طراحی و پیاده سازی معماری نوین برای پردازش تصاویر دیجیتال با استفاده از پیکربندی جزیی بر روی FPGA

امیر چکینی[۱]

سعید عزتی[۲]

چکیده

در این مقاله با استفاده از رویکرد و نگرشی جدید، معماری نوینی برای پردازش تصاویر دیجیتال بر روی سخت افزار ارائه می شود . یکی از بزرگترین معایب راه حل های سخت افزاری برای استفاده به جای سامانه های نرم افزاری ، عدم انعطاف پذیری آنهاست، هرچند که سایر پارامترهای مهم نظیر قابلیت اعتماد و سرعت عملکرد آنها در طراحی بسیار جذاب هستند . در این مقاله فن پیکربندی جزیی پویا[۳] به عنوان راه نوینی برای مرتفع نمودن این مشکل پیشنهاد می گردد. این مقاله با ارائه معماری نوینی مبتنی بر پیکربندی جزیی امکان پیاده سازی چندین فیلتر مختلف برای پردازش تصاویر را به صورت بلادرنگ برروی سخت افزار می دهد . با استفاده از این فن می توان به پردازش بلادرنگ تصاویر در عین مصرف توان کم ، کارآیی و سرعت بالا بر روی سخت افزار FPGA دست یافت . معماری ارائه شده با استفاده از زبان استاندارد توصیف سخت افزار VHDL نوشته و سپس بر روی FPGA پیاده سازی می شود. با مشاهده و ارائه نتایج حاصل از شبیه سازی، سنتز و پیاده سازی نشان خواهیم داد که معماری پیشنهادی و سیستم طراحی شده در این مقاله دارای سرعت و کارآیی بالا ، مصرف توان کم و فضای اشغال شده کم می باشد. مزایای معماری پیشنهادی ، گلوگاه های پردازشی مانند سرعت وپردازش های صنعتی بلادرنگ را برطرف می نماید و استفاده آن را در صنایع و پزشکی امکان پذیر می سازد .

واژه های کلیدی

پردازش تصویر، افزایش وضوح، کاهش نویز ، پیکربندی جزیی، پیاده سازی سخت افزاری، VHDL، FPGA.

۱. کارشناس ارشد برق،دانشگاه شهید بهشتی a.chekini@mail.sbu.ac.ir

۲. دانشکده مهندسی برق و کامپیوتر دانشگاه شهید بهشتی

3.Dynamic Partial Reconfiguration

تاریخ دریافت:۸۹/۱۰/۵ تاریخ پذیرش:۸۹/۱۱/۱۰

مقدمه

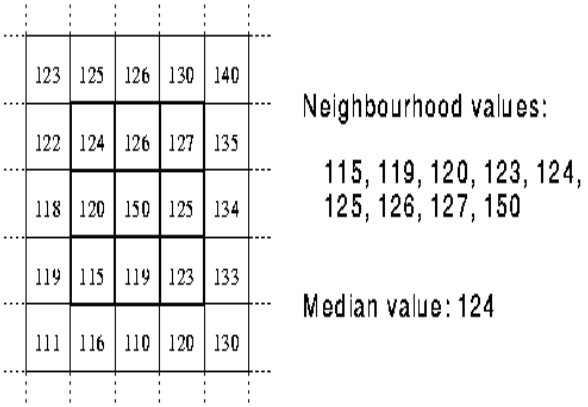
در این مقاله دو فیلتر با استفاده از معماری پیشنهادی برای پیاده سازی سخت افزاری انتخاب می شوند. بنابراین در ابتدا به بررسی این دو فیلتر می پردازیم . فیلتر اول نوعی از نویز را که هنگام ارسال عکس تغییراتی را بر روی عکس ایجاد می کند برطرف می نماید.

معمولا در پیاده سازی های نوین امروزی عکس را رنگی در نظر می گیرند . عکس به صورت پیکسل هایی در نظر گرفته می شود که برای نمایش هر کدام احتیاج است که مختصات پیکسل مورد نظر و میزان شدت رنگهای قرمز، سبز و آبی پیکسل را بدانیم. شدت رنگها را از صفر تا ۲۵۵ در نظر می گیرند. هنگام ارسال عکس‌ها در سامانه‌های دیجیتال بدلیل وجود نویز در مسیر و سیم های ارتباطی با اختلال هایی رو به رو می شویم . معمولا در این حالات برخی از بیت ها صفر شده و برخی دیگر ۲۵۵ می گردند که این مسئله باعث ایجاد حالت برفک گونه بر روی عکس می شود.[۱] یک نمونه از این عکس ها را در شکل ۱مشاهده می نمایید .

شکل ۱- عکس با تاثیر نویز ذکر شده

همانطور که در شکل ۱ نشان داده شده است، بیت های ارسالی خاکستری یا سفید و یا مشکی(۰ یا ۲۵۵) شده اند. در این مقاله برای ارائه معماری پیشنهادی، عکس رنگی در نظر گرفته شده است، به این معنی که در هنگام ارسال این عکس ها ممکن است شدت یکی از سه رنگ ارسالی به حالت حداقل و یا حداکثر یعنی ۰ و یا ۲۵۵ برود که این مسئله باعث کاهش کیفیت عکس می شود . [۴] راه حل برطرف نمودن این مشکل استفاده از فیلتر میانه گیر است. در این فیلتر ابتدا پس از فرایند

ورود عکس به برنامه و ذخیره آن در حافظه داخلی پیکسل اول عکس انتخاب می‌شود و پیکسل های اطراف آن را گرفته شده و به ترتیب چیده می شود. بر داشتن پیکسل های اطراف به صورت زیر است.



در این روش پیکسل ها به صورت ۳×۳ انتخاب شده و در گروه ۹ تایی مرتب می شوند. پس از این مرحله که میانه گیری است مقدار میانه را به جای پیکسل مورد نظر قرار می دهیم البته این تنها برای یکی از سه رنگ (RGB) موجود در تصویر مربوطه است. تعداد پیکسل ها انتخاب شده اطراف پیکسل مورد نظر را افزایش می دهیم. (به عنوان مثال مقدار ۷×۷ را درنظر می گیریم). این کار باعث از بین رفتن لبه های عکس می شود، به همین دلیل به نظر می رسد که بهترین انتخاب همان ۳×۳ باشد.[۲] شکل ۲ عکس هایی را که حاکی از تفاوت این دو حالت است نشان می دهد .

