LOW-POWER DV ENCODER ARCHITECTURE FOR DIGITAL CMOS CAMCORDER

Jeff Y. F. Hsieh and Teresa H. Y. Meng

Electrical Engineering Department
Stanford University
Stanford, California 94305, USA

ABSTRACT

A low-power, large-scale parallel digital video (DV) [1] encoder architecture for a single-chip digital CMOS video camera is discussed in this paper. This architecture is based on the single chip CMOS camera MPEG-2 encoder architecture proposed in [2] with an emphasis on formatting and streaming of the compressed data. The architecture proposed here supports the 625/25 format of 720x576 pixels per frame. When clocked at 40 MHz, this architecture delivers a processing performance of 1.8 billion operations per second (BOPS) capable of supporting a frame rate of 25 fps as well as additional image enhancement processing. Low power consumption is achieved by the use of a parallel architecture and low-power circuit design techniques. When implemented in a 0.2 micron CMOS technology at a 1.5 V supply voltage, the parallel architecture consumes 45 mW providing a power efficiency of 40 billion operations per second per Watt.

1. INTRODUCTION

Traditional video compression systems approach performance enhancements by either reducing the cycle time or increasing processing parallelism. However, these enhancements are often made assuming video data is stored externally. This results in a large power overhead from data transfers alone. Furthermore, as process technology scales down, on-chip data transfer becomes more power efficient than off-chip transfer, because the power for I/O operation does not scale with process technology. Better process technology and the ability to embed DRAM with the processing circuitry circumvent redundant off-chip data transfer while providing high density buffering of video images. A large-scale parallel processing architecture is proposed. This architecture takes advantage of the high throughput parallel access to on-chip memory to achieve high computational throughput and low power consumption.

1.1 Architecture Overview

The proposed architecture is based on the architecture proposed for MPEG2 encoding in [2]. Figure 1 (sizes are not drawn to scale) shows how the processing elements and the photo sensors are integrated. The Frame Buffer (FB) consists of embedded DRAM to buffer images at high density. The FB is partitioned into 45 columns. This results in a column width of 16 pixels. Each column is separately addressed and accessed by an individual Processor Element (PE) in the processor array. An intrinsic advantage of using such processor configuration for image processing is that processing constraints of individual PEs are not affected by horizontal scaling of the image resolution. It is also possible to scale image resolution vertically by placing PEs on the other side of the photo sensors and partition the sensor vertically in two. This configuration will be explored in future works.

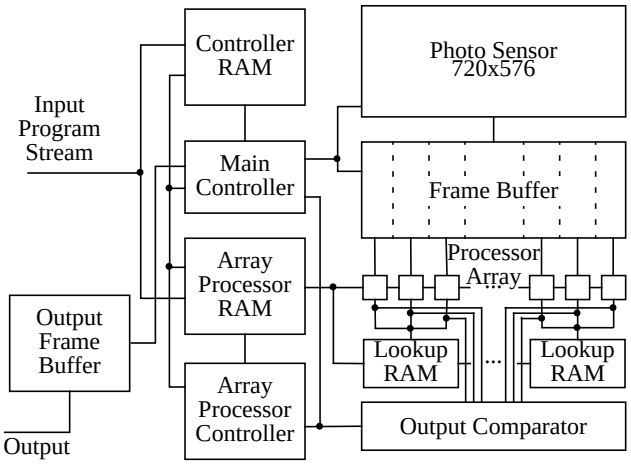

Figure 1: *Digital CMOS camera architecture*

To simplify instruction fetching, instructions for controlling the PEs are pipelined from the Array Processor RAM (AP-RAM) from the left-most processor towards the right. Pipelined PE instructions are stored in separate physical memories from the main controller (MC) instructions. This partitioning of codes reduces programming complexity by addressing different architectural issues at different levels. Data generated by the PE is summarized by the Output Comparator (OC) and fed back to the controlling units. This data can be further processed by either controllers (the MC or the Array Processor Controller, APC) or stored into the Output Frame Buffer (OFB) for streaming. Lookup coefficients are stored into the lookup RAM (L-RAM). An L-RAM is shared between a number of PEs to reduce area as well as power overheads.

