Chapter 2
Background and Related Work

This chapter discusses the basics of video processing in general, while specifically targeting the video coding applications. General video system design and its memory access patterns and resource utilization are deliberated. Fundamentals of HEVC and H.264/AVC video encoding are followed by their associated challenges when designing computationally efficient video processing systems. Modern technological challenges that arise in deploying video systems are also presented in this chapter. Afterwards, the state-of-the-art techniques to meet these design challenges are discussed, with details targeting video processing system's software and hardware layers.

2.1 Overview of Video Processing

The working of a video processing system largely depends upon its complexity. The complexity of a video system can be roughly correlated to two characteristics, processing algorithm and throughput constraint. As discussed in Sect. 1.2, the computational complexity of video processing depends upon the type of the algorithm, along with the video frame dimensions and the FPS requirements (given by f_p). From FPS requirements, the maximum time that can be spent on processing a single video frame is given by $t_{frm}=1/f_p$. A general metric to present the throughput requirements is given by $w \times h \times f_p$, which denotes the number of pixels that must be processed in one second, for a frame of size $w \times h$ pixels. However, most video algorithms process a block of pixels in a time unit, i.e., all the pixels within the block correspond to a particular computational mode. An example of dividing a video frame into blocks of size $b_w \times b_h$ is shown in Fig. 2.1.

The authors would like to point out that this work was carried out when all the authors were in Department of Computer Science, Karlsruhe Institute of Technology, Karlsruhe, Germany.

M.U.K. Khan et al., *Energy Efficient Embedded Video Processing Systems*,
DOI 10.1007/978-3-319-61455-7_2

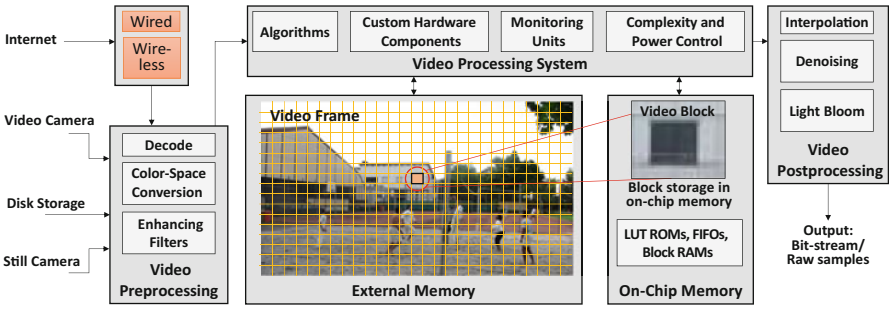

Fig. 2.1 Fundamentals of a video processing system. The video frame is divided into blocks and logically stored in the external memory as shown in the figure. A block of size $b_w \times b_h$ is read at a time from external memory and brought to the on-chip memory for processing

Generally, the video is processed online or offline. Further, the frames themselves are stored in the external memory due to the size of the video frame. A block (or a group of blocks) is transferred from the external memory to the on-chip memory for processing, because accessing the on-chip memory is faster. Once processed, the output of the processed block is concatenated to the output of other blocks.

Figure 2.1 presents an example of video processing system, with pre- and post-processing modules. For video system design discussion, naïve online gaming scenario can be considered. The video application reads the data (possibly compressed video) from the gaming server, decodes it, and then applies different image enhancement techniques before finally forwarding it to the main processing engine. The processing engine may also use data on the local disk and video content captured using the camera(s) at the user's end for processing. Afterwards, this data is displayed on the display device, using post-processing filters like light blooming, denoising, and interpolation. Further, the data/video formed at the user's end is also encoded and encrypted and finally sent back to the gaming server. Note that pre- and post-processing modules themselves can be separate video processing systems, employing the same principles as given in this figure. Further, these modules can be deployed on independent hardware resources, e.g., video preprocessing on CPUs and post-processing on GPU.

For block-based video processing, the throughput requirement can be written as:

$$n_{\text{sec}} = n_{frm} \times f_p = \frac{w \times h}{b_w \times b_h} \times f_p \tag{2.1}$$

Here, n_{sec} presents the number of blocks that must be processed per second, and n_{frm} is the number of blocks within a frame. A larger n_{sec} corresponds to more computational and power requirements and vice versa.

A video sample can be stored in multiple formats. Usually, a sample is stored as a union of three colors, red (R), green (G), and blue (B). This color space format is termed as RGB format. If intensity of each color of video samples is presented by 8-bits, then a sample is stored as a 24-bit value. Usually, R, G, and B "planes" of the

frame are stored separately. Therefore, the number of addresses (or pixels) is three times more. Therefore, the size of the frame can be presented in number of bits b_{frm} by:

$$b_{frm} = w \times h \times 3 \times \text{bits\_per\_sample} \tag{2.2}$$

The RGB color format carries a lot of redundancy and therefore burdens the storage/communication. By focusing predominately on the features which are visually more distinguishable to the human user, this redundancy can be removed. One prime method is to transform RGB to a different space. For example, a large number of video algorithms work with the YCbCr format (also sometimes called YUV format), where Y presents the luminance component of the sample, and Cb and Cr collectively determine the chrominance of the sample. YCbCr 4:4:4 format means that for every luminance pixel, there are two chrominance pixels, whereas YCbCr 4:2:0 format means that the two chrominance pixels are associated with four luminance pixels. Hence, both the chrominance planes are downsampled by two, in horizontal and vertical directions. This still results in high visual quality because the human eye is more susceptible to the luminance component than the chrominance components. For YCbCr 4:2:0 case, the size of the frame (in bits) is denoted by:

$$b_{frm} = w \times h \times (1 + 0.25 + 0.25) \times \text{bits\_per\_sample} \tag{2.3}$$

In Fig. 2.2, an example baseline architecture of a real-time camera-based image/video processing system is illustrated. The camera captures videos in real time at a certain frame acquisition rate in terms of frames per second (typical values are 30, 60, etc.). The raw data (i.e., video samples) is preprocessed via a video input pipeline (VIP). The output of VIP is the streaming data, containing YCbCr 4:2:0 samples, which is converted into words of size ≥ 8 bits using a combination of a FIFO and a shift register. Each frame is stored in a separate location within the main memory. The memory is composed of multiple frame memory partitions in order to simultaneously store multiple video frames and support double or triple buffering. Write address generation unit (AGU) is used to write video frames to the memory partitions, and frames are read via Read AGU. The application requests a new

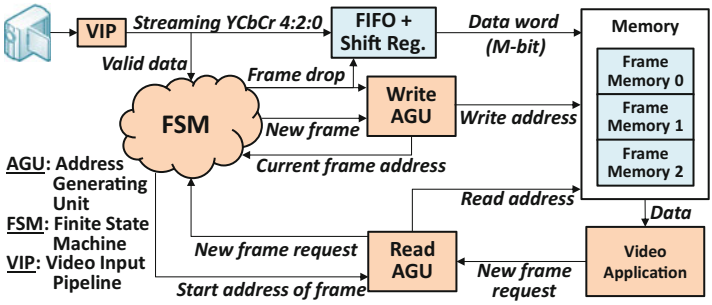

Fig. 2.2 Video capture, store and access management of a video processing system

frame once it has processed the previous frame. Frame drop mechanism (i.e., previously stored frames are overwritten, although they are not processed) is also supported, in case the video processing application has high complexity and it cannot cope with the high frame capture rate of the camera. Other factors that might induce frame-dropping are e.g., constrained underlying hardware with lower clock frequency, lesser CPUs, smaller caches, low bandwidth of wired/wireless connections etc. Frame drops can also occur if the output of the video system is stalled (e.g., scenarios like high congestion and packet loss in wired/wireless output scenarios, the output disk is full).

Generally, small parts of the video processing code (called kernels) take a hefty portion of computational complexity and power. In order to reduce the complexity, these kernels are optimized by developing past-predicts-future paradigms and by tuning the complexity knobs. For example, the number of modes computed to search for the best mode can be reduced, which increases the throughput-per-watt metric, but may reduce the output video quality. These kernels can also be implemented as custom hardware accelerators.

In addition to video applications, other applications are also sometimes referred to as "frame-based" applications. The term "frame-based" application broadly denotes an application that needs to process a set of data periodically, within a specific interval of time.

2.1.1 Intra- and Inter-frame Processing

Broadly, there are two distinct types of video processing algorithms employed by video systems:

1. Intra-frame processing, where the spatial neighborhood of the pixel (or block) under consideration is used for computations. That is, the same frame is used for processing the current pixel/block.
2. Inter-frame processing, where the temporal neighborhood of the pixel (or block) under process is used for computations. That is, the frame(s) processed in the past or future is used in algorithms involving the current pixels/block processing. The frames used for interframe processing are also called reference frames.

The neighborhood concept is illustrated in Fig. 2.3, where a block under consideration has spatial neighbors within the frame and temporal neighbors across the frames. The same concept is applicable to pixel-based processing; however, our main focus in this book will be block-based processing. The spatial and temporal neighbors are exploited for multiple purposes, e.g., noise filtering and motion tracking. The correlation of the current pixel/block with its neighbors plays an important role in determining the video quality, as will be discussed in the coming text. High correlation in the neighborhood usually translates to higher video quality (e.g., better noise removal if neighbors correlate highly with the current block). Mostly, the spatial neighbors are not as highly correlated with the current block; therefore, the video output quality is not high for intra-frame processing, compared

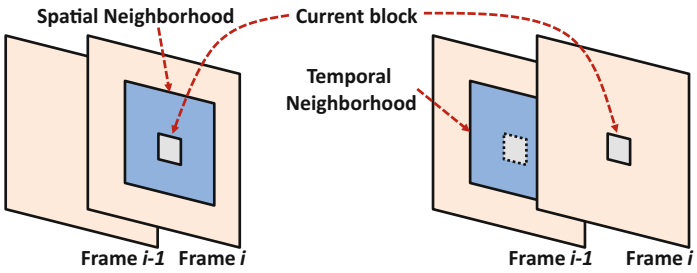

Fig. 2.3 Spatial and temporal neighborhood of current block under consideration

to inter-frame processing. However, the computational requirement of inter-frame processing is higher due to a large amount of data being transferred between external and on-chip memory. This increases the data processing pressure and the power overhead. On the other hand, a larger complexity corresponds to searching for the best mode of computations and hence increases the output video quality. The output video quality can be compared against an ideal output using peak signal-to-noise ratio (PSNR) and Bjøntegaard delta peak signal-to-noise ratio (BD-PSNR) or bitrate (BD-Rate) [1]. This will become clear when we discuss video coding overview in Sect. 2.2.

Usually, the on-chip memory shown in Fig. 2.1 is large enough to hold additional data, and therefore, in addition to the video frame block, its spatial or temporal neighbors can also be stored on-chip. This further reduces the stress on the memory subsystem.

2.2 Overview of Video Coding

Here, we discuss video coding algorithms with reference to the principles mentioned in Sect. 2.1. This section introduces the working principles of H.264/AVC [2] and HEVC [3] video encoders. The working principles of a video encoder are diagrammatically shown in Fig. 2.4. A video frame block (of size $b_w \times b_h$, with samples s) is brought from the external memory to the on-chip memory and is compressed by searching for the best compression mode/configuration, by iteratively testing the intra- and inter-compression modes (also referred to as intra- and inter-predictions). Afterwards, the best compression mode is forwarded to the additional video compression modules. The purpose of both intra- and inter-prediction modes at the encoder is to generate a resembling representation of the current block under test, called prediction (with samples s'). Therefore, the module in Fig. 2.4 is referred to as prediction generator. This prediction is generated using the neighbors of the block. The idea is that since the neighbors will already be present at the decoder, the decoder can recreate the current block using only the information of the current block sent by the encoder. Usually, number of bits for representing the current block is smaller than the number of information bits send

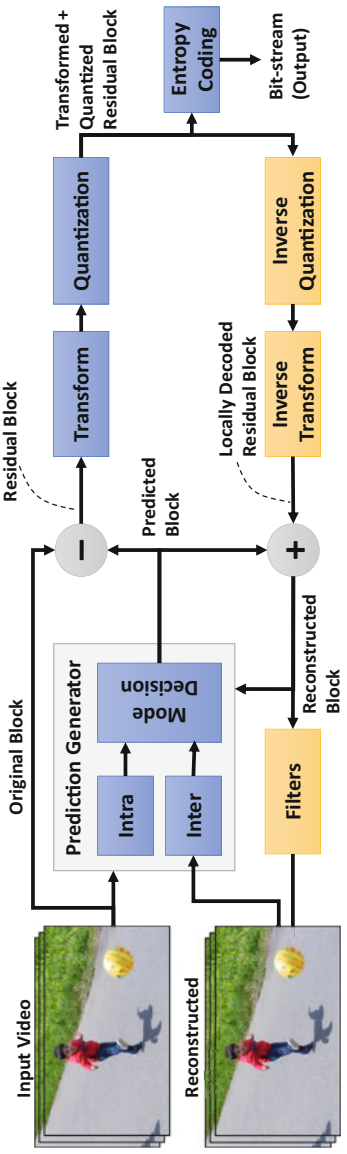

Fig. 2.4 Basic modules of a video encoder. The *blue* boxes are used for encoding purposes, and the *orange* boxes denote the local decoder modules used inside the encoder

by the encoder (and hence, compression is achieved). However, this technique only works with lossless video encoders, which do not provide higher compression rates compared to lossy compression. Therefore, in case of lossy compression, the encoder compresses the current block by using the neighborhood of the block in the reconstructed video frame samples. A reconstructed video frame is a video frame generated by decoding the information sent by the encoder. The encoder can also locally decode and use the locally decoded video frame for generating the predictons, i.e., the video frame samples which will be generated at the decoder, to make sure that there is no error accumulation at the decoder. Thus, the encoder also implements a local decoder (shown by the lower modules in Fig. 2.4). The generation of prediction either uses the spatially neighboring pixels (intra-frame processing) or temporally neighboring pixels (inter-frame processing). Thus, if the quality of prediction (its resemblance with the original block) is high, the residual block, i.e., the difference between the original and the prediction, has low pixel energy (samples with low values). The quality of a prediction is usually tested using the sum of absolute differences (SAD) metric:

$$SAD = \sum_{y=0}^{bh-1} \sum_{x=0}^{bw-1} \left| s(x,y) - s'(x,y) \right| \tag{2.4}$$

Here, (x, y) is the pixel location within the original and predicted blocks. Every intra- and inter-mode has its associated prediction and, therefore, a SAD value. It is evident that, if the number of predictions that are tested is high (i.e., a more complex encoder), probability of finding a prediction with a low SAD/pixel energy is high, which results in higher compression.

