An Ultra Low Power Adaptive Wavelet Video Encoder with Integrated Memory

Thomas Simon and Anantha P. Chandrakasan, *Member, IEEE*

***Abstract*—This paper describes a low-power, single-chip video encoder intended for battery-operated portable applications. Design goals are minimizing system power as well as utilized bandwidth, and maximizing system integration. The encoder achieves competitive compression, with convenient bit rate scalability, using a peak power dissipation of several hundred μW on a video stream of 8-bit gray scale, 30 frame/s, and 128 $\times$ 128 demonstration resolution. Compression is performed using wavelet filtering, zerotrees, and arithmetic coding, all integrated on a single chip (3 million transistors, 1 cm^2, in 0.6-μm CMOS, operating at 500 kHz), with no external memory or control. Results do not include use of motion compensation, however, hooks are included at algorithmic and architectural levels to add motion compensation at the cost of power dissipation a few times higher, and more internal memory. In the absence of motion compensation, temporal correlation is still utilized through the use of simple frame differencing. The architectural centerpiece is a massively parallel, fine granularity SIMD array of processing elements (PE's). A mapping is made between small image blocks (4 $\times$4 pixels on the test chip) and PE's, with each PE containing both memory and logic required for its block. These results are obtained by careful coordination of design in a deep vertical manner, ranging from system, algorithmic, architectural, circuit, and layout, and designing simultaneously for all required algorithmic subcomponents.**

***Index Terms*—DSP, low power, scalability, video compression.**

I. Overview

An EXAMPLE context for the encoder is shown in Fig. 1. An imager senses the video stream which is digitized and compressed. Additional processing may be performed before transmission, such as encryption, error correction coding, framing, etc. The RF transmitter communicates data to some remote base station which may be receiving and transmitting to multiple sensors and other base stations, and returning data for display on the portable device.

Meeting system objectives requires consideration of two regimes of operation, with motion in the video stream and without (which for some applications may represent a large duty cycle). With motion, system power is dominated by the transceiver, and for medium to high distances, transmission power. In this case, system performance (power and bandwidth) is optimized through low data rates (i.e., good compression). Convenient on-the-fly scalability of the data rate (traded off for visual quality) is also desirable as a response to changing conditions such as channel quality, number of users, or application requirements. Of course, local computational cost of encoding should be kept low so as not to compete in the power budget. Most of this paper describes how these goals are achieved. In the absence of motion, compression yields almost no output data, leaving a largely inactive transmitter. Upon detection of prolonged periods of no motion the encoder can reduce its activity to just searching for new motion, which can be done cheaply with simple frame differencing and some kind of thresholding. The idle case is not discussed much here, but it is worth mentioning that reduction of peak encoder power also benefits this mode. Dissipation of the encoder, as well as imager and data converter, can be further reduced during idle periods by lowering frame rate, image resolution, or pixel precision used to detect new motion.

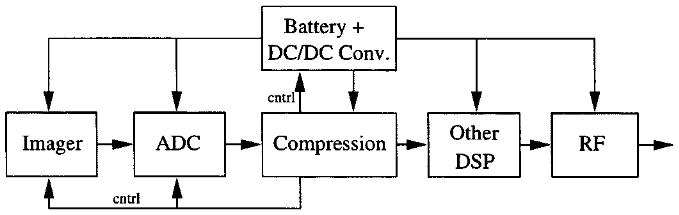


Fig. 1. Block diagram of a portable video sensor.

In order to ward off skepticism, it is worth commenting early on the test chip's small pixel resolution. No limit exists due to poor scaling. Most of the chip power scales linearly with resolution, if process technology were held constant. The test chip is limited purely by area. Because the SIMD array integrates all required memory internally (amounting to three entire frame buffers, 12 bits wide), its area (about 1 cm^2) is dominated by memory elements. Meanwhile, performance required from logic devices is very modest, due to the large amount of parallelism available. Therefore, this application is a perfect candidate for a mixed DRAM/logic technology. However, the chip reported here is implemented in a logic-only process, and a fairly dated one at that (2.4 μm contact pitch, no local interconnect, no stacked vias, etc.). The authors believe that the performance obtained here is indicative of what the techniques used can deliver, and that realistic resolutions would experience similar dissipation levels, with the benefits of modern fabrication technologies being spent on image resolution instead of further power gains.

II. Previous Work

The encoder described here performs all algorithmic components relevant to video compression on one chip, with power

Manuscript received September 28, 1999; revised December 10, 1999. This work was supported in part by the Defense Advanced Research Projects Agency under Contract DAAL01-96-2-0001.

T. Simon is with High Speed Solutions, Hudson, MA 01749 USA.

A. P. Chandrakasan is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: anantha@mtl.mit.edu).

Publisher Item Identifier S 0018-9200(00)02653-6.

dissipation of about $1/2$ mW. These include simple frame differencing (or motion compensation at the cost of an additional 4 mW), wavelet transform, quantization, zero-tree coding, and adaptive arithmetic coding. By contrast, known previous work on low-power video compression falls in one of two categories. First are piecemeal implementations of single algorithmic subcomponents with dissipations ranging from mWs to 10 s of mWs. Examples can be found in [1]–[6]. Second are algorithmically more substantial coders such as [7]–[9], but with much higher power levels, 100 mW to 1 W. In addition, many of these designs rely on external memory, some on external higher control functions, some achieve sub-par compression, and some deliver fixed bit-rate output.

III. Algorithm

The compression algorithm basis is the EZW algorithm reported in [10], in conjunction with wavelet filters found in [11] and arithmetic coding described in [12]. The EZW algorithm is chosen as starting point for three reasons: competitive compression results, which are achieved through a combination of a hierarchical, multiresolution wavelet transform, successive approximation quantization (bit-plane coding), zero-tree coding of coefficient significance maps, and adaptive arithmetic symbol coding; opportunity for aggressive supply voltage scaling enabled by copious and convenient data parallelism available, and that on a fairly modest computational burden to begin with; and finally, the embedded nature of the output bit stream generated, which more than suffices for the scalability needed here. In fact, bitwise embeddedness is overkill, and by restricting scaling choices to larger granularities (bit-plane boundaries in the quantization step), some computational benefits are gained.

Single-frame compression proceeds in three steps. The image is decomposed into octave-sized subbands with repeated use of a separable wavelet filter. Coefficients are quantized by encoding a chosen number of bit planes, one at a time from MSB toward LSB. The encoding of a bit plane involves two passes, termed dominant and subordinate. Dominant passes identify new nonzero coefficients in their respective bit planes with the aid of zero-trees, data structures which organize coefficients by location and spatial frequency in order to incorporate natural image statistics in the encoding process and thereby aid compression. Subordinate passes encode LSB's of coefficients already identified as nonzero. Finally, dominant and subordinate symbols are entropy coded with an adaptive arithmetic coder.

