AVid: **A**nnotation Driven **Vid**eo Decoding for Hybrid Memories

Liviu Codrut Stancu, Luis Angel D. Bathen, Nikil Dutt, Alex Nicolau
School of Information and Computer Science, University of California, Irvine
{lstancu,lbathen,dutt,nicolau}@ics.uci.edu

***Abstract*—Adopting emerging non-volatile memory (NVM) technologies is a viable solution to minimize the increasing memory leakage power in today's embedded systems. However, in order to take advantage of the many benefits in NVMs, software must account for their high write overheads. This paper presents *AVid*, an annotation driven video decoding technique for hybrid memory subsystems. *AVid* exploits the physical characteristics of NVMs by extracting video decoder access patterns and uses this meta-information to minimize write overheads, thereby improving energy savings and performance. Our experimental results on an annotation-aware H.264 codec show that our technique is able to achieve execution time and energy reduction by up to 40.8% and 39.7% respectively when applied to H.264 decoding.**

Keywords: Hybrid Memories; Energy Efficiency; H.264; Video Decoding; Software Annotations

I. INTRODUCTION

The ever-increasing demands for media-rich applications (e.g., video streaming), coupled with the need to manage power and energy have motivated the move towards multi-core platforms. While mobile platforms and their desktop counter parts both need to deal with the evolving software stack and new computing substrates, they have very different energy budgets; for mobile platforms there is a critical need for the embedded software stack to account for its tight energy budget. In today's platforms, the power consumption of the whole system is primarily dominated by the memory hierarchy [1]. Moreover, ITRS predicts that over the next decade memory static/leakage power will continue to surpass dynamic power consumption [2][3].

SRAM-based memories are a major source of leakage power in the system [4] as they may consume up to 90% of the total on-chip area [2] and up to 70% of total power consumed by the system [5], which is aggravated by the move towards multi- and many-core platforms. To reduce the static power of the memory subsystem, researchers have proposed the use of Non Volatile Memories (NVMs) as an alternative to SRAM on-chip memories and DRAM off-chip memories [6][7][8]. NVMs (e.g. PCRAMs, MRAMs) have high densities, low leakage power and comparable read latencies and read dynamic power with respect to traditional memories. One major problem with NVMs is the expensive write operation, which incurs high write latency and high write dynamic power. To mitigate the drawbacks of non-volatile memories researchers have proposed hybrid memory hierarchies that consist of a combination of SRAM, DRAM, and NVM technologies. There have been several research efforts focused on deploying hybrid memory hierarchies for caches [6][8][9], on-chip memories [10][11] and main memories [12] [13].

In order to better exploit hybrid memory hierarchies the runtime system needs higher-level information (e.g., compiler, application) to tune the application's memory allocation-decisions based on their characteristics. The runtime system could also collect statistically information at runtime to react to the application's memory usage patterns. This however takes time and may not necessarily predict future access patterns, thereby motivating the need for both offline high level information and online characterization.

Software annotations have been extensively used to guide run-time systems to make smart resource management decisions to increase system performance or minimize power consumption [14][15][16][17]. However, to the best of our knowledge, no previous efforts have focused on software annotations to exploit the features of hybrid memory organizations.

This work proposes ***AVid***, an annotation driven video decoding scheme for hybrid memory subsystems. *AVid* focuses on extracting meta-information from highly predictable, data intensive applications (e.g., video decoding) and exploits this information to make efficient run-time memory allocation decisions that leverage the physical characteristics of nonvolatile memories for energy savings and performance improvements. Our experimental results on an annotation-aware H.264 codec show that our technique is able to achieve execution time and energy reduction by up to 40.8% and 39.7% respectively when applied to H.264 decoding.

The novel contributions of our work are that we:

- Leverage software annotations to opportunistically exploit the physical characteristics of the memory hierarchy.
- Perform run-time characterization of video streams for efficient data placement on hybrid memories
- Propose a hybrid-memory-aware video codec.

II. RELATED WORK

A. Emerging Memory Technologies

In the quest for reducing leakage power of traditional memory technologies (SRAM, DRAM), hybrid memory hierarchies have been proposed. Joo et al. [7] studied the use of PCM as an alternative to SRAM for on-chip caches. Sun et al. [6] proposed an SRAM-MRAM hybrid L2 cache in a NUCA-based 3D stacked multi-core platform. Mishra et al. [8]

978-1-4673-4967-3/12/$31.00 ©2012 IEEE

introduced STT-RAM as an alternative to MRAM memory and proposed solutions at the on-chip network level to prioritize memory accesses and hide the overheads in write latencies. Wu et al. [9] studied hybrid cache architectures at a finer granularity and proposes a solution with various types of NVMs across cache levels and across the same level of the cache. Hu et al. [10] proposed a dynamic data allocation algorithm to exploit a hybrid on-chip scratch pad memory consisting of PCM and SRAM. Zhou et al. [13] proposed the use of PCM as the main memory for a 3D stacked chip. Dhiman et al. [12] proposed a low overhead hybrid hardware software solution for managing a PCRAM and DRAM main memory hierarchy. Wongchaowart et al. [18] used a placement algorithm that exploits content based signatures to mitigate write overheads in main memory. Mogul et al. [19] described Operating System support based on annotated semantic information for adopting hybrid memory as a solution for hybrid memory. Ferreira et al. [20] proposed an architecture that allows drop-in replacement of a conventional DRAM main memory with PCM. Liu et al. [21] exploited variable data partitioning based on statically extracted application characteristic in the context of hybrid main memory enabled DSP systems. Lee et al. [22] proposed a hybrid aware memory management and data migration using static analysis of application's behavior. Bathen et al. [11] proposed a run-time memory management technique that allows a transparent sharing of distributed on-chip hybrid memories by using a hybrid aware, compiler generated and/or user assisted, application's address space partitioning.

AVid is different from previous hybrid memory subsystem approaches in that we are able to make an informed memory mapping decision with virtually no overheads. Using an application's input data oriented approach we exploit the predictable decoder access patterns by collecting metadata on the fly during encoding. Although *AVid* targets distributed scratchpad and non-volatile memories, our scheme may be extended to hybrid cache based systems [6][8][9].