Afterwards, transformation of residue and its quantization take place. The purpose of transformation is to separate low- and high-frequency components of the residual bock and transmit a part of these components (hence, lossy compression). If high compression is required, the higher frequency components are not sent to the decoder and vice versa. This is because the human eye is more susceptible to the low-frequency image components than high-frequency components. Quantization further reduces the scale of the transformed values before finally the bitstream generation (i.e., the entropy coding module) forms a bitstream which is sent to the video decoder. The size of the bitstream is proportional to the pixel energy in the residual block, and more energy in the residual block will increase the size of the bitstream.

This bitstream consists of the transformed and quantized residue of the current block, along with the prediction mode selected for compressing the block. The decoder can reverse the process, by generating the transformed and quantized residual block (approximately) from the bitstream. The steps to generate the block at decoder are (a) decoding the bitstream to get quantized blocks and prediction mode, (b) inverse quantizing the received block, (c) inverse transforming the block, (d) generating the prediction using the prediction mode, and, finally, (e) adding the prediction to generate a representation of the original block.

In addition to video encoding/decoding algorithms (which attract high attention from research community), algorithmic and architectural designs of video input and

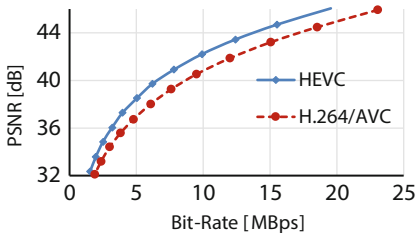

Fig. 2.5 Video quality comparison for H.264/AVC and HEVC

output pipelines are also a critical design step, since if a module within these pipelines becomes the bottleneck, it will hurt the performance of the complete system. Video input pipeline involves color space conversion, deinterlacing, downsampling, etc. More on this is covered in Sect. 5.1.2. Video output pipeline usually incorporates noise reduction algorithms, color space conversion, and content enhancement (like increasing the sharpness, correcting brightness, etc.).

2.2.1 H.264/AVC and HEVC

H.264 or MPEG-4 Part 10 or Advanced Video Coding (MPEG-4 AVC) is the current video compression industry standard developed by JCT-VC [2, 4]. In this book, this standard will be referred to as H.264/AVC. Here, a video frame is divided into blocks of 16×16 pixels called macroblocks (MBs), and each MB is treated as a separate entity for compression. A block is tested with intra-prediction modes using angular directions and inter-prediction via block matching (also called motion estimation (ME)) [5, 6]. However, due to increasing video resolutions and frame rates (e.g., 8K Ultra-HD, 7680×4320 pixels, at 120 fps), JCT-VC has also recently developed the next-generation High Efficiency Video Coding (HEVC) standard. HEVC aims at increasing the compression by 50% compared to the H.264/AVC. From Fig. 2.5, we notice that the bitrate of the video encoded via HEVC is lower than H.264/AVC, for the same video quality (PSNR). These curves are also called rate-distortion (RD) curves. For these curves, higher is better.

On the other hand, HEVC time consumption is substantially higher and our experiments show that HEVC is ~2.2× more complex than H.264/AVC. Compared to H.264/AVC, HEVC brings two major innovations [3, 7]. Firstly, unlike the concept of MB of its predecessor coding standard H.264/AVC, the HEVC introduces the concept of the coding tree unit (CTU) as a coding structure. A CTU is a square block of size 16×16, 32×32, or 64×64. The CTU is recursively subdivided into multiple coding units (CUs) and prediction units (PUs). Each PU serves as an independent, basic entity for compression carrying individual header data. Secondly, a prediction for a PU can be generated using one out of many standard-defined prediction modes.

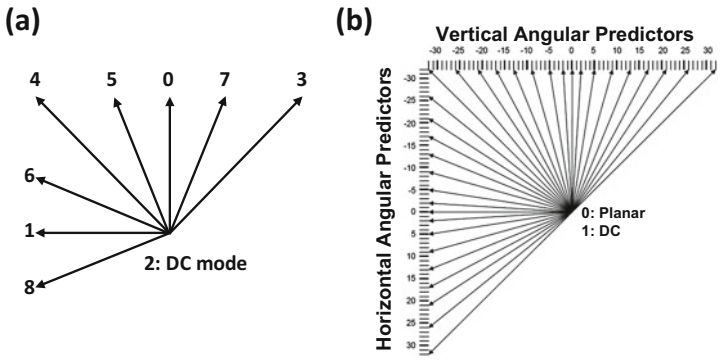

Fig. 2.6 (**a**) H.264/AVC and (**b**) HEVC intra-angular modes

Table 2.1 Comparing HEVC intra-prediction modes for 64×64 CTU with the H.264/AVC intra-modes for a 64×64 image region

	Total intra-angular modes n_{ang}	
Prediction size	HEVC/ H.265 (64×64)	H.264/AVC (16×16)
64×64	4	NA
32×32	35	NA
16×16	35	4
8×8	35	9
4×4	19	9
Total modes	$\psi = 7808$	16×(16×9+4×9+4) = 2944

2.2.1.1 Intra-prediction Modes

As discussed, H.264/AVC and HEVC intra-compression modes generate the prediction block for the current block using the spatial neighboring reconstructed video samples. This is because spatially, the texture flow of the video samples is expected to continue from the surrounding blocks into the current block under consideration. However, the proper direction of texture flow must be estimated for best representation of the original block, which will result in a residual block with low pixel energy. As shown in Fig. 2.6 (a, b), both H.264/AVC and HEVC employ different spatial directional (or angular) modes to determine the best texture flow direction. The encoders employ a brute-force (also called full-search) algorithm to get the best prediction mode, whereby all the predictors are tested to get the best predictor. The best prediction mode is usually chosen as the one which corresponds to the lowest SAD (Eq. 2.4). SAD value is computed between the current block and the prediction block.

Table 2.1 tabulates a comparison of intra-prediction modes for both HEVC and H.264/AVC. It is notable that the total number of prediction modes for the HEVC is significantly higher compared to H.264/AVC. Furthermore, the selection of

RD-wise best PU size and prediction mode is determined by a RD optimization (RDO) process. Therefore, due to the recursive nature of best PU size selection and RDO process, the total number of mode decisions in HEVC (computed using Eq. (2.5) for a CTU of size $b_w \times b_h$=64×64) is ~2.65× more than that in H.264/AVC. n_{ang} denotes the number of angular modes for a particular PU size (column 2 of Table 2.1).

$$\psi = \sum_{i=0}^{\log_2 b_w - 2} \left(2^{2i} \times n_{ang}\right) \tag{2.5}$$

2.2.1.2 HEVC Inter-prediction Modes

Similarly, for generating the inter-predictions, both H.264/AVC and HEVC encoders search for a similar block in temporal neighborhood (i.e., the previously encoded frames called reference frames) using ME. In HEVC, ME is repeated for every PU as shown in Fig. 2.7. This search is pixel-wise and uses the SAD metric. The ME process is shown in Fig. 2.8 along with the hardware architecture of computing the SAD. In the inter-encoding case, a predictor is a block of pixels of the previous frame. Usually, only parts of reference frames, called the search

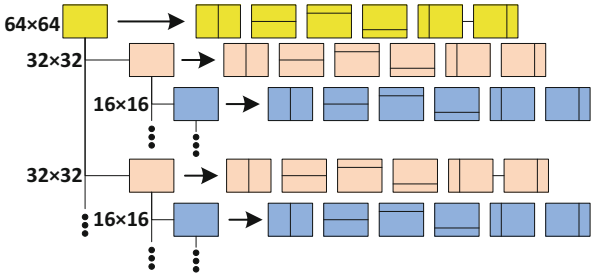

Fig. 2.7 HEVC inter-prediction modes

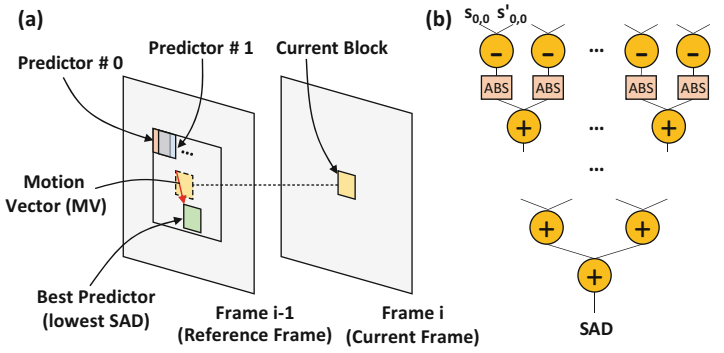

Fig. 2.8 Motion estimation (ME) and SAD computation architecture

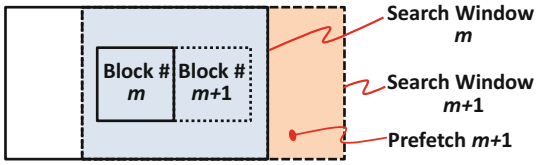

Fig. 2.9 Search window structure and window prefetching

window, are tested for the best predictor. The search window is brought from the external memory to the on-chip memory, and the predictors are drawn from this search window for computing SAD. Search window is a rectangular region of size $s_w \times s_h$. For fast prediction selection, ideally the search window should be loaded into the on-chip memory. Furthermore, note that blocks are processed in a raster scan order, and two neighboring blocks will have most of their search windows overlapped [5]. This concept is outlined in Fig. 2.9. Thus, search window "prefetching" of new reference samples can be done in parallel to the ME process. Further, a larger search window increases the probability of finding a predictor with low SAD value (and thus increases the video quality). However, this also increases the on-chip memory required to store the search window, which increases the power consumption of the system. Additionally, the external memory accesses also increase, resulting in higher power consumption [8, 9] and access latencies.

From the discussion above related to the search window, it is simple to infer that a single reference pixel is written multiple times to the search window, depending upon the height of the window. This is because the search window overlaps for the blocks in adjacent rows. A larger search window height (s_h) denotes that a single pixel will be written more times into the search window storage than having a smaller search window height. A read factor r_f is defined which denotes the total number of times a pixel in the reference frame is read and then stored in the search window. Typical value of this factor is between 3 and 12. For the technique given in Fig. 2.8, where the search window is shifted by b_h for every new row of blocks with size $b_w \times b_h$:

$$r_f = s_h / b_h \tag{2.6}$$

There are multiple ways to determine the best predictor within the search window. One method is to search every predictor in the search window. This scheme employs a brute-force or full-search algorithm and results in the best predictor with the lowest SAD value. However, this scheme is excessively complex and almost never employed in practice. Usually, fast prediction search algorithms (like EPZS, TZ [10]) are used, which result in considerably lower power consumption and complexity with little video quality loss, compared to the brute-force search. In these fast algorithms, not every possible predictor of the search window is tested (i.e., not all pixels of the search window are utilized). An additional step iteratively determines the next most probable predictors which will result in the lowest SAD value. However, this does not suggest that the number of pixels "prefetched" will reduce. The number of pixels fetched from external memory

will remain the same, because the pixels that will be utilized by the iterative step for ME are unknown. Moreover, if a part of the search window is not fully utilized for the current block, it might be utilized for the next adjacent block.

In HEVC, each CU is split into 13 PUs (see Fig. 2.7), and ME is carried out for these PUs. Mathematically, the total number of block-matching iterations for a square CTU of width b_w is given by:

$$\psi = 13 \times \sum_{i=0}^{\log_2 b_w - 3} 2^{2i} \tag{2.7}$$

Simulations show that block matching in HEVC takes around 80% of the total encoding time, and it is ~2.2× more complex than compared to H.264/AVC. In addition, normally the block matching takes about 60% of the total energy in video encoders [11]. Moreover, there can be multiple reference frames (e.g., bi-prediction in HEVC and multiview video coding (MVC)), and the best predictor is searched in a search window for each of these frames.

Many algorithms used in both H.264/AVC and HEVC can be applied in other video processing algorithms, as they exercise the same principles of computations and memory access. Many mainstream video encoders use similar principles of intra- and inter-video encoding (e.g., Google's VP8 and VP9, Microsoft's SMPTE, Cisco's Thor, MJPEG). Moreover, ME algorithm is also used in super-resolution techniques (increasing the resolution of a video frame by concatenating multiple, temporally neighboring frames), temporal frame interpolations (for reducing flicker and algorithmically increasing FPS), motion tracking, corner detection, etc.

Further, almost all video processing algorithms have configuration knobs, which can be tweaked to leverage the computational complexity against video quality. For example, the search window for motion estimation in video coding, denoising, frame interpolation, and super-resolution algorithms can be enlarged or contracted. The number of intra-angular modes tested for HEVC can be increased or reduced. This suggests video processing applications provide prospects for runtime adaptation of workload. However, such adaptation for increasing the throughput might result in lower output quality, e.g., loss in PSNR or precision of tracking.

