | 32% | 33 | 102 | 32% |
| BUFG | 1 | 16 | 6% | 1 | 16 | 6% | 1 | 16 | 6% |
| DSP48A1 | 1 | 16 | 6% | 2 | 16 | 12% | 1 | 16 | 6% |

از دیگر پارامترهای مهم در پیاده‌سازی سخت‌افزاری می‌توان به میزان توان مصرفی، بیشترین فرکانس خروجی تراشه FPGA و دمای پیوند تراشه اشاره کرد. هر چقدر که توان مصرفی سخت‌افزار کمتر باشد (یا دمای نقطه کار و پیوند تراشه پایین‌تر باشد) و فرکانس خروجی تراشه بیشتر باشد، پیاده‌سازی مورد نظر بهینه و سرعت اجرای عملیات بالاتر است. در جدول (۵-۲۴) میزان توان مصرفی کل، بیشترین فرکانس خروجی پردازنده FPGA، و دمای پیوند تراشه در این پروژه و سایر الگوریتم‌های پایه در رفع نویز ضربه‌ای نشان داده شده است. لازم به ذکر است که میزان توان مصرفی کل و دمای پیوندهای تراشه با استفاده از نرم‌افزار Xilinx XPower Analyser از مجموعه نرم‌افزارهای موجود در ISE Design Suite 14.7 حاصل گردیده است. همچنین میزان بیشینه فرکانس تراشه از طریق گزارش طرح سنتز شده در نرم‌افزار ISE Design Suite 14.7 محاسبه گردیده است.

جدول (۵-۲۴): میزان توان مصرفی کل و مشخصات فرکانسی در پیاده‌سازی سخت‌افزاری الگوریتم‌ها

| **Parameter** | **Total power consumption (mw)** | | **Maximum output clock (MHz)** | | **Minimum clock period (nsec)** | | **Junction Temperature (C°)** | |
|---|---|---|---|---|---|---|---|---|
| **Image and Size** | Gray 256×256 | Gray 500×500 | Gray 256×256 | Gray 500×500 | Gray 256×256 | Gray 500×500 | Gray 256×256 | Gray 500×500 |
| **Standard Mean Filter** | 14 | 14 | 66.644 | 66.573 | 15.005 | 15.021 | 25.5 | 25.5 |
| **Adaptive Mean Filter** | 14 | 14 | 69.696 | 64.863 | 14.348 | 15.417 | 25.5 | 25.5 |
| **Standard Median Filter** | 16 | 16 | 69.686 | 68.761 | 14.350 | 14.543 | 25.6 | 25.6 |
| **Adaptive Median Filter** | 16 | 16 | 68.526 | 68.166 | 14.593 | 14.670 | 25.6 | 25.6 |
| **Proposed Filter** | 14 | 14 | 44.058 | 42.186 | 22.697 | 23.704 | 25.5 | 25.5 |

با توجه به نتایج جدول، سیستم پیاده‌سازی شده علاوه بر بهبود عملیات رفع نویز، عملکردی بهینه و سریع دارد. بورد توسعه FPGA استفاده شده برای پیاده‌سازی الگوریتم پیشنهادی در شکل (۵-۲۸) نمایش داده شده است. این بورد توسعه محصولی از شرکت توسعه و طراحی پازج می‌باشد. در قسمت پیوست، مشخصات این بورد معرفی می‌گردد [۲۹].





شکل (۵-۲۸): بورد توسعه FPGA استفاده شده برای پیاده‌سازی سخت‌افزاری الگوریتم پیشنهادی

# فصل ششم

## نتیجه‌گیری و پیشنهادات

## ۶-۱ جمع‌بندی و نتیجه‌گیری

در این پایان‌نامه هدف طراحی و پیاده‌سازی الگوریتمی با استفاده از فیلتر میانه-میانگین تطبیقی و اطلاعات همسایگی بر پایه FPGA برای رفع نویز ضربه‌ای فلفل و نمک بوده است. از آنجا قصد پیاده‌سازی سخت‌افزاری الگوریتم ارائه شده را نیز داریم و در یک سیستم سخت‌افزاری یکی از مهم‌ترین مشخصه‌ها، سرعت اجرای عملیات است، پس بنابراین روش پیشنهادی علاوه بر بازدهی بالا در رفع نویز، باید دارای حجم و محاسبات پیچیده‌ای نیز نباشد. هر چه سرعت اجرا در یک سیستم بالاتر باشد، مسلماً زمان پردازش نیز کمتر خواهد بود. بسیاری از الگوریتم‌های رفع نویز در مقالات اخیر با وجود بازدهی مطلوب، دارای یک محاسبات پیچیده و حجیم هستند و پیاده‌سازی سخت‌افزاری آن‌ها کاری دشوار است.

درفصل دوم به معرفی افزاره‌های منطقی و تراشه FPGA و همچنین روند تکاملی آن‌ها پرداخته شد. در ادامه با زبان توصیف سخت‌افزار VHDL و همچنین مراحل نوشتن، سنتز کردن و پیاده‌سازی یک برنامه کامل VHDL مورد اشاره قرار گرفت. در فصل پنجم به معرفی روش پیشنهادی پرداخته شد. با توجه به مقادیر شاخص‌های ارزیابی PSNR، MSE، MAE و SSIM در الگوریتم ارائه شده و سایر کارهای انجام شده در گذشته در تصاویر یکسان، متوجه می‌شویم که الگوریتم پیشنهادی عملکرد مطلوبی در حذف نویز ضربه (خصوصاً در چگالی‌های بالای نویزی) را دارد.

در ادامه میزان زمان اجرای عملیات روش پیشنهادی و سایر الگوریتم‌های پایه در محیط شبیه‌سازی متلب و پیاده‌سازی سخت‌افزاری مورد بحث قرار گرفت. نتایج نشان‌دهنده سرعت بالای سیستم پیشنهای در رفع نویز ضربه است، بطوری که زمان اجرای الگوریتم پیشنهادی برای تصویر Lena با اندازه ابعاد ۵۰۰×۵۰۰ و چگالی اعمالی ۹۰ درصد نویز در محیط شبیه‌سازی متلب و پیاده‌سازی سخت‌افزاری به ترتیب ۱۹/۹۲۵ ثانیه و ۶/۱۰۳ ثانیه است. از این میزان زمان پردازش سخت‌افزاری، ۱/۲۹۹ ثانیه مربوط به پردازش توسط تراشه FPGA و ۴/۸۰۴ ثانیه مربوط به تبادل اطلاعات تصویر از طریق ارتباط سریال است. با مقایسه زمان اجرا در محیط شبیه‌سازی متلب و

پیاده‌سازی سخت‌افزاری، متوجه می‌شویم که سیستم پیشنهادی دارای سرعت قابل توجهی است. البته علاوه بر کم بودن حجم و محاسبات در الگوریتم، پیاده‌سازی تا حد امکان بصورت پردازش موازی نیز تعیین کننده است. در مرحله بعد میزان استفاده از منابع داخلی تراشه FPGA در روش پیشنهادی و سایر الگوریتم‌های پایه بررسی شده است. با توجه به مقایسه بازدهی الگوریتم‌ها در رفع نویز ضربه، می‌توان گفت که روش ارائه شده استفاده بهینه‌ای از منابع داخلی تراشه را داراست. در پایان بیشینه فرکانس خروجی و میزان توان مصرفی کل برای تراشه FPGA محاسبه و ثبت گردید، بطوری که برای پردازش یک تصویر با اندازه ابعاد ۵۰۰×۵۰۰، بیشترین فرکانس خروجی و توان مصرفی کل تراشه به ترتیب ۴۲/۱۸۶ مگاهرتز و ۱۴ میلی‌وات به دست آمده است که نشان از سرعت بالا و بهینه بودن سیستم رفع نویز پیشنهادی است. پیاده‌سازی سیستم به روش‌های مختلف نرم‌افزاری و سخت‌افزاری نشان داد که سرعت پردازش سیستم‌های سخت‌افزاری شامل FPGA بسیار بیشتر از سیستم‌های مشابه نرم‌افزاری هستند. بنابراین در سیستم‌هایی که سرعت پردازش در آن‌ها دارای اهمیت است، یکی از بهترین روش‌ها، استفاده از سیستم‌های سخت‌افزاری مبتنی بر این تراشه است.

