کدگذاری ویدئو در پردازنده های گرافیکی چند هسته ای

چکیده

در این مقاله، ما، استفاده از واحدهای پردازنده گرافیکی چند هسته ای (GPUs) برای کدگذاری و کدگشایی ویدئوها را بررسی می کنیم. پس از بررسی اجمالی کدگذاری های ویدئویی و GPUs، برخی از کارهای قبلی را در مورد ساختاردهی ماژول های کدگذاری ویدئو مرور می کنیم به طوری که قابلیت پردازش موازی گسترده GPUs را می توان مهار نمود. همچنین کارهای قبلی انجام شده در مورد پارتیشن بندی جریان کدگشایی ویدئو بین واحد مرکزی پردازش (CPU) و (GPU) را مرور می کنیم. سپس، به صورت مفصل، برآورد حرکت سریع مبتنی بر GPU را برای نشان دادن برخی از ملاحظات طراحی در استفاده از GPUها در کدگذاری ویدئو و تعادل بین عملکرد افزایش سرعت و انحراف نرخ مورد بحث قرار می دهیم. نتایج ما، اهمیت مواجهه موازی سازی داده های ممکن را در طراحی الگوریتم ها برای GPUs برجسته می کند.

1. مقدمه

امروزه، کدگذاری ویدئو [5]-[1] تبدیل به یک تکنولوژی مرکزی در محدوده وسیعی از برنامه های کاربردی شده است. برخی از آنها شامل تلویزیون دیجیتال، دی وی دی، پخش اینترنتی ویدئو، ویدئو کنفرانس، آموزش از راه دور و مراقبت و امنیت می باشد. محدوده وسیعی از استانداردها و الگوریتم های کدگذاری ویدئو گسترش یافته اند (H.264/AVC, VC-1, MPEG-2, AVS) تا به نیازها و ویژگی های عملیاتی تجهیزات و نرم افزارهای

کاربردی مختلف رسیدگی کنند. با برنامه های کاربردی رایج در فن آوری های کدگذاری ویدئویی، بررسی تغییرات موثر در سیستم های کدگذاری ویدئویی بر روی جایگاه و پردازنده های محاسبه گر متفاوت مهم است.

اخیراً، واحدهای پردازنده گرافیکی (GPUs) به شکل واحدهای پردازنده همراه با واحد مرکزی پردازش (CPU) بیرون آمده است تا تجهیزات و نرم افزارهای کاربردی مختلف عددی و پردازش سیگنال را شتاب دهد.GPU های مدرن از صدها هسته پردازنده جدا که قادر به دستیابی به عملکرد بسیار زیاد محاسبات موازی می باشد، تشکیل شده است. برای مثال پردازنده NVIDIA GeForce 8800 GTS دارای 96 پردازنده جریان مستقل است که هر کدام در 1.2 GHz عمل می کند. پردازنده های جریان را می توان با هم گروه بندی کرد تا عملیات های Single Instruction Multiple Data (SIMD) (ساختار بندی داده های متعدد) را اجرا کند که برای برنامه های فشرده محاسباتی مناسب است. با مراجع در رابطه با ابزار کدگذاری GPU مانند موضوع و رابط کدگذاری C، GPUها را می توان برای انجام انواع وظایف پردازش علاوه بر عملیات متعارف راس و پیکسل مورد استفاده قرار داد.

با بسیاری از کامپیوترهای شخصی (PCs) و کنسول های بازی مجهز به GPUهای چند هسته ای که قادر به انجام محاسبات با هدف کلی هستند، مطالعه و بررسی چگونگی استفاده از GPU برای کمک به CPU در محاسبات فشرده مثل کارهایی از قبیل فشرده سازی ویدیو / باز کردن مهم است. در حقیقت، از آنجایی که محتویات HD عامه پسند می شود، کدگذاری ویدئو نیاز بیشتر و بیشتری به توان محاسبه دارد. بنابراین، اعمال نفوذ محاسباتی GPU می تواند یک روش مقرون به صرفه برای پاسخگویی به الزامات این تجهیزات و برنامه های کاربردی باشد. توجه کنید که با ده ها استاندارد ممکن در کدگذاری ویدئو (H.264, MPEG-2, AVS, VC-1WMV, DivX) پیروی از یک راه حل انعطاف پذیر در نرم افزار یک مزیت است.

با تمرکز بر روی تجهیزات و برنامه های کاربردی کدگذاری ویدئو که بر پایه نرم افزار هستند و در رایانه های شخصی اجرا می شوند و کنسول های بازی مجهز به CPU و GPU، این مقاله چگونگی استفاده از GPU برای شتاب دادن کدگذاری و کدگشایی ویدئو ها را بررسی می کند. فعالیت های اخیر بکارگیری GPU/CPU چند هسته ای

را برای تجهیزات و برنامه های کاربردی پردازنده تصویر/ویدئو مختلف پیشنهاد داده است. جدول 1 برخی از آنها را خلاصه کرده است. در این مقاله، ما به بررسی کارهای قبلی بر روی کدگذاری ویدئو و کدگشایی آنها برای نشان دادن چالش ها و مزایای استفاده از اجرای GPU می پردازیم. به طور خاص، ما بحث کارهای قبلی در برآورد حرکت مبتنی بر GPU، جبران حرکت و پیش بینی داخلی را مطرح می کنیم. تمرکز ما بر روی این است که چگونه الگوریتم می تواند طراحی شود تا قابلیت پردازش موازی GPU را به شکل گسترده مهار کند. علاوه بر این، ما در مورد کارهای قبلی در پارتیشن بندی جریان کدگشایی بین CPU و GPU بحث می کنیم. (برای تکمیل بحث، ما همچنین گزارش تسریع در نتایج کار قبلی را ارائه می دهیم با این حال که چون فن آوری نرم افزار/سخت افزارGPUهای چند هسته ای به طور چشمگیری در چند سال گذشته تکامل یافته، برخی از نتایج را می توان از رده خارج شده دانست). پس از آن، ما به بررسی برآورد حرکت سریع بر اساس GPU می پردازیم. ما در مورد برخی از استراتژی های شکستن وابستگی بین واحدهای مختلف داده ها و بررسی معاوضه بین افزایش سرعت و کارایی کدگذاری بحث خواهیم نمود.

بقیه این مقاله به شرح زیر سازماندهی شده است. ما برای اولین بار یک نمای کلی از وضعیت مدرن را در کدگذاری های ویدئویی و GPUها آماده می کنیم. ما همچنین از چالش های استفاده از GPUها برای کمک به کدگذاری های ویدئویی صحبت می کنیم. سپس، ما به مرور کار قبلی در کدگذاری های ویدئویی شتاب داده شده GPU می پردازیم. پس از آن، ما به مطالعه برآورد حرکت سریع مبتنی برGPU می پردازیم. در نهایت، این مقاله با نتیجه گیری به پایان می رسد.

2. پیش زمینه

A. کدگذاری ویدئو

آخرین استانداردهای کدگذاری ویدئو از عملکرد کدگذاری مدرن بدست می آید. برای مثال، H.264/AVC، که آخرین استاندارد بین المللی کدگذاری ویدیو تایید شده توسط ITU-T و ISO / IEC می باشد، به طور معمول

نیاز به 60٪ یا کمتر از نرخ بیت در مقایسه با استانداردهای قبلی برای رسیدن به همان کیفیت بازسازی دارد. [5]. سایر الگوریتم های پیشرفته کدگذاری ویدیو، مانند AVS-video توسعه یافته بوسیله کارگروه استاندارد کدگذاری صوتی و ویدئویی چین، یا VC-1 در ابتدا توسط مایکروسافت نیز عملکرد فشرده سازی رقابتی را به دست آورده است. در زیر ما مروری بر استاندارد کدگذاری ویدیو H.264 ارائه خواهیم نمود.

استاندارد H.264 کدگذاری ویدیو مبتنی بر رویکرد کدگذاری ویدیو مبتنی بر بلوک ترکیبی طراحی می شود که به علت استانداردهای اولیه کدگذاری ویدئو استفاده می شود.

جدول I

برنامه های پردازش صوت و تصویر در پردازنده های چندهسته ای	
مثال ها	برنامه ها
تخمین حرکت [23]-[19]، پیش بینی داخلی [27]-[24]، تبدیل [28]	کدگذاری ویدئو
جبران حرکت [10],[29]، طراحی کدگشا [10],[32]-[30]،	کدگذاری ویدئو
فشردگی متون [33]	تصاویر در گستره دینامیک بالا (HDR)
سیستم واترمارکینگ ویدئویی زمان واقعی [14]	واترمارکینگ ویدئویی
محاسبات برداری و ماتریسی [12]، تبدیل فوریه سریع و کانوولوشن [13]	کرنل های پردازش سیگنال
تبدیل Hough [34]، تبدیل تصادفی [36],[35]، تبدیل chirplet [35]،	استخراج ویژگی [37]
تجزیه و تحلیل تصویر	

الگوریتم کدگذاری از ارتباط فضایی بین پیکسل های کناری تصویر استفاده می کند. علاوه بر این نیز برای رسیدن به فشرده سازی، همبستگی زمانی بین تصاویر کناری را در دنباله ویدئویی ورودی مورد استفاده قرار می دهد. شکل 1، بلوک دیاگرام کدگذار را به تصویر می کشد. تصویر ورودی به بلوک های مختلف تقسیم می شود و هر بلوک می تواند با استفاده از پیکسل های بازسازی شده همجوار در همان چارچوب به عنوان پیش بینی کننده، پیش بینی داخلی را انجام دهد. H.264 از اندازه های 16*16، 8*8 و 4*4 بلوک پیش بینی داخلی پشتیبانی می کند، و روش های مختلف را برای ساخت نمونه های پیش بینی از پیکسل بازسازی شده مجاور میسر می سازد. متناوباً، بلوک ورودی ممکن است با استفاده از بلوک های بازسازی در چارچوب مرجع پیش بینی کننده، پیش بینی داخلی را انجام دهد. پیش بینی داخلی می تواند بر اساس اندازه پارتیشن 16*16، 16*8، 8*16، 8*8، 8*4، 4*8، یا 4*4 باشد.