B. *Energy Efficient Decoding*

Several related efforts have proposed techniques for energy-efficient multimedia decoding. Choi et al. [23] presented a DVS technique for MPEG decoding to reduce energy consumption while maintaining the quality of service by predicting the computational workload using a frame-based history. Based on the predicted workload the voltage and frequency are scaled to stretch the decoding time to the limit of the frame deadlines. Using a similar approach to DVS Cornea et al. [14] proposed a technique that performs an off-line profiling and annotation of the stream to reduce the overhead of estimating the frame decoding times at runtime. Furthermore, in [24] the annotated meta-information is used for backlight adjustment of LCD displays during multimedia playback for improved battery life. In another approach on DVS for multimedia decoding Chung et al. [15] proposed the profiling of streams and extraction of meta-data characterizing execution time variations at the server side through static analysis. Huang et al. [17] proposed an offline bit stream analysis and metadata insertion, done while the multimedia file is downloaded to a mobile device, to capture the expected computational demand during decoding. More recently, Gheorghita et al. [25] and Hamers et al. [16] proposed a scenario-based voltage and frequency scaling approach for low energy multimedia decoding. The scenarios capture platform-independent multimedia decoding complexity and resource requirements and are annotated to the stream.

As opposed to previous approaches to energy efficient multimedia decoding *AVid* doesn't require static analysis of the application's execution or profile classes of inputs to infer run-time behavior. The metadata lives in the stream and is specific to each input. However, our work can be complemented by existing DVS schemes that predict decoding time and energy consumption [14][15][16][23] to gain further energy savings.

In summary, we take a different approach on profiling multimedia streams and exploit the aggregated meta-information in an energy efficient memory allocation technique that leverages the unique characteristics of hybrid memory hierarchies. This work explores optimization opportunities for video decoding but can be easily extended to the higher class of data compression applications.

III. MOTIVATION

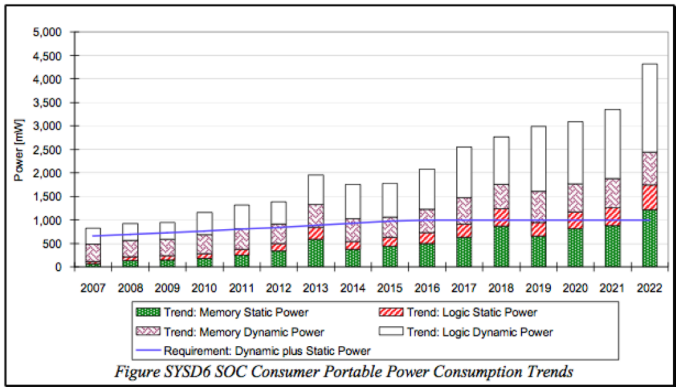

Figure 1: ITRS Power Consumption Trend [26]

Figure 1 shows ITRS' consumer portable power consumption trends for the next decade [26]. As we can observe, the trend for static power (dotter green bar) will overshadow dynamic power consumption (pink diagonal squares). As a result, designers are eager to adopt emerging non-volatile memories to mitigate the growing leakage power issues in today's memory subsystems. Although non-volatile memories are viable candidates for replacing SRAM and DRAM-based memories due to their low leakage power, the adoption of non-volatile memories is challenging due to fact that their write operation overheads in terms of energy and latency are orders of magnitude higher than their read operations. To illustrate this, Table 1 summarizes the essential parameters of two most common types of NVMs (MRAM and PCRAM), in comparison with SRAM; from this table it is clear that NVMs cannot be used as a drop-in replacement for classic memory technologies and that a different approach is needed. An immediate idea is hybrid-memory which tries to get the best from both worlds by complementing NVMs with

SRAMs/DRAMs. The goals of hybrid-memory hierarchy are to obtain reduced leakage, increased capacity, comparable read latency, while minimizing write overheads. To achieve these goals using hybrid-memories, it is critical that a software stack understands and exploits the special characteristics of these hybrid memories.

Table 1: Parameters of memory technologies [9]

Memory Size	Norm. Density	Latency (cycles)	Dyn. Energy (nJ)	Static Power (W)
SRAM (1MB)	1	8	0.388	1.36
MRAM (4MB)	4	Read: 20 **Write 60**	Read: 0.4 **Write: 2.3**	0.15
PCRAM (16MB)	16	Read: 40 **Write: 200**	Read: 0.8 **Write: 1.5**	0.3

Using an application oriented approach we identified video codecs as a good candidate for this purpose. This class of applications is omnipresent in current computer systems, ranging from server machines to mobile platforms. One issue in current battery constrained devices is that video codecs are memory, and implicitly power, hungry. For instance, the Decoded Picture Buffer (DPB), which is an integral data structure in Motion Estimation on the encoder side and Motion Compensation on the decoder side, can take up to 12MBs of memory for 720p video with 9 stored frames. Although the DPB is memory intensive, notice that its access patterns are highly predictable from the decoder's perspective since they can be accurately extracted at the encoding time and embedded in the stream with negligible overhead.

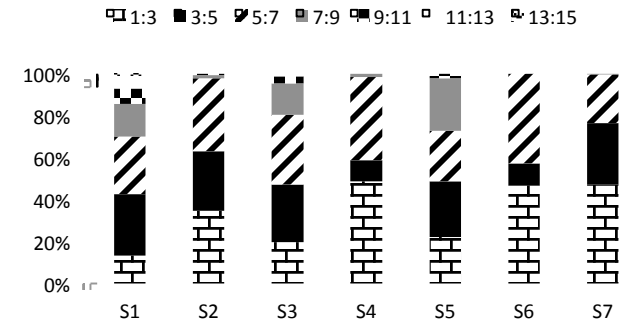

Figure 2: Frames R/W Distribution over 720p Sequences

By profiling and analyzing the encoder's access patterns we determined that frames stored in the DPB have an average Read-to-Write (R/W) ratio of about 3.5, which means that they are written once and read more than three times, and a maximum R/W of about 15. A more detailed distribution of R/W accesses over the frames of several video sequences is presented in Figure 2. We plotted the results over seven 720p sequences (S1-S7 in the horizontal axis), we aggregated the frames in bins based on their R/W ratio (e.g., 1:3 represents the frames that have a R/W ratio between 1 and 3) and show the percentage of frames in each bin (vertical axis). The high number of read accesses compared to write accesses makes the case for mapping this data structure to NVMs to *overcome the disadvantages of unbalanced R/W costs*. Although we will focus on frame level access frequency, our scheme can be easily extended to other levels of data granularity (e.g., Slice and MB level).