Two categories of modifications are made to the algorithm, both for the sake of low-power operation, achieved either by reducing operations or further localizing communication patterns. The first gives up some granularity of the embedded output property. The second sacrifices small amounts of compression performance. The first category includes elimination of sorting significant coefficients by magnitude, interleaving of dominant and subordinate symbols of the same bit plane, and a myriad of dominant symbol scanning order choices designed to coordinate communication with the SIMD architectural model and reduce switching activity. The second category includes exclusion of subordinate and sign bits from adaptive treatment by the arithmetic coder, reduction of conditioning used to encode dominant symbols, and approximation of probability estimates used by the coder. Details concerning all these changes can be found in [13].

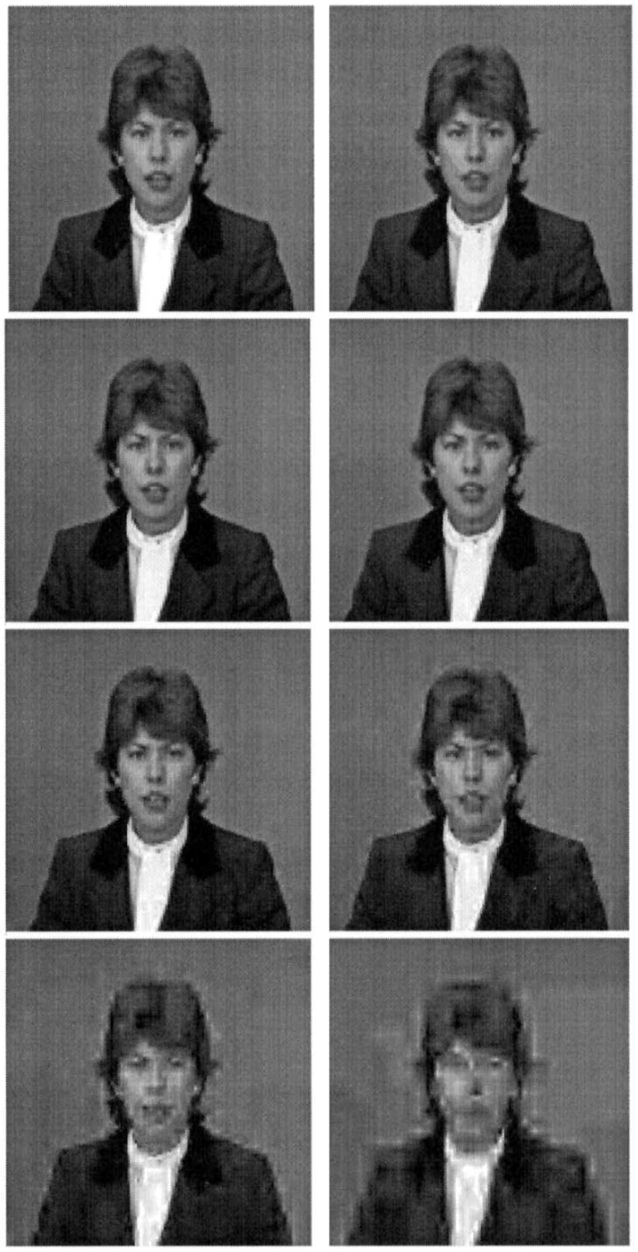


Fig. 2. Compressed frames of head and shoulders sequence. From left to right and top to bottom: original, compress $\times 7$ (PSNR 44.63), $\times 12.8$ (PSNR 43.59), $\times 23.8$ (PSNR 41.36), $\times 49$ (PSNR 37.89), $\times 115$ (PSNR 33.77), $\times 300$ (PSNR 29.85), $\times 967$ (PSNR 26.17). Compression factors are averages over many frames. PSNR's are for this particular frame, but are indicative for all frames.

IV. Extension to Time

Due to test chip memory constraints, the implementation algorithm is extended to time with simple frame differencing. Motion estimation and compensation requires more memory for shifting the previous frame (as well as a great deal more computation, whose estimated power cost is 4 mW). For purposes of error recovery and resynchronization, frame differencing is interrupted and reset every 16 frames. Fig. 2 shows compression

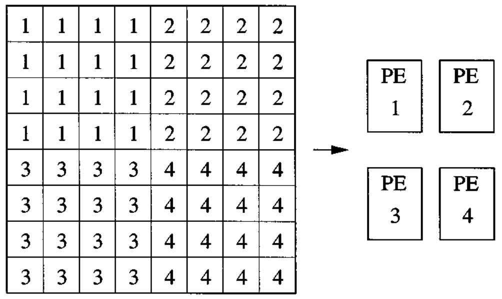


Fig. 3. Mapping of an 8×8 image onto SIMD array. Pixels are labeled with their assigned PE index.

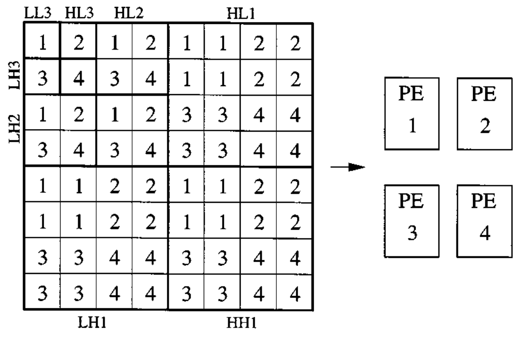


Fig. 4. Mapping of subband coefficients onto SIMD array. Bold outlines separate different filter levels.

results on a head-and-shoulders sequence with little motion. Images are the same frame number from sequences with varying compression. For motion compensation, the authors envision future work toward low power and especially for compatibility with the SIMD architectural model adopted. First, it is expected that some kind of frame-to-frame motion vector prediction, such as in [14]–[17], will be useful. Second, it seems promising to substitute independent motion vectors for every block with a few globally optimized vectors which each block may opt into, or default to the zero vector [13]. Use of a few global vector sets is SIMD-compatible, would lower the bit cost of vector maps, and might improve frame-to-frame vector correlation by identifying real motion of significant objects.

V. Architecture Overview

Approximation of probability estimates in the arithmetic coder is singled out as an example here. The goal is to eliminate expensive multiplies and divides by approximating relevant probabilities or their sums by inverse powers of two (thereby reducing those operations to shifts). While conceptually simple, three separate steps are required to obtain good results. Alphabets of more than two symbols are broken down into a hierarchy of two-symbol sub-alphabets, two-symbol alphabets are ordered by probability after each symbol (and new estimate by the adaptive coder), and probabilities are approximated according to a particular scheme (again, with details in [13]). This approximation completely eliminates multiplies and divides from the coder, at the expense of only 1.5% increased bit rate.As described, the algorithm exhibits a great deal of data parallelism which, if matched by an appropriate architecture, can be leveraged to operate circuits in an energy efficient, low voltage manner. If properly organized, the required communication is highly localized. A fine-granularity SIMD array of mixed memory and logic, with fairly simple interconnect, is a natural choice. Distributed logic provides parallel computational throughput, while finely subdivided memory delivers bandwidth where it is needed, without communication overhead, and with reduced energy per load/store due to small bit lines. Meanwhile, the SIMD organization minimizes or eliminates control overheads such as independent instruction execution, interprocessor synchronization or handshake, addressable communication, etc., associated with more complex architectures. At first glance, it may seem counterintuitive to associate efficiency with central control and decoded instructions globally distributed to small granularity