## ۶-۲ پیشنهادات و کارهای آینده

به دو صورت می‌توان پروژه ارائه شده را ارتقا داد. در مورد اول می‌توان الگوریتم پیشهادی رفع نویز ضربه‌ای را به نحوی تغییر داد تا بازدهی آن در رفع نویز ضربه بهتر شود، البته لازم به ذکر است که باید سعی کرد که الگوریتم دارای حجم و محاسبات پیچیده‌ای نباشد تا پیاده‌سازی آن بصورت سخت‌افزاری نیز بهینه باشد. در مورد دوم می‌توان سخت‌افزار مورد استفاده را جهت پیاده‌سازی یک سیستم بهینه و پرسرعت ارتقا داد. بطور مثال می‌توان از یک ارتباط با سرعت بالاتر برای تبادل اطلاعات تصویر بین نرم‌افزار متلب و بورد FPGA استفاده کرد تا زمان پردازش سخت‌افزاری نیز کاهش یابد. یکی از واسط‌های معروف، ارتباط پرسرعت FIFO[1] با میزان سرعت متوسط ۱۰ مگابایت بر ثانیه می‌باشد که توسط بورد توسعه پازج و همچنین نرم‌افزار متلب قابل پشتیبانی است. با توجه به نمونه

[1] Frist-in Frist-out (FIFO)

پروژه‌ای که بر روی بورد پازج پیاده‌سازی گردید، این واسط در کنار تراشه FPGA و نرم‌افزار متلب قادر است در زمانی کمتر از ۷۰ میلی‌ثانیه، پیکسل‌های یک تصویر خاکستری با اندازه ابعاد ۲۵۶×۲۵۶ را وارونه (NOT) کند. برای افزایش سرعت پردازش در تراشه نیز می‌توان از تراشه‌های جدیدتر استفاده کرد، بطور مثال یک تراشه در خانواده Xilinx Virtex-7 با دو میلیون سلول منطقی، قادر است الگوریتم‌های حجیم پردازش تصویر و یا پردازش سیگنال را با سرعتی بالایی اجرا کند. این پروژه بصورت برون خط[1] پیاده‌سازی شد، بعبارتی تصویر با استفاده از یک واسط مثل نرم‌افزار متلب به بورد ارسال یا دریافت شد. در پروژه‌های برخط[2] از یک دوربین نصب شده بر روی بورد برای دریافت عکس استفاده می‌گردد و همچنین می‌توان برای نمایش تصویر بازیابی شده بر روی یک نمایشگر گرافیکی، از درگاه VGA[3] استفاده کرد.

[1] Offline
[2] Real Time
[3] Video Graphics Array (VGA)

# منابع و مراجع


[1] V. A. Pedroni, "Circuit Design with VHDL", *Massachusetts Institute of Technology*, United States of America, 2004.

[2] D. Chen and J. Cong, "FPGA Design Automation: A Survey", *University of Illinois at Urbana-Champaign*, Department of Electrical and Computer Engineering, 2006.

[3] D. G. Bailey, "Design for embeded image processing on FPGA", *Massey University*, New Zealand, 2011.

[4] M. J. Beauchamp and et al, "Architechtal Modification to enhance in floating point performance of FPGA", *IEEE Transaction on very Large Scale Integration (VLSI) Systems*, vol. 16, Issue. 2, pp. 177-187, Feb 2008.

[5] P. J. Ashenden, "The VHDL Cookbook", *University of Adelaide*, Department of Computer Science, South Australia, 1990.

[6] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", *Prentice Hall*, 3nd edition, 2002.

[7] I. Pitas and A. Venetsanopoulos, "Nonlinear digital filters: principles and applications", *MA: Kluwer Academic*, Boston, 1990.

[8] J. Astola and P. Kuosmanen, "Fundamentals of nonlinear digital filtering", *Boca Raton, FL: CRC*, 1997.

[9] H. Hwang and R. A. Haddad, "Adaptive Median Filters: New Algorithms and Results*", IEEE Transaction on image processing*, vol. 4, no. 4, 1995.

[10] L. Yin and et al, "Weighted Median Filters: A Tutorial", *IEEE Transaction on circuits and systems II: Analog and Digital Signal Processing*, vol. 43, no. 3, 1996.

[11] SJ. KO and et al, "Center Weighted Median Filters and Their Applications to Image Enhancement", *IEEE Transaction on circuits and systems*, vol. 38, no. 9, 1991.

[12] Z. Wang and D. Zhang, "Progressive switching median filter for the removal of impulse noise from highly corrupted images", *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, issue. 1, pp. 78–80, Jan 1999.

[13] T. Chen and H. R. Wu, "Adaptive Impulse Detection Using Center-Weighted Median Filters", *IEEE Signal Processing Letters*, vol. 8, no. 1, Jan 2001.

[14] H.-L. Eng and K.-K. Ma, "Noise adaptive soft-switching median filter", *IEEE Transaction on Image Processing*, vol. 10, no. 2, pp. 242–251, 2001.

[15] R. Garnett and et al, "A Universal Noise Removal Algorithm with an Impulse Detector", *IEEE Transaction on Image Processing*, vol. 14, issue. 11, pp. 1747-1754, Nov 2005.

[16] K. S. Srinivasan and D. Ebenezer, "A New Fast and Efficient Decision-Based Algorithm for Removal of High-Density Impulse Noises", *IEEE Signal Processing Letters*, vol. 14, no. 3, 2007.

[17] S.-S. Wang and C.-H. Wu, "A new impulse detection and filtering method for removal of wide range impulse noises", *Pattern Recognition*, vol. 42, pp. 2194-2202, 2009.

[18] V.R. Vijaykumar and et.al, "Decision based adaptive median filter to remove blotches, scratches, streaks, stripes and impulse noise in images", *IEEE 17th International Conference on Image Processing*, pp. 117-120, 2010.

[19] V. Jayaraj and D. Ebenezer, "A new switching-based median filtering scheme and algorithm for removal of high-density salt and pepper noise in images", *Journal on Advances in Signal Processing (EURASIP)*, 2010.

[20] S. Esakkirajan and et.al, "Removal of High Density Salt and Pepper Noise through Modified Decision Based Unsymmetric Trimmed Median Filter", *IEEE Signal Processing Letters*, vol. 18**,** no. 5, 2011.

[21] T. Sunilkumar and et al, "Removal of high density Impulse Noise through Modified Non-Linear Filter", *Journal of Theoretical and Applied Information Technology*, vol. 48, no. 2, 2013.

[22] S. Sharma and P. Yadav, "Removal of Fixed Valued Impulse Noise by Improved Trimmed Mean Median Filter", *IEEE International Conference on Computational Intelligence and Computing Research*, 2014.

[23] S. Shrestha, "Image denoising using new Adaptive Based Median Filter", *International Journal of Signal & Image Processing (SIPIJ)*, vol. 5, no. 4, 2014.

[24] G. Hanj and M.V. Latte, "Performance Study of Novel Median Filter for Image De-Noising", *International Journal of Computer Applications*, vol. 122, no. 23, 2015.

[25] Z. Wang and A. C. Bovik, "A new look at signal fidelity measures", *IEEE Signal Processing Magazine*, pp. 98-117, 2009.

[26] P. P. Chu, "FPGA Prototypig by vhdl examples: Xilinx Spartan-3 Version", *Cleveland State University*, Ohio, United States, 2008.

[27] M. O. Tokhi and et al, "Parallel Computing for Real-time Signal Processing and Control", *United States*, 2003.

[28] Online Available: www.Altera.com

[29] Online Available: www.Xilinx.com

[30] Online Available: www.posedge.ir

[۳۱] صباغیان‌بیدگلی حسین، *طراحی خودکار مدارهای دیجیتال*، چاپ سوم، انتشارات جنگل، تهران، ۱۳۸۸.

[۳۲] زولینسکی مارک، *طراحی سیستم دیجیتالی با استفاده از VHDL*، ترجمه علیرضا فتاح، چاپ اول، انتشارات نورپردازان، تهران، ۱۳۸۴.

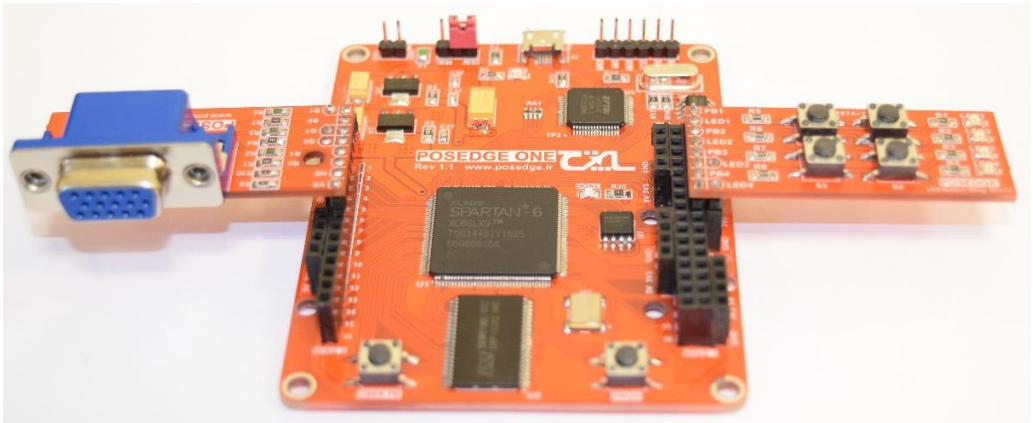
[۳۳] محرابی محمدعلی، *طراحی مدارهای دیجیتال با VHDL*، انتشارات باغبانی، مشهد، ۱۳۸۵.

