
بسم الله الرحمن الرحیم

دانشکده مهندسی برق ورباتیک

پایان نامه دوره کارشناسی ارشد مهندسی برق- الکترونیک

تشخیص برخط تغییر نما و پیاده سازی بلادرنگ در بستر پردازشگرهای سیگنال

نگارش:

مصطفی صفایی

اساتید راهنما:

دکتر هادی گرایلو - دکتر امید معروضی

استاد مشاور:

دکتر علی سلیمانی

پایان نامه جهت اخذ درجه کارشناسی ارشد

بهمن ۹۱

دانشگاه صنعتی شاهرود

دانشکده برق و رباتیک

گروه الکترونیک

پایان نامه کارشناسی ارشد آقای مصطفی صفایی

تشخیص برخط تغییر نما و پیاده سازی بلادرنگ در بستر پردازشگرهای سیگنال

در تاریخ ..... توسط کمیته تخصصی زیر جهت اخذ مدرک کارشناسی ارشد مورد ارزیابی و با درجه ..... مورد پذیرش قرار گرفت.

اساتید راهنما	امضاء	اساتید مشاور	امضاء
نام و نام خانوادگي :		نام و نام خانوادگي :	
نام و نام خانوادگي :		نام و نام خانوادگي :	

اساتید داور	امضاء	نماینده تحصیلات تکمیلي	امضاء
نام و نام خانوادگي :		نام و نام خانوادگي :	
نام و نام خانوادگي :			
نام و نام خانوادگي :			
نام و نام خانوادگي :			

تقدیم به :

مادرم که با عاطفه سرشارش، روحم را از تنهایی و نومیدی رهایی می بخشد و پدرم که وجود پر مهرش در سردترین دوران، تسلی بخش جانهاست و همه کسانی که در راه علم آموزی مشوق من بوده اند و معلمانی که دانش اندکم را مدیون آنها هستم.

سپاس و قدردانی:

سپاس و حمد پروردگار را که توان اندیشیدن به ما عطا فرمود. اکنون که در این مقطع از تحصیلات به درجه کارشناسی ارشد نائل آمده‌ام بر خود لازم می‌دانم از کسانی که مرا در این مسیر یاری نموده اند تشکر و قدردانی نمایم. از معلمانی که مرا با الفبای علم آشنا نمودند و به من قدرت تفکر و تعمق آموختند. از اساتید دوره کارشناسی ارشد علی الخصوص جناب دکتر هادی گرایلو به پاس راهنماییهای بی‌دریغشان و دکتر امید معروضی و دکتر علی سلیمانی که به عنوان اساتید راهنما و مشاور همکاری نزدیکی با بنده داشته و مرا از نظرات سودمندشان بهره مند ساختند و همچنین از دوست عزیزم مهندس آیدین خداشناس که در این سالها از محبت بیدرغش بهره‌مند بودم، کمال تشکر و سپاس را دارم.

تعهد نامه

اینجانب مصطفي صفايي دانشجوی دوره کارشناسی ارشد رشته الکترونیک-دیجیتال دانشکده برق و رباتیک دانشگاه صنعتی شاهرود نویسنده پایان نامه تشخیص برخط تغییر نما و پیاده سازي بلادرنگ در بستر پردازشگرهاي سیگنال با راهنمائی جناب دکتر هادی گرایلو متعهد می شوم:

- تحقیقات در این پایان نامه 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 پایان نامه 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 می باشد 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 پایان نامه تأثیرگذار بوده اند در مقالات مستخرج از پایان نامه رعایت می گردد.
- در کلیه مراحل انجام این پایان نامه، در مواردی که از موجود زنده (یا بافتهای آنها) استفاده شده است ضوابط و اصول اخلاقی رعایت شده است.
- در کلیه مراحل انجام این پایان نامه، 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

امضای دانشجو

مالکیت نتایج و حق نشر

- کلیه حقوق معنوی این اثر و محصولات آن (مقالات مستخرج ، کتاب ، برنامه های رایانه ای ، نرم افزار ها و تجهیزات ساخته شده است) متعلق به دانشگاه صنعتی شاهرود می باشد . این مطلب باید به نحو مقتضی در تولیدات علمی مربوطه ذکر شود .
- استفاده از اطلاعات و نتایج موجود در پایان نامهبدون ذکر مرجع مجاز نمی باشد.

چکیده

امروزه، فایلهای ویدئویی دیجیتال که اطلاعات صوتی و تصویری را ترکیب می‌کنند، در همه جا قابل دسترس می‌باشند. پیشرفت سریع وسایل ویدئویی مانند دوربین‌های دیجیتال، زمینه های استفاده از ویدئو که شامل کنفرانسهای اینترنتی و تحصیلات الکترونیکی می شود را افزایش داده است. این شرایط، پایگاه داده‌ی عظیمی از فایلهای ویدئویی را ایجاد کرده است. لذا نیازمند سازماندهی این پایگاه داده و بازیابی اطلاعات مفید آن می‌باشیم. نیاز به سازماندهی پایگاه داده‌های ویدئویی سبب شده است که تشخیص تغییر نما به یکی از مهمترین موضوعات مورد مطالعه در زمینه پردازش ویدئو تبدیل گردد. لذا ارائه روشی برای تشخیص تغییر نما ضروری به نظر می‌رسد. در این پایان‌نامه، هدف طراحی وپیاده سازی روشی جهت تشخیص تغییر نما و نوع آن (تدریجی یا آنی) می‌باشد بطوری که این روش قابلیت پیاده سازی بلادرنگ در بستر پردازشگرهای سیگنال را داشته باشد. پیاده سازی هم بصورت نرم افزاری (شبیه سازی با نرم افزار کد کمپوزر) و هم بصورت سخت افزاری (به کمک ماژولهای ارزیابی موجود)انجام می‌شود.

در این پایان‌نامه، روشی کارا برای آشکارسازی و تعیین برخط نوع تغییر نما که مناسب برای پیاده‌سازی سخت‌افزاری در بستر پردازشگرهای سیگنال باشد، پیشنهاد شده است که حجم حافظه‌ی مصرفی آن کم و سرعت آن بالاست. روش پیشنهادی روی برد آموزشی مبتنی بر پردازشگر *TMS320C5505* پیاده‌سازی گردیده است. در حوزه شبیه‌سازی، برای ارزیابی کارایی عملکرد روش پیشنهادی از دو معیار دقت و یادآوری و برای ارزیابی کارایی سرعت از معیار متوسط زمان اجرا برای هر فریم استفاده شده است. بررسی‌ها روی چهار دسته از دنباله‌های ویدیویی شامل دسته‌های ورزشی، اخبار، سینمایی و کارتونی انجام شده است. در حوزه پیاده‌سازی تعداد سیکل‌های ماشین صرف شده در پردازشگر به ازاء هر فریم اعلام شده است. مقدار متوسط پارامترهای دقت و یادآوری برابر با به ترتیب

۹۵,۲ درصد و ۹۴,۵ درصد و مقدار متوسط زمان اجرا برای هر فریم در سیستم مورد استفاده برابر با ۰,۰۹۵ ثانیه به دست آمده است.

همچنین در این پایان‌نامه، اقدام به پیاده سازی روش پیشنهادی در بستر *TMS320DM6446* شد، اما به دلیل کمبود وقت، عدم وجود منابع مفید و افراد مجرب، نتیجه مورد نظر حاصل نشد اما تحقیقات صورت گرفته در این زمینه ارائه شده است.

لازم به ذکر است اکثر پایگاه داده‌ها به صورت دستی، تهیه و جمع آوری شده‌اند. از دستگاه *TV Capture* جهت تولید پایگاه داده دنباله‌های ویدیوئی با مشخصات معین (شامل نرخ قاب بر ثانیه و نوع محتوا) استفاده شده است.

کلمات کلیدی: آشکارسازی تغییر نما، پیاده سازی سخت‌افزاری، تعیین نوع تغییر نما، پردازشگر سیگنال *TMS320C55xx*،پردازشگر سیگنال *TMS320DM6446*

لیست مقالات پذیرفته شده/چاپ شده

[۱] هادی گرایلو، مصطفی صفایی و رویا سلطانی "ارائه روشی برای آشکار سازی و تعیین نوع تغییر نما در دنباله های ویدیویی و مناسب برای پیاده سازی سخت افزاریِ زمان حقیقی در بستر *FPGA* و پردازنده های *DSP*"، ارائه شده در هفتمین کنفرانس ماشین‌بینایی و پردازش تصویر، ۲۵ و ۲۶ آبان ۱۳۹۰.

[۲] هادی گرایلو، مصطفی صفایی و آیدین خداشناس "آشکارسازی برخط تغییر نما و تعیین نوع آن در دنباله های ویدیویی و پیاده سازی سخت‌افزاری آن در بستر پردازشگر سیگنال *TMS320C5505*" ارائه شده در چهارمین کنفرانس فناوری اطلاعات و دانش، ۲ لغایت ۴ خرداد ۱۳۹۱.

*[۳]*هادی گرایلو، مصطفی صفایی و رویا سلطانی "ارائه روشی برون‌خط برای انجام سریع آشکارسازی و تعیین نوع تغییر مرز نما در دنباله‌های ویدیویی" پذیرفته شده در اولین کنفرانس بازشناسی الگو و تحلیل تصویر ایران، ۱۶ تا ۱۸ اسفند ۱۳۹۱.

[۴] هادی گرایلو، امین قنبرزاده، آیدین خداشناس و مصطفی صفایی "استفاده از تکنیک انطباق الگوی بهبود یافته و مدل سازی چندجمله‌ای برای فشرده سازی با/بدون اتلاف سیگنالهای گفتار" ارائه شده در بیستمین کنفرانس مهندسی برق ایران، ۲۶ لغایت ۲۸ اردیبهشت ۱۳۹۱.

فهرست عناوین:

فهرست اشکال :

فهرست جداول:

فصل اول

مقدمه

۱-۱- معرفی آنالیز محتوای ویدیو

میزان اطلاعات چندرسانه‌ای در چند سال اخیر رشد زیادی داشته است. این رشد به سبب پیشرفت‌هایی است که در زمینه ذخیره اطلاعات، تکنولوژی ارتباطات و پردازش سیگنال‌های ویدیو/صوت صورت گرفته است. ویدیو به سبب حجم بالایی که دارد، در این رشد نقش مهمی را ایفا می‌کند. بدین سبب، نیاز به سازماندهی کردن مجموعه های بزرگ ویدیوهای دیجیتالی، برای دسترسی و بازیابی مؤثر احساس می‌شود. لازمه‌ی این امر، تکنیک‌هایی می‌باشند که ویدیوی دیجیتالی را به صورت واحدهای معنی‌دار و پیوسته سازماندهی کنند. به چنین اموری، آنالیز محتوای ویدیو می‌گویند که به معنی فهمیدن مفهوم ویدیوست. در این پایان‌نامه یکی از کارهای مهم در زمینه آنالیز محتوای ویدیو به نام تشخیص تغییر نما یا قسمت‌بندی ویدیو توضیح داده می شود. تقسیم‌بندی ویدیو یا تشخیص تغییر نما، در واقع بخش کردن دنباله ویدیو به واحدهای معنی‌دار کوچکتر بر اساس ناپیوستگی‌های زمانیست.

با وجود حجم عظیمی از اطلاعات ویدیویی، کمبود الگوریتم های اتوماتیک برای آنالیز محتوای ویدیویی بیشتر احساس می‌شود. در حالی که انسان در استخراج مفاهیم معنی‌دار از اطلاعات ویدیویی ماهر است، تبدیل این مهارت به یک الگوریتم اتوماتیک به عنوان مشکلی چالش برانگیز باقی مانده است و هنوز نیاز به یک روش منظم برای آنالیز محتوای ویدیویی احساس می شود.

در حالت کلی محتوای آنالیز ویدیویی شامل پنج مورد زیر می‌شود:

- استخراج ویژگی
- آنالیز ساختار ویدیو
- چکیده ساختن

- طبقه‌بندی ویدیو
- اندیس‌گذاری

در ادامه در مورد هر یک از پنج مورد فوق به اختصار توضیحی داده می شود.

۱-۱-۱- استخراج ویژگی

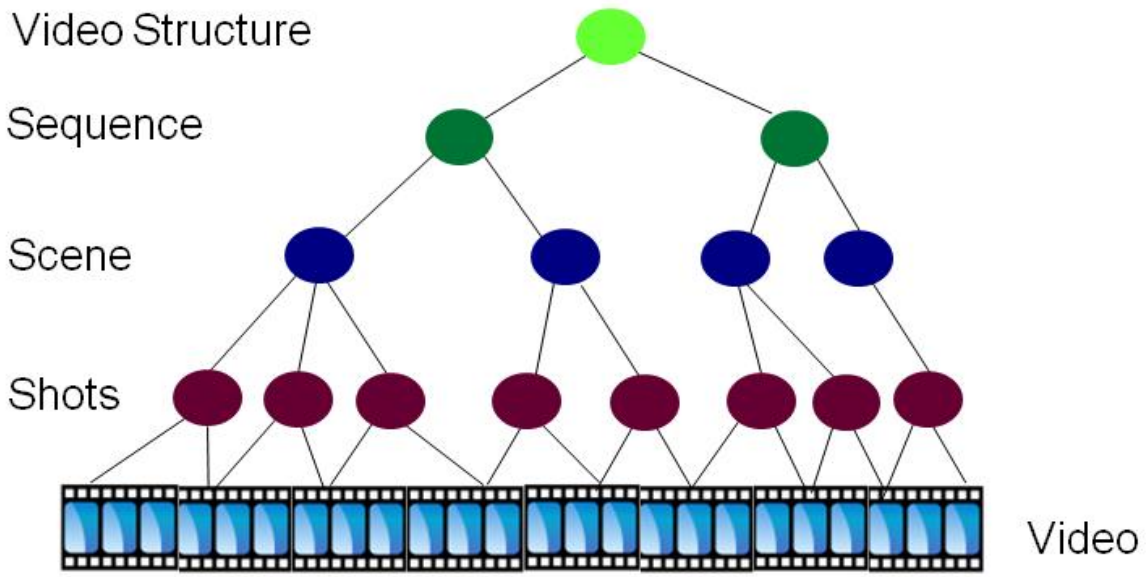
ویژگی در واقع یک پارامتر توصیف کننده است که از یک عکس یا دنباله ویدیو استخراج می شود. اثر آنالیز محتوای ویدیو به اثر ویژگی استخراج شده برای نمایش محتوا بستگی دارد. بر اساس پیچیدگی و استفاده‌های معنایی، ویژگی‌ها می توانند به دو قسمت سطح پایین[۱] و سطح بالا[۲] تقسیم می‌شوند. ویژگیهای سطح پایین مانند رنگ، متن، شکل و حرکت اشیاء می توانند به صورت اتوماتیک استخراج شوند. اما ممکن است از نظر ادراک انسان معنی دار نباشند. ویژگی‌های سطح بالا شامل درجات مختلف معناهای نمایش داده شده در تصویر و ویدیوست.

مسئله اساسی، انتخاب ویژگی های مهم برای کار داده شده است. آنالیز مؤثر محتوای ویدیو با استفاده از ترکیب ویژگیهای سطح بالا وسطح پایین حاصل می شود. ما می توانیم از وِیژگی‌های سطح پایین، برای تقسیم یک دنباله ویدیویی به نماهای منحصر به فرد و تولید فریم های کلیدی نمایش دهنده برای هر نما استفاده کنیم. این فریم‌های کلیدی سپس می توانند برای طبقه بندی و اندیس‌گذاری ویدیوها به کار روند.

[1] *Low-Level*
[2] *High-Level*

۱-۱-۲- آنالیز ساختاری

آنالیز ساختار ویدیو، فرایند استخراج اطلاعات زمانی و ساختاری از ویدیو است. در شکل (۱-۱) یک ساختار ویدیو نشان داده شده است.



شکل(۱-۱) ساختار ویدیو

آنالیز ساختار ویدیو شامل تشخیص مرزهای زمانی و تشخیص قسمت‌های معنی‌دار یک ویدیو می‌باشد. یک دنباله ویدیویی می‌تواند به واحدهای منطقی زیر تقسیم شود:

- فریم: یک فریم نمایش دهنده یک تصویر تنها در دنباله ویدیو می‌باشد.

- نما[1]: معمولاً به این صورت تعریف می‌شود که شامل دنباله‌ای از فریمهای متوالی است که به طور پیوسته و مداوم از یک دوربین
- تهیه شده باشند یا به نظر برسد که اینگونه تهیه شده‌اند[۱]. انواع تغییرنما شامل دو نوع گذر ناگهانی (AT[2]) و گذر تدریجی (GT[3]) است. در گذر ناگهانی، پس از آخرین فریم نمای فعلی، بلافاصله اولین فریم نمای بعدی ظاهر می‌شود. اما در گذر تدریجی، بین دو فریم مذکور، فریمهای دیگری قرار می‌گیرند که تغییرات را نسبتاً تدریجی از یکی به دیگری نشان می‌دهند. درشکل(۲-۱)، شش نما نشان داده شده است:

شکل(۱-۲): شش نمای مختلف

- صحنه[4]: یک صحنه یک دنباله پیوسته از نماها است، که یک معنای
- مشترک دارند. در شکل (۳-۱)، شش صحنه مختلف نشان داده شده است

[1] *Shot*
[2] *Abrupt Transition (AT)*
[3] *Gradual Transition (GT)*
[4] *scene*

شکل(۱-۳): شش صحنه مختلف

- دنباله/داستان: یک دنباله/داستان از یک مجموعه از صحنه‌ها تشکیل شده است. در شکل(۱-۴)، شش دنباله مختلف نشان داده شده است.

شکل(۱-۴): شش دنباله مختلف

در این پایان‌نامه، فقط به توضیح نما و روش آشکارسازی بر خط و بلادرنگ آن پرداخته می‌شود.

۱-۱-۳- چکیدگی ویدیو

چکیدگی ویدیو، فرایند تولید یک نمایش از محتوای ویدیو است که باید از ویدیوی اصلی کوچکتر باشد اما پیغام اصلی ویدیو را برساند. این چکیدگی شبیه استخراج کلمات کلیدی یا خلاصه‌ها از سندهای متنی است. این امر نیازمند استخراج یک زیر مجموعه از اطلاعات ویدیویی مانند فریم‌های کلیدی است. هنگامی که مجموعه وسیعی از اطلاعات ویدیویی در اختیار است، چکیدگی ویدیو بسیار مهم خواهد بود و سبب می شود قادر به جستجوی سریع در میان مجموعه بزرگی از اطلاعات ویدیویی بوده و نمایش مؤثر و دسترسی سریع داشته باشیم.

۱-۱-۴- طبقه بندی ویدیو

طبقه بندی ویدیو به شاخه‌های مختلف یک عمل خطیر بوده و امکان بازیابی و کاتالوگ سازی سریع برای مجموعه های بزرگ ویدیویی را فراهم می‌سازد. با افزایش مقدار ویدیو، جستجو و بازیابی ویدیو مشکلتر خواهد شد. آنالیز معنایی یک روش طبیعی برای طبقه‌بندی ویدیوست زیرا ویدیوهای طبقات مختلف، در معنا متفاوت هستند. به هر حال نمایش معنایی یک عمل چالش برانگیز است زیرا بطور جدی ساختار نیافته است. بنابراین مشکل طبقه بندی ویدیو را می توان از طریق استخراج ومدل کردن ویژگیهای سطح پایین، حل کرد.

۱-۱-۵- اندیس گذاری برای بازیابی و جستجو

نشانه‌های ساختاری و محتوایی که در طی فرایند های استخراج ویژگی، تجزیه ویدیو و طبقه بندی ویدیو به دست می‌آیند، معمولا به عنوان "متا دیتا[1]" نامیده می‌شوند. بر اساس این متا دیتا، ما می‌توانیم اندیس‌های ویدیویی و جدول محتوایی بسازیم. به هر حال به دست آوردن یک راه حل عمومی برای اندیس گذاری ویدیو برای تمام طبقات ویدیو، مشکل است.

۱-۲- سیستم های جایگذاری شده[2]

۱-۱-۶- مقدمه

این واژه برای سیستمهای دیجیتالی‌ای استفاده می شود که تنها برای انجام یک یا چند عمل محدود خاص طراحی شده‌اند و به تنهایی می‌توانند یک عمل یا عملیاتی محدود را انجام دهند. هسته اصلی این

[1] *Metadata*
[2] *Embedded Systems*

سیستم ها یک یا چند میکروکنترلر یا *DSP* است که برنامه‌ریزی شده‌اند و معمولا به عنوان هسته یک سیستم سخت افزاری یا مکانیکی عمل می‌کنند.

۱-۱-۷-اجزای اصلی یک سیستم جایگذاری شده

۱-۲-۲-۱- پردازنده

پردازند ای که در این گونه سیستمها استفاده می‌شود معمولا مناسب برای وظایف سیستم است زیرا اگرپردازنده کندتر یا به عبارتی ضعیفتر باشد نمی‌تواند به موقع پاسخگوی وظایف سیستم باشد و اگر خیلی سریعتر باشد هزینه تمام شده سیستم بالا می رود. خوشبختانه پردازنده‌های زیادی توسط شرکت‌های مختلف تولید شده اند که بسته به نیاز سیستم و قیمت تمام شده و قابلیت های آنها، می‌توان از آنها بهره برد. این پردازنده‌ها که به عبارت هسته یا *Core* هم از آنها یاد می‌شود در خانواده‌های و سریهای مختلفی از میکروکنترلرها استفاده شده‌اند. به عنوان مثال پردازنده ۸ بیتی *AVR* را می‌توان نام برد که در خانواده‌های *Attiny*، *Atmega* ودر میکروکنترلرهای مختلفی از این خانواده‌ها، تولید شده است که هر خانواده دارای قابلیت‌های مختص به خود است. لازم به ذکر است پردازنده‌هایی همچون *AVR* فقط توسط شرکت *Atmel* که خود طراح و سازنده آن بوده، مورد استفاده قرار می‌گیرد و دیگر شرکت ها مانند *Microchip* از پردازنده های مختص خود استفاده می‌کنند. پس می توان نتیجه گرفت کدهای برنامه را (مخصوصا اگر به زبان اسمبلی باشند) نمی توان از یک معماری به معماری دیگر منتقل کرد. ولی در این میان پردازنده‌هایی نیز هستند که معماری و هسته خود را در اختیار شرکتهای مختلف قرار داده (بفروش رسانده‌اند) و شرکتهای مختلف، از آن معماری و هسته، در میکروکنترلرهای خود استفاده کرده‌اند که این کار مشکل انتقال برنامه را از یک میکروکنترلر به میکروکنترلر دیگر برطرف نموده است. همچنین نیاز به مطالعه پردازنده و معماری‌های مختلف را از میان برداشته است.

۱-۲-۲-۲- سیستم عامل

اغلب این سیستم‌ها تک وظیفه ای هستند ولی می‌توان با استفاده از سیستم‌عاملهای بلادرنگ یا *RTOS*[1]، از پردازنده به عنوان سیستم چند وظیفه‌ای استفاده کرد. این سیستم‌عامل‌ها دارای ویژگیهای مخلتفی هستند که از جمله آنها می‌توان موارد زیر را نام برد:

- مدیریت و برنامه اجرای وظیفه‌ها با اولویت‌های مختلف
- مدیریت حافظه
- مدیریت منابع
- ارتباط بین وظیفه ها
- و ...

در این میان نیز سیستم‌عاملهای مختلفی وجود دارد که بسته به نیاز سیستم، قابلیت‌های آنها و هزینه تمام شده، می‌توان از آنها استفاده کرد. قابلیت پورت شدن یا به عبارت دیگر اجرا شدن سیستم‌عامل برای میکروکنترلرها و تاخیر در اجرای وظایف ونوع مدیریت آنها، می توانند ملاکهایی در انتخاب این سیستم‌عامل ها باشند. بعلاوه بعضی از شرکت های ارائه کننده، به همراه با این سیستم‌عاملها بسته‌های نرم‌افزاری کاربردی مختلفی ارائه می‌دهند ازجمله:

- *TCP/IP Stack*
- *GUI design*

[1] *Real-Time Operating Systems*

- *System File*
- *USB device*
- *USB Host*
- و ...

که این بست ها نیز می توانند ملاک خوبی برای انتخاب باشند. همچنین برخی از این سیستم‌عاملها بصورت رایگان و متن باز[1] ارائه می‌شوند مانند سیستم *FreeRTOS* که می‌توانند انتخاب خوبی باشند. ولی از طرفی جزء سیستم‌عاملهای مطمئن نیستند و ممکن است راهنمای کافی برای آنها وجود نداشته باشد. برخی دیگر از این سیستم‌عاملها مانند *IAR PowerPac* دارای قابلیت بالا و فایل‌های راهنمای مناسب می‌باشند ولی برای استفاده از آنها می‌بایست آنها را خریداری کرد. در پروژه‌هایی نیاز به سیستم‌عاملهای سطح بالا مانند *Linux* یا *WinCE* می‌باشد که در این حالت سیستم می‌باسیت دارای حافظه کافی و پردازنده سریع، معمولا بر پایه پردازنده های *ARM*، باشد.

۱-۲-۲-۳- حافظه سیستم

اغلب میکروکنترلرها علاوه بر واحد پردازنده و گاه واحد کمک پردازنده، دارای حافظه‌های *RAM*، *FLASH* و *ROM* داخلی هستند ولی حجم این حافظه ها کم می‌باشد. برای سیستم هایی که نیاز به ذخیره سازی اطلاعات با سرعت بالا دارند باید حافظه های *RAM* با سرعت مناسب در سیستم و بر روی برد قرار گیرد که بسته به نیاز سیستم، می‌تواند از نوع سنکرون و یا آسنکرون باشند.

[1] *Open Source*

همچنین در مواقعی که می‌خواهیم از سیستم عامل های سطح بالا مانند *Linux* ویا *WinCE* استفاده کنیم، می‌بایست حداقل حافظه مورد نیاز این سیستم عامل هارا فراهم کرد. بنابراین اغلب نیاز به حافظه های *RAM* خارجی مانند *SD RAM* و *DDR RAM* داریم.

۱-۲-۲-۴- اجزای جانبی

اجزای جانبی میکروکنترلر نیز یکی از پارامترهای بسیار مهم دیگر برای انتخاب میکروکنترلر می‌باشند.

ابتدا بهتر است تعریفی ازاجزای جانبی میکروکنترلر داشته باشیم:

به عبارت کلی می توان گفت بجز هسته، حافظه های داخلی و کنترلرسیستم، مانند واحد کنترل وقفه‌ها، ریست و همانند آنها، جزء اجزای جانبی خواهند بود. نمونه‌ای از اجزای جانبی میکروکنترلر عبارت اند از:

- تایمر/شمارنده
- مبدل آنالوگ به دیجیتال
- مبدل دیجیتال به آنالوگ
- رابط‌های سریال سنکرون و آسنکرون
- رابط *USB*
- کنترلر *USB Host*
- لایه شبکه *MAC*
- کنترلر حافظه های خارجی
- کنترلر *LCD*
- کنترلر *CAN*

- و...

۱-۳- معرفی *DSP* ها

۱-۳-۱- مقدمه

در قسمت قبل، در مورد سیستم‌های جایگذاری شده توضیحاتی داده شد. یکی از محبوبترین شرکت‌هایی که در این زمینه فعالیت دارد، شرکت *TEXAS INSTRUMENT* یا به اختصار شرکت *TI* است که به تولید میکروکنترلر و پردازنده سیگنال پرداخته است. در این قسمت توضیح مختصری در مورد پردازنده‌های سیگنال مربوط به این شرکت ارائه می‌شود.

در دهه ۷۰ میلادی همزمان با ساخت اولین پردازنده ها توسط شرکت های مختلف، شرکت *TI* تصمیم گرفت پردازنده هایی را مخصوص پردازش سیگنال طراحی و روانه بازار نماید. این پردازنده‌ها که بیشتر با نام پردازنده‌های *DSP*[1] معروف هستند، همگی با *TMS320* شروع می‌شوند. پردازنده‌های *DSP* در حدود ۴۰ سال از حضورشان بسیار تکامل یافته و امروزه وارد حوزه های مختلفی شده‌اند. اولین سری پردازنده‌های *DSP*، با نام *TMS320C10* به بازار عرضه شد. پس از چند سال حضور سری *TMS320C25* باعث معروف شدن *DSP*ها گردید. این پردازنده که در اواخر دهه ۷۰ وارد بازار شد می‌توانست یک تبدیل فوریه را با سرعتی انجام دهد که ۲۰ سال بعد اولین سری های پردازنده‌های پنتیوم ساخت شرکت اینتل به آن رسیدند.

پردازنده‌های *DSP* هم اکنون در دو حوزه جداگانه کاربرد دارند:

الف- محاسبات ریاضی و پردازش سیگنال

[1] *Digital Signal Processing*

ب- به عنوان یک میکروکنترلر پر سرعت

در کشور ما در سالهای اخیر نیاز به پردازنده‌های پر سرعت افزایش یافته است. این نیاز باعث شده تا عده‌ای از طراحان از میکروهای پر سرعت تر(در حدود ۵۰ مگاهرتز) استفاده کنند. در بعضی از کاربردها هم طراحان به *DSP* هایی با سرعت ۱۵۰ تا ۳۰۰ مگاهرتز روی آورده اند. البته این *DSP* ها بیش از ۱۵ سال است که وارد بازار شده اند اما میکروهای پر سرعت به تازگی توسط شرکتهایی نظیر *ATMEL* معرفی شده اند.[۲]

پس یکی از کاربردهای پردازنده های *DSP*، استفاده از سرعت بالای آنها و کامپایلر بسیار قوی و بدون اشکالی است که شرکت *TI* برای آنها معرفی کرده است. در *DSP*ها قابلیتهای زیادی وجود دارد که این قابلیتها با کمک دستورات اسمبلی زیاد و چند کاره آنها ایجاد شده است. این دستورات به کامپایلرها کمک می کنند که بتوانند یک کد اسمبلی سریع از روی برنامه *C* تولید کنند.

۱-۳-۲- نگاهی اجمالی به پردازنده‌های مختلف ساخت شرکت *TI*

شرکت *TI* در طول ۴۰ سال حضور خود در بازار پردازنده‌های پرقدرت، دائما در حال نوآوری و تولید محصولات جدید بوده است. اگر امروز شما سری به سایت این شرکت[1] بزنید تقریباً می توانید برای هر کاربردی یک آی سی مناسب بیابید. البته در بازار ایران مشهورترین آی سی های ساخت این شرکت همان *DSP* ها هستند اما قطعات تولید شده توسط *TI* محدود به پردازنده های *DSP* نمی‌باشد. تنها مشکل این شرکت این است که قطعات تولیدی آن نسبت به دیگر شرکت‌ها کمی گران‌تر می باشد. این گرانی قطعات باعث شده که فروشندگان بازار ایران به دلیل نوع تقاضای مشتریان،ترجیح دهند قطعات

[1] *www.ti.com*

ارزان‌تر مشابه شرکتهای دیگر را به بازار عرضه کنند. البته این گرانی قطعات بیشتر دلایل فنی دارد و به قابلیتهای آنها بر می‌گردد.

۱-۳-۲-۱- سری های قدیمی

شرکت *TI* تا قبل از ارائه نرم افزار *CCS*[1]، پردازنده‌هایی را به بازار عرضه میکرد که امکان اتصال آنها از طریق *JTAG* [2] به کامپیوتر وجود نداشت. شماره اصلی این خانواده ها همگی ۲ رقمی بودند، مثلا *C25* و یا *C50* که نام کامل آنها*TMS320C25* یا *TMS320C50* می‌باشد.

۱-۳-۲-۲- سری های جدید

با ورود *JTAG* به عرصه پردازنده‌ها، شرکت *TI* نیز یک نرم افزار یک پارچه به نام *Code Composer Studio* به بازار عرضه نمود. این نرم افزار با کمک *JTAG* به کامپیوتر وصل شده و کار با پردازنده‌های *DSP* را آسان می نماید. سری‌های جدید ساخت شرکت *TI* چهار رقمی شده و در سه دسته اصلی تقسیم بندی شدند.

الف- سری ۵۰۰۰ (یا *5XXX*)

این سری شامل دو خانواده اصلی می باشد: *54XX* و *55XX*. سری *55XX* کم مصرف ترین پردازنده *DSP* ساخت شرکت *TI* می باشد که در بسیاری از تجهیزاتی که نیاز به پردازش بالا و جریان مصرفی کم (مانند موبایلها) دارند، مورد استفاده قرار گرفته‌اند. در حال حاضر عملا طراحی سری های جدید برای خانواده *54XX* متوقف شده است و هر روز پردازنده‌ای کم مصرفتر بر اساس سری *55XX* به بازار عرضه می‌گردد. در سری *5000* سرعت پردازنده ها بین ۱۰۰ تا ۳۰۰ مگاهرتز می‌باشد و در سری *55XX* می‌تواند تا ۶۰۰

---

[1] نرم افزار جامعی است که شرکت TI برای تمام انواع پردازنده‌های ساخت خود تولید نموده است.این کلمه مخفف *Code Composer Studio* می باشد.

[2] یک مدار کوچک سخت افزاری است که برای اتصال *DSP*ها به کامپیوتر استفاده می شود.با کمک *JTAG* می توان برنامه ها را پس از کامپایل شدن،با کمک نرم افزار *CCS* بر روی پردازنده بارگذاری کرده و سپس اجرا نمود.

میلیون ضرب را در ثانیه انجام دهد. کاربرد اصلی پردازنده‌های سری *5000* در پردازش صوت و پروسه‌هایی که نیاز به پردازش پر سرعت دارند، می‌باشد. از بعضی از سری‌ها که حجم حافظه داخلی آنها بیشتر از ۱۲۸ کیلوبایت است می‌توان برای پردازش تصویر استفاده نمود.

ب- سری *2000* (یا *2XXX*)

این سری شامل دو خانواده اصلی می‌باشد: *24XX* و *28XX* ، سری *28XX* یک خانواده با عملکردی نزدیک به میکروکنترلرها می‌باشد. این سری تنها سری است که در آن حافظه *FLASH* وجود دارد. وجود حافظه *FLASH* داخلی، برنامه ریزی این پردازنده‌ها را نسبت به دیگر خانواده ها آسانتر نموده است. در این سری ها حجم حافظه داخلی از نوع *SRAM* کمتر از ۳۲ کیلو می‌باشد. به همین دلیل این سری‌ها برای پردازش تصویر مناسب نمی‌باشند. کاربرد اصلی این سری بیشتر به عنوان یک میکروکنترلر پرسرعت[1] میباشد و در سایت شرکت *TI* نیز این سری جز پردازنده های *DSP* طبقه بندی نشده است.

ج- سری *6000* (یا *6XXX*)

این سری شامل سه خانواده اصلی می‌باشد: *62XX*، *64XX* و *67XX*. این سری‌ها پیشرفته‌ترین پردازنده‌های ساخت شرکت TI هستند. در این سری ها فرکانس کاری پردازنده بین ۱۵۰ مگاهرتز تا ۱/۲ گیگاهرتز می باشد. اما سرعت واقعی این پردازنده ها ۸ برابر کلاک کاری آنها می‌باشد. در این پردازنده‌ها در هر کلاک حداکثر ۸ دستور به شکل همزمان قابل اجراست. به همین دلیل این پردازنده‌ها می‌توانند تا حدود ۱۰ گیگا دستورالعمل را در ثانیه اجرا نمایند. این خانواده برای تمامی انواع پردازشهای پر سرعت مناسب هستند اما سری *64XX* با قابلیتهای خاص آن مناسبترین سری برای پردازش تصویر است. در بین خانواده‌های مختلف، پیچیده‌ترین سری از لحاظ سخت افزار، سری ۶۰۰۰ می‌باشد. سری‌های ۲۰۰۰ و ۵۰۰۰ از نظر طراحی سخت افزار، پیچیدگی یکسانی دارند.

[1] *DSC=Digital Signal Controler*

علاوه بر موارد فوق، شرکت *TI*، در سالهای اخیر محصولات جدیدتری ارائه داده است. از جمله این محصولات می‌توان از سری‌های *DAVINCI* و *OMAP* نام برد. اساس کار این دو خانواده، استفاده از دو هسته[1] غیرهمگون می‌باشد. یک هسته وظایف عمومی مانند انتقال اطلاعات و اجرای رابط گرافیکی را انجام می‌دهد، به همین سبب به آن پردازنده هدف عمومی(یا هسته همه کاره)[2] می‌گویند و هسته دیگر وظایفی مانند فشرده کردن جریان‌های ویدیویی و گفتار و صوت یا از حالت فشرده خارج کردن آنها را انجام می‌دهد. آن هسته را پردازنده سیگنال دیجیتال می‌گویند.

پردازنده‌های *DAVINCI*، برای سیستم‌های ویدیوی دیجیتال بهینه شده‌اند و برای بسیاری از برنامه‌های مربوط به ویدیو، تصویر و بینایی ماشین مناسب می‌باشند. این پردازنده‌ها علاوه بر دو هسته توضیح داده شده در فوق، شامل شتاب دهنده‌های ویدیویی[3] و اجزای جانبی کافی بوده که با توجه به عملکرد بالا، حافظه کافی برای ذخیره سازی و قیمت مناسب، آن را به انتخابی ایده‌آل تبدیل کرده است. پردازنده‌های داوینچی با عبارت *TMS320DM* یا به اختصار *DM* آغاز می‌شوند. سیستم عاملی که در این سری از محصولات استفاده می شود، سیستم عامل لینوکس است که در قسمت بعدی، توضیحاتی در مورد آن ارائه شده است.

در فصل ۳، در مورد پردازنده *TMS320C5505* از سری *C55XX* توضیحاتی داده خواهد شد و در فصل۴، پردازنده *TMS320DM6446* از خانواده *DAVINCI* مورد بررسی قرار می‌گیرد.