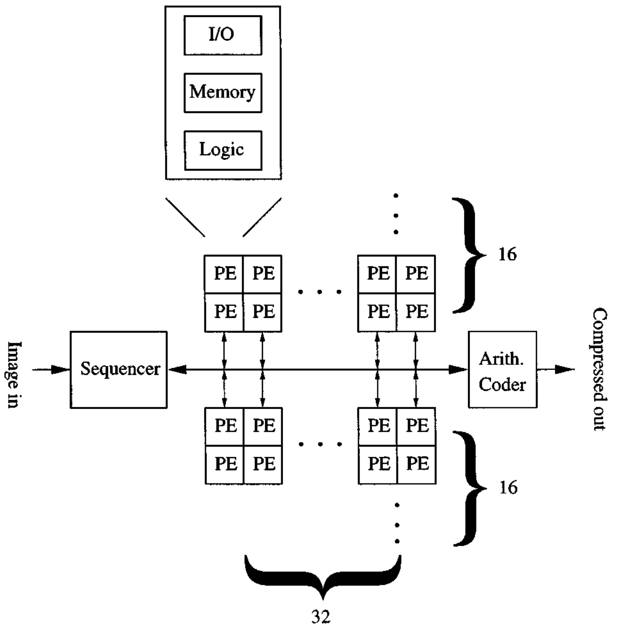


Fig. 5. Block diagram of architecture shows 32×32 PE array and peripheral circuitry, including arithmetic coder and sequencer. The center trench carries backbones for three of the four communication networks.

destinations. Small control overhead is usually associated with locality. However, in a SIMD architecture these two worlds blur and merge. The simplicity and regularity of a SIMD array affords customizable, efficient, abuttable layout, which places useful consumers of distributed instructions densely and efficiently packed together, both within and across PE's. In addition, slow, low voltage operation allows use of relatively modest instruction buffer sizes, further reducing control overhead. Wire resistance becomes negligible, and there is plenty of time to accommodate slow rise times. Note that, once supply voltage has been scaled close to device thresholds, it becomes more productive to reduce physical capacitance than to attempt further voltage reductions. These arguments apply equally to distributed clock signals, aided by use of simple, complementary transmission gate latches and flip-flops, which have good tolerance to poor rise times and skew, especially operated at low voltage themselves. In addition, both power and area are minimized by distributing separate, gated clocks to registers described below, instead of a global clock and enables.

Figs. 3 and 4 show how pixels and wavelet coefficients of an 8×8 image region are mapped onto four neighboring PE's.

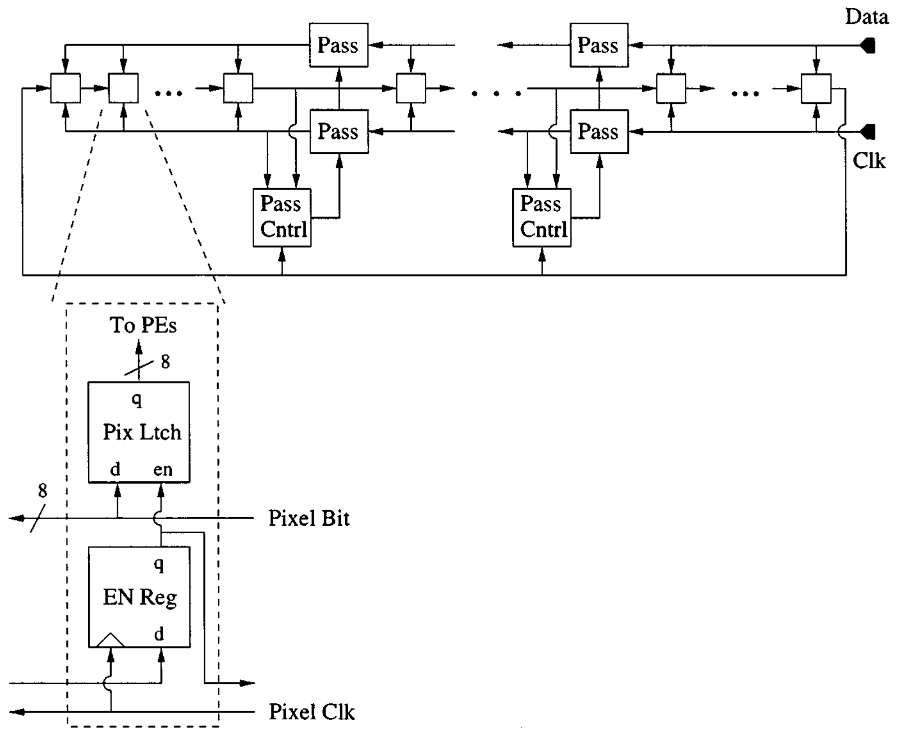


Fig. 6. Segmentation of pixel load network.

Coefficients in the example result from three levels of filtering. PE's use 12-bit-wide datapaths to limit rounding errors on coefficients and intermediate values so visually perfect reconstruction can be obtained with the appropriate compression choice. The 4×4 PE granularity is determined by dividing die area left over from required memory area into as many logic blocks as will fit.

Fig. 5 shows a block diagram of the architecture. The SIMD array consists of a square arrangement of 32×32 PE's, and is responsible for computation, communication, and memory requirements for most of the algorithm, including motion compensation, frame differencing, wavelet filtering, quantization, and zero-tree coding. Peripheral functions are handled by the arithmetic coder and sequencer, which decodes microcode instructions, generates timing signals, controls loading of incoming pixels and unloading of outgoing symbols to the arithmetic coder, schedules refreshes for PE DRAM cells, and keeps track of global state used to sequence through the microcode. Communication is served by four independent networks, a pixel load, symbol unload, one hop north–east–west–south (NEWS), and a 1-bit wired OR control network, distributed throughout the PE array and center backbone trench.

A load network performs serial to parallel conversion and distributes incoming 8-bit pixels to the PE array. To maintain SIMD access to PE internal datapaths and memory, PE's use holding registers as part of the second dimension of parallel conversion. This greatly simplifies row instruction decode logic. Pixel rows are captured in holding registers until all PE rows have received data, when a single SIMD instruction effects transfer to PE memories. Sixteen such load passes (one for each of the 4×4 pixels per PE), with a particular pixel scanning order, complete an image frame. The 16 SIMD load instructions are generated by the sequencer transparent to the microcode, to decouple the regular timing of loads from the unpredictable, branching instruction sequence. Collisions between pixel loads and microcode accesses is cheaply avoided by the implementation of PE memories, described below, which provides effectively free second memory ports.

A 4-bit unload network performs parallel to serial conversion and delivers instructions and EZW symbols to the arithmetic coder. Interestingly, despite already-compacted representation of the outgoing data, the unload network is cycled more frequently than the pixel load network. This is due to the bit-plane encoding used and the SIMD architecture. The discrepancy is that an overwhelming fraction of outgoing symbols are NOP's which the arithmetic coder ignores. Fortunately, this comes at a fairly small price. A large succession of identical symbols does not cause power-consuming switching on data wires. In addition, the average number of coder cycles is minimized by simple lookahead circuitry in the coder's interface which allows parallel to serial conversion to progress past NOP symbols in parallel with the coder's processing of substantial symbols, which is unpredictable and can take numerous cycles each.