# پیوست

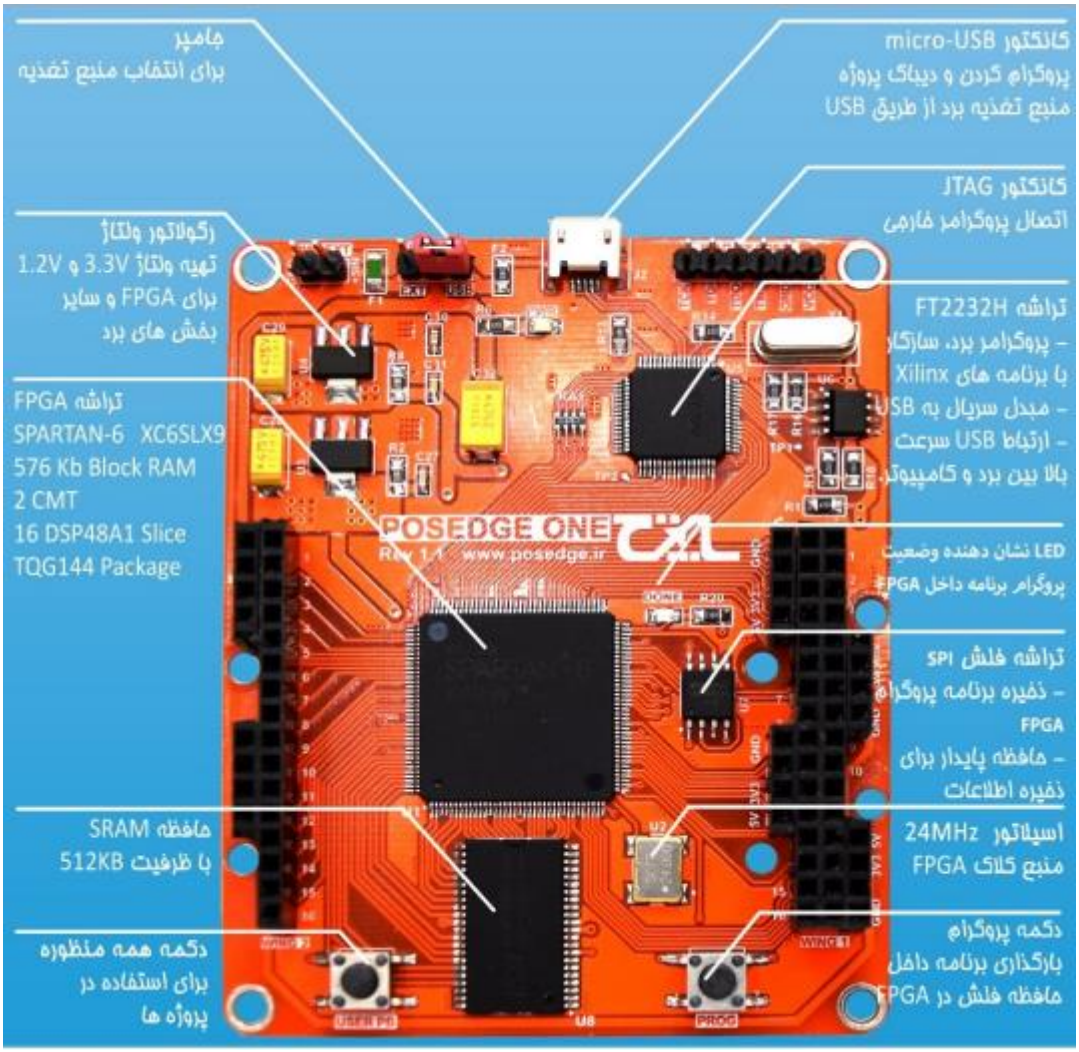
## ضمیمه ۱: آشنایی با بورد توسعه FPGA پازج-یک، مبتنی بر Xilinx Spartan-6

"پازج یک" یک پلتفرم آموزشی FPGA با پشتیبانی و آموزش‌های فارسی و مناسب برای دانشجویان و نوآموزانی است که قصد ورود به دنیای طراحی با FPGA را دارند. این بورد مبتنی بر تراشه‌ای از خانواده Xilinx SPARTAN-6 بنا شده است. در ادامه مشخصات بورد را بیشتر مورد بررسی قرار می‌دهیم.

- تراشه‌ی Xilinx Spartan6-LX9
- پروگرامر USB روی بورد با قابلیت پشتیبانی توسط مجموعه‌ی نرم‌افزاری Xilinx
- چهار مگابیت حافظه‌ی SRAM، تراشه‌ی IS61WV5128BLL
- واسط USB 2.0 با حداکثر سرعت ۱۰ مگابایت بر ثانیه توسط تراشه‌ی FT2232H
- حافظه‌ی Flash با ظرفیت ۶۴ مگابیت جهت ذخیره برنامه‌ی FPGA توسط تراشه‌ی W25Q64
- ۴۸ پین ورودی-خروجی همه منظوره
- توسعه‌ی وسائل جانبی (peripheral) از طریق کانکتور بال(Wing)
- اسیلاتور ۲۴ مگاهرتز




شکل ۱: نمایی از بورد توسعه پازج به همراه بال‌های کلید-LED و VGA




شکل۲: بورد پازج با ذکر جزئیات قسمت‌های مختلف

### ۱-۱ تراشه‌ی Xilinx Spartan6-LX9

تراشه استفاده شده در این پروژه Xilinx Spartan-6 سری XC6SLX9 می‌باشد. این تراشه از خانواده Spartan و از شرکت معروف Xilinx است که نسبت به نسل قبلی خود (Spartan-3) دارای برتری‌های زیادی دارد که می‌توان موارد زیر را بیان کرد.

- ❖ مصرف توان پایین‌تر (حدود ۵۰٪) به لطف تکنولوژی ساخت ۴۵ نانومتر.
- ❖ افزایش دو برابری ظرفیت لاجیک FPGA به‌واسطه‌ی جایگزینیLUTهای چهار ورودی نسل قبلی باLUTهای شش ورودی در Spartan-6.
- ❖ افزایش ظرفیت بلوک‌هایDSP

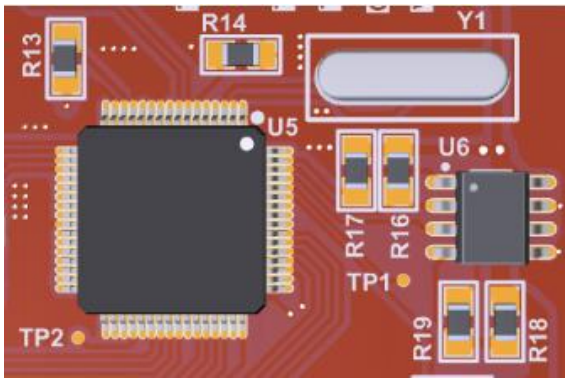
❖ پشتیبانی از استاندارد ارتباطی TMDS با امکان برقراری ارتباط ویدئویی HDMI و DVI بدون نیاز به هیچ‌گونه سخت‌افزار اضافی.

بهبود منابع کلاک با استفاده از واحدهای CMT[1] که علاوه بر DCM[2] حاوی یک واحد PLL[3] نیز می‌باشند (برخلاف Spartan-3 که تنها دارای DCM است).

### ۱-۲ درگاه USB دو کاناله

"پازج یک" برای برقراری ارتباط با کامپیوتر از تراشه‌ی مبدل FT2232H استفاده می‌کند که دو کانال ارتباطی مجزا را در اختیار قرار می‌دهد:

- کانال A: این کانال تنها جهت برنامه‌ریزی FPGA و SPI Flash متصل به آن به کار می‌رود.
- کانال B: این کانال قابلیت پیکربندی هر دو حالت ارتباط سریال UART با حداکثر سرعت ۱۲ مگابیت بر ثانیه و ارتباط Asynchronous FIFO با سرعت ۱۰ مگابایت بر ثانیه را دارد.



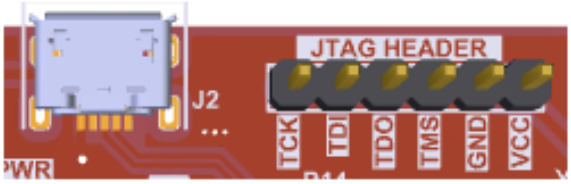

شکل۳: نمایی از تراشه FT2232H بر روی بورد پازج

### ۱-۳ پروگرمر USB روی بورد

یکی از ویژگی‌های قابل توجه "پازج یک" برخورداری از پروگرامر USB روی بورد است که توسط مجموعه‌ی نرم افزاری Xilinx ISE بطور کامل پشتیبانی می‌شود. این ویژگی جالب که تا قبل از این تنها در چند مدل از بوردهای آموزشی گران قیمت به چشم می‌خورد، کاربر را از خرید پروگرامرهای

[1] Clock Management Tile (CMT)
[2] Digital Clock Manager (DCM)
[3] Phase Locked Loop (PLL)

USB گران قیمت موجود در بازار (که اکثراً بی‌کیفیت هستند) بی‌نیاز می‌سازد. نکته‌ای که لازم است بر روی آن تاکید شود این است که پروگرامر مذکور بدون نیاز به نصب هیچ‌گونه نرم‌افزار اضافی و اعمال تنظیمات خاص از سوی کاربر قابل استفاده بوده و کلیه‌ی درایورهای مورد نیاز جهت کارکرد این پروگرامر در هنگام نصب مجموعه‌ی ISE بصورت اتوماتیک نصب خواهند شد. لازم به ذکر است که پروگرامر مذکور علاوه بر نرم‌افزار Impact (جهت برنامه‌ریزی FPGA)، در نرم‌افزارهای Chipscope (جهت عیب‌یابی سخت‌افزار) و SDK (جهت برنامه‌ریزی و عیب‌یابی MicroBlaze) نیز بطور اتوماتیک قابل شناسایی و استفاده است.