۴-۱- سیستم عامل لینوکس

لینوکس، یک سیستم عامل آزاد و متن‌باز است که تحت مجوز *GNU/GPL* منتشر شده است. متن‌باز

[1] *Core*
[2] *General Purpose Processor*
[3] *Video Accelerators*

به این معنی که هر شخصی آزاد است تا از آن استفاده کند و آن را تغییر دهد، حتی می‌تواند آنرا دوباره توزیع کند. لینوکس در سال ۱۹۹۱ در دانشگاه هلسینکی فنلاند توسط یک دانشجوی جوان به نام تروالدز نوشته شد. تراولدز در اصل با *Minix* (که یه شبه یونیکس خلاصه شده است) کار می‌کرد اما تصمیم گرفت تا سیستم‌عامل دلخواه خود و بر اساس *Unix* خلق کند و این ماجرا با انتشار نسخه‌ی ۰،۰۲ در سال ۱۹۹۱ توسط او آغاز شد. لینوس تروولدز طی یک نامه‌ی الکترونیکی خبر نوشتن سیستم‌عامل متن‌باز خود را اعلام کرد، طولی نکشید که صدها نفر از سراسر دنیا خواهان کار با این سیستم‌عامل متن‌باز که به *Unix* شباهت داشت، شدند و شروع به توسعه‌ی آن کردند. امروزه توزیع‌های فراوانی از لینوکس وجود دارد که از هسته و نرم افزارهای آزاد[1] همراه هسته تشکیل می‌شوند. همه‌ی این توزیع‌ها از یک هسته‌ی واحد به نام لینوکس استفاده می‌کنند. اگر بخواهیم دقیقتر توضیح بدهیم، لینوکس یکسری کد است که ارتباط بین سخت افزارها و نرم‌افزارها را برقرار می‌کند (به عنوان هسته یا کرنل) و یک توزیع لینوکس(که در زبان عام به آن لینوکس می‌گویند) شامل هسته و تعداد زیادی نرم‌افزار متن‌باز می‌باشد مانند توزیع‌های ردهت [2]، اسلاکویر[3]، جنتو[4] و دبیان[5]. شایان ذکر است که امروزه برای راحتی به توزیع‌های لینوکس همان لینوکس گفته می شود که البته صحیح آن گنو/لینوکس می باشد، چرا که بیشتر نرم افزارهای استفاده شده در این توزیع‌ها تحت مجوزهای گنو انتشار یافته‌اند. بنابراین این از این پس هر جا که از لینوکس نام می‌بریم منظورمان همان توزیع های گنو/لینوکس است مگر ان که صریحاً به هسته‌ی لینوکس اشاره کنیم

.

همانطور که اشاره کردیم، گنو/لینوکس یک سیستم‌عامل متن‌باز است. متن‌باز بودن گنو/لینوکس باعث شده تا عده‌ی فراوانی از سرتاسر جهان توسط اینترنت گردهم آمده و برروی این سیستم‌عامل کار کنند.

---

[1] *Free Software*
[2] *Redhat*
[3] *Slackware*
[4] *Gento*
[5] *Debian*

نتیجه‌ی این گردهمایی سیستم‌عاملی قدرتمند با ویژگیهایی منحصر به فرد شده است. رایگان و متن‌باز بودن و بسیاری ویژگی‌های دیگر سبب شده است تا لینوکس، که در ابتدا در محیط‌های دانشگاهی و آکادمیک رشد کرده است، پابه عرصه ی تجارت و دولت بگذارد تا جایی که دولت‌های بزرگ اروپا مانند آلمان و اسپانیا سیستم‌عامل ملی خود را لینوکس اعلام کنند. البته نه تنها دولت‌ها بلکه نهادها و وزارتخانه‌های بسیاری سیستم‌عامل نهاد یا وزارت خانه ی خود را به لینوکس تبدیل کرده‌اند مانند وزارت دفاع آمریکا، اداره‌ی پست آمریکا و نیروهای مسلح ترکیه. حتی در ایالت کالیفرنیای امریکا نیز برای اداره‌ی امور از این سیستم‌عامل استفاده می‌شود. شاید برای شما این پرسش مطرح شده باشد که ویژگیهای لینوکس چیست که این گونه همگان را به سوی خود جذب کرده است؟ در ادامه برخی از ویژگی‌های مهم لینوکس ذکر می‌شود.

۱-۴-۱- ویژگی های مهم لینوکس

✓ هزینه

گنو/لینوکس یک سیستم عامل رایگان است، البته توزیع‌هایی هم وجود دارد که به صورت تجاری ارائه می‌شوند اما قیمت آنها همواره کمتر از سیستم‌عامل‌های تجاری مانند مایکروسافت ویندوز یا *Unix* است. البته این نکته نیز گفتنی است که اغلب توزیع‌های تجاری لینوکس را نیز می‌توان بصورت رایگان از اینترنت دریافت کرد یا حتی برای دوستانتان نیز می توانید کپی کنید اما این نسخه‌هایی که بابت آنها پولی پرداخت نشده، دارای خدمات پشتیبانی یاکتابچه‌ی راهنما نیستند (هر چند که همواره هزاران صفحه اطلاعات رایگان در اینترنت در بارهی توزیعهای مختلف گنو/لینوکس وجود دارد). پایین بودن هزینه های

گنو/لینوکس یکی از عواملی است که دولت‌های بسیار و شرکت‌های بزرگ را واداشته تا این سیستم‌عامل را سیستم‌عامل رسمی خود اعلام کنند.

✓امنیت و پایداری

لینوکس با توجه به ساختار خود سیستم‌عاملی امن و پایدار است. پایداری لینوکس مدیون روش استفاده‌ی صحیح از سخت افزار است که این خصلت را از *Unix* به ارث برده و امنیت آن، علاوه بر ساختار صحیح آن، مدیون بازمتن بودنش می‌باشد. باز متن بودن باعث شده است تا در صورت مشاهده‌ی کوچکترین مشکل در هسته یا نرم افزارهای جانبی، در چند ساعت یاحد اکثر چند روز، توسعه دهندگان و برنامه نویسان ضعف و مشکل را حل کرده و راه حل را در اختیارعموم کاربران قرار دهند. اینگونه است که گنو/لینوکس در پایداری و امنیت به یک افسانه تبدیل شده است و جزو ایمن‌ترین و پایدارترین سیستم‌عامل‌های جهان به شمار می‌رود. موضوع پایداری و امنیت در حوزه ی سرورها و شبکه بسیار مهم است تا جایی که لینوکس بیش از ۳۰ درصد از سرورهای جهان را شامل می‌شود و۷۰ درصد دیگر شامل*SCO Unix*، *SUN Unix*، *IRIX*، *HP-UX* ودیگر یونیکس‌ها و *Apple Mac* و درصدی هم ویندوز می‌شود. با توجه به تعدد این سیستم‌ها می‌بینیم ۳۰ در صد خود رقمی قابل توجه است. باید خاطر نشان کنیم که تا به حال برای لینوکس هیچ ویروس یا کرمی به آن شکل که برای ویندوز مایکروسافت وجود دارد نوشته نشده است و برای نرم افزار های مدیریت ایمیل یا جستجوی وب آن هم ویروسی شناسایی نشده و برعکس*IE* [1] ویندوز که همواره باید برای آن وصله‌های امنیتی دانلود کرد برای مرورگرهای لینوکس نیازی به تلف کردن وقت با این کارها نیست.

[1] *Internet Explorer*

✓ نیاز سخت افزاری اندک

گنو/لینوکس به دلیل ساختار هسته‌ی مناسب و تعدد توزیع ها می‌تواند با حداقل امکانات سخت افزاری به خوبی کار کند تا جایی که حتی توزیع‌هایی وجود دارد که بدون وجود هارد دیسک برروی سیستم یا با حافظه *RAM* معادل ۱۶ مگابایت می‌تواند سیستم شما را راه اندازی کند.

✓ تنظیم و شخصی سازی

گنو/لینوکس را میتوان با رابط خط فرمان[1] همانند *MS-DOS* و یا با رابط گرافیکی مانند مایکروسافت ویندوز یا *Apple Mac* اجرا و استفاده کرد. دردسر های معمول تنظیم و شخصی سازی میزکار با وجود محیط‌های گرافیکی مختلف مانند کی دی ای[2] یا گنوم[3] از بین رفته است و شما می‌توانید از دهها محیط گرافیکی گنو/لینوکس حداکثر بهره را ببرید. محیط های گرافیکی گنو/لینوکس که با هدف رفع نیاز کاربران رائه شده است و به صورت متن‌باز می‌باشد، توسط گروه بزرگی از برنامه نویسان خلق شده و در حال توسعه هستند. شما میتوانید باتوجه به شرایط سخت افزاری و سلیقه ی خود از هر کدام از این محیط های گرافیکی بهره برده و بر اساس نیازتان تنظیمات لازم را انجام دهید.

✓ آزادی

لینوکس یک سیستم‌عامل آزاد است. شما می‌توانید آن را با در دست داشتن کدهای منبع، که برنامه‌نویسان و خالقین لینوکس در اختیار شما قرار می دهند، مطابق میل خود تغییر و توسعه دهید و

---

[1] *Command Line Interface*
[2] *KDE*
[3] *GNOME*

توزیع‌های خود را منتشر کنید. کافیست شما به زبان های سی، سی پلاس و اسمبلی آشنا باشید. حتی برای کاربران غیر فنی که به این زبان ها اشنایی ندارند ابزار ها و توزیع‌هایی معرفی می شود تا با حداقل دانش برنامه نویسی به انتشار توزیع مخصوص خود بپردازند. این آزادی در تغییر و توزیع مجدد با در دست داشتن کدهای منبع باعث رواج هر چه بیشتر گنو/لینوکس شده است تا جایی که شما می‌توانید برای هر کاربردی یک لینوکس بیابید. برای نمونه لینوکسی فقط برای کار های وب وجود دارد و لینوکسی دیگر فقط برای پخش مولتی‌مدیا و یا لینوکسی برای شبکه. حتی لینوکسی نیز برای پردازنده‌های سیگنال دیجیتال(مانند خانواده *DAVINCI*) ارائه شده است. البته آزاد بودن گنو/لینوکس باعث پیشرفت این سیستم‌عامل تا جایی شده است که دولت هایی مانند چین تصمیم به نوشتن توزیع مخصوص به خودشان گرفته‌اند که کاملا با ویژگیهای زبان آنها سازگاری دارد. در صورتی که این موضوع در سیستم‌عامل‌های غیرآزاد و غیر متن‌باز میسر نمی‌شود.

۱-۴-۲- کاربردهای لینوکس

گنو/لینوکس را میتوان از میز کار دانش آموزان دبستان در اسپانیا تا ماهواره های کوچک در فضا یافت! از آن جایی که لینوکس یک سیستم‌عامل ذاتاً چند کاربره[۱] و چند کاره[۲] است(بدین معنی که در یک لحظه بیش از یک کاربر می‌تواند با ان کار کند و بیش از یک برنامه را اجرا می‌کند) و در کنار این دو مزیت، سیستم‌عاملی پایدار و امن است بنابراین می‌تواند گزینه‌ای مناسب برای سرویس دهنده‌های شبکه باشد. در حال حاضر اغلب شبکه‌های بزرگ و معتبر از سیستم عامل لینوکس به عنوان سرویس دهنده‌ی اصلی خود استفاده می‌کنند. حتی سرویس دهنده های سایت *Hotmail* بعد از کرک های متناوب از سوی مهاجمین، به جای ویندوز مایکروسافت از لینوکس برای مدتی استفاده کرد تا بتواند در مقابل

[1] *Multi User*
[2] *Multi Task*

کرک‌های مداوم مهاجمان دوام بیاورد. البته شرکت بزرگ *IBM* نیز محصولات سرویس دهندهی خود را مانند سرورهای وب و شبکه با سیستم‌عامل لینوکس می‌فروشد.

البته کاربرد لینوکس به سرویس دهنده‌ها ختم نمی‌شود بلکه آن را می‌توان بر روی تقریبا هر ابزاری نصب کرد. در هند، لینوکس به عنوان سیست عامل کامپیوترهای دستی[1] استفاده می شود. در ژاپن شرکت سونی، در لوازم صوتی و تصویری خود از گنو/لینوکس استفاده می‌کند. خلاصه هر کجا که به یک سیستم‌عامل احتیاج هست، میتوان از لینوکس استفاده کرد و اگرلینوکس برای آن کار مناسب نباشد جامعه‌ی توسعه دهنده‌ی لینوکس، آن را برای کار مورد نظر توسعه می‌دهند.

البته میز کار کاربران معمولی را فراموش کردیم. همان طور که در بالا گفتیم با توجه به توزیع های متفاوت و محیط‌های گرافیکی بسیار زیاد برای کار با لینوکس، امروزه دیگر این سیستم‌عامل یک سیستم‌عامل حرفه‌ای نیست بلکه یک سیستم‌عامل حرفه‌ای و خانگی شده است.

شما می‌توانید در خانه به شنیدن موسیقی یا تماشای فیلم بپردازید یا کار های خود را مدیریت کنید یا در اینترنت به گشت وگذار بپردازید یا با دوستتان یک گپ اینترنتی بزنید و یا می‌توانید در خواست مرخصی خود را با برنامه‌های لینوکسی تایپ کنید و به مدیر خود بدهید. البته اگر شما یک برنامه‌نویس کنجکاو هستید یا برنامه نویس وب ، گنو/لینوکس بهشت شماست. البته از دیگر کاربرد های لینوکس استفاده از آن به عنوان سرور اشتراک فایل بجای سیستم‌عامل گران قیمت ویندوز *NT* مایکروسافت است. این کار را سامبا[2] که یک برنامه‌ی به اشتراک گذاری فایل است، برایتان انجام می‌دهد. البته می‌توانید به کمک یک سرور *SQL* قدرتمند که به وسیله پایگاه‌های داده‌ای متن‌بازی که همراه گنو/لینوکس ارائه می شوند، مانند *MySQL* یا *PostgerSQL* ، به رفع نیاز پایگاه داده خود بپردازید.

---

[1] *Handheld or PDA*
[2] *Samba*

۱-۲- مسائل مطرح شده در این پایان‌نامه

در قسمتهای قبل، بطور مختصر در مورد مسائل شامل شده در آنالیز محتوای ویدئو، سیستم‌های جایگذاری شده و سیستم عامل قدرتمند لینوکس، توضیحاتی داده شد. در این پایان‌نامه، بر ویژگی‌های تقسیم‌بندی ویدیو و پیاده‌سازی آن در بستر سخت افزار تمرکز خواهد شد. مشکلی که در تشخیص مرزنما وجود دارد، تشخیص ناپیوستگی‌های زمانی در دنباله‌های ویدیویی است. مسائل کلیدی، انتخاب ویژگی‌هایی برای نمایش تصاویر، انتخاب یک استاندارد در شباهت/غیر شباهت و یک الگوریتم به اندازه کافی کلی برای تشخیص هر دو گذرهای تدریجی و ناگهانی می‌باشند. در این پایان‌نامه با استفاده از ویژگی‌هایی مانند میانگین، واریانس، لبه و هیستوگرام، مسائل فوق را در نظر گرفته و روشی جدید ارائه نمودیم که علاوه بر عملکرد بهتر، قابل پیاده‌سازی در بستر پردازشگرهای سیگنال می‌باشد. همچنین این روش در بستر پردازشگر *TMS320C5505*پیاده‌سازی‌گردید. علاوه بر آن،به پیاده سازی این روش در بستر پردازنده *TMS320DM6446* اقدام گردید. به دلیل کمبود وقت و عدم وجود منابع مفید و افراد مجرب، نتیجه مورد نظر حاصل نشد اما تحقیقات صورت گرفته در این زمینه ارائه شده است.ذکر این نکته حائز اهمیت است که:" طبق دانسته های ما، تا کنون در مورد پیاده سازی الگوریتم های تغییر نما در بستر پردازشگر های سیگنال کار خاصی صورت نگرفته است".

۱-۳- ساختار پایان‌نامه

در فصل۲، مروری بر روشها و تحقیقات صورت گرفته در مورد تشخیص تغییر نما انجام خواهد شد. سپس روش جدیدی برای تشخیص تغیر مرز نما و زیر بخشهای مربوطه در فصل ۳ ارائه می‌شود. همچنین توضیحاتی در مورد*TMS320C5505* در این فصل بیان شده است و نتایج ارائه خواهد گردید. در فصل ۴

مروری خواهیم داشت بر ویژگی ها و نرم افزارهای مربوط به پردازشگر سیگنال *TMS320DM6446* و در نهایت در فصل ۵ نتیجه گیری و پیشنهادات آینده بیان خواهد گردید.

فصل دوم

مروری برروشهای موجود در تشخیص تغییر نما

۱-۲-مقدمه

آشکارسازی تغییر مرزنما یکی از اساسی‌ترین مراحل مورد نیاز در برخی کاربردها مانند بازیابی ویدیو، مرور ویدیو، اندیس‌گذاری ویدیو، استخراج قاب کلیدی، و فشرده‌سازی ویدیو است. در ادامه مثال‌هایی از اهمیت آشکار سازی مرز نما بیان می‌گردد.

امروزه برخی کتابخانه‌های دیجیتال از طریق شبکه‌های داده‌رسانی مانند اینترنت قابل دسترسی می‌باشند. گاهاً ابزار تعامل این کتابخانه‌ها به کاربران اجازه‌ی مرور و جستجوی ویدیویی را نیز می‌دهند. حجم انبوه داده‌های ویدیویی از یک طرف و محدود بودن پهنای باند انتقال شبکه‌های داده‌رسانی مزبور از طرف دیگر باعث شده است قطعه‌بندی زمانیِ دنباله‌های ویدیویی یکی از اساسی‌ترین نیازهای مورد توجه در کتابخانه‌های مذکور باشد.

در کاربرد فشرده‌سازی ویدیو از آشکارسازی تغییر مرزنما استفاده می‌شود.از آنجا که در هر نما، تزاید بین فریمهای متوالی بسیار زیادتر از تزاید بین دو فریم مربوط به دو نمای مختلف می‌باشد، برای افزایش نرخ فشرده‌سازی، باید مرزنماهای مختلف تعیین شود. در کاربرد صفحه‌ی داستان مربوط به یک دنباله‌ی ویدیویی، اولین نیاز، تهیه فریم‌کلیدی[1] است. برای استخراج فریم‌کلیدی از آشکارسازی تغییر مرزنما استفاده می‌شود.

با توجه به وجود محاسبات زیاد برنامه‌های مربوط به ویدیو، تقسیم بندی زمانی نما می‌تواند تأثیر مهمی بر کاهش پیچیدگی‌های محاسبات بگذارد*[۳]*.

[1] *Key frame*

امروزه آرشیو فیلم‌ها، مقادیر زیادی از از فیلم ها را در خود دارند. دسترسی به ابن فیلم‌ها، بعلت اینکه تقسیم بندی نشده اند و دارای تفسیر خاصی نیز نیستند، مشکل می‌باشد. تشخیص اتوماتیک مرز نما اولین قدم برای فعال سازی دسترسی غیر خطی و جستجو در این ویدیو هاست*[۴]*.

در *[۵]* ،اهمیت تشخیص مرز نما اینگونه بیان شده است*"*: برای اینکه کاربرها بتوانند به راحتی، اطلاعات ویدیویی را مورد جستجو قرار دهند یا آنها را بازیابی کنند، نیاز به طراحی سیستمی است که بطور دقیق و اتوماتیک، مقادیر بزرگی از ویدیوی غیر همگن را مورد پردازش قرار دهد. به منظور تسهیل و تسریع در پردازش، اندیس گذاری ویدیو امری است ضروری و اولین قدم در اندیس گذاری، تشخیص مرز نما می‌باشد*"*.

با افزایش برنامه‌هایی که با دیتابیس‌های عظیم ویدیویی کار می‌کنند (مانند کتابخانه‌های دیجیتال، آموزش راه دور، ویدیو بر تقاضا[1]، تبلیغات ویدیویی دیجیتال و ...)،نیاز به ابزاری که بتواند بطور مؤثر عملیات جستجو، اندیس‌دار کردن، جستجو و بازیابی را انجام دهد احساس می‌گردد. بدین منظور، اندیس‌های توصیف کننده‌ی محتوای ویدیویی لازمند و این اندیس‌ها تا جای ممکن باید غنی و کامل باشند. اولین قدم در بدست آوردن این اندیس ها، به کار گیری تقسیم بندی زمانی و آشکار سازی مرز نماست[۶]، [۷].

حال که به اهمیت تشخیص مرز نما پی بردیم،این سؤال مطرح می‌گردد که نما چیست؟

برای نما تعاریف زیادی مطرح شده است. مثلا در *[۸]*، نما این گونه بیان شده است*"*: تقسیم بندی زمانی ویدیو، بطور گسترده ای به عنوان اولین مرحله در آنالیز ساختاری ویدیو، مورد استفاده قرار می‌گیرد. الگوریتم‌های تقسیم بندی، ویدیو را به یک سری از زیردنباله‌ها تقسیم می‌کنند. هر زیر دنباله یک نما نامیده می‌شود*"*.

[1] Video on demand

یا در *[۹]* داریم″: سازماندهی کردن ویدیو و مشخص نمودن محل اطلاعات خواسته شده بطور مؤثر، چالشی عظیم پیش روی بازیابی ویدیو قرار می‌دهد.این مورد نیازمند این است که ویدیو به قسمت‌های کوچکتر و قابل مدیریت تقسیم شود. به هر یک آز آن قسمتهای کوچک و قابل مدیریت، نما گویند″.

اما نما در حالت کلی بصورت زیر تعریف می‌شود:

یک نما شامل دنباله‌ای از فریمهایی است که به طور پیوسته و مداوم از یک دوربین تهیه شده باشند یا به نظر برسد که این‌گونه تهیه شده‌اند.

تغییرات نما در حالت کلی،به دو دسته تقسیم می‌شوند:

۱. **تغییرات ناگهانی**: پس از آخرین فریم نمای فعلی، بلافاصله اولین فریم نمای بعدی ظاهر می‌شود تغییر ناگهانی را اغلب برش می‌گویند. شکل (۱-۲) دو نمونه برش را نشان می‌دهد.

شکل ۱-۲- دو نمونه برش

۲. **تغییرات تدریجی**: بین فریم نمای فعلی و فریم نمای بعدی، فریمهای دیگری قرار می‌گیرند که تغییرات را نسبتاً تدریجی از یکی به دیگری نشان می‌دهند.این تغییرات به چهار دسته تقسیم می‌شوند:

- **محو شدگی –درون**[1]: این نوع تغییر تدریجی، یک تغییر تدریجی در روشنایی است که از یک فریم ثابت (معمولا سیاه) شروع می‌شود و به فریم مورد نظر ختم می‌گردد. شکل (۲-۲) دو نمونه از این تغییرات را نشان می‌دهد.

شکل ۲-۲- محو شدگی-درون

- **محو شدگی-برون**[2]: این نوع تغییر تدریجی، نیز یک تغییر تدریجی در روشنایی است که از فریم مورد نظر شروع می‌شود و به یک فریم ثابت (معمولا سیاه) ختم می‌شود. شکل (۳-۲) دو نمونه از این تغییرات را نشان می‌دهد.

شکل۳-۲- محوشدگی-برون

- **حل شدن**[1]: در این تغییر، تصاویر نمای اول تیره‌تر شده و تصاویر نمای دوم روشن‌تر می‌گردند. در داخل این انتقال، تصاویر یک نما برروی نمای دیگر نمایش داده می‌شوند.

[1] *Fade-in*
[2] *Fade-out*

تعریف دیگری که می‌توان از حل شدن ارائه نمود این است که: حل شدن عبارت است از توالی محوشدگی-برون بعد از محوشدگی-درون. شکل (۴-۲) دو نمونه از حل شدن را نشان می‌دهد.

شکل ۴-۲- حل شدن

- **پاک شدن**[۲]: این تغییر هنگامی رخ می دهد که پیکسلها از نمای دوم جایگزین پیکسلها در نمای اول در یک الگوی خاص(مثلا در یک خط) می‌شوند. شکل(۵-۲) این تغییر را نشان می‌دهد.

شکل۵-۲- پاک شدن

[1] *Dissolve*
[2] *Wipe*

تا کنون اهمیت تشخیص تغییر نما و انواع تغییرات نما بیان گردید. اما همواره، در مسیر تشخیص تغییر نما، مشکلاتی وجود داشته است. قبل از بررسی روشهای موجود در زمینه تشخیص تغییر نما، تعدادی از این مشکلات را بیان می‌کنیم.

در تشخیص تغییرات تدریجی، سه مشکل اصلی وجود دارد*[۴]*:

۱- تغییرات تدریجی دارای ۴ نوع مختلف می‌باشند.

۲- طول تغییرات تدریجی، متغیر است.

۳- حرکت دوربین و حرکت اشیا سبب می‌شود سیگنالی شبیه سیگنال تغییرات تدریجی تولید گردد.

یا در تشخیص تغییرات نما در کلیپ‌های ورزشی، مخصوصا فوتبال، ما دو مشکل عمده خواهیم داشت*[۱۰]*:

۱- فریم‌های قبل و بعد از تغییر سریع نمای بلند و تغییر سریع نمای متوسط، دارای هستوگرام رنگ شبیه به هم بوده و نسبت رنگ غالب یکسان دارند. در شکل (۶-۲)، نمای بلند و نمای متوسط نشان داده شده‌اند.

الف ب

شکل۶-۲- الف)نمای بلند ب)نمای متوسط

۲- بر اثر حرکت سریع دوربین، محتوای بصری همانند تغییرات تدریجی تغییر می‌کنند که سبب تشخیص اشتباه می‌شود. شکل (۷-۲)، حرکت سریع دوربین که معمولا با تغییرات تدریجی اشتباه گرفته می‌شود، را نشان می‌دهد.

در *[۱۱]*، دلیل اصلی مشکل بودن تشخیص مرز نمای اتوماتیک، اشتباه گرفته شدن هر نوع گذر نما با حرکت شیء یا دوربین در ویدیو بیان شده است.(در این مرجع، این موضوع را در تمام ویدیوها در نظر گرفته است نه فقط ویدیوهای ورزشی).

شکل ۷-۲- حرکت سریع دوربین که معمولا با تغییرات تدریجی اشتباه گرفته می‌شود

در مرجع *[۹]*، عوامل تشخیص اشتباه را در سه گروه آورده است:

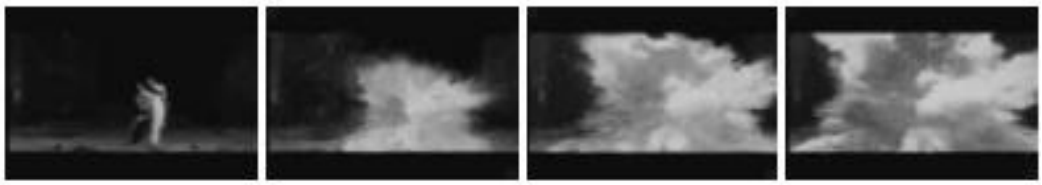
۱- **تغییرات روشنایی**[1]**:** در فیلم برداری، نورپردازی سبب می‌شود که تصاویر بطور متفاوتی به نظر برسند. مثلا در شکل(۸-۲)، نورپردازی سبب می‌شود هنرپیشه زن کاملا متفاوت به نظر برسد. این امر برای انسان، یک امر طبیعی است ولی الگوریتم‌های تشخیص تغییر نما را گیج می‌کند.

[1] *Illumination Changes*

شکل۸-۲- نورپردازی و متفاوت به نظر رسیدن هنرپیشه زن

۲- اثرات دوربین: این اثرات شامل زوم کردن، حرکت سریع شیء، حرکت سریع دوربین و ... می‌باشد که نمونه‌ای از حرکت سریع دوربین، در شکل(۷-۲) نشان داده شده است.

۳- اثرات ویژه: اثراتی مانند نشان دادن انفجار، یا نمایش تصویر در تصویر از این نوع می‌باشند. این اثرات در شکل‌های (۹-۲) و (۱۰-۲) نشان داده شده‌اند.



شکل ۹-۲- نمایش انفجار در دوربین

شکل۱۰-۲- نمایش تصویر در تصویر

با توجه به موارد بیان شده، می‌توان به این نتیجه رسید که مشکلاتی که در تشخیص مرز نما مکرراً با آنها مواجه می شویم، عبارتند از :

۱-هنگامی که آخرین فریم قبل از تغییر و اولین فریم بعد از تغییر، هر دو پس زمینه‌های یکسانی داشته

باشند، ویژگی‌های آنها، هیستوگرام‌های رنگ ناحیه‌ای و نسبت‌های رنگ غالب در آنها خیلی شبیه خواهد بود و این مسئله سبب می شود که تغییر تشخیص داده نشود.

۲-در حرکت سریع دوربین که معمولا برای تعقیب یک بازیکن در حال دو با نمایی نزدیک مورد استفاده قرار می‌گیرد، هیستوگرام‌های رنگ ناحیه‌ای به همان سرعتی که این نوع هیستوگرام‌ها در تغییر تدریجی تغییر می‌کنند، تغییر خواهند کرد و این امر سبب تشخیص اشتباه می‌گردد.

۲-۲-اجزای الگوریتم‌های تشخیص تغییر نما

الگوریتم‌های تشخیص مرز نما، با استخراج یک یا چند ویژگی از فریم ویدیویی یا یک زیر مجموعه از آن، که ناحیه‌ی جذاب [1] نامیده می شود، عمل می‌کنند. سپس، یک الگوریتم می‌تواند از روش‌های مختلف برای تشخیص مرز نما از روی این ویژگیها استفاده کند. در زیر اجزای الگوریتم‌های تشخیص مرز نما به همراه مزیت‌ها و معایب آنها آورده شده است:

۱-۲-۲- ویژگی‌های استفاده شده

تقریبا تمام الگوریتم‌های تشخیص تغییر نما، ابعاد بزرگِ (دیمانسیون بزرگ) حوزه ویدئو را، با استخراج تعدادی از ویژگی‌ها از یک یا چند ناحیه جذاب در هر فریم ویدئویی، کاهش می‌دهند. برخی از این ویژگیها عبارتند از:

الف- روشنایی/رنگ

ساده‌ترین ویژگی که می‌تواند برای توصیف یک ناحیه جذاب مورد استفاده قرار گیرد، میانگین روشنایی سطح خاکستری آن است. این ویژگی به تغییرات شدت روشنایی حساس است. انتخاب بهتر آن است که

[1] *Region Of Interest=ROI*

از یک یا چند آماره‌ی مقادیر (مانند میانگین‌ها) در یک فضای رنگ مناسب استفاده کنیم.[۱۲]، [۱۳]،[۱۴]

ب-هیستوگرام روشنایی/رنگ

یک ویژگی قویتر برای ناحیه جذاب، هیستوگرام رنگ یا هیستوگرام سطح خاکستری است. این ویژگی تقریباً تفکیک کننده است. براحتی قابل محاسبه بوده و اغلب نسبت به حرکت دوربین و زوم کردن غیر حساس است به این دلایل بطور وسیعی مورد استفاده قرار می‌گیرد.[۱۵]،[۱۶]

ج-لبه‌های تصویر

یک انتخاب واضح از گزینش ویژگیها، اطلاعات لبه در ناحیه جذاب می‌باشد. لبه‌ها می‌توانند آن طوری که هستند استفاده گردند یا با اشیاء ترکیب شوند یا برای استخراج آماره‌های ناحیه علاقه مورد استفاده قرار گیرد. لبه ها نسبت به تغییرات شدت روشنایی غیرمتغیرند و تا حدی به ادراک بینایی انسان مربوط می‌شوند. اصلی ترین عیب‌های این روش هزینه محاسباتی، حساسیت به نویز و دیمانسیون بالا می‌باشد. [۱۴]، [۱۷]،[۱۸]

د-ضرایب تبدیل (تبدیل موجک،تبدیل فوریه،تبدیل کسینوسی)

این موارد یک روش کلاسیک برای توصیف اطلاعات تصویر در یک ناحیه علاقه می‌باشند . ضرایب *DTC* این مزیت را دارند که در فایل‌ها یا جریان‌های ویدئویی کدگذاری شده *MPEG* نیز حضور دارند. بزرگترین مشکل این است که عموماً نسبت به زوم غیر متغیر نیستند .[۱۸]،[۱۹]،[۲۰]،[۲۱]،[۲۲]

ه- حرکت (بردار حرکت)

گاهی اوقات حرکت بعنوان یک ویژگی برای تغییرات نما استفاده می‌شود. اما معمولاً با ویژگی های دیگر ترکیب می‌گردد زیرا حرکت به تنهایی می‌تواند در داخل یک نما، به شدت ناپیوسته باشد. همچنین واضح است که اگر در ویدیو حرکتی نباشد، این ویژگی بلا استفاده است.*[۱۸]*،*[۲۳]*

و- آماره‌ها

در این روش، تصویر به چند ناحیه تقسیم می‌شود و آماره‌های پیکسلها در آن نواحی، محاسبه می‌گردند. معمول‌ترین آماره‌ها میانگین، واریانس و انحراف معیار می‌باشند. این روش نسبت به نویز غیر حساس است، اما به دلیل پیچیدگی فرمول‌های آماری کندتر عمل می‌کند.*[۲۴]*

ی- ویژگی‌های چندگانه

بیشتر الگوریتم‌ها چندین نوع از ویژگی‌ها را یا برای ترکیب کردن و یا پردازش و آنالیز بعدی استخراج می‌کنند.

۲-۲-۲-حوزه ویژگی فضایی

سایز ناحیه‌ای که ویژگی‌های منحصر به فرد از آن استخراج می‌شوند، نقش مهمی در عملکرد تشخیص تغییر نما دارد. نواحی کوچک تغییر ناپذیری تشخیص نسبت به حرکت را کاهش می‌دهد در حالی که نواحی بزرگ به سمت تشخیص ندادن تغییرات در بین نماهای مشابه متمایل می‌شوند .

الف- پیکسل تنها

برخی از الگوریتم‌ها به ازای هر پیکسل از تصویر یک ویژگی استخراج می‌کنند. این ویژگی می‌تواند روشنایی، قدرت لبه یا چیزهای دیگر باشد*[۱۴]*. به هر حال چنین روشهایی بردار ویژگی‌های بزرگی را

تولید می‌کنند و نسبت به حرکت حساس می‌باشند، مگر اینکه در مرحله‌ی بعد جبران حرکت صورت گیرد.

ب-بلوک مستطیلی

روش دیگر تقسیم کردن هر فریم به بلوک‌هایی با سایز مساوی و استخراج کردن یک سری ویژگی ها(مانند میانگین رنگ یا جهت،هیستوگرام رنگ) از هر بلوک می باشد*[۱۲]*،*[۱۳]*. این روش نسبت به حرکت های کوچک دوربین و شی تغییر ناپذیر است و می‌تواند مشخص کننده مرز نما باشد.

ج-ناحیه با شکل دلخواه

استخراج ویژگی با بکار بردن نواحی با شکل و سایز دلخواه امکان پذیر است. این روش از بیشترین نواحی هم‌شکل بهره‌برداری کرده و تشخیص بهتری از ناپیوستگی‌ها را امکان می‌سازد . استخراج ویژگی شیء-اساس[1] نیز در این گروه قرار می‌گیرد. معایب اصلی این روش، پیچیدگی محاسباتی بالا و ناپایداری، به سبب پیچیدگی الگوریتم‌های شامل شده می‌باشد .

د-کل فریم

الگوریتم‌هایی که از کل فریم استفاده می‌کنند، ویژگی‌ها را فوراً از کل فریم استخراج می‌کنند. چنین روشهایی این مزیت را دارند که نسبت به حرکت خیلی مقاوم هستند، اما در تشخیص تغییر بین دو نمای مشابه عملکرد ضعیفی دارند .

[1] *Object based*

۳-۲-۲-متریک شباهت ویژگی

به منظور ارزیابی ناپیوستگی بین فریم‌ها بر اساس ویژگی‌های انتخاب شده، نیاز است یک متریک شباهت/غیر شباهت انتخاب شود. فرض کنید یک فریم با K ویژگی اسکالر $F(k)$ مشخص شده باشد که در آن $k=1,2,...,K$ می باشد.سنتی ترین انتخاب میتواند استفاده از نرم زیر باشد.

$$D_{Ln}(i,j) = \sqrt[n]{\sum_{K=1}^{K}\left|F_i(k) - F_j(k)\right|^{n}} \qquad (۲-۱)$$

بعنوان مثال در اختلاف هسیتوگرامی، $F(k)$ بین‌های[1] هستوگرام و $n=1$ خواهد بود و در اختلاف تصویر، $F(k)$ تعداد پیکسل های تصویر خواهد بود و $n=2$.

۴-۲-۲- متریک‌های پیوسته حوزه‌های زمانی

صورت مهم دیگر الگوریتم های تشخیص مرز نما، پنجره زمانی از فریمهاست که برای تشخیص تغییرنما بکار می‌رود. این پنجره‌ها می‌توانند یکی از انواع زیر باشند:

الف- دو فریم

ساده ترین راه برای تشخیص ناپیوستگی، جستجو به دنبال مقدار بزرگ متریک ناپیوستگی بین دوفریم متوالی است[۱۳]،[۲۷]،[۲۵]،[۲۶]. به هر حال این روش هنگامی که تغییر قابل توجهی در فعالیت قسمتهای مختلف ویدئو وجود داشته باشد یا هنگامی که نماهای ویژه‌ای شامل رخدادهایی باشند که ناپیوستگی‌های کوتاه مدت را سبب می‌شوند، شکست می‌خورد. همچنین در تشخیص تغییرات تدریجی به مشکل بر می‌خورد.