Power is reduced in load and unload networks by using tristate busses and gated clocks (as opposed to shift registers). This drastically reduces row clock activity, avoids extra internal register node capacitance on data wires, and allows substitution of latches for edge triggered flip-flops. Further, both data and clock wires are segmented and electrically isolated to limit average switched capacitance. Vertical data wires are split in half by the center trench. Both data and clock in trench backbones are segmented by pass gates into quarters. Controlling circuitry at each segmentation point keeps track of serial–parallel con-

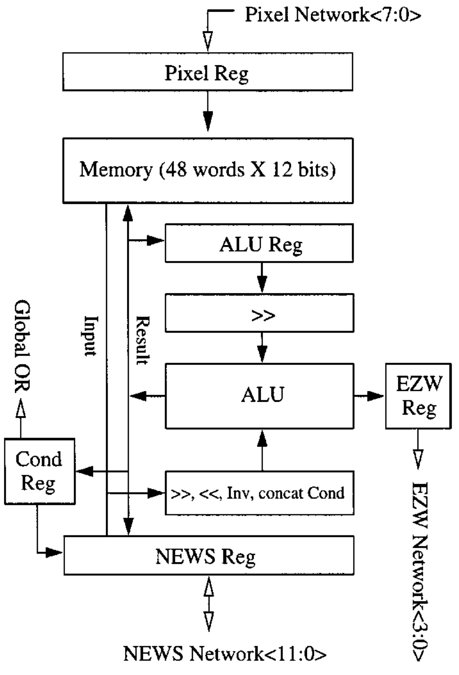


Fig. 7. Block diagram of PE. Memory and datapath are 12 bits wide.

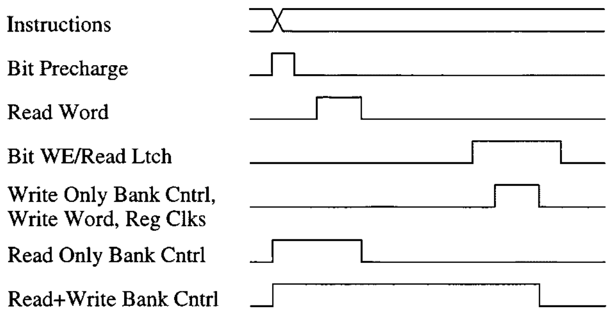


Fig. 8. PE timing diagram.

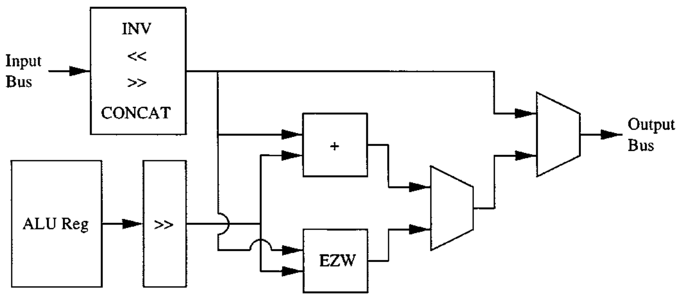


Fig. 9. ALU block diagram shows three modes of operation.

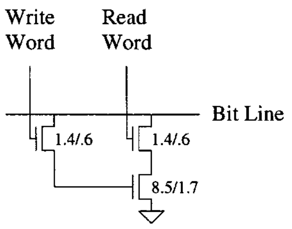


Fig. 10. 3T DRAM cell. Device sizes are in microns. The storage device is sized to obtain a 4-m/s refresh time. (The cell leaks away 10% of its charge in 4 m/s.)

version progress and only exposes consecutive wire segments as needed. To save the power of extra clock signals for pass control FSM's, they are operated from the clocks they control. The FSM's carefully gate off their own clocks at appropriate times and are asynchronously reset once each data row. Fig. 6 shows example segmentation points along the pixel load backbone. A 1-bit shift register (wrapped around at the end) passes a token around the ring to indicate which column of the current row receives data next. The token gates eight corresponding data latches which drive column wires. The left-most token is set on at reset. When a token passes a segmentation point from left to right, the FSM turns off data and clock pass gates, protecting wiring to the left from further switching activity for the rest of the row time. The returning token wrapped back from the last column asynchronously resets all three FSM's. As the reader can easily verify, the asymptotic power savings limit of many segmentation points is a factor of two off continuous backbone wires. This limit is approached rapidly, with three segmentation points producing a $3/8$ savings. Applying this reduction to both load and unload backbones results in 10% overall chip power savings.

A 1-bit wired OR network returns control information from the SIMD array to the sequencer, which can execute a branch instruction on the value. The wired OR network is precharged and each PE contains an evaluate device which may be set from any local data. There are no separate cut-off devices. Evaluate devices are forced off during precharge under microcode control.

Finally, a 12-bit wide one-hop NEWS network carries all intra-array data including pixels, wavelet coefficients, and EZW symbols or state information. The NEWS network operates in a SIMD fashion. All PE's transmit in the same direction at the same time.

VI. PE Implementation

A block diagram of a single PE is shown in Fig. 7. Operation statistics are taken into account to reduce PE power and area. For example, multiplies, which are required infrequently during wavelet filtering, are executed with serial shifts and adds, and using a paltry one-position shifter at that. On the other hand, a common set of logical operations used repeatedly to compute EZW output symbols and update state during quantization and encoding are special cased in a separate ALU mode, although the hard wired resources dedicated to this function are a modest, dozen or so, two or three input gates.

PE's use a simple read, modify, write timing (diagram shown in Fig. 8 with signals as described below). The ALU register is used to preload a second argument for binary operations (often the write result of the previous cycle). The ALU register contents can be down-shifted 1 bit, with sign extension, for arithmetic operations. The main ALU argument from the read value can be modified with more options, which serve both arithmetic and logical operations. These are: up- or down-shift one bit, invert, and concatenate to the LSB the 1-bit conditional register

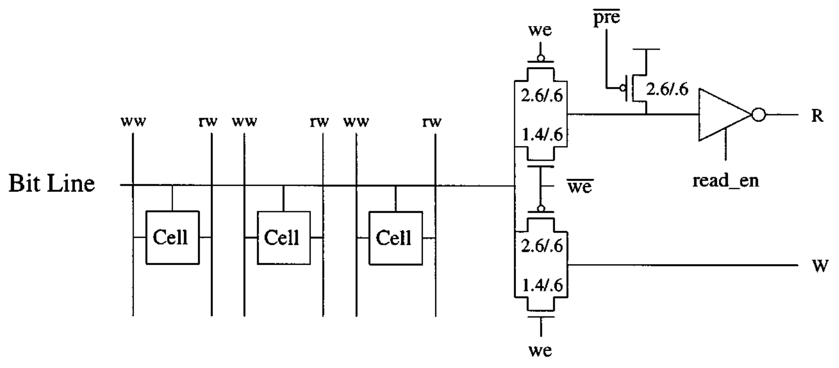


Fig. 11. PE bit line and peripheral circuits, including sense, latch, and write.