شکل۴: نمایی از USB روی بورد جهت پروگرم کردن تراشه

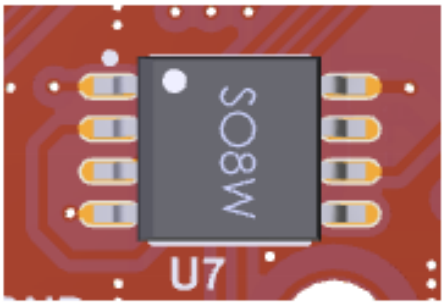
### ۴-۱ حافظه‌ی خارجی SRAM

هر چند در یک قیمت مساوی SRAM در مقایسه با نوع حافظه‌ی DRAM از ظرفیت کمتری برخوردار است، اما مزیت مهم SRAMها، سادگی استفاده از آن‌ها است که برای یک بورد آموزشی از اهمیت بیشتری برخوردار است. "پازج یک" مجهز به حافظه IS61WV20488BLL است که ۵۱۲ کیلوبایت حافظه‌ی SRAM را در اختیار کاربر قرار می‌دهد. این میزان حافظه، محدودیت حافظه‌ی برنامه‌ی MicroBlaze را که ناشی از محدود بودن آن به حافظه‌ی داخلی FPGA است، برطرف نموده و فضای کافی برای نوشتن کدهای نسبتاً طولانی برای پردازنده‌ی MicroBlaze را فراهم می‌نماید. استفاده بعنوان Frame Buffer جهت نمایش تصویر روی خروجی VGA نیز می‌تواند یکی دیگر از نمونه کاربردهای این حافظه باشد.


شکل۵: نمایی از حافظه‌ی خارجی SRAM

### ۱-۵ حافظه‌ی SPI Flash

از آنجایی که FPGAها عموماً فاقد حافظه‌ی غیرفرار جهت ذخیره‌ی فایل پیکربندی هستند، با قطع تغذیه نیاز به برنامه‌ریزی مجدد دارند. بنابراین نیاز است تا فایل پیکربندی آن‌ها در یک حافظه‌ی غیرفرار ذخیره شود و هر بار پس از اتصال تغذیه مجدداً فراخوانی شده و FPGA را پیکربندی نماید. پازج-یک جهت ذخیره‌ی فایل پیکربندی FPGA از حافظه‌ی فلش W25Q64 با ظرفیت ۶۴ مگابیت استفاده می‌کند. این حجم از حافظه آنقدر زیاد است که کاربر می‌تواند تا ۲۳ فایل پیکربندی متفاوت را کنار یکدیگر روی حافظه ذخیره کرده و با استفاده از قابلیت MultiBoot موجود در SPARTAN-6، هر بار FPGA را توسط یکی از آن‌ها پیکربندی نماید.

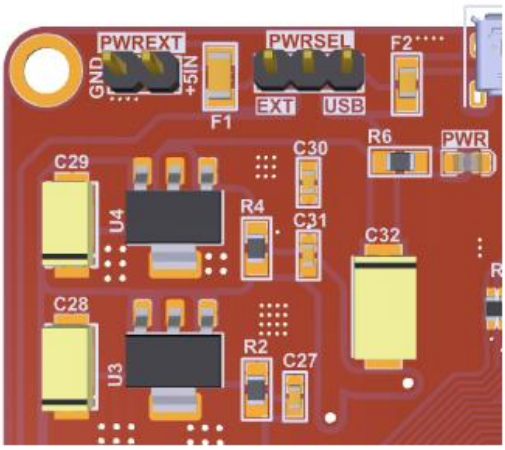



شکل ۶: نمایی از حافظه‌ی SPI Flash

### ۱-۶ بخش تغذیه

تغذیه‌ی "پازج یک" توسط کانکتور Micro USB موجود روی بورد تامین می‌شود (همان کانکتوری که نقش پروگرامر و واسط ارتباطی USB با کامپیوتر را نیز دارد). دو عدد رگولاتور خطی

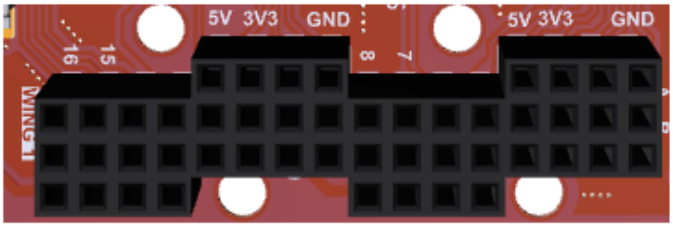
وظیفه‌ی تبدیل ولتاژ ۵ ولت تامین شده توسط پورت USB به ولتاژهای ۳/۳ و ۱/۲ ولت را دارند. مطابق استاندارد، حداکثر جریان قابل تامین توسط درگاه USB 2.0 معادل ۵۰۰ میلی‌آمپر است که در اکثر آن‌ها این مقدار جریان جهت کارکرد بورد و بال‌های آن کافی است. اما چنانچه در موارد معدودی این مقدار کفاف نیاز بورد را ندهد، امکان تامین تغذیه از خارج توسط کانکتور دو پین تعبیه شده روی بورد نیز فراهم است.




شکل ۷: نمایی از بخش تغذیه بورد پازج

### ۱-۷ ورودی-خروجی‌ها

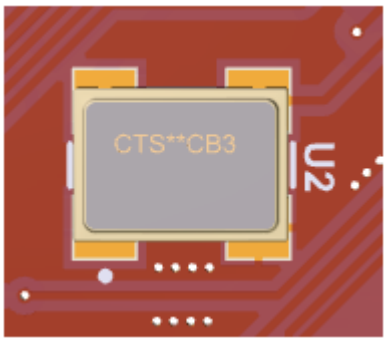
"پازج یک" برای ارتباط با جهان خارج از ۴۸ عدد پین ورودی-خروجی (GPIO) همه منظوره استفاده می‌کند که در قالب کانکتوری با طرح ویژه به نام بال (wing) در اختیار کاربر قرار گرفته‌اند. ویژگی شاخص کانکتور بال این است که در کنار هر ۸ پین GPIO، ۴ عدد پین تغذیه (شامل GND و ۳ ولتاژ تغذیه‌ی متفاوت که عموما ۵، ۳/۳ و ۲/۵ ولت هستند) قرار داده است که می‌تواند جهت تامین تغذیه‌ی تجهیزات جانبی متصل شده به بورد بکار رود. کاربر می‌تواند به نحو دلخواه از پین‌های ورودی-خروجی فراهم شده استفاده نموده و یا از آن‌ها برای اتصال بورد به ابزارهای جانبی مخصوص آن (با نام "بال") استفاده نماید.




شکل ۸: ورودی- خروجی‌های همه منظوره بورد پازج

### ۱-۸ اسیلاتور

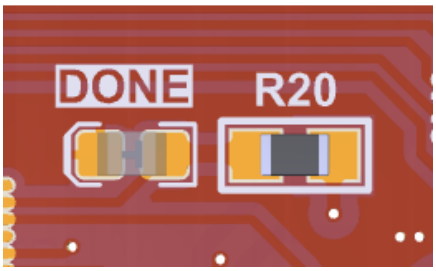
یک اسیلاتور ۲۴ مگاهرتزی بعنوان مولد پالس ساعت ورودی FPGA بکار رفته است که توسط واحدهای CMT داخلی FPGA می‌تواند محدوده‌ی وسیعی از فرکانس‌ها را تولید نماید. لازم به ذکر است تراشه‌ی SPARTAN6-LX9 دارای دو واحد CMT است که هر یک دارای یک واحد PLL و دو واحد DCM است که این امر انعطاف‌پذیری بالایی جهت تولید فرکانس‌های دلخواه فراهم می‌نماید.




شکل ۹: نمایی از اسیلاتور ۲۴ مگاهرتزی بورد پازج

### ۱-۹ LED نشان‌دهنده‌ی وضعیت پیکربندی FPGA

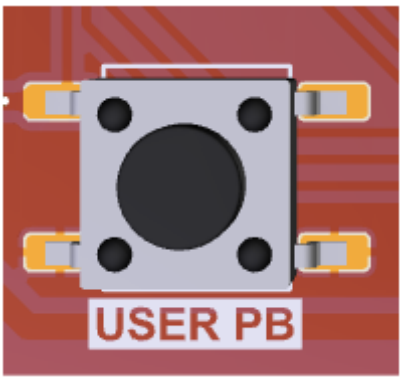
یک LED آبی رنگ با برچسب DONE بر روی بورد قرار گرفته است که نشان‌دهنده‌ی وضعیت پیکربندی FPGA است. روشن بودن این LED نشان‌دهنده‌ی این است که عملیات پیکربندی به طور کامل انجام شده است و خاموش بودن آن نشان‌دهنده‌ی عدم پیکربندی کامل FPGA است.