Table 1: Image processing algorithms supported‡

• Color conversion	• White balancing
• Gamma correction	• DCT/IDCT
• 2-D FIR filtering	• Color space conversion
• Quantization	• Motion estimation†
• Sub-band coding	• Color subsampling
• Variable length coding	• Median filtering

† May require some modifications to the architecture

This architecture configuration introduces several advantages. First, data transfers are internalized and the exported data is compressed. This leads to a significantly lower power consumption than a comparable multi-chip solution where uncompressed data streams are transferred between chips. Second, by increasing the number of processing elements, the processing requirement per image column is reduced. This leads to a lower clock rate and supply voltage, resulting in an overall reduction of power con-

sumption. Third, this architecture can be programmed to process image algorithms other than MPEG2 and DV (Table 1). This provides the flexibility needed by camera product developers to determine optimal processing and image quality tradeoffs in software rather than in hardware. The cost of development therefore can be significantly reduced. Hardware reuse also implies that these single-chip designs can conceivably be used in different applications between different host modules. Programmability also implies a separation of performance optimization between hardware and software, which parallels the development of the microprocessor industry.

1.2 DV vs. MPEG2

A variety of video compression technologies exist today in which MPEG2 has been widely accepted by numerous applications. The DV standard is also emerging as a popular alternative in digital video compression. These two technologies have fundamental differences in their coding strategies that are beneficial to the specific application domains they serve. MPEG2 supports a compression ratio of greater than 100:1, which is suitable for video conferencing and video archiving where data bandwidth is limited. However, MPEG2 is not resilient to error propagation due to a strong dependence of past images. DV, on the other hand, is developed for acquisition of video where a high capacity video storage medium is assumed (i.e. mini tape cassettes). It has a compression ratio of approximately 3:1 to 5:1 and is suitable for applications such as digital camcorder, broadcasting, and video editing. These applications and the physical constraints of the tape medium requires that the DV standard be highly robust to bit errors and allow for quick access to stored data via trick plays (high speed bi-directional linear searches). This is achieved with independent coding of macroblocks and a feedforward compression scheme.

A summary of the DV encoding algorithm is discussed in section 2.1 and 2.2. The proposed single-chip DV camera architecture is presented in section 3. Finally, issues and considerations relating to programming the PEs and the controllers are discussed in section 4.

2. DV ALGORITHM OVERVIEW

The DV encoding algorithm is based on a feedforward video compression scheme. A detailed discussion on the DV encoding algorithm can be found in [1]. A brief outline of the DV algorithm is provided below.

2.1 DV Formatting

The image (720x576 pixels) is first formulated into macroblocks (MB) each containing 8x8 pixel blocks of 4 luminance (Y) blocks and 2 chrominance blocks, Cr and Cb. In the 625/25 system, 4:2:0 color subsampling is employed, whereas, in the 525/30 system, 4:1:1 color subsampling is employed. Five MB's are put together to form a segment. These 5 MBs are "shuffled" meaning that they are taken from different parts of the image as shown in Figure 2. Motion adaptive discrete cosine transforms (DCT) are performed on each of the 8x8 blocks in the MB's. These MB's then undergo error correction coding and channel modulation, formatted into synchronization blocks, and finally redistributed, "re-mapped", into superblocks (a cluster of 3x9 MB's). These superblocks are then mapped into tracks and written to the cassette medium.

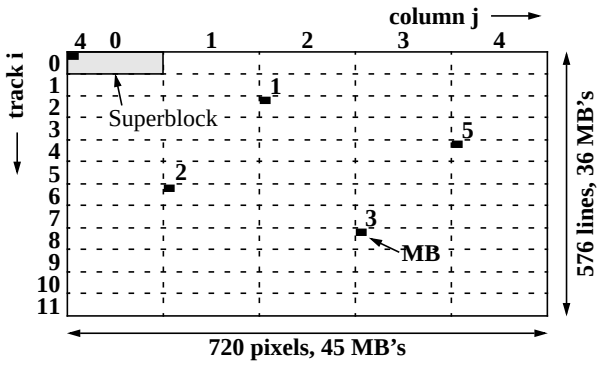

Figure 2: *Selection of MBs for segment construction, "shuffling".*

2.2 Motion Adaptive DCT

The motion adaptive DCT employs a motion detector from which a motion indicator signal is generated. Based on this indicator, one of two types of DCT algorithms, a standard 8x8 DCT algorithm or a motion based DCT algorithm, is applied. The motion based DCT algorithm performs the normal DCT algorithm on the horizontal pass and an N/2-point DCT algorithm [1] on the vertical pass. The resulting DCT block and the corresponding motion indicator are fed to the feedforward adaptive quantization unit.