[1] *bin*

ب- پنجره های N فریمی

متداول‌ترین روش برای کم کردن مشکلات بالا، تشخیص ناپیوستگی با استفاده از ویژگی‌های تمام فریم‌ها در داخل یک پنجره زمانی است[۱۳]،[۱۴]،[۲۸]،[۲۹]. این کار یا با محاسبه یک آستانه پویا که با یک متریک ناپیوستگی فریم به فریم مقایسه می‌شود، یا از طریق محاسبه متریک ناپیوستگی بطور مستقیم بر روی پنجره، انجام می‌گیرد.

ج-فاصله از آخرین تغییر نما

شاید روشی که برای آشکارسازی انتهای نما، بیشتر مشاهده شده است، محاسبه‌ی یک یا چند آماره از آخرین نمای آشکار شده تا نقطه فعلی و بررسی اینکه آیا فریم بعدی نیز شامل آن است یا خیر[۱۲]،[۱۶]. مشکل چنین روشهایی، این است که معمولا درون نماها تغییرات بزرگی وجود دارد، بدین معنی که آماره‌های محاسبه شده در داخل نما ممکن است نشان دهنده‌ی انتهای آن نباشند.

د- کل نمای فعلی

کاملترین روش این است که مشخصات تمام ویدئو را هنگامی که یک تغییر نما در حال آشکار سازی است، مورد بررسی قرار دهیم. دوباره مشکل این است که ویدئو می تواند در داخل و بین نماها تغییرپذیری بزرگی داشته باشد.

۵-۲-۲- روش تشخیص تغییر نما

با تعریف یک یا چند ویژگی محاسبه شده از یک یا چند ناحیه جذاب برای هر فریم، یک الگوریتم تشخیص تغییر نما برای آشکارسازی محل ناپیوستگی نشان داده شده توسط این ویژگیها نیاز است. در زیر تعدادی از این الگوریتم‌ها آورده شده است:

الف- آستانه گذاری

این روش به معنی مقایسه کردن مقدار ناپیوستگی با یک آستانه ثابت است[۱۶]. این روش خوب عمل می‌کند اگر محتوای ویدئویی در طول زمان، دارای حالت ثابت و بدون تغییر باشد و اگر آستانه برای هر ویدیو به صورت دستی تنظیم گردد .

ب- آستانه وفقی (پویا)

یک راه حل آشکار برای مشکلات آستانه گذاری ساده این است که آستانه بر طبق آماراه(مثل میانگین) متریک‌های اختلاف ویژگی در داخل یک پنجره زمانی تغییر کند[۲۷]،[۲۹].

ج- تشخیص احتمالاتی

یک روش سخت برای تشخیص تغییرات نما، مدل کردن الگویی از انواع ویژه‌ای ازتغییرات نما و بهینه انجام دادن یک تخمین تغییر نمای استقرایی با پیش فرض کردن توزیع‌های احتمالاتی خاصی برای نماها می باشد[۱۲]،[۱۳].

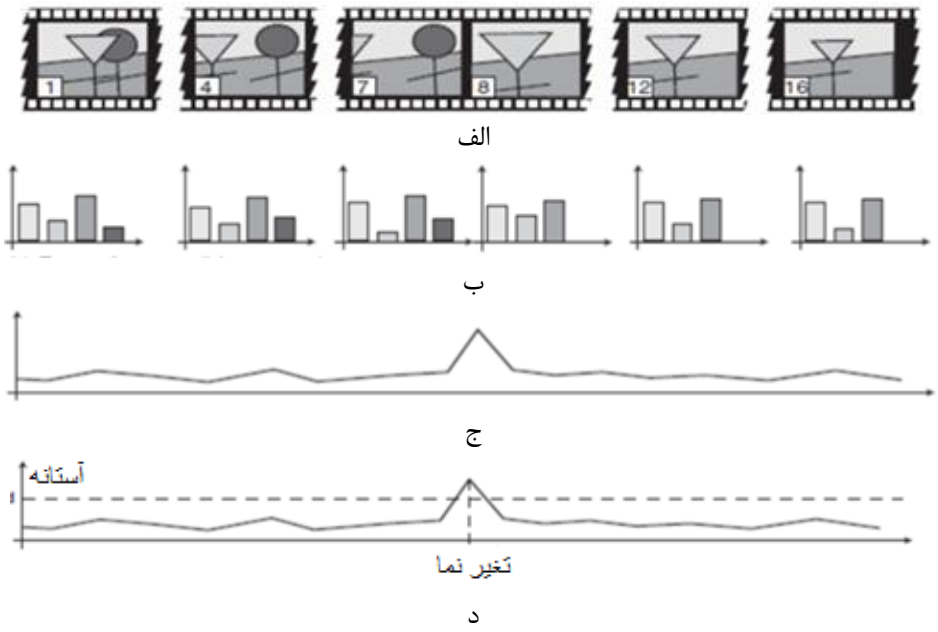
د- طبقه بند آموزش دیده

یک روش کاملا متفاوت برای آشکارسازی تغییرات نما، فرمولی کردن مسئله بعنوان یک مسئله طبقه بندی کردن است. طبقه‌ها،"تغییر نما" و "نبودن تغییر نما" خواهند بود[۲۸].

و-تعامل با کاربر

اگر فرایند تشخیص اتوماتیک شکست بخورد، آشکارسازی برش در موارد مبهم می‌تواند با استفاده از اعمال ورودی که توسط کاربر صورت می‌گیرد، انجام شود.

در شکل (۱۱-۲)،یک طرح کلی برای تشخیص تغییر نما ارائه شده است.



شکل ۱۱-۲- طرحی برای تشخیص تغییر نما (الف)فریم‌های ویدیو(ب)ویژگی فریم‌ها(ج)متریک ناپیوسته(د)تغیرنما

همان طور که در شکل فوق می‌بینید، ابتدا ویژگی‌ها استخراج شده‌اند (که در این روش هیستوگرام‌ها می‌باشند). سپس متریک ناپیوستگی را محاسبه می‌کنیم. در انتها مقدار متریک ناپیوستگی با آستانه مقایسه می‌شود و در صورتی که از آستانه تجاوز کند، تغییر نما تشخیص داده می‌شود.

۳-۲- ارزیابی عملکرد

برای ارزیابی کارایی الگوریتمهای آشکارسازی تغییر نما از دو پارامتر دقت[۱] و یادآوری[۲] مطابق با تعریف زیر استفاده می‌شود. در روابط زیر منظور از *Detect* تعداد نماهایی است که به درستی تشخیص داده و اعلان شده‌اند. منظور از*MD* [۳] تعداد مرزنماهای اعلام نشده است و منظور از *FA* [۱]تعداد مرزنماهایی است که به اشتباه به عنوان مرزنما اعلان شده‌اند.

[1] *Precision*
[2] *Recall*
[3] *Missed Detects*

$$\text{Recall} = \frac{\text{Detects}}{\text{Detects+MD's}} \qquad (۲-۲)$$

$$\text{Precision} = \frac{\text{Detects}}{\text{Detects+FA's}} \qquad (۲-۳)$$

برخی مولفان درکار خود پارامتر FA را در نظر نمی‌گیرند، که این کار در حالت کلی نادرست است زیرا در این صورت، روشی که هر فریم را به عنوان یک نمای جدید اعلان می‌کند بهتر از هر روشی عمل می‌کند که کمی محتاطانه عمل کرده باشد. از طرف دیگر از آنجا که تشخیص نما در کاربرد فشرده‌سازی ویدیو مورد نیاز است، اعلان بیش از حد تعداد نماها موجب افت در کارایی نمایش ویدیو و فشرده‌سازی آن می‌شود.

گاهاً علاقه‌مندیم که کارایی عملکرد در یک روش آشکارسازی مرزنما را با یک (و نه دو) عدد بسنجیم. در این صورت معمولاً دو معیار دقت و یادآوری با هم ترکیب شده و از یک معیار جدید به نام معیار هارمونیکی F استفاده می‌شود که برابر با میانگین هارمونیکی دقت و یادآوری تعریف می‌شود:

$$\text{F} = \frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision+Recall}} \qquad (۲-۴)$$

[1] *False Alarms*

۴-۲-الگوریتم های ویژه تشخیص تغییر نما

۱-۴-۲- تشخیص تغییر نما در ویدیوی فشرده *H.264/AVC*

این مقاله با مسألهٔ تشخیص تغییر نما روی ویدیوی فشرده شده با استاندارد *H.264/AVC* سروکار دارد. از آنجایی که *H.265/AVC* ابزار جدید کدینگ را بکار می‌برد، اطلاعات آماری ماکروبلوک[۱] از ویدیوی *MPEG* متفاوت است. در این مقاله، آنالیز نوع ماکرو بلوک بعلاوهٔ قانون محدودیت آماری درون حالت[۲] برای تشخیص تغییرات نما در ویدیوهای فریم *P* و فریم *B* از *H.264/AVC* پیشنهاد شده است. برای تشخیص تغییرات نما در ویدیوی کد شده فریم *I* ، یک هیستوگرام درون حالت پیشنهاد شده و از فاصله‌ی بلوکی وزن داده شده برای اندازه‌گیری شباهت بین فریمهای *I* استفاده کردیم. همبستگی بردارهای ضریب *DCT* برای اندازه گیری شباهت به فریم های *I* از دنباله های *MPEG* تعریف شده‌اند. گسستگیها[۳] در فریم های *B* با چک کردن تعداد ماکروبلوک های پیشرو و پسرو غیر صفر در فریم *B* آشکار می‌شوند. ضرایب *DC* می توانند مستقیماً از فریم *I* استخراج شوند و به سرعت از فریم *B* و فریم *P* بازسازی گردند. بر اساس تصویر *DC* ، الگوریتم‌هایی که برای ویدیوی غیرفشرده طراحی شده‌اند میتوانند مستقیماً به ویدیوی فشرده انتقال داده شوند. اخیراً تکنیک آنالیز ترتیبی آماری برای تشخیص تغییر در صحنه در دنبالهٔ تصویر *DC* ارائه شده است. گوشه‌ها برای هر ماکروبلاک نیز می‌توانند با آنالیز ضرایب *AC* استخراج شده و برای تطابق فریم‌ها با جهت و قدرت گوشه در آشکار سازی تغییر نما بکار روند. در کنار ضرایب *DC*، بردار حرکت *MB* از بین فریم های متوالی *P* برای تشخیص حرکت دوربین، استخراج شده است. این روش، بطور مؤثری می‌تواند تغییرات صحنهٔ واقعی را از آن دسته تغییرات که توسط حرکت دوربین ایجاد

---

[1] *Macro Block*
[2] *Intra-Mode*
[3] *Breaks*

شده است، مجزا کند. تاکنون، تعداد کمی از عملیات آشکار سازی بر روی ویدیوی فشرده *H.264/AV* فراهم شده است. در استاندارد جدید، حالت پیش بینی درون فریمی[1] برای کد کردن *MB* بعنوان نوع درونی[2] معرفی شد. علاوه بر آن، هر *MB* می‌تواند به بلوک با سایز متغیر در جبران سازی حرکت تقسیم بندی شود. در جبران سازی حرکت چندین فریم مرجع در نظر گرفته شده است. بعضی از ماکروبلاک‌ها با بافت هموار ممکن است با درون‌حالت کدگذاری شوند. حتی اگر جبران سازی حرکت موفق باشد، به منظور کارایی آشکارسازی تغییرات نما، مجموعه‌ای از الگوریتم‌ها را برای بررسی مشکل آشکارسازی تغییرنما در ویدیوی فشرده *H.264/AVC* مطرح شده است. برای ویدیو با *I* فریم کد شده، فاصله هیستوگرام درون‌حالتی را برای توصیف تغییر محتوای فریم های *I* طراحی گردید. برای آشکارسازی تغییر نما در طی فریمهای *P* و *B*، ما از جهت مرجع حرکت برای هر *MB* برای تصمیم گیری وابستگی زمانی بین فریم *B/P* و فریم مرجع آن استفاده می‌کنیم. از آنجایی که وابستگی زمانی در ویدیوی *H.264/AVC* ضعیف است، قانون محدودیت آماری درون حالتی برای کاهش اعلان‌های غلط بکار رفته است*[۳۰]*.

۲-۴-۲- تشخیص نما بصورت بلادرنگ در حوزهٔ فشرده شده توسط *MPEG*

الگوریتم های مرسوم برای جزء بندی کردن ویدیو، اساساً برروی آنالیز ویژگی‌های ویدیوی فشرده شده تمرکز می‌کند، زیرا اطلاعات مربوط به پردازش جزء بندی[3] می‌تواند به طور مستقیم از فایل فشرده شده *MPEG* استخراج گردد و برای تشخیص مرزهای نما مورد استفاده قرار گیرد. هرچند بیشتر الگوریتم‌های

[1] *Intra-Prediction*
[2] *Intra-Type*
[3] *Partitioning*

مطرح شده قابلیت های بلادرنگ را که برای کاربردهای ویدیویی ضروری هستند را نشان نمی‌دهند. این مقاله یک الگوریتم زمان حقیقی برای تشخیص برش را نشان می‌دهد. این روش آماره‌های ویژگی‌های استخراج شده از فایل فشرده توسط *MPEG* را مورد بررسی قرار می‌دهد، مانند نوع ماکرو بلوک، و متریک‌ها و اندازه های یکسانی را برای الگوریتم های تشخیص تغییر تدریجی تعمیم می‌دهد. از آنجایی که تعیین ساختارهای زمانی ویدیو در اندیس گذاری ویدیویی و بازیابی، کار مهمی است، تشخیص نما به عنوان اولین مرحله پیاده سازی الگوریتم اندیس‌گذاری و مرحله‌ای ضروری در این امر مورد پذیرش واقع شده است. "نما" به عنوان یک دنباله از فریم‌ها که بطور پیوسته از یک دوربین گرفته شده باشد یا به نظر برسد این طور تهیه شده باشد، تعریف می‌شود. صحنه بعنوان مجموعه ای از یک یا چند نمای مجاور که روی یک یا چند شیء مورد نظر تمرکز دارند، نامیده می‌شود. الگوریتم های تشخیص تغییر نما را می‌توان نسبت به ویژگی‌هایی که برای پردازش استفاده می‌کنند، به دو دسته ی الگوریتمهای حوزه فشرده و غیرفشرده تقسیم کرد. الگوریتمهایی که در حوزه‌ی غیرفشرده مورد استفاده قرار می‌گیرند، مستقیماً از اطلاعات حوزه‌ی فضایی ویدیو استفاده می‌کنند: تفاوت پیکسلی، هیتسوگرام‌ها، تعقیب لبه و ... این تکنیک‌ها از نظر محاسباتی طاقت فرسا و از نظر زمانی، هدر دهندهٔ زمان هستند بنابراین نسبت به روش‌هایی که بر اساس ویژگی های فشرده شده می‌باشند، در درجهٔ دوم قرار دارد. در حوزهٔ فشرده شده، یک روش استفاده از دنبالهٔ *DC* است، این دنباله، یک دنباله از تصاویر کاهش یافته می باشد که از ضرایب *DC* استخراج شده است. در این مقاله، هدف اصلی این مقاله، ارائه ی روشی مناسب برای بازیابی و جستجوی ویدیو بطور زمان حقیقی می‌باشد. الگوریتم مطرح شده، بالاترین دقت را در تشخیص تغییرات سریع نشان می‌دهد در حالی که تعداد کمی از تلاشهایی که برای اجرای تشخیص سریع تغییرات تدریجی انجام شده است، نتایج جالبی را ارائه می‌دهند. ویژگی های حرکت استخراج شده از *MPEG* ناپایداری بالا را نشان می‌دهد، در حالی که اندازه‌گیریِ تغییرات طولانی‌تر نامعلوم و مبهم است[۳۱].

۳-۴-۲- تشخیص تدریجی تغییرنما در ویدئوهای فوتبال توسط فرکتلها

تشخیص دقیق تغییرنما، نقش مهمی را در تقسیم کردن فایلهای ویدئویی به بخشهای معنی‌دار برای آنالیز صحنه‌های ویدئویی بازی می‌کند . تغییرات نما ممکن است بر اثر عبور ناگهانی یا تدریجی نما رخ دهد. در تغییر سریع نما، تغییر مضمون ویدئویی در یک فریم تنها رخ می‌دهد. در تغییر تدریجی نما، تغییر مضمون بطور تدریجی و در طی دنباله های کوتاهی از فریم ها رخ می‌دهد. تغییر تدریجی هم چنین بر زیرگروه‌هایی مانند حل شدن، بزرگ نمایی، پاک شدن و محو شدن تقسیم می‌شود . درتحقیقات، روش‌های زیادی برای تشخیص تغییر نما ارائه شده است . در حالی که این روشها کاملاً تغییرات سریع را تشخیص می‌دهند اما به درستی نمی توانند تغییرات تدریجی را تشخیص دهند. این مقاله روشی را برای تشخیص موفق تغییر نمای تدریجی ارائه می دهد. این روش از اطلاعات دیماسیون فرکتالی فریم‌های ویدئویی مقیاس سطح خاکستری استفاده می کند. در عمل، میزان تشخیص تغییر نما براساس فرکتال بر روی ویدئوهای فوتبال که شامل انواع مختلفی تغییرتدریجی نما هستند، آزمایش شده است و با روش‌های مشهور تغییر نما مانند پیکسل- اساس و هیستوگرام مقایسه شده است. نتایج نشان می‌دهد که الگوریتم مطرح شده دقت بالاتری نسبت به روش های دیگر در بیشتر موارد دارد*[۳۲]*.

۴-۴-۲- تشخیص تغییر نما با استفاده از مقیاس زمان چندگانه

آزمایشات تشخیص مرز نما را برای یک دنبالۀ ویدئویی با استفاده از الگوریتم تشخیص نمای با مقیاس زمان چند گانه توضیح می‌دهیم. هدف الگوریتم این است که انواع شکستگی در ویدئو را با توجه به تفاوت فریم‌ها در یک رنج مقیاس زمانی، تشخیص و آنها را از هم جدا کند. بررسی کردن رنج وسیعتری از فریم‌ها نسبت به آنهایی که در مجاورت هم قرار دارند، ما را قادر می سازد تا تغییرات تدریجی مانند محو شدن و حل شدن را تشخیص دهیم در حالی که باعث می‌شود تغییرات زودگذر مانند آنچه دراثر فلاش‌های

دوربین‌های عکاسی در تصاویر ویدئویی رخ می‌دهد، حذف شوند. در حالی که عملکرد سیستم‌ها متعادل بود، کمی بهتر از میانگین عمل کردند، چند مسئله مشهود بود: واضح است تعداد دیگری تفکیک کننده علاوه بر رنگ نیز موجود است مثلا یک هدف که بتواند در سرتاسر مرز نمای فرض شده دنبال شود این مرز می‌تواند نزول پیدا کند. تفکیک کننده‌های دیگری مانند بافت[1] یا شکل[2] نیز می‌تواند مفید باشند . مواردی وجود داشتند که بطور تجربی تعیین کردند آستانه‌های تعیین شده در همه موارد بهینه نیستند، از این رو پیشنهاد می‌شود که در برخی موارد تنظیم خودکار آستانه به کار رود. به هر حال این امر ممکن است بعد از دوبار عبور دیتا نیاز باشد*[۳۳]*.

۵-۴-۲- تشخیص تغییر نما توسط یک ویژگی جدید هیستوگرام در ویدئوی فشرده

در این مقاله یک روش موثرتشخیص تغییر نما برای فایلهای ویدئویی فشرده *MPEG* با استفاده از یک ویژگی برتر هیستوگرام که با کانولوشن متفاوت است، ارائه گردیده است. این الگوریتم براساس روش هیستوگرام کوانتیزاسیون تفاوت شدت پیکسلهای مجاور *(APIDQ)*[3] می‌باشد، که این روش بطور قابل اعتمادی در مورد بازشناسی تصویر صورت انسان بکاررفته است . الگوریتم مطرح شده توسط دیتابیس *MPEG* مورد ارزیابی قرار گرفته است . نتایج عملی نشان می‌دهد که روش هیستوگرام *APIDQ* دقیق‌تر و قویتر از روشهای کانولوشن می‌باشد. ما یک الگوریتم تشخیص تغییر نما که سریع و قابل اعتماد می‌باشد را توسط آستانه وفقی براساس روش هیستوگرام کوانتیراسیون تفاوت شدت پیکسل‌های مجاور در فایلهای *MPEG* گسترش دادیم. در مقایسه با روش های کانولوشن، الگوریتم مطرح شده می‌تواند تغییر نما را بصورت دقیقتری آشکار کند. علاوه بر این چون الگوریتم‌های کانولوشن و روش مطرح شده در این مقاله از

[1] *Texture*
[2] *Shape*
[3] *Adjacent Pixel Intensity Difference Quantization*

ویژگیهای متفاوتی استفاده می‌کنند، می‌توان با ترکیب این روش‌ها الگوریتم قویتری در آینده ارائه داد[۳۴].

۵-۲- مسائل مطرح شده در تشخیص تغییر نما

با توجه به مسائل و روش‌ها مطرح شده، مشخص می‌شود که تشخیص مرز نما بین دو فریم متوالی، نیازمند محاسبه‌ی یک متریک پیوستگی(در بعضی مراجع،ناپیوستگی) یا متریک شباهت(در برخی مراجع،غیرشباهت) می‌باشد. به هر حال این مفهوم ساده درای سه پیچیدگی می‌باشد:

۱- اولین پیچیدگی موجود در مورد تعریف یک متریک پیوسته برای ویدیو است بصورتی که تغییرات تدریجی در پارامترهای دوربین (مانند زوم کردن،حرکت سریع و ...)، نورها و... غیرحساس بوده و همچنین این متریک به سادگی قابل محاسبه و جداکننده باشد. ساده‌ترین راه برای انجام آن، استخراج یک یا چند ویژگی اسکالر یا برداری از هر فریم و تعریف تابعهای فاصله برروی حوزه ویژگی است. ویژگیها می‌توانند برای خوشه‌بندی فریمها به نماها یا تشخیص الگوهای گذر نما، مورد استفاده قرار بگیرند.

۲-دومین پیچیدگی، مربوط به تعیین این موضوع است که کدام مقدار متریک وابسته به تغییر نما مربوط می‌شود و کدام مقدار مربوط نمی‌شود. این امر، یک بحث کوچک و جزئی نیست زیرا تغییرات ویژگی در داخل نماهای ویژه می‌تواند از تغییرات مربوطه در طی نماها فراتر روند. روشهای تصمیم‌گیری برای تشخیص مرز نما شامل آستانه‌های ثابت، آستانه وفقی و روش‌های تشخیص آماری است.

۳- سومین پیچیدگی مربوط به این موضوع است که تمام تغییرات نما، از نوع ناگهانی نیستند و بعضی تغییرات بصورت تدریجی می‌باشند. تشخیص تعیین نوع تغییر از مهمترین و در عین حال مشکلترین موضوعات، در تشخیص تغییر نما می‌باشد.

با توجه به روشهایی که در زمینه تشخیص تغییرات تدریجی وجود دارد، می‌توان نتیجه گرفت که الگوریتم های موجود در این زمینه، به دو گروه عمده تقسیم می‌شوند:

۱- روشهای متحد شده، یعنی یک آشکارساز برای تمام انواع گذر تدریجی.

۲- روشهایی که برای هرگونه تغییر تدریجی، یک آشکارساز مخصوص ارائه داده اند.

در مرجع *[۳۵]*، بیان شده که یک سیستم تشخیص مرزنما را دارای سه مرحله‌ی پردازشی است:

۱- نمایش محتوای بصری

۱-۵- منظور از نمایش محتوای بصری، استخراج ویژگی است. ویژگی‌هایی که نسبت به حرکت دوربین و شیء یا نسبت به حرکت شیء و اثرات ویدیویی غیرحساس باشند. می‌توان از ویژگی‌هایی مانند هیستوگرام رنگ یا هیستوگرام روشنایی نام برد.

۲-ساخت یک سیگنال پیوسته

در این مرحله، ویژگیهای فریم‌های متوالی برای گرفتن اطلاعات در مورد تغییرات سیگنال را مقایسه می‌کنیم. این تغییرات، تغییرات نما یا تغییرات مهم دیگر در فیلم را آشکار می‌کنند.

۳-طبقه بندی

۱-۶- مشخص کردن اینکه تغییر تدریجی است یا ناگهانی و اینکه هر فریم شامل تغییر تدریجی است یا خیر.

در برخی از مراجع، ویژگی های مطرح شده در قسمت(۲-۲) را به دو گروه عمده تقسیم می‌کنند:

- ویژگیهای غیر فشرده

۱-۷- مانند اختلافات پیکسلی، اختلافات آماری و اختلافات هیستوگرامی

- ویژگیهای فشرده

۱-۸- مانند بردارهای حرکت و اختلافات تراکمی

از دیگر نتایجی که می‌توان با توجه به روشهای ارائه شده بدست آورد، این است که هر چند روشهای زیادی در مورد آشکارسازی مرز نما وجود دارد، اما به سختی می‌توان این روشها را مقایسه کرد. زیرا اولاً جزئیات کامل پیاده‌سازی سیستم همیشه منتشر نمی‌شود واین امر بازسازی را مشکل می‌کند. دوماً بیشتر سیستمها روی دنباله‌های کوچک و همگن ویدیویی ارزیابی شده‌اند. این نتایج به خوبی نشان نمی‌دهند که این سیستمها در گستره بزرگتر و وسیعتر از انواع محتوای ویدیویی، چگونه عمل می‌کنند.

موضوع دیگری که از اهمیت خاصی برخوردار است، موضوع انتخاب آستانه می‌باشد. بیشتر روشهای موجود از آستانه های عمومی از پیش تعریف شده یا از آستانه‌های وقفی بر اساس پنجره محلی ثابت استفاده می‌کنند. آستانه‌ی عمومی بطور قطع کافی نیست زیرا خاصیت ویدیو می‌تواند در هنگام تغییر محتوا، تغییر کند و معمولاً غیرممکن است که روش آستانه‌گذاری جامع و بهینه‌ای پیدا شود. همچنین آستانه‌ی عمومی محدودیتهایی دارد زیرا در برخی موقعیتها، آماره‌های محیطی توسط نویزهای قوی مانند

حرکات و نورهای فلش آلوده می‌شوند. از نکات دیگری که در مورد آستانه ها وجود دارد این است که اگر مقدار آستانه خیلی پایین انتخاب گردد، سبب می‌شود نماهایی مشخص گردند که وجود ندارند (یعنی سبب تشخیص اشتباه می‌شود) و اگر مقدار آستانه بالا انتخاب شود، سبب می‌شود برخی نماها تشخیص داده نشوند.

با توجه به نکات ذکر شده در این قسمت و مروری که بر روشها در قسمت های قبل صورت گرفت، می‌توان گفت که فقط یک نوع ویژگی و/یا متریک شباهت به اندازه ی کافی برای آشکار سازی گذرهای نما کافی نیستند. علاوه بر آن انتخاب پنجره بر دقت تشخیص مرز نما اثر می‌گذارد. در نهایت، بیشتر الگوریتمها نسبت به آستانه‌ی استفاده شده بر متریک شباهت/فاصله حساس هستند. همچنین اکثر روشها دارای خاصیت اجرای بلادرنگ نمی باشند یا دارای دو مشکل عمده هستند:

اول اینکه حجم محاسباتی خود را در نظر نگرفته اند و فقط به کارایی عملکرد توجه داشته‌اند. دوم اینکه هر دو گذر ناگهانی و تدریجی را آشکار سازی نکردند و فقط برروی یک گذر(غالباً گذر ناگهانی)، تمرکز داشته‌اند.

در این پایان‌نامه سعی شده است که مشکلات مطرح شده، برطرف گردد.برای تعیین گذر نما، از چهار ویژگی آماری، هیستوگرام، لبه و مرکز ثقل استفاده شده است. همچنین ویژگیها در هر مرحله نرمالیزه گردید. برای تایید این مطلب که الگوریتم ارائه شده، دارای حجم کم و بلادرنگ می‌باشد، این الگوریتم در بستر سخت افزار پیاده سازی گردید.

در نهایت ذکر این نکته حائز اهمیت است:

"طبق دانسته های ما، تا کنون در مورد پیاده سازی الگوریتم های تغییر نما در بستر پردازشگر های سیگنال کار خاصی صورت نگرفته است".

فصل سوم

روش پیشنهادی و نتایج آزمایشات

۱-۳-توضیحاتی در مورد *TMS320C5505*

این دستگاه، یک عضو از خانواده ممیز-ثابت[1] *TMS320C5000* می‌باشد و برای کاربردهایی با توان مصرفی پایین [2] طراحی شده است. پردازنده‌ی سیگنال دیجیتال ممیز ثابت *TMS320C5505* بر پایه نسل پردازنده های *TMS320C55x* می‌باشد. معماری *DSP* سری *C55x* با افزایش موازی کردن و کاهش تلفات توان، به عملکرد بالا و مصرف توان کم دست یافته است. این *CPU* از یک ساختار باس داخلی متشکل از: یک باس برنامه، سه باس برای خواندن داده (یک باس ۳۲ بیتی و دو باس ۱۶ بیتی)، دو باس نوشتن داده (دو باس ۱۶بیتی) و باسهای اضافی که به دستگاه‌های جانبی و فعالیت *DMA*[3] اختصاص داده شده است، پشتیبانی می‌کند. این باسها، توانایی انجام خواندن حداکثر چهار داده(۱۶ بیتی) و نوشتن دو داده(۱۶بیتی) در یک سیکل را برای *CPU* فراهم می‌کند. این دستگاه دارای چهار کنترل کننده *DMA* می‌باشد که هر کنترل کننده، چهار کانال دارد. این بدین معنی است که دستگاه می‌تواند بدون فعالیت *CPU*، از طریق ۱۶ کانال مستقل، اطلاعات را جا به جا کند.کنترل کننده *DMA*، می‌تواند جدا از فعالیت *CPU* در هر سیکل، ۳۲ بیت داده را منتقل کند. *CPU* سری *C55x* دارای دو واحد ضرب و جمع[4] است که هر کدام قادرند در یک سیکل،ضرب ۱۷ بیتی در ۱۷ بیتی و یک جمع ۳۲بیتی را انجام دهند. همچنین دارای یک واحد محاسبه و منطق[5] مرکزی ۴۰ بیتی است که توسط یک واحد محاسبه و منطق ۱۶بیتی اضافی پشتیبانی می‌شود. استفاده از واحدهای محاسبه و منطق، تحت کنترل مجموعه دستورالعمل‌ها می‌باشد و استفاده از آنها، توانایی بهینه سازی فعالیت موازی و مصرف توان را فراهم

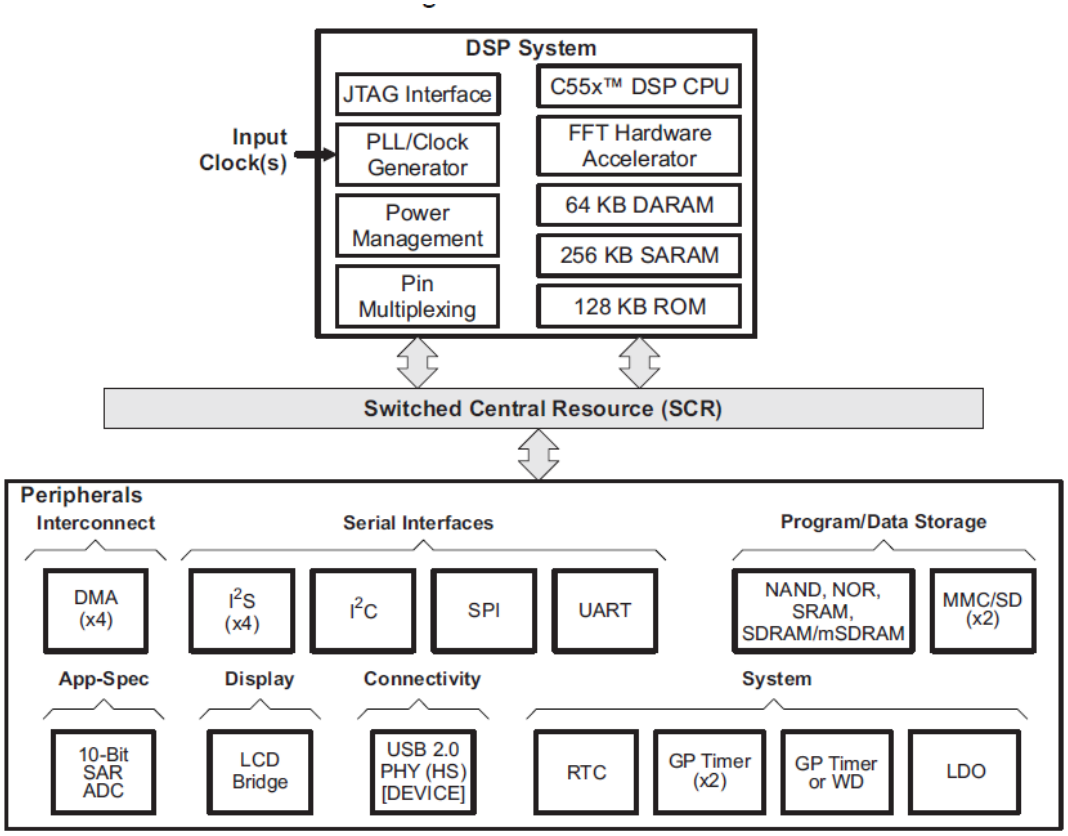
---

[1] *Fixed-point*
[2] *Low Power*
[3] *Direct Memory Access*
[4] *MAC=Multiply-Accumulate*
[5] *Arithmetic/Logic Unit*

می‌کند. این تواناییها در واحد آدرس[۱] و واحد دیتا [۲] از *CPU* مدیریت می‌شوند. *CPU* سری *C55x* برای بهبود چگالی کد[۳] از مجموعه دستورالعملها با طول متغیر استفاده می‌کند. واحد دستورالعمل[۴] عملیات واکشی[۵] برنامه ۳۲ بیتی از حافظه داخلی یا خارجی را انجام می‌دهد و دستورالعملها را برای واحد برنامه[۶]، به ترتیب در صف قرار می‌دهد. واحد برنامه دستورالعملها را رمزگشایی کرده و وظایف را به سمت واحدهای آدرس و برنامه هدایت و خط لوله را مدیریت می‌کند. توانایی پیش‌بینی دستورات پرش، از خالی شدن خط لوله به هنگام اجرای دستورالعملهای شرطی جلوگیری می‌کند. مجموعه دستگاه‌های جانبی شامل یک رابط حافظه خارجی[۷] می‌باشند که دسترسی به حافظه‌های آسنکرون مانند *NOR*،*NAND*،*EPROM* و *SRAM* را به همراه حافظه‌های پرسرعت مانند *DRAM* سنکرون[۸] و *SDRAM* مربوط به موبایل[۹] را فراهم می‌کند. اجزای جانبی دیگر شامل: باس پرسرعت عمومی سریال[۱۰] و کلاک بلادرنگ[۱۱]. این دستگاه همچنین، شامل سه تایمر همه کاره با یک پیکربندی و حلقه قفل شونده فاز آنالوگ[۱۲] می‌باشد. سه پورت سریال چندکاناله[۱۳] ارتباط دوطرفه با وسایل استاندارد و ارتباط چند کاناله(تا ۱۲۸ کانال) را فراهم می‌کنند. علاوه بر آن، این دستگاه شامل یک شتاب‌دهنده‌ی سخت‌افزاری *FFT* نیز می‌باشد.در شکل(۱-۳)،نمایش بلوکی از *TMS320C5505* نشان داده شده است.

---

[1] *AU=Address Unit*
[2] *DU=Data Unit*
[3] *Code Density*
[4] *Instruction Unit=IU*
[5] *Fetch*
[6] *Program Unit=PU*
[7] *External Memory Interface=EMIF*
[8] *synchronous DRAM=SDRAM*
[9] *Mobile SDRAM=mSDRAM*
[10] *Universal Serial Bus*
[11] *Real-time Clock=RTC*
[12] *Analog Phase-Locked Loop*
[13] *Multichannel Buffered Serial Port=McBSP*



شکل۱-۳- نمیاش بلوک دیاگرامی از *TMS320C5505*

همچنین این دستگاه از معماری *VLIW* [۱] بهره می‌برد.

این معماری، یک نوع معماری مورد استفاده در پروسسورها می‌باشد. این معماری با زبان کامپایلری که دارد و توسط پیش‌پردازشهایی که روی دستورالعملهای طولانی انجام می‌دهد، قادر است این دستورالعملهای نسبتا بزرگ را به چندین دستورالعمل پایه بشکند سپس همه آنها را به کمک یک کلمه‌ی دستورالعمل[۲] طولانی به طور موازی اجرا می کند.

[1] *Very Long Instruction Word*
[2] *Instruction Word*

در حقیقت این معماری یک مرحله بالاتر از معماری *RISC*[1] می باشد. در این معماری سعی شده تا هر چه بیشتر از پیچیدگی سخت افزاری کاسته و به نقش نرم افزار افزوده شود که این باعث می شود چیپ و سایر عناصر سخت افزاری ساده تر شود و هزینه و توان مصرفی آن هم کاهش یابد.

مهمترین قسمت همان قسمتی است که وظیفه ساخت کلمه‌ی دستورالعمل را بر عهده دارد. این قسمت بیشتر تحت عنوان یک کامپایلر یا پیش پردازنده یاد می‌شود. هوشمندی این قسمت از معماری *VLIW* ، در چگونگی شکستن دستورالعملهای بزرگ به دستورالعملهای پایه می‌تواند سطح کارایی و سرعت این پردازنده ها را ارتقا دهد.

پیش پردازنده یا کامپایلر که یک مجموعه کدهای نرم‌افزاری است معمولا در یک چیپ حافظه فلش ذخیره می شود و به منظور ساخت کلمه‌ی دستورالعمل در اختیار پردازنده قرار می‌گیرد.