شکل ۱۰: نمایی از LED نشان دهنده‌ی وضعیت پیکربندی تراشه

### ۱-۱۰ کلید فشاری همه منظوره

یک کلید فشاری[1] روی بورد پازج قرار داده شده است که می‌تواند به عنوان یک ورودی همه منظوره بکار گرفته شود. یکی از موارد پرکاربرد این کلید می‌تواند استفاده بعنوان Reset خارجی پردازنده‌ی MicroBlaze باشد. توجه داشته باشید که عملکرد این کلید بصورت Active-Low است، بدین مفهوم که در حالت عادی مقدار پین متصل شده به آن "۱" و در حالت فشردن کلید مقدار آن "۰" می‌شود.




شکل ۱۱: نمایی از کلید همه منظوره بورد پازج

[1] Push Button

## ضمیمه ۲: آشنایی با نرم‌افزار ISE Design Suite 14.7 و شروع کار با پازج-یک

برای شروع کافی است:

۱- بورد پازج را از طریق کابل USB همراه بورد به یکی از درگاه‌های USB کامپیوتر خود متصل نمایید.

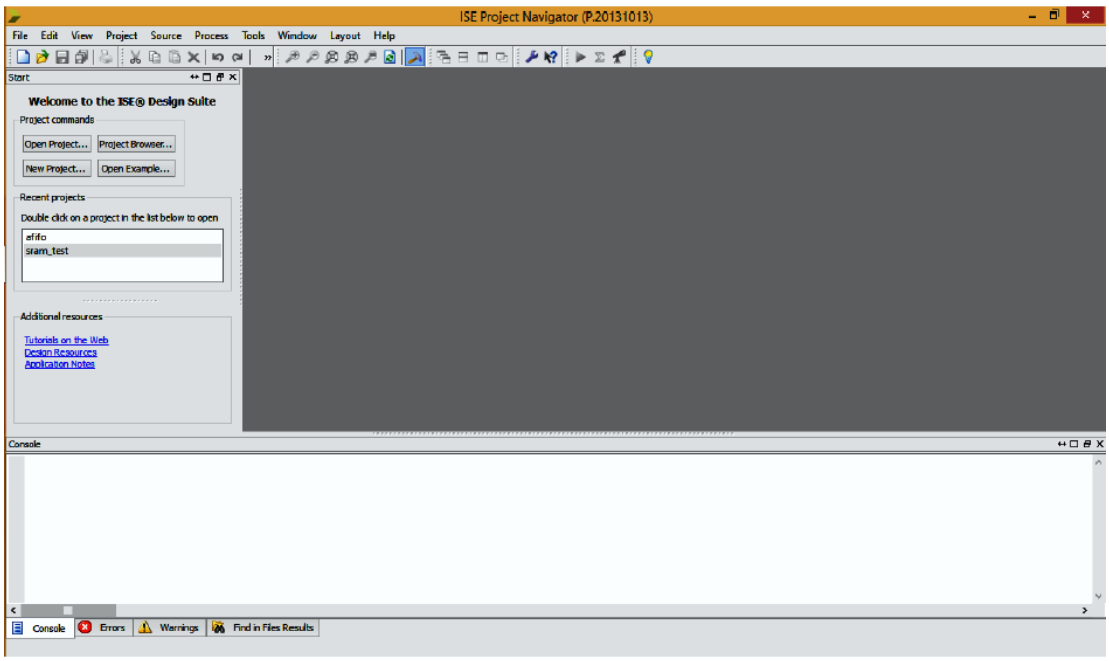
۲- اتصال تامین تغذیه بورد را در حالت استفاده از USB به عنوان منبع تغذیه قرار دهید.

۳- برنامه مد نظرتان را در نرم افزار ISE نوشته، سنتز و آماده پیاده‌سازی نمایید.

۴- بال مناسب با کاربردتان را بر روی یکی از کانکتورهای انتخاب شده نصب نمایید.

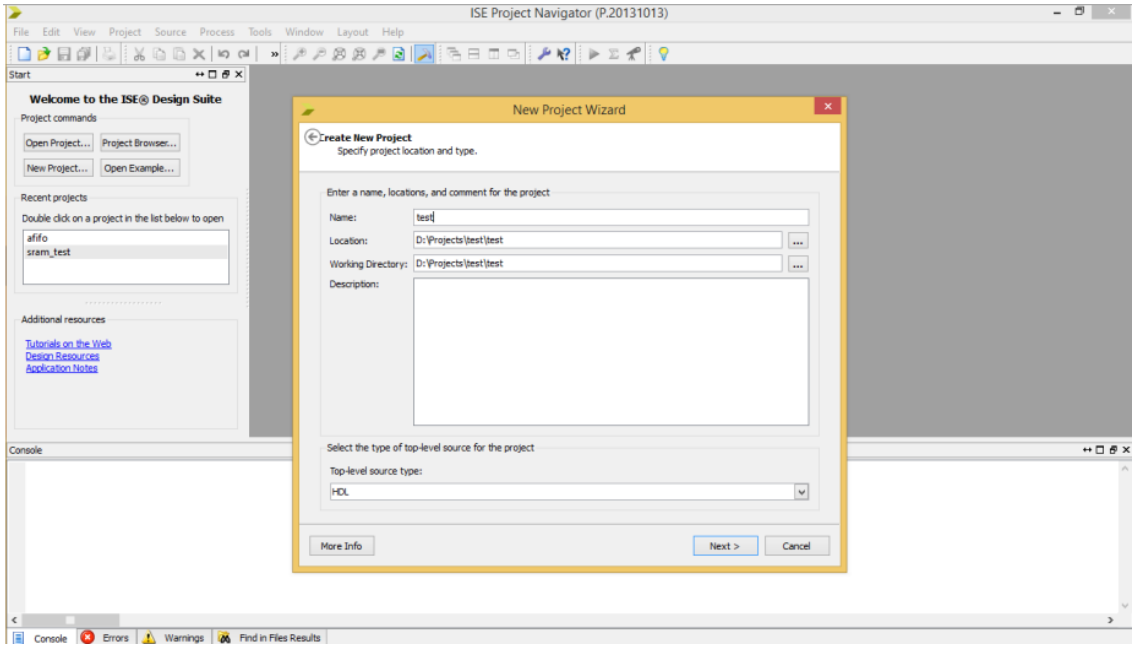
۵- پازج را پروگرام کنید. از مشاهده نتیجه تلاشتان لذت ببرید! در ادامه مراحل فوق با یک مثال نشان داده شده است.

نرم‌افزار ISE را باز کنید. پیشنهاد ما استفاده از آخرین نسخه‌ی این نرم‌افزار (نسخه‌ی ۱۴/۷) است که از وبسایت گروه پی‌سی دانلود قابل دانلود است. در صورتی که از نسخه‌های قدیمی‌تر این نرم‌افزار استفاده کنید، ممکن است پروگرامر روی بورد توسط نرم‌افزار به درستی شناسایی نشود!



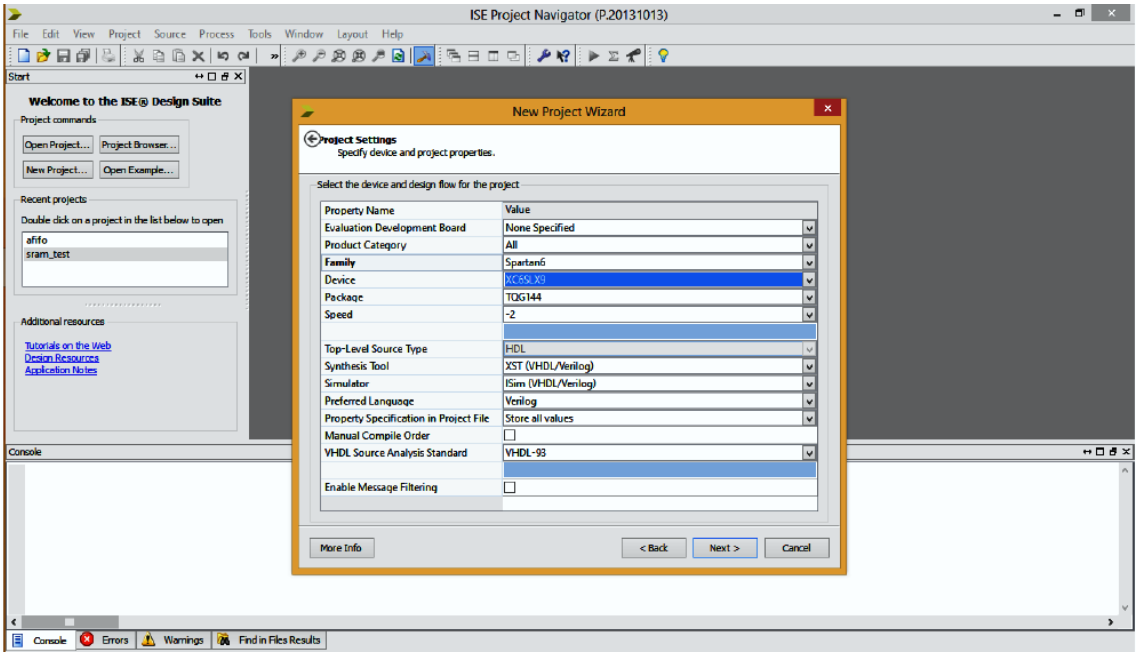

شکل ۱۲: نمایی از محیط نرم‌افزار ISE 14.7

با انتخاب file – new project، ویزارد مربوط به ساخت یک پروژه جدید را آغاز کنید.