With the increasing number of multi-core on-chip platforms researchers have proposed different approaches to exploit video decoding data parallelism. The most promising approach seems to be macro-block level parallelization, a highly scalable scheme. Even with the state of the art implementations of parallel H.264 decoding, a major limitation is main memory bandwidth requirements [27]. Due to their higher densities, NVMs are a good candidate for *enhancing the on-chip memory space while maintaining the same area*, and thus decreasing accesses to main memory.

Besides increasing the on-chip memory space with minimal area overhead, NVMs, both on-chip and off-chip, have very low leakage power, thus *reducing total memory consumption*, which is an important aspect in mobile platforms.

IV. Video Codec Background

Multimedia applications are a special case of streaming applications, which process a continuous incoming stream of data. Unlike general-purpose applications, multimedia workloads tend to have a more regular execution being executed in a pipelined fashion, with each stage of the pipeline being a well-defined algorithm. This exposes unique opportunities for observing and extracting execution patterns that can be used for content oriented power and performance optimizations.

For our study we choose video processing since it is the most resource demanding class of multimedia applications. Out of the many video standards H.264/AVC stands out as being widely adopted due to its compression capabilities; however, our scheme can be easily extended to other standards.

A. H.264

The H.264/Advanced Video Coding (AVC) codec achieves dramatic compression ratios necessary for the storage and transmission of large format, high quality video.

H.264 stream has a hierarchical structure. A stream is composed of Groups of Pictures (GOP): each GOP contains three types of frames (I, P, B), each frame can be divided in slices, each slice is further divided into macro-blocks of 16x16 pixels, and each macro-block can be subdivided in partitions.

In earlier standards only *I* and *P* frames were used for reference. Each *P* frame was predicted from the preceding *I* or *P* frames and each *B* frame was predicted from the *I* and/or *P* frames on either side of it. An example of 'classic' GOP structure is presented in Figure 3. The H.264 standard introduced new prediction structures. One example is the hierarchical GOP structure which enables prediction from *B* frames. Moreover, it is possible to use multiple reference frames for prediction, which means that the encoder can search up to N reference frames to find the best match for each

macro-block. This offers potential for improved compression efficiency by trading-off increased computational expense at the encoder and increased storage at encoder and decoder, since the DPB must now store N reference frames [28].

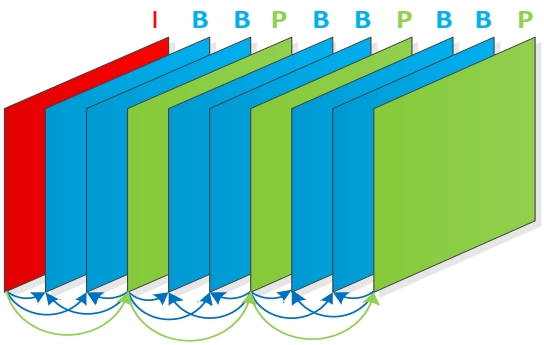

Figure 3: Frames in a H.264 stream

The H.264 decoder pipeline consists of four multimedia kernels: Entropy Decoder (ED), Inverse Discrete Cosine Transform (IDCT), Quantization (Q), Motion Compensation (MC) and the Deblocking Filter. A high level diagram of the decoder is given in Figure 4. A macro-block is decoded, re-scaled and inverse transformed to form a decoded residual macro-block. The decoder generates the same prediction that was created at the encoder and adds this to the residual to produce a decoded macro-block.

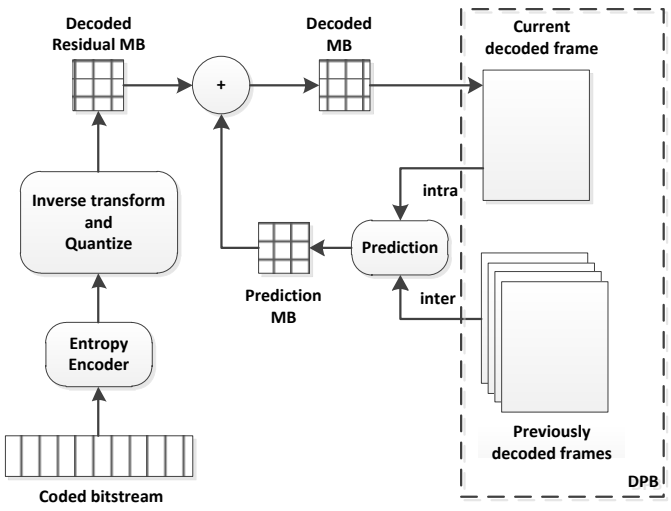

Figure 4: H.264 decoder [28]

B. Prediction

The widespread adoption of H.264 is mainly because of its improved compression performance. Compared to previous video codecs it has a reduction of 50% in bandwidth requirements for a given image quality [27]. The new standard seeks to minimize redundancy in the compressed stream through extensive analysis. Much of the performance gain is due to the efficient prediction methods used. At the encoder for each macro-block a prediction is created based on an extensive search of previously decoded data either from the current frame using intra prediction or from other frames that have already been coded using inter prediction. The prediction is then extracted from the macro-block to form a residual. At the decoder the reverse operation is executed, the residual is combined with the prediction block to form the original macro-block value.

The predicted macro-block is identified by: 1) a motion vector, which gives the position inside a specific frame, and 2) a reference frame index, which gives the corresponding reference frame. Each macro-block can have one or two predictions. Each reference frame is stored in the DPB and its lifetime is dependent on the macro-blocks in other frames referencing it and the display rate.

The rate at which a frame is referenced during motion-compensated prediction depends on the video dynamic visual characteristics and frame types. Generally, an *I* frame has a longer lifetime and higher reference rate than a *P* or *B* frame. In more dynamic videos the GOPs are shorter, thus *I* frames have shorter lifetimes.

C. Parallel H.264 decoder

A lot of effort has been invested in parallelizing the H.264 decoder. Analyzing the structure of the H.264 stream, it's possible to identify several levels at which data parallelism can be exploited: GOP level, frame level, slice level and macro-block level.

The first level of parallelism is exposed by the fact that GOPs are processed independently in decoder's pipeline. However, GOP level parallelism has tremendous memory requirements for HD resolution and can lead to high latencies.

Frame level parallelism can be implemented by exploiting the dependencies between P and B frames, but it is not scalable beyond two or three concurrently processed frames and has high memory requirements.

Slice level seems to be a better idea since a slice is a group of independently encoded macro-blocks within a frame. Encoding many slices per frame is beneficial when streaming over a lossy network due to increased error correction performance. However, by increasing the number of slices per frame the compression efficiency is affected. That is why H.264 videos are usually encoded with only one slice per frame. Due to this reason even if slice level parallelism seems to be a good approach, it is not feasible in practice.