پردازنده به کمک این مجموعه نرم‌افزار قادر است با کمترین استفاده از سخت افزار هر دستورالعمل را تبدیل به تعدادی ریز عملیات کند و همه آنها را در قالب یک کلمه دستورالعمل در یک لحظه به طور موازی اجرا کند که این روند باعث می‌شود پردازنده با کمترین توان مصرفی بیشترین کارایی را ارائه دهد.

بر طبق توضیحات فوق می توان نتیجه گرفت که معماری *VLIW* در صدد است تا خود را به جرگه پردازنده هایی که قادر به اجرای هر دستورالعمل در یک سیکل کلاک هستند برساند البته با استفاده کمتر از سخت افزار و استفاده بیشتر از نرم افزار که این خود باعث بوجود آمدن امتیازات فوق العاده‌ای در این نوع معماری می‌شود.

۱-۱-۳- ویژگی های *C55x* :

در زیر خلاصه‌ای از ویژگیهای *C55x* بیان شده است:

[1] *Reduced Instruction Set Computing*

- قابلیت سه بار خواندن و دو بار نوشتن در حافظه در هر سیکل
- فرکانس کاری ۱۶۰ تا ۲۰۰ مگاهرتز
- دو واحد *MAC* با قابلیت ضرب *17bit x 17bit* در هر سیکل
- واحد *EMIF* برای دسترسی به حافظه خارجی
- سیستم محاسباتی ممیز ثابت
- قابلیت افزودن سخت افزار اختصاصی برای شتاب دهی به عملیات
- معماری پیشرفته چند باسه با یک باس برنامه ، ۳ باس داده و۴ باس آدرس
- دارای معماری *VLIW*
- قابلیت اجرای موازی چند دستور
- اجرای الگوریتم میانگین حداقل مربعات[1]

کاربردهای زیررا می توان برای سری *C55X* در نظر گرفت:

- کدینک و دیکدینگ صحبت
- حذف اکو - حذف نویز
- مدولاسیون و دمدولاسیون
- فشرده سازی تصویر و صدا
- رمز نگاری صحبت
- تشخیص صحبت و باز سازی صحبت
- پردازش تصویر

[1] *Least Mean Square = LMS*

۲-۱-۳- توسعه برنامه بلادرنگ

پیاده سازی یک نرم‌افزار بصورت بلادرنگ بر روی سخت افزار از اهمیت بالایی برخوردار است. در پیاده‌سازی بلادرنگ، هزینه تجهیزات همانند پارامترهای کیفی سیستم اهمیت زیادی داشته و بایستی سعی شود که یک الگوریتم به روش بهینه پیاده‌سازی گردد. در عمل، پیاده سازی بلادرنگ یک الگوریتم بر روی چیپ *DSP* شامل مراحل زیر می‌شود :

الف) بهبود تئوری الگوریتم

ب) آزمایش الگوریتم بوسیله شبیه سازی کامپیوتری با استفاده از یک زبان سطح بالا مانند *C*

ج) تبدیل کد سطح بالا به کد اسمبلی *DSP* مربوطه

د) آزمایش کد بلادرنگ با استفاده از ابزارهای توسعه گر موجود(مانند شبیه سازها،که در اینجا،*CCS* می‌باشد)

ه) انتخاب سخت افزار مناسب

در قسمت (ج) برای تبدیل کد سطح بالا به کد *DSP* سه روش وجود دارد:

روش اول) استفاده از کراس کمپایلر *DSP* است که زبان سطح بالای *C* را به اسمبلی *DSP* ترجمه می‌کند.

روش دوم) برنامه نویسی دستی و مستقیم الگوریتم با استفاده از مجموعه دستورالعملهای *DSP* می‌باشد.

روش سوم) از استفاده از کراس کمپایلر و همچنین برنامه نویسی دستی در قسمتهایی که از نظر زمان اجرا محدودیت وجود دارد.

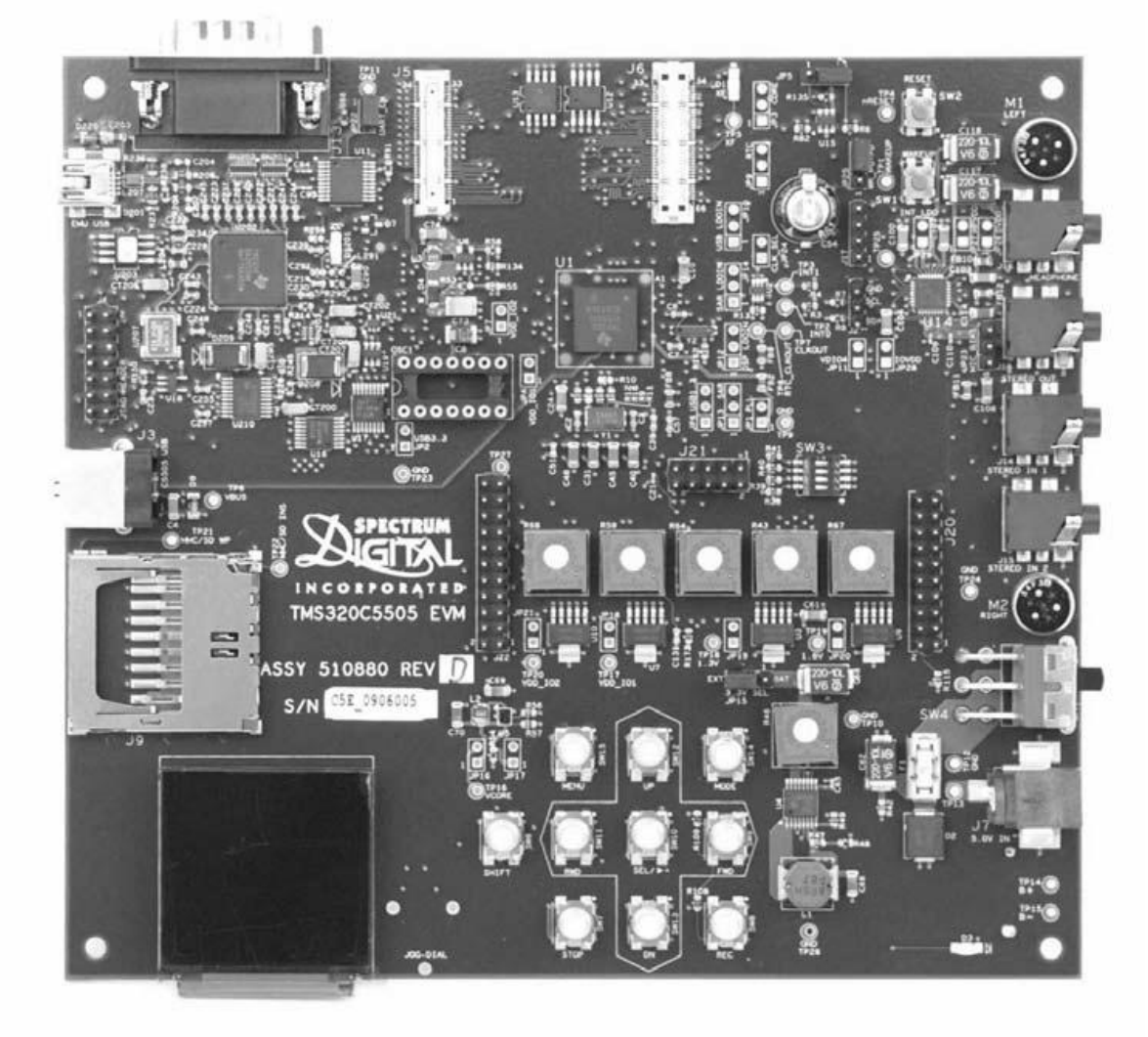
از آنجا که برنامه نویسی دستی کاری مشکل و وقت گیر می‌باشد و مخصوصاً وقتی که اندازه برنامه بزرگ باشد، آزمایش و اشکال‌زدایی آن وقت زیادی صرف می‌کند بنابراین معمولاً ازروش سوم در برنامه‌های بزرگ بیشتر استفاده می‌شود. در قسمت (د) نتایج شبیه سازی کامپیوتری با خروجی های معادل *DSP*

مقایسه می‌شوند. این کار معمولاً برای داده های آزمایش محدودی انجام می‌شود زیرا که پردازش مقدار زیادی داده در شبیه ساز *DSP* کار وقت‌گیر و دشواری می‌باشد. پس از مقایسه و درستی خروجی *DSP* و بررسی اجرای نرم افزار بصورت بلادرنگ آنگاه می‌توان از درستی پیاده سازی بلادرنگ الگوریتم اطمینان حاصل کرد.

۳-۱-۳- اجرای برنامه روی ماژول ارزیابی *C5505*

پس از آشنایی با پردازنده‌ی *TMS320C5505*، حال نوبت به معرفی ماژول ارزیابی *C5505* [1]می‌رسد . این ماژول، به کاربرها امکان می‌دهد تا برنامه‌های خود را برروی پردازنده *TMS320C5505* آزمایش کرده و توسعه دهند. همچنین این ماژول، به عنوان یک طراحی مرجع سخت‌افزاری برای *TMS320C5505* شناخته می‌شود. در شکل (۲-۳)، این ماژول نمایش داده شده است. این ماژول به همراه مجموعه‌ی کاملی از اجزای برروی[2] برد ارائه می‌شود و محیطی مناسب برای گستره‌ی وسیعی از برنامه‌ها فراهم می‌کند.

[1] *TMS320C5505 EVM*
[2] *On Board*



شکل ۲-۳- ماژول ارزیابی *TMS320C5505 EVM*

ماژول شامل اجزای زیر است :

- منبع توان متغیر/انعطاف پذیر[1]
- *TLV320AIC3254 stereo codec*
- ۱۰۰ *MHz VC5505 DSP*
- محل اتصال *MMC/SD*

[1] *Flexible/Variable*

- پورت *USB*
- *I^2C EEPROM* (۲۵۶ کیلوبایت) و *SPI EEPROM* (۲۵۶ کیلوبایت)
- *EMIF*, *I^2S*, *I^2S* و رابط *SPI*
- کنترل کننده جایگذاری شده‌ی *JTAG*
- *LCD* رنگی
- ۱۰ سوئیچ برای کاربر(این سوئیچ‌ها،بصورت دکمه‌های فشاری می‌باشند)
- رابط صوتی میکروفن/ بلندگو
- رابط تلفن
- رابط داده ناهمزمان *RS-232* (*UART*)
- رابط *JTAG* برای نمونه سازی
- *Stereo line in /out, headphone out and microphone in (L/R)*
- باتری یا منبع قدرت عمومی *+5V*
- نگه دارنده‌ی باتری

این ماژول برای کار کردن با *CCS* طراحی شده است. در واقع برنامه‌ی *CCS* از طریق *JTAG* با این ماژول ارتباط برقرار می‌کند.

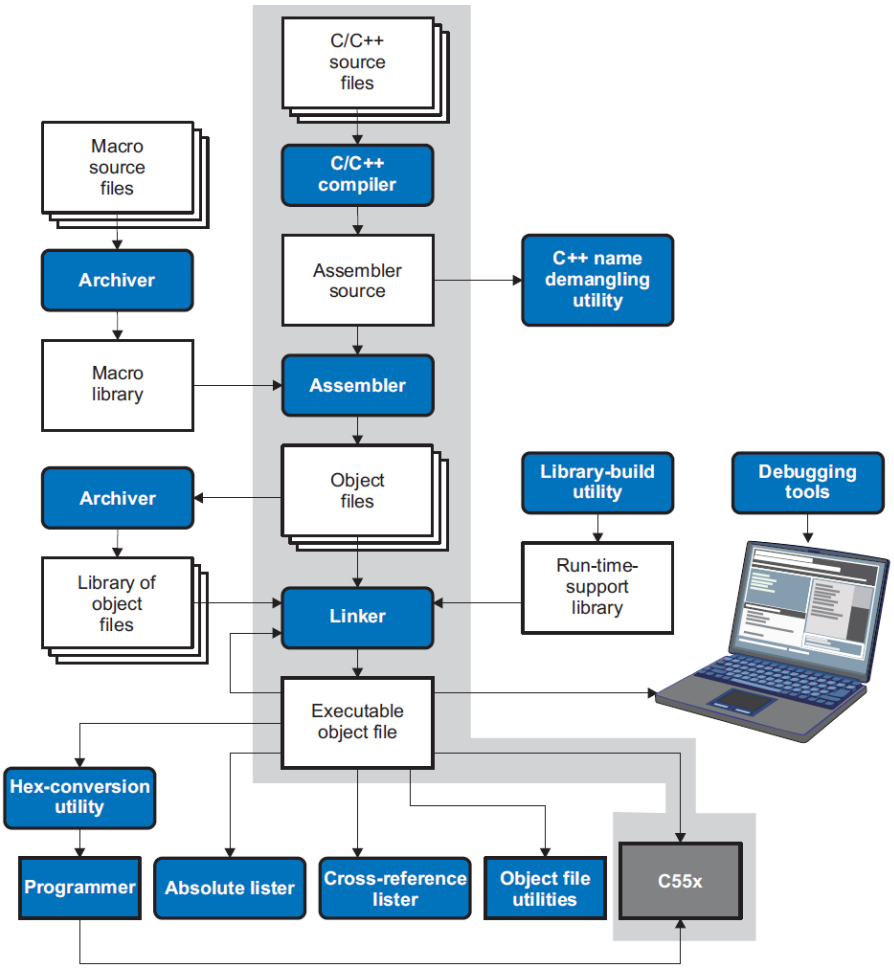
۴-۱-۳- بکارگیری ابزارهای توسعه نرم افزار

شکل (۳-۳) دیاگرام توسعه نرم افزاری *C55x* را نشان می‌دهد .بخش سایه زده مسیری را مشخص می‌کند که بیشتر استفاده می‌شود و بقیه بخش ها انتخابی هستند.

حال به بررسی اجزای مختلف مطرح شده در شکل می‌پردازیم:

- *C/C++ Compiler*

کد *C/C++* را به کد زبان اسمبلی ترجمه می‌کند .



شکل(۳-۳) : دیاگرام توسعه نرم افزاری *TMS320C55x*

- *Assembler*

فایلهای اسمبلی را به زبان ماشین و بصورت یک فایل *COFF Object* تبدیل می‌نماید .

- *Linker*

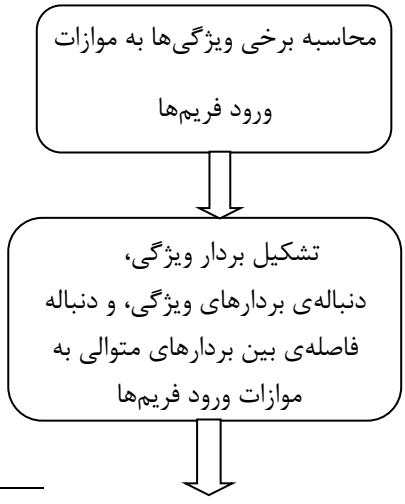
فایل‌های *Object* تولید شده توسط اسمبلر را بصورت یک ماژول *COFF Object* قابل اجرا تبدیل می‌نماید .

- مبدل *Hex*

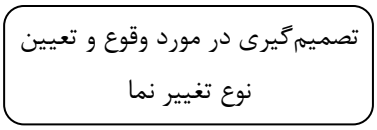
DSP های *C55x* می‌توانند فایل *COFF* را بعنوان ورودی قبول کنند ولی اغلب برنامه‌ریزهای *EPROM* قادر به این کارنیستند، ازاین رو باید فایل *COFF* به یکی از فرمت‌های *TI-tagged*، *Intel*، *Motorola* ، *Tektronix* تبدیل شود تا در برنامه‌ریز *EPROM* مربوطه ریخته شود .

۲-۳-روش پیشنهادی

در این بخش روش پیشنهادی برای آشکارسازی برون خط[1] تغییر نما و تعیین نوع گذر مربوطه معرفی می‌شود. دیاگرام بلوکی روش پیشنهادی در شکل (۴-۳) نشان داده شده است:



[1] *On-Line*



شکل ۴-۳ - دیاگرام بلوکی روش پیشنهادی برای آشکارسازی تغییر نما و تعیین نوع آن

هر یک از زیربخشهایی که در ادامه می‌آیند، یکی از بلوکهای دیاگرام شکل(۴-۳) را توضیح می‌دهند.

۱-۲-۳- محاسبه برخی ویژگیها به موازات ورود فریمها

ویژگیهایی که برای آشکارسازی برخط تغییر نما استفاده می‌شوند باید حساس به محتویات فریمها باشند، حجم پردازشی و حجم حافظه‌ی مصرفی کمی داشته باشند، استخراج آنها وابسته به اطلاعات فریمهایی که هنوز در دسترس نیستند و حتی فریمهایی که مربوط به گذشته نسبتاً دور بوده‌اند، نباشد.

شرایط فوق کمک می‌کنند تا الگوریتم مورد استفاده در عین این که کارایی عملکرد و کارایی زمانی مناسبی داشته باشد، دارای قابلیت برخط بودن و پردازش بلادرنگ نیز باشد.

در محاسبه‌ی این ویژگیها، فرض بر این است که فریم‌ها از نوع رنگی می‌باشند. اگر فریمها، از نوع سطح خاکستری باشند، نحوه‌ی محاسبه‌ی ویژگیها تغییر چندانی نمی‌کند و حتی، حجم پردازشی تا حد قابل ملاحظه‌ای کم می‌شود.

۱-۱-۲-۳-میانگین و واریانس

در صورتی که فریمها رنگی باشند:

به موازات ورود مقادیر مولفه‌های رنگ هر پیکسل از فریم فعلی می‌توان میانگین و واریانس کل آن فریم را محاسبه کرد. این ویژگیها را پس از محاسبه، نرمالیزه می‌کنیم. عمل نرمالیزه کردن میانگین‌ها با تقسیم مقدار هر میانگین بر بیشترین مقدار میانگین مولفه‌های رنگی وارد شده تا کنون قابل انجام است. نرمالیزه کردن واریانس به طرز مشابهی انجام می‌شود. بنابراین، در این مرحله، شش ویژگی محاسبه می‌شود. محاسبه‌ی این ویژگیها نیاز به حجم حافظه و حجم پردازشی چندانی ندارد. از این شش ویژگی، سه

ویژگی، میانگین‌ها می‌باشند(برای هرکدام از مؤلفه‌های R، G و B میانگین بطور جداگانه محاسبه می‌شود) و سه ویژگی واریانس‌ها (برای هر کدام از مؤلفه‌های R، G و B واریانس بطور جداگانه محاسبه می‌شود).

در صورتی که فریم‌ها خاکستری باشند:

به موازات ورود مقادیر شدت روشنایی هر پیکسل از فریم فعلی می‌توان میانگین و واریانس کل آن فریم را محاسبه کرد. این ویژگیها را پس از محاسبه، نرمالیزه می‌کنیم. عمل نرمالیزه کردن میانگین‌ها با تقسیم مقدار هر میانگین بر بیشترین مقدار میانگین محاسبه شده تاکنون قابل انجام است. نرمالیزه کردن واریانس به طرز مشابهی انجام می‌شود. بنابراین، در این مرحله، دو ویژگی محاسبه می‌شود. محاسبه‌ی این ویژگیها نیاز به حجم حافظه و حجم پردازشی چندانی ندارد و به صورت برخط قابل انجام است. از این دو ویژگی، یک ویژگی میانگین است و یک ویژگی واریانس.

۲-۱-۲-۳-اختلاف هیستوگرام فریم فعلی با فریم قبلی

در صورتی که فریمها رنگی باشند:

محاسبه‌ی هیستوگرام فریم فعلی، عملی است که با ورود مقادیر مولفه‌های رنگ هر پیکسل از فریم فعلی قابل انجام است. داده‌های نمودار هیستوگرام هر فریم تا زمان محاسبه‌ی نمودار هیستوگرام فریم بعدی باید حفظ و ذخیره شوند تا فاصله‌ی بین دو هیستوگرام محاسبه شود. بنابراین نیاز به بافری به اندازه ۷۶۸ کلمه(سه مؤلفه‌ی رنگ وجود دارد و برای هر رنگ بافری به اندازه ۲۵۶ کلمه نیاز است)می‌باشد. برای محاسبه‌ی فاصله‌ی بین دو نمودار هیستوگرام از رابطه‌ی متوسط مجموع مجذورهای اختلاف مقادیر متناظر از دو نمودار استفاده می‌کنیم. بنابراین، در این مرحله، سه ویژگی محاسبه می‌شود. ویژگیهای محاسبه شده، به طرز مشابه با قبل، نرمالیزه می‌شوند.

در صورتی که فریم‌ها خاکستری باشند:

محاسبه‌ی هیستوگرام فریم فعلی، عملی است که با ورود مقدار شدت روشنایی هر پیکسل از فریم فعلی قابل انجام است. داده‌های نمودار هیستوگرام هر فریم تا زمان محاسبه‌ی نمودار هیستوگرام فریم بعدی باید حفظ و ذخیره شوند تا فاصله‌ی بین دو هیستوگرام محاسبه شود بنابراین نیاز به بافری به اندازه‌ی ۲۵۶ کلمه می‌باشد. برای محاسبه‌ی فاصله‌ی بین دو نمودار هیستوگرام از رابطه‌ی متوسط مجموع مجذورهای اختلاف مقادیر متناظر از دو نمودار استفاده می‌کنیم. بنابراین، در این مرحله، یک ویژگی محاسبه می‌شود. ویژگی محاسبه شده، به طرز مشابه با قبل، نرمالیزه می‌شوند.

۳-۱-۲-۳- مرکزثقل و پهنای باند مولفه‌های رنگ

برای هر تابع یک بعدی *f(t)* می‌توان دو کمیت مرکز ثقل (یا مرکز تجمع) انرژی، C، و پهنای باند، Δ_f، تعریف و استفاده کرد. مرکز ثقل، معرف نقطه‌ای است که انرژی سیگنال عمدتاً در آن نقطه متمرکز شده است. پهنای باند منحنی، معرف محدوده‌ای متقارن نسبت به مرکز ثقل است که سیگنال عمده‌ی انرژی خود را در این بازه بروز می دهد:

$$C = \frac{1}{E_f}\int_{-\infty}^{+\infty} t.f(t)^2 dt$$

$$\Delta_f = \frac{1}{E_f}\int_{-\infty}^{+\infty} (t-C)^2.f(t)^2 dt \qquad (۳-۱)$$

در حالت سیگنال گسسته‌ی f، به جای انتگرال از سیگما استفاده می‌کنیم. همچنین برای افزایش سرعت محاسبات نیازی به تقسیم حاصل سیگما بر انرژی سیگنال نمی‌باشد بلکه با عمل شیفت دادن، حاصل سیگما را در محدوده یک عدد ۱۶ بیتی نگه می‌داریم زیرا خانه‌های حافظه در پردازشگرهای سیگنال *TMS320C55xx*، ۱۶ بیتی می‌باشند.

دز صورتی که فریمها رنگی باشند:

حال در این مرحله، به ازاء هر ستون و هرمولفه‌ی رنگ از فریم فعلی، یک منحنی شامل مقادیر آن مولفه‌ی رنگی تشکیل داده و مقادیر مرکز ثقل و پهنای باند آن را محاسبه می‌کنیم. در انتها، میانیگن نرمالیزه شده‌ی این مراکز ثقل و پهناهای باند را محاسبه و ذخیره می‌کنیم.

در ادامه، تمام محاسبات فوق را دوباره برای هر سطر از فریم فعلی تکرار می‌کنیم. بنابراین، در پایانِ این مرحله، شش ویژگی مربوط به مرکز ثقل و شش ویژگی مربوط به پهنای باند به دست می‌آید.(توجه کنید که در این مرحله، برای هر مؤلفه رنگ دو مرکز ثقل و دو پهنای باند بدست می‌آید. یک مرکز ثقل و یک پهنای باند در هر ستون و یک مرکز ثقل و یک پهنای باند در هر سطر).

در صورتی که فریمها خاکستری باشند:

حال در این مرحله، به ازاء هر ستون از ماتریس فریم فعلی، یک منحنی شامل مقادیر شدت روشنایی پیکسلهای آن ستون تشکیل داده و مقادیر مرکز ثقل و پهنای باند آن را محاسبه می‌کنیم. در انتها، میانگین نرمالیزه شده‌ی این مراکز ثقل و پهناهای باند را محاسبه و ذخیره می‌کنیم.

در ادامه، تمام محاسبات فوق را دوباره برای هر سطر از فریم فعلی تکرار می‌کنیم. بنابراین، در پایانِ این مرحله، دو ویژگی مربوط به مرکز ثقل و دو ویژگی مربوط به پهنای باند به دست می‌آید.

۴-۱-۲-۳- قدرت لبه‌ها

در این مرحله، ابتدا تصویر رنگی مربوط به هر فریم را به تصویر سطح خاکستری تبدیل کرده و سپس، به کمک ماسک لبه‌یابیِ *Prewitt* و انجام یک آستانه‌گیری ساده، یک تصویر دودویی به دست می‌آوریم. ویژگی قدرت لبه را برابر تعداد کل پیکسلهای سفید متعلق به تصویر دودویی مذکور تعریف می‌کنیم که البته، به طرز مشابه با قبل، آن را نرمالیزه می‌کنیم. به منظور کاهش حجم محاسبات، برای تبدیل یک تصویر رنگی (شامل مولفه‌های رنگیِ R، G، و B) به تصویر سطح خاکستری Y می‌توانیم از تقریب زیر استفاده کنیم.

$$Y \cong R + G + B \qquad (۳-۲)$$

مقدار آستانه‌ی لازم برای انجام آستانه‌گیری و به دست آوردن تصویر دودویی از رابطه‌ی زیر تعیین می‌شود. برای هر نمای جدید کافی است یک بار این محاسبه انجام شود. بنابراین، سربار محاسباتی آن ناچیز است.

$$T = \mu + k.\sigma \qquad (۳-۳)$$

پارامترهای μ و σ به ترتیب میانگین و انحراف معیار مقادیر تصویر حاصل از لبه‌یابی با ماسک *Prewitt* بوده و پارامتر k نیز یک عدد ثابت است که از طریق آزمایش مقدار مناسب آن انتخاب می‌شود.

۲-۲-۳-محاسبه‌ی فاصله‌ی بردارهای ویژگی

در صورتی که فریمها رنگی باشند:

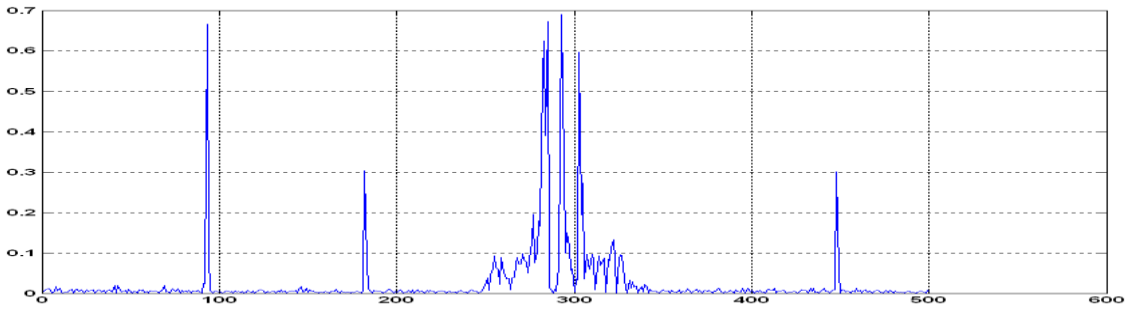
پس از محاسبه‌ی تمام بیست و یک ویژگی مطرح شده در بخش قبل، به ازاء هر فریم یک بردار ویژگی تشکل شده و فاصله‌ی اقلیدسی آن با بردار محاسبه شده‌ی قبلی محاسبه می‌شود. برای کاهش حجم محاسبات، فاصله‌ی مذکور را می‌توان به طور تقریبی به کمک محاسبه‌ی مجموع مجذورهای اختلاف مقادیر متناظر از دو بردار به دست آورد. بنابراین، در این مرحله، یک دنباله‌ی فاصله به دست می‌آوریم.

در صورتی که فریمها خاکستری باشند، به جای بیست و دو ویژگی، هشت ویژگی خواهیم داشت.

۳-۲-۳- تعیین وقوع و نوع تغییر نما

با توجه به این که ویژگیهایی که انتخاب کرده‌ایم به نحوی معرف محتویات فریم‌ها می‌باشند، انتظار داریم با تغییر نما، مقدار دنباله‌ی فاصله یک جهش ناگهانی داشته باشد. شکل (۵-۳) مربوط به دنباله‌ی محاسبه شده برای قسمتی از یک دنباله‌ی ویدیویی است که شامل سه تغییر ناگهانی و یک تغییر آرام است.

ملاحظه می‌کنید که در حالت تغییر آرام، دنباله‌ی مذکور مقادیر بزرگی در محدوده‌ی زمانی چند فریم اختیار می‌کند. اما در حالت تغییر ناگهانی، این مقادیر بزرگ فقط در محدوده‌ی زمانی یک یا دو فریم رخ می‌دهند.



شکل ۵-۳- یک نمونه دنباله‌ی فاصله مربوط به یک دنباله‌ی ویدیویی شامل سه تغییر ناگهانی و یک تغییر آرام

برای اعلام وقوع یا عدم وقوع تغییر نما شرط زیر را بررسی می‌کنیم:

$$if \frac{D(i)}{M1} > Th1 \implies Shot\ Boundary\ at\ i'th\ Frame\ , \qquad (۳-۴)$$

در این رابطه، $D(i)$ برابر مقدار دنباله‌ی فاصله در i اُمین نقطه و پارامتر $M1$ برابر با میانگین آخرین N مقدار قبلی (نسبت به مقدار فعلی iاُم) از دنباله‌ی فاصله است.

برای تعیین نوع تغییر نما، در نقاطی مانند i که تغییر نما اعلام شده باشد، شرط زیر را بررسی می‌کنیم:

$$if\ Mean\{D(i-N\ to\ i+N)\} > Th2$$

$$\implies GT, \qquad else \implies AT \qquad (۳-۵)$$

۴-۲-۳-پیچیدگی محاسباتی الگوریتم پیشنهادی

اگر تعداد کل پیکسلهای هر فریم را برابر N، تعداد سطوح هیستوگرام را برابر B، و ابعاد ماسک مربعی عملگر *Prewitt* را برابر L فرض کنیم، به سادگی می‌توان نشان داد تعداد اعمال جمع و ضرب مورد نیاز برای محاسبه ویژگیهای مورد استفاده در الگوریتم پیشنهادی به ازاء هر فریم مطابق با جدول (۱) می‌باشد.

جدول (۱): تعداد اعمال جمع و ضرب مربوط به هر ویژگی

نوع ویژگی	تعداد اعمال جمع	تعداداعمال ضرب
Mean	*N*	-
Variance	2*N*	*N*
Histogram	*N+2B*	*B*
Bandwidth	*2N*	*5N*
Edge	*N.(L2+1)*	*N.L2*
مجموع تقریبی	*N.(L2+6)*	*N.(L2+5)*

بنابراین برای هر فریم در کل در حدود $N\times(L^2+11)$ عمل ضرب و جمع نیاز داریم. اگر ابعاد هر فریم را N=۵۰۰×۵۰۰، تعداد سطوح هیستوگرام را $B=256$ و ابعاد ماسک لبه‌یابی را $L=3$ در نظر بگیریم و نیز با توجه به اینکه هر دستور پردازشگر در یک سیکل ماشین قابل اجراست، بار محاسباتی پردازشگر به ازاء هر فریم حدود *MIPS* ۷,۲۵ خواهد بود. اگر نرخ فریم بر ثانیه برابر *fps* ۲۵ باشد، در چنین شرایطی نیاز به پردازشگری با توان محاسباتی حدود *MIPS* ۱۸۱ داریم که به راحتی می‌توان چنین پردازشگری را در

خانواده *TMS320C55xx* یافت. البته این نکته مهم قابل ذکر است که پردازشگرهای مذکور دارای قابلیت پردازش موازی هستند و می‌توانند دو دستور را در یک سیکل ماشین اجرا کنند، بنابراین پردازشگری با توان محاسباتی کمتر از *MIPS* ۱۸۱ نیز ممکن است قادر به اجرای بلادرنگ روش پیشنهادی باشد. البته لازم به ذکر است که خانواده مذکور عمدتاً برای پردازش سیگنالهای یک و دو بعدی مورد استفاده قرار می‌گیرد.

۳-۳-نتایج آزمایشها

در کار این مقاله، از پایگاه‌های دنباله‌های ویدیویی که هر پایگاه شامل ۳۰ دقیقه (معادل حدوداً ۴۷۰۰۰ فریم) می‌باشد، استفاده شد که از نظر نوع، به چهار دسته کارتونی، سینمایی، ورزشی، و اخبار قابل تقسیم می‌باشند. سهم زمانی هر یک از این دسته‌ها تقریباً مساوی با هم انتخاب شده است.

مقادیر به کار رفته برای پارامترهای مورد استفاده در روش پیشنهادی در جدول (۲) آورده شده است.

جدول (۲): مقادیر پارامترهای استفاده شده

آستانه	مقدار
Th1	۵
Th2	۰,۱
N	۵

مقادیر چهار پارامتر ارزیابی شامل دقت، یادآوری، معیار هارمونیکی، و متوسط زمان اجرای برنامه برای هر فریم، برای هر دسته از دسته‌های چهارگانه‌ی مذکور در جدول (۳) آورده شده است. جدول (۴) مقدار همین معیارها را برای روش مبتنی بر هیستوگرام [۳۶] نشان می‌دهد. با مقایسه این دو جدول، برتری روش پیشنهادی از نظر دقت و یادآوری مشخص می‌شود. همچنین با توجه به متوسط زمان صرف شده در

روش پیشنهادی (در سیستمی با پردازنده *Intel P4 2.4 GHz* و حافظه *1 GByte* و با استفاده از نرم افزار *MATLAB*) برای هر فریم، مشخص است که روش پیشنهادی قادر به پردازش بلادرنگ دنباله‌های ویدیویی در حالت واقعی است زیرا در این دنباله‌ها که شامل حدود ۲۵ فریم بر ثانیه می‌باشند، برای هر فریم حدود ۰,۰۴ ثانیه زمان در اختیار داریم.

جدول (۳): مقادیر معیارهای ارزیابی مربوط به روش پیشنهادی

دسته/معیار	دقت	یادآوری	معیار هارمونیکی	م. ز. ا. برای هرفریم (ثانیه)
اخبار	۹۴,۱	۹۵,۴	۹۴,۷	۰,۱۴
ورزشی	۹۳,۵	۹۳	۹۳,۲	۰,۱۲
سینمایی	۹۵	۹۴,۱	۹۴,۵	۰,۰۷
کارتونی	۹۸,۲	۹۵,۳	۹۶,۷	۰,۰۵

جدول (۴): مقادیر معیارهای ارزیابی مربوط به روش مبتنی برهیستوگرام

دسته/معیار	دقت	یادآوری	معیار هارمونیکی	م. ز. ا. برای هر فریم (ثانیه)
اخبار	۸۸,۵	۸۴,۳	۸۶,۳	۰,۰۹
ورزشی	۸۶,۱	۸۳,۵	۸۴,۷	۰,۰۸
سینمایی	۸۹,۸	۸۷,۳	۸۸	۰,۰۳
کارتونی	۹۱,۲	۹۰,۴	۹۰,۸	۰,۰۱

همانطور که در دو جدول فوق مشاهده می‌کنید، نتایج حاصل از روش پیشنهادی و روش هیستوگرام از نظر متوسط زمان اجرا برای هر فریم، تفاوت چندانی ندارند در حالیکه، دو معیار دقت و یادآوری در روش پیشنهادی به مراتب بالاتر می‌باشند. این نتایج تاییدی است بر عملکرد بهتر و قویتر روش پیشنهادی.

با توجه به نتایج بدست آمده، روش پیشنهادی در سخت افزاری مبتنی بر پردازشگرهای *DSP* (با توجه به فرکانس کاری بالا و نیز مجهز بودن این پردازشگرها به دستورات قوی جهت پردازش سیگنال) و یا ادوات برنامه پذیر *FPGA* (با توجه به سرعت بالای سخت افزار نسبت به نرم افزار و نیز قابلیت پردازش موازی اینگونه ادوات) ، به راحتی قادر به پیاده‌سازی بلادرنگ خواهد بود.

برای پیاده‌سازی الگوریتم پیشنهادی از برد آموزشی *TMS320C5505* استفاده شده است که تصویر آن در شکل (۵-۳) نشان داده شده است.

شکل (۶-۳) تصویری از برد آموزشی *TMS320C5505* مورد استفاده

این برد آموزشی از طریق کابل رابط *JTAG* به کامپیوتر شخصی متصل و از طریق نرم افزار *Code Composer Studio* با برنامه‌های کابردی کاربر ارتباط برقرار می‌کند. در پردازشگرهای سیگنال، از جمله پردازشگرهای خانواده‌ی *TMS320C55xx*، برخی قابلیتهای سخت‌افزاری و نرم‌افزاری لحاظ شده است که

منجر به پردازش سیگنالهای دیجیتال با کارایی بسیار بالاتری نسبت به پردازشگرهای همه‌منظوره گردیده‌اند. مهمترین این ویژگیها شامل معماری داخلی *VLIW*، ساختار خط لوله، قابلیت اجرای موازی چند دستور، برخورداری از دستورات قابل اجرا در یک سیکل، و بالاخره، برخورداری از دستورات قدرتمند و تخصصی پردازش سیگنال مانند دستورات فیلتر کردن، اجرای الگوریتم *LMS*، اجرای الگوریتم ویتربی، انجام همزمان ضرب و جمع می‌باشد.