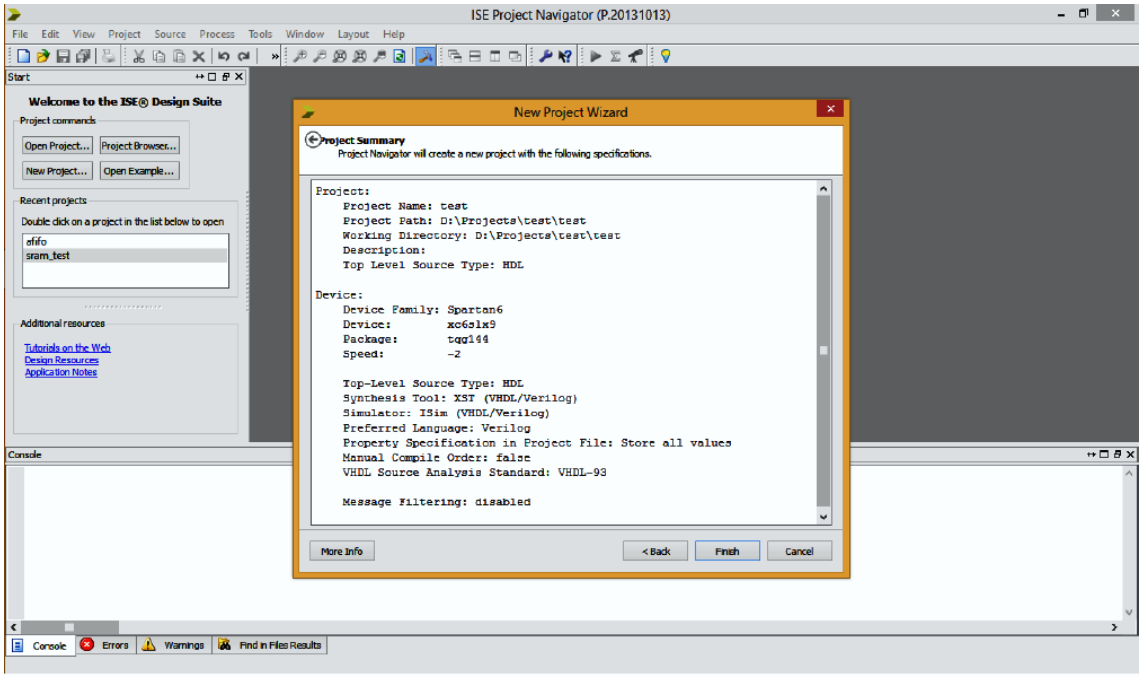



شکل ۱۳: نمایی دیگر از محیط نرم‌افزار ISE 14.7

مشخصات تراشه مورد استفاده را مانند تصویر زیر تنظیم نمایید.

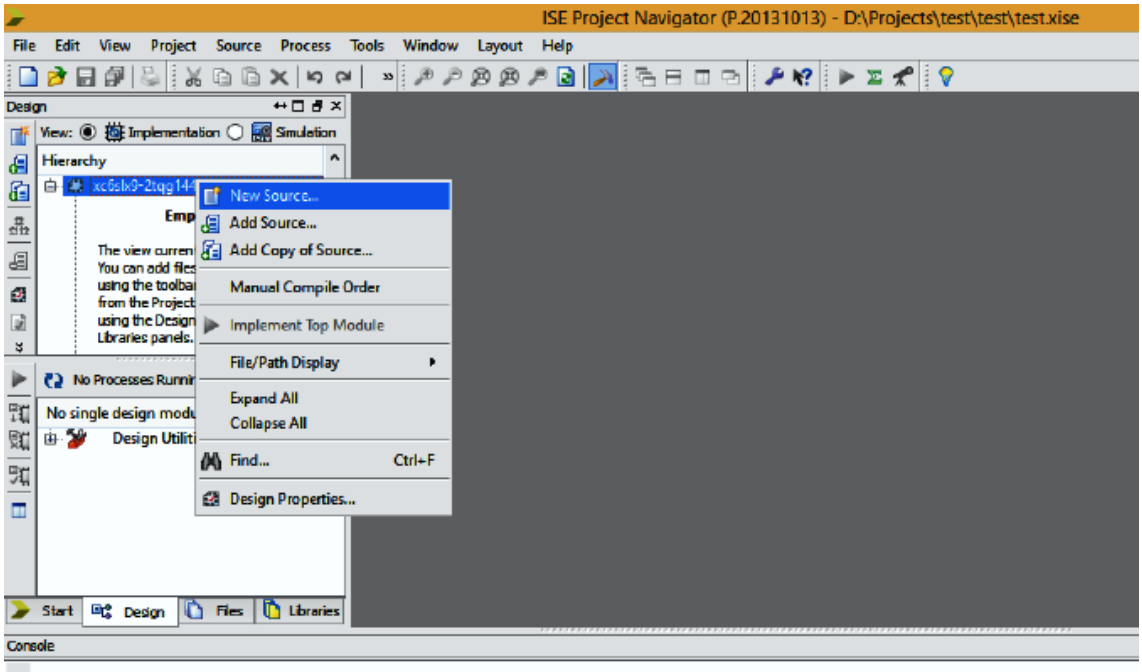



شکل ۱۴: انتخاب تراشه محیط نرم‌افزار ISE 14.7



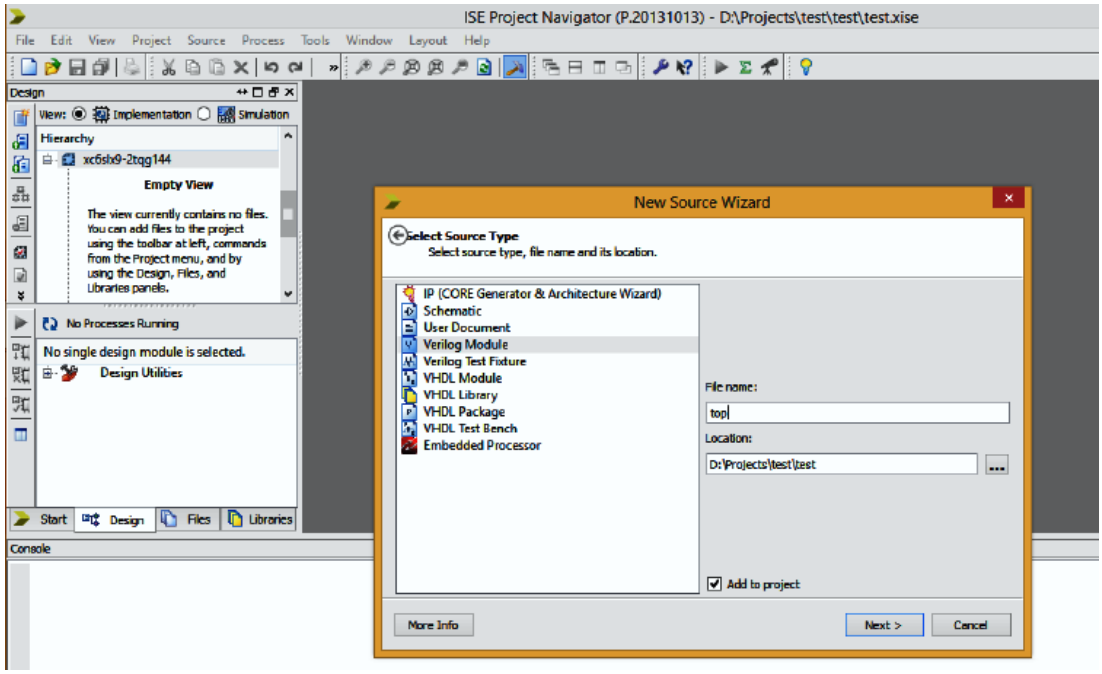

شکل ۱۵: مشخصات تراشه انتخابی محیط نرم‌افزار ISE 14.7

مطمئن شوید که همه چیز درست بوده و Finish را بزنید. در ادامه با کلیک راست بر روی نام تراشه یک طرح جدید را ایجاد نمایید.