میانه گیری ۷×۷ میانه گیری ۳×۳

شکل ۲- تفاوت انتخاب پیکسل ها به صورت۳×۳ یا ۷×۷

با استفاده از این روش پیکسل هایی که مقدار آنها بسیار بیش از میانه باشد از بین می روند و به همین دلیل پیکسل هایی که مقدار آنها صفر و یا ۲۵۵ می باشد از تصویر حذف می شوند و همین مسئله باعث افزایش کیفیت تصویر می شود.[۳] اما از طرفی باید به این نکته توجه داشت که اگر تعداد پیکسل هایی را که از آنها میانه می گیریم تا مقدار میانه را به جای پیکسل مورد نظر قراردهیم زیاد شود آنگاه تصویر مات می گردد. تنها فرقی که در تصویر رنگی با آن مواجه هستیم آن است که این تصاویر دارای ۳ رنگ متفاوت هستند، که شدت هرکدام از ۰ تا ۲۵۵ قابل تغییر است. زمانی که می خواهیم آنها را بررسی کنیم می بایست هرکدام از رنگها را به صورت مجزا بررسی نماییم. پس در حافظه هر پیکسل ۳ بایت برای شدت رنگ اشغال می شود و لذا می بایست میانه گیری هر کدام از قسمت های قرمز، آبی و سبز یک پیکسل معین با رنگ مشابه پیکسل های مجاور انجام شود. مثلا میانه گیری برای رنگ آبی پیکسل(۲و۲) باید با قسمت آبی پیکسل های همسایه انجام شود.

برای آشنایی با دومین فیلتر استفاده شده برای ارائه معماری پیشنهادی در این مقاله از فیلتری برای افزایش وضوح و کیفیت یک تصویر استفاده می نماییم. معمولا تصاویری از بدن انسان در پزشکی، که از وضوح کافی برخوردار نیستند و یا تصاویری که مثلا در اثر حرکت دوربین، کیفیت و وضوح نامناسبی را بدست می آورند می توانند به عنوان تصویر ورودی این فیلتر لحاظ شوند. در واقع هدف از بهبود کیفیت ظاهری تصویر، آشکارسازی بعضی از جزئیات تصویر برای مشاهده کننده ی انسانی یا ماشین می باشد که ممکن است در اثر عدم نورپردازی صحیح و یا مناسب نبودن ابزار عکسبرداری به اندازه کافی وضوح نداشته باشد. به منظور برآورده شدن این هدف از فیلترهای تصویر[1] مختلفی می توان استفاده نمود. برای مثال می توان blur filter , Sharpen filter , edge finder filter , mean filter , emboss filter و... را نام برد.[۱۲][۹]

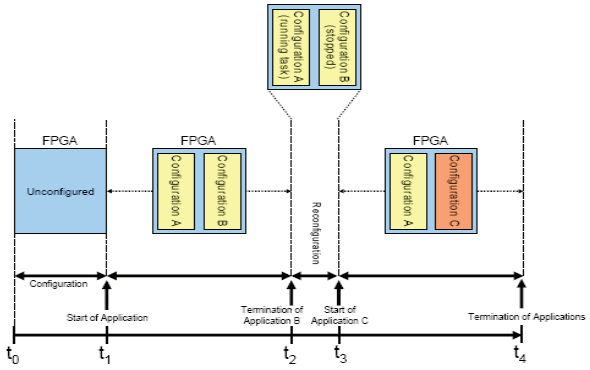
در این مقاله چند فیلتر مختلف، از جمله Edge Finder filter , Sharpen filter , Blur filter را انتخاب نموده و نتایج حاصل از شبیه سازی آنها را در قسمت چهار بیان می نماییم .[۱۰] تنها تفاوتی که در کد VHDL آنها پدید می آید، در تعریف مقادیر درآیه های ماتریس هر فیلتر است. دراین جا مانند حالت قبل ماتریس فیلتر را ۳ × ۳ در نظر می گیریم، که این مقدار برای انواع مختلف فیلترها قابل تغییر است. [۱۱][۷] لازم به ذکر است سایر قسمت های کد VHDL برنامه با اعمال هر کدام از فیلترهای فوق یکسان است. در ادامه مقاله درقسمت ۲ ، به بیان فن پیکربندی جزیی پرداخته و سپس در قسمت ۳ معماری پیشنهادی را ارائه می نماییم ،در قسمت ۴ الگوریتم بیان شده

1.Image filter

را به زبان VHDL نوشته وآن را با ابزار سنتز و شبیه ساز های رایج برای زبان های توصیف سخت افزار شبیه سازی نموده و به صورت عملی آن را روی سخت افزار پیاده سازی می نماییم و نتایج حاصل را بیان می کنیم . برای مشاهده کارآیی این معماری نرم افزار ISE شبیه ساز و سنتزکننده کدهایVHDL ، از شرکت Xilinx را استفاده می نماییم . نهایتا در پایان در قسمت پایانی نتایج معماری نوین پیشنهادی را ارائه می نماییم .

تکنیک پیکربندی جزیی

یکی از بزرگترین معایب راه حل های سخت افزاری برای استفاده آنها به جای سیستم های نرم افزاری ، عدم انعطاف پذیری آن هاست هرچند که سایر پارامترهای مهم نظیر قابلیت اعتماد و سرعت عملکرد آنها در طراحی بسیار جذاب هستند . یکی از راه های نوینی که برای مرتفع نمودن این مشکل استفاده می شود پیکربندی جزیی پویا[1] می باشد . این تکنیک سخت افزارهایی با خاصیت پیکربندی مجدد مانند FPGA ها را قادر می سازد ، در هنگام اجرا و پردازش ساختار داخلی خود را اصلاح نموده و به ساختاری با عملکردی متفاوت تبدیل شوند . به این ترتیب عدم انعطاف پذیری سخت افزارها بدون خاموش نمودن آن ها و پیکربندی مجدد آن ها در حال پردازش بلادرنگ اطلاعات رفع شده است . با این تکنیک رویکرد به سوی پیاده سازی سخت افزاری توابعی که تا به حال به دلیل محدودیت در انعطاف پذیری ، پیاده سازی سخت افزاری نمی شدند فراگیر شده است . پیکربندی جزیی فرآیند پیکربندی قسمتی از FPGA ، درحالی که سایر قسمت ها در حال اجرا هستند می باشد . این قابلیت در شکل ۳ نشان داده شده است .