2.2.2 *Parallelization*

Like all compute-intensive video applications, H.264/AVC and HEVC standards allow for parallel operation. A system designer can utilize a multi- or many-core system and exploit the parallelism in order to gain computational advantages. To process a video sequence, a hierarchical approach for video partitioning is employed. Video frames are divided into sets of frames, called groups of pictures (GOPs). Each GOP can be processed independently of other GOPs, supporting GOP-level parallelization [12]. Within a GOP, video frames can also be processed in parallel on independent processing cores [13]. A frame is divided into slices or

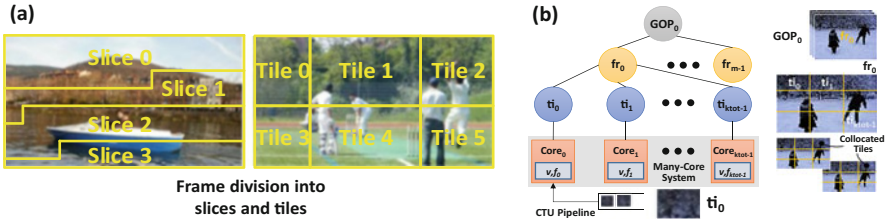

Fig. 2.10 (**a**) Frame division into slices and tiles for parallel processing, (**b**) video partitioning hierarchy. Here, *fr* frame, *ti* tile

tiles (see Fig. 2.10a), and each slice/tile can be processed in parallel [14–16]. As discussed before, usually the image and video processing algorithms are block-based algorithms, where a set of pixels in a rectangular region is considered as a basic processing entity. Each slice/tile is divided into blocks, and a single block is processed one at one time (the blocks within the slice/tile are processed sequentially). However, intra-angular predictions and inter-predictions can be tested in parallel. An example video partition hierarchy is given in Fig. 2.10b, where each frame in the GOP is divided into k_{tot} tiles. This figure also shows collocated tiles, which are the video tiles at the same location but in adjacent frames.

Video Workload Balancing Via parallelization, the workload of the whole application is divided into chunks, and thus the workload corresponds to the computational complexity and energy consumption of a processing core. That is, the larger the workload, the more computational complexity and energy consumption of the underlying hardware (processing resources like cores, FPGAs, GPUs, etc.) are expected. Generally, the workload of the complete application should be distributed to the physical resources in a manner that the hardware utilization is maximized. Therefore, every resource should ideally consume the same amount of time processing their assigned jobs or subtasks. Collectively, this process can be termed as workload or load balancing. Workload balancing will increase the throughput of the system along with increase of the throughput-per-watt metric.

Workload balancing can be achieved via centralized or distributed techniques [17, 18], which gather statistics and objectively distribute the subtasks among the resources. However, for fully distributed strategies, optimal scheduling decision is difficult to make due to rapidly changing environment, randomness, and unpredictability. The communication delays in fully distributed strategies can also invalidate a correct decision at a previous time, as the state of the system might change rapidly after the distributed load balancing decision. Further, distributed algorithms reduce the range of subtask migration from one core to another, because physically far apart resources, with highest and lightest workloads, cannot be balanced in a single iteration of load balancing. Also, the overhead of the technique increases with the system size due to increased message passing, control logic, scheduling algorithms, etc. Further, the response of distributed strategies saturates after tens of compute nodes. For the centralized strategy of load balancing, the overhead of the assignment algorithm is large and has a large communication

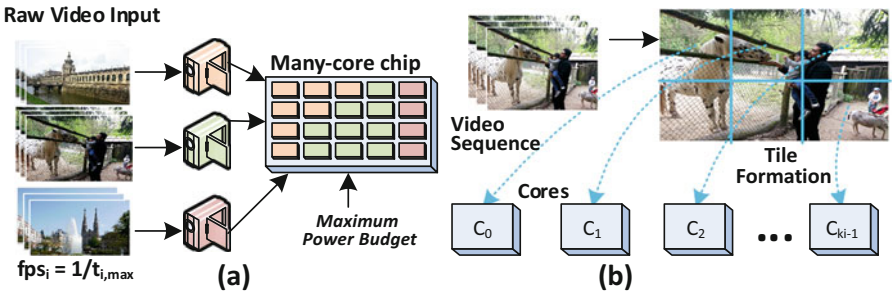

Fig. 2.11 (**a**) Multi-video capture and encoding on a many-core chip, (**b**) frame division into multiple tiles

and storage overhead. This is due to the chip-wide information gathering and storage of statistics. Further, centralized load balancing algorithms are vulnerable to faults, and a complete system failure will occur if the central node (running the subtask assignment algorithm) fails. Thus, both centralized and distributed strategies for load balancing do not always produce optimal performance. This discussion not only is applicable for video processing systems but also applies to any many-core system running parallel workloads.

Multicasting Multicasting refers to information transfer to a set of predefined destination nodes. With reference to video communication, a multicasting or multichannel video encoder should be capable of encoding concurrent [19, 20] video streams in parallel and generating appropriate compressed bitstreams for each of these videos, as shown in Fig. 2.11. Use cases of multicast video encoding include security, entertainment, video logging, etc. Each video can have its own resolution, texture/motion content-dependent workload, and throughput requirement like FPS. This system can be implemented using a many-core platform or by designing custom hardware. In case of a many-core system, it is a design challenge to efficiently distribute the processing cores and power among multiple, concurrently executing encoders, while fulfilling their throughput requirements. Moreover, the load balancing techniques must be applicable to such paradigms. For a custom hardware solution, some of the challenges for the multicast encoder are being area efficient (due to multiple encoders working in parallel) and being able to efficiently access video data.

2.2.3 DVC and HDVC

As discussed, the high compression efficiency of H.264/AVC and HEVC has enabled a wide range of applications under low-bandwidth constraints. They follow the predictive video coding (PVC) model, where the predictions of the block under consideration for compression are generated at the encoder side (using intra-angular

modes or ME). PVC is typically well suited for circumstances where encoding devices have large-enough computational power, while decoders are resource-/power-constrained devices, like mobile devices engaged in video streaming, battery driven hand-held phones etc. Similarly, if the video is encoded once and transmitted to numerous decoders multiple times, then PVC approach is most suitable.

2.2.3.1 Distributed Video Coding

It is possible that the computational complexity of a video application is too high to be feasible for a particular system. For example, the significantly increased computations at the HEVC encoder prohibits its use in constrained encoded scenarios. Distributed video coding (DVC) has been emerged as an attractive solution for scenarios where the encoding devices are resource constrained and must use low-complexity encoding (like infield wireless video sensor nodes, small autonomous flying robots, mobile devices with low processing capabilities, etc.), while the decoding devices have high computational power (like high-end servers) and can execute high-complexity decoding tasks. DVC paradigm provides means to shift/offload the computational workload from the video encoder to decoder [21–27].

A DVC encoder typically consumes only 7% of the total power consumed by a PVC H.264 encoder [28–30]. In DVC paradigm, the decoder performs the ME for frame interpolation, extrapolation, and upsampling. Only a subset of a GOP is transmitted by the encoder, and the decoder generates the complete GOP by exploiting the inter-frame correlation. At the DVC decoder side, Slepian-Wolf decoder and ME with interpolation contribute 90% towards the total decoding complexity. To improve the estimation quality, the DVC decoder requests auxiliary information from the encoder (i.e., parity bits generated by turbo coding), which results in a higher transmission power compared to the PVC case (see Fig. 2.12). A DVC encoder may require ~4× higher transmission energy compared to an H.264 PVC encoder for a video sensor node [28]. Increase in parity bits positively influences the video quality at the decoder side, but results in higher transmission energy at the encoder side.

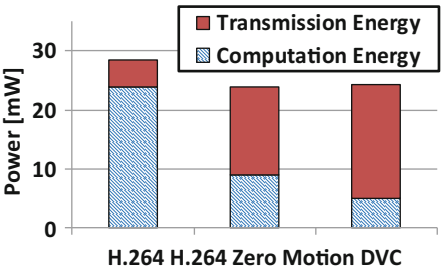

Fig. 2.12 Comparing the computational and transmission power for an ASIC-based video sensor [82]

2.2.3.2 Hybrid Distributed Video Coding

DVC completely offloads the ME from the encoder to the decoder. The major drawback of DVC is lower video quality compared to that provided by PVC. On the other hand, PVC results in best video quality at the expense of high computational complexity at the encoder side. PVC and DVC become power-/energy-wise inefficient in scenarios where both encoder- and decoder-side devices are resource constrained and/or subjected to runtime varying conditions of available energy levels and computational resources. In these cases, only one or neither of the encoder-/decoder-side devices has adequate computational and/or transmission power to deliver the required throughput and/or video quality. Examples of such scenarios are (1) collaborative distributed video sensor networks for smart energy-aware surveillance; (2) mobile devices on Internet of Things (IoT) – with varying battery levels – communicating with each other or other power-constrained devices; (3) heterogeneous devices from different vendors with distinct energy consumption properties, etc. Besides, DVC may not facilitate complete offloading in scenarios where multiple encoding devices concurrently offload their computational workload to a single, shared decoding device [31].

To cope with the energy-related issues for video coding in such dynamic scenarios, Hybrid Distributed Video Coding (HDVC) has emerged as an attractive solution which combines the positive aspects of both PVC and DVC, i.e., providing high video quality close to PVC and low computational power close to DVC. In HDVC, the decoder-side ME complexity is relaxed at the cost of partial ME at the encoder side. This means that the encoder also fully processes some of the video frame blocks and leave the rest to the decoder. The partial ME at the encoder side results in better reconstruction of frames at the decoder side that corresponds to a high video quality and low energy consumption at the decoder side.

Concisely, general workload offloading improves the performance if [32]:

- The throughput requirement of the (video) application is high and is not sustainable by the video processing device.
- The (DVC/HDVC) decoder/receiver is fast and results in computational/power benefit if the jobs are offloaded to the decoder.
- A small amount of auxiliary bits is transmitted.
- The bandwidth of the channel between the encoder and decoder is not a bottleneck.

2.3 Technological Challenges

In the previous chapter, we briefly discussed some of the technological challenges that might arise while implementing a system with high throughput-per-watt metric. Here, details about these challenges are given.

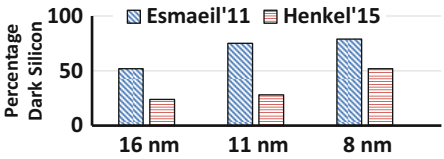

Fig. 2.13 Prediction/trends of dark silicon. See [8] for Esmaeil'11 and [164] for Henkel'15

2.3.1 Dark Silicon or Power Wall

Reduced transistor sizes in the modern fabrication technologies have led to new unforeseen challenges for system designers. Though the chip designers are living up to the Moore's law challenge, it is expected that most of the transistors etched on the chip will not be completely utilized due to the power wall problem. Specifically, the failure of Dennard scaling has resulted in the emergence of the Dark Silicon Age, where the chip's real estate cannot be utilized continuously, at full capacity. In fact, current estimates (as shown in Fig. 2.13) suggest that only 30–50% of the chip's usable resources will be bright (fully utilized) for 8 nm technology, while the rest will be kept dark (unutilized) or dim/gray (partially utilized or underutilized). This forced underutilization arises from the fact that power per unit of area is increasing monotonously with increasing transistor density. Therefore, the temperature of the chip may reach levels which will not be contained by the available state-of-the-art coolants and result in permanent damage of the chip. Thus, power-efficient designs are of prime importance for modern systems.

A digital circuit consumes two types of powers, dynamic (p_{dyn}) and static (p_{sta}) power. The dynamic power is due to switching the transistors *on* and *off*, whereas the leakage power is a result of subthreshold current through the transistor's channel and the leakage through the transistor's gate, when the transistor is *off*. The static power can be reduced by lowering the supply voltage v_{dd}. On the other hand, for a CMOS circuit, the dynamic power consumption can be written as:

$$p_{dyn} = \alpha \cdot c_p \cdot v_{dd}^2 \cdot f \tag{2.8}$$

Here, α is the switching activity level, c_p is the capacitance of the circuit, v_{dd} is the supply voltage, and f is the clock frequency at which the circuit is operated. This shows that dynamic power can be reduced by reducing the supply voltage. However, reducing v_{dd} will increase the time delay (t_d), and, therefore, the frequency must be reduced. In fact, t_d and v_{dd} are related by the following equation:

$$t_d \propto \frac{v_{dd}}{v_{dd} - v_{th}} \tag{2.9}$$

Here, v_{th} is the threshold voltage. Therefore, we can rewrite Eq. (2.8) as:

$$p_{dyn} \propto v_{dd}^3 \propto f^3 \tag{2.10}$$

Using this relationship, dynamic voltage frequency scaling (DVFS) can reduce the power of the circuit. However, note that reducing the frequency of the circuit also reduces the throughput and may result in deadline misses. Moreover, as discussed above, the voltage and frequency of a processor scale together. However, this relationship does not hold for the complete frequency range, since there is a certain threshold below which the voltage reduction results in unstable processor behavior [33]. Thus, only dynamic frequency scaling (DFS) can be employed in these scenarios.

For video applications, a strict throughput constraint usually exists that must be met. Current trends for user-demanded frame resolution and FPS suggest that this throughput demand is increasing (Equation (2.1) and hence puts pressure on the hardware to perform. However, new fabrication technologies – with about half of the cores turned OFF due to dark silicon constraints – require careful consideration of available TDP and its distribution among possibly multiple, multithreaded video applications competing for system resources. Moreover, memory may consume in excess of 40% of the total chip's power [8]. This power includes access to external memory, read/write energy consumption, and leakage/standby power. The memory power consumption is especially of concern for video applications that are memory intensive, and their memory requirements continue to grow with the growing throughput demands. Therefore, it might not be possible to achieve higher power efficiency without considering the memory power.

The discussion above suggests that the software and hardware must be power efficient to exercise the minimum amount of power, while fulfilling the throughput requirements. This will not only address the dark silicon issue but will provide supplementary power to other parallel running applications. Further, the parallelization potential of a video application can be exploited to meet the throughput requirements and ideally distribute the compute power along the complete chip. Also, hardware accelerators, running at a lower frequency/power but generating a higher throughput than its software counterpart, can be strategically placed on the die to reduce the chip's temperature. Additionally, the advantages of the new memory technologies (like MRAM) can be exploited to replace the on-chip SRAM, in order to reduce the leakage power of the memory subsystem and limit the access to the external memory due to their higher density/sizes.

