

Low-Power Design of Memory Intensive Functions

David B. Lidsky, Jan M. Rabaey

University of California, Berkeley, California 94720 USA
 lidsky@eecs.berkeley.edu, jan@eecs.berkeley.edu

Abstract

Much of the recent efforts into low power design techniques neglects the impact memory accesses have on power consumption. This paper provides a summary of the techniques used in the design of a low-power vector quantizer (VQ), many or all of which can be applied to other memory intensive applications.

Design Approach

High level algorithmic decisions tend to have the most profound effect on power, but architectural and circuit level optimizations are also critical. For memory dominant applications, memory power can be reduced at the expense of slight increases in computation power.

A video image is vector quantized by breaking it into blocks (vectors) of pixels that are mapped to a codebook of probable vectors using mean squared error as the distortion measure [1]. The VQ was designed for the portable battery-operated Info-Pad terminal [2], which employs a codebook of 256 vectors. To process 30 frames/sec, a 4x4 pixel-vector must be compressed every 17.3 μ sec.

Algorithm: Reducing the number of operations saves power by reducing switched capacitance and reducing the critical path, enabling operation at lower voltages and frequency. Table 1 shows the computation complexity for three methods of quantization. A full search through the codebook requires intensive computation which can be significantly reduced through the use of a binary tree search (TSVQ), with only a minor cost in accuracy. The number of computations can be further reduced by combining terms of the differential search a priori, as indicated in Figure 1 [3]. The differential tree search was used in all of the implementations.

Architecture: In one approach, a centralized memory, processing element, and controller are time multiplexed as shown in Figure 2. This method requires 18 cycles to process one node of the tree, consuming a total of 146 clock cycles per vector at a frequency of 8.3MHz.

A technique often used in low-power designs is to read bytes of words in parallel to reduce throughput constraints enabling the reduction of clock speed and voltage [2]. This architecture cannot be used to reduce speed requirements in TSVQ since the location of the vector to be chosen is dependent on the results at the previous node of the tree. This same architecture (Figure 3), however, reduces the switched capacitance by reducing the number of memory accesses. Table 3 shows power savings by using parallel access for different memory and byte sizes.

In TSVQ each level of a tree has specific codevectors that are found only at that level. Therefore the memory can be partitioned into separate memories for each level of the tree [4].

Associated with each memory are identical processing elements and controllers. Since this architecture can be pipelined the critical path can be reduced drastically (Figure 4), allowing clock frequency to be reduced by eight, and the supply voltage to be dropped from 3.0V to 1.1V.

The distributive memory architecture switches significantly less capacitance reading its codevectors than the centralized case since there is less overhead in reading from smaller memories (Figure 5). Though there are now eight controllers and eight processors on chip, they are clocked at one eighth the frequency, so the energy dissipated per vector does not change. Interblock power is negligible since it is clocked at an eighth of the clocking frequency.

Circuit: All circuits were designed using UC Berkeley's low-power cell library which employs minimal transistor sizes to reduce parasitic capacitance. To minimize transition activity, registers with gated clocks hold the previous inputs to adders during idle clock cycles. To reduce clock power, a single clock is used and registers are implemented using minimum sized True Single Phase Clock Registers (TSPCR). Memory is implemented with a low-power, on-chip SRAM whose bit-lines are pre-charged to an NMOS threshold voltage below the supply voltage to reduce swing [2].

To illustrate the multi-level approach to power reduction, three architectures were implemented: single memory with serial access, and single and distributive memory TSVQ encoders with parallel memory access. At this time, complete testing has not been accomplished, so all numbers are from extensive IRSIM and SPICE simulations. Power and area numbers are summarized in Table 2. Modifying the centralized design to include parallel memory accesses reduced the switching capacitance by 60%. The capacitance, as well as the critical path, is further reduced by memory partitioning.

To reduce power in memory intensive functions, proper choice of algorithm and architecture is essential. Algorithmic level decisions have been shown to reduce computational complexity and to reduce the number of memory accesses by a factor of 30. Architectural optimizations further reduced power by a factor of 17.