جابجایی بین بلوک حاضر و بلوک مرجع می تواند تا حدود دقت یک چهارم واحد باشد و با بردار حرکت و شاخص تصویر مرجع نمایان می شود.

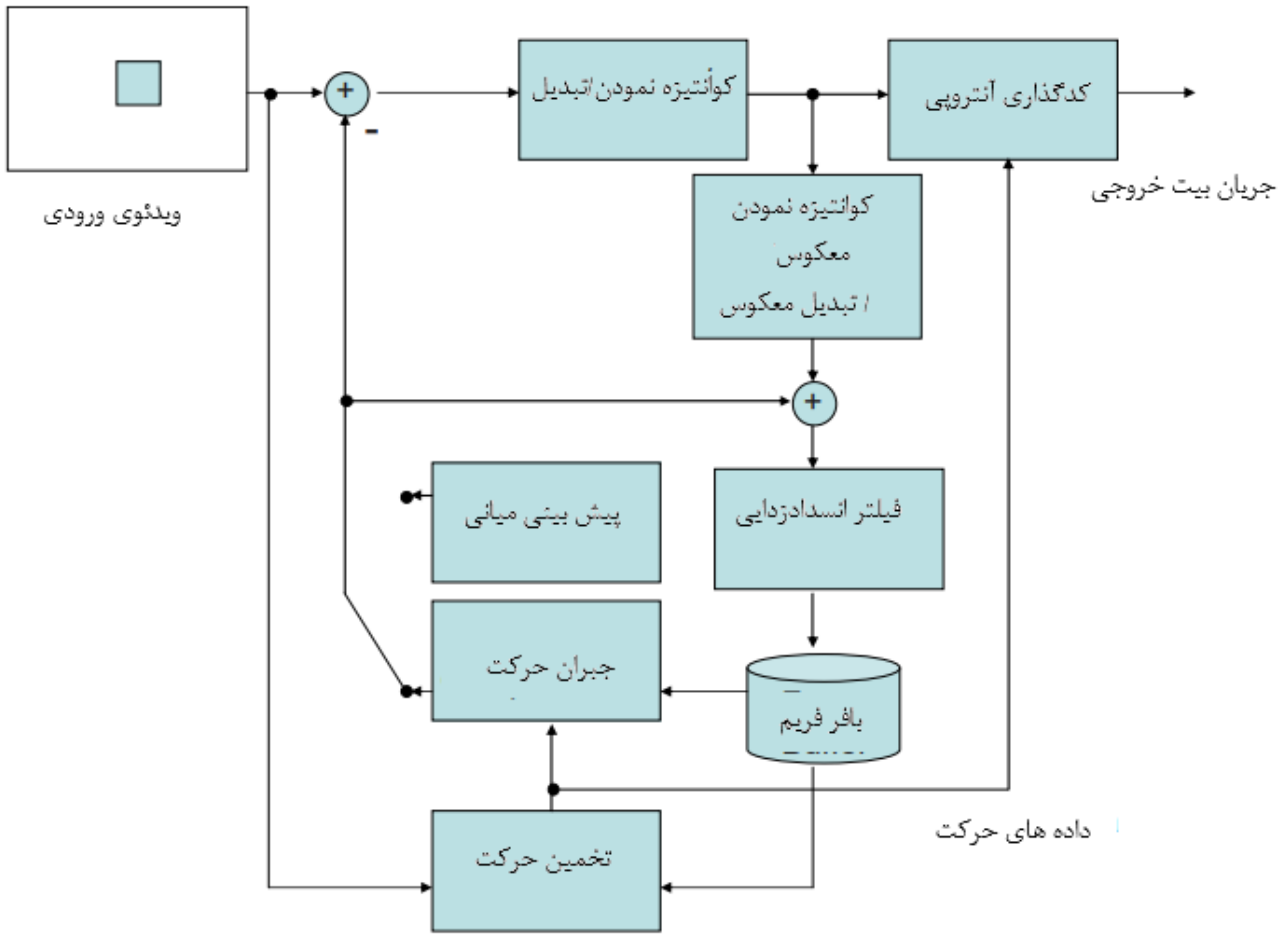
بنابراین سیگنال پیش بینی باقی مانده از پیش بینی داخلی یا پیش بینی میانی، متحمل تبدیل می شود تا ارتباط زدایی داده ها صورت گیرد. در H.264، یک تبدیل عدد صحیح 4*4 قابل جدا شدن استفاده می شود، که شبیه به 4*4 DCT است اما از عدم تطابق بین تبدیل مستقیم و معکوس اجتناب می کند. سپس، ضرایب تبدیل اسکالر کوانتیزه خواهد شد و به صورت زیگزاگ اسکن می شود. از Context Adaptive Variable Length Coding (CAVLC) برای کدگذاری آنتروپی ضرایب تبدیل اسکن شده استفاده نمود. CAVLC یک نقشه کدگذاری تطبیقی است، و می تواند بین جداول کد واژه مختلف در طول کدگذاری بسته به ارزش عناصر کد تغییر کند. متناوباً، ضرایب تبدیل می توانند توسط Context-Adaptive Binary Arithmetic Coding (CABAC) کدگذاری شوند. برای کاهش مسدود شدن مصنوعات، یک فیلتر تطبیقی انسدادزدایی درون حلقه برای بازسازی از حلقه بازخورد استفاده می شود.

B. واحد های پردازنده گرافیکی (GPUs)

در اصل GPUها به عنوان سخت افزار تخصصی برای گرافیک های D3 طراحی شده است که به تازگی برای سرعت بخشیدن به برنامه های کاربردی فشرده ریاضی در رایانه های شخصی و کنسول های بازی به عنوان واحد پردازش همزمان پدید آمده است. یک ویژگی کلیدی GPUهای مدرن اینست که آنها قابلیت محاسبات موازی انبوه را از طریق صدها هسته پردازش عمدتاً جدا انجام می دهند. به عنوان مثال، پردازنده NVIDIA GeForce 8800 GTS شامل 96 پردازنده جریان است که هر یک در 1.2 گیگاهرتز عمل می کنند.

فلسفه طراحی GPUها از هدف کلی CPU متفاوت است. در این سال ها، GPUها با هدف پشتیبانی از تعداد بیشماری از محاسبات و مقدار زیاد تبدیل داده های که برای بازی های پیشرفته لازم است طراحی شده اند. بعلاوه، آنها نیاز به پاسخگویی به احتیاجات دقیق هزینه برنامه های کاربردی مصرف کننده دارند. بنابراین، GPUها بسیار

برای محاسبات ریاضی مقرون به صرفه شده اند. علاوه بر این، قابلیت محاسبه اوج GPUها در سرعت بیش تر از پردازنده های همه منظوره در حال افزایش است.



شکل 1. الگوریتم کدگذاری H.264/AVC [5].

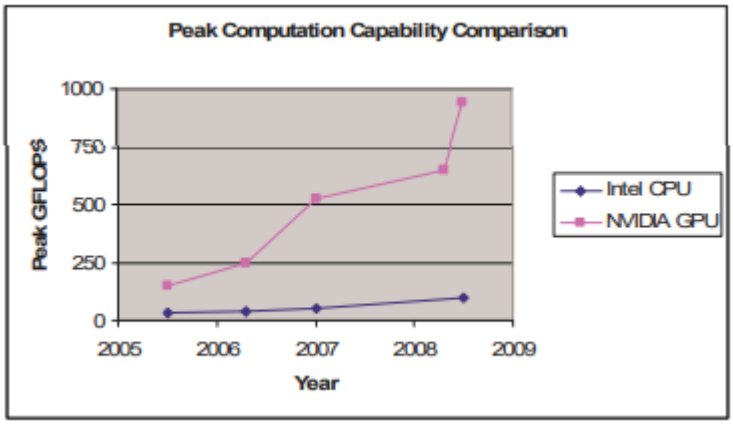
علاوه بر قابلیت محاسبات ریاضی، تفاوت های اساسی دیگری بین CPU ها و GPUها وجود دارد. اول، به منظور پرداختن به طیف گسترده ای از برنامه های کاربردی، پردازنده های همه منظوره از بسیاری از ترانزیستورها برای پیاده سازی سخت افزار کنترل پیچیده استفاده می کنند که می تواند از برخی از توابع کنترل پیشرفته مانند پیش بینی شاخه را حمایت کند. بر خلاف انتظار، GPUها حوزه تراشه را به محاسبات ریاضی اختصاص می دهند. در نتیجه، GPUها ممکن است به خوبی برای برنامه ها با بسیاری از عبارات شرطی عمل نکنند. دوم، CPUها به منظور کاهش تاخیرهای دستور العمل ها و دسترسی به داده ها از تعداد زیادی از حوزه تراشه برای پیاده سازی حافظه کش استفاده می کنند. GPUها، از سوی دیگر، از مدل های حافظه بسیار ساده تر استفاده می کنند، اما با

تکیه بر درجه بالایی از موازی سازی در نرم افزار برای مخفی کردن تاخیر دسترسی به حافظه. بنابراین، این مورد در راس مواجهه مقدار زیادی از موازی سازی داده ها در برنامه های GPU است.