2.3.2 NBTI-Induced SRAM Aging

To provide fast read/write accesses, application-specific architectures normally employ dedicated SRAM-based on-chip memories (like scratchpads instead of caches) for storing data, thus saving the extra power overhead of tags and other supportive circuitries. These on-chip memories are managed using specialized

address generation units and/or explicitly programmed to exploit applications' attributes (more on this later). However, due to continuous technology scaling resulting in small feature sizes, high-power densities (dark silicon paradigm), and resulting temperatures, SRAM-based on-chip memories are exposed to various reliability issues like transient errors (soft errors) and permanent errors (device aging). Memory-intensive video applications have flourished into various mission-critical domains like surveillance and security, automotive, satellite imaging and video transmissions, sensor-based image/video processing over long durations, etc. For these applications, reliable operation over their lifetime or an extended lifetime is an imperative system requirement.

This book considers SRAM aging due to NBTI, which has emerged as one of the most critical reliability threats for the new fabrication technology. NBTI occurs in PMOS transistors due to negative voltage at the gate (i.e., $v_{gs} = -v_{dd}$). This voltage results in interface traps because of the breakdown of the Si-H bond at the Si-SiO_2 interface. This manifests as a surge in threshold voltage and reduction in noise margin (i.e., short-term aging) that may lead to timing errors/delay faults and/or runtime performance degradation. To encounter this threshold voltage increase (more than 50 mV [34]), the clock frequency of the device must be reduced by more than 20% over its lifetime. However, due to rising NBTI issues and cost/power/performance constraints, the degradation of the cell stability can no longer be addressed by simply providing a design time delay margin [35]. This aging-based phenomenon is partially reversed in the so-called recovery mode (Si-H bond is reformed in a few cases) once the stress is removed from the PMOS gate, i.e., at v_{gs}=0. An abstract view of this process is shown in Fig. 2.14a for a PMOS transistor. Such a situation occurs when a "zero" overwrites the "one" stored in the SRAM cell and vice versa. However, 100% recovery is not possible and NBTI results in continuous degradation over years (i.e., long-term aging). The total aging throughout the lifetime depends upon the stress and recovery cycles; see Fig. 2.14b. For ease of discussion, this book defines the duty cycle (Δ) as the percentage of a cell's lifetime when the stored value is "one."

This book considers a memory composed of numerous 6T SRAM cells. Each cell is composed of two inverters to store a bit value (see Fig. 2.14c), and these inverters store complementary values at all times. The word line (WL) is enabled to write a value, while the bit line (BL) is used to deliver data to be stored in the cell.

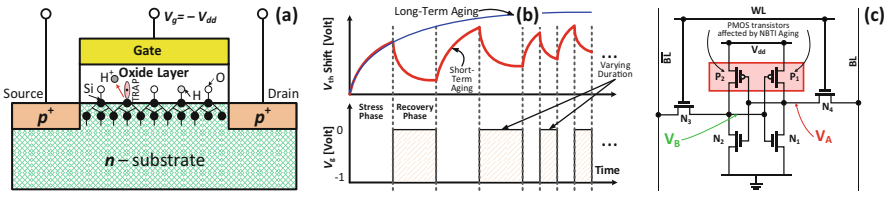

Fig. 2.14 (**a**) PMOS transistor demonstrating NBTI aging with breaking Si-H bond at the Si-SiO_2 interface, (**b**) example of stress and recovery phases for a PMOS transistor, and (**c**) a standard 6T SRAM cell

The data is retained in the cell by turning off access transistors. To read data, WL is set high and the BL value is retrieved. Both these transistors are in complementary states at all times. In case a "zero" or "one" value is stored in an SRAM cell, one of its PMOS transistors will be under stress and the other in the recovery phase. Since the aging of an SRAM cell is determined by the worst-case aging of one of the two PMOS transistors, the overall lowest aging is achieved when both PMOS transistors are stressed by the same amount of time during the whole lifetime. That is, an SRAM cell contains "one" value for 50% of its lifetime. This corresponds to a duty cycle (Δ) of 50%. In short, SRAM aging depends upon the duty cycle of the transistors in an SRAM cell. If the duty cycle is balanced (which denotes Δ=50%), the aging rate of SRAM cell will minimize.

Note that NBTI is not the only deteriorating mechanism active in SRAM cells. Hot carrier injection (HCI, which mainly degrades the NMOS transistor) is another aging mechanism which injects high-energy (hot) carriers inside the gate oxide. This causes interface traps and thus introduces threshold voltage shift [36], which is presented by the following equation:

$$\Delta v_{th} = \psi_{HCI} s_a f e^{\frac{v_{dd}-v_{th}}{d_{ox}\psi_1}} t^{0.5} \tag{2.11}$$

In this equation, ψ_{HCI} and ψ_1 are aging rate-dependent constants, s_a is the switching activity, and d_{ox} is the oxide thickness. This equation shows that a higher switching activity will increase the HCI-induced aging rate and vice versa. On the contrary, higher switching activity will reduce NBTI. Studies like [37, 38], however, emphasize that NBTI has a greater impact and is the dominating factor in limiting the life of a circuit.

2.3.3 Other Challenges

Some other challenges, which arise due to recent fabrication technology advancements, are enabling high instruction-level parallelism (ILP) and having a high memory bandwidth. Sometimes, they are referred to as ILP wall and memory wall. ILP wall makes it difficult to parallelize the instruction streams in order to keep the resources busy, and, thus, it prohibits the increase in throughput. Further, the reduced bandwidth between the caches/external memory and the computing resources (e.g., CPUs) is a performance-limiting factor contributed by the so-called memory wall. However, this book will only focus on power wall and SRAM aging.

2.4 Related Work

2.4.1 Video System Software

This section presents state-of-the-art techniques for addressing different challenges of a video system at the software layer. These issues include parallelization, complexity reduction, and power/resource budgeting.

2.4.1.1 Parallelization and Workload Balancing

As previously discussed, parallelization is a fundamental requisite of high-complexity video applications, which must be exploited on many-core systems, possibly having hardware accelerators and custom logic/interfaces. The general classification of parallelization and workload mapping practices on many-core systems is presented in the literature [39]. However, one of the objectives of this book is to present and analyze the application-specific properties for workload mapping on a many-core system and improving power efficiency of the video system. Numerous works are reported to enable parallel processing of video applications. These works include parallel video coding/decoding [40, 41], tracking [42], image/face recognition [43, 44], nonnegative matrix factorization [45], and others. However, these works generally do not consider resource management, hardware characteristics, and workload balancing.

Workload Balancing General load balancing of compute jobs among compute entities is presented in [46, 17, 18, 47]. For power efficiency (i.e., increased throughput-per-watt ratio), either the load of an application can be dispersed on a given platform (load balancing [48, 49]) or a platform can be synthesized for the given load/application (load-driven synthesis [50, 51]) under throughput and/or power constraints. Most of the load balancing techniques consider homogeneous many-core systems, jobs with almost equal complexity and do not consider load variation at runtime [46, 52]. For example, [47] considers load balancing for distributed stream processing applications in wide-area environment, under dynamic resource consumption. In [53], load balancing between mirror multimedia servers is discussed for both centralized and distribution load balancing techniques. Ref. [46] deals with assigning each resource (core) with equal number of subtasks and reaches an equilibrium state if no more jobs can be migrated from one core to its physical, homogeneous neighbors. However, the present architectural and physical challenges for homogeneous many-core system design are not considered. Smaller feature sizes result in physical variability of underlying transistors (also called process variation), which transforms into variable leakage power and maximum frequency achievable for the homogeneous cores on the same die [54]. Thus, compute cores can have different characteristics, even though they form a homogeneous many-core system. Research has also focused on combining the

distribution and balancing of load and DVFS and dynamic power management (DPM) of the underlying hardware resources [55]. Ref. [56] defines a single clock frequency for the entire chip for maximum efficiency, whereas [16, 57] independently determine the frequency of each core while distributing application load. Authors in [16] target minimizing the power consumption for a fixed deadline, while [57] tries to maximize the throughput of parallel running, multithreaded applications for a given power budget.

Workload Balancing on Heterogeneous Nodes Further, to increase the throughput-per-watt under modern system design challenges, heterogeneous multi-/many-core systems are becoming progressively popular [33, 58]. Using architectural heterogeneity, it is now possible for the designer to schedule a processing job on a compute node (e.g., a core, a hardware accelerator) that will increase the throughput-per-watt metric [59]. This way, maximum power efficiency is achieved. For example, ARM big.LITTLE architecture [60] integrates high-performance Cortex-A57 big cores with low-power Cortex-A53 little cores, in order to achieve maximum throughput-per-watt by capitalizing on adaptive application mapping techniques. Thus, general load balancing methods are not applicable in heterogeneous paradigms having cores/compute nodes with unequal compute capabilities. Parallelization and load balancing of H.264/AVC is carried out in [61], using heterogeneous CPU+GPU systems [62, 63], without considering the impact of power which is substantial when GPUs are used. In [33], authors target energy-efficient workload allocation and voltage-frequency tuning of the underlying single-ISA computing nodes. The goal is to minimize energy/power of the system. However, their approach does not consider fulfilling the required throughput of the application(s). In [64], authors propose to identify the program and cores' characteristics and then appropriately match them for scheduling. Reference [65] studies parallelized database on heterogeneous, single-ISA architectures. These proposed workload balancing approaches do not consider modern fabrication technological challenges like power budgeting, dark silicon, etc. Moreover, generally the complexity of each subtask is not equal. For example, ME can have considerable different complexity for different video frame blocks, depending upon the content properties and texture within the block [66, 11]. Further, the cache behavior of the application and the physical locality of the compute node (e.g., its distance from the external memory controller) also determine the complexity of a subtask. All these challenges must be addressed if an efficient workload mapping and balancing policy needs to be implemented. This is especially true for video systems under throughput constraints.

Parallelization of Video Systems Multiple parallelization methods for video systems are available in the literature. For example, a many-core based SIMD implementation of H.264/AVC is given in [67], but it does not consider workload mapping and balancing. Using partial frame-level parallelism, a 12-core system for parallel HEVC encoding is given in [40]. Ref. [41] discusses a technique to parallelize H.264/AVC on Cell multiprocessor. In [68], a hierarchical parallelization of H.264/AVC is presented for low-cost cluster of cores, by

combining multilevel parallelism. In [69], a parallel implementation of particle filter on shared memory architectures is given. In [70], a hardware/software partitioning is targeted for a heterogeneous processor, for MPEG-2 encoder. Ref. [71] proposes a parallel implementation of the nonlocal means filter (NLMF) on a GPU for denoising 3D data. Parallel video super-resolution methods are proposed in [72, 73]. In general, not all these parallelization techniques consider workload balancing on the many-core system, power reduction, and meeting the throughput requirements. In addition, these techniques might require access to multiple video frames (in the external memory) at the same time, increasing the latency of the application. Thus, they either increase the power consumption of the system by needlessly increasing the core frequency beyond requirement or reduce the throughput and increase latency by burdening each compute resource with divergent workloads.

H.264/AVC parallelization and workload balancing are also discussed in the literature. Authors in [14] present a history-based technique to allocate the number of slices per frame dynamically, for balancing workload among the multiple cores. In their technique, each slice is mapped to a single compute core. A similar history-based technique can be found in [15] where the skipped video frame blocks determine the slice boundaries for parallel encoding. A two-pass slice partitioning technique for workload balancing of H.264/AVC is given in [74], where each frame is preprocessed, prior to being assigned into slices. However, no adaptation of workload and frequency of the cores (and thus, of power consumption) is proposed for these techniques.

2.4.1.2 Power-Efficient Video Processing Algorithms

A multitude of works in reducing the complexity of computationally heavy image/video applications also exist in the literature. In a nutshell, by sacrificing a small amount of output quality (e.g., a reduced PSNR or accuracy of tracking, increased bitrate), the workload of the application is curtailed to meet the throughput.

Numerous complexity reduction techniques exist in the literature for HEVC encoding [75–78]. The work in [75] (basically inspired from [79, 80]) presents a gradient-based fast intra-mode decision for a given PU size and results in about 20% time savings. In [76], authors have also presented a fast PU size selection algorithm for inter frames (exploiting temporal correlations for frame compression, similar to open loop for H.264/AVC presented in [81]). A divide-and-conquer strategy for choosing the best intra-angular prediction is given in [77]. First, eight equally spaced modes (at a distance of 4 in both directions; see Fig. 2.6b) are tested. Afterwards, six best modes with a distance of 2 are tested. In the end, two best modes are left which are tested with a distance of 1 to select the best mode. However, the number of modes selected for RD-cost determination is static and fairly large. Similarly in [78], to reduce the total number of intra-prediction modes tested for selecting the best predictor, an open-loop technique is utilized. Using the current pixels instead of reconstructed pixels, the total number of predictors is

reduced from 35 to 9. These nine modes are used for computing the rate-distortion (RD) cost. In [82], the video frame block (CTU) is downsampled and then texture-complexity (via variance) is computed, to determine the appropriate PU sizes for best encoding. Reference [83] exploits the correlation of intra-prediction modes of the current block with the final predictors of the neighboring blocks for determining a highly probable intra-prediction mode for the current block. An edge-based intra-prediction candidate selection technique is given in [84] to reduce the total number of modes tested by 73% and a time reduction from ~8% to ~32%. The 4×4 pixels in each PU are treated for determining the principal direction, and a set of nine intra-predictors is used for testing. However, the selection and truncation of intra-prediction modes is not adaptive. Similarly, numerous works exist to reduce the complexity and energy consumption of the inter-prediction engine (i.e., ME engine). Techniques to reduce the total number of operations in ME are also widely studied and employed [85, 10, 11]. For example, in [86], authors have proposed a technique to reduce the off-chip memory accesses for ME, which results in 56% memory access reduction. However, their technique is tested for a very small search window size of 16, which is not recommended to be used for large resolution sequences.