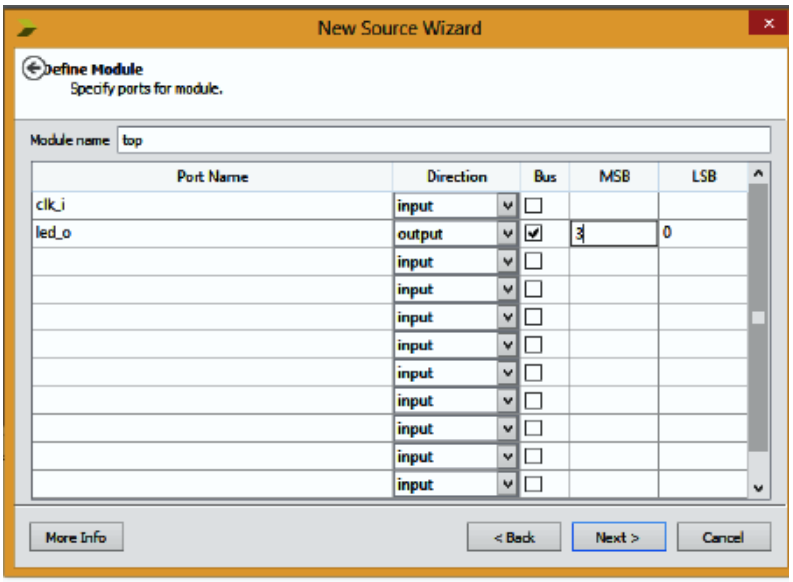

شکل ۱۶: ایجاد طرح جدید در محیط نرم‌افزار ISE 14.7

در ویزارد، Verilog moduleیا VHDL moduleرا انتخاب کنید و طرح را نامگذاری نمایید.



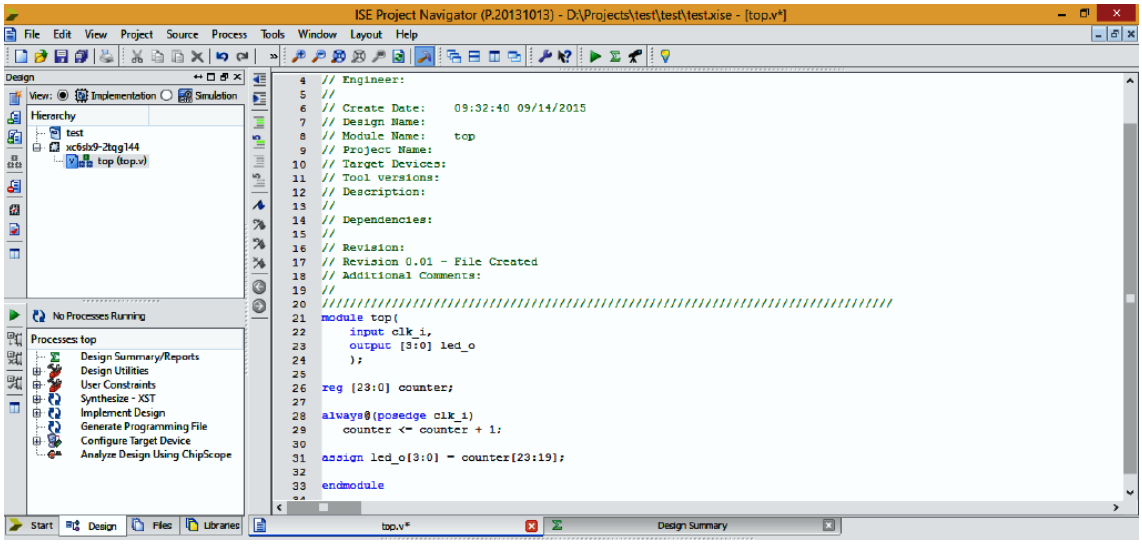

شکل ۱۷: نحوه ایجاد ماژول در طرح جدید

پین‌ها و پورت‌های ورودی–خروجی را نامگذاری کنید، جهت هر کدام (ورودی یا خروجی) را مشخص نموده و برای هر پورت عرض آن را تعیین نمایید (در ادامه نیز می‌توان توسط نرم‌افزار PlanAhead پورت‌ها را مدیریت و تنظیم کرد).



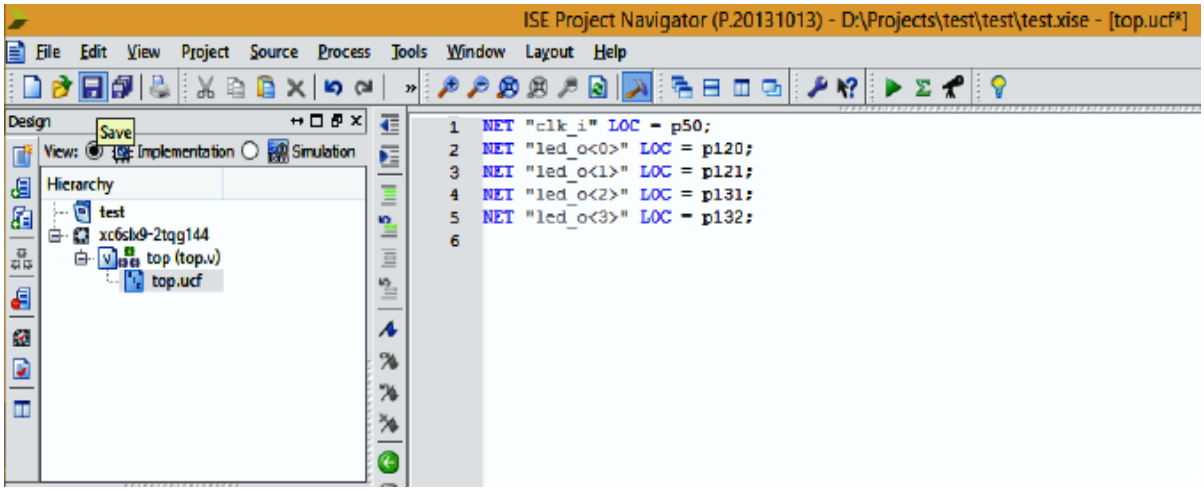

شکل ۱۸: نحوه تنظیم کردن پورت‌های تراشه

مطمئن شوید که همه چیز درست بوده، Next و سپس Finish را بزنید. در ادامه برنامه را بنویسید!



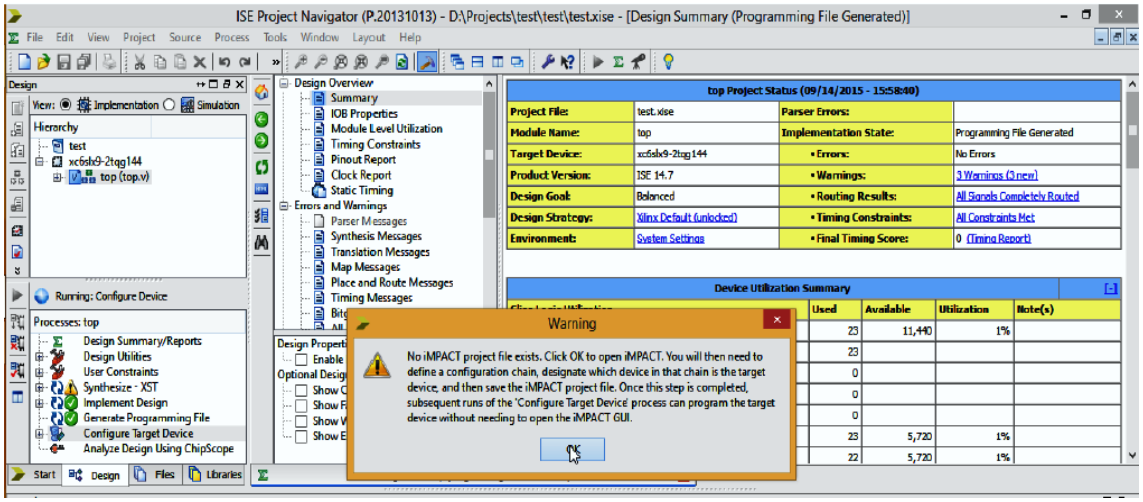

شکل ۱۹: محیط برنامه نویسی نرم‌افزار ISE

مجددا یک طرح فایل جدید ایجاد کنید. این بار Implementation Constrains File را انتخاب نموده و آن را نامگذاری نمایید. فایلی با پسوند ucf ایجاد می‌شود. در این فایل شما باید اطلاعات مربوط به پایه‌های ورودی-خروجی‌های FPGA را وارد نمایید. بعبارتی مشخص کنید که هر کدام از ورودی-خروجی‌های طرح شما باید به کدام یک از پایه‌های FPGA متصل شود. جهت تسهیل این مرحله از طراحی، اطلاعات تمامی پایه‌های FPGA در بورد پازج یک و محل اتصال آن‌ها در قالب یک فایل ucf آماده شده که از طریق وبسایت گروه پازج قابل دانلود است. در این مثال ورودی-خروجی‌های طرح، به پایه‌هایی از FPGA متصل شده‌اند که به بال GPIO وصل خواهند شد.