C. کدگذاری ویدئو به کمک CPU: چالش ها

به دنبال بحث قبل، روشن است که فقط انواع خاصی از محاسبات برای اجرای GPU مناسب می باشند. به طور خاص، مهار کامل قدرت محاسباتی در GPU، با استفاده از تعداد گسترده ای از هسته های پردازش به شکل موازی نیاز به طراحی الگوریتم دارد. به عنوان مثال، یک برنامه خوب ممکن است هزاران موضوع را به طور همزمان بر روی یک high-end GPU اجرا کند تا تمام هسته های پردازش کار را به طور مداوم نگه دارد. بنابراین، یکی از چالش های اصلی در استفاده از GPU برای کدگذاری های ویدئویی این است که چگونه به ساختار یک ماژول خاص برای نشان دادن داده های موازی ممکن برسیم. توجه کنید که این نمی تواند برای برخی از ماژول های کدگذاری ویدئو ناچیز باشد، زیرا وابستگی می توانند بین واحدهای مختلف داده ها در محاسبه وجود داشته باشد، همانطورکه توسط کار قبلی نشان داده شده است [22]، [27],[25],[24]. علاوه براین، دستورالعمل های کنترل جریان (if switch, do, for, while) می توانند به طور چشمگیری، عملکرد اجرای GPU را تنزل دهند، زیرا چنین دستورالعمل هایی می توانند سبب شوند تا شیارهای مختلف از مسیرهای اجرای مختلف پیروی نمایند و این اجرا باید سریال شود [39]. بنابراین، استفاده از GPUها برای کدگذاری آنتروپی مانند CAVLC می تواند چالش برانگیز باشد. علاوه براین، یک پیاده سازی باید برای اجتناب از دسترسی به داده ها خارج از تراشه تا حد ممکن آزمایش شود که می تواند متحمل نهفتگی قابل توجهی شود (به خاطر داشته باشید که GPU برای نهفتگی دسترسی حافظه بهینه سازی نمی شود). به طور مثال، برخی GPUها ممکن است نیاز به 400 تا 600 چرخه نهفتگی برای دسترسی به حافظه خارج از تراشه داشته باشد (در حالیکه آنها می توانند نقطه شناور تک دقت چندافزودنی را در یک چرخه در هر هسته اضافه نمایند). [39]. توجه داشته باشید که امکان پنهان نمودن چنین نهفتگی دسترسی حافظه وجود دارد، اگر محاسبات ریاضی مستقل کافی وجود داشته باشد. بنابراین، در صورت

امکان، یک ماژول کدگذاری ویدئویی باید با شدت محاسباتی بالا پیاده سازی شود (که به صورت تعداد عملیات های ریاضی در هر عملیات دسترسی حافظه تعریف می شود). در برخی وضعیت ها، محاسبه دوباره برخی متغیرها نسبت به بارگذاری آنها از حافظه خارج از تراشه می تواند کارآمدتر باشد.



شکل 2. توانمندی محاسباتی پیک GPUها و CPUها [41],[39]

3. کار قبلی

در این بخش، ما کار قبلی در مورد اعمال GPUها را برای کدگذاری ویدئو بازنگری می کنیم. کار قبلی برای استفاده از GPUها برای انجام تخمین حرکت [22]-[19]، پیش بینی میانی [27]-[24] و جبران حرکت [29],[10] پیشنهاد می شود. توجه داشته باشید که تخمین حرکت، پیش بینی میانی و جبران حرکت، برخی از ماژول های محاسباتی شدید به ترتیب در کدگذاری داخل فریم، کدگذاری داخل فریم و کدگشایی هستند. بنابراین، درک این مورد مهم است که چگونه این ماژول ها می توانند به طور کارآمد روی GPUها پیاده سازی شوند. علاوه بر این ماژول ها، تبدیل cosine گسسته مبتنی بر GPU (DCT) در [28] بررسی شده است. به نظر می رسد کار قبلی در مورد فیلتر رفع انسداد مبتنی بر GPU وجود ندارد. چون فیلتر انسدادزدایی شامل برخی از اظهارنظرات شرطی برای تعیین قوت فیلتر کردن در هر مرز بلوک می شود، برخی مطالعات ممکن است برای تعیین عملکرد آن روی GPUها لازم باشد.

A. تخمین حرکت روی GPUها

تخمین حرکت (ME) یکی از ماژول های با شدت محاسباتی بالا در کدگذاری ویدئویی می باشد و علاقه زیاد به تخلیه بار آن به GPUها برای بهبود عملکرد کلی کدگذاری وجود دارد. کار قبلی در این حوزه روی الگریوتم های ME تمرکز می کند که در آن مجموع تفاوت های مطلق (SAD) در تطبیق بلوک برای تعیین بهترین کاندید استفاده می شود. محاسبه SAD را می توان به آسانی زمانی که هر پیکسل فردی در بلوک کنونی به طور مستقل با پیکسل متناظر در بلوک مرجع کاندید مقایسه می شود موازی سازی نمود. توجه کنید که ME مبتنی بر SAD معمولاً در MPEG-1/2 و H/263 استفاده می شود.

الگوریتم های کدگذاری ویدئویی اخیر از طرف دیگر، می تواند ME بهینه سازی شده (RD) دارای اعوجاج سرعت را به کارگیرد که اعوجاج و نرخ انتخاب بهترین کاندید را در نظر گیرد. به طور مثال، یک مقیاس رایج، مجموع وزندهی شده SAD (بین بلوک کنونی و بلوک کاندید است) و نرخ کدگذاری بردارهای حرکت است (MVها). در استاندارد H.264، کدگذاری پیش بینی گر برای کدگذاری MV در بلوک کنونی استفاده می شود و پیش بینی گر، میانه MVها در بلوک های باقیمانده مجاور، بالا و سمت راست بالا است. بنابراین، در ME بهینه سازی شده RD، MVهای بلوک ها همسایه باید ابتدائاً تعیین شوند. بنابراین بر اساس میانه MVهای همسایه، نرخ کدگذاری MVکنونی میتواند تعیین شوند و هزینه بلوک کنونی می تواند در تطبیق بلوک محاسبه شود. چنین وابستگی، استفاده از GPUها برای ME بهینه سازی شده RD را مشکل می سازد. ما طرح های نمونه را برای پرداختن به این موضوع بررسی خواهیم کرد.

1) تخمین حرکت مبتنی بر GPU در باز کردن حلقه: به منظور افزایش درجه موازی سازی، [20] باز کردن حلقه محاسباتی در جستجوی کامل مبتنی بر SAD پیشنهاد می شود. حلقه محاسبه ME در شکل 3 نشان داده شده است و باز کردن حلقه ممکن است زیرا هیچ وابستگی بین بلوک های کلان (MBها) وجود ندارد که در آن SAD به عنوان مقیاسی برای تطبیق استفاده می شود. به علت محدودیت منبع در GPU های اولیه، الگوریتم [20] باید در دو پاس جداگانه پارتیشن بندی شود به طوری که حافظه GPU می تواند دستورالعمل ها را جاسازی نماید.

آزمایشات در [20] در مقایسه به ME جستجوی کامل روی یک Intel Pentium 4 3.0GHz CPU و روی NVIDIA GeForce 6800 GT GPU و نتایج پیشنهاد می دهند که ME مبتنی بر GPU می توانند به دو برابر و14 برابر تسریع برای ME نیم واحد و واحد عدد صحیح به دست آیند. بهبود قابل توجه در ME نیم واحد ناشی از این حقیقت است که [20] از حمایت سخت افزاری داخلی در GPU برای درون یابی استفاده می کند.

توجه داشته باشید که با باز کردن حلقه، امکان زمانبندی تعداد انبوهی از شیارهای موازی (تحت محدودیت وسیله) وجود دارد. یک مثال را برای انتساب یک شیار برای محاسبه یک SAD بین یک MB و بلوک کاندید در پنجره جستجو در نظر بگیرید. بنابراین در مورد جستجوی کامل، تعداد شیارهای مستقل می تواند به اندازه تعداد دفعات MBها و تعداد بلوک های نماینده در هر MB باشد (اندازه پنجره جستجو). برای ویدئوهای HD 720P (بلوک های کلان 1280 * 720 * 3600 در هر فریم). و گستره جستجوی 64 (129 * 129 اندازه پنجره جستجو)، تعداد شیارها می تواند به اندازه 3600 * 129 * 129 = 59907600 باشد.

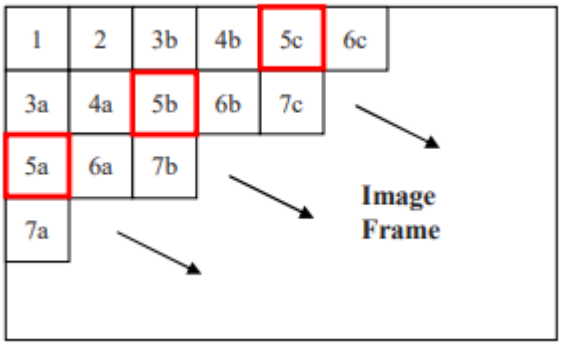
اگر چه جستجوی کامل عمدتاً موازی است، ممکن است به دلیل نیاز محاسباتی سنگین به ویژه برای محتوای ویدیو HD تنها سود عملی کمی داشته باشد. علاوه بر این، هنگامی که MBها به طور مستقل پردازش می شوند و MVها به صورت همزمان در شیارهای مختلف محاسبه می شوند، استفاده از پیش بینی بردار حرکت دشوار می شود که در آن MVها از بلوک های همسایه برای مقداردهی اولیه جستجوی فعلی MB استفاده می شود و این زمانی که پنجره جستجو کوچک است ممکن است روی عملکرد ME تاثیر می گذارد. در بخش چهارم ما در مورد پیاده سازی GPU از ME سریع بحث می کنیم که می تواند به طور کلی به عملکرد قابل مقایسه کدگذاری به عنوان جستجو کامل با تعداد بسیار کمتری از محاسبات دستیابی پیدا کند [42].

```
Loop (rows of macroblocks) {
    Loop (columns of macroblocks) {
        Loop (rows of search range) {
            Loop (columns of search range) {
                SAD computation;
                SAD comparison;
            }
        }
    }
}
```

شکل 3. شبه کد ME عدد صحیح- PEL معمولی بر اساس SAD .