Other power-efficient techniques for H.264/AVC (e.g., [87, 88]) may not be directly or efficiently applicable to new video encoders like HEVC, due to the novel CTU structure of HEVC and nature of its angular prediction modes. Moreover, these techniques usually do not jointly consider power efficiency and workload balancing and do not exploit the speedup achieved via parallel encoding on a many-core platform. Further, these techniques do not consider the underlying platform properties (i.e., do not exploit the opportunities provided by the hardware) and the new challenges introduced by the reduced feature sizes, while managing their workload.

In addition to video coding, there is a plethora of other video processing algorithms, where complexity knobs are tuned at the cost of output quality. For example, see [89, 90]. Libraries like "open-source computer vision" (OpenCV [91]), and standards like OpenVX [92], provide plentiful implementations of these video algorithms.

2.4.1.3 Mitigating Dark Silicon at Software Level

The purpose of these techniques is to limit the maximum temperature or maximum power (TDP) consumed by the system at the software level. The abovementioned software-level techniques (i.e., parallelization, complexity reduction, budgeting, and offloading) implicitly address the dark silicon challenges. For realizing these techniques, mainly two distinct approaches are employed:

- Dynamic thermal management (DTM), which involves adjusting the voltage-frequency or power of the cores (DVFS [93]) and even severing the power to the

compute resources (via power gating, also called dynamic power management (DPM))
- Tasks/thread/workload migration, which involves migrating the subtask from one compute resource to another, in case the former's temperature becomes critical [94]

The frame-based energy management technique for real-time systems [95] exploits workload variations and the interplay between DVFS and DPM, for a real-time embedded application. It determines the optimal voltage-frequency setting and power levels of the devices to minimize the system's energy. In [96], a trial-and-error-based centralized algorithm determines the appropriate DVFS settings for the many-core system, to maximize application speedup under chip's power constraint. A thread mapping methodology under the constraints of thermal safe power (TSP) is presented in [97]. The authors argue that TDP is very conservative, and power more than TDP can be pumped into the chip to speed up execution if intelligent task-mapping decisions are made. The work in [93] proposes PID controller-based mechanisms at runtime for efficient utilization of the TDP budget, in order to maximize performance of architecturally heterogeneous cores synthesized with different power and performance targets. However, [93] does not target power budgeting among multithreaded applications with thread-level workload variations. In [98], a two-level closed-loop power control technique is given. Using voltage/frequency islands on a chip, the power distribution is divided among the compute resources. However, fine-grained power distribution and configuration selection using this technique are not possible, which are important for multithreaded applications with varying workloads. Furthermore, the closed-loop control usually responds slowly, causing performance issues for applications with abrupt workload changes. Single thread-based power budgeting is discussed in [99], which is not applicable to modern multithreaded applications. PEPON [100] also presents a two-level power budgeting technique to maximize performance within an allocated power cap. The technique in [57] targets control-based chip-level and application-level power budgeting, while accounting for the critical threads of the application.

Most of these techniques do not consider assigning compute resources to the applications, and the varying workload of the applications at runtime, which might require readjustment of the resource allocation. Moreover, these techniques do not target power allocation to dependent applications or subtasks of a single application, where the critical application or subtask will reduce the throughput of the complete system. These works also ignore the tuning characteristics of the applications and the opportunity they might provide for increased throughput-per-watt. Further, applications with throughput constraints must meet their deadlines (e.g., multimedia applications having soft deadlines and mission-critical applications with hard deadlines), which is usually not addressed by these works; rather, speedup is the optimization target. This poses additional challenges if the system load (e.g., due to parallel running applications, delay in delivering Ethernet packets) may change at runtime.

In summary, most of these techniques do not exploit the opportunities provided by the following:

- Selecting appropriate operating modes (configuration of different variables) of the applications and dark silicon
- Determining the right compute resources at nominal frequency/voltage settings even for unadaptable applications

Collectively, the operating modes of the application and the dark silicon provide opportunities of having multiple power modes [101], for instance:

- Powering on more cores at lower frequency to facilitate more applications or applications with high thread-level parallelism
- Powering on less cores at higher frequency to facilitate high instruction-/data-level parallelism
- Choosing appropriate operating modes of the applications to enable higher throughput at the same power budget

Mostly, DTM only gathers system statistics and manages the system stack excluding the application layer. Since these techniques do not take the application-specific characteristics into consideration, therefore, they lack power efficiency in cases of abrupt workload variations and/or when multiple threads of different applications (especially with mixed workload characteristics) are competing for the power budget. Hence, fine-grained power distribution and configuration selection is not possible.

This book will only focus on DTM-based techniques for power management of video applications, and runtime workload migration techniques will not be considered.

2.4.2 Video Systems Hardware

This section introduces the state-of-the-art approaches that target design and implementation of video system hardware architecture to address multiple challenges.

2.4.2.1 Efficient Hardware Design and Architectures

Numerous state-of-the-art techniques exist for designing compute- and power-efficient video systems. For encoding HD videos, new methods and tools to administer the necessary data processing tasks and dependencies of video encoders are required. For example, [102] discuss an H.264/AVC intra-encoder chip operating at 54 MHz clock. However, it is only capable of handling a small throughput requirement (720 $\times$ 480 4:2:0 video at 30 fps). Additionally, the authors use a parallel structure for generating the predictors that increases silicon area footprint. In [103], a fast prediction selection preprocessor, based on spatial domain filtering,

is proposed. A four-stage pipeline for edge extraction increases the latency of the design, and processing one video block requires 416 cycles at a maximum possible clock rate of 66 MHz. The design in [104] presents a 1920 × 1080 (1080p) intra-encoder, providing 25 fps at 100 MHz. It takes about 440 cycles to compute the intra-predictions. In [105], a 4K UHD (3840 × 2160) resolution at 60 fps intra-prediction architecture is presented, which replicates hardware for high throughput and needs to run at 310 MHz to achieve 4K UHD while still using suboptimal prediction selection methods. In [79], authors propose a fast method of selecting the best prediction, based on the texture flow. Authors in [106] present a low-latency 1080p, 61 fps intra-encoder architecture operating at 150 MHz. However, the proposed design tests the predictions in parallel and takes about 300 cycles to encode one block. Reference [81] presents a so-called open-loop (OL) method to determine the most likely predictor based upon the original image pixels rather than the reconstructed pixels. Similarly, a multitude of novel architectural designs for HEVC are also available. In [107], the authors proposed an HEVC intra-prediction HW for only 4×4 blocks. In addition to video coding, several other multimedia processing systems are realized using efficient architectures, for example, deblocking filters, AES, and CRC [108, 109].

Motion Estimation Many approaches to reduce the energy consumption of the video processing systems target the ME engine for optimization, because ME (also called block matching) is the most time- and energy-consuming process of a video encoder. In [110], ME energy is reduced by dropping the supply voltage and then employing error resiliency features. This results in energy savings of up to 60% on 130 nm CMOS technology. However, this technique results in extra control and noise-tolerance circuitry and worsens the output video quality as well. Reference [111] lessens the search space for ME and thus avoids redundant memory accesses. However, an unnecessary brute-force full-search algorithm results in high computation effort. In addition to the technique's failure to account for sudden motion, their results are for CIF/QCIF videos which already require a small search space for ME. In [112], the on-chip memory is replaced with a cache. Further, they do not explore the opportunities by reducing the leakage energy (which is dominant for submicron technology [113]). The work in [114] reduces the external memory accesses by frame buffer compression, but requires additional computations. In [115, 116], different data reuse schemes for reducing the external memory accesses by video processing algorithms are categorized from levels A to D, with highest latency (smallest on-chip memory) to the lowest latency (largest on-chip memory). There are other extensions of the originally proposed levels, like level C+ [5]. Reducing the total number of predictions (ME operations [85, 10]) also decreases the latency of execution but has little improvement for memory energy consumption and external memory access (see details in Sect. 2.2.1.2).

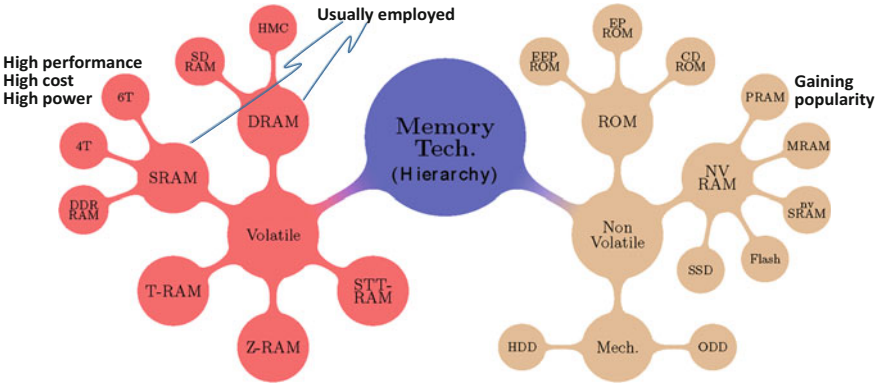

Fig. 2.15 Different memory technologies and their design hierarchy

Table 2.2 Abstract comparison between NVM and VM memory technologies

Technology	Access Speed		Power Consumption			Density	NVM
	Read	Write	Dynamic		Leakage		
			Read	Write			
SRAM	HH	HH	L	L	HH	LL	No
DRAM	H	H	H	H	H	H	No
MRAM	H	L	L	HH	L	H	Yes
PRAM	L	LL	H	HH	L	HH	Yes

H denotes high, *L* denotes low, *blue* color denotes advantage, *red* color denotes disadvantage

2.4.2.2 Memory Subsystem

The leakage energy of the memory is of more importance in the submicron era, as it surpasses the dynamic power of the memory [113]. Therefore, new memory technologies are evolving that address the issues of density and leakage power. A synopsis of some of the memory technologies and their hierarchy is given in Fig. 2.15. Next-generation NVMs like MRAM [117–119] and Phase-change RAMs (PRAM) [120, 121] have shown promising results towards leakage power reduction compared to SRAM or DRAM. Application designers are considering to exploit these memories by analyzing their advantages and disadvantages. Table 2.2 summarizes the main differences between the NVM and VM technologies. NVMs provide high capacity and low leakage power but their write latency and energy is substantially larger compared to that of SRAMs. However, the nonvolatility of the NVMs can be sacrificed, and NVMs like STT-RAM can be used as VMs.

In [122] and [123], a hybrid memory architecture comprising of PRAM and DRAM is proposed. A hybrid of scratchpad and NVMs for on-chip memories is

given in [124]. A technique which utilizes hybrid memory for video decoding is given in [125], where frame-level decisions for storing H.264 frames on hybrid-memories are used. However, H.264/AVC encoder presents a harder challenge as it is ~10× to 20× more complex than the decoder [2]. Reference [9] targets the HEVC application and limits the external memory access by storing the video samples (which are expected to be used later) in a hybrid combination of SRAM and STT-RAM. Other approaches utilizing MRAMs as replacements and augmentation of the traditional fast SRAMs are reported in [126, 127].

Most state-of-the-art memory subsystem designs do not jointly reduce the power consumption of the system in conjunction with meeting the throughput demands. Additionally, overall system characteristics (e.g., data transfer from external to on-chip memory) are not considered.

2.4.2.3 Accelerator Allocation/Scheduling

In order to combine the advantages of both programmable and application-specific custom architectures (e.g., ASICs), accelerator-based many-core systems are becoming increasingly popular in the industry [128, 129]. Accelerators are implemented as custom hardware for high-complexity parts of programs (called subtasks), and a programmable core can offload its assigned tasks to these accelerators. For examples of accelerators, refer to Sect. 2.4.2.1. Accelerators naturally lend themselves to occupy the underutilized chip's area, i.e., occupy dim/gray silicon. In addition to increasing the bright silicon, accelerators are designed to quickly process the assigned tasks. Therefore, accelerators are fundamental to high-complexity, deadline-conscious applications. Examples include video encoding and decoding [130] (also see Intel's Quick Sync technology), software-defined radios [131], etc.

For ease of discussion, we broadly classify accelerators into three categories, based upon their flexibility and access mechanisms. These categories are also shown in Fig. 2.16:

- First are the in-core accelerators, which are embedded as a part of the programmable core's computation pipeline (e.g., Nios II custom instructions [132, 133], vector instructions [134], application-specific instruction-set processors (ASIPs) [135, 136]). However, note that the corresponding core can only access these accelerators, and they are a part of the execution stage in the computational pipeline. Therefore, these accelerators exhibit the least flexibility, as their cores can only access these accelerators.
- The second category is clustered accelerators, where an accelerator can be accessed by only a specific set of cores and they reside in vicinity of these cores (e.g., within a computation tile). Such accelerators are also called tightly coupled accelerators [137, 138]. Techniques like [139, 140] allocate the accelerator to the corresponding cores by offloading the soft-core subtasks, using past-predicts-future paradigms and dynamic programming. However, these

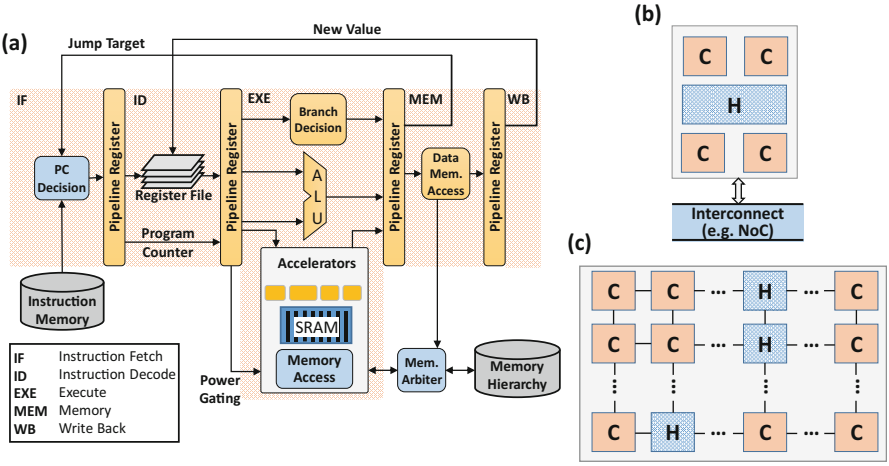

Fig. 2.16 Accelerator locality and access-based classification, (**a**) In-core, (**b**) tightly coupled, (**c**) loosely or decoupled accelerators

techniques do not consider the complete power consumption of the system, and neither do they account for the deadlines of the running applications.