شکل ۳ : مفهوم عملکرد تکنیک پیکربندی جزیی مرجع[۱]

1.Dynamic Partial Reconfiguration

انواع روش های پیکربندی جزیی سخت افزارهایی با قابلیت پیکربندی مجدد عبارتند از: پیکربندی جزیی ایستا[1] (SPR) و پیکربندی جزیی پویا[2] (DPR)

در نوع اول پیکربندی، قطعه در حال فرآیند پیکربندی فعال نیست . کل سیستم از کار افتاده تا پیکربندی جدید بر روی سخت افزار پیاده شود ، سپس مدار روشن شده و به این ترتیب عملکرد مدار تغییر می نماید . پیکربندی SPR ساده ترین و رایجترین راه برای پیاده سازی برنامه های کاربردی با سخت افزارهای قابل پیکربندی مجدد می باشد . اما در نوع دوم قسمتی از سخت افزار تغییر نموده در حالی که سایر قسمت های سخت افزار در حال اجرا هستند . در این روش باید الگوریتم را به قسمت های مجزایی که از نظر زمانی با هم تداخلی ندارند تقسیم نمود و سپس ترتیب و زمان استفاده هر کدام از قسمت ها را مدیریت نمود ، در این صورت عمده ترین مشکلاتی که با آن ها رو به رو هستیم عبارتند از:زمان مورد نیاز پیکربندی مجدد، پیچیدگی در طراحی، افزایش هزینه سخت افزار و افزایش زمان طراحی ، اما با وجود این محدودیت ها این روش دارای مزایایی نظیر کاهش مصرف توان ، کاهش فضای اشغالی ، انعطاف پذیری ، استفاده مجدد از سخت افزار[3] و نیز علاوه بر مزایای پیکربندی SPR می توان مصالحه ای را بین زمان و فضای اشغالی انجام داد. به طور کلی در هنگام طراحی باید طرح را به دو قسمت تقسیم نمود . یک قسمت که در کل اجرای برنامه بر روی سخت افزار بدون تعییر باقی می ماند . این قسمت به نام SR[4] معروف است . قسمت ایستا باید شامل کنترل کننده پیکربندی و واسط هایی برای ارتباط با قسمت پویا باشد . تمامی ورودی ها و خروجی ها به این قسمت متصل می شوند . ناحیه ایستا به وسیله یک واسط ثابت با ماژول های قسمت پویا در ارتباط می باشد .

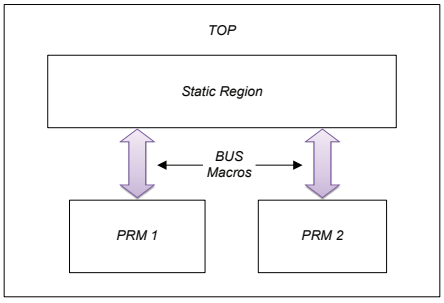
اما قسمت پویا در هنگام اجرای الگوریتم با توجه به فرمان قسمت کنترلی تغییر می نماید. معمولا در طراحی بخش هایی از طرح که با هم به صورت موازی اجرا نمی شوند را در این قسمت قرار می دهند. معمولا این ماژول ها با آمدن فرمان مناسب از قسمت کنترلی از حافظه ای در خارج از FPGA به صورت رشته بیت هایی بر روی آن بارگذاری می شوند . تاکنون انواع روش های پیکربندی سخت افزارهایی با قابلیت پیکربندی مجدد بررسی شدند و در پایان به این نتیجه رسیدیم که روش پیکربندی جزیی پویا دارای مزایایی است که می توان از آن برای پیکربندی سخت افزارها استفاده نمود . اکنون به بررسی انواع روندهای موجود برای روش پیکربندی جزیی پویا می پردازیم .

1.Static Partial Reconfiguration(SPR)
2.Dynamic Partial Reconfiguration(DPR)
3.Hardware Reuse
4.Static Region

به طور کلی پیکربندی جزیی پویا به دو صورت انجام می شود که در زیر آن ها را به صورت جداگانه بررسی می نماییم .

پیکربندی جزیی پویای مبتنی بر ماژول[1]

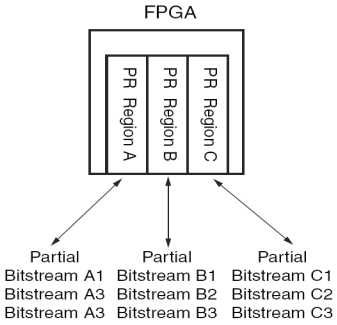
در این روش می توان هر ماژول را به صورت جداگانه طراحی نمود و برای هرکدام از ماژول هایی که می خواهیم در قسمت ایستا قرار بگیرند به صورت جداگانه رشته بیت هایی را ایجاد می نماییم و هر رشته بیت را بر روی حافظه خارجی قرار داده و در موقع مناسب از آن ها استفاده می نماییم . برای اطمینان از ارتباط صحیح بین قسمت ایستا با قسمت پویا از واسط هایی به نام BUS MACRO ها استفاده می نماییم . در هنگام طراحی به این نکته کلیدی توجه می نماییم که باید تمامی ماژول های پویای ما دارای ورودی و خروجی های یکسان و با نام های مشابهی باشند تا در ارتباط با قسمت ایستا دچار مشکل نشویم . شکل ۴ این روش را نشان می دهد. همانطوری که در این شکل ملاحظه می نمایید می توان از چندین PRM[2] استفاده نمود . مزایا و معایب این روش پیاده سازی و طراحی نظام مند ، طرح های بزرگ ، طراحی ماژولار، هزینه ذخیره سازی زیاد ، تاخیر بالا در پیکربندی مجدد و اطمینان ازعدم پیکربندی مجدد بافرهای سه حالته می باشند .



شکل ۴ : پیکربندی جزیی پویای مبتنی بر ماژول

می توان در طراحی به این روش از چندین قسمت پویا استفاده نمود به طوری که بر روی هر قسمت پویا چندین ماژول پویا را پیکربندی نمود . این مسئله در شکل ۵ نشان داده شده است .

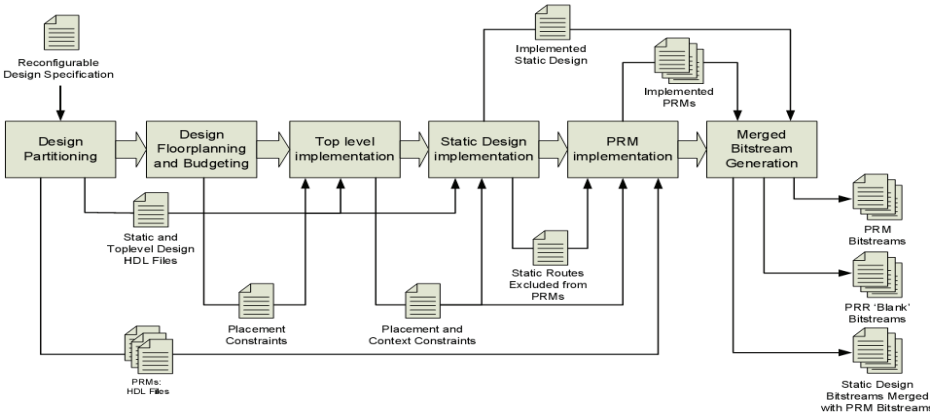
1.Module-Based Partial Reconfiguration
2.Partial Reconfiguration Module



شکل ۵ : استفاده از چندین PRR و PRM

پیکربندی جزیی پویای مبتنی بر تفاوت[1]

این روش برای زمانی مناسب است که تغییرات اندکی بین طرح اولیه و ماژول بعدی که می خواهد در ناحیه پویا به جای ماژول اولی قرار گیرد باشد . در این روش رشته های بیتی فقط شامل اطلاعات تفاوت بین ساختار کنونی سخت افزار و ساختار جدید سخت افزار می باشند . این روش برای طرح هایی که بسیار بزرگ و عملکرد قسمت پویای مشابهی دارند بسیار مناسب می باشد . به این ترتیب زمان پیکربندی مجدد در سیستم های بلادرنگ به شدت کاهش می یابد . باید توجه نمود که این دو روش هر دو دارای آرایش۲ یکسانی هستند . استفاده از هر دو روش در نهایت منجربه تولید رشته بیت هایی می شود که برای پیکربندی جزیی بر روی FPGA به کاربرده می شوند . روند طراحی به کمک این تکنیک در شکل ۶ نشان داده شده است .در ابتدا ابزار بسیار محدودی برای پیاده سازی این تکنیک وجود داشت اما در حال حاضر نرم افزار PLAN Ahead از شرکت Xilinx به صورت گرافیکی ابزاری را برای پیاده سازی تکنیک پیکربندی جزیی ارائه می نماید .