Macro-block level exposes a finer grain data parallelism opportunity emerging from the fact that inter-coded blocks in the same frame are independent. Macro block level parallelism has an increased performance and scalability potential. Baker et al. [27] demonstrate how this can be implemented on an IBM Cell Broadband Engine (CBE) [29] chip multiprocessor by taking advantage of the platform characteristics. Their design uses both functional and data partitioning by mapping MBs from inter-coded frames onto the lightweight SPU cores and dedicating one additional SPU to deblocking filter. The general purpose PPU core manages entropy decoding, which is inherently not parallelizable, and intra decoding data dependencies.

V. EXPLOITING VIDEO STREAM ANNOTATIONS

This section will cover how we exploit video stream annotations to efficiently manage the hybrid memory subsystem. Figure 5 shows a high level view of our system. Here we show how the meta-information is extracted during encoding with virtually no cost and is embedded in the stream payload in the form of annotations. Section V.B discusses the stream profiler, which is a part of the encoder and can be run online in case of streaming media or offline in case of stored media. Section V.C discusses the annotated streams, which are transmitted to the decoder where the reverse operation is executed. The annotations are extracted from the stream during stream decoding and handed to the runtime system, which dynamically maps the data structures to the preferred memory type. Section V.D discusses the overheads incurred by our technique.

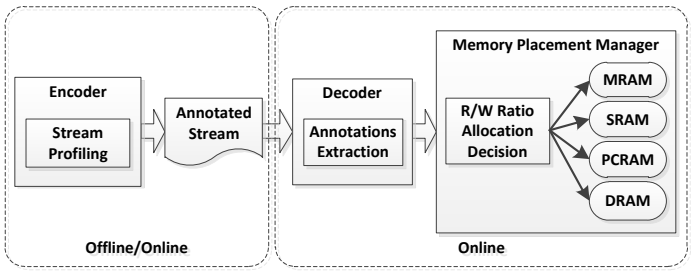

Figure 5: Hybrid Memory Aware Codec High Level View

A. Target Platform and Assumptions

Figure 6 depicts a high level view of our decoder target platform. We choose a chip multiprocessor design similar with IBM Cell Broadband Engine [29] which was proven to have the potential to speedup complex data intensive applications such as H.264 video decoding given an effective parallelization scheme [27]. It consists of a Power Processing Element (PPE), a general purpose control/OS processor, connected to a number of Synergistic Execution Units (SXU), which have extensive support for Single Instruction Multiple Data (SIMD) operations, via an AMBA AHB bus-based communication fabric. Each SXU has a local store consisting of a scratch pad memory (SPM) and a non-volatile memory (NVM). The data flow is controlled by a Direct Memory Access (DMA) unit. The main memory consists of DRAM and NVM. We chose to explore configurations that seem realistic in terms of memory sizes in future mobile platforms.

We make the following assumptions:

1. Programmers have access to the source code of the target video codec in order to enhance it.
2. The platform exposes memory types (SRAM, MRAM, etc.) and capacities to the runtime system.
3. The technique operates over blocks of data (e.g. 1KB mini-pages) such that sections of decoded frames can be mapped to various on-chip physical memories.
4. All instructions/data are mapped to the on-chip/off-chip memories and there are no caches.

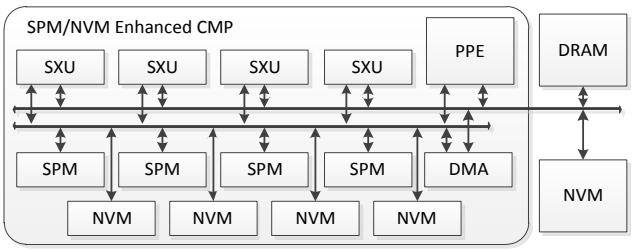

Figure 6: Target Platform

B. Encoder: Aggregating Meta-information

The decoded frame's meta-information is collected at the encoding time with minimal overhead and embedded in the frame control data slots. The meta-information consists of R/W ratio for each frame in the DPB.

Reference frames access frequencies are extracted through profiling during the encoder's ME operation. Each motion vector is characterized by a source frame (a previously decoded frame or the current decoding frame when the vector references a previously decoded section of it), a destination frame (the current decoding frame) and a partition of variable size. Keeping track of the generated motion vectors during encoding we count the number of bytes read from each decoded frame and write to each decoding frame. Thus, our metric consists of a simple R/W ratio for each frame.

The encoder embeds the annotations in the stream using Supplemental Enhancement Information (SEI) NAL units. As defined by the standard, SEI messages contain information that may assist decoding or displaying video but are not essential for constructing decoded video frames. Each annotation is transmitted in a separate SEI Raw Byte Sequence Payload (RBSP) and each SEI message is sent as a separate NAL Unit. To embed the metadata we use User Data Unregistered field, which contains user data defined by a Unique User ID, the contents of which are not defined by the standard. Each annotation contains a float number, 4 bytes, and we send one SEI message per frame, so the overhead is negligible.

By choosing this approach our codec does not affect the standard and does not conflict with other codecs. Thus, a stream with annotated metadata can be processed by a non-hybrid-memory aware decoder, which would simply ignore the User Data Unregistered field and would decode the stream as usual.

At the same time our hybrid-memory aware decoder is able to decode streams that were not encoded with frame R/W meta-information payloads. Moreover it does a hybrid-memory aware DPB mapping using the frame type as a heuristic. As expected, we observed that I and P frames have higher R/W ratios, but it was interesting to see that I frames don't always have a higher R/W ratio than P frames. Therefore we map I and P frames to the most write constrained memory spaces on a first come, first served basis.