الگوریتم پیشنهادی در نرم افزار *Code Composer Studio* به زبان اسمبلی نوشته و اجرا گردید. متوسط تعداد سیکلهای ماشین به ازاء هر فریم حدود *MIPS* ۶,۱ به دست آمد که با توجه به محاسبات انجام شده در زیربخش(۴-۲-۳)، این عدد نشان دهنده قابلیت پردازش بلادرنگ الگوریتم پیشنهادی در بستر پردازشگرهای *TMS320C5505* است.

۴-۳-نتیجه گیری

در این بخش روشی برخط و سریع برای آشکارسازی و تعیین نوع تغییر نما پیشنهاد شد. روش پیشنهادی از ویژگیهایی استفاده کرد که به موازات ورود اطلاعات فریمها قابل محاسبه بوده و نیازی به دانستن اطلاعات فریمهای بعدی ندارند. همچنین، حجم محاسباتی روش پیشنهادی تا حد قابل ملاحظه‌ای کم است طوری که مناسب پیاده‌سازی بلادرنگ سخت‌افزاری در بسترهایی مانند پردازشگرهای سیگنال و مدارات منطقی برنامه‌پذیر می‌باشد. همچنین روش پیشنهادی روی برد آموزشی مبتنی بر *TMS320C5505* اجرا گردید. کارایی روش پیشنهادی با کمک چهار معیار دقت، یادآوری، معیار هارمونیکی و متوسط زمان اجرای برنامه به ازاء هر فریم ارزیابی و با یک روش مبتنی بر هیستوگرام مقایسه شد.

یکی از موارد مهمی که روش پیشنهادی دچار اشتباه در تشخیص می‌شود، زمانی است که در دو فریم متوالی و متعلق به یک نمای مشترک، تغییر قابل توجهی در محتوای فریمها (از جهت رنگ و شکل اشیاء) رخ دهد.

فصل ۴

پردازشگرهای داوینچی و اسناد مرتبط با آن

۱-۴- پردازشگرهای *DAVINCI*

تولید و استفاده از گجت‌ها[۱]، به عنوان روشی برای طراحان، مهندسان و بسیاری از مصرف کننده ها تبدیل شده است. این گجت‌ها بسته به اینکه برای موزیک، ویدیو، بازی ها و یا ارتباطات مورد استفاده قرار بگیرند، مدام پیشرفته تر ، باهوش‌تر و کوچک‌تر می‌شوند.

در داخل این گجت‌ها، پردازنده‌های دیجیتال جایگذاری شده با تکنولوژی بالا قرار دارند[۲]. بسته به دستگاه موردنظر، پردازنده ممکن است مسئول نمایش و غیرفشرده کردن[۳] ویدیو، ذخیره موزیک یا پخش آهنگ‌های جالب باشد. اینها نمونه وظایفی که توسط پردازنده‌های جایگذاری شده کنترل می‌شوند و معمولاًدر یکی از دو دسته ی زیر قرار می‌گیرند :

- وظایف پردازش همه کاره[۴] همانند انتقال اطلاعات یا اجرای رابط گرافیکی
- وظایف پردازش سیگنال دیجیتال[۵] همانند فشرده و غیر فشرده کردن ویدیو، صوت و گفتار .

شرکت *TI* با خانواده ی سیستم بر روی چیپ[۶] خود مخصوصاً خانواده‌های*OMAP* و *Davinci*، در زمینه‌ی مجتمع‌سازی پردازنده پیشگام است. این وسایل با ترکیب پردازنده‌های هدف عمومی، پردازنده‌های سیگنال دیجیتال و شتاب دهنده‌های سخت‌افزاری اختصاص داده شده، بسیار هوشمند می‌باشند. اگر اتومبیل هایبرید[۷]را در نظر بگیرید، این اتومبیل نیروی بنزین و الکتریکی را برای افزایش اثر سوخت ترکیب می‌کند. عملکرد بهتری دارد و شتاب آن نیز بیشتر می‌باشد و فراموش نکنید آلودگی کمتری ایجاد می‌کند. همانند ماشین‌های هایبرید، پردازنده‌های سیلیکونی *SOC* محبوبیت زیادی به دست آورده‌اند زیرا

---

[1] *Gadgets*
[2] *High-Tech*
[3] *Decompressing*
[4] *General Purpose Processing Tasks*
[5] *Digital Signal Processing Tasks*
[6] *System On Chip = SOC*
[7] *Hybrid*

توان کمتری مصرف می‌کنند، عمکرد بهتری دارند و آلودگی گرمایی کمتری در ایستگاه‌های الکترونیکی ایجاد می‌نمایند. همان طور که ترانزیستورها کوچکتر شدند و این امکان فراهم شد تا صدها میلیون (یا حتی میلیاردها) تزانزیستور را بر روی یک تراشه[1] قرار داد، طراحان تراشه سیلیکونی به این نتیجه رسیدند که قرار دادن چندین پردازنده بر روی یک تراشه که هر کدام در سرعت کلاک پایینتری فعالیت می‌کنند به جای قرار دادن یک پردازنده با سرعت کلاک بالاتر، عملی‌تر خواهد بود. بنابراین پردازنده چند هسته‌ای به وجود آمد .

در طراحی‌های سیستم‌های چند هسته‌ای دو روش کلی وجود دارد :

الف) روش هسته‌های مشابه[2] (همگون)

در این روش دو یا چند هسته‌ی مشابه بر روی یک تراشه تنها قرار می‌گیرند. دستگاه‌های چهار، شش یا هشت هسته‌ای هم اکنون وجود دارند. این شکل ساده‌ترین معماری چند هسته‌ای است اما محدودیت‌هایی اساسی دارد .

ب) روش هسته های غیر مشابه[3] (ناهمگون)

در این روش هسته‌های متفاوتی را بر روی یک تراشه قرار می‌دهند. دلیل این امر، این است که این هسته‌های متفاوت تنظیم می شوند تا بیشترین اثر را در وظایف متفاوت داشته باشند. یک نمونه از این روش، دستگاه *Davinci* از خانواده *TI* است که در آن یک هسته نقش *GPP* و یک هسته در نقش *DSP* است .

در سیستم‌های چند هسته‌ای با هسته‌های یکسان، شما می توانید وظایف یکسان را بارها و بارها تکرار کنید. مثلاً فرض کنید می‌خواهید برنامه‌ی کدک صدا را اجرا نمایید. فرض کنید یک هسته می تواند با

---

[1] *Chip*
[2] *Homogeneous Core approach*
[3] *Heterogeneous Core approach*

هشت کانال کار کند. در این صورت دو هسته‌ی مشابه می‌توانند با شانزده کانال ،چهار هسته با سی و دو کانال و الی آخر کار کنند. مورد دیگری که می‌توانید از سیستم‌های چند هسته‌ای با هسته‌های یکسان استفاده کنید، این است که چند کار مستقل را به راحتی بشکافید و هر کار را در یک هسته انجام دهید . به طوری که ارتباط بین کارها خیلی کم باشد یا اصلاً ارتباطی نداشته باشند.

حال فرض کنیم بخواهیم صدا و ویدیو را به همراه رابط گرافیکی پیشرفته مدیریت کنیم. همچنین بخواهیم از یک کد سیستم عملیاتی سطح بالا، که در محصولات قبلی گسترش پیدا کرده است، دوباره استفاده کنیم. در این‌حالت، پردازنده‌های همه‌کاره که در مدیریت سیستم های عملیاتی سطح بالا و بسیاری از وظایف ورودی / خروجی عالی عمل می‌کنند، در مدیریت پردازش سیگنال صدا، صوت و ویدیو مناسب نمی‌باشند. از طرفی پردازنده‌ی سیگنال دیجیتال در اجرای ویدیو، صوت و گفتار مناسب است اما از برنامه‌های کاربردی سیستم عملیاتی سطح بالا، حمایت نمی‌کند. دستگاه های *OMAP* و *Davinci*، در کاربردهایی که نیازهای متفاوت را ترکیب می‌کنند، می‌درخشند. این دستگاه‌ها روش برنامه‌ریزی چند هسته‌ای غیرمشابه را با ترکیب هسته‌های *GPP* و*DSP* در یک تراشه، در خود جای داده اند .

دستگاه‌های *Davinci* و *OMAP* شامل دو پردازنده زیر می‌باشند:

- پردازنده هدف عمومی از شرکت *ARM*

پردازنده *ARM*، برای اجرای سیستم عملیاتی رایج مانند لینوکس بسیار مناسب است و می‌تواند بیشتر توابع ورودی / خروجی و توابعی که باعث باقی ماندن سیستم در حالت کارکرد مناسب[1]می‌شوند، مانند رابط گرافیکی کاربر[2]را، کنترل کند .

- پردازنده ی سیگنال دیجیتال *C64x+* با عملکرد بالا

[1] *Housekeeping Functions*
[2] *Graphical User Interface = GUI*

این *DSP* تمامی توابع قوی و مفید پردازش سیگنال را که در برنامه های چند رسانه‌ای امروزی مورد نیاز است، کنترل می‌کند .

همچنین توجه داشته باشید که بسیار بهتر خواهد بود اگر از شتاب دهنده‌های سخت افزاری در اجرای برنامه‌ها استفاده کنیم. یک مثال از این شتاب هنده‌ها، پردازنده کمکی تصویر و ویدیو است. این شتاب‌دهنده‌ها معمولاً تبدیلات دو بعدی تصویر و تخمین حرکت را برای کد کردن جابه جایی بین فریم‌های ویدیویی، به کار می‌برند. در *OMAP* و *Davinci* نیز از این شتاب‌دهنده استفاده شده است.

۲-۴- ویژگی‌های ماژول ارزیابی *DM6446*[1]

همان‌طور که از نام ماژول‌های ارزیابی پیدا است، بسترهای ایده آلی هستند که برای ارزیابی سخت افزار و اجرای نرم افزار اولیه‌ی شما به کار می‌روند. *EVM* ها به صورت زیبا طراحی نشده اند اما کارا هستند.

ماژول ارزیابی شرکت *TI* شامل سخت‌افزار و نرم‌افزار است که بصورت موارد زیر مورد استفاده قرار می‌گیرد:

- یک بستر برای ارزیابی سریع تراشه *TMS320DM6446*، که عضوی از خانواده‌ی پردازشگرهای *DAVINCI* است.
- محیطی که به راحتی قابل استفاده است و به گسترش‌دهندگان سیستم‌های جایگذاری شده *ARM-Linux* (کسانی که ممکن است در مورد *DSP* تجربه‌ای نداشته باشند)، امکان می‌دهد تا برنامه‌ی خود را بدون نیاز به برنامه ریزی *DSP*، گسترش دهند.

[1] *Digital Video Evaluation Module = DVEVM*

می‌توان گفت قلب ماژول ارزیابی(از دیدگاه درجه اهمیت)، برد گسترش[1] است که شامل تراشه *DM6446* و انواع گوناگونی اجزای جانبی است. تراشه *DM6446*، شامل یک پردازنده هدف عمومی *ARM* و یک پردازنده سیگنال دیجیتال *C64x+* می‌باشد.

علاوه بر برد گسترش، این ماژول دارای موارد زیر نیز می‌باشد:

- اجزای نرم‌افزاری شامل نسخه‌ی ارزیابی *Monta vista Linux* و کدک‌های ویدیو و صوت
- اجزای سخت‌افزاری شامل دوربین و مانیتور

همانطور که ذکر شد، برروی برد گسترش، تراشه‌ی *DM6446* قرار دارد که شامل پردازنده هدف عمومی*ARM9* و پردازنده‌ی سیگنال دیجیتال *C64x+* و یک ماژول شتاب دهنده‌ی ویدیو است. در یک طراحی معمول، *ARM9* بعنوان یک پردازنده کاربردی به خدمت گرفته می‌شود که لینوکس، رابط کاربر وکنترل عمومی سیستم را اجرا می‌کند در حالیکه*C64x+* و شتاب دهنده‌ی ویدیو برای اجرای وظایفی که برروی پردازش سیگنال دیجیتال تمرکز دارند، مانند کدک‌های چندرسانه‌ای، به کار می‌روند. *DVEVM* نحوه‌ی کاربرد و عملکرد زیربنای *DSP* را به گسترش دهنده‌ی نرم‌افزار *ARM* نمایش نمی‌دهد، در عوض تمام قابلیتهای *DSP* را بصورت توابعی در می‌آورد و به نرم‌افزاری که برروی *ARM* قرار دارد این امکان را می‌دهد که کدکهای *DSP*-اساس را از طریق مکانیزم معمول فراخوانی و رابط برنامه‌ی کاربردی[2]، درخواست کند. بنابراین *DVEVM* برای گسترش‌دهندگان *ARM-Linux* که می‌خواهند کدکهای متمرکز برروی *DSP* را در برنامه‌های خود به کار برند بدون اینکه نیازی به دانستن جزئیات معماری *DSP*، مدل برنامه‌ریزی، ابزار توسعه نرم‌افزار و کدکهای چندرسانه‌ای داشته باشند، بسیار مناسب است.

---

[1] *Development Board*
[2] *Application Program Interface = API*

DVEVM شامل تعداد زیادی اجزا می‌باشد که برای تعیین قابلیتهای پردازنده *DAVINCI* و آغاز گسترش برنامه‌های چند رسانه‌ای، ضروری اند. این اجزا عبارتند از:

- برد گسترش که براساس پردازنده *TMS320DM6446* می‌باشد و شامل تعداد زیادی اجزای جانبی است
- دوربین ویدیویی *NTSC* و میکروفون
- نمایش‌دهنده *NTSC LCD* و بلندگوها
- یک *IR Remote* که برای کنترل دموها از طریق منوهای رو صفحه نمایش به کار می‌رود
- یک نسخه از *Monta Vista* که برای خانواده *DAVINCI* گسترش یافته است.
- سورس کد و *Make File* برای کاربردهای دمو
- یک راهنمای آغاز به کار یا *Getting Started Guide*

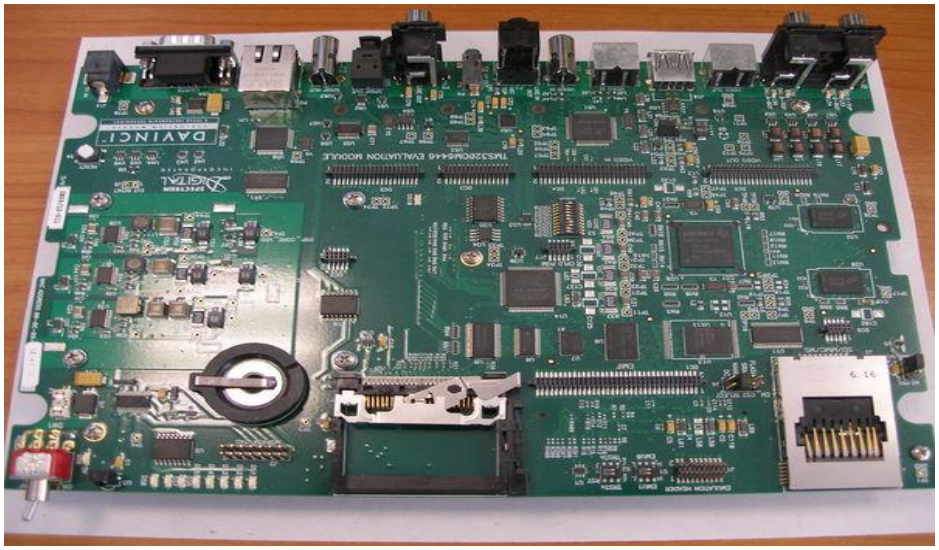
برنامه‌های نمایشی *DVEVM* که از طریق شرکت *TI*فراهم شده‌اند عبارتند از:

- دموی کدگذار-کدگشا[1]
- دموی کدگذار[2]
- دموی کدگشا[3]

نحوه‌ی سرهم کردن[4] سخت افزاری *DVEVM* در پیوست "الف" و اجرای دموها در *DEVEM* در پیوست "ب" آورده شده است.

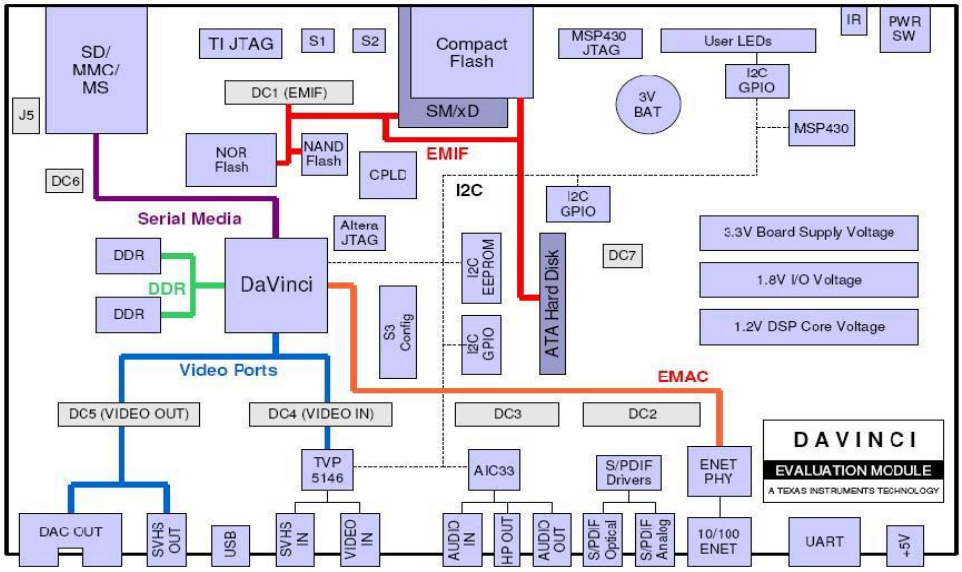
[1] *Encode Decode Demo*
[2] *Encode Demo*
[3] *Decode Demo*
[4] *Assemble*

در شکل(۱-۴)، برد گسترش نشان داده شده است. در شکل (۲-۴)، محتوای کیت *DVEVM* به نمایش در آمده است و در نهایت، در شکل(۳-۴)، نمایش بلوکی *DVEVM* را مشاهده می‌کنیم.



شکل ۱-۴- برد گسترش و اجزای جانبی موجود برروی آن

شکل ۲-۴- محتوای کیت *DVEVM*



شکل ۳-۴- نمایش بلوکی *DVEVM*

حال که نمایش بلوکی را نشان دادیم، توضیحی مختصر در مورد برخی از اجزا می‌دهیم.

DM644x واقع برروی *Davinci EVM* با اجزای جانبی روی برد از طریق پینهای واسطه *EMIF*[1] شانزده بیتی مواجه می‌شود. حافظه *DDR2* نیز به باس ۳۲ بیتی مربوط به خود متصل است. باس *EMIF* همچنین به فلش، *SRAM* ، *NAND* و *daughter card* متصل است.

رمزگشای برروی برد و رمزگذارهای روی تراشه وظیفه‌ی روبرو کردن *DM644x* با جریان های ویدیویی را دارند. یک رمزگشا و چهار مبدل دیجیتال به انالوگِ بروی تراشه، برای *EVM* استاندارد می‌باشد. توابع نمایش برروی صفحه[2] در نرم‌افزاری که برروی پردازنده *DM644x* قرار دارد، به کار رفته‌اند.

کدکِ *AIC33* که برروی برد قرار دارد، این امکان را به پردازنده سیگنال دیجیتال می دهد که سیگنال‌های آنالوگ صوت را دریافت کرده و انتقال دهد. باس I^2C[3] برای کنترل رابط کدک به کار می‌رود

[1] *External Memory Interface*
[2] *Screen On Display = OSD*
[3] *Inter-Integrated Circuit*

در حالی که *McBSP*[1] جریان صدا را کنترل می‌کند. پردازنده با سیگنالهای صدا از طریق پایه‌های ۳,۵ میلیمتری که مربوط به ورودی میکروفون، ورودی خط و خروجی صدا می‌باشند، روبرو می‌شود.کدک می‌تواند ورودی خط را بعنوان ورودی فعال انتخاب کند.

EVM شامل ۸ عدد *LED*، رابط*IR* و ساعت زمان حقیقی می‌باشد که می‌تواند بعنوان فیدبک عمل کاربر به کار رود. این رابط‌ها با نرم‌افزاری که برروی *MSP430* به کار رفته‌اند و از طریق خواندن و نوشتن در رجیسترهای I^2C، قابل دسترس می‌باشند.

کارت‌های رسانه[۲] رابط*ATA* ، رابط‌های شبکه *MAC* و *VLYNQ* نیز اجزای جانبی مجتمع شده بر روی پردازنده*DM644x* می‌باشند.

یک منبع ولتاژ ۵*v* نیز برای روشن کردن برد به کار رفته است. رگولاتورهای سوئیچینگ ولتاژ که بر روی برد قرار دارند، ۱,۲*v* برای هسته پردازنده سیگنال، ۳,۳*v* برای اجزای جانبی و ۱,۸*v* برای حافظه و ورودی/خروجی *DM644x* فراهم می‌کنند. برد تا هنگامی که این منابع در مرحله تعیین عملیات هستند، در حالت ریست می‌باشد.

Code Composer از طریق *JTAG* که به متصل کننده‌ی ۲۰ پینی *JTAG* برروی برد وصل می‌شود، با *EVM* ارتباط بر قرار می‌کند.

همانطور که ذکر شد، پردازنده *DM6446* شامل یک پردازنده *ARM* و یک پردازنده *C64x+* می‌باشد. هنگامی که دموها اجرا می‌شوند،کد این دموها در پردازنده *ARM* تحت لینوکس به اجرا درآمده و به کدکهای صوت، ویدیو وگفتار که در *DSP* قرار دارند، دسترسی خواهند داشت.

توجه داشته باشید که در *DVEVM*، پردازنده *ARM* در واقع یک محیط گسترش نرم‌افزاری است. بدین معنی که گسترش‌دهندگان می‌توانند سورس کد خود را گسترش دهند، تغییر داده یا اصلاح کنند. در

---

[1] *Multichannel Buffered Serial Port*
[2] *Media Cards*

حالیکه پردازنده *C64x+*، هیچ سورس کدی را فراهم نمی‌کند و کاربر به توابعی که توسط نرم‌افزار مربوط به این پردازنده فراهم شده است و در *DVEVM* قرار دارد، محدود می‌شود.

تا به حال توضیحاتی در مورد *DVEVM* داده شد، در زیر ویژگیهای کلیدی این ماژول بیان شده است:

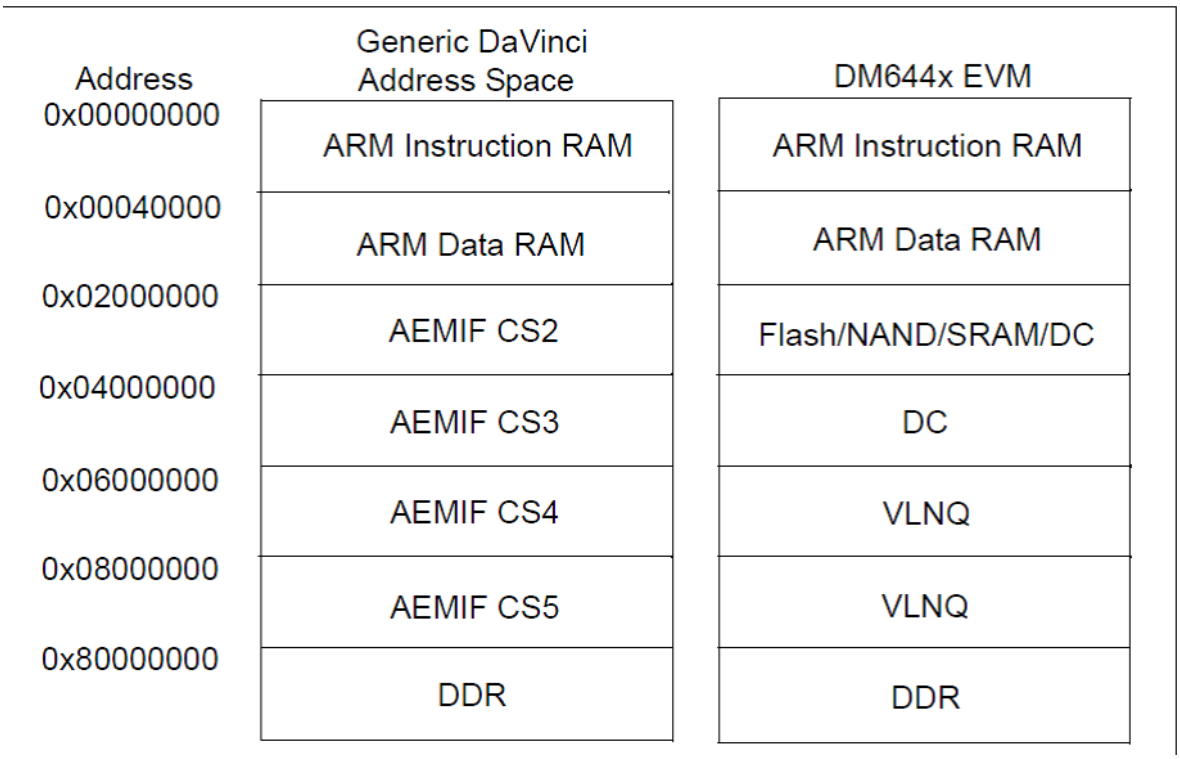
- یک پردازنده*DM644x* که شامل یک پردازنده *ARM* که می‌تواند تا فرکانس *Mhz* ۳۰۰ و یک پردازنده *C64xx* که می‌تواند تا فرکانس *Mhz* ۶۰۰ عمل کند، می‌باشد.
- یک پورت ورودی ویدیو که ویدیو های *Composite* و *S* را حمایت می‌کند.
- ۴ پورت خروجی دیجیتال به آنالوگ ویدیو.
- یک حافظه *DDR2 DRAM* به ظرفیت *Mbyte*s ۲۵۶.
- پورت سریال *UART* و رابط کارت سریال (*SD card, xD card, SM card, MS card,MMC,Media Card interface*).
- حافظه فلش غیرفرار به ظرفیت *Mbytes* ۱۶، حافظه فلش *NAND* به ظرفیت *Mbytes* ۶۴، حافظه *SRAM* به ظرفیت *Mbytes* ۴.
- کدک استریوی *AIC33* .
- واسط *USB2* .
- *10/100 MBS Ethernet Interface*
- واسط کنترل *IR*، ساعت زمان حقیقی توسط *MSP430* .
- بار شدن بوت قابل پیکربندی.
- رابط*JTAG*.
- ۸ عدد *LED*
- یک منبع ولتاژ (*v* ۵+)

- اتصال‌کننده های گسترش یافته برای استفاده از *daughter card* (کارتی که برروی کارت اصلی نصب میگردد).
- واسط *ATA*، واسط *Vlynq* .
- واسط *SPDIF*

از مسائل مهم دیگری که در مورد *DVEVM* مهم است، نقشه‌ی حافظه مربوط به *DM6446* و پیکربندی سوئیچ‌هایی است که برروی برد قرار دارد.

پردازنده‌های خانواده داوینچی، فضای حافظه بزرگی دارند که بصورت بایتی آدرس‌پذیر می‌باشد. کد برنامه و اطلاعات می‌توانند در هر جایی از فضای حافظه متحد شده قرار گیرند. آدرس ها، بسته به کاربرد سخت‌افزار دارای چندین سایز می‌باشند.[۳۷]،[۳۸]،[۳۹]

در شکل (۴-۴) نقشه حافظه نشان داده شده است. قسمت سمت چپ مربوط به پردازنده عمومی *DAVINCI* است در حالیکه قسمت سمت راست، جزئیات این که هر ناحیه چگونه در *DM644x* مورد استفاده قرار گرفته است را نشان می‌دهد. بطور پیش فرض حافظه داخلی در شروع فضای حافظه قرار می‌گیرد. قسمتهای حافظه را از طریق نرم افزار می‌توان دارای نقشه جدید کرد. رابط *EMIF* دوبخش دارد. یک *EMIF* تخصیص داده شده، مستقیما با حافظه *DDR2* مواجه می‌شود. *EMIF* دوم، چهار ناحیه آدرس‌پذیر مجزا دارد که فضاهای تراشه نامیده می‌شوند.(*CS2-CS5*). فلش، فلش *NAND* یا *SRAM* در داخل فضای *CS2* نگاشت شده‌اند و توسط سوئیچ *J4* که برروی برد قرار دارد، قابل انتخاب می‌باشند. *Daughter cards* از *CS2* و *CS3* استفاده می‌کند. در حالیکه *CS2* برای *Daughter card* مورد استفاده قرار میگیرد، *J4* باید بصورت مناسبی تنظیم گردد. *CS4* و *CS5* برای واسط *VLYNQ* بر روی برد نگه داشته شده اند.



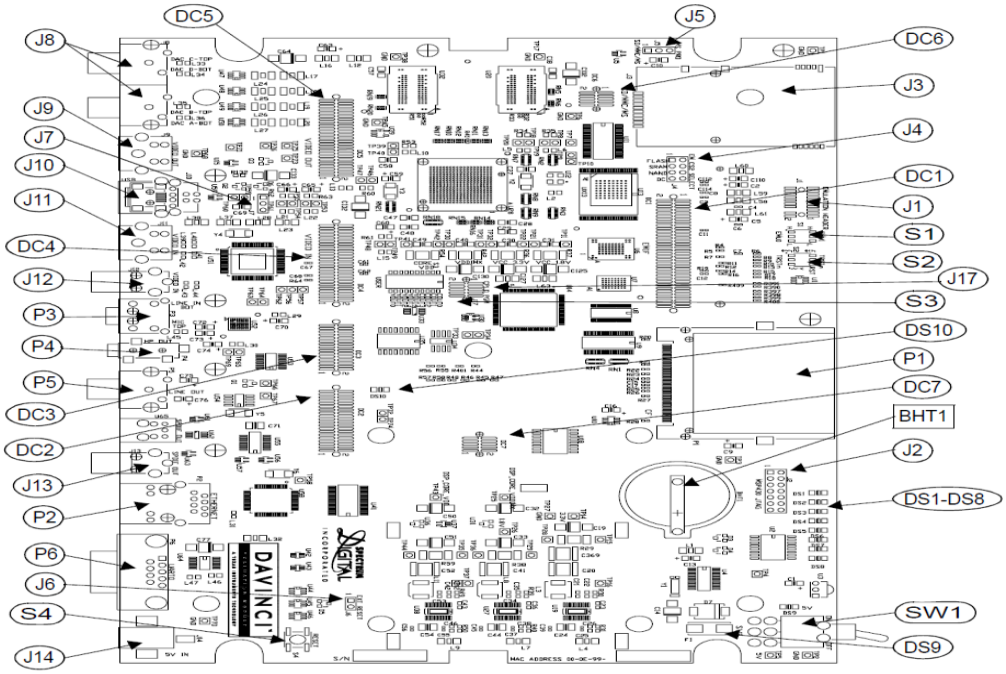
شکل ۴-۴- نقشه حافظه در پردازنده‌های *DAVINCI*

EVM یک سوئیچ پیکربندی ۱۰ موقعیتی دارد که امکان کنترل حالت عملیاتی پردازنده را هنگامی که از حالت ریست خارج می‌شود، به کاربر می‌دهد. سوئیچ پیکربندی برروی برد با برچسب *S3* مشخص شده است. سوئیچ *S3* حالت بوت را پیکربندی می‌کند. این حالت بوت هنگامی که پردازنده سیگنال دیجیتال اجرا کردن را شروع می‌کند، مورد استفاده قرار می‌گیرد. شکل زیر تنظیمات را برای سوئیچ *S3* نشان می‌دهد.

Position	Name	Function	Boot Mode
1	COUT0	Boot Mode 0	00 - Boot from ROM NAND 01 - Boot from AEMIF 10 - Boot from ROM HPI 11 - Boot from ROM UART
2	COUT1	Boot Mode 1	
3	COUT2	Bus width: 0=8 bit, 1=16 bit	
4	COUT3	0=ARM boots DSP, 1=C64xx self boots	
5	YOUT4		
6	YOUT3		
7	YOUT2		
8	YOUT1		
9	YOUT0		
10	USER	For Demos	

شکل ۵-۴- نحوه پیکربندی سوئیچ *S3* برای حالت‌های مختلف بوت

در شکل (۶-۴)، محل قرارگیری سوئیچ‌های *S3* و *J4* را نشان می‌دهد.[۳۷]



شکل ۶-۴- محل قرار گیری سوئیچ‌های *J4* و *S3* و دیگر سوئیچ‌ها

۳-۴-استانداردها و ماژول‌های نرم‌افزاری مربوط به *DAVINCI*

۱-۳-۴- *DSP/BIOS* و لینوکس

می‌دانیم که برخی از *SOC*ها شامل دو پردازنده هستند: یک پردازنده هدف عمومی و یک پردازنده پردازش سیگنال دیجیتال. بنابراین باید عاملی باشد تا وظایف را به طور موثر بر روی هر دوی این پردازنده‌ها برنامه‌ریزی کند .

برای *DSP*، برنامه ریز وظیفه، *DSP/ BIOS* نامیده می شود و برای *GPP* دنیای شگفت انگیز لینوکس.

BIOS معمولاً یکی از آن سرنام‌های غیر قابل استفاده است.[۱]

BIOS را این گونه تعریف می‌کنیم: هسته‌ی چند وظیفه‌ای زمان حقیقی رایگان (سیستم عملیاتی کوچک) که برای خانواده‌ی *TMS320* از *DSP* های شرکت *Texas Instrument* ایجاد شده است. بخش *DSP* از *DSP/ BIOS* را در نظر بگیرید. این بخش برای قسمت *DSP* از وسایل چند هسته ای غیرهمگون *TI* ساخته شده است.

DSP / BIOS برای هر گونه استفاده‌ای مناسب نیست اما برای برنامه‌ریزی وظایف زمان حقیقی *DSP* به خاطر سه خاصیت زیر ایده آل است :

- مقیاس پذیری
- سرعت
- تأخیر کم[۲]

---

[۱] سرنام یعنی نامی که از مخفف کلمات دیگر به دست آمده باشد .

[2] *Low Latency*

این سه ویژگی باعث می شود که *DSP/ BIOS* یک برنامه‌ریز زمان حقیقی باشد و از مشکلات اجرایی ناخواسته در برنامه های شما جلوگیری کند. همیشه این بحث وجود دارد که آیا *DSP/ BIOS* یک سیستم عملیاتی است یا فقط یک هسته ی زمان بندی است؟ موضوعی که می‌توان اظهار کرد این است که *DSP/ BIOS* یک زمان‌بند زمان‌حقیقی ایده‌آل برای آن دسته از وظایفی است که *DSP* طراحی شده است تا آنها را اجرا کند. حال هر یک از موارد فوق را به اختصار توضیح می‌دهیم.

- مقیاس پذیری

اگر کوچک بودن را زیبا در نظر بگیریم، *DSP/ BIOS* قطعاً زیبایی خواهد بود. یک هسته‌ی برنامه‌ریزی کننده کوچک (و زیبا) به شما کمک می‌کند تا از منابع روی تراشه برای مواردی که واقعاً رخ می‌دهند مانند الگوریتمهای ویدیو، تصویر، گفتار و صوت استفاده کنید. برای هسته‌ی برنامه ریزی کننده[1] روی *DSP* ضروری است که مقیاس پذیر باشد. بنابراین از مواردی استفاده می‌کند که واقعاً برای انجام کار نیاز است. از آنجایی که آن هسته با کل کد رابطه برقرار نمی‌کند، *DSP/ BIOS* می‌تواند دقیقاً هنگامی که مورد نیاز است اجرا گردد. همچنین *DSP/ BIOS* می‌تواند در حافظه ی خارجی آهسته‌تر (از لحاظ سرعت) قرار گیرد .

- سرعت

یک هسته‌ی برنامه ریزی کننده‌ی کوچک *DSP* به اندازه‌ی کافی خوب نیست اگر تعداد زیادی *MIPS*[2] را بر روی هسته پردازنده مصرف کند. هر چرخه‌ی در نظر گرفته شده توسط زمان‌بند، یکی کمتر از چرخه موجود برای کاری است که قرار است *DSP* انجام دهد که این کار، الگوریتمهای پردازش سیگنال نامیده می‌شود. در این مورد *DSP/ BIOS* می‌درخشد زیرا تمام ماژولهای آن برای هسته *DSP* به طور ایده‌آلی

[1] *scheduling kernel*
[2] *Million Instructions Per Second= MIPS*

بهینه شده اند. همچنین برای مدیریت اطلاعات جاری شونده[1]، که دقیقاً چیزی است که *DSP* نوعی در کاربردهای *Video/ audio* با آن مواجه می‌شود، بسیار مناسب است.

- تأخیر کم

یک سیستم عملیاتی *DSP* باید قطعی[2] باشد بدین معنی که سیستم باید در هر اجرا نتایج یکسانی را بدون هیچ گونه ابهام تولید کند. اگر یک سیستم عملیاتی، کاری را در *1ms* انجام دهد این زمان، زمان میانگین است یا بهترین حالت؟ اگر بهترین حالت باشد، بدترین حالت چیست؟ اگر بدترین حالت *50ms* باشد و شما یک سیستم ویدیویی را اجرا کنید که اتفاقات مهم در هر *33.3ms* رخ می دهد، آنگاه چه اتفاقی خواهد افتاد؟

تأخیر، در واقع دیرکرد بین زمانی است که عملیات آغاز می‌شود تا زمانی که موثر واقع می‌شود. تأخیر یکی از عواملی است که سیستم های عملیاتی زمان حقیقی (مانند *DSP/ BIOS*) را از سیستمهای عملیاتی مشابه بزرگتر جدا می‌کند.

تا به حال در مورد *DSP/BIOS* توضیحاتی ارائه شد، حال به توضیح لینوکس که سیستم عاملی ایده‌آل برای پردازنده‌های *DAVINCI* است و نحوه‌ی عملکرد آن می‌پردازیم.