2.3 Feedforward Adaptive Quantization

The goal of the feedforward adaptive quantization is to control the post-compressed bit rate such that the compressed data from different segments is approximately fixed rate (as required by trick plays). The feedforward quantization unit first computes the "activity" (i.e. energy level or information content) of the DCT block. This "activity" value is used to select the quantization class associated with the block. There are 4 quantization classes and 16 quantization strategies used in the DV standard. After a quantization class is chosen, the corresponding quantization strategies are used to quantize the DCT block. The 16 quantized DCT blocks are variable length coded and the total word lengths of the variable length codes are extracted. These word lengths are combined with word lengths from other MB's belonging to the same segment to calculate the segment data rate. A quantization strategy is chosen which corresponds to the segment data rate (calculated based on that quantization strategy) that is closest to the ideal "fixed" segment data rate. Lastly, the entire segment is quantized with the chosen quantization strategy before DV streaming (error/channel coding and formatting).

3. Single-Chip DV Camera Architecture

The proposed architecture is described in the context of the DV algorithm requirements and issues discussed in section 2.

3.1 Controllers

There are two controllers (MC and APC) that controls signal and data flows for the entire camera chip. The main controller (MC) is the primary controlling unit whereas the array processor controller (APC) performs simple decoding function to pipeline instructions to the PEs.

In addition to performing control monitoring, the MC also processes data sent from the PEs through the output comparator (OC). This allows a single controlling unit to handle output for-

matting as well as streaming protocol thereby reducing programming effort for streaming synchronization. The MC is separated from the PE instruction pipelining control for the reason that MC is needed to perform post-processing of compressed data. This separation enables the MC to run in parallel to PE instruction pipelining. In addition, this separation also implies a separation of coding tasks into PE programming and MC programming which results in the separation of the physical memory spaces for storing PE/MC program codes. To reduce the amount of controlling overhead, the PE instruction pipeline control unit (APC) is simplified to perform simple branching and pipelining of instructions while the MC performs an overall monitoring of the PE program flow and processing status. This control overhead to the MC unit is small compared to the processing overhead needed to post-process the compressed PE data. As a result, an additional control unit for monitoring the PE program flow is not required. Also, since the MC does not have direct control over the PEs, the MC needs to access status information from the PEs via the OC and interprets the result to oversee the program flow of the PE instruction pipelining.

3.2 Output Comparator

The output comparator (OC) serves as a bridge between the controlling units and the PEs. It delivers information both ways and can be used for several controlling as well as data transferring tasks. It is required by DV encoding primarily to transfer bit cost information from the "shuffled" MB's to the MC to perform optimal quantization strategy search. It also serves to communicate, to the MC, PE status information such as PE execution completion (for data dependent operations in which program codes are stored in the PE program memory), lookup pipeline status, etc. It is used by the MC to communicate the optimal quantization strategy to the PEs. The OC is also needed in several other image processing algorithms. Examples include auto whitening and auto exposure control. Although data sharing is required across the entire sensor array, the amount of data transfer is limited implying s that OC is not a significant contributor to power consumption.

3.3 Processor Element Architecture

The PE architecture is illustrated in Figure 3. Since the DV algorithm does not require access to image data belonging to other image columns, a simple direct-memory-access (DMA) unit is implemented. The DMA unit serves as an interface between the PE and the frame buffer such that the DRAM access time is decoupled from the processor cycle time.

The block visible RAM (16x16 pixels) and the auxiliary RAM (8x8 pixels) provide small but flexible buffering of image blocks. These local memories are addressed by a 2-D vector address with an optional automatic offset compensation. This provides the flexibility for implementing efficient algorithms such as fast DCT. The block visible RAM can also be used to store 16-bit words. This is needed for temporary storage of the DCT coefficients. The auxiliary RAM provides a temporary buffer for buffering past image pixels for motion detection calculation. It can also store lookup coefficients in case the L-RAM cannot support the bandwidth demanded by the algorithm.