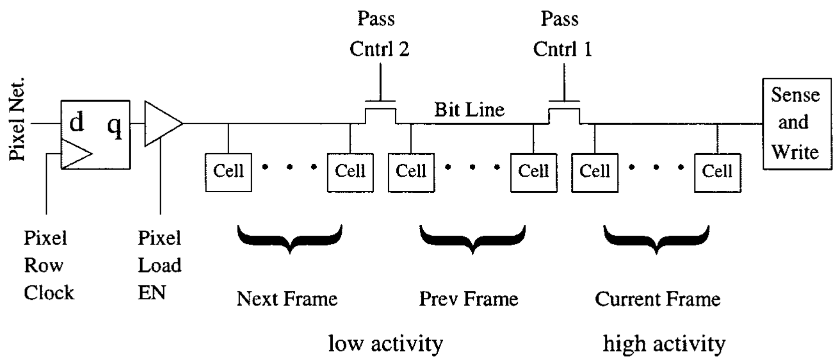


Fig. 12. PE bit line segmentation and pixel load port.

described below, one of whose functions is to gather logical results.

The ALU computes one of three possible functions, arithmetic, logical, and EZW specific, as shown in Fig. 9. Logic mode simply routes the shifted and inverted input to the result. The EZW function is executed once each bit plane for each wavelet coefficient. Inputs, packed into ALU register and main input, include sign, magnitude, previous encoding state, and zero-tree information for the coefficient. Two results are generated. A renormalized magnitude, with sign bit, updated state, and zero-tree information for child nodes, is routed to the output for storage and use in succeeding coefficients and bit planes. A 4-bit arithmetic coder instruction for the current coefficient and plane is loaded into the EZW holding register, which will drive the EZW network in a following unload pass.

The NEWS register serves several purposes besides NEWS network interface (along with muxes and tristates not explicitly shown). It is often used as a generic, one-word cache to store temporary results without the power cost of using PE bit lines and memory. In conjunction with the conditional register, the NEWS register can perform conditional writes, the SIMD equivalent of branch instructions. The conditional register can act as NEWS register load enable, process generic 1-bit logical results such as collated significance and zero-tree information, and controls the evaluate pulldown device to the wired OR network.

PE memories are hardly more than register files of 48 words [4×4 pixels $\times$ 3 words per pixel (current, next, and previous frame values, with some doubling as temporary workspace)]. Each PE memory also includes a forty-ninth ROM word containing PE dependent information for addressing, such as flags for the four array edges and even–odd row and column. For the logic process test chip, since pseudo-DRAM cells are used, it only costs marginally more area to use 3T DRAM cells, as shown in Fig. 10 (especially in light of pitch matching constraints with PE logic elements and aspect ratio limits). This allows large bit line voltage swings to be developed, as opposed to that with charge sharing between a cell and heavily loaded bit line. Therefore, a bit line sense amp is eliminated in exchange for more bit line power. This is a good tradeoff with small bit lines. In addition, 3T cells do not suffer destructive reads, so power and peripheral circuit complexity is saved by foregoing corresponding refreshes. The simplicity of resulting bit line peripheral circuitry is shown in Fig. 11. Note that the tiny bit lines do not require extra write buffers. A simple write pass gate shares timing signals with the read latch. Of course, write word lines are driven higher than V_{dd} to allow full swing at cell storage nodes. This is not a significant power burden due to the low switching frequencies of word lines. Low power up converters are discussed below.

Fig. 12 shows a cheap optimization which reduces bit line power. The three words corresponding to each pixel experience widely varying access frequencies, large (99%) for the current frame, small (1%) for next and previous frames. By segmenting the bit line with pass gates, most memory accesses to the current frame, placed closest to the business end of the bit line, only switch $1/3$ the bit line capacitance. The area penalty per segmentation point of one N-channel gate and control wire is quite small, and while controls are driven to the higher write word line voltage, their switching frequencies are negligible. As with write drivers, the pass gate series resistance is made tolerable by the light loading of the small bit lines. Fig. 8 includes three possible timings of bit line pass controls for read only, write

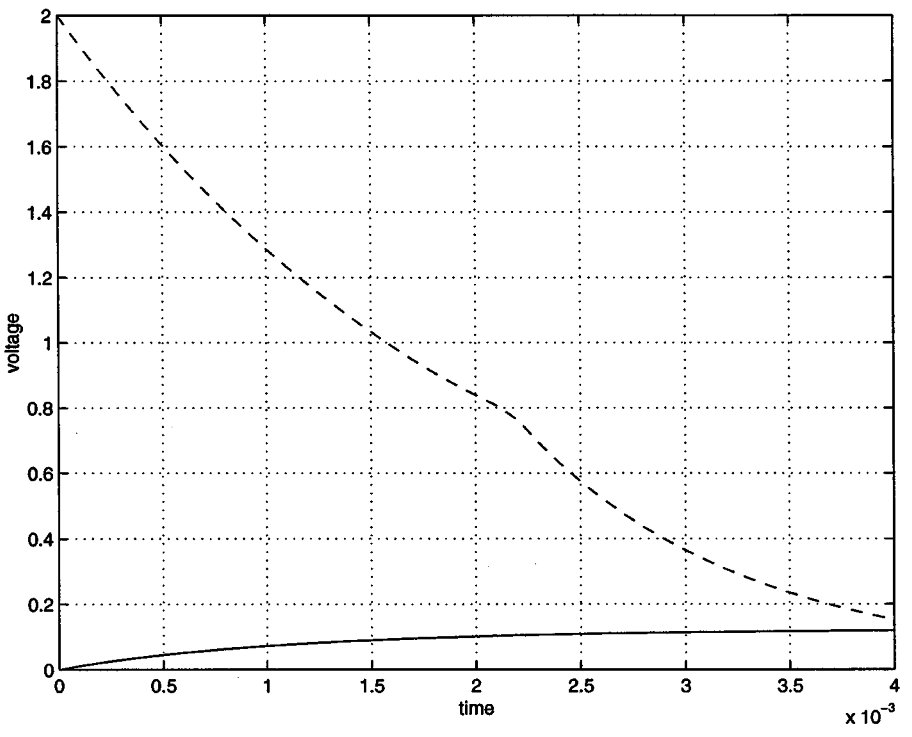


Fig. 13. DRAM leakage waveforms show cell leakage to opposite state bit lines.

only, and same cycle read and write access past a segmentation point. Given the access statistics, the segmentation scheme effectively reduces bit line power by a factor of three and chip peak power by a further 10%.

As shown in Fig. 12, separate bit line segments provide a back door which constitutes the cheap pseudo second memory port alluded to above. This allows next-frame pixels to load transparent to microcode activity, without the need for special collision avoidance circuitry. It is incumbent on the microcode to avoid accesses to the back of the bit lines after a certain time. However, this is made easy by the synchronization of the start of each frame computation with the loading of the last pixel pass, and the use of holding registers to gather new data until the next load. This makes an ample $1/16$ of the frame time available for safe, next-frame, microcode accesses.