- The third and the most flexible category of accelerators can be accessed by all the cores (e.g., via a network-on-chip (NoC) and PCIe) and are called decoupled accelerators or loosely coupled accelerators. It is obvious that the clustered and decoupled accelerators are the most versatile and offer maximum advantages. However, state-of-the-art accelerator allocation techniques presented in the literature [141–145] for decoupled accelerators usually try to reduce the resources used, maximize the processing speed, or reuse the accelerators' memory as cache or reconfigurable logic. No reference to the power consumption, frequency tuning of the cores, and deadlines of the applications is made.

Since the shared accelerator can only be allotted to a single compute resource at a given time, therefore some of the applications running on these cores might miss their deadlines, or these applications might change their workload at runtime. Further, it is likely that the accelerator is not continuously utilized, which defeats their purpose of providing power and complexity efficiency. In addition, it is also possible that to meet the deadlines, higher than required power is pumped to the cores. This will increase the power consumption of the system and, therefore, elevate the temperature of the chip.

2.4.2.4 SRAM Aging Rate Reduction Methods

In general, state-of-the-art techniques for aging mitigation mainly target aging optimization (i.e., aging rate reduction) for SRAM-based register files. However,

these techniques do not target large memories which have distinct access behavior and require different architectural support. The first category of techniques is based on the principle of bit rotations (i.e., moving LSB by one position with every write to the memory location) to improve duty cycle of registers [146, 147]. These techniques perform ineffectively for registers with successive zeros and are beneficial only when the bits inside registers are frequently modified, which is typically not the case for large-sized memories. Moreover, applying bit rotations requires barrel shifters at the read and write ports of the memory. We know that the total number of multiplexers required to implement an n-bit barrel shifters is $n\log_2 n$, and this is in addition to the control logic which is used to configure the barrel shifters. Therefore, the area overhead of such techniques might be high.

In [35], a redundancy-based SRAM mircoarchitecture extends the life of an SRAM cell. Similar to [148–150], this also requires architectural modification of the 6T SRAM cell. The register value inversion techniques result in additional reads/writes and power. The recovery boosting technique [148] adds dedicated inverters in the SRAM cells to improve the recovery process. However, this incurs significant power overhead, which may be infeasible for large-sized video memories, for instance, targeting image buffers for ME at high-definition (HD, 1920×1280 bytes) and 4K UHD (3840×2160 bytes) resolutions. Additionally, it requires an alteration to the SRAM 6T cell circuitry. Another category of work is based on bit flipping each bit at every write to the memory [147, 151, 148]. In [152], an algorithm is introduced for balancing the duty cycle of SRAM data caches by exploiting cache characteristics (i.e., tag bits). A similar technique is presented in [153, 154]. These architectures and techniques depend upon the inherent properties of caches (like flushing, cache hits, etc.) and are not directly pertinent to general on-chip SRAM memories. Moreover, some of the mentioned balancing policies are designed for capturing and exploiting the occurrence of a certain bit pattern and thus perform inefficiently for other content properties and varying stress patterns. Further, many of the reported works for aging balancing require multiple read/write of the same data in the memory, rendering themselves to be power hungry and increasing the latency of the application by halting their access to the memory.

Summarizing, state-of-the-art aging balancing techniques incur significant power and area overhead by employing bit flipping or rotation at every bit level, and reading and writing to the memories multiple times. These techniques do not explore the tradeoff between power consumption and aging balancing. Moreover, most of these techniques only provide elementary circuitry without exploring benefits of different aging balancing architectures and lack full architectural solution with power-aware aging control and adaptations.

2.4.2.5 Encountering the Power Wall at Hardware Level

Brief details about handling the power wall or dark silicon at the hardware layer of the video system is given in this section. However, the abovementioned state-of-the-art techniques (for designing efficient hardware accelerators, scheduling the

shared accelerator, power-efficient memories, etc.) implicitly address these challenges at the hardware layer.

At the hardware layer, different control knobs (e.g., for DVFS and DPM) are provided to throttle the chip's temperature within safe limits. Other techniques employ architectural heterogeneity to trade off performance and power. Via heterogeneity, the system supports runtime management of tasks by providing several degrees of freedom to the system designer. One can classify the different forms of heterogeneity as [155]:

- *Functional heterogeneity*, where compute nodes exist with varying functional behaviors and architectural details. Examples are application-specific hardware accelerators, superscalar cores and RISC processors, GPUs in conjunction with CPUs, reconfigurable architectures, etc. Thus, using task migrations and using scheduling, design challenges for modern fabrication technologies can be encountered.
- *Accelerator heterogeneity*, same as discussed in Sect. 2.4.2.3, i.e., in-core, clustered, and decoupled accelerators, providing different levels of performance and flexibility of usage. Further, approximate accelerators [156, 157] with controllable amount of approximations can also be used to increase the throughput-per-watt metric. This will not only increase the amount of bright silicon but also enable higher performance.
- *Microarchitectural heterogeneity*, whereby different cores on the same die have varying power and performance properties, but employ the same instruction set architecture (ISA). An example is ARM big.LITTLE architecture [60]. For example, [158] presents a methodology to design multi-core systems while considering the dark silicon paradigm. The purpose of their technique is to maximize the utilization of the silicon. In [50], depending upon the characteristics of parallel running applications, dark silicon-aware multiprocessors are synthesized using a library of available core types. In [159], special-purpose conservation cores (c-cores) are discussed, the goal of which is to reduce the energy consumption of the system rather than boasting performance. Device-level heterogeneous multi-cores and resource management are exploited in [160] to speed up the performance, as well as save energy.
- *On-chip interconnect heterogeneity*, whereby the network routers that connect the multiple cores of the chip are designed with heterogeneous architectures [161]. This provides diverse power and performance design points, available for the system designer.
- *Process heterogeneity*, where the nonideal fabrication process results in core-to-core and chip-to-chip variations in the maximum achievable frequency and leakage power. This variation can be exploited to adaptively grow the speedup of applications [162, 163] while meeting the TDP budgets of the chip.

2.5 Summary of Related Work

A plethora of techniques to tackle challenges imposed by video system software, hardware, and new fabrication technologies are presented in the state-of-the-art. Summarizing, the state-of-the-art does not exploit the complete design space concerning both hardware-software co-design and co-optimization. This is specifically important for multimedia systems, under dark silicon and reliability threats. For best throughput-per-watt ratio, the designer needs to consider the full-system stack, which involves the design of software layer, to the knowledge and exploitation of the hardware layer. This knowledge can be used to fully exploit complexity, power, and resource savings and reliability improvement potential for long-term system deployment.

Usually, the dark silicon mitigation techniques proposed in the state of the art do not consider the throughput constraints, and they do not exploit application-specific properties. As discussed, multimedia systems have deadline constraints, which require intelligent power budget distribution (i.e., frequency allocation) among the resources. Mostly, the state of the art does not consider the impact of deadlines and resource and power budgeting for shared accelerator-based systems. This results in suboptimal performance of the system and reduction in power efficiency.

Similarly, for SRAM aging-rate reduction, state-of-the-art techniques employ fixed aging balancing algorithms and architectures, with significant energy overhead. Therefore, these techniques are unable to explore the tradeoff between aging balancing and energy consumption. Moreover, due to the added power consumption, the state-of-the-art techniques might result in a higher temperature, which will increase the aging rate in a positive feedback cycle. Further, exploring the application-specific properties might result in high power efficiency and high reliability, which is mostly ignored by the state of the art.

References

1. Bjontegaard, G. (2001). Calculation of average PSNR differences between RD-curves. VCEG Contribution VCEG-M33.
2. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., & Wedi, T. (2004). Video coding with H.264/AVC: Tools, performance, and complexity. *IEEE Circuits and Systems Magazine, 4*(1), 7–28.
3. Sullivan, G. J., Ohm, J., Han, W., & Wiegand, T. (2012). Overview of high efficiency video coding. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1649–1668.
4. Wiegand, T., Sullivan, G., Bjontegaard, G., & Luthra, A. (2003). Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology, 13*(7), 560–576.
5. Chen, C., Huang, C., Chen, Y., & Chen, L. (2006). Level C+ data reuse scheme for motion estimation with corresponding coding orders. *IEEE Transactions on Circuits and Systems for Video Technology, 16*(4), 553–558.

6. Zhu, S., & Ma, K.-K. (2000). A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing, 9*(2), 287–290.
7. Bossen, F., Bross, B., Suhring, K., & Flynn, D. (2012). HEVC complexity and implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1685–1696.
8. Sze, V., Finchelstein, D. F., Sinangil, M. E., & Chandraksan, A. P. (2009). A 0.7-V 1.8-mW H.264/AVC 720p video decoder. *IEEE Journal of Solid-Sate Circuits, 44*(11), 2943–2956.
9. Sampaio, F., Shafique, M., Zatt, B., Bampi, S., & Henkel, J. (2014). Energy-efficient architecture for advanced video memory. In *International Conference on Computer-Aided Design*.
10. Purnachand, N., Alves, L. N., & Navarro, A. (2012). Improvements to TZ search motion estimation algorithm for multiview video cod-ing. In *IEEE International Confernce on Systems, Signals and Image Processing (IWSSIP)*, pp. 388–391.
11. Shafique, M., Bauer, L., & Henkel, J. (2010). enBudget: A run-time adaptive predictive energy-budgeting scheme for energy-aware motion estimation in H.264/MPEG-4 AVC video encoder. In *Design, Automation and Test in Europe*.
12. Gurhanli, A., Chen, C.-P., & Hung, S.-H. (2010). GOP-level parallelization of the H.264 decoder without a start-code scanner. In *International Conference on Signal Processing Systems (ICSPS)*.
13. VideoLAN - x264. [Online]. Available: http://www.videolan.org/developers/x264.html. Accessed 5 Oct 2015.
14. Zhao, L., Xu, J., Zhou, Y., & Ai, M. (2012). A dynamic slice control scheme for slice-parallel video encoding. In *International Conference on Image Processing*.
15. Ba, K., Jin, X., & Goto, S. (2010). A dynamic slice-resize algorithm for fast H.264/AVC parallel encoder. In *International Symposium on Intelligent Signal Processing and Communication Systems*.
16. Khan, M. U. K., Shafique, M., & Henkel, J. (2014). Software architecture of high efficiency video coding for many-core systems with power-efficient workload balancing. In *Design, Automation and Test in Europe*.
17. Ahmad, I., & Ghafoor, A. (1991). Semi-distributed load balancing for massively parallel multicomputer systems. *IEEE Transactions on Software Engineering, 17*(10), 987–1004.
18. Williams, R. (1991). Performance of dynamic load balancing algorithms for unstructured mesh calculations. *Concurrency: Practice and experience, 3*(5), 457–481.
19. Juice Encoder– 4 in 1 MPEG-4 AVC/H.264 HD encoder. Antik Technology, [Online]. Available: http://www.antiktech.com/iptv-products/juice-encoder-EN-5004-5008/
20. Marvell 88DE3100 High-Definition Secure Media Processor System-on-Chip (SoC). [Online]. Available: http://www.marvell.com/digital-entertainment/armada-1500/assets/Marvell-ARMADA-1500-Product-Brief.pdf/
21. Distributed Coding for Video Services (DISCOVER). Application scenarios and functionalities for DVC.
22. Wyner, A., & Ziv, J. (1976). The rate-distortion function for source coding with side information at the decoder. *IEEE Transaction on Information Theory, 22*, 1–10.
23. Girod, B., Aaron, A. M., Rane, S., & Rebollo-Monedero, D. (2005). Distributed Video Coding. *Proceedings of the IEEE, 93*(1), 71–83.
24. Puri, R., & Ramchandran, K. (2003). PRISM: An uplink-friendly multimedia coding paradigm. In *International Conference on Acoustics, Speech, and Signal Processing*.
25. Chen, J., Khisti, A., Malioutov, D., & Yedidia, J. (2004). Distributed source coding using serially-concatenated-accumulate codes. In *Information Theory Workshop*.
26. Tseng, H.-Y., Shen, Y.-C., & Wu, J.-L. (2011). Distributed video coding with compressive measurements. In *International conference on Multimedia*.
27. Sejdinovic, D., Piechocki, R. J., & Doufexi, A. (2009). Rateless distributed source code design. In *Mobile Multimedia Communica-tions Conference*.