2) برآورد حرکت مبتنی بر GPU بر اساس چینش دوباره مرتبه کدگذاری: با توجه به وابستگی بین بلوک های مجاور به صورت بحث شده، RD ME- بهینه سازی شده معمولا به کار گرفته شده در استانداردهای کدگذاری اخیر ویدئوها را نمی توان به سادگی توسط باز کردن حلقه موازی نمود. در [21] ، [22] ، بازآرایی مرتبه کدگذاری، برای افزایش درجه موازی ارائه شده است. در این الگوریتم ، به جای پردازش بلوک ها در نظم مرسوم اسکن-شطرنجی، بلوک ها در امتداد جهت مورب برای حل مشکل وابستگی پردازش می شوند. این مورد در شکل 4 برای مورد 4 * 4 ME نشان داده شده است. در نظم کدگذاری پیشنهادی آنها، در هر تکرار ، ME، تمام بلوک هایی که بلوک همسایه (سمت چپ، بالا و راست-بالا) پردازش نموده اند را پردازش می کند . یعنی، ME در هر تکرار تمام بلوک هایی که MVهای همسایه محاسبه نموده اند را پردازش می کند و پیش بینی کننده های متوسط در دسترس هستند. توسط بلوک های پردازش در امتداد جهت مورب بازآرایی پیشنهادی به طور قابل ملاحظه ای می توان درجه موازی سازی را افزایش داد. به عنوان مثال، [22] گزارش نموده که حداکثر درجه موازی سازی به ترتیب می تواند تا 44 ، 160 و 240 برای CIF، p720 و ویدیو p1080 باشد. توجه داشته باشید که برای هر بلوک 4* 4، نقاط جستجوی منحصر به فرد در پنجره جستجو را می توان به صورت موازی مورد بررسی قرار داد (در موارد جستجوی کامل و یا برخی از جستجوهای سریع با نمونه برداری منظم از پنجره جستجو) . بنابراین، با موازی سازی بلوک در سطح 240 (به عنوان مثال ، 240 4 *4 بلوک در فریم فعلی را می توان به صورت موازی پردازش نمود) و محدوده جستجوی 64 (129 *129 اندازه پنجره جستجو)، در اصل 240* 129 *129 = 3993840 شیار مستقل را می توان به طور همزمان راه اندازی نمود. موازی سازی سطح پیکسل نیز می تواند، به عنوان مثال ، با تجزیه محاسبه SAD به

شیارهای مختلف پیاده سازی شود. نتایج به دست آمده در [22] نشان می دهد که بیش از 40 برابر سرعت را می توان در یک سیستم با پردازنده اینتل پنتیوم 4 GHz3.2 پردازنده و پردازنده گرافیکی جیفورس GTS 8800 NVIDIA به دست آورد. توجه داشته باشید که ریزپردازنده ی پنتیوم 4 نسبتا در مقایسه با پردازنده های اخیر آهسته است. همچنین کد برنامه بر روی پنتیوم ممکن است به خوبی بهینه سازی شود. بنابراین تسریع گزارش شده در [22] می تواند بالاتر از نسبت به پیاده سازی های CPU کارآمدتر WRT باشد. با این حال، نتایج به دست آمده نشان می دهد که ME بهینه سازی شده -RD را می توان بر روی GPU پیاده سازی نمود.



شکل4. مرتبه کدگذاری بلوک پیشنهاد شده در [22] برای H.26 ME 4*4 4 RD-بهینه سازی شده. هر مربع نشان دهنده 4* 4 بلوک است. بلوک ها با همان تعداد (به عنوان مثال، A5، 5B و C5) به صورت موازی پردازش می شوند.

B. اعوجاج-سرعت بهینه سازی شده در تصمیم گیری مد روی CPUها

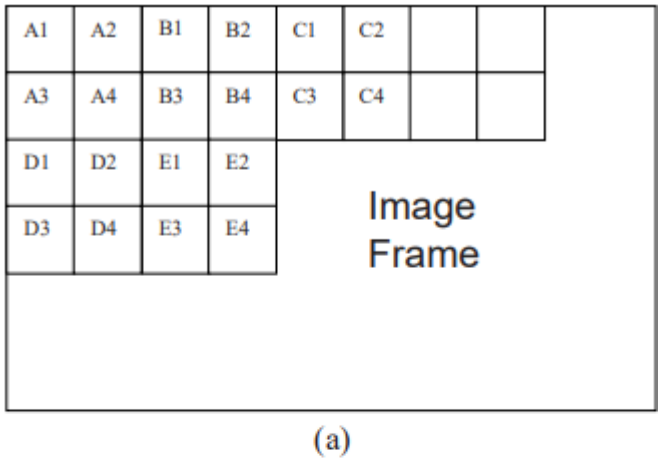
الگوریتم های کدگذاری اخیر ویدئوها، از انتخابی حالت داخلی RD بهینه سازی شده برای تعیین جهت پیش بینی داخلی مطلوب استفاده می نمایند. در این روش ها، کدگذار، هزینه های لاگرانژ را در تمام حالت های پیش بینی شده کاندید محاسبه و حالت پیش بینی را که کمترین هزینه را دارد انتخاب می کند. هزینه های لاگرانژی می تواند مجموع وزنی از مجموع تفاوت های مربعی (SSD) بین بلوک اصلی و بازسازی شده و سرعت کدگذاری برای هدر و بلوک کوانتیزه باقیمانده باقی مانده باشد. محاسبه هزینه حالت کاندید، می تواند شامل محاسبه باقی مانده داخل

پیش بینی، تحول و کوانتیزاسیون در باقی مانده پیش بینی، کوانتیزاسیون معکوس و تحول معکوس، و کدگذاری آنتروپی برای ضرایب تبدیل کوانتیزاسیونی باشد. بنابراین، پیچیدگی محاسباتی انتخاب حالت داخلی بهینه سازی شده RD می تواند بسیار قابل توجه باشد [43] - [45].

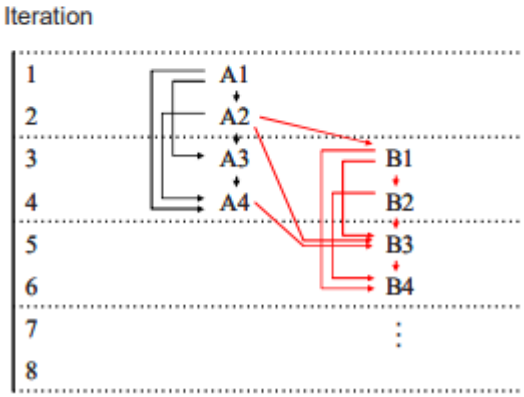
دستیابی به موازی سازی عظیم در تصمیم گیری داخلی RD بهینه سازی شده درون را می توان به چالش کشید. این به این علت است که در درون پیش بینی، پیکسل های بازسازی شده از بلوک های مجاور برای محاسبه نمونه های مرجع استفاده می شوند .بنابراین، حالت های پیش بینی شده داخل از بلوک های همسایه باید برای اولین بار مشخص شوند، و این بلوک می تواند کدگذاری شود و متعاقباً بازسازی شود.بنابراین، حالات مختلف کاندید بلوک کنونی می تواند بر اساس پیکسل های بازسازی شده در بلوک های همسایه مورد بررسی قرار گیرد. چنین وابستگی مانع موازی سازی RD بهینه سازی شده در تصمیم گیری داخلی برای پیاده سازی GPU می شود.

برای پرداختن به موضوع وابستگی، کار قبلی، استراتژی های مختلفی را برای تغییر نظم پردازش بلوک پیشنهاد کرده است [26] [27] [46] . به طور خاص، [27] به تجزیه و تحلیل محدودیت وابستگی می پردازد و پردازش بلوک ها را پس از یک استراتژی حریص پیشنهاد می دهد: در هر تکرار، کدگذار تمام بلوک هایی که بلوک های پدر و مادر کدگذاری نموده است را پردازش می نماید (در گراف وابستگی، بلوک A، بلوک پدر و مادر از بلوک بلوک B است اگر بلوک B نیاز به پیکسل های بازسازی شده از بلوک A تحت حالت های مختلف پیش بینی کاندید داشته باشد). همچنین، در استراتژی حریص، یک بلوک ویدئویی برای پردازش بلافاصله بعد از اینکه تمام بلوک های پدر و مادر آن پردازش شدند، برنامه ریزی زمانی خواهد شد. شکل 5 نشان دهنده محدودیت وابستگی در H.26 4*4 پیش بینی داخل و برنامه ریزی زمانی تحت این استراتژی حریص است. [27] استدلال می کند که این استراتژی حریص برای H.264 و کدگذاری AVS بهینه است: تحت محدودیت های خاص اعمال شده توسط H.264/AVS، و در میان تمام مراتب پیروی کننده از محدودیت ها، نظم کدگذاری مبتنی بر حریص مستلزم حداقل تعداد تکرار برای پردازش تمام بلوک ها است. نتایج شبیه سازی نشان می دهد که استفاده از استراتژی حریص ، تصمیم گیری حالت درونی مبتنی بر GPU می تواند در مورد رسیدن به دو برابر افزایش سرعت در یک سیستم با یک اینتل پنتیوم 4 پردازنده

GHz3.2 و یک جیفورس GTS 8800 پردازنده گرافیکی NVIDIA (جدول) به دست آید. با توجه به [27] ، موازی سازی متوسط برای فیلم های P1080 حدود 127 است، و به نظر می رسد افزایش سرعت دو برابر با نتایج ما در بخش IV برای برآورد حرکت سریع موافق باشد.



(a)



(b)

شکل 5. (a) نمادگذاری برای گراف وابستگی: هر بلوک مربوط به یک بلوک 4 ❋4 است. (b) گراف وابستگی در هنگام پردازش قاب تصویر در انتخاب حالت داخل RD H.264 بهینه سازی شده. هر گره نشان دهنده یک بلوک 4❋4 است (نگاه کنید به شکل 5 (a) برای نمادگذاری). یک لبه هدایت شده برای عبور از بلوک A (گره پدر و مادر) به بلوک B (گره کودک) نشان می دهد که بلوک B نیاز به پیکسل های بازسازی شده از بلوک A دارد تا تعیین هزینه های RD از حالت های مختلف پیش بینی کاندید صورت گیرد. این گراف پس از استراتژی حریص پیشنهاد شده در [27] پردازش می شود و این رقم نشان دهنده تکرار است که در آن هر بلوک پردازش می شود.