To minimize power, different PE DRAM cell refresh mechanisms are tailored to varying requirements. During extended periods of SIMD activity, explicit refresh instructions are included in the microcode for locations not covered by computational activity. During idle periods (possibly lengthy with no motion in the video stream), needed values are refreshed by automatically scheduled sequencer instructions (without collision avoidance circuitry), while others are allowed to decay. Finally, special treatment of bit lines during idle periods extends the refresh time at no cost. Very asymmetric decay times are experienced by the two state cases due to subthreshold conduction in the write access device. A high storage voltage leaking to a low bit line decays faster than low from high because the second case is self-limiting due to falling (in fact, negative) gate–to–source voltage on the access device as the leakage progresses. Fig. 13 shows spice simulations of the two subthreshold leakage cases. The effect of subthreshold conduction on refresh time is minimized by keeping bit lines continuously precharged during idle periods. Refresh time is thus largely dependent on the remaining diffusion diode leakage. The relative benefit is highly process dependent. Note that no power is wasted switching precharge signals during idle periods. The sequencer enters a timing bypass mode which continuously holds the precharge signals asserted. For example, on the test chip, refresh times of 4 m/s. were achieved with device sizes shown in Fig. 10. A factor of four on the refresh time can be attributed to idle bit line precharging. This is a noticeable benefit even during peak motion periods when refresh operations amount to 8% of all SIMD cycles (with the 4 m/s. refresh times). This becomes a dominating factor during idle, no motion periods when the encoder only performs frame differencing and thresholding to scan for new motion.

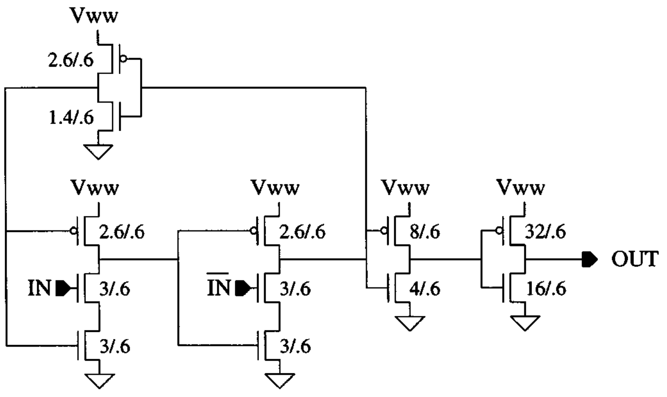


Fig. 14. Low power up converter and buffer. Self-resetting up converter avoids ratioed circuits and overlap current. IN and $\overline{\text{IN}}$ are smaller swing signals, from V_{dd}. OUT is driven from V_{ww}, a separate, higher voltage write word line supply. (Some weak anti-leakage keeper device are not shown.)

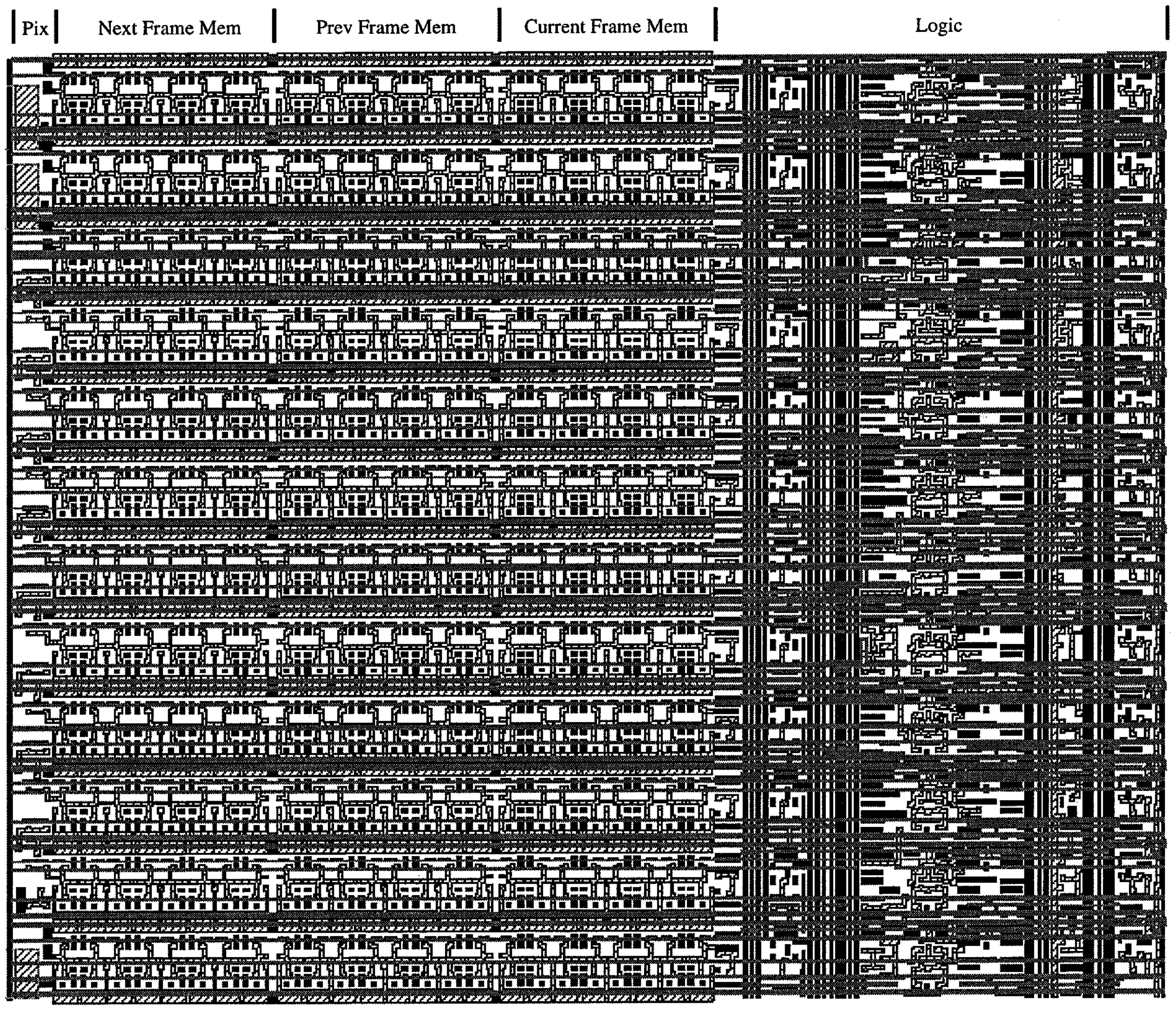


Fig. 15. PE layout (metal 1 and 3).

VII. Sequencer

Besides decoding microcode instructions, scheduling pixel loads and refreshes, timing SIMD control signals, and coordinating the various networks and arithmetic coder, the sequencer contains some global state to control instruction flow. This includes a program counter, an accumulator for the image mean, a 4-bit counter which records the frames in a group before resetting and resynchronization with the decoder, a 3-bit counter which records the number of encoded bit planes, and a 1-bit flag set when significant wavelet coefficients are found which must be encoded in the current frame.

The sequencer uses the low-power up converter shown in Fig. 14 to drive write word lines and bit line pass controls from V_{ww}, a separate, higher voltage supply. Instead of the usual differential amplifier based design, this up converter uses a self-timed precharge scheme to avoid the use of process sensitive ratioed circuits and overlap current during settling.