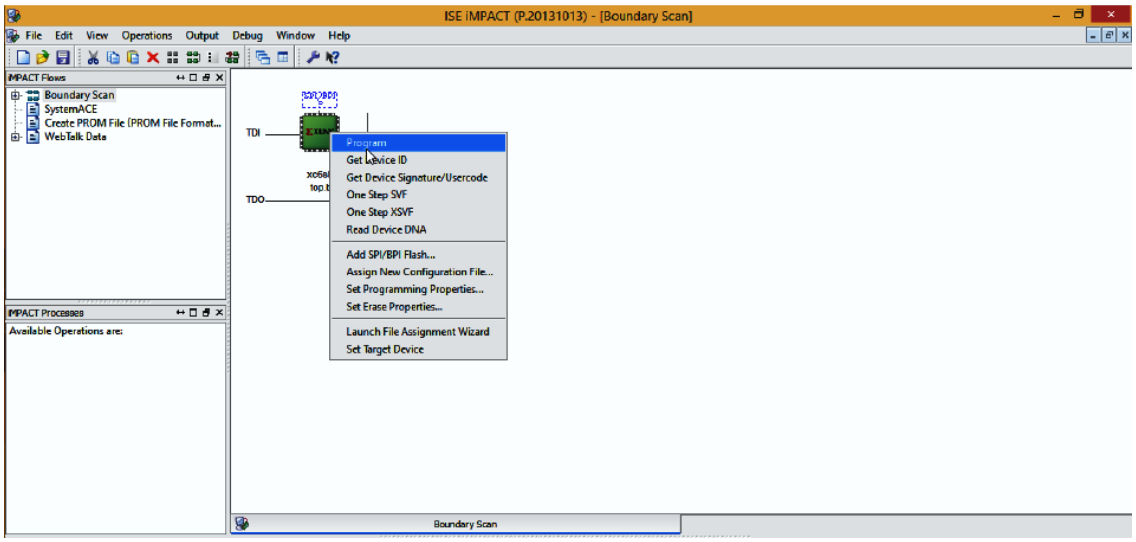

شکل ۲۰: محیط فایل ucf در نرم‌افزار ISE

پروژه آماده است، باید آن را سنتز و پیاده‌سازی کنید. بر روی Configure Target Device کلیک راست نموده و این بار Run را انتخاب نمایید.




شکل ۲۱: سنتز، پیاده‌سازی و ایجاد فایل پیکربندی bit در نرم‌افزار ISE

در نهایت برنامه Impact باز می‌شود. بر روی گزینه‌ی Boundary Scan دابل کلیک کنید. در ادامه روی قسمتی از پنجره‌ی اصلی کلیک راست کرده و Initialize Chain را بزنید. در این حالت اگر بورد به درستی از طریق USB به کامپیوتر شما متصل باشد، نرم‌افزار Impact، نوع پروگرمر (که در قسمت پایین سمت راست شکل با عبارت Digilent JTAG-HS2) مشخص شده و همچنین FPGA روی بورد را شناسایی کرده و نشان می‌دهد. پیشنهادات برنامه برای AutoAssign و Program properties را فعلاً رد کنید. روی تراشه کلیک راست کرده و Assign New Configuration File را انتخاب نمایید. و سپس فایل پیکربندی پروژه را با پسوند bit انتخاب کرده و تراشه را برنامه‌ریزی کنید. در صورت انجام دادن درست مراحل، پیام موفقیت‌آمیز بودن پیکربندی تراشه ظاهر می‌گردد.




شکل ۲۲: پروگرم کردن تراشه توسط نرم‌افزار Impact

## ضمیمه ۳: حل مشکل نرم‌افزار ISE Design Suite 14.7 در ویندوزهای 8 و 10

با توجه به اینکه اجرای این نرم‌افزار در نسخه‌های ۶۴ بیتی سیستم عامل‌های ویندوز 8 و 10 نیاز به تنظیمات خاصی دارد، در اینجا به این تنظیمات اشاره می‌شود.

فرض کنیم نرم‌افزار ISE در این مسیر نصب شده باشد:

C:\Xilinx\14.7\ISE_DS\ISE\

۱- این پوشه را باز کنید:

C:\Xilinx\14.7\ISE_DS\ISE\lib\nt64

۲- فایل libPortability.dll را پیدا کنید و نام آن را به libPortability.dll.orig تغییر دهید.

۳- فایل libPortabilityNOSH.dll را پیدا کنید و از آن یک کپی تهیه نمایید. سپس نام کپی را به libPortability.dll تغییر داده و آن را در همین پوشه قرار دهید.

۴- یک کپی دیگر از libPortabilityNOSH.dll را در پوشه زیر قرار دهید.

C:\Xilinx\14.7\ISE_DS\common\lib\nt64

۵- در پوشه زیر فایل libPortability.dll را پیدا کنید و نام آن را به libPortability.dll.orig تغییر دهید.

C:\Xilinx\14.7\ISE_DS\common\lib\nt64

۶- نام فایل libPortabilityNOSH.dll را به libPortability.dll تغییر دهید.

پس از طی این مراحل، نرم افزارهای Project Navigator، IMPACT و License Manager قابل استفاده هستند. اما برای استفاده از PlanAhead لازم است کارهای زیر نیز انجام گیرد:

در پوشه زیر فایل rdiArgs.bat را پیدا کنید و نام آن را به rdiArgs.bat.orig تغییر دهید.

C:\Xilinx\14.7\ISE_DS\PlanAhead\bin

فایل rdiArgs.bat را در پوشه زیر کپی کنید. این فایل از وبسایت گروه پازج قابل دانلود است.

C:\Xilinx\14.7\ISE_DS\PlanAhead\bin

**ضمیمه ۴: راه‌اندازی ارتباط سریال RS-232 در نرم‌افزار متلب نسخه R2015b**

از آنجا که نرم‌افزار متلب ارتباط سریال را پشتیبانی می‌کند، در این قسمت قصد داریم یک ارتباط سریال ساده توسط متلب برای ارسال و دریافت هشت بیت داده بدون علامت (متناظر با هر پیکسل تصویر) راه‌اندازی کنیم.

۱- در ابتدا باید شی پورت سریال را برای متلب ایجاد کرد. بسته به این که پورت چه شماره‌ای دارد، می‌توان مانند زیر عمل کرد:

```
s = serial('COM1');
```

۲- در ادامه باید مشخصات ارتباطی را برای ارتباط سریال ایجاد شده تعیین کرد. این مشخصات عبارت‌اند از نرخ ارسال داده، بیت خطایاب، تعداد بیت داده، میزان زمان خواندن و نوشتن، تعداد بیت توقف، اندازه بافر ورودی و اندازه بافر خروجی.

```
s = serial ('COM1',
      'Baudrate', 921600, ...
      'Parity', 'none', ...
      'Databits', 8, ...
      'Timeout', 1, …
      'Stopbits', 1, ...
      'InputBufferSize', (8) ...
      'OutputBufferSize', (8));
```

بطور مثال ارتباط سریال بالا، مشخصاتی با نرخ ارسال ۹۲۱۶۰۰ بیت بر ثانیه، بدون هیچ بیت خطایاب، هشت بیت داده، زمان خواندن و نوشتن یک ثانیه‌ای، یک بیت توقف، آرایه بافر ورودی با طول ۸ بیت و آرایه بافر خروجی با طول ۸ بیت را داراست.

۳- با دستور fopen پورت سریال ایجاد شده را باز می‌کنیم، به عبارتی برای استفاده آماده می‌گردد.

```
fopen(s);
```

۴- با دستور fwrite و fread می‌توان مقدار هشت بیت داده بدون علامت را در پورت سریال نوشت و یا از آن خواند.

```
fwrite (s, pixel,'uint8');

q = fread (s, pixel,'uint8');
```

۵- با دستور fclose هم می‌توان درگاه ارتباط سریال را بست.

```
fclose(s);
```

## Abstract

One of the main topics in image processing is salt and pepper impulsive noise removal from digital images. This type of noise randomly generating white (salt) and black (pepper) points in images and causing the loss of information and details in images. The purpose of this paper is designing an algorithm to remove impulsive noise from gray and color images using combination of median and mean filters. In addition to high efficiency in noise removal, volume calculations are not complicated. The proposed method consists of two phases identify noise and noise removal. In first considered the low and high threshold values to identify noisy pixels. In the second stage the decision is using the adaptive median and mean filters based on the 1st order neighborhood and diagonal neighborhood pixels. Furthermore, the proposed algorithm is compared with recent articles using PSNR, MSE, MAE and SSIM parameters in Different densities impulsive noise. Visual and numerical results show good performance of the proposed method is the removal of impulsive noise. In the end, according to parallel processing and real-time features in FPGA chip, proposed method using VHDL hardware description language writing, synthesized and implemented on Xilinx Spartan-6 series XC3SLX9. The total power consumption, maximum output frequency and average processing time in FPGA for a color image with dimensions 256×256 and 90% densities impulsive noise, respectively values of 14mw, 42.186 MHz and 1.299 sec is obtained Which reflects the performance of the proposed system is efficient and fast.

## Keywords

Salt and pepper impulsive noise, Noise removal, Median filter, Mean filter, Neighborhood information, VHDL, FPGA.


**Faculty of Electrical and Robotics Engineering**

**MSc Thesis in Electronic Engineering**

# Design and Implementation of a Noise Cancellation System for Digital Images Based on FPGA

**By: Moslem Khanebabaei**

**Supervisor:**

**Dr. Ali Solimanieyvari**

**Advisor:**

**Dr. Mohammadreza Ashraf**

**September 2016**