C. جبران حرکت بر روی GPUها

جبران حرکت مبتنی بر GPU (MC) توسط [10] و [29] برای Windows Media video (WMV) و کدگشایی ویدئویی H.264 شده است. جبران حرکت نیاز به مقدار زیادی از محاسبات دارد، از اینرو استانداردهای کدگذاری تصویری، حرکت بردارها به نقطه ای برای موقعیت یابی پیکسل فرعی را میسر می سازد (به عنوان مثال، نیم pel و یا quarterpel) و درون یابی فشرده پیکسل برای تولید نمونه های پیش بینی در جابه جایی حرکت با مقادیر کسری لازم خواهد بود. برای مثال، در H.264، یک نمونه نیمه pel از شش نمونه دیگر با استفاده از فیلتر درون یابی شش حالته تولید می شود. و ممکن است تولید یک نمونه چهار PEL به درون یابی خطی اضافی نیاز داشته باشد.

کار [10] در مورد تکنیک های بررسی با مسئله سرریز و گرد کردن در درون یابی در MC بوجود می آید. توجه داشته باشید که MC را می توان همسان نمود، زیرا هر بلوک را می توان به طور مستقل با استفاده از اطلاعات بردار حرکت آن پردازش نمود، و این مورد توسط یک خط لوله از روشهای سایه زن راس / پیکسل در [10] انجام می شود. در اجرای GPU آنها، آنها از روش MULTIPASS استفاده می کنند که دسته باقیمانده ها و پارامتر کنترل گرد کردن را در یک پاس جداگانه برای جلوگیری از سرریز هدایت می کند در حالی که دقت حفظ می شود. علاوه بر این، [10] بحث می کند که چگونه ماژول های مختلف در کدگشایی تصویری را می توان بین CPU و GPU تقسیم نمود، و چگونه محاسبه CPU می تواند به طور حداکثر با محاسبات GPU همپوشانی شود (این مورد بیشتر مورد بحث قرار خواهد گرفت).

	QP= 28	QP= 36	QP= 44
CIF:			
flower_cif	1.14	1.12	1.14
paris_cif	1.12	1.14	1.12
mobile_cif	1.14	1.12	1.12
Average (CIF)	1.13	1.13	1.13
1280 × 720:			
crew	1.38	1.40	1.37
night	1.49	1.42	1.39
city	1.48	1.47	1.43
Average (1280 × 720)	1.45	1.43	1.39
1920 × 1080:			
blue_sky	1.90	1.82	1.73
riverbed	1.93	1.82	1.76
station	1.89	1.81	1.80
Average (1920 × 1080)	1.91	1.82	1.76

جدول IIامقایسه بین پیش بینی داخبی H.264 موازی روی GPU پیشنهاد شده در [27] و پیش بینی داخلی H.264 مرسوم روی CPU. اعداد، نسبت های زمان اجرای CPU به زمان اجرای GPU هستند. توجه داشته باشید که زمان اجرای GPU شامل تمام سربار انتقال داده ها می شود.

نتایج شبیه سازی نشان می دهد که در یک سیستم با اینتل پنتیوم III 667MHz GPU و یک NVIDIA GeForce3 Ti200 GPU ، با اعمال نفوذ به GPU سیستم می توان به بیش از سه برابر سرعت رسید و ممکن است به کدگشایی زمان واقعی WMV (نسخه 8) برای ویدیو با تعریف فوق تا حدود رزولوشن تا 1280 * 720 [10] رسید.

D. پارتیشن بندی وظیفه بین CPU و GPU

برای به دست آوردن عملکرد سیستم رقابتی، CPU و GPU باید با هم برای کدگذاری / کدگشایی در نظر گرفته شوند. با این حال، بررسی پارتیشن بندی بهینه از وظایف محاسبات بین CPU و GPU، می تواند بسیار درگیرکننده باشد و نیاز به ارزیابی جدی در بسیاری از مسائل دارد. برای مثال:

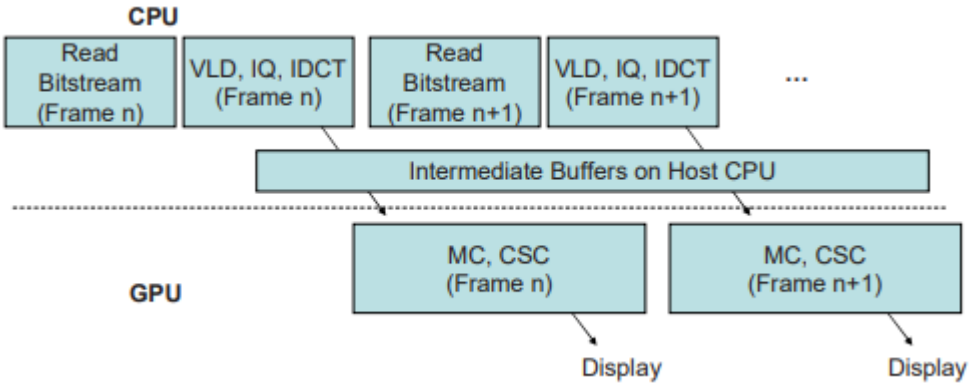
* لازم است بررسی شود که چگونه تخصیص وظایف صورت می گیرد به طوری که محاسبه GPU می تواند با محاسبات پردازنده تا آنجا که ممکن است همپوشانی داشته باشد و در نتیجه حداکثر پردازش موازی به دست می آید.

* از آنجا که پهنای باند بین حافظه GPU و حافظه اصلی می تواند آهسته باشد، بررسی چگونگی به حداقل رساندن انتقال داده ها بین حافظه اصلی و حافظه GPU مهم است.

* همچنین، مطالعه و ارزیابی اینکه کدام ماژول ها در جریان کدگذاری / کدگشایی می توانند به طور موثر به برای GPU بارگذاری شوند، در حالی که دیگران می توانند بر روی پردازنده اجرا شوند، مهم است.

تمرکز بر روی کدگشایی WMV، [10]، یک استراتژی پارتیشن بندی را که در آن کل حلقه بازخورد، از جمله جبران حرکت و تبدیل فضای رنگی (CSC)، به GPU داده می شود پیشنهاد می کند. با انجام این کار، آنها می توانند از انتقال داده ها از GPU به CPU جلوگیری نمایند. از آنجا که خواندن و پشتیبانی از حافظه GPU به حافظه اصلی ممکن است به دلیل اجرای مشترک نامتقارن باس حافظه [10] آرام باشد، از جمله خواندن پشتیبانی باید به حداقل برسد. شکل 6 استراتژی پارتیشن را به تصویر می کشد. توجه داشته باشید که در حالی که GPU در حال انجام MC و CSC قاب n است، CPU کدگشایی طول متغیر (VLD)، کوانتیزاسیون معکوس (IQ) و معکوس DCT (IDCT) از قاب n +1 را انجام می دهد. همچنین توجه داشته باشید که بافر میانه حافظه بین CPU و GPU برای جذب شوک ها در زمان پردازش CPU / GPU استفاده می شود. نتایج شبیه سازی در [10] نشان دهنده اندازه بافر میانی از چهار فریم است که می تواند به طور قابل ملاحظه ای سرعت کلی کدگشایی را بهبود بخشد.

در حالی که [11] برخی از مسائل (به عنوان مثال، پهنای باند مورد نیاز) را در مورد تخلیه برآورد حرکت به GPU مورد بحث قرار داده است، به نظر می رسد هیچ کار قبلی در تحقیقات دقیق در مورد چگونگی تقسیم کدگذاری فایلهای ویدئویی بین CPU و GPU وجود ندارد. ما خاطر نشان می کنیم که اجرای چندین ماژول کدگذاری مهم GPU (از جمله برآورد حرکت، تصمیم گیری درون حالت، جبران حرکت و تبدیل) در گذشته مورد بررسی قرار گرفته اند، در حالی که رفع انسداد و کدگذاری آنتروپی نیاز به تحقیقات بیشتری دارد.



شکل 6. پارتیشن بندی در کدگشایی WMV پیشنهاد شده توسط [10].

4. مطالعه موردی: تخمین حرکت FAST بر اساس GPU

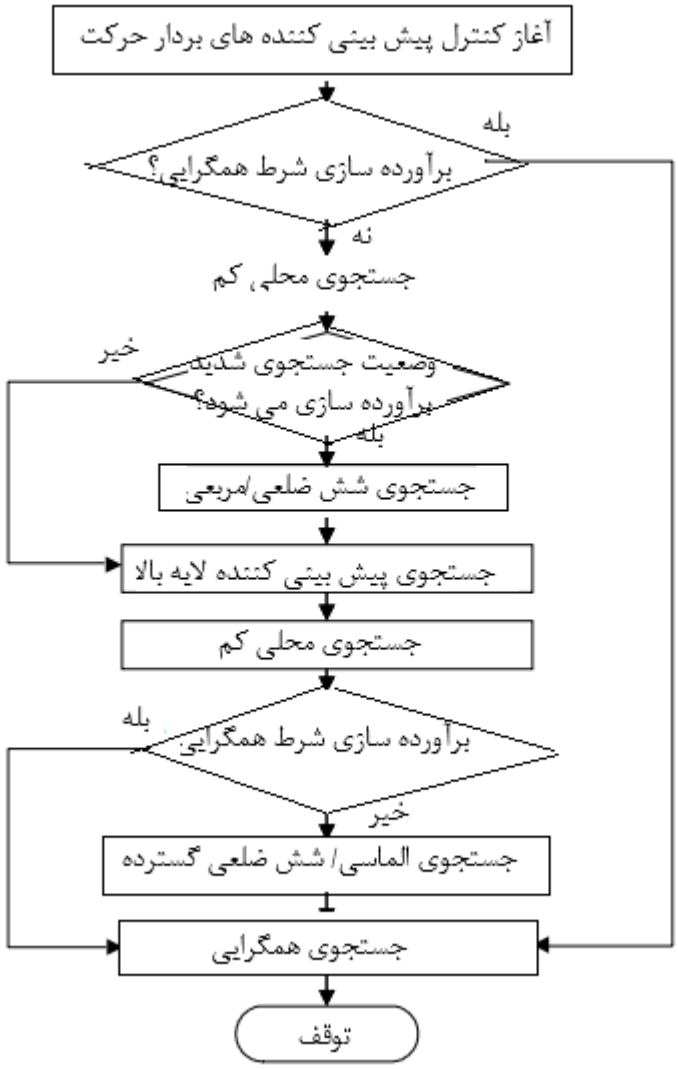
برای نشان دادن برخی از ملاحظات طراحی در استفاده از GPUها برای کدگذاری های ویدئویی، ما به تفصیل در این بخش ME سریع مبتنی بر GPU را مورد بحث قرار می دهیم (کد ME GPU توسط نویسندگان بر اساس JM نرم افزار H.264 14.2 مرجع توسعه داده شد). تمرکز روی چگونگی رسیدگی به وابستگی داده ها در الگوریتم برای بهره برداری از قابلیت پردازش موازی GPUها متمرکز، و چگونگی افزایش سرعت تبادل عملکرد با نرخ اعوجاج (RD) است.