The ALU consists of a 24-bit adder, a 12x12 bit multiplier, a 24-bit barrel shift register, and a 24-bit accumulator. Two datapaths feed the ALU to provide high processing efficiency. Data can be

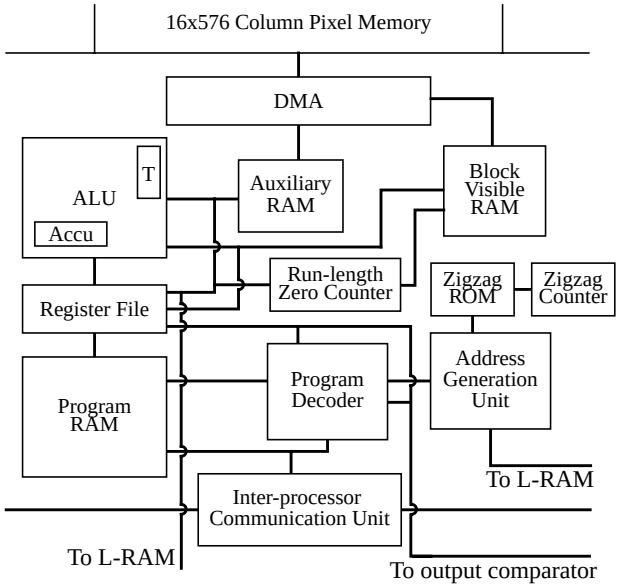

Figure 3: *Array Processor architecture.*

retrieved from the L-RAM, the block visible RAM, the auxiliary RAM, the register file, or the run-length zero counter. The register file provide fast and efficient access to intermediate process variables.

VLC and quantization are the most computationally intensive tasks in DV encoding. As a result, it is necessary to include a partial implementation of VLC in hardware. The zigzag units and the run-length zero counter are implemented to reduce the amount of overhead associated with VLC. The zigzag units feeds the address generation unit with hard-coded addresses. Two zigzag patterns are stored in the zigzag ROM, one for stationary blocks and the other for motion blocks. The run-length zero counter works in conjunction with the zigzag units to compute run, amplitude pairs. The output of the run-length zero counter can be sent directly to the L-RAM to perform word length lookup.

The program control units provide instruction decode as well as simple program flow control. Program flow control is needed only during data dependent processing (e.g. VLC codeword construction). In the data dependent processing mode, the instructions are locally stored in the program RAM.

The image column width is 16 pixels and is chosen based on the width of the MBs. By making the column width the same as the MB width, less programming overhead is needed to synchronize the transfer of the bit cost, associated with the feedforward quantization, to the MC.

3.4 Memory Requirement

DV requires 1 full sized frame and 4 partial frames of pixel memory. Part of the Frame Buffer (Figure 1) is a luminance frame (720x576) of the past image used for motion detection. The FB also contains three partial frames (see Section 3.5 below) with a resolution of 720x96 for buffering a portion of the current image in full color. The Output Frame Buffer buffers compressed data which requires 4 to 5 times less memory than a full sized frame. It is approximately 146KB. Memory requirement for DV is realizable with existing embedded DRAM technology.

3.5 Memory Optimization

Due to the need to buffer images on-chip, it is strongly desirable to optimize the memory usage for the algorithm at hand without compromising complexity. For DV encoding, buffering of newly acquired pixels can be a large overhead if the entire image needs to be buffered. An alternative buffering scheme is proposed to reduce the amount of buffering by a factor of 6. This scheme is illustrated in Figure 4. Memory usage of MB shuffling is analyzed to determine the minimum amount of memory to buffer the shuffled MB's. As shown in the figure, this requirement is satisfied when 6 full rows of MB's (16x720 pixels = 1 row of 45 MB's) are buffered. This result is obtained by observing the following. First, at any time instance, 9 segments are being processed by the PEs in parallel. If these 9 segments are taken from a row of MB's within a superblock (1x9 MB's), then shuffling requires that 4 addition rows of MB's be retrieved to formulate the segments. Refreshing specific regions in the sensor area introduces 2 dimensions of addressing overheads. Rather, 1 dimension (row-wise) addressing overhead can be achieved by computing an entire row (1x45 MB's) of pixels before retrieving the next row. 6 rather than 5 rows of MB's must be buffered due to the geometry of the shuffled MB's.