REFERENCES

- [1] A. Gersho, *et al.*, "Vector quantization and signal compression," Kluwer Academic Publishers, Boston, MA, 1992.
- [2] A. Chandrakasan, *et al.*, "A low power chipset for portable multimedia applications," ISSCC, March 1994
- [3] W. Fang, *et al.*, "A systolic tree-searched vector quantizer for real-time image compression," *Proc. VLSI Signal Processing IV*, IEEE Press, NY, 1990.
- [4] R. Kolagotla, *et al.*, "VLSI implementation of a tree searched vector quantizer," *IEEE Trans. on Signal Proc.*, vol. 41, no 2, February 1993.

$$MSE = \sum_{i=0}^{15} (C_i - X_i)^2 \quad \begin{array}{l} C_i \rightarrow \text{Code-vector} \\ X_i \rightarrow \text{Input-vector} \end{array}$$

$$MSE_{ab} = \sum_{i=0}^{15} (C_{ai} - X_i)^2 - \sum_{i=0}^{15} (C_{bi} - X_i)^2$$

$$MSE_{ab} = \sum_{i=0}^{15} C_{ai}^2 - 2X_i C_{ai} + X_i^2 - (C_{bi}^2 - 2X_i C_{bi} + X_i^2)$$

$$MSE_{ab} = \underbrace{\sum_{i=0}^{15} (C_{ai}^2 - C_{bi}^2)}_{\text{Calculated a priori}} + \sum_{i=0}^{15} X_i^2 \underbrace{(C_{bi} - C_{ai})}_{\text{Calculated a priori}}$$

Figure 1: Mathematical Optimization

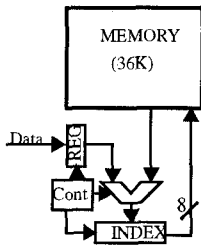


Figure 2: Centralized Memory

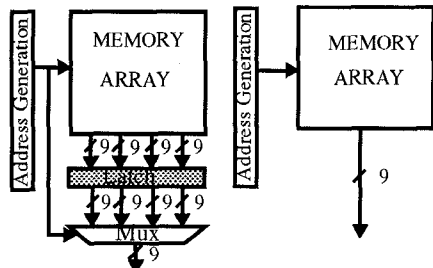


Figure 3: Parallel vs. Serial Memory Access

TABLE 1. Computational Complexity

| | Memory Access | Multiplication | Add/Subtract |
|--------------------------|---------------|----------------|--------------|
| Full Search | 4096 | 4096 | 8448 |
| Tree Search | 256 | 256 | 520 |
| Differential Tree Search | 136 | 128 | 136 |

TABLE 2. Simulation Results

| | Power | Memory Power | | Core Area | Operating Voltage | Clock Frequency |
|---------------------------|---------|--------------|----------|----------------------|-------------------|-----------------|
| | | % of Total | Absolute | | | |
| Centralized Serial Access | 6.96 mW | 60% | 4.15mW | 53.3 mm ² | 3.0 Volts | 8.30 MHz |
| Centralized | 4.28 mW | 46% | 1.95mW | 46.6 mm ² | 3.0 Volts | 8.30 MHz |
| Distributed | 412 μW | 28% | 114 μW | 92.9 mm ² | 1.1 Volts | 1.04 MHz |

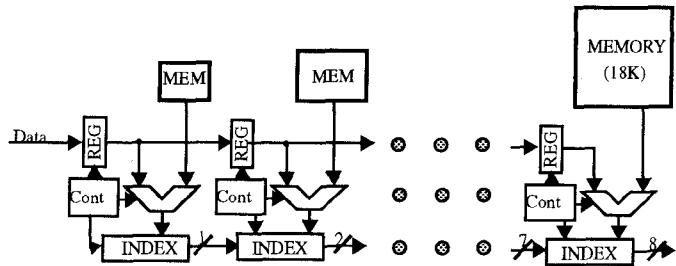


Figure 4: Distributive Memory

TABLE 3. Energy per vector access vs. Memory Organization

| | 4 access + 0 multiplex | 2 access + 2 multiplex | 1 access + 4 multiplex |
|----------|------------------------|------------------------|------------------------|
| 16 Kbits | 360.4 pJ | 320.0 pJ | 290 pJ |
| 8 Kbits | 243.2 pJ | 200.2 pJ | 186.3 pJ |
| 4 Kbits | 186.0 pJ | 150.6 pJ | 135.5 pJ |

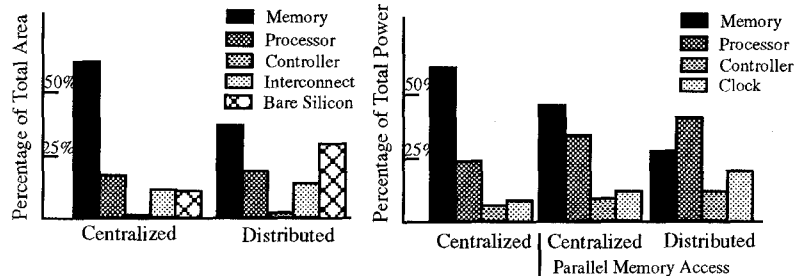


Figure 5: Area and Power Breakdown



Figure 6: Centralized Memory Chip