VIII. Arithmetic Coder

The arithmetic coder operates from an asynchronous and gated clock. The clock is gated to avoid switching during possibly long idle times. It is asynchronous to allow for higher frequencies than the SIMD array, without requiring the use of a frequency multiplier. The SIMD array operates at the incoming pixel rate for convenience (about 500 kHz), which is more than enough to cover the array's required computational throughput. The higher frequency of the arithmetic coder is required to cover its worst case throughput. The coder operates serially on the outgoing symbol stream. On the test chip, the coder clock frequency is ten times higher than for the sequencer and SIMD array, and is generated by a simple ring oscillator. Fortunately, neither power or critical path delay is an issue because of the coder's simplicity, especially after the elimination of multiply and divide operations, as discussed above.

IX. Chip Statistics

The test chip is fabricated in a 0.6 μ drawn channel length process with V_T's of 0.7–0.9 V. Table II shows how area is apportioned among PE functions. Fig. 15 shows a PE layout. PE's are designed to abut vertically (after flipping) and horizontally. Table I gives statistics for the entire chip, whose die photo is shown in Fig. 16. Table III gives peak power estimates, assuming $V_{dd} = 1.5$ V and $V_{ww} = 2.5$ V. The power estimate

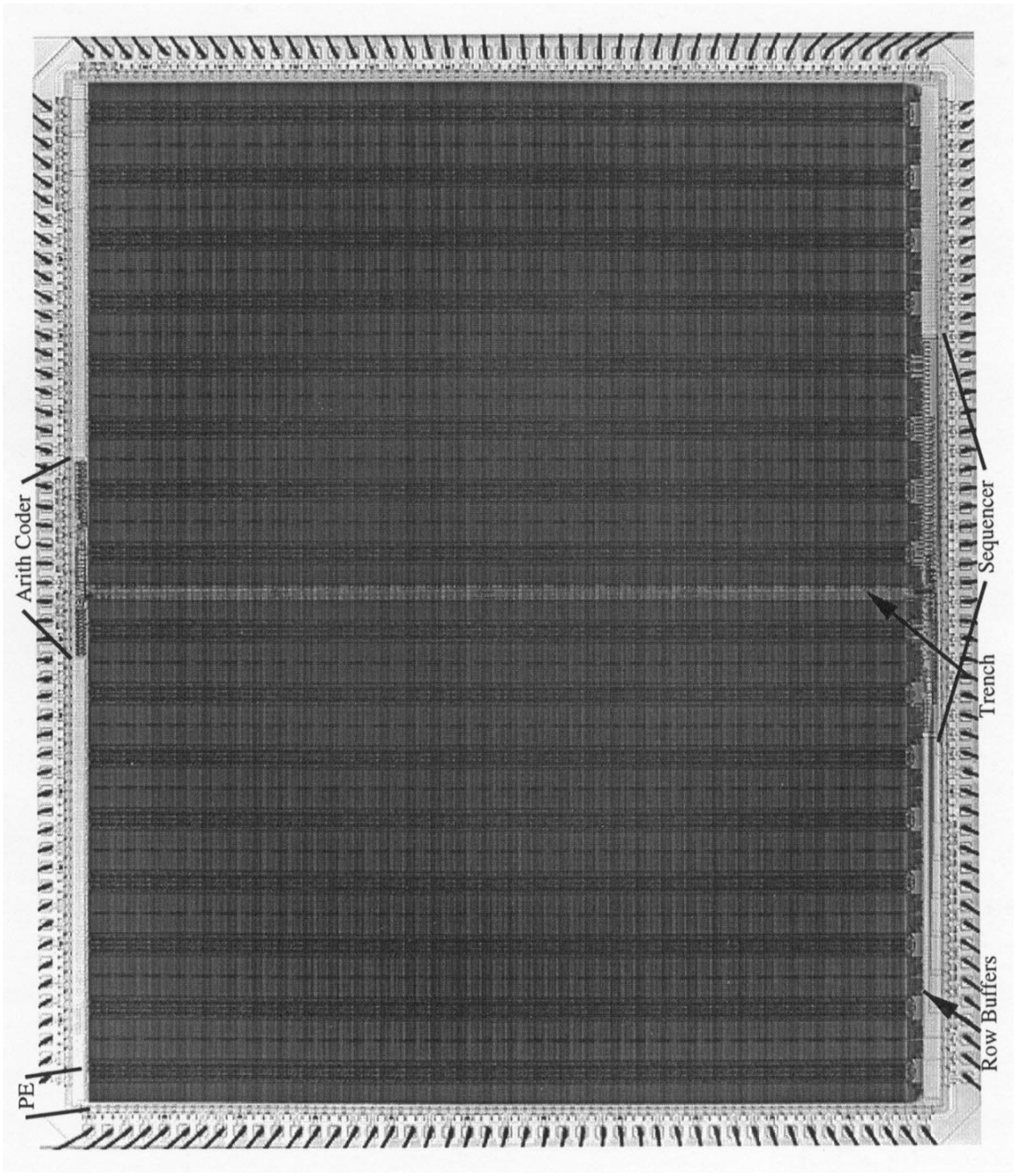


Fig. 16. Test chip die photo.

TABLE I
TEST CHIP TRANSISTOR AND AREA STATISTICS BY CATEGORY. TOTAL AREA IS 9980 $\mu \times 11\,580\ \mu$

Function	Transistors	% Area
SIMD array	3.06M	79
Trench	6.2K	1.2
Inst. distribution	3.7K	1.7
Arithmetic coder	3.5K	0.25
Sequencer	6.4K	0.58
I/O	1K	15.5
Total	3.08M	98.23

TABLE II
PE AREA ($269\ \mu \times 331\ \mu$) USAGE. INCLUDES 3000 TRANSISTORS (1752 MEMORY, 1234 LOGIC)

Function	% Area
Memory	57
Addr ROM, bit line peripherals	4.5
Pixel load reg. and network	4
NEWS reg. and network	11
ALU and ALU reg.	18.5
EZW reg. and logic, cond. reg.', wired OR	5

does not include I/O, which would be determined by system integration choices. Test chip I/O's are operated from a separate separate 5 V supply for interface to off-the-shelf testing and measurement parts. Power drawn from the 5 V I/O supply is also not included in measurements reported below. In addition,

TABLE III
CHIP PEAK POWER ESTIMATE, BY CATEGORY. $V_{dd} = 1.5$ V

Component	Power (μW)	% of Total
SIMD instruction distribution	220	38
PE bit line	30	5
PE datapath and NEWS	180	32
Pixel load backbone	12	2
Pixel load vertical	10	2
EZW unload backbone	60	10
EZW unload vertical	30	5
Arithmetic coder	0.1	0
Misc. Sequencer	30	5
Total	572	

a small, on-chip microcode ROM (1.5 K words, 20 bits) is omitted. An off-chip EPROM is used for debugging convenience. The omitted ROM's power dissipation is expected to be negligible.