3.6 Lookup Memories

DV requires greater flexibility and amount of looking up coefficients due to quantization strategy searches. A power, area, and performance efficient architectural solution is required to support lookup memory update as well as access. In our design, shared lookup memories are provided to reduce the power and area overhead, and lookup coefficients are pipelined to reduce routing overhead.

3.7 Post-compression Coding

Final packaging/formatting of compressed data requires that the MBs be stored in sync blocks, error correction coded and channel modulation coded. It is desirable to incorporate error and channel coding onto the camera chip so as to reduce the power overhead incurred by data transfer to external processing units. Internally, error correction and channel modulation can be performed either at the controller level or in hardware since channel encoding hardware is simpler than decoding for ECC, which is the opposite for source coding.

3.8 PE Performance

Each individual PE consumes approximately 1 mW of power at a clock rate of 40 MHz and a supply voltage of 1.5V in a 0.2 micron CMOS technology. This amounts to a total power consumption for the array processors of approximately 45 mW. The number of cycles necessary to perform DV encoding for the 625/25 system at 25 fps is estimated to be 35 MIPs.

4. PROGRAMMING CONSIDERATIONS

The proposed architecture requires a different programming methodology than the conventional single processor and parallel DSP architectures. Programming complexity is often introduced with added architectural parallelism. Complexity in parallel systems normally reside in determining the optimal partitioning of resources and in synchronization of signal flow. The proposed architecture takes advantage of the high repetitiveness of image processing algorithms and the large-scale parallelism of the PEs

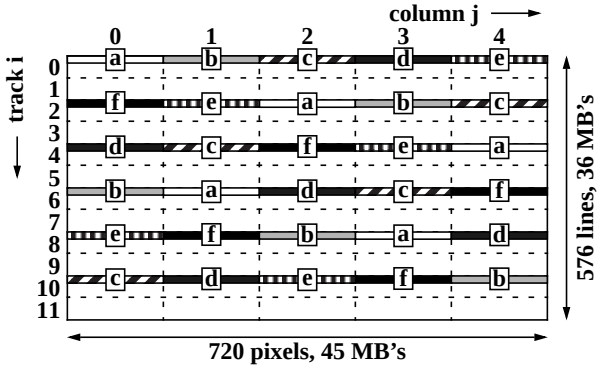

Figure 4: *Optimized buffering of pixel data for MB shuffling.*

to achieve full utilization of processing resources. The proposed architecture also achieves low programming complexity by separating processing and synchronization codes. This enables programmers to code PE almost independently of the MC.

Programming complexity can further be reduced by categorizing the data dependency of the image processing algorithm. Data independent codes can be pipelined and executed by the PEs on the fly. Data dependent codes must be stored in local PE program RAM. Also, local execution of data dependent algorithms incurs controlling overhead at the global level since new instructions cannot be pipelined until all processors have completed local execution. Converting data dependent codes into data independent codes is an alternative that may offer better performance. As a result, data dependency is correlated with the programming complexity for this architecture.

5. CONCLUSION

This paper proposes a parallel architecture for a single-chip digital CMOS video camera with real-time DV encoding and streaming capability. This architecture encompasses new technological developments in process technology, embedded DRAM, and CMOS image sensors to achieve a low power, highly computationally efficient solution to address the issue of integrated image sensing.

6. REFERENCES

[1] P.H.N. de With, et. al., "Design Considerations of the Video Compression System of the New DV Camcorder Standard", *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 4, November 1997.

[2] Jeff Y.F. Hsieh and Teresa H.Y. Meng, "Low-power MPEG2 Encoder Architecture for Digital CMOS Camera", *Proceedings of the 1998 IEEE Symposium on Circuit and Systems*, Session WAA4-8.

[3] Francky Catthoor, "Power-efficient data storage and transfer methodologies: current solutions and remaining problems", *Proceedings of the IEEE Computer Society Workshop on VLSI*, Orlando, Florida, april 1998, pp. 53-59.

[4] Teresa H.Y. Meng, Benjamin M. Gordon, Ely K. Tsern, and Andy C. Hung, "Portable Video-on-Demand in Wireless Communication," invited paper, *Proceedings of IEEE*, Vol. 83, No. 4, pp. 659-680, april 1995.