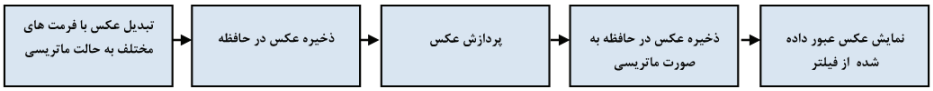
شکل ۶ : روند طراحی پیکربندی جزیی

1.Difference-Based Partial Reconfiguration
2. Layout

برای آشنایی بیشتر با عملکرد این نرم افزار می توان به منبع [۵] مراجعه نمود. در حال حاضر پیکربندی جزیی در همه FPGA ها پشتیبانی نمی شود . با بررسی های صورت گرفته برای مثال FPGA های Spartan [3], Virtex II, Virtex II Pro و Virtex [4] از شرکت Xilinx قابلیت پشتیبانی از این خاصیت را دارند . برای آشنایی بیشتر با روند این تکنیک می توانید به مرجع [۶] مراجعه نمایید.

ارائه معماری پیشنهادی

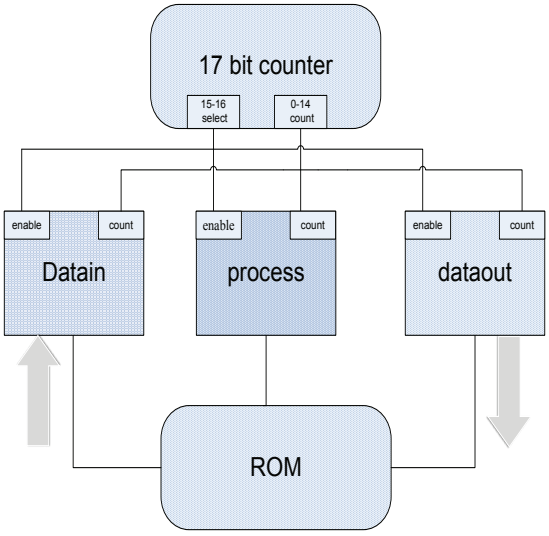
در این قسمت به ارائه معماری پیشنهادی می پردازیم. همان طور که قبلا ذکر شد برای پیاده سازی معماری پیشنهادی و به منظور بیان چگونگی عملکرد معماری ارائه شده ، نتایج با چند فیلتر مختلف تست شده ارائه می شود. در ابتدا فیلترهای مختلف ارائه شده در قسمت ۱ را به صورت جداگانه طراحی و سپس با زبان توصیف سخت افزاری VHDL به نوشتن کد آنها می پردازیم . برای طراحی فیلتر اول معمولا میانه گیری را برای تک تک پیکسل ها انجام داده و به همین دلیل است که تصویر کمی مات می گردد برای بهبود این وضعیت تغییری در فیلتر ایجاد می نماییم . باید توجه داشته باشیم که در این نویز تنها مقدار شدت رنگ پیکسل ها صفر یا ۲۵۵می گردد، لذا میانه گیری برای پیکسل هایی که مقدار آنها بین این دو عدد است کاری بیهوده می باشد. پس در این فیلتر ابتدا بررسی می کنیم که آیا شدت رنگ ۰ یا ۲۵۵ می باشد و اگر این گونه بود آنگاه مقدار میانه را جایگزین میزان شدت رنگ می کنیم. برای پیاده سازی این روش ابتدا می بایست عکس را به فرمتی که در قسمت ۱ به آن اشاره شد تبدیل نماییم.[۸]



شکل ۷ : بلوک دیاگرام روند کار برای فیلتر اول

همانطور که در بلوک دیاگرام شکل ۷ مشاهده می شود، معمولا عکس ها با فرمت های مختلفی مثل JPEG در حافظه کامپیوتر ذخیره می شوند. ابتدا می بایست از این فرمت ها حالت ماتریسی مورد نظر خود را که یک ماتریس ۳×۱۰۰×۱۰۰است استخراج نماییم. برای این منظور با استفاده از نرم افزار MATLAB عکس با هر فرمتی را دریافت و آن را به حالت ماتریسی تبدیل می کنیم. نرم افزار MATLAB فایلی به صورت text را در خروجی به ما می دهد که این فایل بعنوان ورودی کدهای نوشته شده به زبان توصیف سخت افزار VHDL در نظر گرفته می شود. سپس تصویر مورد نظر

در حافظه ذخیره شده و در نهایت پیکسل به پیکسل بررسی می کنیم که آیا شدت هر کدام از رنگ ها حداقل (صفر) و یا حد اکثر (۲۵۵) می باشد یا خیر. در صورت صفر یا ۲۵۵ بودن الگوریتم میانه گیری را اعمال می کنیم که به صورت حبابی عمل می نماید. پس از انجام پردازش پیکسل ها به صورت بایت به بایت ذخیره شده ، سپس برنامه فایلی خروجی به صورت text بوجود می آورد که می توان آن را به عنوان ورودی به نرم افزار MATLAB وارد نموده و شکل تغییر یافته را بدست آورد. اما برای توضیح نحوه عملکرد این فیلتر می دانیم که تصویر فرض شده مورد نظر فوق با ماتریس (۳, ۱۰۰ ,۱۰۰) نشان داده می شود . این بدان معنی است که می بایست ۳۰۰۰۰ عدد را بررسی کنیم. برای این منظور از مجموعه ای از بلوک ها برای انجام عملیات پردازش اطلاعات استفاده می نماییم . شکل۸ این بلوک دیاگرام را نشان می دهد .