در دستگاه های *DAVINCI*، پردازنده‌ی هدف عمومی یکی از اعضای خانواده *ARM* است. این هسته‌ها گروهی از اجزای جانبی دارند که به هسته متصل است مانند: پورت های سریال، پورت *USB*، پورت *Ethernet* ، *Multi media cards* و ... . چندین ویژگی لینوکس که در فصل اول ذکر شد، آن را انتخاب مناسبی برای پردازنده هدف عمومی کرده است. لینوکس انواع مختلفی دارد. یک نوع آن که برای اجرا در برنامه های سبکتر پیکربندی شده است، لینوکس جایگذاری شده[3]نامیده می شود. این نوع لینوکس در پردازنده‌ی هدف عمومی مورد استفاده قرار می‌گیرد و زمانی که یک سیستم‌عملیاتی محسوب می‌شود،

[1] *streaming data*
[2] *deterministic*
[3] *embedded Linux*

توسعه‌دهندگان برنامه مشتاقند که از آن پیروی کنند. خیلی از برنامه های جدید منبع باز هستند و بدون هیچ هزینه‌ای در اختیار کاربر قرار می‌گیرد. یک مثال خوب در مورد اینگونه برنامه‌ها، می توان از *Gstreamer* نام برد که یک چارچوب چند رسانه ای مستقل از سیستم‌عامل[1] است که به لینوکس منتقل شده و از برنامه های چند رسانه‌ای حمایت می‌کند.

برای مثالی دیگر می‌توان از موتور کدک[2] نام برد که یک برنامه ابتدایی است و بر روی *DSP/ BIOS* روی *DSP* اجرا می‌شود. موتور کدک، یک چارچوب نرم‌افزاری منحصر به فرد است که توسط شرکت *TI*، طراحی، نگهداری و حمایت شده است تا مخصوصاً تمام الگوریتم های پردازش سیگنال را به طور موثر بر روی *DSP* از *SOC* اجرا کند .

علاوه بر *Linux* سیستم‌عامل های دیگری نیز بر روی *DAVINCI* حمایت می‌شوند . دو نمونه از آنها عبارتند از : *Microsoft win CE*و *Green hills Integrity* .

برای اینکه دو پردازنده هیبرید (*GPP, DSP*) به طور موثر با یکدیگر کار کنند، شما به بلوکهای بیشتری نیاز دارید. گسترش‌دهندگان برنامه‌ی لینوکس نمی‌خواهند جزئیات دقیق *DSP* و برنامه‌ریز *BIOS /DSP* را یاد بگیرند. این بدین معنی است که کارهایی که *DSP* قادر به انجام آن است باید به یک روش ساده و موثر خلاصه گردد. این امر سبب می‌شود که نرم افزار کاربردی موتور کدک[3] که بر وی *DSP* اجرا می‌گردد، دارای یک قسمت همراه باشد که بر روی پردازنده *ARM* اجرا می‌شود. این دو قسمت چارچوب با یکدیگر از طریق قسمت نرم‌افزاری دیگری به نام *DSP/ BIOS link* ارتباط برقرار می‌کنند. بنابراین گسترش‌دهنده‌ی برنامه نیازی نیست که در مورد جزئیات *DSP* نگران باشد. ضمناً گروه دیگری از گسترش‌دهندگان در حال ساختن الگوریتم‌های پردازش سیگنال واقعاً باهوشی هستند که بر روی *DSP*

[1] *OS- Independent Multimedia Frame Work*
[2] *Codec Engine*
[3] *Codec Engine Framework Application Software*

اجرا می‌شود. شرکت *TI* از این الگوریتم‌های پردازش سیگنال بعنوان *expressDSP Digital Media Software* یاد می‌کند. برای اینکه این الگوریتمها به راحتی بر روی *DSP* اجرا شوند، نیاز است که قوانین و خطوط راهنمای[1] ویژه‌ای را دنبال کنند تا به راحتی و به درستی در سیستم نهایی قرار گیرند. این قوانین و خطوط راهنما، بر اجزای الگوریتم گسترش‌دهندگان اعمال می‌شود. همچنین بهتر است مصرف‌کنندگان الگوریتم، اجزای الگوریتم خود را قبل از قرار دادن در سیستم نهایی، چک و تایید کنند. اگر الگوریتمها قوانین را رعایت نکنند، وجود آشفتگی و بی‌نظمی تقریباً قطعی خواهد بود و در دنیای نرم‌افزار، آشفتگی و بی نظمی برابر است با حوادث غیر قابل توضیح که کشف یا توضیح داده نمی‌شود و تنها روزها و هفته‌های ارزشمند را هدر می‌دهد.

۲-۳-۴- نرم افزار رسانه ای دیجیتال[2]

دستگاه‌های الکترونیکی جدید از پردازش سیگنال دیجیتال برای اجرا کردن، ذخیره کردن، فرستادن و بازیابی انواع مختلف رسانه‌ها[3] استفاده می‌کنند. برای اینکه تمام این موارد رخ دهد، طراحان از کدکهای نرم‌افزاری پیچیده‌ای استفاده می‌کنند که قادرند ویدیو، تصاویر ممتد، صوت و گفتار را فشرده کند و از حالت فشرده خارج نماید، انتقال دهد و دوباره اجرا کند. برای اینکه *DAVINCI* کاملاً قادر به انجام این امر باشد، طراحان یک یا چند الگوریتم را برای اجرا برروی دستگاه انتخاب می‌کنند. این الگوریتمها، *expressDSP Digital Media Software* نامیده می‌شوند. در این بخش سه استاندارد کدک برای *DAVINCI* را توضیح می‌دهیم. ابتدا از مهمترین آنها که در واقع پدربزرگ آنها نیز می‌باشد، یعنی *XDAIS*[4]، استاندارد قابلیت همکاری الگوریتم *expressDSP*، شروع می‌کنیم. بعد نگاهی به *XDM*[5]،

---

[1] *Rules and Guidlines*
[2] *Digital Media Software*
[3] *Media*
[4] *expressDSP Algorithm Interoperability standard*
[5] *expressDSP Digital Media*

رسانه‌ی دیجیتال*expressDSP*که درواقع یک توسعه از *XDAIS* است که رابط‌های استاندارد برای کلاس‌های ویژه‌ای از کدکها را فراهم می‌کند، می‌اندازیم و در نهایت *RTSC*[1] که یک مجموعه از ابزار برای اجزای نرم افزاری زمان‌حقیقی است، را بیان می‌کنیم. این استانداردها، توسعه‌دهندگانی که اطلاعات کمی در مورد نحوه‌ی عملکرد داخلی الگوریتمهای پیچیده‌ی تصویر/صوت دارند، را قادر می‌سازند تا به راحتی چندین کدک را در داخل سیستم های خود قرار داده و از آنها استفاده کنند .

حال نگاهی اجمالی به استانداردهای ارائه شده از طرف شرکت*TI* می‌اندازیم.

در حال حاضر صدها الگوریتم مختلف از شرکت *TI* و شرکت های واسطه در دسترس است و الگوریتمهای جدید نیز در حال تولید هستند. برخی الگوریتمها استانداردهای صنعتی موجود را (مانند *MPEG4*، *H.264* و ...) را اجرا می‌کنند و برخی دیگر با استانداردهای جدید مواجه می‌شوند. گاهی اوقات نیز الگوریتمها اختصاصی هستند زیرا تولید کنندگان آنها معتقدند که این الگوریتمها « چاشنی اضافی» دارند که طراحی آنها را از بسته الگوریتمها[2] متمایز می‌سازد. متاسفانه هنگامی که انعطاف‌پذیری زیاد باشد، خطر شکل‌گیری نرم‌افزارهای هرز و بی‌استفاده نیز وجود دارد. منظور از برنامه‌های هرز، برنامه‌هایی است که روی کاغذ خوب به نظر می‌رسند ولی در سیستمهای حقیقی ایجاد مشکل می‌کنند. این امر سبب می‌شود تا استانداردهای بیشتری نیاز شود . *Texas Instrument*یک مجموعه استاندارد را به کار برده است که امکان استفاده از کدکهای چند رسانه ای بر روی *DAVINCI* با کمترین مشکل ممکن را فراهم کرده است. حال به توضیح استانداردها می‌پردازیم.

[1] *Real–Time Software Components*
[2] *Pack*

۱-۲-۳-۴- *XDAIS* اطمینان از خوب کار کردن کدک ها

استاندارد قابلیت همکاری الگوریتم *expressDSP* در سال ۱۹۹۹ معرفی شد. *XDAIS* یک مجموعه از قوانین است که سبب می‌شود کدکها به صورت مناسب عمل کنند و امکان یکپارچه‌سازی بی‌نقص سیستم را فراهم می‌کند. قبل از *XDAIS* گسترش دهندگان کدک تقریباً قانون خاصی نداشتند. مشکلات از آنجایی شروع شد که یکپارچه‌سازان سیستمها قصد داشتند که یک یا چند کدک را انتخاب کنند و آنها را با هم کار به کار ببرند. برخی از نتایج ضعیف گسترش کدکها قبل از *XDAIS* در زیر آمده است :

- کدکها نمی‌توانستند در حافظه سیستم دوباره قرار بگیرند و یکپاچه سازان سیستم مجبور بودند تا مکانهای ویژه ای برای کدکها فراهم کنند .
- برخی از کدکها مستقیماً با سخت افزار داخلی کار می‌کردند و این امر احتمال برخورد با پردازشهای دیگری که اکنون از سخت‌افزارها استفاده می‌کنند، را افزایش می‌دهد .
- برخی از کدکها قابل بازگرد و دوباره واردشونده[1] نبودند. این امر از عملکرد مناسب و چندکاناله بودن کدک جلوگیری می‌کند .
- وقفه ها برای دوره‌های بیش از اندازه طولانی غیرفعال می‌شوند و این امر سبب می‌شود که دیگر پردازش ها نتوانند حرفی برای گفتن داشته باشند .

XDAIS در حالت کلی به سه گروه مجزا تقسیم می‌شود: قوانین ، خطوط راهنما و رابطهای عمومی.

در ادامه هر کدام از بخش ها به طور مختصر توضیح داده می‌شود .

الف)قوانین *XDAIS*

[1] *Re- Entrant*

نیازی به گفتن نیست که قوانین *XDAIS* واجبند. قوانین باید برای یک الگوریتم پیروی شوند تا تطابق[1] آن با استاندارد بیان شود. در حال حاضر ۴۶ قانون وجود دارد. هر چند این عدد ممکن است بزرگ باشد، اما تعداد زیادی از قوانین بر یک اساس هستند. دیگر قوانین ممکن است قراردادی به نظر برسند اما آنها بر توافق و سازگاری پافشاری می‌کنند. به عنوان مثال قانون ۲۵ نیاز دارد به اینکه تمام الگوریتمهای بر اساس *C6x* در فرمت پایان-کوتاه[2] باشند. خبر خوب برای مصرف کننده الگوریتم (به جای ایجاد کننده اصلی)، این است که لازم نیست تعداد قوانین و اینکه دقیقاً چه کارهایی انجام می‌دهند، را بدانند. تنها امری که از اهمیت خاصی برخوردار می‌باشد، این است که بدانند قوانین پیروی شده‌اند. اگر الگوریتم از قوانین پیروی کرده باشد، به آسانی در سیستم قرار می‌گیرد و همانطور که انتظار می‌رود، عمل می‌کند . برخی از مهمترین قوانین *XDAIS* عبارتند از :

- قانون ۲

تمام الگوریتمها در داخل یک محیط انحصاری، باید بازگشتی باشند: این امر مارا مطمئن می‌سازد که الگوریتمها ، مثالهای چندگانه را حمایت می‌کنند .

- قانون ۴

تمام کد الگوریتم باید جابه‌جاپذیر باشد : این قانون مطمئن می‌سازد که یکپارچه‌سازان (ایجادکنندگان) سیستم، آزادانه می‌توانند کدکها را در جایی که مناسب است قرار دهند، نه محلهایی که خالقان الگوریتم ممکن است مجبور کنند .

- قانون ۶

[1] *Compliance*
[2] *little endian*

الگوریتمها هرگز نباید به هیچ یک از اجزای جانبی، بطور مستقیم دسترسی داشته باشند : این قانون، مشکل دسترسی پیدا کردن الگوریتمهای هرز به اجزای جانبی در حالی که فرآیند دیگر ممکن است در حال حاضر از آن جز جانبی استفاده کند، را حل می‌کند .

- قانون ۱۲

تمام الگوریتم ها باید رابط *IALG*[1] را به کار ببرند : *IALG* یک رابط مدیریت منبع حافظه است که تمام الگوریتمها آن را به کار می‌برند بدین‌منظور که به سازندگان سیستم اجازه دهند، منابع حافظه را آن طور که صلاح می‌دانند پخش کنند .

- قانون ۲۳

تمام الگوریتمها باید بدترین حالت تاخیر وقفه را برای هر عملیات مشخص کنند : قوانین ۲۴-۱۹، قوانین مشخص سازی عملکرد نامیده می‌شوند که خالقان الگوریتم را مجبور می‌کند تا استفاده از منابع و عددهای عملکرد را سندسازی کنند .

- قانون ۲۵

تمام الگوریتم های *C6x* باید در فرمت پایان-کوتاه تهیه شود : این قانون یکی از قوانین بالقوه اختیاری است. پایان-بلند[2] چه می‌شود؟ با انتخاب حداقل یک فرمت برای هر نفر برای گسترش، سازندگان سیستم گارانتی خواهند شد که حداقل یک مجموعه از الگوریتمهای پایان-کوتاه به جای یک مجموعه مخلوط از الگوریتمهای پایان-کوتاه و پایان-بلند، که معمولاً دردسرساز هستند، خواهند داشت .

ب)خطوط راهنمای *XDAIS*

[1] *IALG interface*
[2] *Big-Endian*

XDAIS قوانینی دارد و همچنین دارای خطوط راهنما است. اختلاف این دو در چیست؟ خطوط راهنمای *XDAIS* در واقع شبیه آن راهنمای رژیمی است که به شما «پیشنهاد» می‌کند که فرزند شما چه مقدار شکر باید مصرف کند. خطوط راهنما بیشتر اوقات عملی هستند اما همیشه استثناهایی را پیدا خواهید کرد که می‌توانید آنها را بشکنید. برخی از مهمترین خطوط راهنمای *XDAIS* عبارتند از :

- خط راهنمای ۵

الگوریتم ها باید سایز مورد نیاز پشته را در مینیمم مقدار نگه دارند : سازندگان سیستم می‌خواهند بدانند که سیستم های نهایی آنها چه مقدار سایز پشته کلی نیاز خواهد داشت. در حالت کلی آنها می‌خواهند که سایزها را قابل مدیریت قرار دهند بنابراین طرحی برای بدترین حالت استفاده بیش از حد از منابع ندارند. خط راهنمای ۵ پیشنهاد می کند که هر کدک نقش خود را به منظور کمک به هدف نهایی سایز پشته ایفا کند .

- خط راهنمای ۱۲

تمام الگوریتم ها باید در هر دو حالت پایان-کوتاه و پایان-بلند تهیه شوند : این خط راهنما در واقع در همراهی قانون ۲۵ است. هر چند قانون ۲۵ نیاز دارد که گسترش دهنده الگوریتم تمام الگوریتم ها را حداقل در فرمت پایان-کوتاه تهیه کند، بهتر است اگر گسترش دهنده هر دو کاربرد پایان-کوتاه و پایان-بلند از الگوریتم را آماده کند .

ج)رابطهای *XDAIS*

استاندارد *XDAIS* استفاده از یک یا چند رابط که روش درخواست کدکها را همگون می‌سازند و با منابع تأمین شده‌اند، را اجباری می‌کند. اساسی که در پشت این رابطها است، مطمئن شدن از این موضوع است که تخصیص منابع حیاتی به خوبی صورت می‌گیرد. بنابراین به جای الگوریتمهای احمقانه که فقط تمام منابع خوب را تصرف می‌کنند، رابطهای *XDAIS* نحوه‌ی درخواست الگوریتمها مبنی بر استفاده از منابع را

همگون می‌سازند. سپس سازندگان سیستم در مورد مدت زمان طراحی یا زمان اجرا، تصمیم‌گیری می‌کنند .

استاندارد *XDAIS* سه رابط هسته‌ای تعریف می‌کند :

- *IALG*

این واسط توسط همه‌ی الگوریتمها مورد نیاز است و نیازمندیهای حافظه هر الگوریتم را کنترل می‌کند .

- *IDMA3*

این نسخه جایگزین نسخه‌های قبلی *IDMA2* و *IDMA* شده است. *IDMA3* ، اگر الگوریتم انواع ویژه ای از منابع *DMA* را نیاز داشته باشد، مستلزم خواهد بود .

- *IRES*

این واسط هنگامی که الگوریتم به منابع دیگر مانند شتاب دهنده‌های سخت افزاری نیاز داشته باشد مستلزم خواهد بود .

۲-۲-۳-۴- *XDM* : واسطه های استاندارد برای طبقات رایج کدک ها

XDAIS قوانین، خطوط راهنما و رابطها را برای تمام کدکها به کار می‌برد. اما محدودیت هنگامی ظاهر می‌شود که *XDAIS* در مورد طبیعت رابطهای استفاده شده در کدکهای ویدیو، تصویر، صوت و گفتار اطلاعاتی ندارد. در نتیجه، بیشتر گسترش‌دهندگان کدک عادت می‌کنند تا رابطهای منحصر به فرد خود را تولید کنند. این امر خوب بود تا زمانی که سازندگان سیستم می‌خواستند یک کدک ویدیو را جایگزین دیگری کنند یا بطور ساده‌تر، علامت تجاری[1] *A* را با علامت تجاری *B* معاوضه کنند. با رابطهای متفاوت کدک، عملیات اضافی به منظور جاگذاری کردن کدک جایگزین در سیستم، مورد نیازاست. *XDM* ،

[1] *Brand*

استاندارد اصلی *XDAIS* را به منظور حل نمودن این مسئله خاص، گسترش و بسط می‌دهد. *XDM* رابطهای رمزگذار و رمزگشا را برای چهار طبقه الگوریتم مشخص می‌کند : ویدیو ، صوت ، گفتار و تصویر . رابطهای *XDM* ساده و سبک بوده و در جایی که توسعه دهندگان به کدک نیاز داشته باشند، قابل تعمیم است. مزیت اصلی گسترش رابطهای *XDM* بر روی طبقات ویژه‌ای از کدکها ، قابلیت تعویض آن است . برای مثال، جایگزین کردن یک کدک ویدیویی *XDM MPEG4* از یک فروشنده با نسخه‌ی دیگری از آن فروشنده، آسان خواهد بود اگر برنامه فقط در مورد واسط *XDM* (که تغییر نمی کند) اطلاعات داشته باشد و لازم نیست در مورد ویژگیهای کدک ویدیویی خاصی که به کار رفته است، نگران باشد .

هنگامی که با کدکها کار می‌کنید ممکن است به کلمه ی *VISA*[1] برخورد کنید. *VISA* نام یک رابط مرحله‌ی کاربردی[2] است که برای فراخوانی چهار طبقه استاندارد از کدک های *XDM* به کار می‌رود : ویدیو ، تصویر ، صوت و گفتار . استاندارد *XDM*، هشت رابط متفاوت و عمومی کد را معرفی می‌کند. این هشت رابط عبارتند از :

۱- *IVIDENCE x*: برای رمزگذارهای ویدیو

۲- *IVIDDEC x*: برای رمزگشاهای ویدیو

۳-*IAUDENC x*: برای رمزگذارهای صوت

۴-*IAUDDEC x*: برای رمزگشاهای صوت

۵-*ISPHENC x* : برای رمزگذارهای گفتار

۶-*ISPHDEC x*: برای رمزگشاهای گفتار

۷-*IIMGENC x*: برای رمزگذارهای تصویر

۸-*IIMGDEC x* : برای رمزگشاهای تصویر

---

[1] *Video,Image,Speech,Audio*
[2] *Application Level Interface*

مانند تمام اشیاء خوب که در طول زمان تکامل پیدا می‌کنند، برخی از رابطهای اضافی مانند *IVID DEC2*و *IVID DEC1* معرفی شده‌اند. این رابطها، بهبودی رابطهای اصلی همانند فعال کردن مدیریت پیشرفته بافر روی بسترهای *DAVINCI* را فراهم می‌کنند .

پیشنهادی که برای تولیدکنندگان کدک مطرح می‌شود این است که آخرین رابطها را به کار ببرند زیرا این رابطها معمولاً قابلیتی را فراهم می‌کنند که می‌توانند بدون بسط دادنهای عرفی[1] مورد استفاده قرار گیرند.

۳-۲-۳-۴- *RTSC* بسته های متعارف و همگون شده[2] برای همه ی کدک ها

اجزای نرم‌افزاری زمان-حقیقی، یک ابتکار منبع باز است که اولین بار توسط شرکت *TI* برای همگون سازی اجزای جایگذاری به کار رفت. مفاهیم نرم‌افزار جزئی[3]، چندین سال است که وجود داشته است . کل مفهوم به دوباره استفاده کردن از اجزای نرم‌افزار توسط چندین کاربر بستگی دارد. برای اینکه استفاده دوباره از اجزا امکان پذیر باشد، به استانداردهایی نیاز است که انتقال آسان اجزا را بین مصرف‌کنندگان و تولیدکنندگان آسان می‌کند .

حال این سؤال پیش می‌آید که چرا*RTSC*؟

به چند دلیل تکنولوژی جزئی[4]خیلی دیر به دنیای برنامه‌نویسی جایگذاری‌شده پا گذاشت. شاید چندین دلیل کلیدی برای این امر وجود داشته باشد. یک دلیل سرباری[5] است که با مدل دهی جزئی[6] همراه شده است .

کلمه‌ی «سربار» معمولاً با «برنامه نویسی جایگذاری شده » جور در نمی‌آید. سیستم های جایگذاری شده، همگی در مورد عملکرد و طراحی سبک هستند. بنابراین کلمه‌ی «سربار» کلمه‌ی مناسبی نیست .

---

[1] *Custom Extensions*
[2] *Standardized*
[3] *Component Software*
[4] *Component Technology*
[5] *Overhead*
[6] *Component Models*

RTSC برای برنامه‌نویسی جاگذاری شده مناسب است زیرا مسئله‌ی سربار را بر عهده گرفته و حل می‌کند. دلیل دومی که تکنولوژی جزئی آهسته و کند حرکت کرد، ذات تولید دنیای جایگذاری شده بود .

اگر بخواهیم به جمع بندی کلی برسیم می توانیم *RTSC* را این گونه توصیف کنیم :

RTSC یک برنامه طراحی شده است تا گسترش‌های اجزا-اساس را به سمت برنامه‌نویسهای زبان سی جایگذاری شده[1]، بیاورد. مزیت اصلی *RTSC* این است که دسترسی به محتوای هدف را همگون می‌سازد و اینکه محتوای هدف مشمول برنامه شود را ساده‌تر می‌کند .

RTSC شامل قوانین و خطوط راهنمایی می‌شود که بسته‌ها (بسته های کدک مثلاً) برای اینکه در تطابق با *RTSC* قرار بگیرند، باید آنها را رعایت کنند. همانند *XDAIS* مصرف کننده بسته‌ها لازم نیست در مورد قوانین ویژه زیاد نگران باشد زیرا بیشتر مسئولیت بر عهده تولید کننده است .

اجزای اصلی که مورد نیاز بسته بندی *RTSC* هستند عبارتند از : تحویل[2] ، قابلیت پیکربندی[3] و سرهم کردن[4].

در زیر مثال هایی از مهمترین قوانین آورده شده است :

نام گذاری بسته :

- هر بسته باید اسم منحصر به فردی داشته باشد که شامل حروف کوچک است برای مثال *ti.sdo.codecs.mpeg4dec*
  پیشنهاد می‌شود آغاز هر بسته با نام شرکت شما باشد .
- هر بسته باید در ساختار شاخه‌ای قرار گیرد که با نام بسته تطابق داشته باشد. مانند : */alan/workdir/dummies book/ ti/ sdo/codecs/mpeg4 dec*

---

[1] *Embedded C*
[2] *delivery*
[3] *configurability*
[4] *assembly*

فایل های بسته

تمام فایل های بسته باید در شاخه‌ی اصلی یا زیرشاخه‌ی بسته قرار داشته باشند.مانند: *impeg4dec.h* در شاخه ی *mpeg4dec*

قانون اصلی بسته

تمام مرجع‌های نام فایل در داخل یک بسته باید از *\not/* استفاده کنند بنابراین بسته ها می‌توانند در لینوکس و هم در ویندوز مورد استفاده قرار گیرند .

سازگاری بسته

بسته ها باید مشخص کنند که به کدام بسته ها بستگی دارند و همینطور مشخص کنند کدام نسخه‌ها از آن بسته ها مورد نیاز است .

سندسازی بسته

سندسازی باید در داخل بسته بوده و در فرمت *pdf* باشد. به جز آنهایی در واقع *online release notes* می باشند و باید در فرمت *HTML* باشند .

بسته های منبع[1] :

- بسته های منبع باید ٪۱۰۰ از سیستم عامل میزبان (لینوکس یا ویندوز) مستقل باشند .
- بسته های منبع قابل پیکربندی نباید به بسته های مرجوع داده شده[2] برای بازسازی نیاز داشته باشند .

[1]Source Packages
[2]Referenced Packages

۳-۳-۴ محصولات چارچوب چند رسانه ای [1]

شرکت *TI* مشغول ساختن زیربنای نرم افزاری بوده است که نشان دهد برنامه نویسی با الکتریسیته و تیرهای چوبی برابر است(ینی این زیربنا همان نقشی رو بازی می‌کند که الکتریسیته و تیرهای چوبی در ساختن خانه بازی می‌کنند). عناصر این زیربنا، محصولات چارچوب چند رسانه‌ای یا به طور اختصار *MFP* نامیده می‌شود. اگرچه احتمال دارد بیشتر توسعه دهندگان، این مواد را خودشان تولید کنند اما اصلاح موارد یکسان به طور پیاپی ارزش چندانی ندارد. *TI* این محصولات را ساخته و گسترش داده است تا شما مجبور به اصلاح پی در پی نباشید. محصولات چارچوب چند رسانه‌ای، هنگامی که شما بخواهید برنامه‌های خودتان را که بر اساس *DAVINCI* است، گسترش دهید، زیربنای نرم‌افزاری را فراهم می کند. *MFP* ها به دو گروه مجزا تقسیم می‌شوند :

- اجزای چارچوب[2]

این اجزا ماژول های استانداردی هستند که به طور مخصوص برای کمک به نمونه‌سازی و برای کدک‌های *XDAIS / XDM*طراحی شده‌اند. در واقع این اجزا برای ساختن خانه‌ی نرم افزاری به کار می‌روند که بر برنامه شما اثر می‌گذارد. این اجزای چارچوب به طور رایگان در دسترس خواهند بود .

- موتور کدک[3]

بیشتر شبیه یک خانه‌ی مدل است. این موتور استفاده‌ی سنگین از اجزای چارچوب را به خوبی دیگر اجزای زیربنا، مهیا می‌سازد. هر چند *TI* یک خانه‌ی مدل ساده و کاملی را فراهم می‌کند، اما همچنان به

[1] *Multimedia Framework Products*
[2] *Framework Components= FC*
[3] *Codec Engine= CE*

کاربر بستگی دارد که محل دکوراسیون را انتخاب کند. در دنیای *DAVINCI* این امر به معنی انتخاب کدکهای استاندارد، اضافه کردن الگوریتمهای مشخص و نوشتن برنامه‌های سطح بالا است .

محصولات چارچوب نرم افزاری جایگزین گستره‌ی قبلی چارچوب های نرم افزاری به نام چارچوب های مرجع[1] شده‌اند. هر گاه شما رجوعاتی مانند *RF3* و *RF5* دیدید، این مخفف ها به سطوح مختلف چارچوب های مرجع ارجاع می‌دهند . خیلی از ویژگی های کلیدی چارچوب های مرجع اصلی هم اکنون در *MFP* گنجانده شده‌اند. یک مزیت *MFP* جدی ، این است که اجزای چارچوب منحصر به فرد[2] به طور مجزا قابل دسترس هستند. چیزی که در تولیدات قبلی چارچوب های مرجع لحاظ نشده بود .

۱-۳-۳-۴- انتخاب لیستی از اجزای چاچوب *XDAIS*

استاندارد *XDAIS* شامل یک مجموعه از قوانین، خطوط راهنما و رابطهایی است که تمام نرم‌افزارهای چندرسانه‌ای *expressDSP* (کدک ها) باید به کار ببرند تا مطابق با استاندارد در نظر گرفته شوند. همانطور که قبلاً ذکر شد قوانین *XDAIS* اجباری ولی خطوط راهنما اختیاری است (هر چند به شدت توصیه می شود البته). کدک ها باید خطوط راهنما و قوانین را رعایت کنند تا به سازندگان سیستم اجازه دهند، چه در زمان طراحی و چه زمان اجرا ، تا بر اساس و برحسب منابع ارزشمند روی تراشه تعیین کنند به هر عضو،چه منابعی می‌رسد. مواجه شدن با کدک های همگون شده، به یک سری اجزای نرم‌افزاری پایه نیاز دارد که هر گسترش‌دهنده برنامه‌ای به منظور نمونه سازی، جستجو و تحقیق و اجرای مناسب کدک در محیط های *DAVINCI* باید از آن استفاده کند. *TI* چندین جزء از پیش ساخته شده کلیدی فراهم کرده است که در زیر به توضیح آنها می پردازیم:

---

[1] *Reference Framework*
[2] *Individual Framework Components*

- *DSKT2*

این جزء چارچوب رابط *XDAIS IALG* را به کار می‌برد. رابط*IALG* مسئول تعیین کردن این موضوع است که چه منابع حافظه‌ای از سیستم، توسط کدک مورد نیاز است. برای اطمینان از عملکرد مناسب، کدک ها اجازه به تصرف منابع نخواهند داشت اما باید با واسط *IALG*در تطابق باشند. *DSKT2*، نیازمندی های منابع را برقرار می‌کند، حافظه را بطوری که سیستم مناسب به نظر برسد، تخصیص می‌دهد و سپس کدک را نمونه سازی می‌کند بنابراین کدک به طور مناسب عمل می‌نماید .

- *DMAN3*

این جز از رابط *XDAIS IDMA3* برای تعیین و برقرار کردن این موضوع که کدام منابع *DMA* توسط کدام کدک در سیستم مورد نیاز است،استفاده می‌کند. درست همانند کنترل منابع حافظه با *IALG* ، کدک ها اجازه دسترسی آسان به منابع *DMA* را نخواهند داشت . شاید کسی بخواهد بداند چگونه *DSKT2* و *DMAN3* را به دست می آوریم و برای *DSKT*، *DMAN* و *DMAN2* چه اتفاقی افتاد؟ آیا این موارد خارج شده‌اند؟ بیشتر هسته‌های پیشرفته *DSP* مانند *C64x+* که در وسایل *OMAP* و *DAVINCI* یافت می‌شوند، ویژگی‌های پیشرفته *DMA* که *QDMA* و *EDMA 3.0* نامیده می‌شوند را به طور بارز نشان می‌دهند. به منظور استفاده از این قابلیتها، *DMAN* و *DMAN2* باید کنار گذاشته شوند و *DMAN3* که قابلیت بیشتری دارد به منظور استفاده کامل از ویژگی ها و مزیت های*C64x+* آورده شود.

- *ACPY3*

این جزء به صورت مخصوص برای استفاده کدک ها به منظور کپی کردن مقدار زیادی اطلاعات از یک حافظه به حافظه دیگر طراحی شده است. *ACPY3* از کپی‌های سریع حافظه *DMA*-اساس[1] استفاده می‌کند. این منابع *DMA* در واسط *DMAN3* قرار دارند. *ACPY3* برای *C64+DSP* به شدت بهینه

[1] *DMA-based Memory Copies*

شده است و اگر بخواهید بدانید چه اتفاقی برای *ACPY* و *ACPY2* افتاده است دقیقاً همون اتفاقی که برای *DMAN* و *DMAN2* رخ داده و در بالا توضیح داده شد.

۲-۳-۳-۴- آغاز کردن موتور کدک

خانه‌ای را در نظر بگیرید که از قبل ساخته شده است. خیلی از موارد را می‌توان در مورد نقل مکان به این خانه ذکر کرد. داشتن خانه ای که با علایق مشخصی ساخته شده باشد، مزیت هایی دارد ولی مطمئناً زمان و هزینه جزء آنها حساب نمی‌شود. با کمی شانس شما می‌توانید خانه ای را پیدا کنید که برای نیازهای شما مناسب است و سپس هنگامی که به داخل خانه منتقل شدید، می‌توانید کارهای نهایی را انجام دهید . *TI* یک چارچوب نرم افزاری چند رسانه ای را که از قبل ساخته شده و آماده ی استفاده است به نام موتور کدک را پیشنهاد می‌دهد و موتور کدک به طور ایده آل برای اجرا روی *DAVINCI* مناسب است .

هر چند که بیشتر وسایل *DAVINCI* و *OMAP* شامل دو پردازنده*DSP* و *ARM* هستند اما استثناهایی وجود دارد که ممکن است فقط یک هسته *DSP* یا *ARM* وجود داشته باشد . خبر خوب این است که برنامه ی شما اگر بر اساس موتور کدک باشد به سادگی در بین بسته های سیلیکونی متفاوت قابل انتقال خواهد بود .

موتور کدک بر روی *DSP/ BIOS* اجرا شونده روی *DSP* ساخته می‌شود. از اجزای چارچوب *XDAIS*[1] که به ترتیب نمونه‌سازی کرده و کدک های مطابق *XDAIS/ XDM* را اجرا می‌کند ، استفاده می‌کند . برای ارتباطات داخل پردازنده‌ای[۲] از *DSP/ BIOS link* استفاده می‌کند و در نهایت مفاهیم اجرای نرم افزار زمان حقیقی *(RTSC)* را مورد استفاده قرار می‌دهد. موتور کدک هم اکنون روی سیستم

[1] *XDAIS Framework Components*
[2] *Inter – processor*

عامل های لینوکس *Win CE* و *Green Hills Integrity* قابل اجراست و بقیه‌ی سیستم عامل ها در آینده اضافه خواهند شد .

الف) *VISA*

VISA در واقع بیان کننده‌ی *Video* ، *Imaging* ،*Audio*و *Speech* است. این *API*ها یا رابط برنامه‌نویسی کاربردی[1] به گسترش دهندگان برنامه سطح بالا امکان نمونه‌سازی و اجرای کدکهای مختلف را می‌دهد بدون اینکه نیازی به دانستن اطلاعات در مورد جزئیات رابط کدک و این که کدک به چه طبقه‌ای تعلق دارد، احساس شود. زیربنای موتور کدک ، به کدک امکان می دهد تا به راحتی با کدک دیگری از همان طبقه با اثری خیلی کوچک برنامه جایگزین شود .

هر کدک *VISA* یا رمزگشاست یا رمرگذار. *API* های جداگانه‌ای برای رمزگشایی و رمزگذاری وجود دارند که به هشت دسته تقسیم می شوند :

۱- *VIDENC*
۲- *VIDDEC*
۳- *IMG ENC*
۴- *IMGDEC*
۵- *SPHENC*
۶- *SPHDEC*
۷- *AUD DEC*
۸- *AUD ENC*

[1] *Application Programming Interface*

مانند دیگر *API* ها، برخی از این موارد هم اکنون به روز شده‌اند مانند *VIDENC2*. در یک طراحی جدید همیشه از شماره نسخه بالاتر استفاده کنید. در داخل هر طبقه ای که ذکر شد، چهار *API*دیگر وجود دارد :[1]

۱- *XXX- Create ()*
۲- *XXX-Control()*
۳- *XXX-Process()*
۴- *XXX-Delete()*

این چهار رابط یک روش استاندارد برای هر طبقه کدک فراهم می‌کند تا مورد استفاده قرار گیرند. ابتدا سیستم، کدک را با آماده کردن و بار کردن برنامه و حافظه های دیتا در محل های مناسب ایجاد کرده و کدک را برای اجرا آماده می‌کند. سپس کدک یک یا چند بلوک اطلاعات را پردازش می‌کند و نتایج را به مشتری[2] ارسال می‌کند. معمولاً بعد از این مرحله، پارامترهای کنترل برای اصلاح و تغییر عملیات کدک می‌توانند به کار روند. در انتها، هنگامی که به کدک نیازی نبود، می‌توان کدک را پاک کرد و منابع پردازنده و حافظه را برای پردازش‌های دیگر آزاد گذاشت .

ب)خلاصه کردن عملکرد توسط موتور کدک[3] :

اتومبیل های هایبرید را در نظر بگیرید که شامل دو نوع موتور هستند : یکی از الکتریسیته تغذیه می‌کند و دیگری از بنزین. حال تصور کنید که اتومبیل هایبرید شامل ۲ شتاب دهنده باشد (دو پدال گاز) و این که هر پدال را چقدر باید فشار داد تا اتومبیل شتاب گیرد، به عهده‌ی راننده باشد. این موضوع گیج کننده است. البته اتوموبیل های هیبرید حقیقی فقط یک پدال گاز دارند و سیستم‌های

---

[1] *API*= یک پروتکل است که به قصد رابط بودن توسط اجزای نرم افزار برای ارتباط با یکدیگر به کار می رود

[2] *Client*
[3] *Abstracting Performance with Codec Engine*

تعبیه شده در اتوموبیل از مفهوم « خلاصه سازی» برای نشان دادن این که چگونه و چه زمان هر دو موتور یا هر کدام موتورها به کار می‌روند، استفاده می‌کنند .