A. برآورد حرکت سریع

اجرای GPU ما در مورد ME سریع بر اساس جستجوی چند شش گوش ساده نامتقارن (smpUMHexagonS ها) [42] است که یکی از الگوریتم های ME سریع اتخاذ شده توسط نرم افزار JM H.264 مرجع می باشد. ما smpUMHexagonS را به این دلیل انتخاب می کنیم که می تواند معاوضه بسیار خوبی را بین پیچیدگی محاسباتی و بهره وری کدگذاری به دست آورد. برای مثال، در پنتیوم 4 پردازنده smpUMHexagonS این می تواند کاهش 94٪ در زمان اجرای Me با راندمان RD قابل مقایسه، در زمان مقایسه با جستجوی سریع کامل در نرم افزار JM گزارش شد. علاوه بر این، smpUMHexagonS کاملا جمع و جور است، بنابراین می تواند محدودیت

حافظه GPU را تامین کند. در پیاده سازی ما، تمام هسته های GPU که با برآورد صحیح واحد ارتباط دارد حدود 600 خط کد دارد.

شکل 7 نمودار جریان smpUMHexagonS را به تصویر می کشد. برای هر ماکروبلاک smpUMHexagonSها، MVها را برای تمام پارتیشن های ماکروبلاک (16 *16، 16* 8، ... 4 * 4) محاسبه می کند. MVها با به حداقل رساندن هزینه لاگرانژ D + R انتخاب می شوند که در آن D SAD بین بلوک در حال حاضر و کاندید قرار دارد، و R نرخ بیت برای کدگذاری MV است. در smpUMHexagonS، کاهش محاسبات به طور عمده توسط نمونه برداری فضای جستجو عاقلانه، و با استفاده از تکنیک های مختلف از جمله پیش بینی حرکت بردار ، الگوهای جستجوی مختلف (متقابل ، شش گوش ، الماس) و ختم زود هنگام به دست می آید. به طور خاص، بردارهای حرکت از بلوک های فضایی مجاور و از پارتیشن های دیگر ماکروبلاک برای مقداردهی اولیه جستجو برای پارتیشن جاری مورد استفاده قرار می گیرد. توجه داشته باشید که همانطور که در شکل 7 نشان داده شده است، smpUMHexagonS از آزمایشات مختلفی استفاده می کند برای تعیین اینکه آیا جستجو (پارتیشن) را می توان بر اساس حداقل هزینه تا کنون محاسبه شده خاتمه داد یا نه. به عنوان یک نتیجه، ماکروهای مختلف با محتویات مختلف ممکن است تحت مسیرهای مختلف پردازش (که به نوعی در بسیاری از ME سریع الگوریتم [47]) قرار گیرد و این ممکن است عملکرد اجرای GPU را تحت تاثیر قرار دهد.



شکل 7. در برآورد حرکت سریع با استفاده از smpUMHexagonS [42] . شکل این مراحل را برای جستجوی عدد صحیح واحد برای پارتیشن ماکروبلاک به تصویر می کشد.

B. اجرای GPU با استفاده از کاشی کاری

برای استفاده از موازی سازی در GPU، ما فریم فعلی را به کاشی های متعدد پارتیشن بندی می کنیم و هر کاشی شامل K (ارتفاع) L (عرض) MBها می شود. برای مثال، در شکل 8 موردی با K = 4 = 1، L را به تصویر می کشد. هر کاشی توسط یک موضوع واحد GPU پردازش می شود، به عنوان مثال، هر شیار، K * L MBها را در

کاشی های پی در پی توسط کاشی های متفاوت با موضوعات مختلف مستقل به طور همزمان بر روی GPU پردازش می شود.

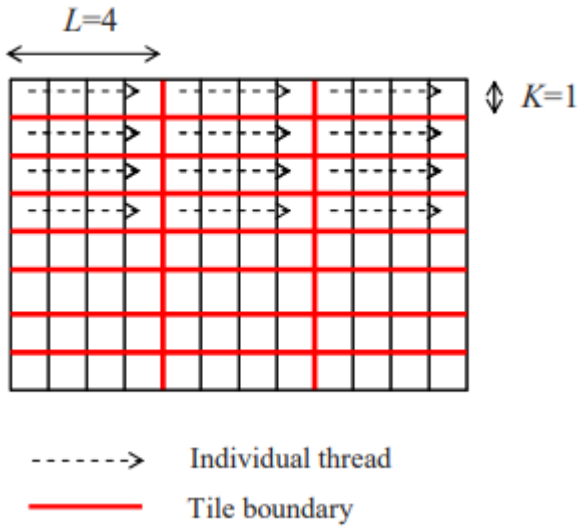
پس از بحث در بخش IV -A ، MBهای فردی تحت smpUMHexagonS مستقل نیستند. به طور خاص، با روش های زیر ماکرو به همسایگان خود بستگی دارد:

* اول، برای محاسبه عبارت نرخ R در هزینه های لاگرانژی از بردارهای حرکت MBهای همسایه مورد نیاز است. اگر MB همسایه متعلق به کاشی دیگری باشد، فرض می کنیم بردار حرکت آن در محاسبات R برابر صفر است. بنابراین ، با کاشی کاری، هزینه لاگرانژی محاسبه شده بسیار دقیق نیست و بردارهای حرکت کمتر از حد مطلوب می تواند توسط smpUMHexagonS به عنوان یک نتیجه انتخاب شود. تاثیر کاشی کاری در این مورد به مقدار * و از این رو نرخ بیت بستگی دارد. برای برنامه های کاربردی نرخ بیت پایین (نرخ محدود)، کدگذاری ها روی نرخ بازده تمرکز بیشتری دارند و λ بزرگ انتخاب می شود و عبارت نرخ بر هزینه های لاگرانژی غالب خواهد بود [48]. در نتیجه کاشی کاری باید دارای بارزتر تأثیر منفی بر روی عملکرد smpUMHexagonS در برنامه های کاربردی نرخ بیت پایین (ازاینرو کاشی کاری روی عبارت نرخ تاثیر می گذارد) باشد.

* دوم، smpUMHexagonS ها (و بسیاری دیگر از ME سریع [47]) از پیش بینی بردار حرکت استفاده می کنند، به عنوان مثال، بردارهای حرکت از MBهای همسایه برای راه اندازی جستجو مورد استفاده قرار می گیرند. بر اساس کاشی کاری، برخی از اطلاعات در مورد بردارهای حرکتی همسایه در دسترس نیستند، و این ممکن است به نقاط جستجوی اولیه با کیفیت ضعیف منجر شود و بردارهای حرکت کمتر از حد مطلوب ممکن است در پایان جستجو (از این رو عملکرد RD است به خطر بیافتد) انتخاب شوند. علاوه بر این، از آنجا که smpUMHexagonS ختم زودرس را به کار می گیرد، نقاط ضعیف اولیه نیز ممکن است به مدت زمان طولانی تر پردازش منجر شود، همانطور که نقاط جستجوی بیشتری را باید مورد بررسی قرار گیرد تا هزینه به اندازه کافی برای پایان دادن به جستجو کوچک باشد (به عنوان مثال، ما حدود 4 درصد افزایش را در پردازش ME زمانی که کدگذاری HD 720P

بندرگاه توالی با استفاده از کاشی کاری K = 1؛ L = 1 در smpUMHexagonS متوالی صورت می گیرد، مشاهده نمودیم).

بررسی های بالا برای بسیاری از الگوریتم های ME سریع نیز قابل کاربرد هستند. توجه داشته باشید که در شبیه سازی ما، کاشی کاری فقط در ME برای تسهیل محاسبات GPU استفاده می شود و بقیه اقدامات کدگذاری به همان شیوه نرم افزار مرجع صورت می گیرد. بنابراین، کاشی کاری ما متفاوت از دیگر ایده های پارتیشن بندی مانند برش است [47]، که در آن پارتیشن های فردی به طور مستقل در بسیاری از کدگذاری ها در نظر گرفته می شوند.



شکل 8. برآورد حرکت سریع مبتنی بر GPU: فریم فعلی به کاشی های متعدد تقسیم می شود تا پردازش موازی در ME تسهیل شود. در اینجا هر مربع نشان دهنده یک ماکروبلاک است.

C. آزمایش

برای بررسی عملکرد سریع ME سریع بر اساس GPUبا استفاده از کاشی کار ، ما آزمایشاتی را بر روی رایانه های شخصی مجهز به یک GTS 8800 از PCIe کارت گرافیک جیفورس با 96 پردازنده جریان [15]، و یک پردازنده اینتل Core 2 پشتیبانی از چهار Q9400 2.66 گیگاهرتز CPU 3.23 GB RAM انجام دادیم. ما از NVIDIA CUDA [39] برای اجرای کد GPU استفاده نمودیم. ما CUDA را صرفا به دلیل در دسترس بودن دستگاه

NVIDIA در آزمایشگاه خود انتخاب نمودیم و به ما خاطر نشان نمودیم که دیگر مدل های کدگذاری GPU به خوبی طراحی شده مانند [ATI CTM [49]، Stream Computing SDK و Brook + [50] وجود دارند.