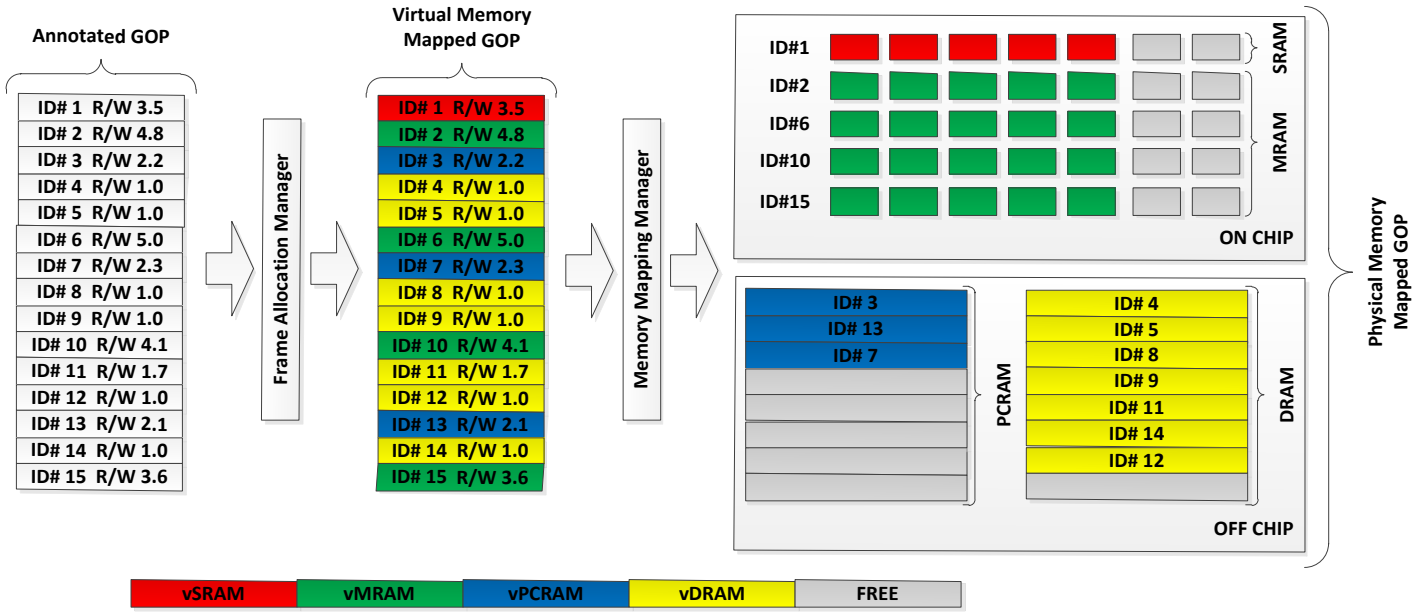

Figure 7: Frames Allocation Example

C. *Decoder: Exploiting Meta-information*

At the decoder we extract the R/W ratio from the compressed stream and use it to give hints to the runtime environment about the desired memory mapping of decoded frames. The runtime environment is a virtualization layer on top of the physical memory hierarchy composed of two main modules: a *frame allocation manager*, which does the frame allocation to a unified virtualized memory space, and a *memory mapping manager*, which does the actual mapping of the virtual memory in the psychical memory space.

Hiding the details of the memory organization such as memory types and sizes to the application level has the advantage of exposing a unified address space to the application. This means that even if the frames sizes are of the order of MBs and the hardware platform has distributed on chip memories with different granularities, 256 KB SRAM and 512 MRAM per core in hybrid mixed configuration, the application does not have to be tuned to this specific configuration. This allows a seamless migration of the same application on a hardware platform with different memory configuration with no additional effort from application's developer perspective.

The *frame allocation manager* tries to maximize the usage of on-chip memories and minimize accesses to off-chip memories. It does this by opportunistically mapping the frames with a higher R/W ratio to the higher levels of memory hierarchy. The mapping is dynamic in nature since the frames are mapped at runtime based on R/W ratios across frames in same GOP. Once a frame is assigned to a specific memory space it is not reassigned later on. We are able to do this since we have a holistic view on the R/W distribution across frames in the entire GOP and we avoid unnecessary migrations between different memory spaces based on newly decoded frames R/W ratio.

The *memory mapping manager* takes care of the mapping and translation between the virtual address space and the physical address space. It does this by presenting the upper levels with an Intermediate Physical Addresses (IPA) which points in the virtual address space. Whenever a frame data needs to be accessed the application gives the IPA and the offset inside the frame, and then is the memory mapping manager's job to retrieve that data from the physical memory.

Using a virtualization layer for embedded on chip memories to isolate the application from varying hardware configurations is not a new idea. For instance, HaVOC [11] does this by using user-assisted and compiler-driven policies enforced by an embedded hardware memory manager implementation. However, our approach is implemented purely in software and does not require any changes in the upper levels of the software stack.

In Figure 7 we give an example of frame mapping in the hybrid memory hierarchy to better understand how the runtime system works. Here we show how the decoded frames are assigned to different memories in the virtual space based on the R/W ratio and how the physical memory mapping is done. The 'free' on-chip space is reserved for code and application's stack.

To apply our technique into a live streaming scenario, the video needs to be delayed with the time necessary to encode and transmit a GOP. This is because in order to make an informed memory mapping decision our technique needs the R/W ratio distribution over the entire GOP. This is only a one time delay, for the first GOP in the sequence. Usually in live streaming the frequency of I frames is higher to be able to recover more gracefully from frame drops due to network errors, thus the GOP is shorter. For example with a typical GOP of 15 frames and an encoding and transmission frame rate of 25 fps the initial delay would only be 600ms.

D. Overheads

Our scheme will incur some overheads; however, these overheads are negligible as demonstrated next. During encoding the overheads are minimal since we don't use a profiler that processes the stream in a separate pass, rather we enhanced the encoder to count accesses to each frame and embed this information in the stream. The stream size overheads are minimal, only four bytes per frame, as shown in section V.B. Overall the compressed stream size increases with less than 0.01% on average, thus we don't put additional pressure on the network. Of course the runtime system will incur more overheads due to the decision making and differential data mapping, but as we show in section VI.C, these overheads don't cause frame loss and overall our enhanced decoder is able to decode frames even earlier before the deadline.

VI. Experimental Evaluation

A. Experimental Goals

The main goal of the experiments was to show that our proposed scheme is able to gain significant energy savings. For this purpose we measured total energy consumption (static and dynamic) of the entire memory hierarchy (off-chip and on-chip). To show that our technique is applicable in practice and doesn't cause video quality loss due to missed deadlines because of higher latency of accesses to NVMs, we also measured the effect on overall memory latency.

We present our results as savings of the three hybrid memory configurations compared to the uniform memory configuration. In each of the graphs on the vertical axis we plot the percentage of savings and on the horizontal axis we plot the results for each of the video sequences and the average case. The three cases are: 1) uniform versus hybrid-on-chip, 2) uniform versus hybrid-off-chip, and 3) uniform versus hybrid-mixed.