در دنیای پردازنده‌های *DAVINCI* و *OMAP*، نه تنها دو هسته‌ی ناهمگن وجود دارند بلکه دو قسمت برای موتور کدک در نظر گرفته شده است. یک قسمت در *ARM* قرار می‌گیرد و دیگری در *DSP*. این امر سبب می‌شود که خلاصه سازی عملکرد نهایی حاصل گردد. یک گسترش‌دهنده برنامه‌های سطح بالا، لینوکس را بر روی پردازنده *ARM* اجرا می کند و می‌تواند کدک های بسیار قدرتمندی را نمونه سازی و اجرا کند بدون اینکه در مورد چگونگی و محل اجرای واقعی کدک نگران باشد. در یک سیستم *DSP/ARM* که کدک به سیکل‌های پردازنده ای[1] زیادی نیاز دارد، پردازش کدک بر روی *DSP* و شتاب دهنده های متصل شده انجام خواهد شد. اگر اجزای یک کدک، به تعداد سیکل پردازنده‌ای کمی نیاز داشته باشد، در این صورت این امکان وجود دارد که کدک فقط روی هسته‌ی *ARM* اجرا شود.

مدت زمانی که طول می‌کشد گسترش‌دهنده برنامه بتواند یک جریان پایدار[2] از اطلاعات فراهم نماید تا کدک بتواند روی آن عمل کند، به اندازه ی مدت زمانی است که طول می‌کشد تا *MIPS* پردازشی کافی برای اجرای کدک در دسترس باشد. بعد از تأمین شدن این مدت زمانی، موتور کدک می تواند تقریباً مراقب هر چیز دیگری باشد. بنابراین گسترش‌دهنده برنامه می‌تواند به جای نگرانی در مورد اینکه کدام پردازنده آدرس‌دهی می‌شود و موارد جزئی مانند ارتباطات داخل پروسسوری، تغییر و ترجمه آدرس و مدیریت حافظه نهان، بر روی خود برنامه تمرکز کند.

[1] Processor cycle
[2] Steady Stream

ج)مرور فرآیند موتور کدک

در زیر فرآیندها و مراحلی که لازم است تا خلاصه سازی ممکن در موتور کدک رخ دهد، بیان می‌شود:

- مرحله ی ۱-الگوریتم های تایید شده و سبک[۱]

الگوریتم‌های کدک یا بطور انبوه تولید می‌شوند یا بطور خانگی[۲] . اما به هر روشی که تولید شوند نیاز دارند تا *XDM* و *XDAIS* را برای هر موردی به کار ببرند تا به طور مناسب مجتمع شوند. اگر الگوریتم های شما به طور مناسب در سیستم های مجتمع کار نکنند شما نمی‌توانید به مرحله ی بعد بروید .

- مرحله ی ۲ – سرور موتور کدک [۳]

به منظور اینکه موتور کدک، کدک هایی که بر روی هسته‌ی دیگر اجرا می‌شوند را بتواند حمایت کند، باید سروری وجود داشته باشد. این سرور، کدک هسته را به همراه قطعات زیربنایی دیگر (*DSP/ BIOS*، اجزای چارچوب *DSP/ BIOS Link*و ...) ترکیب می‌کند و در نهایت یک فایل قابل اجرا تولید می‌کند که از هسته‌ی دیگر قابل فراخوانی است. این امکان نیز وجود دارد که بیشتر از یک کدک را از طریق راه های مختلف با هم ترکیب کرد. به هر حال، حیاتی و مهم است که کسی نیازمندی‌های منابع سیستم (حافظه، *DMA*، *MIPS* و...) را ارزیابی کند تا مطمئن شود که ترکیب کدک می‌تواند وجود داشته باشد (کدک ها می‌توانند با هم وجود داشته باشند و ترکیب شوند) و همانطور که مورد نیاز برنامه است، اجرا شود. همچنین این امکان نیز وجود دارد چندین سرور متفاوت ساخته شود .

---

[1] *Cool Algorithms*
[2] *Home Made*
[3] *Codec Engine Server*

- مرحله ی ۳- مجتمع سازی موتور کدک[۱]

سازنده یا مجتمع کننده‌ی سیستم می‌تواند یک یا بیشتر پیکربندی موتور[۲] تولید کند. این پیکربندی‌ها شامل نام موتورها، کدک یا کدک هایی که در داخل آن موتورها وجود دارد، محلی که کدک ها اجرا می‌شوند (معمولاً *DSP* ولی می‌تواند استثنا وجود داشته باشد) و نام سرور اگر موتور شامل کدک های دور[۳] (یعنی کدک هایی که در هسته‌ی دیگر اجرا می‌شوند)، می‌باشند .

- مرحله ۴- برنامه کاربردی موتور کدک [۴]

برنامه از *API* های موتور کدک که توسط خود موتور کدک فراهم شده است، استفاده می‌کند تا مثال های موتور را ایجاد کند و پاک کند، کدکها را ایجاد کرده و پاک کند و با آنها فعل و انفعال داشته باشد، بافرهای اطلاعات برای عمل کردن کدک ها تعیین شود و... . موتور کدک خودش عملیات *I/O* را انجام نمی‌دهد بلکه آن را به برنامه سطح بالا واگذار می‌کند تا به کار ببرد.

اگر بخواهیم به طور خلاصه موتور کدک را توصیف کنیم :

موتور کدک یک مجموعه از *API* ها است که شما برای نمونه‌سازی و اجرای الگوریتم های *XDAIS* از آن استفاده می‌کنید. یک رابط*VISA* به خوبی فراهم شده است تا با الگوریتم های *XDAIS* مطابق با *XDM* فعل و انفعال داشته باشد. *API* برای موقعیت های زیر یکسان است :

- الگوریتم ممکن است به طور محلی (روی *GPP*)اجرا شود یا به طور ریموت (بر روی *DSP*)
- سیستم ممکن است *GPP+ DSP* ،فقط *DSP* و یا فقط *GPP* باشد .

---

[1] *Codec Engine Integration*
[2] *Engine Configurations*
[3] *Remote Codec*
[4] *Codec Engine Application*

- تمام *GPP* ها و *DSP* های حمایت شده، *API* یکسان دارند .
- تمام سیستم عامل‌های حمایت شده *API* یکسان دارند به عنوان مثال:لینوکس، *WinCEVxworks, PrOs*

موتور کدک طراحی شده است تا برخی از مشکلاتی که به طور رایج با گسترش برنامه‌های *SOC* همراه می‌شود، را حل کند. مهمترین مشکلات عبارتند از :

- دیباگ کردن در پردازنده ی ناهمگن به علت وجود چندین دیباگر دردسرساز است .
- کاربردهای متفاوت از یک الگوریتم یکسان مانند *API*، *MP3* های متفاوت دارد .
- مسائل قابل انتقال بودن با وجود دو پردازنده پیچیده خواهد شد .

موتور کدک این مشکلات را با فراهم کردن یک ساختار نرم افزاری و رابط‌هایی برای اجرا حل می‌کند.

در واقع موتور کدک :

- به آسانی قابل استفاده است

گسترش دهندگان برنامه تعیین می‌کنند چه الگوریتمی باید اجرا شود ، نه چگونه و کجا

- قابل پیکربندی است

الگوریتمهای جدید توسط هر شخصی می‌تواند اضافه شود. با استفاده از ابزار و تکنیک های استاندارد.

- قابل انتقال است

API ها نسبت به هدف، بستر و حتی کدک مستقل هستند .

۴-۳-۴ *DSP/BIOS Link*

هنگامی که یک سیستم نرم‌افزاری را روی یک *SOC* چند هسته‌ای می‌گذارید، باید مسئله‌ی چگونگی ارتباط پردازنده‌ها با یکدیگر را در نظر بگیرید. در قدیم، توسعه‌دهندگان طرح‌های ارتباط داخل پروسسوری[1] را برای حل این مسئله ایجاد کردند .

در حدود ۱۰ سال پیش شرکت *TI* یک ارتباط داخل پردازنده‌ای را طراحی کرد که *DSP/ BIOS Link* نامیده می‌شود. گاهی اوقات به آن *DSP Link*، *BIOS Link* یا حتی *Link* می‌گویند.

۱-۴-۳-۴- چرا از *DSP/ BIOS Link* استفاده می کنیم ؟

عملکرد *DSP/ BIOS Link* حیاتی است اما هیچ کس به آن تا هنگامی که همه چیز درست کار می‌کند فکر نمی‌کند. شما مزیت زیادی را بدست نمی‌آورید اگر زمان خود را برای طراحی یک ارتباط داخل پروسسور صرف کنید (هدر دهید) در حالیکه *DSP/ BIOS Link* در دسترس بوده و آماده استفاده است.

مزیت های اولیه استفاده از *DSP/ BIOS Link* عبارت است از :

- برای برنامه‌ها، مانند رابط *API*عمومی عمل می‌کند بنابراین امکان قابل انتقال بودن آن برنامه‌ها را فراهم می‌کند .
- لایه‌ی خلاصه سازی سخت افزاری بین برنامه‌ها و سخت‌افزار زیربنایی را فراهم می‌کند. همچنین قابلیت انتقال بین بسترهای سخت افزاری را فراهم می‌نماید.
- همانند *DSP/ BIOS* مقیاس پذیر است بنابراین تنها ماژول‌های مورد نیاز برنامه با آن در تماس خواهد بود .

[1] *Inter processor communication*

۲-۴-۳-۴- *DSP/ BIOS Link* چه چیزی را پیشنهاد می دهد ؟

DSP/ BIOS Link سرویس های مهمی را ارائه می‌دهد که می‌توان آنها را به سه دسته تقسیم کرد:

- کنترل پایه ای پردازنده[۱]

این سرویس به *ARM* امکان می دهد تا خود را به *DSP* متصل کند،*DSP* را با کد بار کند،*DSP* را آغاز و متوقف کند و خود را از *DSP* جدا کند.

- پروتکل های ارتباط داخل پردازنده‌ای[۲]

این سرویس انتقال اطلاعات بین هسته ها را فراهم می‌کند .

- بلوکهای بنایی ارتباط داخل پردازنده‌ای[۳]

کنترل پایه‌ای پردازنده و پروتکل های ارتباط داخلی پردازنده‌ای بر روی بلوک های بنایی ارتباط داخل پردازنده‌ای ساخته می‌شوند. بلوک های بنایی به طور جداگانه برای نویسندگان چارچوب، برای گسترش پروتکل های خود در دسترس است .

در زیر لیست جزئیات بیشتری از سرویس های فراهم شده توسط *DSP/ BIOS Link* آورده شده است :

- ***PROC* :کنترل پایه‌ی پردازنده**
  - *GPP* به *DSP*متصل می‌شود .
  - *GPP*، *DSP* را با فایل‌های قابل اجرای *DSP* بار می‌کند .
  - *GPP* باعث می‌شود *DSP* شروع به کار کند .

[1] *Basic Processor Control*
[2] *Inter- processor Communication Protocols*
[3] *Inter- processor Communication Building Blocks*

- *GPP* باعث می‌شود *DSP* متوقف شود .
- *GPP* از *DSP*جدا می‌شود .

- **پروتکل های *IPC***

- *MSGQ* : هدف پیغام
- *CHNL* : جریان اطلاعات بر اساس مدل اصلاح مسئله[1]
- *RingIO* : جریان اطلاعات بر اساس یک بافر حلقوی

- **بلوک های بنایی ارتباط داخل پردازنده‌ای**

- *POOL* : مدیریت حافظه
- *NOTIFY* : خلاصه سازی وقفه و کاهش مولتی پلکس کردن برای اعلان رخدادها[2]
- *MPCS* : انتخاب حیاتی چند پردازنده برای دسترسی دو به دو ناسازگار به اشیا به اشتراک گذاشته شده[3]
- *Proc_read / Proc _write* : خواندن یا نوشتن در حافظه ی *DSP*.

[1] *issue- reclaim*
[2] *de- Multiplexing for event notification*
[3] *Multi- processor critical section for mutually exclusive access to shared objects*

۵-۳-۴- انتخاب کدک چند رسانه ای مناسب

یکی از موارد خیلی مفید در مورد دستگاه های *DAVINCI*، تعداد نامحدود ترکیب کدک‌ها و الگوریتمها است که می‌توانند به طور پی در پی یا همزمان اجرا شوند. در ادامه منابع مختلف کدک‌ها را توضیح می دهیم.

۱-۵-۳-۴- گروه‌های کدک

چهار طبقه مهم کدک ها عبارتند از: ویدیو ، صوت ، تصویر و گفتار. شما ممکن است رجوعاتی به *VISA* ببینید (یک *API* کاربردی که این چهار طبقه ی کدک را آدرس دهی می‌کند) . توجه داشته باشید که ممکن است طبقات جدیدتری مانند تحلیل های ویدیویی[1] نیز وجود داشته باشد .

در زیر سیستمی از کدک های محبوب که برای اجرا روی پروسسورهای *DAVINCI* در دسترس است دیده می‌شود :

- *Video: MPEG2, MPEG4, H.263, H.264, WMV9, DIVX*
- *Image: JPEG*
- *Speech:G.711, G.723, G.726, G.729*
- *Audio: MP3, WMA8, WMA9, AAC, MPEG1 L2*

۶-۳-۴- کیت های گسترش نرم افزار ویدیوی دیجیتال[2]

ماژول‌های *EVM* به شما کمک می‌کنند تا سخت افزار را ارزیابی کنید. مشابه نرم‌افزاری آن، کیت گسترش نرم افزار ویدیوی دیجیتال (*DVSDK*) نامیده می‌شود. *DVSDK* یک مجموعه‌ی کامل

[1] *Video Analytics*
[2] *Digital video software development kits= DVSDK*

نرم‌افزاری است که به شما امکان می‌دهد به سرعت برنامه‌ها را روی *EVM* اجرا کنید. *DVSDK* معمولاً شامل اجزای زیر است :

- بسته حمایت لینوکس *[Linux support package = LSP]*
- کیت گسترش دهنده *XDAIS [XDAIS developers kit]*
- کدک‌های ارزیابی *[Evaluation Codecs]*
- موتور کدک *[codec engine]*
- کاربردهای لینوکس *[Linux utilities]*
- کاربردها و برنامه‌ی دموی رمزگشا *[Decode Demo application]*
- مثال رمزگذاری فایل – اساس *[file- based "Encode" example]*
- اطلاعات *AIV[AIV data]*
- نیمکت تست ویدیوی دیجیتال *[Digital video test bench= DVTB]*
- *DSP/BIOS*
- *DSP/ BIOS Link*
- سندسازی

۷-۳-۴- وسایلی که برای پروسسور داوینچی لازم است

TI کدک‌های چندرسانه ای را فراهم می‌کند و یک زیربنای نرم افزار کامل را روی *DSP* آماده کرده است. این موضوع بدین معنی است که شما مجبور نیستید *DSP* روی *DAVINCI* را به طور مستقیم برنامه‌ریزی یا دیباگ کنید . به هر حال برنامه‌های ویژه‌ای هستند و گسترش‌دهندگانی هستند که می‌خواهند یا نیاز دارند در گسترش *DSP* عمیقتر عمل کنند. یک مجموعه از ابزار دیباگ و توسعه‌ی

DSP وجود دارد. یک نمونه از آنها *Code Composer Studio* است که یک محیط گسترش مجتمع[1] می باشدکه توسط شرکت *TI* برای گسترش *DSP* فراهم شده است .

از آنجایی که گسترش و دیباگ کردن *DSP* نسبت به پروسسور *ARM* متفاوت است، *CCS* بطور مخصوص برای *DSP* بهینه شده است.

ویژگی های پایه *CCS* عبارتند از :

- *IDE* : ویرایشگر مجتمع ، مدیریت پروژه
- *Debugger* : *DSP* را دیباگ می‌کنند و دیده شدن اطلاعات و حافظه ی نهان را فراهم می‌کند.
- *Real- time Debug* : دسترسی حافظه را امکان می‌سازد و هنگامی که وقفه ها متوقف شده‌اند را کنترل می‌کند.
- *Simulation* : شامل شبیه سازی دقیق و پوشش کد .
- *Code Generation Tools* : عملکرد صنعتی و بهینه سازی برنامه را پیشنهاد می‌دهد .

۴-۴- نتیجه‌گیری

اساس کار پردازنده‌های *DAVINCI*، استفاده از دو هسته غیرهمگون می‌باشند. یک هسته وظایف عمومی مانند انتقال اطلاعات و اجرای رابط گرافیکی را انجام می‌دهد، به همین سبب به آن پردازنده هدف عمومی(یا هسته همه کاره) می‌گویند و هسته دیگر وظایفی مانند فشرده کردن جریان‌های ویدیویی و

[1] *IDE= Integrated Development Environment*

گفتار و صوت یا از حالت فشرده خارج کردن آنها را انجام می‌دهد. آن هسته را پردازنده سیگنال دیجیتال می‌گویند.

پردازنده‌های *DAVINCI*، برای سیستم‌های ویدیوی دیجیتال بهینه شده‌اند و برای بسیاری از برنامه‌های مربوط به ویدیو، تصویر و بینایی ماشین مناسب می‌باشند. این پردازنده‌ها علاوه بر دو هسته توضیح داده شده در فوق، شامل شتاب دهنده‌های ویدیویی و اجزای جانبی کافی بوده که با توجه به عملکرد بالا، حافظه کافی برای ذخیره سازی و قیمت مناسب، آن را به انتخابی ایده آل تبدیل کرده است.

پردازنده *TMS320DM6446*، دارای تعداد زیادی منابع بوده و قدرت محاسباتی بالایی دارد.این پردازنده برای کاربردهای ویدیویی دیجیتال بسیار مناسب بوده و دارای مشخصات زیر است:

- عملکرد بالا
- زیر سیستم ویدیویی اختصاص داده شده
- ظرفیت حافظه بالا
- تعداد زیاد اجزای جانبی
- مصرف کم و حالت‌های چندگانه مدیریت توان

علاوه بر موارد ذکر شده، شرکت *TI* با ارائه استانداردهای مختلف، این امکان را برای گسترش‌دهندگان فراهم کرده است تا برنامه‌های مورد علاقه‌ی خود را بر طبق آنها گسترش دهند.

فصل پنجم

نتیجه گیری،پیشنهادات و کارهای آینده

در این پایان نامه، روشی برخط و سریع برای آشکارسازی و تعیین نوع تغییر نما پیشنهاد شد. روش پیشنهادی از ویژگیهایی استفاده کرد که به موازات ورود اطلاعات فریمها قابل محاسبه بوده و نیازی به دانستن اطلاعات فریمهای بعدی ندارند. همچنین، حجم محاسباتی روش پیشنهادی تا حد قابل ملاحظه‌ای کم است طوری که مناسب پیاده‌سازی بلادرنگ سخت‌افزاری در بسترهایی مانند پردازشگرهای سیگنال و مدارات منطقی برنامه‌پذیر می‌باشد. همچنین روش پیشنهادی روی برد آموزشی مبتنی بر *TMS320C5505* اجرا گردید. کارایی روش پیشنهادی با کمک چهار معیار دقت، یادآوری، معیار هارمونیکی و متوسط زمان اجرای برنامه به ازاء هر فریم ارزیابی و با یک روش مبتنی بر هیستوگرام مقایسه شد.

علاوه بر اقدام به پیاده‌سازی این روش در بستر پردازشگر *TMS320DM6446* از خانواده *DAVINCI* گردید. اساس کار پردازنده‌های *DAVINCI* ، استفاده از دو هسته غیرهمگون می‌باشند. یک هسته وظایف عمومی مانند انتقال اطلاعات و اجرای رابط گرافیکی را انجام می‌دهد، به همین سبب به آن پردازنده هدف عمومی(یا هسته همه کاره) می‌گویند و هسته دیگر وظایفی مانند فشرده کردن جریان‌های ویدیویی و گفتار و صوت یا از حالت فشرده خارج کردن آنها را انجام می‌دهد. آن هسته را پردازنده سیگنال دیجیتال می‌گویند.

پردازنده‌های *DAVINCI*، برای سیستم‌های ویدیوی دیجیتال بهینه شده‌اند و برای بسیاری از برنامه‌های مربوط به ویدیو، تصویر و بینایی ماشین مناسب می باشند. این پردازنده‌ها علاوه بر دو هسته توضیح داده شده در فوق، شامل شتاب دهنده‌های ویدیویی و اجزای جانبی کافی بوده که با توجه به عملکرد بالا، حافظه کافی برای ذخیره سازی و قیمت مناسب، آن را به انتخابی ایده آل تبدیل کرده است.

به دلیل کمبود وقت و عدم وجود منابع مفید و افراد مجرب، نتیجه مورد نظر در مورد پردازنده *TMS320DM6446* حاصل نشد اما تحقیقات صورت گرفته در این زمینه ارائه شده است.

روش پیشنهادی دارای دقت بالایی است اما یکی از موارد مهمی که روش پیشنهادی دچار اشتباه در تشخیص می‌شود، زمانی است که در دو فریم متوالی و متعلق به یک نمای مشترک، تغییر قابل توجهی در محتوای فریمها (از جهت رنگ و شکل اشیاء) رخ دهد.

بنابراین یکی از کارهای پیشنهادی، اصلاح روش برای حل این مشکل می‌باشد. همچنین، مراحل پیاده‌سازی روش پیشنهادی در بستر پردازشگر سیگنال *TMS320DM6446*، کامل خواهد شد.

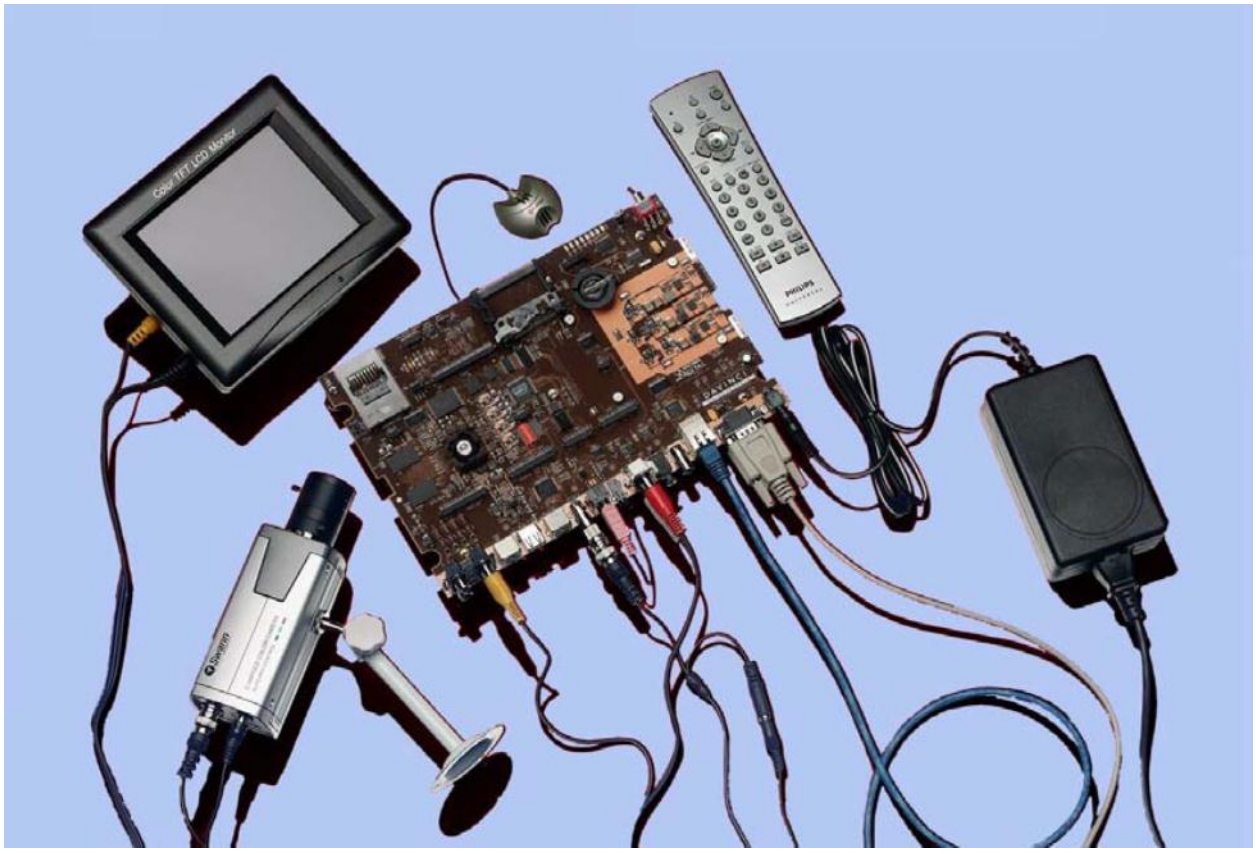
از دیگر پیشنهادات آینده، راه‌اندازی دوربین مربوط به *DM6446 EVM* ، ضبط فیلم‌های ویدیویی به کمک آن و اعمال روش پیشنهادی بر ویدیوهای ضبط شده خواهد بود.

همچنین بر آنیم تا مدت زمان پایگاه‌های دنباله‌های ویدیویی را افزایش دهیم.

پیوست الف

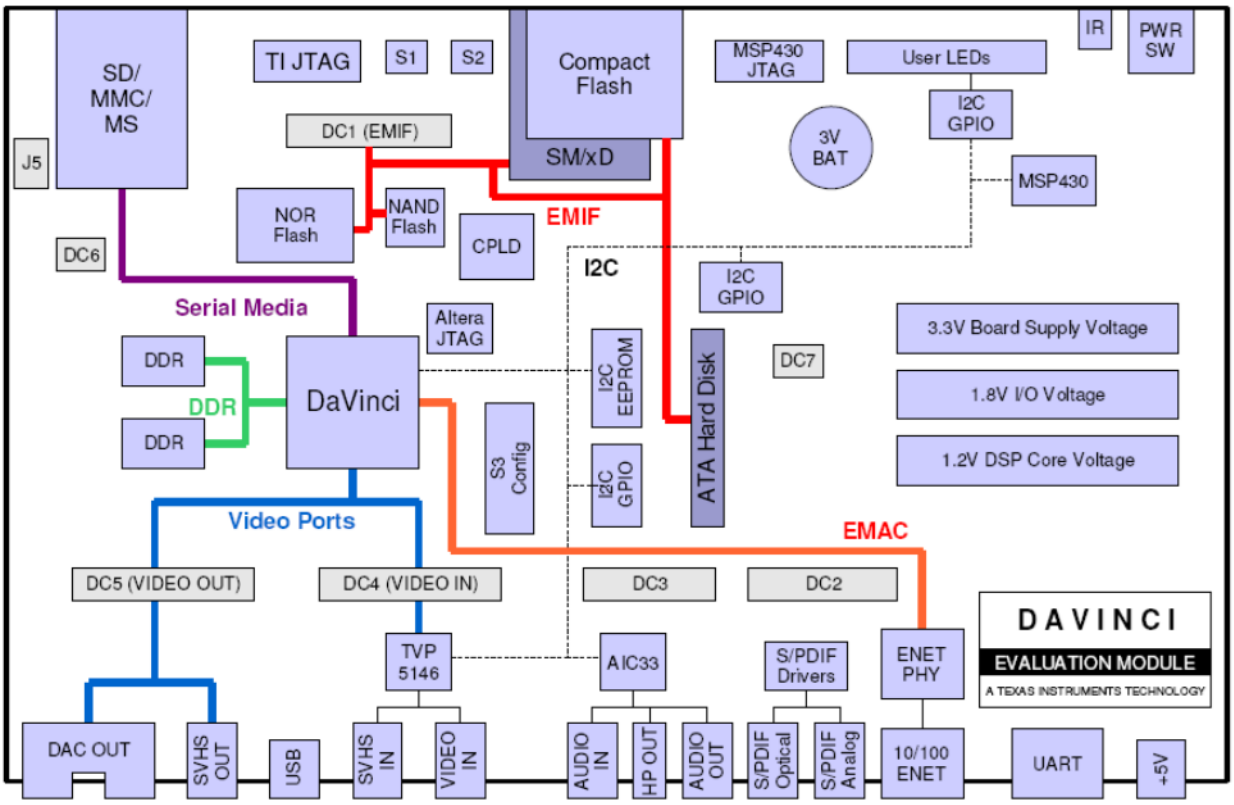
نصب سخت افزاری *DM6446*

در شکل زیر *DM6446 EVM* را همراه با تمام تجهیزات مشاهده می‌کنید. در ادامه نحوه‌ی اتصال تجهیزات به برد را توضیح می‌دهیم.



شکل ۱- *DM6446 EVM* به همراه تمامی تجهیزات

*اگر از ویدیو های نوع *PAL* استفاده می‌کنید، سوئیچ ۱۰ از بانک سوئیچ *S3* را روی وضعیت *ON* قرار داده و اگر از ویدیوهای نوع *NTSC* استفاده می‌کنید، این سوئیچ را روی وضعیت *OFF* قرار دهید. برای اطلاع از مکان بانک سوئیچ *S3*، به شکل زیر رجوع کنید.

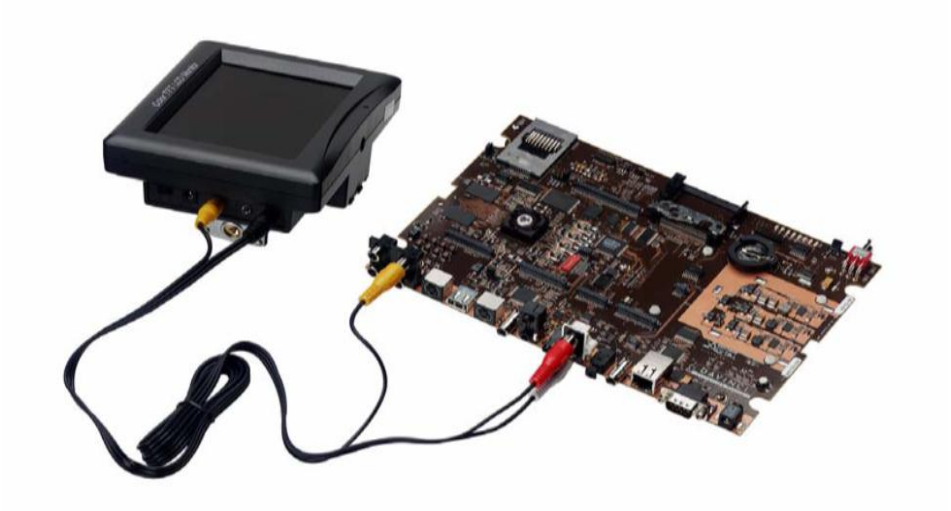


شکل۲- نمودار بلوکی *DM6446 EVM* و مکان سوئیچ *S3* برروی آن

*قبل از اتصال تجهیزات به برق، مطمئن شوید که تمام کابل های ارتباطی وصل هستند.

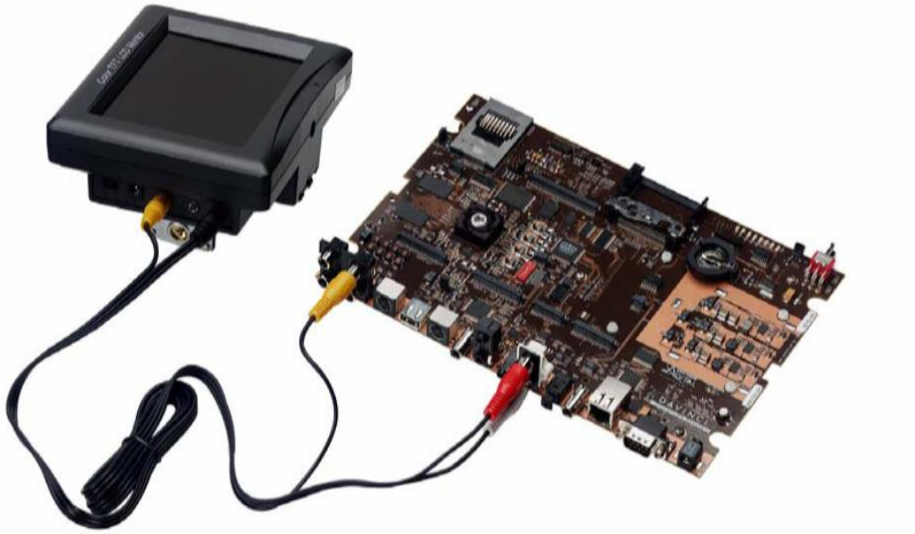
الف-*LCD*

در بسته ی حاوی *LCD*، کابلی موجود است که یک طرف آن ۳ سر بوده و طرف دیگر آن ۲ سر است. در هر دو طرف، یه فیش به رنگ زرد موجود است. فیش زرد رنگ موجود در طرف ۳ سر را به خروجی ویدیو بر روی برد و فیش زرد رنگ موجود در طرف ۲ سر را به قسمت *Video Input* بر روی *LCD*، همانند شکل زیر، وصل کنید.



شکل ۳- نحوه اتصال *LCD* به برد

در طرف ۳سر، دو فیش دیگر به رنگ های سفید و قرمز موجود است که باید به خروجی *Audio* برروی برد متصل گردند. در طرف ۲سر، فیشی مشکی رنگ موجود است که باید به *R/L Audio Input* بر روی *LCD* وصل شود. شکل زیر نحوه ی اتصال را نشان می‌دهد.



شکل۴- ادامه‌ی اتصال *LCD* به برد

LCD دارای یک آداپتور نیز می‌باشد که یه طرف آن به برق و طرف دیگر آن به *LCD* وصل می‌شود. توجه کنید تا مراحل نصب کامل نشده است، آداپتور را به برق وصل نکنید.

ب- دوربین

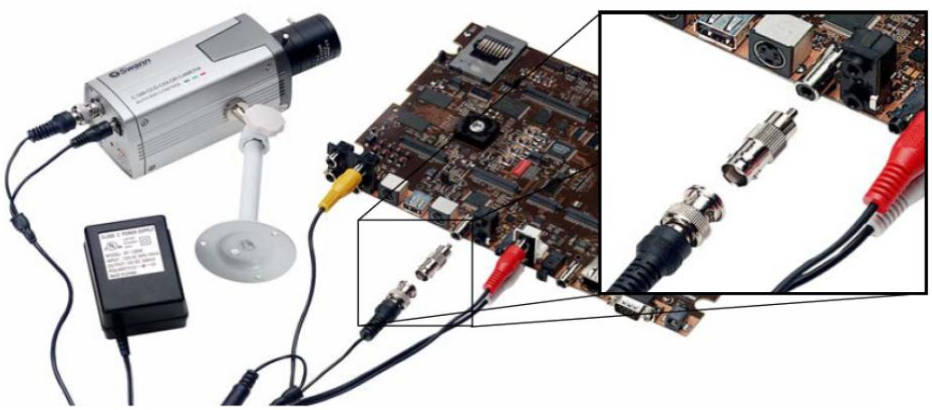
دوربین دارای کابلی است که در هر طرف آن دو سر موجود است. در هر دو طرف فیش *Coax* وجود دارد. اگر بخواهید فیش *Coax* را به برد متصل کنید، باید از مبدل *BNC-to-RCA* استفاده کنید. بعد از اتصال این مبدل به فیش *Coax*، آنرا به قسمت *Video Input* بر روی برد وصل کنید.

*برای اطلاع از این موضوع که کدام فیش *Coax* باید به مبدل و سپس به برد وصل گردد، باید توجه شود که علاوه بر دوفیش *Coax*، دو فیش دیگر نیز موجود است: یک فیش نرگی و یک فیش مادگی.

در طرفی که فیش مادگی وجود دارد، هم فیش مادگی و هم فیش *Coax* باید به دوربین متصل گردند و در طرفی که فیش نرگی وجود دارد، فیش *Coax* باید به مبدل متصل شده و سپس به برد و فیش نرگی به آداپتور متصل شود.

توجه کنید تا مراحل نصب کامل نشده است، آداپتور را به برق وصل نکنید.

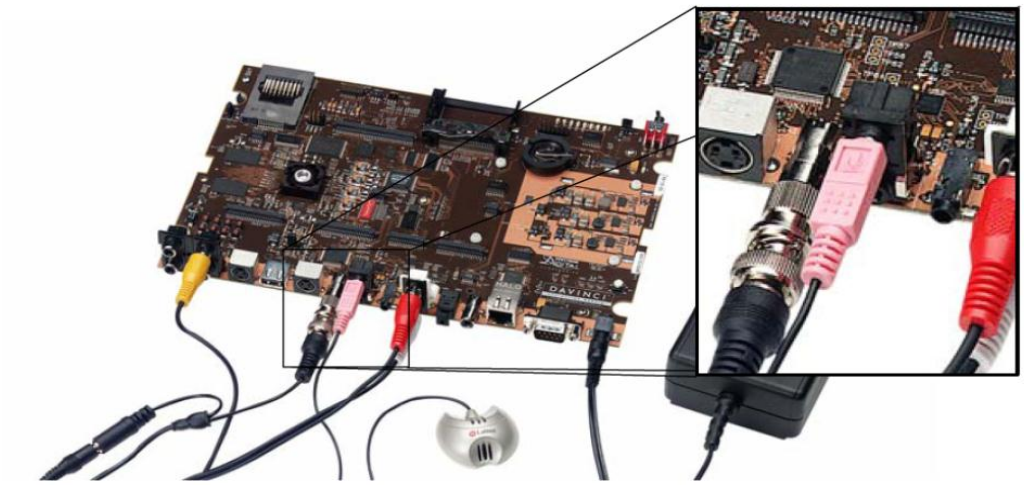
شکل زیر نحوه ی اتصال دوربین را نشان می‌دهد.