28. Chien, S.-Y., Cheng, T.-Y., Chiu, C.-C., Tsung, P.-K., Lee, C.-H., Somayazulu, V., & Chen, Y.-K. (2012). Power optimization of wireless video sensor nodes in M2M networks. In *Asia and South Pacific Design Automation Conference*.
29. Huang, Y.-W., Chen, T.-C., Tsai, C.-H., Chen, C.-Y., Chen, T.-W., Chen, C.-S., Shen, C.-F., Ma, S.-Y., Wang, T.-C., Hsieh, B.-Y., Fang, H.-C., & Chen, L.-G. (2005). A 1.3TOPS H.264/AVC single-chip en-coder for HDTV applications. In *International Solid-State Circuits Conference*.
30. Chiu, C.-C., Chien, S.-Y., Lee, C.-H., Somayazulu, V., & Chen, Y.-K.. (2011). Distributed video coding: A promising solution for distributed wireless video sensors or not?. In *Visual Communications and Image Processing*.
31. Shafique, M., Khan, M. U. K., & Henkel, J. (2013). Content-driven adaptive computation offloading for energy-aware hybrid distributed video coding. In *International Symposium on Low Power Electronics and Design (ISLPED)*.
32. Kumar, K., Liu, J., Lu, Y.-H., & Bhargava, B. (2012). A survey of computation offloading for mobile systems. *Mobile Networks and Applications, 18*(1), 129–140.
33. Colin, A., Kandhalu, A., & Rajkumar, R. (2015). Energy-efficient allocation of real-time applications onto single-ISA heterogeneous multi-core processors. *Journal of Signal Processing Systems,* pp. 1–20.
34. Schroder, D. K., & Babcock, J. A. (2003). Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing. *Journal of Applied Physics, 94*(1), 1–18.
35. Shin, J., Zyuban, V., Bose, P., & Pinkston, T. (2008). A proactive wearout recovery approach for exploiting microarchi-tectural redundancy to extend cache SRAM lifetime. In *International Symposium on Computer Architecture (ISCA)*.
36. Tiwari, A., & Torrellas, J. (2008). Facelift: Hiding and slowing down aging in multicores. In *International Symposium on Microarchitecture (MICRO)*.
37. Vattikonda, R., Wang, W., & Cao, Y. (2006). Modeling and minimization of PMOS NBTI effect for robust nanometer design. In *Design Automation Conference (DAC)*.
38. Velamala, J. B., Sutaria, K., Sato, T., & Cao, Y. (2012). Physics matters: Statistical aging prediction under trapping/detrapping. In *Design Automation Conference (DAC)*.
39. Singh, A., Shafique, M., Kumar, A., & Henkel, J. (2013). Mapping on multi/many-core systems: Survey of current and emerging trends. In *Design Automation Conference (DAC)*.
40. Chi, C. C., Alvarez-Mesa, M., Juurlink, B., Clare, G., Henry, F., Pateux, S., & Schierl, T. (2012). Parallel scalability and efficiency of HEVC parallelization approaches. *IEEE Transactions on Circuits and Systems on Video Technology, 22*(12), 1827–1838.
41. Alvanos, M., Tzenakis, G., Nikolopoulos, D. S., & Bilas, A. (2011). Task-based parallel H. 264 video encoding for explicit communication architectures. In *International Conference on Embedded Computer Systems*.
42. Brun, O., Teuliere, V., & Garcia, J. M. (2002). Parallel particle filtering. *Journal of Parallel and Distributed Computing, 62*(7), 1186–1202.
43. Rujirakul, K., So-In, C., & Arnonkijpanich, B. (2014). PEM-PCA: A parallel expectation-maximization PCA face recognition architecture. *The Scientific World Journal*.
44. Jing, X.-Y., Li, S., Zhang, D., Yang, J., & Yang, J.-Y. (2012). Supervised and unsupervised parallel subspace learning for large-scale image recognition. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(10), 1497–1511.
45. Dong, C., Zhao, H., & Wang, W. (2010). Parallel nonnegative matrix factorization algorithm on the distributed memory platform. *International Journal of Parallel Programming, 38*(2), 117–137.
46. Shah, S., & Kothari, R. (2013). Convergence of the dynamic load balancing problem to Nash equilibrium using distributed local interactions. *Information Sciences, 221*, 297–305.
47. Drougas, Y., Repantis, T., & Kalogeraki, V. (2006). Load balancing techniques for distributed stream processing applications in overlay environments. In *IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*.

48. Robertazzi, T. G. (2003). Ten reasons to use divisible load theory. *Computer, 36*(5), 63–68.
49. Wang, K., Zhou, X., Li, T., Zhao, D., Lang, M., & Raicu, I. (2014). Optimizing load balancing and data-locality with data-aware scheduling. In *International Conference on Big Data*.
50. Turakhia, Y., Raghunathan, B., Garg, S., & Marculescu, D. (2013). HaDeS: Architectural synthesis for heterogeneous dark silicon chip multi-processors. In *Design Automation Conference (DAC)*.
51. Buss, M., Givargis, T., & Dutt, N. (2003). Exploring efficient operating points for voltage scaled embedded processor cores. In *Real-Time Systems Symposium (RTSS)*.
52. Rosas, C. Morajko, A. Jorba, J., & Cesar, E. (2011). Workload balancing methodology for data-intensive applications with divisible load. In *Symposium on Computer Architecture and High Performance Computing*.
53. Matthur, A., & Mundur, P. (2003). Dynamic load balancing across mirrored multimedia servers. In *International Conference on Multimedia and Expo*.
54. Bowman, K., Duvall, S., & Meindl, J. (2002). Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid-State Circuits, 37*(2), 183–190.
55. Kim, J., Yoo, S., & Kyung, C.-M. (2011). Program phase-aware dynamic voltage scaling under variable computational workload and memory stall environment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 30*(1), 110–123.
56. Devadas, V., & Aydin, H. (2010). DFR-EDF: A unified energy management framework for real-time systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*.
57. Ma, K., Li, X., Chen, M., & Wang, X. (2011). Scalable power control for many-core architectures running multi-threaded applications. In *Internation Symposium on Computer Architecture*.
58. Dehyadegari, M., Marongiu, A., Kakoee, M., Mohammadi, S., Yazdani, N., & Benini, L. (2015). Architecture support for tightly-coupled multi-core clusters with shared-memory HW accelerators. *IEEE Transactions on Computer, 64*(8), 2132–2144.
59. Sarma, S., Muck, T., Bathen, L., Dutt, N., & Nicolau, A. (2015). SmartBalance: A sensing-driven linux load balancer for energy efficiency of heterogeneous MPSoCs. In *Design Automation Conference (DAC)*.
60. ARM big.LITTLE Architecture. ARM, [Online]. Available: http://www.arm.com/products/processors/technologies/biglittleprocessing.php. Accessed 07 Aug 2015.
61. Momcilovic, S., Ilic, A., Roma, N., & Sousa, L. (2014). Dynamic load balancing for real-time video encoding on heterogeneous CPU+GPU systems. *IEEE Transactions on Multimedia, 16*(1), 108–121.
62. Xiao, W., Li, B., Xu, J., Shi, G., & Wu, F. (2015). HEVC encoding optimization using multi-core CPUs and GPUs. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 9*(99), 1–14.
63. Momcilovic, S., Roma, N., & Sousa, L. (2013). Exploiting task and data parallelism for advanced video coding on hybrid CPU + GPU platforms. *Journal of Real-Time Image Processing,* pp. 1–17.
64. Jian, C., & John, L. (2009). Efficient program scheduling for heterogeneous multi-core processors. In *Design Automation Conference (DAC)*.
65. Mühlbauer, T., Rödiger, W., Seilbeck, R., Kemper, A., & Neumann, T. (2014). Heterogeneity-conscious parallel query execution: Getting a better mileage while driving faster!. In *International Workshop on Data Management on New Hardware*.
66. Shafique, M., Molkenthin, B., & Henkel, J. (2010). An HVS-based adaptive computational complexity reduction scheme for H.264/AVC video encoder using prognostic early mode exclusion. In *Design, Automation and Test in Europe Conference (DATE)*.
67. Bariani, M., Lambruschini, P., & Raggio, M. (2012). An efficient multi-core SIMD implementation for H.264/AVC encoder. In *VLSI Design*.

68. Rodríguez, A., González, A., & Malumbres, M. P. (2006). Hierarchical parallelization of an H.264/AVC video encoder. In *International Symposium on Parallel Computing in Electrical Engineering*.
69. Gong, P., Basciftci, Y., & Ozguner, F. (2012). A parallel resampling algorithm for particle filtering on shared-memory architectures. In *Parallel and Distributed Processing Symposium Workshops*.
70. Henkel, J., & Yanbing, L. (1998). Energy-conscious HW/SW-partitioning of embedded systems: a case study on an MPEG-2 encoder. In *International Workshop on Hardware/ Software Codesign*.
71. Cuomo, S., Michele, P. D., & Piccialli, F. (2014). 3D data denoising via nonlocal means filter by using parallel GPU strategies. In *Computational and Mathematical Methods in Medicine*.
72. Moustafa, M., Ebied, H. M., Helmy, A., Nazamy, T. M., & Tolba, M. F. (2014). Satellite super resolution image reconstruction based on parallel support vector regression. In *Advanced Machine Learning Technologies and Applications*, Springer, pp. 223–235.
73. Garcia Freitas, P., Farias, M. and De Araujo, A. (2014). A parallel framework for video super-resolution. In *Graphics, Patterns and Images (SIBGRAPI)*.
74. Jung, B., & Jeon, B. (2008). Adaptive slice-level parallelism for H.264/AVC encoding using pre macroblock mode selection. *Journal of Visual Communication Image Representation, 19* (8), 558–572.
75. Jiang, W., Mal, H., & Chen, Y. (2012). Gradient based fast mode decision algorithm for intra prediction in HEVC. In *International Conference on Consumer Electronics, Communications and Networks*.
76. Cassa, M. B., Naccari, M., & Pereira, F. (2012). Fast rate distortion optimization for the emerging HEVC standard. In *Picture Coding Symposium.*
77. Zhang, H., & Ma, Z. (2012). Fast intra prediction for high efficiency video coding. In *Advances in Multimedia Information Processing*.
78. Sun, H., Zhou, D., & Goto, S. (2012). A low-complexity HEVC Intra prediction algorithm based on level and mode filtering,. In *International Conference on Multimedia and Expo (ICME)*.
79. Pan, F., Lin, X., Rahardja, S., Lim, K. P., Li, Z. G., Wu, D., & Wu, S. (2005). Fast mode decision algorithm for intraprediction in H.264/AVC video coding. *IEEE Transactions on Circuits and Systems for Video Technology, 15*(7), 813–822.
80. Tsai, A. C., Paul, A., Wang, J. C., & Wang, J. F. (2008). Intensity gradient technique for efficient intra-prediction in H.264/AVC. *IEEE Transactions on Circuits and Systems on Video Technology, 18*(5), 694–698.
81. Fonseca, T. A., Liu, Y., & Queiroz, R. L. D. (2007). Open-loop prediction in H.264 / AVC for high definition sequences. In *SBrT*.
82. Tian, G., & Goto, S. (2012). Content adaptive prediction unit size decision algorithm for HEVC intra coding. In *Picture Coding Symposium*.
83. Zhao, L., Zhang, L., Ma, S., & Zhao, D. (2011). Fast mode decision algorithm for Intra prediction in HEVC. In *Visual Communications and Image Processing (VCIP)*.
84. Silva, T. D., Agostini, L. V., & Cruz, L. A. D. S. C. (2012). Fast HEVC intra prediction mode decision based on EDGE direction information. In *European Signal Processing Conference (Eusipco)*.
85. Haan, G. D., & Biezen, P. (1998). An efficient true-motion estimator using candidate vectors from a parametric motion model. *IEEE Transactions on Circuits and Systems for Video Technology, 8*(9), 86–91.
86. Shim, H., & Kyung, C.-M. (2009). Selective search area reuse algorithm for low external memory access motion estimation. *IEEE Transactions on Circuits and Systems for Video Technology, 19*(7), 1044–1050.
87. Kun, Z., Chun, Y., Qiang, L., & Yuzhuo, Z. (2007). A fast block type decision method for H.264/AVC intra prediction. In *International Conference on Advanced Communication Technology*.

88. Lin, Y.-K., & Chang, T. (2005). Fast block type decision algorithm for intra prediction in H.264/FRex. In *Internatianal conference on Image Processing (ICIP)*.
89. Kivanc Mihcak, M., Kozintsev, I., Ramchandran, K., & Moulin, P. (1999). Low-complexity image denoising based on statistical modeling of wavelet coefficients. *IEEE Signal Processing Letters, 6*(12), 300–303.
90. Khan, M. U. K., Bais, A., Khawaja, M., Hassan, G. M., & Arshad, R. (2009). A swift and memory efficient hough transform for systems with limited fast memory. In *International Conference on Image Analysis and Recognition (ICIAR)*.
91. OpenCV. [Online]. Available: http://opencv.org/. Accessed 08 Aug 2015.
92. OpenVX. [Online]. Available: https://www.khronos.org/openvx/. Accessed 08 Aug 2015.
93. Muthukaruppan, T. S., Pricopi, M., Venkataramani, V., Mitra, T., & Vishin, S. (2013). Hierarchical power management for asymmetric multi-core in dark silicon era. In *Design Automation Conference (DAC)*.
94. Khdr, H., Ebi, T., Shafique, M., Amrouch, H., & Henkel, J. (2014). mDTM: Multi-objective dynamic thermal management for on-chip systems. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*.
95. Devadas, V., & Aydin, H. (2012). On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications. *IEEE Transactions on Computers, 61*(1), 31–44.
96. Isci, C., Buyuktosunoglu, A., Cher, C., Bose, P., & Martonosi, M. (2006). An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In *Microarchitecture*.
97. Pagani, S., Khdr, H., Munawar, W., Chen, J.-J., Shafique, M., Li, M., & Henkel, J. (2014). TSP: Thermal safe power: Efficient power budgeting for many-core systems in dark silicon. In *International Conference on Hardware/Software Codesign and System Synthesis*.
98. Mishra, A., Srikantaiah, S., Kandemir, M., & Das, C. R. (2010). CPM in CMPs: Coordinated power management in chip-multiprocessors. In *International Conference on High Performance Computing, Networking, Storage and Analysis*.
99. Winter, J. A., Albonesi, D. H., & Shoemaker, C. A. (2010). Scalable thread scheduling and global power management for heterogeneous many-core architectures. In *Parallel Architectures and Compilation*.
100. Sharifi, A., Mishra, A., Srikantaiah, S., Kandemir, M., & Das, C. R. (2012). PEPON: Performance-aware hierarchical power budgeting for NoC based multicores. In *Parallel Architectures and Compilation Techniques*.
101. Shafique, M., Garg, S., Henkel, J., & Marculescu, D. (2014). The EDA challenges in the dark silicon era. In *Design Automation Conference*.
102. Huang, Y.-W., Hsieh, B.-Y., Chen, T.-C., & Chen, L.-G. (2005). Analysis, fast algorithm, and VLSI architec-ture design for H.264/AVC intra frame coder. *IEEE Transactions on Circuits and Systems for Video Technology, 15*(3), 378–401.
103. Wang, J.-C., Wang, J.-F., Yang, J.-F., & Chen, J.-T. (2007). A fast mode decision algorithm and its VLSI design for H.264/AVC intra-prediction. *IEEE Transactions on Circuits and Systems for Video Technology, 17*(10), 1414–1422.
104. Roszkowski, M., & Pastuszak, G. (2010). Intra prediction hardware module for high-profile H.264/AVC encoder. In *Signal Processing Algorithms, Architectures, Arrangements and Applications Conference*.
105. He, G., Zhou, D., Zhou, J., & Goto, S. (2010). Intra prediction architecture for H.264/AVC QFHD encoder. In *Picture Coding Symposium*.
106. Diniz, C., Zatt, B., Thiele, C., Susin, A., Bampi, S., Sampaio, F., Palomino, D., & Agostini, L. (2011). A high throughput H.264/AVC intra-frame encod-ing loop architecture for HD1080p. In *International Symposium on Circuits and Systems*.
107. Li, F., Shi, G., & Wu, F. (2011). An efficient VLSI architecture for 4×4 intra prediction in the High Efficiency Video Coding (HEVC) standard. In *International Conference on Image Processing*.