B. Experimental Setup

Our software configuration uses the H.264/AVC reference implementation JM 18.3 [30] as the encoder and the open source FFmpeg H.264 [31] as the decoder. We chose to use the reference implementation of H.264 on the encoding side in the detriment of its highly optimized open source counterpart, FFmpeg, since it does not affect the way the stream is encoded and the results of applying our technique on the decoder side. We enhanced the encoder to keep track of decoded frames access frequencies and to embed this meta-information in the stream. For the decoder we used the MB level parallel programming model described in section IV.C, and enhanced it to extract the frame R/W ratio meta-information and pass it to the runtime system.

The benchmark videos used to test our technique were taken from *Xiph.org Test Media* collection [32] and are listed in Table 2. We experimented with sequences at 720p and 1080p resolutions. Each sequence is 500 frames long and use YUV420 color space. We choose YUV420 since it is the most commonly used color space. The sequences have been encoded using settings based on the High 5 profile.

Table 2: Test Sequences

Resolution	Name	ID
720p	mobcal	S1
	parkrun	S2
	shields	S3
	stockholm	S4
	ducks_take_off	S5
	in_to_tree	S6
	old_town_cross	S7
	park_joy	S8
1080p	crowd_run	S1
	ducks_take_off	S2
	in_to_tree	S3
	old_town_cross	S4
	park_joy	S5
	rush_hour	S6
	sunflower	S7
	tractor	S8

Table 3: Simulator Configurations

Resolution	Scenario	Name	Size (MB)				Cores
			SRAM	MRAM	DRAM	PCRAM	
720p	1	**Uniform**	2		1024		8
		HybridOnChip	1	4	1024		
		HybridOffChip	2		512	512	
		HybridMixed	1	4	512	512	
	2	**Uniform**	4		1024		16
		HybridOnChip	2	8	1024		
		HybridOffChip	4		512	512	
		HybridMixed	2	8	512	512	
	3	**Uniform**	6		1024		24
		HybridOnChip	3	12	1024		
		HybridOffChip	6		512	512	
		HybridMixed	3	12	512	512	
1080p	1	**Uniform**	6		1024		24
		HybridOnChip	3	12	1024		
		HybridOffChip	6		512	512	
		HybridMixed	3	12	512	512	
	2	**Uniform**	8		1024		32
		HybridOnChip	4	16	1024		
		HybridOffChip	8		512	512	
		HybridMixed	4	16	512	512	
	3	**Uniform**	12		1024		48
		HybridOnChip	6	24	1024		
		HybridOffChip	12		512	512	
		HybridMixed	6	24	512	512	

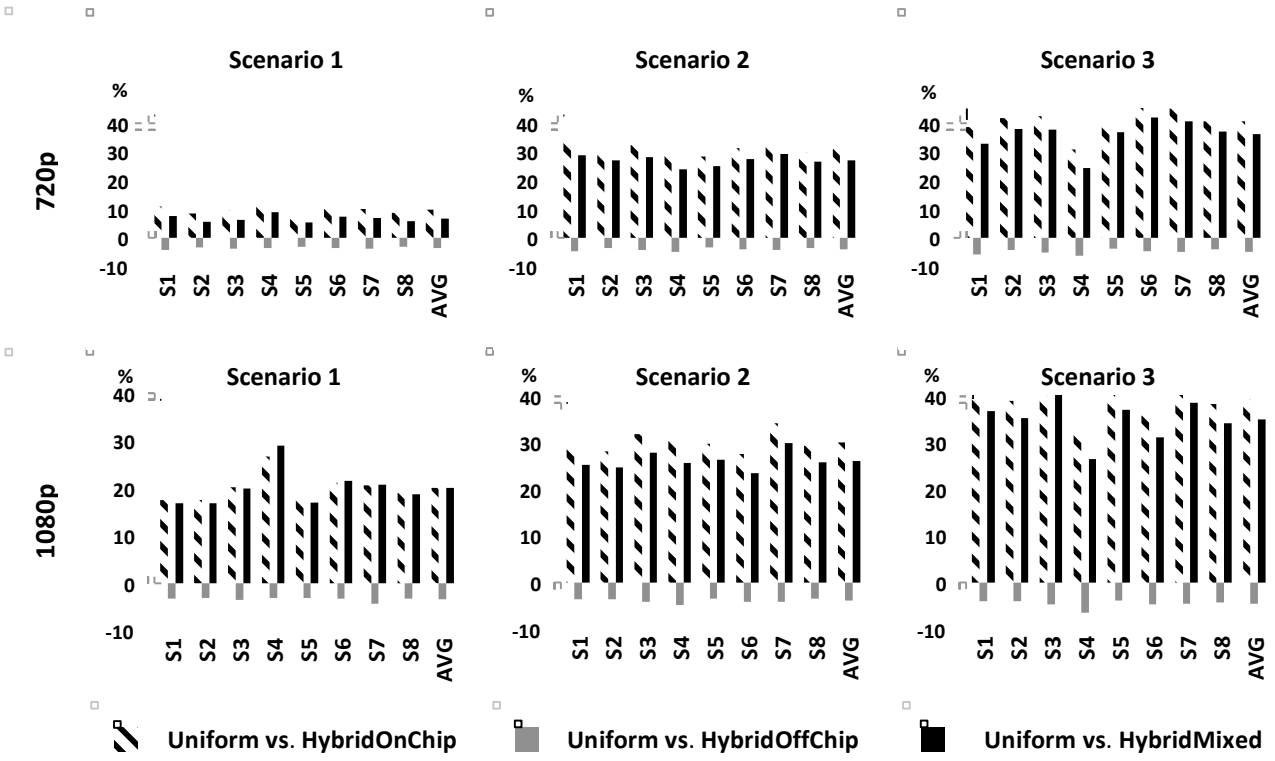

Figure 8: Slack Time *(higher is better)*

Our hardware simulator emulates the multicore platform described in section *V.A*. From here on we will generically refer to SXU units as cores, while the PPE is named power processor. Each core has 256KB SRAM in the uniform memory configuration and 128KB SRAM/ 512KB MRAM in the hybrid memory configuration. The off-chip memory is either 1024MB DRAM or 512MB DRAM and 512MB PCRAM. For extensive experimental results we varied the number of cores from 8 to 48 as shown in Table 3. Since we consider the same memory space per core in all configurations, increasing the number of cores also increases the available on-chip memory space. For 720p videos we run tests for 8, 16 and 24 cores configurations. We chose the lower limit as being 8 cores to match 720p's decoder minimum computation requirements for achieving a 25fps rate. Going beyond 24 cores for 720p would give us better results but mainly due to increase on-chip memory space and not so much due to our tuned memory mapping. Due to increase memory requirements for 1080p video we test this in 24, 32 and 48 cores configurations. For each scenario we consider 4 configurations: Uniform, Hybrid On-Chip, Hybrid Off-Chip and Hybrid Mixed memory hierarchies.