شکل ۸ : بلوک دیاگرام برای انجام عملیات پردازش اطلاعات در فیلتر اول

بلوک دیاگرام نشان داده شده در شکل ۸ از یک شمارنده ۱۷ بیتی تشکیل شده است که از بیت صفر تا ۱۴ آن برای ذخیره بیت ورودی از infile.txt در حافظه یا فراخوانی عدد مورد نظر از حافظه و یا بوجود آوردن فایل outfile.txt استفاده می شود. همانطور که می دانیم با ۱۵ بیت امکان شمردن از ۰ تا ۳۲۷۶۷ برای ما امکان پذیر است . این مقدار برای شمردن ۳۰۰۰۰ بایت اطلاعات تصویر کافی است. دو بیت آخر آن نیز در طراحی برای فعال کردن component های (۰۰) datain (۱۰) dataout و یا (۰۱) process استفاده می شود. در ابتدا شمارنده صفر می شود، صفر بودن

تمام بیت ها بدان معنی است که datain فعال می باشد. با فعال شدن datain اعداد ازصفرتا ۲۹۹۹۹ از infile.txt که توسط MATLAB بوجود آمده است در حافظه(romp) ذخیره می شود. پس در این مرحله تنها خواندن اطلاعات از حافظه وذخیره آن را در حافظه خواهیم داشت . به محض آنکه شمارنده به عدد ۲۹۹۹۹ رسید متوجه می شویم که تمام تصویر۳×۱۰۰×۱۰۰ ذخیره شده است. پس از این مرحله process فعال می گردد. یعنی بیت ۱۵ و ۱۶ شمارنده ۰۱ می گردد و عدد ۱۰۱ (در مبنای ۱۰) به صورت hex روی شمارنده قرار می گیرد. لازم به ذکر است که در این نوشتن کد VHDL عدد ۰ را به رنگ قرمز عدد۱ را به رنگ سبز و عدد ۲ را به رنگ آبی اختصاص داده ایم. همچنین دقت داشته باشید که به هنگام پردازش تصویر احتیاجی به پیکسل های حاشیه ای تصویر نداریم لذا از پیکسل ۱۰۱ که اول پیکسل غیر حاشیه ای است شروع می کنیم حال اگر عدد ۱۰۱ را در ۳ ضرب کرده و با عدد رنگ مورد نظر جمع کنیم شدت همان رنگ را در مکان مورد نظر به ما می دهد. مثلا اگر شدت رنگ آبی را در پیکسل۲۱۵ بخواهیم داریم:

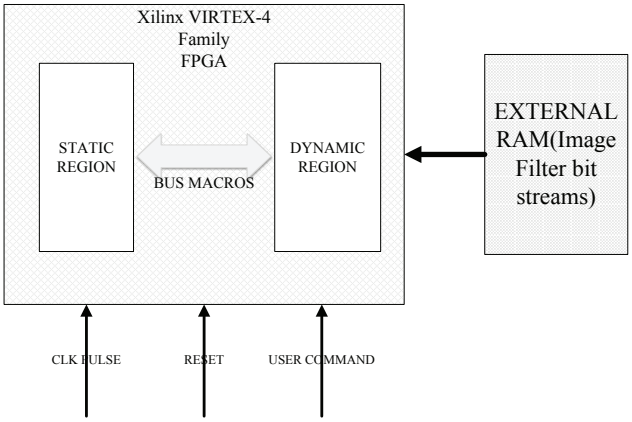
$$(۲۱۵\times۳)+۲=۹۴۷$$

بنابراین اگر شماره ۹۴۷ را از حافظه بخوانیم شدت رنگ آبی را نشان می دهد. در این مرحله شمارنده از ۱۰۱ در مبنای ده تا ۹۸۹۸ عمل شمارش را انجام داده (پیکسل های غیر حاشیه ای) و در هر مرحله بررسی می کند که آیا عدد شدت رنگ ۰ یا ۲۵۵ هست یا خیر اگر بود میانه گیری را انجام می دهد و در غیر این صورت به پیکسل بعدی می رود. در این فیلتر، میانه گیری با استفاده از الگوریتم حبابی نوشته شده است و نحوه فراخوانی شدت هر کدام از رنگ ها در هر پیکسل نیز مانند فراخوانی هر کدام از پیکسل هاست و میانه گیری به صورت ۳×۳ انجام شده است. پس از این مرحله تصویر تغییر یافته در حافظه ذخیره می شود و سپس مقدار ۱۰ hex در بیت ۱۵ و ۱۶ شمارنده قرار می گیرد که به معنی فعال شدن component dataout است. در این مرحله شمارنده از ۰ تا ۲۹۹۹۹ می شمرد و فایل dataout.txt را بوجود می آورد حال اگر این فایل را به عنوان ورودی به نرم افزار MATLAB وارد نماییم . در پیاده سازی این فیلتر برای تبدیل تصویر به ماتریسی از اعداد برای ورود به زبان VHDL از ساختار زیر استفاده می نماییم :

```
clc
clear
a = imread('image.bmp');
count = 1;
b = zeros(1,30000);
for i = 1:100
for j = 1:100
for k = 1:3
b(count)= a(i,j,k);
count = count + 1;
end
end
end
dlmwrite('infile.txt',b,'\n');
```

در ماژولTop ، Component های مختلف تعریف شده اند. برای افزایش کارآیی در زمان پردازش اطلاعات و عیب یابی و ساخت نمونه اولیه ، کلیه کدها را به صورت ماژولار می نویسیم. در ماژول Counter نحوه عملکرد شمارنده توصیف شده است که در بالا به شرح چگونگی عملکرد آن پرداختیم. ماژول Proc وظیفه پردازش تصویر را انجام می دهد و عملا قلب سامانه دیجیتال ما خواهد بود . ماژول datain برای ذخیره سازی اطلاعات در حافظه و ماژول dataout تصویر فیلترشده را به صورت فایل txt در آورده که امکان نمایش آن در MATLAB وجود دارد. نهایتا ماژول rom نشان دهنده حافظه ای برای ذخیره اطلاعات می باشد . لازم به ذکر است در پیاده سازی فیلتر اول برای دستیابی به حافظه از فن دسترسی shared memory استفاده نموده ایم . اما برای فیلتر دوم، معماری ارائه شده از سه بلوک اصلی که عبارتند از خواندن تصویرو تبدیل عکس به فایل متن توسط برنامه ی read_im.m، برنامه ی VHDL که فقط دارای یک ماژول با نام filter.vhd است و بلوک خواندن فایل متنی و بازسازی دوباره ی عکس توسط برنامه ی write_im.m، تشکیل شده است. در این حالت تصویر مورد استفاده یک عکس ۲۴۰×۳۲۰ است. این تصویر توسط دستور imread('image.jpg') به یک ماتریس ۳×۳۲۰×۲۴۰ تبدیل شده و این ماتریس نیز توسط سه حلقه‌ی for ساده به یک ماتریس ستونی یک بعدی که ترتیب درایه‌های آن red، green، blue،