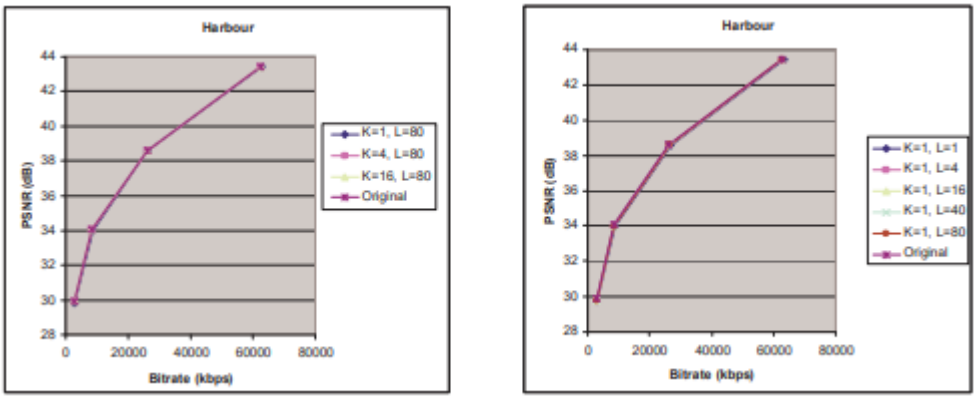
ما برای اولین بار چگونگی تاثیر کاشی کاری روی عملکرد RD را ارزیابی نمودیم. ما از JM 14.2 برای کدگذاری توالی HD 720P استفاده نمودیم (1280 720، 60 فریم در هر ثانیه)، خدمه، شهر، بندرگاه و شب (به دلیل نیاز محاسباتی بالا، و به دلیل علاقه رو به رشد در مورد محتویات HD، ما روی کدگذاری فیلم HD تمرکز می کنیم). ما از H.264 با مشخصات بالا با دامنه جستجوی 64 استفاده نمودیم. همه عکسها به صورت فریم های P- کدگذاری می شوند به جز قاب اولیه I. شکل 9 نشان دهنده عملکرد RD با اندازه های مختلف کاشی برای دنباله Harbour است. همانطور که در شکل نشان داده شده است تاثیر کاشی کاری در این مورد تا زمانی کوچک است که اندازه کاشی پایین K = 1؛ L = 1 است، زمانی که تنزل نسبت به نرم افزار اصلی مرجع حدود 0.2 دسی بل است (با smpUMHexagonS) . جدول III نشان دهنده متوسط PSNR تنزل و متوسط افزایش در نرخ بیتی با استفاده از اندازه های مختلف کاشی است که به ترتیب توسط BDPSNR و BDBR اندازه گیری شده است. توجه داشته باشید که BDPSNR و BDBR غالبا در جامعه استاندارد ویدیویی [51] استفاده می شوند. نتایج نشان می دهند که کاشی کاری ممکن است به متوسط گیری تنزل بین 0.08 دسی بل تا 0.4 دسی بل با اندازه کاشی K = 1 L = 1 برای این توالی منجر شود.

سپس ما بحث می کنیم که چگونه کاشی کاری ممکن است روی تسریع تاثیر بگذارد. جدول IV نشان دهنده زمان اجرای GPU (در عدد صحیح ME pel) با اندازه های مختلف کاشی و شکل 10 نشان دهنده افزایش سرعت بین اجرای GPU (با کاشی کاری و با استفاده از پردازش موازی در چند هسته ای) و از اجرای ترتیبی CPU (بدون کاشی کاری و با استفاده از پردازش متوالی در یک هسته واحد) است. مقایسه با کد برنامه موازی در هسته های CPU چند بعدی مورد بحث قرار خواهد گرفت. زمان اجرای GPU شامل سرباری برای انتقال فریم های ویدئویی از حافظه سیستم به حافظه GPU می شود. بهینه سازی کامپایلر برای هر دو برنامه GPU و CPU برنامه استفاده می شود. با این حال، هر دو کد GPU / CPU دارای فضاهایی برای بهبود سرعت بیشتر هستند. به طور خاص، کد

GPU، داده های پیکسل را در حافظه کلی (حافظه خارج از تراشه) ذخیره می کند که دارای تاخیر دسترسی قابل توجهی است [39] . زمانی که برآورد حرکت نسبتا فشرده حافظه دسترسی (محاسبه SAD تنها سه عملیات ریاضی را در هر دو بار حافظه انجام می هد که شدت محاسباتی 1.5 ارائه می دهد که برای GPU محاسبات [37] کوچک است)، از جمله تاخیر ممکن است عملکرد کد GPU را تحت تاثیر قرار دهد. بنابراین، این کد را می توان با استفاده صحیح از حافظه داخلی (حافظه بر روی تراشه) [37] بهبود داد. همانطور که در شکل 10 نشان داده شده است، تسریع با حجم کوچکتر کاشی افزایش می یابد زمانی که شیارهای مستقل تر می توانند برنامه ریزی شوند. این در کد GPU حاضر برای مخفی کردن تاخیر دسترسی به حافظه بسیار مهم است. همچنین توجه داشته باشید که توالی های مختلف دارای زمان های مختلف اجرای GPU و تسریعات هستند همانطور که محتویات ویدئویی مختلف ممکن است منجر به مسیرهای مختلف اجرا در smpUMHexagonS و مقادیر مختلف مجازات های متحمل شده توسط ترتیب اجرا شود. شکل 10 نشان می دهد که سرعت های رهاسازی 1:5 به 3:5 را می توان در smpUMHexagonS واحد عدد صحیح در این دنباله با استفاده از اندازه کاشی K = 1 ؛ L = 1 به دست آورد.

شکل 11 افزایش سرعت بین اجرای GPU وپیاده سازی پردازنده موازی با استفاده ازچهار هسته پردازنده اینتل Core 2 Quad را نشان می دهد. برای رسیدن به پردازش CPU موازی، قاب فعلی به چهار کاشی از تعداد مساوی از ردیف MB (یعنی L = عرض قاب ویدئو در MB، K = ارتفاع قاب ویدئو در MB / 4) تقسیم می شود و هر کاشی توسط یک شیار مستقل در حال اجرا بر روی یک هسته پردازنده پردازش می شود. ما از OpenMP برای پیاده سازی برنامه های CPU موازی استفاده می کنیم [52]. مشاهده نمودیم که موازی سازی، زمان اجرای CPU را حدوداً توسط یک فاکتور سه کاهش می دهد. توجه داشته باشید که حداکثر تسریع نظری چهار نمی تواند این استراتژی موازی سازی را حاصل نماید، همانطور که smpUMHexagonS می تواند زمان اجرای متفاوتی را بر روی هر مگابایت صرف نماید و متعادل کردن بار بهینه را نمی توان با کاشی کاری ساده نمی توان به دست آورد. شکل 11 نشان می دهد که زمان اجرای پیاده سازی GPU و پیاده سازی موازی CPU می تواند در برخی از موارد

قابل مقایسه باشد (در حالی که پیاده سازی GPU متحمل برخی از تنزل های عملکردی RD می شود همانطور که در جدول III نشان داده شده است) .



شکل 9.عملکرد بندرگاه RD با اندازه های مختلف کاشی در برآورد حرکت سریع. در اینجا "اصلی" اشاره به نرم افزار مرجع دارد (یعنی، بدون کاشی کاری).

		Crew		City		Harbour		Night	
Tile size	Number of tiles	BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)	BDBR (%)	BDPSNR (dB)
$K=1, L=1$	3600	3.135	-0.082	12.933	-0.407	5.578	-0.221	4.636	-0.17
$K=1, L=4$	900	3.081	-0.079	11.115	-0.352	2.385	-0.094	3.546	-0.13
$K=1, L=16$	225	3.116	-0.08	11.171	-0.35	2.246	-0.089	3.415	-0.125
$K=1, L=40$	90	3.224	-0.083	10.821	-0.339	2.205	-0.087	3.4	-0.124
$K=4, L=80$	12	0.63	-0.016	1.412	-0.044	0.57	-0.022	1.19	-0.043
$K=16, L=80$	3	0.094	-0.003	0.261	-0.008	0.07	-0.003	0.161	-0.006

جدول III معاوضه بین اندازه کاشی و عملکرد RD. متوسط در نرخ بیت و تنزل PSNR به طور متوسط با توجه به نرم افزار مرجع محاسبه می شوند (یعنی، بدون کاشی کاری).

در این آزمایش، ما انتقال سربار را برای یک قاب از CPU به GPU حدود 1.6 میلی ثانیه مشاهده نمودیم، و این در مورد حدود 0.1٪ به 0.2٪ از زمان اجرای عدد صحیح PEL ME (جدول IV) است. به طور کلی، هزینه های سرریز انتقال داده می تواند یک مسئله جدی کمتر در کدگذاری درون قاب در مقایسه با کدگشایی و کدگذاری داخل قاب باشد، زیرا کدگذاری درون قاب به طور کلی نیاز به مقدار قابل توجهی بزرگتر از زمان اجرا دارد.

در نهایت، ما دوست داریم که پیاده سازی CPU و GPU بیشتر بهینه سازی شود. بحث ما پیشنهاد کرده است که رسیدن به اوج عملکرد ارائه شده توسط این دستگاه های چند هسته ای در کدگذاری های ویدئویی، و پژوهش الگوریتم بیشتر و بهینه سازی سطح آموزش مورد نیاز غیر بدیهی است.