Of the total available on-chip memory space we use only 75% for data since the considered parallel model needs space for execution stack and code mapping. In case of hybrid on-chip memory we map the code to on-chip NVM since it is written only once per application and executed many times.

C. *Effects on Slack Time*

To determine how performance is affected we measured the memory subsystem latency. The results in this case, presented in Figure 8, are interpreted as slack time deviations, i.e., the difference between the frame deadline and the frame decoding time over all configurations. The results show hybrid memory configurations slack time with respect to the uniform memory configuration slack time. Our results show than in the optimal configuration for energy savings we also obtain slack time improvements of 36.2% for 720p, 24 cores, hybrid mixed memory, and of 35% for 1080p, 48 cores, hybrid mixed memory. In the case of hybrid on-chip configuration the slack time improvement is even higher, above 40% in some cases. The difference is due to the impact of high write time that the off-chip NVM has. This effect can be clearly observed in hybrid-off chip configuration where the slack time improvement is as low as negative 5% on average, which translates in potentially missing 5% of the frames deadlines and slightly degrading the perceived video quality. We say potentially since this value is with respect to the uniform memory hierarchy slack time, and it is reached only when in case of uniform memory the frame decoding time matches exactly the frame deadline. A higher slack time translates into increased decoding performance; the frames are decoded earlier than their deadlines. This could be exploited by a DVS scheme which would slow down the cores to the limit of meeting the deadlines, thus saving more computation energy.

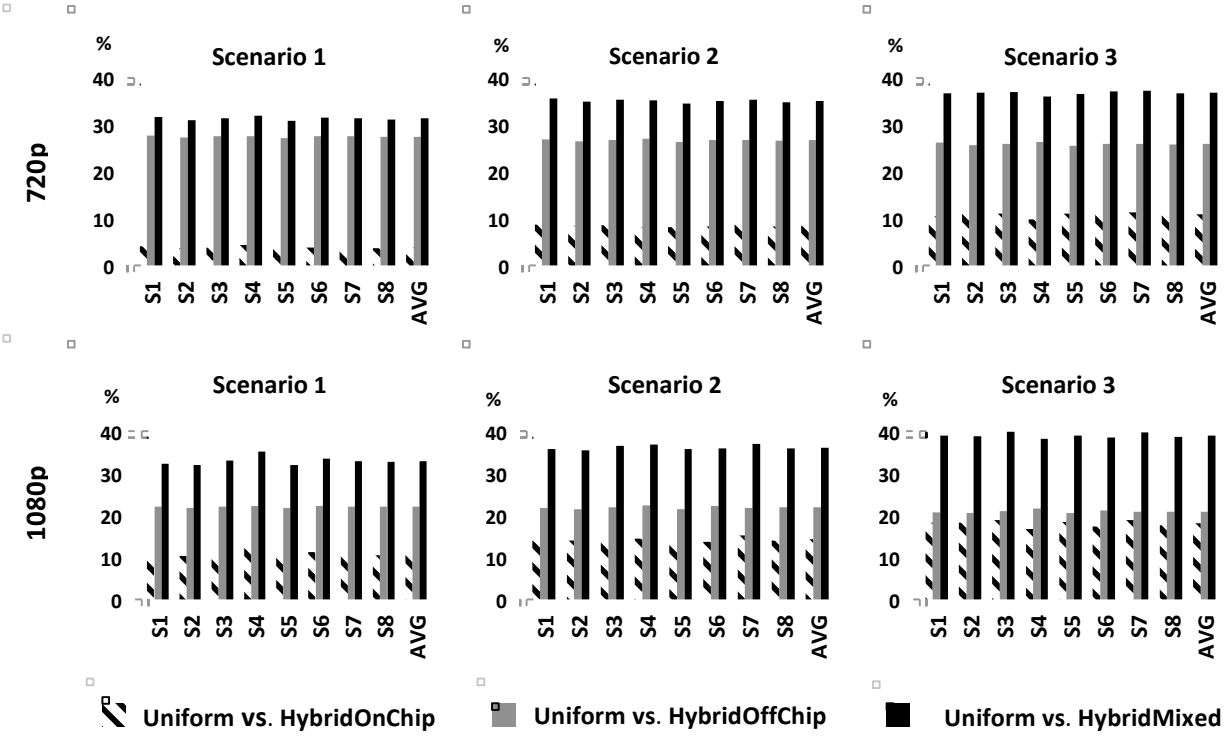

Figure 9: Energy Savings *(higher is better)*

D. *Energy Savings*

When looking at energy savings (Figure 9), it is important to note that varying the number of cores, which implicitly increases the available on-chip memory space considerably, does not affect the results significantly. Even in the lowest configurations we obtain average savings of 31.3 % for 720p, 8 cores, hybrid mixed memory, and of 32.9% for 1080p, 24 cores, hybrid mixed memory. In the highest configurations we obtain savings of 36.7% for 720p, 24 cores, hybrid mixed memory, and of 38.9% for 1080p, 48 cores, hybrid mixed memory.

Looking at the three different hybrid memory configurations across all graphs it is clear that hybrid mixed gives the highest memory energy savings. This is because in this configuration our technique is able to exploit dynamic energy savings for both on-chip and off-chip and because NVMs have lower leakage power. In the hybrid off-chip configuration the savings are lower but still considerable. In this case our technique can only exploit the lower static energy and does its best to overcome the impact of higher dynamic energy of PCRAM. For hybrid on-chip configuration we observe even lower savings due to the limited optimization opportunities offered by the small on-chip space. However it is interesting to observe that increasing the on-chip memory space leaves less optimization opportunities for hybrid off-chip configuration since many of the high R/W ratio will fit on-chip and for example in 1080p, 48 cores scenario the hybrid-on chip does almost as good as hybrid off-chip, 18% versus 20.8% on average.

Overall, combining hybrid-on chip with hybrid-off chip configurations seems to be the winning scenario when targeting energy savings. The energy results are complemented by the increased slack time which can be exploited by other optimizations targeting performance. An important part of our savings comes from increasing the on-chip memory space by taking advantage of higher density of NVMs while keeping the same area. By tuning the memory mapping using reference frames access R/W higher level information ratio we are able to overcome the unbalanced R/W costs of NVMs.