تبدیل می شود . سپس این ماتریس ستونی نیز توسط دستور dlmwrite در داخل یک فایل متنی نوشته می شود . لازم به ذکر است که دقیقا عکس این عمل نیز در فایل write_im انجام می شود . با اجرای write_im هر دو عکس فیلتر نشده و فیلتر شده بصورت دو subplot که در زیر هم قرار دارند نمایان خواهد شد. در این جا با استفاده از نرم افزار Plan ahead می توانیم کل سخت افزار را به دو قسمت تقسیم نماییم . همانطوری که در شکل ۹ نشان داده شده است قسمت های مشترک الگوریتم ها را در قسمت ایستا و قسمت هایی که فیلترها با هم متفاوت هستند را در قسمت پویا قرار می دهیم . با این کار رشته بیت های تولیدی برای هر ماژول که بیان کننده یک فیلتر هستند در حافظه خارجی قرار گرفته و هر گاه کاربر بخواهد ، سخت افزار به فیلتر مورد نظر تبدیل می شود . حسن این کار تبدیل بلادرنگ سخت افزار می باشد . همانطور که در شکل ۹ ملاحظه می شود ، دو قسمت پویا و ایستا با استفاده از واسط های مشخصی به نام Bus Macro ها با یکدیگر در ارتباط هستند . معماری پیشنهادی مانند هر سخت افزاری دارای پایه های RESET و CLK می باشد که به وسیله ان ها می توان سخت افزار را کنترل نمود .

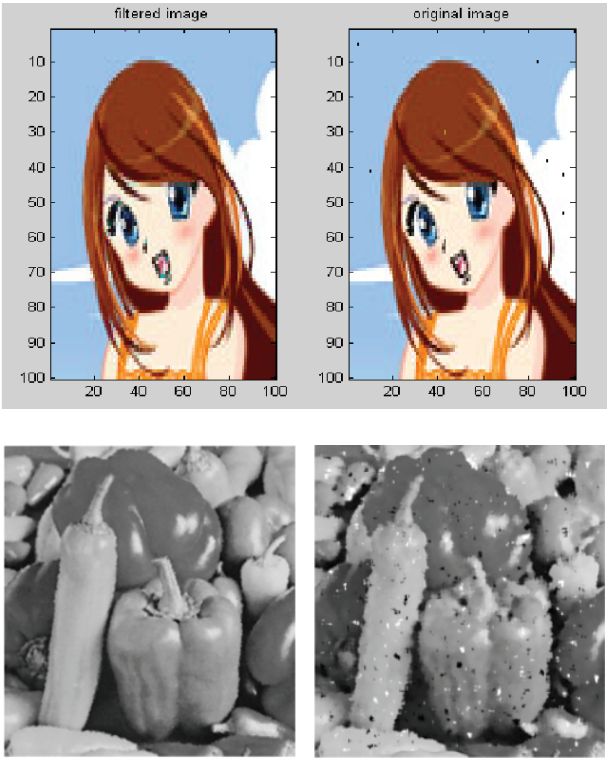


شکل ۹ : بلوک دیاگرام معماری پیشنهادی

شبیه سازی و پیاده سازی معماری پیشنهادی

به منظور نشان دادن نتایج معماری پیشنهادی ابتدا از شبیه سازی استفاده نموده و سپس به سنتز کدهای VHDL و پیاده سازی بر روی FPGA می پردازیم . در این قسمت ابتدا برای شبیه سازی فیلتر اول تصویری را به عنوان ورودی به MATLAB می دهیم . این نرم افزار فایل infile.txt را برای ما بوجود می آورد. این فایل را به برنامه نوشته شده به زبان VHDL وارد نموده و پس از اجرای

آن outfile.txt را در خروجی تولید می نماید. این فایل را با استفاده از MATLAB دوباره به تصویر تبدیل می نماییم . لازم به ذکر است که نویز را در این تصاویر با استفاده از نرم افزار Paint به آن وارد نموده ایم . نتایج حاصل از شبیه سازی در شکل ۱۰ نشان داده شده اند.



شکل ۱۰ : نتایج شبیه سازی فیلتر اول برای دو عکس ورودی متفاوت

به دلیل این مسئله که فایل تصویر حاوی اطلاعات زیادی است و به طریق دیگری نمی توان حجم بالایی از داده را به شبیه ساز VHDL وارد و یا خارج کرد از TextIO[1] استفاده می نماییم . Textio یکی از بسته های از پیش تعریف شده ی VHDL است. بسته ی TextIO توابع و روال هایی دارد که با استفاده از آنها می توان فایل های متنی قالب بندی شده را خواند و یا در آنها نوشت. TextIO فایل های متنی را به صورت فایل هایی از خطوط تلقی می کندکه هر خط رشته ایست که توسط یک کاراکتر بازگشت به ابتدای خط خاتمه یافته است. روال هایی برای خواندن و نوشتن یک خط کد و

1.Text Input and Output

توابعی وجود دارند که پایان فایل را بررسی می کنند. بسته ی TextIO نیز چند نوع را تعریف می کند که هنگام پردازش فایل های متنی استفاده می شوند. نوع line در بسته ی TextIO اعلان شده و برای نگهداری یک خط جهت نوشتن در یک فایل ، یا خطی که از فایل خوانده شده است استفاده می شود. ساختار خط واحد مبنایی است که عملیات های TextIO براساس آن انجام می شوند. برای مثال هنگام خواندن از یک فایل ، نخستین مرحله خواندن یک خط از فایل در ساختاری از نوع line است، سپس ساختار خط فیلد به فیلد پردازش می شود. اما برای نوشتن یک فایل ، ابتدا ساختار خط فیلد به فیلد در یک ساختار داده ی خط موقتی ساخته می شود، سپس خط در فایل نوشته می شود.

در روال بلوک نوشته شده به زبان VHDLدر داخل پهنای تصویر یعنی تعداد ستونهای آن که برابر با ۲۴۰ است را قرار داده ایم و در ثابت دیگری ، ارتفاع تصویر یعنی تعداد سطرهای آن را که برابر با ۳۲۰ است ، قرار داده ایم. سپس تصویر به صورت یک آرایه تعریف می نماییم . پس از آن ماتریس فیلتر از نوع آرایه تعریف شده است، ماتریس فیلتر را ۳× ۳ در نظر گرفته و قابل ذکر است که این مقدار برای انواع مختلف فیلترها قابل تغییر است. Process ای که در این برنامه استفاده شده است با تغییر حالت و آمدن لبه بالا رونده پالس ساعت شروع به کار می کند. حال با توجه به نوع فیلتری که برای افزایش وضوح و کیفیت تصویر انتخاب می کنیم، درآیه های ماتریس فیلتر می تواند متفاوت باشد که این درآیه ها را به صورت سطری در متغیر دیگری به نام filtmat قرار می دهیم .