شکل ۵- نحوه‌ی اتصال دوربین به برد

ج- میکروفون

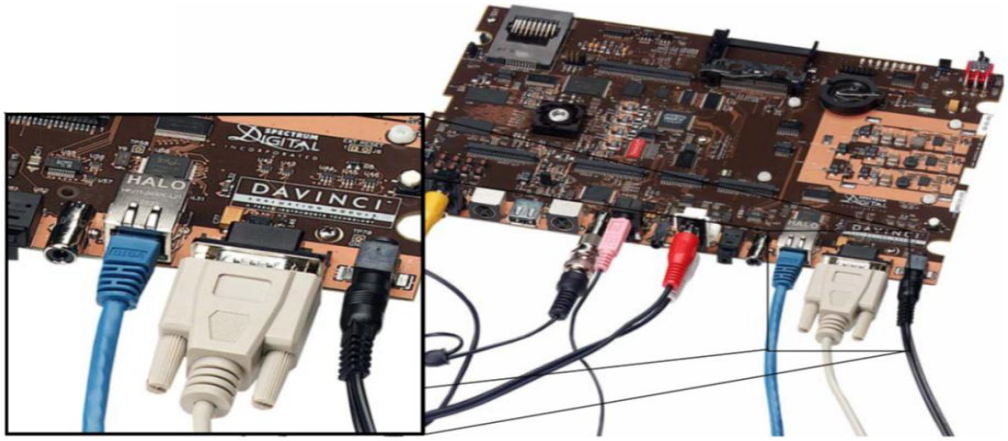
برای اتصال میکروفون کافی است که فیش آن را به مکان مربوطه رو برد وصل کنید. (دقیقا در سمت راست محلی که کابل *Coax* دوربین وصل شده است). شکل زیر نحوه‌ی اتصال را نشان می‌دهد.



شکل۶- اتصال میکروفون به برد

د- کابل *UART* و کابل اتصال به شبکه

اگر میخواهید از کابل *UART* برای یک پنجره خروجی (*Console Window*) استفاده کنید، یه سر کابل *RS-232* را به قسمت *UART port* بر روی برد و سر دیگر آنرا به *Com port* بر روی کامپیوتر وصل کنید. همچنین می‌توانید کابل شبکه را به برد وصل کنید (دقیقا در سمت چپ محل *UART port*). شکل زیر نحوه ی اتصال را نشان می‌دهد.



شکل۷- نحوه اتصال کابل *UART* و کابل اتصال به شبکه

بعد از انجام مراحل فوق،کافی است مراحل زیر انجام شوند:

۱. آداپتور *LCD* را به برق وصل کنید.
۲. آداپتور دوربین را به برق وصل کنید.
۳. یک سر آداپتور برد را به برد و سر دیگر آن را به برق وصل کنید. برای جلوگیری از شک الکتریکی، ابتدا فیش مربوط به برد را متصل کنید و سپس آداپتور را به برق وصل کنید.
۴. *LCD* را روشن کنید.
۵. دوربین را روشن کنید.
۶. برد *EVM* را روشن کنید.

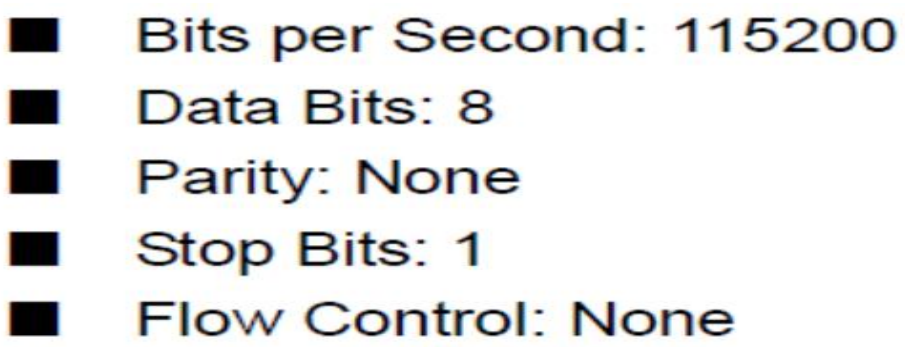
بعد از انجام مراحل فوق، نمایش ابتدایی دمو روی *LCD* ظاهر خواهد شد.

ه- اتصال به پنجره خروجی(*Console Window*)

شما میتوانید یک پنجره خروجی باز کنید و از طریق آن پیغام های بوت شدن *EVM* را مشاهده کرده یا در آنها وقفه ایجاد کنید. برای انجام این کار مراحل زیر را دنبال کنید:

۱. پورت سریال به *EVM* (قسمت مربوط به پورت سریال یا *UART*) و به قسمت سریال پورت کامپیوتر (مثلا *COM1*) وصل کنید.

۲. یک *Terminal Session* (مثلا *Minicom* در لینوکس یا *HyperTerminal* در ویندوز) اجرا کنید و برای اتصال آن به پورت سریال، مانند زیر پیکربندی کنید:



شکل۸- نحوه‌ی پیکربندی *Terminal Session*

۳. بعد از اینکه *EVM* را روشن کردید، پیغام ای بوت را مشاهده می‌کنید. با فشردن هر کلیدی می‌توانید در پیغام ها وقفه ایجاد کرده و دستوراتی در پوسته *U-Boot* تایپ کنید.

از این به بعد، دستوراتی که در پوسته *U-Boot* تایپ می‌شوند، بصورت زیر نمایش داده می‌شود:

EVM # prompt.

پیوست ب

اجرای دموها

۱-آغاز دموهای مستقل

هنگامی که *EVM* را متصل می‌کنید، دموهایی که از قبل در قطعاتی مانند *Hard disk* و یا *NAND* ذخیره و بارگذاری شده‌اند، بصورت اتوماتیک اجرا می‌شوند. این دموها تصویر، صوت، گفتار و ویدیو را *Encode* و *Decode* می‌کنند.

برای استفاده از این دموها، دو روش وجود دارد:

۱-روش مستقل

این روش حالت پیش فرض روشن شدن و شروع به کار دستگاه است. دموها بصورت اتوماتیک، بدون هیچ اتصالی به پایانه کامپیوتری[1] و در پیکربندی پیش فرض بوت اجرا می‌گردند.

۲-خط دستور

به محض اتصال *EVM* به میزکار و نصب نرم‌افزارهای مربوطه، شما می‌توانید دموها را از خط دستور لینوکس مربوط به برد اجرا کنید.

* هنگامیکه دموها را از خط دستور اجرا می‌کنید، مطمئن شوید که فرآیند رابط[1] که توسط دموهای حالت مستقل استفاده می‌شود، در حال اجرا نباشد. در غیر این صورت پیغامهای خطایی مبتنی بر اینکه درایورها نمی‌توانند باز شوند، مشاهده خواهید کرد.

[1] *Workstation*

به محض اینکه برد *EVM* بوت گردید، صفحه نمایش باید تصویر کنترل را نشان دهد. شما باید از دستگاه کنترل[۲] به منظور کنترل دموها استفاده کنید.

ترتیب دکمه ها بر روی کنترل حقیق ممکن است متفاوت باشد، اگر کنترل شما متفاوت است، دکمه هایی با برچسب یکسان را پیدا کنید.

برای استفاده از دموها در حالت مستقل، این مراحل را دنبال کنید:

۱. مطمئن شوید که باتریها در کنترل باشند.

۲. شکل (۱)، نمایش اولیه یک دیاگرام از کنترل را نشان می‌دهد که شما برای کنترل دموهای مستقل استفاده می‌کنید.

۳. از آنجایی که این کنترل، یک کنترل عمومی[۳] است، ممکن است برای استفاده از کدهایی که برای اجرای دموهای *DVEVM* ضروری است، نیاز داشته باشید آنرا تنظیم کنید. برای این کار، دکمه‌ی"*Code Search*" را تا هنگامی که چراغ قرمز روی کنترل روشن باقی بماند، نگه دارید. سپس دکمه‌ی "*DVD*" را فشرده و کد "*0020*" را وارد نمایید.

۴. اگر بطور تصادفی کنترل را در حالت TV یا حالت های دیگر قرار دادید، دکمه ی"*DVD*" را برای بازگشت به حالت درست فشار دهید.

۵. اگر کنترل کد *DVD+۰۰۲۰* را نپذیرفت، ریست کامل انجام دهید. بدین ترتیب که باتری ها را درآورده، دکمه ی "*Power*" را حداقل به مدت ۱ دقیقه نگه داشته و سپس باتری ها سر جای خود بگذارید. بعد مرحله ی ۳ را تکرار کنید.

---

[1] *Interface Process*
[2] *IR Remote*
[3] *Universal*

شکل ۱- دستگاه کنترل

۲- اجرای دموهای مستقل

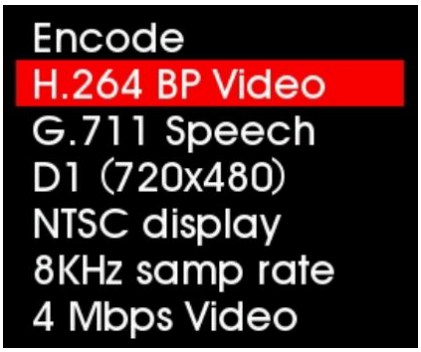
کلید "*Press*" یا "*Ok*" را روی کنترل فشار دهید. با فشردن این کلید از دیاگرام کنترل، به صفحه منوی اصلی خواهید رفت. این صفحه همانند شکل(۳) می‌باشد.

دموی *Encode + Decode* این امکان را به شما می‌دهد که ویدیو را ضبط و پخش کنید. دموی *Encode*، صوت/گفتار و ویدیو را در فرمتهایی که شما انتخاب می‌کنید، ضبط می‌کند. دموی *Decode* فایلهای گفتار/صوت و ویدیویی که شما انتخاب کردید، را پخش می‌کند. منوی *Third Party* میتواند برای اضافه کردن دموها به کار رود.

به کمک جهت های بالا و پایین می‌توانید بین گزینه ها حرکت کنید. برای انتخاب دموی مورد نظر دکمه ی"*Play*" یا "*Ok*" را فشار دهید.

۱. هنگامی که وارد دموی مورد نظر شدید، ابتدا صفحه تنظیمات را می‌بینید. در این صفحه، شما گزینه های کنترلی که به کمک آنها می‌توانید دمو را اجرا کنید، مشاهده می‌کنید. این گزینه ها در پایین صفحه قرار دارند و تنظیمات فعلی در قسمت بالای صفحه سمت راست واقع شده اند.

برای مثال، دموی *Encode* امکان تنظیم فرمت ویدیو و نرخ بیت را به شما میدهد. همچنین تنظیمات ثابت نیز نمایش داده میشود. در شکل زیر منوی دمو آورده شده است:



شکل۲- منوی دموی *Encode*

۱-۳- به کمک جهت‌های بالا و پایین، می‌توانید بین تنظیماتی که می‌خواهید تغییر دهید، حرکت کنید.

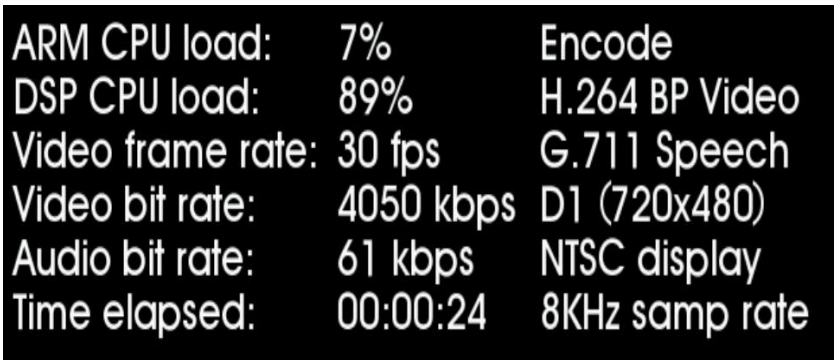
۲-۳- از جهت‌های چپ و راست برای چرخش بین گزینه‌های موجود استفاده کنید تا گزینه‌ی مورد نظر خود را بیابید.

شکل۳- نمایش منوی اصلی

۲. به کمک دکمه‌ی"*Play*" می‌توانید دموهای *Decode* و*Encode + Decode* را آغاز کنید .برای آغاز دموی *Encode* دکمه‌ی "*Rec*" را دوبار فشار دهید. این دکمه ها در زیر نمایش داده شده اند:

شکل۴- دکمه‌های مورد نیاز برای آغاز دموها

۳. هنگامی که یک دمو اجرا می‌گردد، اطلاعات در مورد تنظیمات، بار شدن پروسسور و نرخها نمایش داده می‌شوند. تنظیمات ثابت در سمت راست و اطلاعات متغیر در سمت چپ نمایش داده می‌شوند. شکل زیر یک نمونه را که مربوط به دموی Encode است را نشان می‌دهد:



شکل۵- نمایش تنظیمات مربوط به دموی *Encode*

۴. اطلاعات فوق بر روی ویدیوی در حال پخش نمایش داده می‌شود، بنابراین ویدیویی که شما مشاهده می‌کنید از ویدیوی حقیقی، تیره‌تر است. برای مخفی کردن اطلاعات و در نتیجه مشاهده‌ی بهتر ویدیو، دکمه "*Info/Select*" بر روی کنترل را فشار دهید. شما می‌توانید هنگامی که دمو در حال اجراست، شفافیت نمایش روی صفحه (*OSD=On Screen Display*) را به کمک جهت های چپ و راست تغییر دهید.

۵. اگر می‌خواهید به نمایش دمویی خاتمه دهید یا آنرا متوقف کنید، دکمه ی "*Stop*" و یا "*Pause*" را روی کنترل فشار دهید. اولین بار که دکمه‌ی "*Stop*" را فشار می‌دهید، وارد صفحه تنظیمات می‌شوید، با دوباره فشار دادن آن، وارد منوی اصلی می‌گردید.

* دموها برای اجرای الگوریتمها از *Codec Engine* استفاده می‌کند.

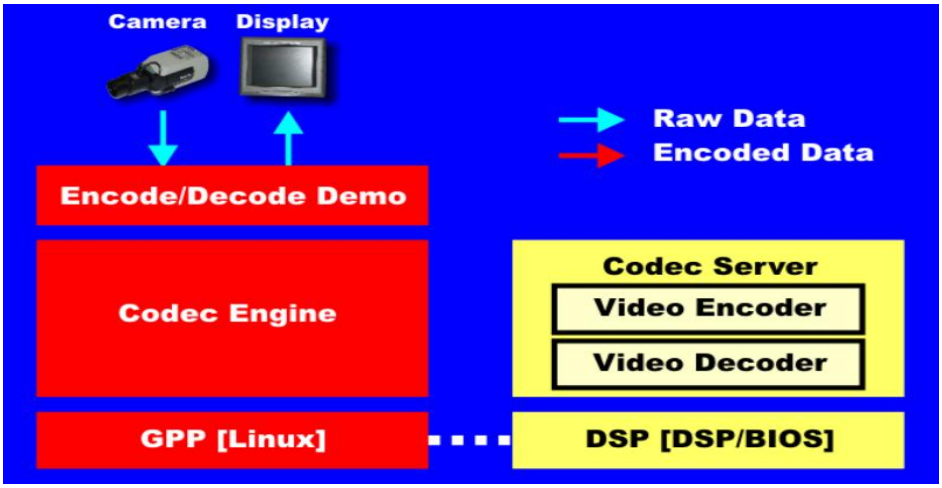
* ممکن است به نظر آید که بارگذاری *DSP CPU* در ابتدا بالاست، حتی هنگامی که هیچ الگوریتمی اجرا نمی‌شود. اما توجه کنید که بارگذاری *CPU* هنگامی که *DSP* بوت می‌گردد، از ۱۰۰٪ آغاز شده و سپس هنگامی که*DSP* منتظر عمل درخواست شده توسط *GPP* است، این بارگذاری کاهش می‌یابد. حتی هنگامی که از *DSP* هیچ استفاده ای نشود، مدت زمان کوتاهی (در حدود چند ثانیه) طول می‌کشد تا بارگذاری *CPU* به صفر برسد. این امر به دلیل محاسبات بارگذاری *Codec Engine* مربوط به *CPU* است.

۱-۲- خاموش کردن دموها

هنگامی که در صفحه‌ی منوی اصلی هستید، می‌توانید با فشردن دکمه‌ی "*POWER*" بر روی کنترل، بطور کامل از دموها خارج شوید.

۲-۲- دموی *Encode + Decode*

دموی *Encode + Decode* به شما امکان ضبط و پخش ویدیو را می‌دهد. ویدیو از دوربین می‌آید، *Encode* و سپس *Decode* می گردد و در انتها به *LCD* فرستاده می‌شود. مراحل کلی در شکل زیر آورده شده است:



شکل ۶- مراحل کلی دموی *Encode+Decode*

Encode + Decode فقط پردازش ویدیو را انجام می‌دهد و گفتار یا صوت را *Encode* و *Decode* نمی‌کند. الگوریتم ویدیویی ساپورت شده *H.264 Baseline Profile* می‌باشد.

در شکل زیر، لیست دکمه ها و عملکرد آنها در دموی *Encode + Decode* آورده شده است.

IR Remote Button	Mode	Action Performed
Up/Down	--	-- no action --
Play or OK	Setup	Begin demo
Record	--	-- no action --
Info/Select	Setup	Show / hide block diagram for demo
Info/Select	Run	Toggle information display
Left/Right	Run	Change information transparency level
Pause	Run	Pause demo (press Play to resume)
Stop	Setup / Run	Return to previous screen

شکل ۷- دکمه ها و عملکرد آنها در دموی *Encode+Decode*

برنامه ها بروی *ARM* با استفاده از لینوکس اجرا می‌شوند. سیگنال ویدیویی توسط *Codec Engine* وارد *Encoder* ها و *Decoder* های *DSP* می‌شود.

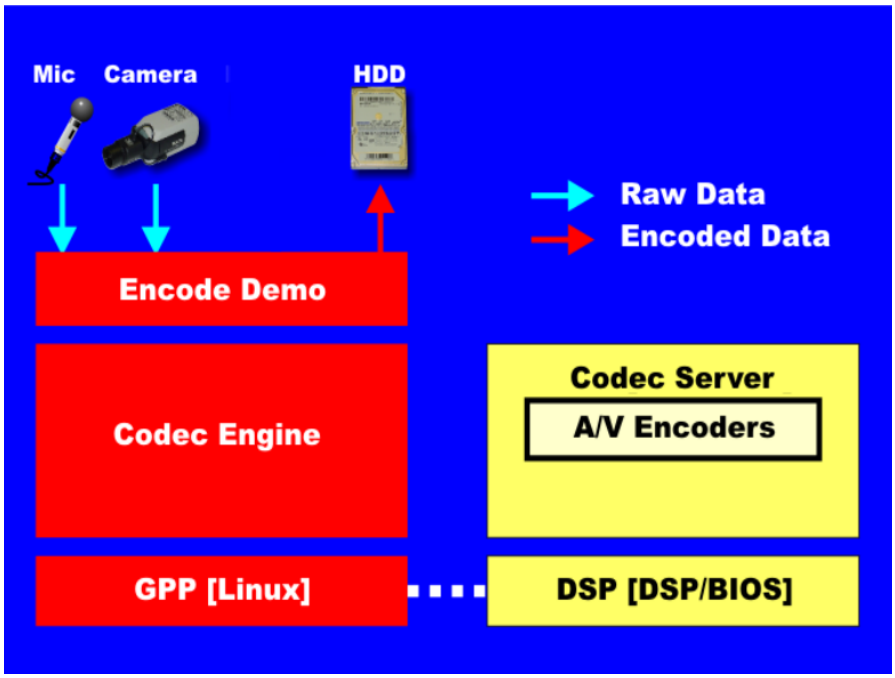
۳-۲- دموی *Encode*

همانند دموی *Encode + Decode*، دموی *Encode* نیز ویدیو را رمز گذاری می‌کند. علاوه بر آن، این دمو گفتار را نیز رمزگذاری می‌کند. منبع گفتار نیز میکروفون است. اطلاعات رمزگذاری شده در فایلهایی در *Hard Drive* مربوط به *EVM* نوشته می‌شوند .نامهای ممکن برای فایلها عبارتند از:

Demo.264, demo.mpeg4, demompeg4.g711 و *demo264.g711* .

خروجی، کدگشایی و فرستاده به اسپیکر یا *LCD* نمی‌گردد، فقط تنظیمات و اطلاعات متغیر در مورد بارگذاری و نرخ ها نمایش داده می‌شود.

مراحل کلی در شکل زیر نشان داده شده است:



شکل ۸- مراحل کلی دموی *Encode*

توجه کنید که شما فقط می توانید گفتار را رمزگذاری کنید و این رمز گذاری در مورد صوت بکار نمی‌رود. الگوریتم های ویدیویی ساپورت شده عبارتند از:

MPEG4 و *H.264*

و الگوریتم ساپورت شده برای گفتار عبارت است از: *G.711*

در شکل زیر، لیست دکمه ها و عملکرد آنها در دموی *Encode* آورده شده است.

IR Remote Button	Mode	Action Performed
Up/Down	Setup	Change option selection
Left/Right	Setup	Change setting of selected option
Play	Setup	Switch to decode demo setup
Record (twice) or OK	Setup / Run	Begin encode demo, send unencoded data to display

IR Remote Button	Mode	Action Performed
Info/Select	Setup	Show / hide block diagram for demo
Info/Select	Run	Toggle information display
Left/Right	Run	Change information transparency level (There is no display for encode demo behind the information.)
Pause	Run	Pause demo (press Record to resume)
Stop	Setup / Run	Return to previous screen

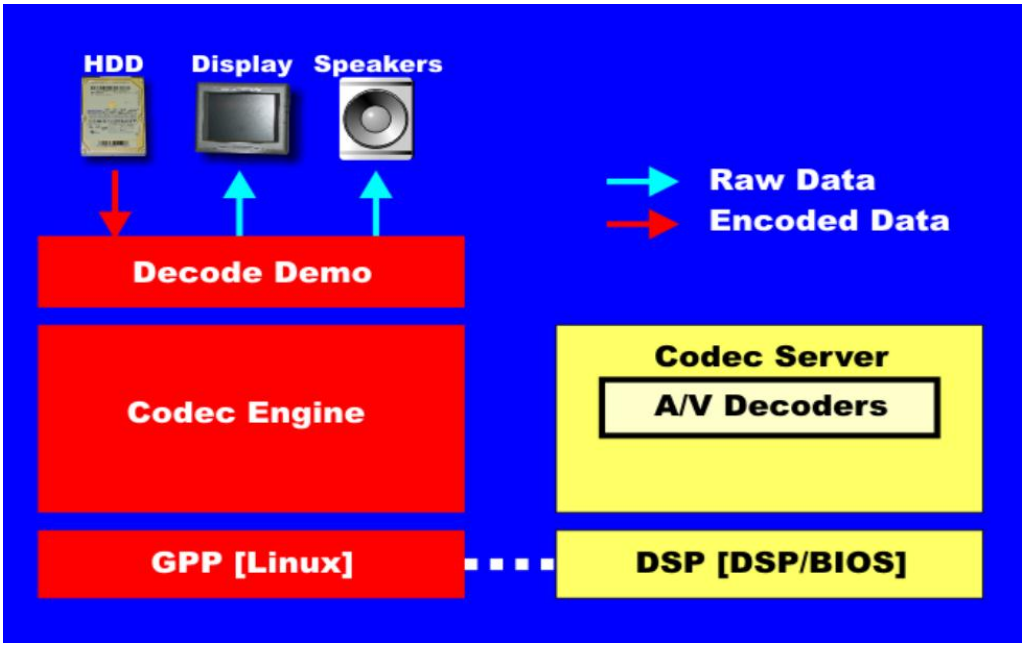
شکل ۹- دکمه ها و عملکرد آنها در دموی *Encode*

برنامه ها بروی *ARM* با استفاده از لینوکس اجرا می‌شوند. سیگنالهای صوت و ویدیو توسط *Codec Engine* وارد *Encoder* های *DSP* می‌شود.

۴-۲- دموی *Decode*

دموی *Decode* فایلهای ویدیویی و گفتار/صوتی که انتخاب کردید را نمایش می‌دهد. شما می‌توانید یک سورس فایل ویدیویی و یک سورس فایل صوت یا گفتار انتخاب کنید. از جهت های چپ و راست برای انتخاب فایل های دمو و فایلهایی که توسط دموی *Encode* ایجاد شده و در

هارد درایو ذخیره گشته است، استفاده کنید. سیگنال های کدگشایی شده به اسپیکرها و *LCD* فرستاده خواهند شد. شکل زیر روند کلی این دمو رانشان می‌دهد.



شکل ۱۰- روند کلی دموی *Decode*

الگوریتم های ویدیویی ساپورت شده عبارتند از:

MPEG4 ، *MPEG2* و *H.264*

الگوریتم صوت ساپورت شده *ACC* و الگوریتم گفتار ساپورت شده *G.711* می‌باشد.

در جدول زیر، لیست دکمه ها و عملکرد آنها در دموی *Decode* آورده شده است.

IR Remote Button	Mode	Action Performed
Up/Down	--	-- no action --
Left/Right	Setup	Select a different file combination
Play or OK	Setup	Begin decode demo
Record	--	-- no action --
Info/Select	Setup	Show / hide block diagram for demo
Info/Select	Run	Toggle information display
Left/Right	Run	Change information transparency level
Pause	Run	Pause demo (press Play to resume)
Stop	Setup / Run	Return to previous screen

شکل ۱۱- دکمه ها و عملکرد آنها در دموی *Decode*

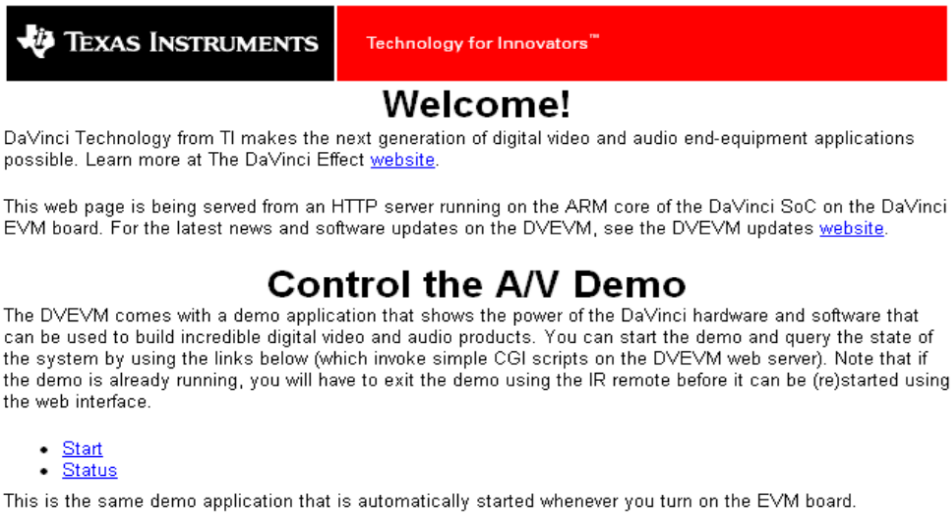
برنامه‌ها بروی *ARM* با استفاده از لینوکس اجرا می‌شوند. سیگنالهای صوت و ویدیو توسط *Codec Engine* وارد *Decoder* های *DSP* می‌شود.

۳- اجرای دموی شبکه یا دموی ***Network***

بعنوان مثالی از ساپورت شبکه‌ای *TCP/IP*، مثالهای *DVEVM* شامل یک وب سرور *HTTP* می‌باشند. این وب سرور در قسمت *GPP* بعنوان قسمتی از برنامه هایی که با اجرای لینوکس آغاز به کار می‌کنند(*Startup Sequence*)، اجرا می گردد. این وب سرور برای سرویس دادن به درخواستهایی از جستجوگرهای با استاندارد *TCP/IP* پورت ۸۰ پیکربندی شده است.

بعد از اینکه برد *EVM* بوت گردید، یک کامپیوتر را به همان شبکه ای که برد متصل است، وصل کنید. آدرس اینترنتی به شکل *http://ip-address-of-evm* دریک جستجوگر وب(مثل *Internet Explorer*، *Firefox* یا *Opera*) وارد کنید. آدرس *IP* مربوط به برد، در گوشه سمت راست و پایین منوی اصلی دموهای *A/V* نشان داده خواهد شد.

شما باید صفحه وبی با اطلاعاتی در مورد تکنولوژی داوینچی و نرم افزار *DVEVM* همانند شکل زیر، ببینید:



از این صفحه برای عملکرد متقابل با برد و اجرای دموهای *A/V* استفاده کنید. دو فایل آغاز کننده *CGI* بر روی *EVM* شما را قادر می سازد تا دموها را آغاز کرده و ببینید چه فرایندهایی برروی برد اجرا می شوند. اگر قصد دارید دموی آغاز شده از صفحه وب راببینید، مطمئن شوید که ابتدا از دمو خارج شده باشید. (در صفحه منوی اصلی، دکمه "*POWER*" را فشار دهید.).

نرم افزار وب یک بسته‌ی *Open-Source* است که *THTTPD* نامیده می‌شود. (*http://www.acme.com/software/thttpd*) .این بسته کم حجم، سریع و قابل حمل است. سورس کد آن به همراه نرم افزار *DVEVM* آورده شده است.

فهرست مراجع

[1] U. Gargi, R. Kasturi, and S. H. Stayer(2000)" Performance Characterization of Video-Shot-Change Detection Methods", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 1, pp. 1-13

[2] شفاعی کیانوش،"مرجع کامل پردازنده های *DSP* چاپ اول انتشارات آستان قدس، سال ۱۳۸۹

[3] shang-hong lai,wei-kuang li,"New video shot change detection algorithm based on accurate motion and illumination estimation" ,proceeding of SPIE,Vol.4676

[4]markus seidi,Matthias zeepzauer,chiristian breitender,'A Study of Gradual Transition Detection in Historic Film Material"

[5] collim o'toole,alan smeaton,noel murphy and sean marlow,'Evaluation of Automatic Shot Boundary Detection on a Large Video Test Suite'

[6] carlos cuevas,narciso Garcia,"Real-time shot detection based on motion analysis and multiple low level techniques",E.T.S .ing. Telecominication

[7]snoek,Worring,"Multimodal video Indexing:a reviw of the state-of-the-art",Multimedia tools and applications,pp.5-35

[8]wei zeng,wen gao,"Shot Change Detection on H.264/AVC Compressed Video",IEEE

[9] nithya manickam,aman paranami,sharat chandran,"Reducing False Positives in Video Shot Detection Using Learning Techniques"

[10] yichuan hu,bo han,guijin wang,xinggang lin,"Enhanced shot change detection using motion features for soccer video analysis",ICME,2007

[11] paul browne,alan f smeaton,noel murphy,"Evaluating and Combinig digital video shot boundary detection algorithms"

[12] D. Lelescu and D. Schonfeld, "Statistical sequential analysis for real-time video scene change detection on compressed multimedia bitstream," IEEE Trans. Multimedia, vol. 5, no. 1, pp. 106 – 117, Mar. 2003.

[13] A. Hanjalic, "Shot-boundary detection: Unraveled and resolved?" IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 2, pp. 90 – 105, Feb. 2002.

[14] J. Nam and A. Tewfik, "Detection of gradual transitions in video sequences using b-spline interpolation," IEEE Trans. Multimedia, vol. 7, no. 4, pp. 667–679, Aug. 2005.

[15] H. Zhang and S. S. A. Kankanhalli, "Automatic partitioning of full-motion video," ACM Multimedia Systems, vol. 1, no. 1, pp. 10 – 28, Jan. 1993

[16] Z. Cernekova, C. Kotropoulos, and I. Pitas, "Video shot segmentation using singular value decomposition," in Proc. 2003 IEEE International Conference on Multimedia and Expo, vol. II, Baltimore, Maryland, USA, July 2003, pp. 301 – 302.

[17] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classification production effects," ACM Multimedia Systems, vol. 7, no. 1, pp. 119 – 128, Jan. 1999.

[18] R. Zabih, J. Miller, and K. Mai, ''A feature-based algorithm for detecting and classifying scene breaks,'' Proc. ACM Multimedia 95, pp. 189–200, San Francisco, CA ~1995!.

[19] Mehdi Sharifzadeh, Farnaz Azmoodeh, and Cyrus Shahabi," Change Detection in Time Series Data Using Wavelet Footprints", SSTD 2005, LNCS 3633, pp. 127–144, 2005

[20] Zheng-Yun Zhuang, Chiou-Ting Hsu, Hemg-Yow Chen, Ming Ouhyoung, Ja-Ling Wu ," Efficient Multiresolution Scene Change Detection By Wavelet Transformation"

[21] Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer," Performance Characterization of Video-Shot-Change Detection Methods", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 10, NO. 1, FEBRUARY 2000

[22] Byung Cheol Song and Jong Beom Ra," Automatic Shot Change Detection Algorithm Using Multi-stage Clustering for MPEG-Compressed Videos", Journal of Visual Communication and Image Representation 12, 364–385 (2001)

[23] H. Ueda, T. Miyatake, and S. Yoshizawa, ''IMPACT: an interactive natural-motion-picture dedicated multimedia authoring system,'' Proc. CHI. 1991, pp. 343–350 ACM, New York ~1991!.

[24] R. Kasturi and R. Jain, ''Dynamic vision,'' in Computer Vision: Principles, R. Kasturi and R. Jain, Eds., IEEE Computer Society Press, Washington ~1991!.

[25] Z.-N. Li, X. Zhong, and M. S. Drew, "Spatialtemporal joint probability images for video segmentation," Pattern Recognition, vol. 35, no. 9, pp. 1847 – 1867, Sept. 2002.

[26] W. K. Li and S. H. Lai, "Integrated video shot segmentation algorithm," in Storage and Retrieval for Media Databases, ser. Proceedings of SPIE, vol. 5021, Jan. 2003, pp. 264 – 271.

[27] J. Yu and M. D. Srinath, "An efficient method for scene cut detection," Pattern Recognition Letters, vol. 22, no. 13, pp. 1379 – 1391, Nov. 2001.

[28] R. Lienhart, "Reliable dissolve detection," in Storage and Retrieval for Media Databases 2001, ser. Proceedings of SPIE, vol. 4315, Jan. 2001, pp. 219–230.

[29] G. Boccignone, A. Chianese, V. Moscato, and A. Picariello, "Foveated shot detection for video segmentation," IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 3, pp. 365–377, Mar. 2005.

[30] Wei Zeng, Wen Gao," Shot Change Detection on H.264/AVC Compressed Video"

[31] Janko Calic ,Ebroul Izquierdo," TOWARDS REAL-TIME SHOT DETECTION IN THE MPEG-COMPRESSED DOMAIN",

[32] Efnan Sora Gunal, Selcuk Canbek, and Nihat Adar ," Gradual Shot Change Detection in Soccer Videos via Fractals"

[33] markus piccering,stefun ruger,"muti-timescale video shot-change detection"

[34] feifei lee,koji kotani,qui chen,tadahiro ohmi,"shot change detection using a novel histogram feature in compressed video",IEICE Electronics Express,Vol.5,No14,pp.530-509

[35] j.yuan,H.wang,L.xiao,W.zheng,j.li,f.lin and b.Zheng."A formal study od shot boundary detection",IEEE transactions on Cuircuits and systems for video Technology,vol.17,2007,pp.168-186

[36] R. Dugad, K. Ratakonda, and N. Ahuja, Robust Video Shot Change Detection, IEEE Workshop on Multimedia Signal Processing , Redondo Beach, California, Dec. 1998

[37] " Davinci-DM644x Evaluation Module" Spectrum Digital, INC

[38] "TMS320DM6446 DVEVM v2.0Getting Started Guide", SPRUE66E December 2008

[39]Staff of BDTI,"texas Instruments Digital video evaluation module (dvevm)",2007

[40] "Codec Engine Application Developer User's Guide", SPRUE67D ,September 2007

[41]Tms320dm644,SPRS283H–DECEMBER 2005–REVISED SEPTEMBER 2010

[42] xDAIS-DM (Digital Media) User Guide, SPRUEC8B ,January 2007

[43] TMS320C5505,SPRS660E ‐AUGUST 2010‐REVISED JANUARY 2012

[44]Steve Blonstein,Alan Campbell,"OMAP and DAVINCI Software for DUMMIES",Wiley Publishing inc.

Abstract:

Nowadays, digital video files which contain voice and image data are accessible anywhere. Rapid development of digital video equipment such as digital cameras has extremely widened application area of video including video conferencing and electronic education. Therefore, a large amount of video data has been created which the need to handling and organizing such a large database has caused shot boundary detection to be an active research area in digital video processing field.

In this thesis, we aim to design and implement a video shot change and shot type (i.e. gradual and abrupt change) detection method such that is real-time implementable on DSP processors platform. We perform this DSP implementation in both software (using Code Composer Studio®) and hardware (using an available Evaluation Module-EVM).

Our proposed method consumes low memory and has high execution speed. We implemented this method on an EVM based on TMS320C5505 DSP processor. In simulation, we evaluated the proposed method efficiency using precision, recall, and mean run time measures for each frame of our collected video sequences. This video collection consists of four categories of sport, news, cinema, and cartoon. We reached 95.2% and 94.5% for precision and recall measures respectively and 0.095 seconds for mean run time measure.

For DSP implementation, we computed and reported the computational complexity in terms of consumed machine cycle and total number of multiplication and summation. After implementation on TMS320C5505 DSP platform, we tried to real-time implement our proposed method on TMS320DM6446 digital video processor which is still continuing, although it is difficult because of not access to useful resources and professional individuals. In this thesis, we report the respective results obtained till now.

We gathered a relatively large video database using a digital/analog TV signal capture with specified features such as frame rate and content type.

Keywords: Shot Change Detection- Hardware Implementation- Shot Type- TMS320C5505 DSP Processor- TMS320DM6446 DSP Processor.

Shahrood University of Technology

Faculty of Electrical Engineering

Online video shot change detection and real-time implementation on DSP processor platform

Thesis submitted to the Graduate Studies Office in partial fulfillment of the requirements for the

Degree of Master of Science (M.Sc.)

Mostafa Safaie

Supervisor:

Dr. Hadi Grailu

Dr. Omid Marozi

Associate Supervisor:

Dr. Ali Soleimani

January 2013