VII. Conclusions And Future Work

In this paper we presented *AVid*, an annotation driven video decoding scheme for hybrid memory subsystems. We show how to exploit video decoder access patterns by embedding this meta-information as annotations in the video stream to exploit the physical characteristics of nonvolatile memories for energy savings and performance improvements. Our experimental results show that our technique is able to achieve execution time and energy reduction by up to 40.8% and 39.7% respectively when applied to H.264 decoding. To the best of our knowledge there is no other work proposing hybrid memory hierarchies for optimizing video decoding applications by exploiting reference frames access frequency meta-information.

Due to increased slack period our technique can be complemented by voltage scaling schemes to save even more power. Combining our technique with existing DVS schemes might give interesting results.

In the future we would like to extend the current technique to sub-frame granularity, i.e., quantify accesses to portions of the frames during Motion Compensation and map frame sections differentially.

VIII. Acknowledgements

This work was partially supported by NSF Variability Expedition Grant Number CCF-1029783.

References

[1] L. A. Barroso and U. Holzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, no. 12, pp. 33-37, 2007.

[2] "ITRS. System drivers.," 2003. [Online]. Available: http://www.itrs.net/.

[3] N. S. Kim et al., "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68-75, 2003.

[4] N. Azizi, F. N. Najm, and A. Moshovos, "Low-leakage asymmetric-cell SRAM," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 11, no. 4, pp. 701-715, 2003.

[5] "ITRS. System drivers," 2005. [Online]. Available: http://www.itrs.net/.

[6] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *High Performance Computer Architecture, 2009. HPCA 2009. IEEE 15th International Symposium on*, 2009, pp. 239-249.

[7] Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie, "Energy- and endurance-aware design of phase change memory caches," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010, pp. 136-141.

[8] A. K. Mishra, X. Dong, G. Sun, Y. Xie, N. Vijaykrishnan, and C. R. Das, "Architecting on-chip interconnects for stacked 3D STT-RAM caches in CMPs," in *Proceedings of the 38th annual international symposium on Computer architecture*, 2011, pp. 69-80.

[9] X. Wu, J. Li, L. Zhang, E. Speight, R. Rajamony, and Y. Xie, "Hybrid cache architecture with disparate memory technologies," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 34-45.

[10] J. Hu, C. J. Xue, Q. Zhuge, W.-C. Tseng, and E. H.-M. Sha, "Towards energy efficient hybrid on-chip Scratch Pad Memory with non-volatile memory," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, 2011, pp. 1-6.

[11] L. A. Bathen and N. Dutt, "HaVOC: a hybrid memory-aware virtualization layer for on-chip distributed ScratchPad and non-volatile memories," in *Proceedings of the 49th Annual Design Automation Conference*, 2012, pp. 447-452.

[12] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, 2009, pp. 664-669.

[13] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 14-23.

[14] R. Cornea, A. Nicolau, and N. Dutt, "Video Stream Annotations for Energy Trade-offs in Multimedia Applications," in *Proceedings of the Proceedings of The Fifth International Symposium on Parallel and Distributed Computing*, 2006, pp. 17-23.

[15] E.-Y. Chung, G. De Micheli, and L. Benini, "Contents provider-assisted dynamic voltage scaling for low energy multimedia applications," in *Proceedings of the 2002 international symposium on Low power electronics and design*, 2002, pp. 42-47.

[16] J. Hamers and L. Eeckhout, "Scenario-Based Resource Prediction for QoS-Aware Media Processing," *Computer*, vol. 43, no. 10, pp. 56-63, 2010.

[17] Y. Huang, S. Chakraborty, and Y. Wang, "Using offline bitstream analysis for power-aware video decoding in portable devices," in *Proceedings of the 13th annual ACM international conference on Multimedia*, 2005, pp. 299-302.

[18] B. Wongchaowart, M. K. Iskander, and S. Cho, "A content-aware block placement algorithm for reducing PRAM storage bit writes," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 2010, pp. 1-11.

[19] J. C. Mogul, E. Argollo, M. Shah, and P. Faraboschi, "Operating system support for NVM+DRAM hybrid main memory," in *Proceedings of the 12th conference on Hot topics in operating systems*, 2009, p. 14.

[20] A. P. Ferreira, B. Childers, R. Melhem, D. Mosse, and M. Yousif, "Using PCM in Next-generation Embedded Space Applications," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, 2010, pp. 153-162.

[21] T. Liu, Y. Zhao, C. J. Xue, and M. Li, "Power-aware variable partitioning for DSPs with hybrid PRAM and DRAM main memory," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, 2011, pp. 405-410.

[22] K. Lee and A. Orailoglu, "Application specific non-volatile primary memory for embedded systems," in *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, 2008, pp. 31-36.

[23] K. Choi, K. Dantu, W.-C. Cheng, and M. Pedram, "Frame-based dynamic voltage and frequency scaling for a MPEG decoder," in *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, 2002, pp. 732-737.

[24] R. Cornea, A. Nicolau, and N. Dutt, "Software annotations for power optimization on mobile devices," in *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, 2006, pp. 684-689.

[25] S. V. Gheorghita, T. Basten, and H. Corporaal, "Application Scenarios in Streaming-Oriented Embedded-System Design," *Design Test of Computers, IEEE*, vol. 25, no. 6, pp. 581-589, 2008.

[26] "ITRS. 2008 update overview.," 2008. [Online]. Available: http://www.itrs.net/.

[27] M. A. Baker, P. Dalale, K. S. Chatha, and S. B. K. Vrudhula, "A scalable parallel H.264 decoder on the cell broadband engine architecture," in *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis*, 2009, pp. 353-362.

[28] I. E. Richardson, *The H.264 Advanced Video Compression Standard*, 2nd ed. Wiley Publishing, 2010.

[29] D. C. Pham et al., "Overview of the architecture, circuit design, and physical implementation of a first-generation cell processor," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 1, pp. 179-196, 2006.

[30] K. Sühring, "JM 18.3." [Online]. Available: http://iphome.hhi.de/suehring/tml/.

[31] F. Bellard, "FFmpeg." [Online]. Available: http://www.ffmpeg.org/.

[32] "Xiph.org Test Media." [Online]. Available: http://media.xiph.org/video/derf/.