[X11 X12 X13]

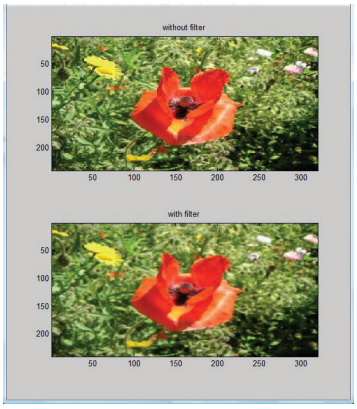
filtmat = [X21 X22 X23]

[X31 X32 X33]

[X11, X12, X13, X21, X22, X23, X31, X32, X33]

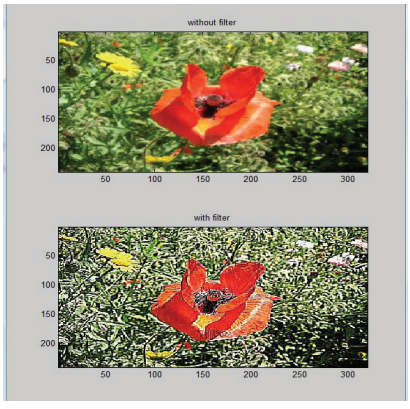
بعد از خواندن فایل ها باید فیلتر مورد نظر خود را اعمال نماییم.برای این کار با استفاده از سه‌حلقه‌ی for که یکی از آنها برای سطرهای تصویر ، یکی برای ستون های آن و دیگری نیز به خاطر استفاده از تصویر رنگی و اینکه تصاویر رنگی از سه مؤلفه رنگ اصلی قرمز ، سبز و آبی تشکیل شده اند می باشد، تصویر اولیه را فیلتر می کنیم. در برنامه نوشته شده به زبان VHDL هر پیکسل که از تصویر ورودی خوانده شود در یک متغیر و هر پیکسل از تصویر خروجی که پس از اعمال فیلتر بر تصویر ورودی ساخته می شود در متغیری دیگر قرار می گیرد. فیلتر کردن تصویر در واقع به این‌صورت انجام

می‌شود که هر پیکسل توسط یک ماتریس ضرایب در پیکسل‌های همسایه‌ی خود ضرب می‌شود. هر نوع فیلتر علاوه بر ماتریس ضرایب، دارای دو پارامتر دیگر factor و bias است که factor در هر پیکسل در انتهای کار ضرب می‌شود و bias نیز با هر پیکسل جمع می‌شود و بعد از آن فیلترهایی که دارای مقدار کمتر از ۰ یا بیشتر از ۲۵۵ هستند به ترتیب به ۰ و ۲۵۵ مقداردهی شده‌اند. در نهایت باید فایل متنی تولید شده مربوط به تصویر خروجی را که بر اثر اعمال فیلتر بر تصویر ورودی و انجام پردازش های مختلف روی آن، ایجاد شده است را در MATLAB نوشته و در نتیجه تصویر خروجی را به دست می آوریم . خروجی حاصل از اعمال چند نمونه از فیلترهایی که در قسمت ۱ به آن ها اشاره شد در شکل های ۱۱ و۱۲ و ۱۳ ملاحظه می شوند .

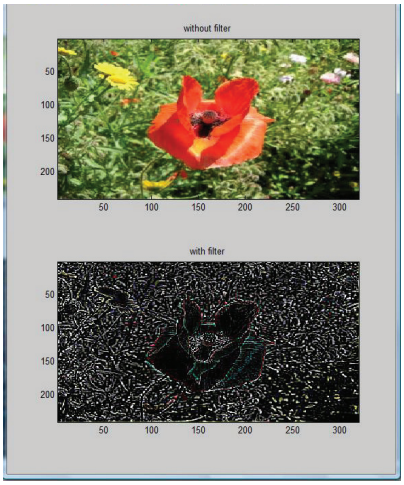


شکل ۱۱ - نتایج شبیه سازی با اعمال فیلتر blur filter

همان‌طوری‌که در شکل ۱۱ ملاحظه می شود وضوح و روشنایی تصویر بهتر شده است.



شکل ۱۲- نتایج شبیه سازی با اعمال فیلتر Sharp filter



شکل ۱۳- نتایج شبیه سازی با اعمال فیلتر edge finder

تنها ورودی ای که در برنامه VHDL برای شبیه سازی لحاظ شده است، پالس ساعت است که با تغییر حالت آن و در واقع با رخ دادن لبه ی بالا رونده برنامه شروع می شود. اکنون پس از مشاهده نتایج مطلوب در شبیه سازی به دنبال پیاده سازی برنامه نوشته شده بر روی FPGA هستیم . کد VHDL معماری پیشنهادی را به صورت قابل سنتز نوشته و همانطور که در قسمت ۲ اشاره شد برای پیاده سازی فن پیکربندی جزیی از نرم افزار ISE و نرم افزار Plan Ahead استفاده می نماییم . در ادامه برای درک بهتر و عملی تر طرح بیان شده در قسمت ۳ را با زبان استاندارد VHDL نوشته و نتایج حاصل از شبیه سازی و سنتز آن را ارائه می نماییم. برای سنتز از FPGA های خانواده Virtex–4 از شرکت Xilinx استفاده نموده ایم .

نتایج سنتز و بررسی توان مصرفی اإپأ

در ابتدا تک تک ماژول ها را به صورت معمولی نوشته و سنتز می نماییم . سپس با استفاده از فن پیکربندی جزیی ماژول ها را باهم در آمیختیم و از ابتدا طرح را سنتز نمودیم . نتایج توان مصرفی در جدول ۱ نشان داده شده است .

جدول ۱ : میزان توان مصرفی برای معماری پیشنهادی بر روی خانواده virtex – 4

Virtex- 4 family Xilinx company	Static Power consumption (mW)	Dynamic Power consumption (mW)	Total Power consumption (mW)
Non- partial reconfiguration	720	230	950
Proposed partial reconfiguration architecture	380	150	530

همانطوری که در جدول ۱مشاهده می شود با استفاده از فن پیکربندی جزیی برای پیاده سازی فیلترهای دیجیتال بر روی سخت افزار FPGA به دلیل کاهش فضای اشغالی سخت افزار میزان توان مصرفی نیز کاهش می یابد . درمرجع [۱] [۳] تمام پیاده سازی ها بر روی نرم افزار صورت گرفته است که طبیعتا دارای سرعت پایین تری نسبت به پردازش های سخت افزاری است .