108. Cervero, T., Otero, A., López, S., de la Torre, E., Callicó, G., Riesgo, T., & Sarmiento, R. (2013). A scalable H.264/AVC deblocking filter architecture. *Journal of Real-Time Image Processing,* pp. 1–25.
109. Mangard, S., Aigner, M., & Dominikus, S. (2003). A highly regular and scalable AES hardware architecture. *IEEE Transactions on Computers, 52*(4), 483–491.
110. Varatkar, G. V., & Shanbhag, N. R. (2008). Error-resilient motion estimation architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 16*(10), 1399–1412.
111. Saponara, S., & Fanucci, L. (2004). Data-adaptive motion estimation algorithm and VLSI architecture design for low-power video systems. *IEE Proceedings-Computers and Digital Techniques, 151*(1), 51–59.
112. Tsai, C.-Y., Chung, C., Chen, Y.-H., Chen, T.-C., & Chen, L.-G. (2007). Low power cache algorithm and architecture design for fast motion estimation in H. 264/AVC encoder system. In *In-ternational Conference on Acoustics, Speech and Signal Processing*.
113. Kim, N. S., Austin, T., Blaauw, D., Mudge, T., Flautner, K., Hu, J. S., Irwin, M. J., Kandemir, M., & Narayanan, V. (2003). Leakage current: Moore's law meets static pow-e. *Computers, 36*(12), 68–75.
114. Ma, Z., & Segall, A. (2011). Frame buffer compression for low-power video coding. In *International Conference on Image Processing*.
115. Hsu, M.-Y. (2000). Scalable module-based architecture for MPEG-4 BMA motion estimation.
116. Tuan, J.-C., Chang, T.-S., & Jen, C.-W. (2002). On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture. *IEEE Transactions on Circuits and Systems for Video Technology, 12*(1), 61–72.
117. Wu, X., Li, J., Zhang, L., Speight, E., Rajamony, R., & Xie, Y. (2009). Hybrid cache architecture with disparate memory technologies. In *International Symposium on Computer Architecture (ISCA)*.
118. Diao, Z., Li, Z., Wang, S., Ding, Y., Panchula, A., Chen, E., Wang, L.-C., & Huai, Y. (2007). Spin-transfer torque switching in magnetic tunnel junctions and spin-transfer torque random access memory. *Journal of Physics: Condensed Matter, 19*(16), 1–13.
119. Dong, X., Wu, X., Sun, G., Xie, Y., Li, H., & Chen, Y. (2008). Circuit and microarchitecture evaluation of 3D stacking magnetic RAM (MRAM) as a universal memory replacement. In *Design Automation Conference (DAC)*.
120. Qureshi, M. K., Srinivasan, V., & Rivers, J. A. (2009). Scalable high perfor-mance main memory system using phase-change memory technology. In *International Symposium on Computer Architec-ture (ISCA)*.
121. Hanzawa, S., Kitai, N., Osada, K., Kotabe, A., Matsui, Y., Matsuzaki, N., Takaura, N., Moniwa, M., & Kawahara, T. (2007). A 512KB Embed-ded phase change memory with 416kB/s write throughput at 100uA cell write current. In *International Solid-State Circuits Conference (ISSCC)*.
122. Yang, S., & Ryu, Y. (2012). A memory management scheme for hybrid memory architecture in mission critical computers. In *International Conference on Software Technology*.
123. Dhiman, G., Ayoub, R., & Rosing, T. (2009). PDRAM: A hybrid PRAM and DRAM main memory system. In *Design Automation Conference*.
124. Bathen, L., & Dutt, N. (2012). HaVOC: A hybrid memory-aware virtualization layer for on-chip distributed scratchpad and non-volatile memories. In *Design Automation Conference*.
125. Stancu, L. C., Bathen, L. A. D., Dutt, N., & Nicolau, A. (2012). AVid : Annotation driven video decoding for hybrid memories. In *Embedded Systems for Real-Time Multimedia*.
126. Desikan, R., Lefurgy, C., Keckler, S., & Burger, D. (2002). *On-chip MRAM as a high-bandwidth, low-latency replacement for DRAM physical memories*. University of Texas at Austin.

127. Nomura, K., Abe, K., Yoda, H., & Fujita, S. (2012). Ultra low power processor using perpendicular-STT-MRAM/SRAM based hy-brid cache toward next generation normally-off computers. *Journal of Applied Physics, 111*(7).
128. Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., & Hanrahan, P. (2008). Larrabee: A Many-Core x86 Architecture for Visual Computing. *ACM Transactions on Graphics, 27*(3).
129. Mattina, M. (2014). *Architecture and Performance of the Tile-GX Processor Family,* White Paper.
130. Shafique, M., Bauer, L., & Henkel, J. (2010). Optimizing the H.264/AVC video encoder application structure for reconfigurable and application-specific platforms. *Journal of Signal Processing Systems (JSPS), 60*(2), 183–210.
131. Liu, C., Granados, O., Duarte, R., & Andrian, J. (2012). Energy efficient architecture using hardware acceleration for software defined radio components. *Journal of Information Processing Systems, 8*(1), 133–144.
132. Nios II Custom Instruction User Guide. Altera, (2011).
133. Khan, M. U. K., Shafique, M., & Henkel, J. (2013). Hardware-software collaborative complexity reduction scheme for the emerging HEVC intra encoder. In *Design, Automation and Test in Europe (DATE)*.
134. Shojania, H., & Baochun, L. (2007). Parallelized progressive network coding with hardware acceleration. In *International Workshop on Quality of Service*.
135. Doan, H. C., Javaid, H., & Parameswaran, S. (2014). Flexible and scalable implementation of H.264/AVC encoder for multiple resolutions using ASIPs. In *Design, Automation and Test in Europe Conference and Exhibition (DATE)*.
136. Kim, S. D., Lee, J. H., Hyun, C. J., & Sunwoo, M. H. (2006). ASIP approach for implementation of H.264/AVC. In *Asia and South Pacific Conference on Design Automation (ASP-DAC)*.
137. Swanson, S., & Taylor, M. B. (2011). GreenDroid: Exploring the next evolution in smartphone application processors. *IEEE Communications Magazine, 49*(*4*), 112–119.
138. Teich, J., Henkel, J., Herkersdorf, A., Schmitt-Landsiedel, D., Schröder-Preikschat, W., & Snelting, G. (2010). Invasive computing: An overview. In *Multiprocessor System-on-Chip*, Springer, pp. 241–268.
139. Sheldon, D., & Forin, A. (2010). An online scheduler for hardware accelerators on general purpose operating systems. Microsoft Research.
140. Huang, C., Sheldon, D., & Vahid, F. (2008). Dynamic tuning of configurable architectures: The AWW online algorithm. In *International Conference on Hardware/Software Codesign and System Synthesis*.
141. Majumder, T., Pande, P. P., & Kalyanaraman, A. (2013). High-throughput, energy-efficient network-on-chip-based hardware accelerators. *Journal of Sustainable Computing: Informatics and Systems, 3*(1), 36–46.
142. Cong, J., Ghodrat, M. A., Gill, M., Grigorian, B., & Reinman, G. (2012). Architecture support for accelerator-rich CMPs. In *Design Automation Conference*.
143. Cota, E., Mantovani, P., Petracca, M., Casu, M., & Carloni, L. (2012). Accelerator memory reuse in the dark silicon era. *Computer Architecture Letters,* pp. 1–4.
144. Clemente, J. A., Beretta, I. V., Rana, V., Atienza, D., & Sciuto, D. (2014). A mapping-scheduling algorithm for hardware acceleration on reconfigurable platform. *Transactions on Reconfigurable Technology and Systems, 7*(2).
145. Paul, S., Karam, R., Bhunia, S., & Puri, R. (2014). Energy-efficient hardware acceleration through computing in the memory. In *Design, Automation and Test in Europe (DATE)*.
146. Kothawade, S., Chakraborty, K., & Roy, S. (2011). Analysis and mitigation of NBTI aging in register file: An end-to-end approach. In *International Symposium on Quality Electronic Design (ISQED)*.

147. Amrouch, H., Ebi, T., & Henkel, J. (2013). Stress balancing to mitigate NBTI Effects in register files. In *Dependable Systems and Networks (DSN)*.
148. Siddiqua, T., & Gurumurthi, S. (2010). Recovery boosting: A technique to enhance NBTI recovery in SRAM arrays. In *Annual Symposium on VLSI*.
149. Sil, A., Ghosh, S., Gogineni, N., & Bayoumi, M. (2008). A novel high write speed, low power, read-SNM-Free 6T SRAM cell. In *Midwest Symposium on Circuits and Systems*.
150. Abella, J., Vera, X., Unsal, O., & Gonzalez, A. (2008). NBTI-resilient memory cells with NAND gates. US Patent US20080084732 A1.
151. Wang, S., Jin, T., Zheng, C., & Duan, G. (2012). Low power aging-aware register file design by duty cycle balancing. In *Design, Automation and Test in Europe (DATE)*.
152. Wang, S., Duan, G., Zheng, C., & Jin, T. (2013). Combating NBTI-induced aging in data caches. In *Great lakes symposium on VLSI*.
153. Gunadi, E., Sinkar, A. A., Kim, N. S., & Lipasti, M. H. (2010). Combating aging with the colt duty cycle equalizer. In *International Symposium on Microarchitecture*.
154. Calimera, A., Loghi, M., Macii, E., & Poncino, M. (2011). Partitioned cache architectures for reduced NBTI-induced aging. In *Design, Automation and Test in Europe (DATE)*.
155. Henkel, J., Bukhari, H., Garg, S., Khan, M. U. K., Khdr, H., Kriebel, F., Ogras, U., Parameswaran, S., & Shafique, M. (2015). Dark silicon – From computation to communication. In *International Symposium on Networks-on-Chip (NOCs)*.
156. Esmaeilzadeh, H., Sampson, A., Ceze, L., & Burger, D. (2012). Neural acceleration for general-purpose approximate programs. In *International Symposium on Microarchitecture*.
157. Mahajan, D., Yazdanbakhsh, A., Park, J., Thwaites, B., & Esmaeilzadeh, H. (2015). Prediction-based quality control for approximate accelerators. In *Workshop on Approximate Computing Across the System Stack*.
158. Allred, J., Roy, S., & Chakraborty, K. (2012). Designing for dark silicon: A methodological perspective on energy efficient systems. In *International Symposium on Low Power Electronics and Design (ISLPED)*.
159. Venkatesh, G., Sampson, J., Goulding, N., Garcia, S., Bryksin, V., Lugo-Martinez, J., Swanson, S., & Taylor, M. B. (2010). Conservation cores: Reducing the energy of mature computations. In *Architectural Support for Programming Languages and Operating Systems*.
160. Swaminathan, K., Kultursay, E., Saripalli, V., Narayanan, V., Kandemir, M., & Datta, S. (2013). Steep-slope devices: From dark to dim silicon. *IEEE Micro, 33*(5), 50–59.
161. Bokhari, H., Javaid, H., Shafique, M., Henkel, J., & Parameswaran, S. (2014). darkNoC: Designing energy-efficient network-on-chip with multi-Vt cells for dark silicon. In *Design Automation Conference (DAC)*.
162. Raghunathan, B., Turakhia, Y., Garg, S., & Marculescu, D. (2013). Cherry-picking: Exploiting process variations in dark-silicon homogeneous chip multi-processors. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*.
163. Shafique, M., Gnad, D., Garg, S., & Henkel, J. (2015). Variability-aware dark silicon management in on-chip many-core systems. In *Design, Automation and Test in Europe Conference and Exhibition*.
164. Huang, W., Rajamani, K., Stan, M., & Skadron, K. (2011). Scaling with design constraints: Predicting the future of big chips. *IEEE Micro, 31*(4), 16–29.

http://www.springer.com/978-3-319-61454-0

Energy Efficient Embedded Video Processing Systems
A Hardware-Software Collaborative Approach
Khan, M.U.K.; Shafique, M.; Henkel, J.
2018, VIII, 238 p. 107 illus., 105 illus. in color.,
Hardcover
ISBN: 978-3-319-61454-0