5. نتیجه گیری ها و بحث

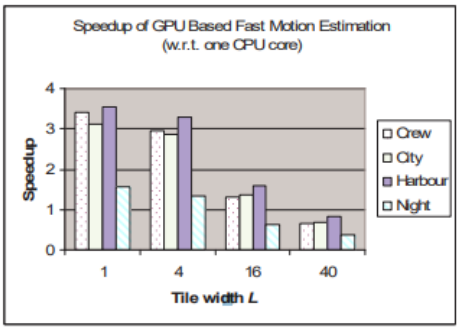
ما کار قبلی در مورد استفاده از GPUها را برای کدگذاری و کدگشایی ویدئو بازنگری نمودیم. به طور خاص، ما بحث نمودی که چگونه برخی از ماژول های کدگذاری ویدیو می توانند به طرق خاص برای افشای موازی داده ها تا حد امکان اجرا شوند، به طوری که قابلیت عظیم پردازش موازی GPUها را می توان به طور کامل مورد استفاده قرار داد. نتایج شبیه سازی در کار قبلی نشان می دهد که پیاده سازی مبتنی بر GPU می توانید تسریع های قابل توجهی برای برخی از مهم ترین ماژول های محاسبات فشرده در کدگذاری های ویدئویی حاصل نماید. بنابراین ، می تواند یک روش مقرون به صرفه برای اهرم قدرت محاسباتی GPUها برای پاسخگویی به احتیاجات پردازش داده ها در کدگذاری های ویدئویی باشد. در حال حاضر نیز مثالی را برای پارتیشن بندی ویدیو جریان کدگشایی بین CPU و GPU برای رسیدن به حداکثر هم تداخل محاسبه شده را مورد بحث قرار دادیم. علاوه بر این، ما برآورد حرکت سریع مبتنی بر GPU را مورد بحث قرار دادیم و سبک سنگین کردن بین تسریع و عملکرد نرخ اعوجاج را مورد بررسی قرار دادیم.

چند مسئله مربوط به پژوهش وجود دارد . اول ، به نظر می رسد که هیچ مطالعه ای در مورد پارتیشن بندی جریان کدگذاری بین CPU و GPU وجود ندارد. دوم، با در دسترس بودن بسیاری از فرمت های ویدیویی مختلف (به عنوان مثال ، SD، HD) و کدگذاری استاندارد نیاز رو به رشدی برای تبدیل کد یک کدگذاری شده به فرمت های تصویری دیگر [53] - [55] دارد. با این حال ، در حالی کهدر برنامه های کاربردی چند تغییر کد تجاری در دسترس وجود دارد [56] ، [57]، به نظر می رسد هیچ کار قبلی در بررسی استفاده بهینه از GPUها برای کدگذاری وجود ندارد. توجه داشته باشید که بر خلاف کدگذاری / کدگشایی ویدئو، هیچ الگوریتم استانداردی برای تغییر کد ویدئوها وجود

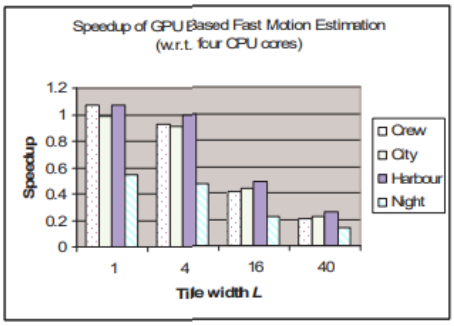
دارد، و بسیاری از روش های که قبلا پیشنهاد شده است که برای رسیدن به طیف گسترده ای از کدگذاری متقابل و با کیفیت با پیچیدگی های مختلف مورد نیاز [53] وجود دارد . این کار، مطالعه تغییر کد مبتنی بر GPU را پیچیده می کند.

Tile width	Number of threads	GPU execution time (ms)			
		Crew	City	Harbour	Night
$L = 1$	3600	835.05	927.32	1248.95	1688.50
$L = 4$	900	959.16	1005.55	1341.45	1975.95
$L = 16$	225	2169.25	2108.71	2763.79	4175.44
$L = 40$	90	4373.63	4165.28	5318.38	6920.73

جدول IV زمان اجرای GPU برای تخمین حرکت INTEGER PEL FAST با عرض های مختلف کاشی (کاشی K ارتفاع برابر با یک است). سربار انتقال داده گنجانده شده است.



شکل 10. معاوضه بین عرض کاشی و افزایش سرعت (K کاشی ارتفاع برابر با یک است) . تسریع نسبت زمان اجرای CPU (کد برنامه متوالی در یک هسته پردازنده) به زمان اجرای GPU (از جمله سربار انتقال داده ها).



شکل 11. معاوضه بین عرض و تسریع کاشی (ارتفاع کاشی K برابر یک است) . تسریع نسبت زمان اجرای CPU به کد برنامه موازی بر (چهار هسته CPU) به زمان اجرای GPU (از جمله سربار انتقال داده ها).

تقدیرات

این کار تا حدودی توسط کمیسیون نوآوری و تکنولوژی (پروژه شماره GHP/048/08) و تحقیقات شورای کمک های بلاعوض (پروژه . RPC07/08.EG22 و پروژه 610،109) از منطقه ویژه اداری هنگ کنگ، پشتیبانی چین حمایت شده است. نویسندگان همچنین می خواهند از سردبیر و داوران ناشناس برای نظرات خود که به به بهبود مقاله به طور قابل توجهی کمک کرد، تشکر نمایند.

REFERENCES

[1] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Prentice Hall, 2002.

[2] T. Wiegand, "Joint final committee draft for joint video specification H.264," Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, Tech. Rep. JVT-D157, 2002.

[3] G. Wen, W. Qiang, and M. Siwei, "Digital audio video coding standard of AVS," *ZTE Communications*, 2006.

[4] L. Yu, F. Yi, J. Dong, and C. Zhang, "Overview of AVS-Video: tools, performance and complexity," in *Proc. Visual Communications and Image Processing (VCIP)*, 2005.

[5] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560 – 576, July 2003.

[6] G. Sullivan, J.-R. Ohm, A. Ortega, E. Delp, A. Vetro, and M. Barni, "dsp Forum - Future of video coding and transmission," *IEEE Signal Processing Magazine*, vol. 23, no. 6, Nov. 2006.

[7] J. Loomis and M. Wasson, "VC-1 technical overview," *http://www.microsoft.com/windows/windowsmedia/*, 2007.

[8] B. G. Haskell, A. Puri, and A. N. Netravali, *Digital Video: An introduction to MPEG-2*. Springer, 1996.

[9] L. Fan, S. Ma, and F. Wu, "An overview of AVS video standard," in *Proc. IEEE International Conference on Multimedia and Expo*, 2004.

[10] G. Shen, G. Gao, S. Li, H. Shum, and Y. Zhang, "Accelerate video decoding with generic GPU," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 685 – 693, 2005.

[11] A. Mather, "GPU-accelerated video encoding," SIGGRAPH Tech Talks, 2008.
[12] J. Kruger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," in *Proc. International Conference on Computer Graphics and Interactive Techniques*, 2005.
[13] O. Fialka and M. Cadik, "FFT and convolution performance in image filtering on GPU," *Information Visualization*, pp. 609 – 614, 2006.
[14] A. Brunton and J. Zhao, "Real-time video watermarking on programmable graphics hardware," in *Proc. Canadian Conference on Electrical and Computer Engineering*, 2005.
[15] NVIDIA, "NVIDIA GeForce 8800 architecture technical brief," NVIDIA, Tech. Rep., 2006.
[16] ——, "CUDA - compute unified device architecture," *http://www.nvidia.com/object/cuda_home.html*, 2009.
[17] Khronos Group, "OpenCL - The open standard for parallel programming of heterogeneous systems," *http://www.khronos.org/opencl/*, 2009.
[18] M. Macedonia, "The GPU enters computing's mainstream," *IEEE Computer*, pp. 106 – 108, 2003.
[19] F. Kelly and A. Kokaram, "General purpose graphics hardware for accelerating motion estimation," in *Proc. Irish Machine Vision and Image Processing Conference*, 2003.
[20] Y. Lin, P. Li, C. Chang, C. Wu, Y. Tsao, and S. Chien, "Multi-pass algorithm of motion estimation in video encoding for generic GPU," in *Proc. IEEE International Symposium of Circuits and Systems*, 2006.
[21] C.-W. Ho, O. Au, G. Chan, S.-K. Yip, and H.-M. Wong, "Motion estimation for H.264/AVC using programmable graphics hardware," in *Proc. IEEE International Conference on Multimedia and Expo*, 2006.
[22] M. Kung, O. Au, P. Wong, and C. Liu, "Block based parallel motion estimation using programmable graphics hardware," in *Proc. International Conference on Audio, Language and Image Processing*, 2008.
[23] M. L. Schmit, R. Meeyakhan Rawther, and R. Giduthuri, "Software video encoder with GPU acceleration," *U.S. Patent Application 20090016430*, 2009.
[24] G. Jin and H.-J. Lee, "A parallel and pipelined execution of H.264/AVC intra prediction," in *Proc. IEEE International Conference on Computer and Information Technology*, 2006.
[25] W. Lee, S. Lee, and J. Kim, "Pipelined intra prediction using shuffled encoding order for H.264/AVC," in *Proc. IEEE Region 10 Conference (TENCON)*, 2006.
[26] M. Kung, O. Au, P. Wong, and C. Liu, "Intra frame encoding using programmable graphics hardware," in *Proc. Pacific Rim Conference on Multimedia (PCM)*, 2007.
[27] N.-M. Cheung, O. Au, M. Kung, H. Wong, and C. Liu, "Highly parallel rate-distortion optimized intra mode decision on multi-core graphics processors," in *IEEE Transactions on Circuits and Systems for Video Technology - Special Issue on Algorithm/Architecture Co-Exploration of Visual Computing*, vol. 19, no. 11, pp. 1692-1703, Nov. 2009.
[28] A. Obukhov and A. Kharlamov, "Discrete cosine transform for 8x8 blocks with CUDA," NVIDIA, Tech. Rep., Oct. 2008.
[29] B. Pieters, D. Van Rijsselbergen, W. De Neve, and R. Van de Walle, "Motion compensation and reconstruction of H.264/AVC video bitstreams using the GPU," in *WIAMIS '07: Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services*. Washington, DC, USA: IEEE Computer Society, 2007, p. 69.
[30] A. Hirvonen and T. Leppanen, "H.263 video decoding on programmable graphics hardware," in *Signal Processing and Information Technology,*