X. VERIFICATION AND RESULTS

Chip functionality was verified by performing binary comparisons of the output stream with that of a software emulator over many sample video streams, and by software decoding of the outputs. Functionality was verified down to V_{dd} of 1.5 V, with V_{ww} 1 V higher. Table IV shows measured power for $V_{dd} = 1.5$ V. The encoder compresses full-motion video with dissipation a little over $1/2$ mW. These numbers agree quite well with estimates, including the effects of bit planes encoded, SIMD instructions executed, output bit rate, etc.

XI. CONCLUSIONS

Compared to known previous work on low-power video compression, the results achieved in this work are fairly surprising. While roughly maintaining the compression performance of a known, competitive algorithm, this work has resulted in a test implementation which integrates the functionality of several other efforts, with total power dissipation an order of magnitude lower than any of them.

This can be attributed to two principal factors. The design approach was broad, meaning that from the outset, the requirements of all necessary components were considered together. This includes: memory capacity, data distribution/loading, motion compensation, filtering/transformation, quantization, symbol coding, and entropy coding. Second, the design approach was deep, meaning that considerations at all design levels were taken into account and allowed to influence each other. These include: system, algorithmic, architectural, circuit, and layout. As such, the design was more an annealing process than a flow chart. For example, the author was as likely to make an algorithmic change after being frightened by a transistor

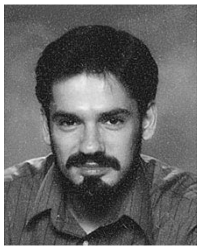
TABLE IV
MEASURED CHIP POWER FOR $V_{dd} = 1.5$ V AS A FUNCTION OF ENCODED BIT PLANES. BOTH MOTION AND NO MOTION CASES ARE TAKEN FROM THE HEAD AND SHOULDERS SEQUENCE SHOWN IN FIG. 2. THE NORMALIZED BIT RATE (OUTPUT/INPUT BIT RATE) AND PSNR IS FOR THE MOTION CASE

Bit	Normalized	PSNR	Power(μW)	
Planes	Bit Rate		Motion	No Motion
4	.0087	33.5	492	488
5	.0204	37.5	540	512
6	.0420	40.9	590	561
7	.0781	43.3	639	615
8	.1429	44.3	675	648

schematic, as to devise an efficient circuit for a given job. A comprehensive list of techniques, optimizations, and hacks that were brought to bear cannot be summarized here. For a complete description, the reader is referred to [13].

REFERENCES

[1] W. Namgoong, M. Devenport, and T. Meng, "A low-power encoder architecture for pyramid vector quantization of 2-D subband coefficients," *VLSI Signal Processing VIII*, pp. 391–400, 1995.
[2] B. Gordon and T. Meng, "A 1.2 mW video-rate 2-D color subband decoder," *IEEE J. Solid State Circuits*, vol. 30, pp. 1510–1516, December 1995.
[3] A. Chandrakasan *et al.*, "A low power chipset for portable multimedia applications," in *IEEE Int. Solid-State Circuits Conf.*, 1994, pp. 82–83.
[4] T. Kuroda *et al.*, "A .9V 150MHz 10mW 4mm^2 2-D discrete cosine transform core...," in *IEEE Int. Solid-State Circuits Conf.*, 1996, pp. 166–167.
[5] T. Xanthopoulos and A. Chandrakasan, "A low power dct core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," in *1999 Symp. VLSI Circuits*, 1999, pp. 11–12.
[6] G. Yeh, Y. Lu, and J. Burr, "A low-power video motion estimation array processor," in *1996 Symp. VLSI Circuits*, 1996, pp. 162–163.
[7] Vanhoof *et al.*, "A scalable architecture for MPEG-4 embedded zero tree coding," in *IEEE Custom Integrated Circuits Conf.*, 1999, pp. 65–68.
[8] S. Molloy, R. Jain, and K. Nishibori, "A video CODEC chipset for wireless multimedia networking," *VLSI Signal Processing VIII*, pp. 381–390, 1995.
[9] M. Harr *et al.*, "A single chip videophone video encoder/decoder," in *IEEE Int. Solid-State Circuits Conf.*, 1995, pp. 292–293.
[10] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing*, vol. 41, pp. 3445–3462, December 1993.
[11] E. Adelson, E. Simoncelli, and R. Hingorani, "Orthogonal pyramid transforms for image coding," in *SPIE Visual Communications and Image Processing II*, 1987, pp. 50–58.
[12] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, June 1987.
[13] T. Simon, "A low power video compression chip for portable applications," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1999.
[14] W. B. Rabiner and A. P. Chandrakasan, "Network-driven motion estimation for wireless video terminals," *IEEE Trans. Circuits . Syst. Video Technol.*, vol. 7, pp. 644–653, August 1997.
[15] H. M. Hang, A. Puri, and D. Scilling, "Motion-compensated transform coding based on block motion-tracking algorithm," in *IEEE Int. Conf. Communications'87*, vol. 1, 1987, pp. 136–140.
[16] R. M. Armitano and R. W. Schafer, "Motion vector estimation using spatio-temporal prediction and its application to video coding," in *Proc. SPIE*, vol. 2668, 1996, pp. 290–301.
[17] S. Kim and C. C. Kuo, "A stochastic approach for motion vector estimation in video coding," in *Proc. SPIE*, vol. 2304, 1994, pp. 111–122.



Thomas Simon received the S.B., S.M., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, in 1990, 1994, and 1999, respectively.

He is currently with High Speed Solutions, Hudson, MA.

Anantha P. Chandrakasan (S'92, M'95) received the B.S, M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1989, 1990, and 1994, respectively.

Since September 1994, he has been at the Massachusetts Institute of Technology, Cambridge, and is currently an Associate Professor of electrical engineering and computer science. He held the Analog Devices Career Development Chair from 1994 to 1997. He is a coauthor of the book *Low Power Digital CMOS Design* (Nowell, MA: Kluwer) and a coeditor of *Low Power CMOS Design* (Piscataway, NJ: IEEE Press). His research interests include the ultra low power implementation of custom and programmable digital signal processors, distributed wireless sensors, multimedia devices, emerging technologies, and CAD tools for VLSI.

Dr. Chandrakasan has served as a Technical Program Co-Chair for the 1997 International Symposium on Low-Power Electronics and Design (ISLPED), VLSI Design '98, and the 1998 IEEE Workshop on Signal Processing Systems, and as a General Co-Chair of the 1998 ISLPED. He is the Signal Processing Sub-Committee Chair for ISSCC'00 and an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS. He is also a member of the Design and Implementation of Signal Processing Systems (DISPS) Technical Committee of the Signal Processing Society. He has served on the Technical Program Committee of various conferences including ISSCC, VLSI Circuits Symposium, DAC, and ISLPED. He received the National Science Foundation Career Development Award in 1995, the IBM Faculty Development Award in 1995, and the National Semiconductor Faculty Development Award in 1996 and 1997. He has received several Best Paper Awards, including the 1993 IEEE Communications Society's Best Tutorial Paper Award, the IEEE Electron Devices Society's 1997 Paul Rappaport Award for the Best Paper in an EDS publication during 1997, and the 1999 Design Automation Conference Design Contest Award.