نتایج سنتز روی مساحت اشغال شده اإپأ

با سنتز و پیاده سازی سخت افزاری بر روی این خانواده از شرکت Xilinx نتایج قابل پیش بینی ملاحظه گردید. همانطوری که در جدول ۲ نشان داده شده است با استفاده از این فن میزان حجم مصرفی تراشه ها نیز کاهش می یابد، و این مسئله قابل پیش بینی است. با افزایش سطح اشغالی تراشه و استفاده از عناصر منطقی بیشتر، طبیعی است که برای اجرای یک الگوریتم نیازمند مصرف توان بیشتری می باشیم. نتایج سنتز و فضای اشغالی نشان می دهد که معماری پیشنهادی فضای کمی را از سخت افزارFPGA اشغال می نماید . این مسئله برای داشتن یک سخت افزار بر روی یک تراشه SOC[1] به منظور پردازش تصویر بسیار مطلوب است . جدول ۲ نشان می دهد که هنگام استفاده از فن پیکربندی جزیی برای داشتن تمامی فیلترهای بیان شده میزان استفاده از سخت افزار کاهش می یابد . در سایر مراجع ، کل مجموعه الگوریتم ها معمولا در خارج از سخت افزار مرکزی و بر روی حافظه جانبی قرار می گیرند و باعث افزایش مدت زمان پردازش اطلاعات و دسترسی به حافظه می شوند . از طرفی با قرار گیری مجموعه الگوریتم ها بر روی حافظه های خارجی امنیت سامانه نیز به شدت کاهش می یابد . به خاطر این واقعیت که کارآیی حافظه ها خیلی کمتر از پردازنده ها است ، زمان بین درخواست داده ای توسط پردازنده و لحظه معتبر شدن داده ها درخروجی حافظه نسبتا زیاد و طولانی است . از این رو در این معماری ها مدت زمان زیادی از پردازنده صرف همزمانی و دریافت اطلاعات از حافظه می شود. در حالی که در معماری پیشنهادی به خاطر استفاده از حافظه های داخلی FPGA سرعت پردازش به مراتب بالاتر از این معماری ها است . با مشاهده نتایج سنتز معماری پیشنهادی در نرم افزار ISE فرکانس کاری سخت افزار ۲۱۰ MHzمی باشد . با این‌معماری

1.System on Chip

عملا می توان از سرعت بالای پردازش تصاویر توسط سخت افزار FPGA استفاده نمود ، این در حالی است که به طور همزمان با استفاده از این معماری میزان انعطاف پذیری[1] سامانه که از گلوگاه های سامانه های سخت افزاری است را برطرف نموده ایم .

جدول ۲ : میزان منابع مصرفی FPGA

Resource types	Usual hardware implementation				Partial reconfiguration architecture (All filters)	
	Mean filter	edge finder	blur filter	Sharp filter	Static Utilized Resources	Dynamic Utilized Resources
Flip flop	182	150	170	166	193	107
Slices	195	184	215	202	198	115
LUTs	39	25	47	32	22	35

نتیجه گیری

در این مقاله معماری و تکنیکی جدید برای پیاده سازی الگوریتم های پردازش تصاویر دیجیتال بر روی سخت افزار FPGA ارائه شد. در ابتدای این مقاله به بیان تکنیک مورد نیاز برای ارائه معماری پیشنهادی پرداخته و به طور ساده با نحوه عملکرد این تکنیک جدید در پیاده سازی مدارهای سخت افزاری آشنا شدیم . سپس مجموعه ای از فیلترهای پردازش تصاویر دیجیتال ساده را بیان نموده و عملکرد آن با کدهایVHDL شبیه سازی نموده و نتایج به دست آمده را ارائه نمودیم. در این معماری با انتقال حافظه های خارجی که به صورت رایج در معماری فایروال های سخت افزاری استفاده می شوند، به داخل FPGA سرعت پردازش و مصرف توان آن را بهبود بخشیدیم . عملا این مقاله به ارائه معماری نوینی پرداخته که هرگاه کاربر بخواهد به صورت بلادرنگ فیلتر مربوطه را بر روی سخت افزار قرار می دهد. معماری پیشنهادی به اختصاردارای ویژگی های زیر می باشد : استفاده از حافظه های داخلی به جای استفاده از حافظه های خارجی، استفاده از تکنیک پیکربندی جزیی و پیاده سازی همزمان چندین فیلتر مختلف بر روی یک سخت افزار واحد با استفاده از نرم افزار Plan Ahead برای کاهش مصرف توان در معماری و افزایش سرعت پردازش. با مقایسه مختصری با

1. Flexibility

معماری های فعلی برخی از مزیت های معماری پیشنهادی را برشمردیم . با استفاده از این معماری می توان سامانه‌ی واحد بر روی یک تراشه با مصرف توان کم ، ابعاد کوچک و وزن کم در سامانه های پردازش تصاویر نظامی و پزشکی و صنعتی داشت .

مراجع

1. De S. Zhang, Donald J. Kouri, “Varying Weight Trimmed Mean Filter For The Restoration Of Impulse Noise Corrupted Images”, 0-7803-8874-7/05, ICASSP 2005, IEEE, 2005.

2. Xuming Zhang, Youlun Xiong, “Impulse Noise Removal Using Directional Difference Based Noise Detector and Adaptive Weighted Mean Filter”, IEEE Signal Processing Letters, Vol. 16, No. 4, April 2009.

3. A. Bovik, “Handbook of Image and Video Processing”, New York Academic, 2000.

4. Daniel Leo Lau, Juan Guillermo Gonzalez, “The Losest-To-Mean Filter: An Edge Preserving Smoother For Gaussian Environments”, 0-8186-7919-019, IEEE, 1997.

5. “PlanAhead User Guide”, www.xilinx.com, UG632 (v 11.3.1) September 16, 2009.

http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/PlanA-head_Tutorial_Reconfigurable_Processor_Peripheral.pdf .

6.“PlanAhead Software Tutorial Overview of the Partial Reconfiguration Flow”, www.xilinx.com, UG 743 (v 12.1) May 3, 2010 .

http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_1/PlanA

head_Tutorial_Overview_of_Partial_Reconfiguration_Flow.pdf .

7. Truong Nguyen, Gilbert Strang, “Wavelets and Filter Banks”, Wellesley College, 1996.

8. Xiong Liu, Alan N. Willson, “A 1Gsample/Sec Non-Recursive Sharpened Cascaded Integrator-Comb Filter with 70 dB alias rejection and 0.003 dB droop in 0.18-Jlm CMOS”, 978-1-4244-3870-9/09, IEEE, 2009.

9. A.G. Quareshi, M.M. Fahmy, “2-D Killman Filtering For The Restoration Of Stochastically Blurred 1maages”, H2561-9/88/0000-1024, IEEE, 1988.

10. Alan Y. Kwentus, Zhongnong Jiang, and Alan N. Willson, “Application of Filter Sharpening to Cascaded Integrator-Comb Decimation Filters”, IEEE Transactions On Signal Processing, Vol. 45, No. 2, February 1997.

11. J. F. Kaiser, R. W. Hamming, “Sharpening the Response of a Symmetric Non recursive Filter by Multiple Use of the Same Filter” IEEE TRANSACTIONS ON ACOUSTICS, SPEECH, AND SIGNAL PROCESSING, VOL. ASSP-25, NO. 5, OCTOBER 1977.

12. Shenghua Xu, Litao Han, Lihua Zhang, “An Algorithm to Edge Detection Based on SUSAN Filter and Embedded Confidence”, Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA’06) 0-7695-2528-8/06, IEEE, 2006.