Implementation of an MPEG Codec on the Tilera™ 64 Processor

Whitney Flohr
Supervisor: Mark Franklin, Ed Richter

Department of Electrical and Systems Engineering
Washington University in St. Louis
Fall 2008

***Abstract*—**This project mapped an algorithm for MPEG video encoding to a mesh-connected, multi-core processor. The most promising mapping splits the work load evenly over the cores, with the routing overhead being very small. Because the computation time for each core was millions of times greater than the overhead generated by loading and shifting data through the mesh network, results with the algorithm show speed-ups that scaled directly with the number of cores on chip, regardless of movie size or the number of frames.

I. Introduction

MPEG is a standard for the compression and transmission of video and audio, with a wide array of qualities and sizes available. MPEG spans from HD-DVD technology, to the high compression rates necessary for online video sharing. This project explored the performance of implementing an MPEG codec on a tiled multi-core processor developed by Tilera™. The chip has 64 tiles, each with its own processor, memory, and switching controls. Unlike other multi-core processors that must route information for different cores through a centralized intersection, the Tilera chip links its cores with a mesh network[1]. This project sought to exploit the tiled architecture and achieve speed-ups over a single core processor by splitting the work over the tiles and efficiently routing information between tiles.

II. MPEG Background

The MPEG (Motion Pictures Experts Group) is a working group of the ISO/IEC that defines compression and decompression for video and audio [2]. The MPEG-1 standard was used to compress raw VHS quality video and CD audio to 1.5 Mb/s stream [3]. The MPEG-2 standard deals with broadcast-quality television, and supports interlacing and HD (high definition) video [2]. The MPEG-4 standard can achieve higher compression than MPEG-2 with more complex coding, such as global motion compensation and object tracking [3], most of which were not used in this implementation. MPEG-2 and MPEG-4 support a variety of bit rates and resolutions, a sampling of which are listed in Table 1.

Table 1
Video sizes common to the MPEG-2 and MPEG-4 standards

	Max Bit rate (Mb/s)	Max Hrz. Resolution (pixels)	Max Vert. Resolution (pixels)
Low Level	4	352	288
Mid Level	15	720	576
High-1440	60	1400	1152
High Level	80	1920	1152

Compression in MPEG is achieved by reducing bits made unnecessary by psycho-visual redundancy in pre-processing, spatial redundancy in encoding, and temporal redundancy in feedback operations [3]. A block diagram of the encoding and feedback processes is shown in Figure 1.

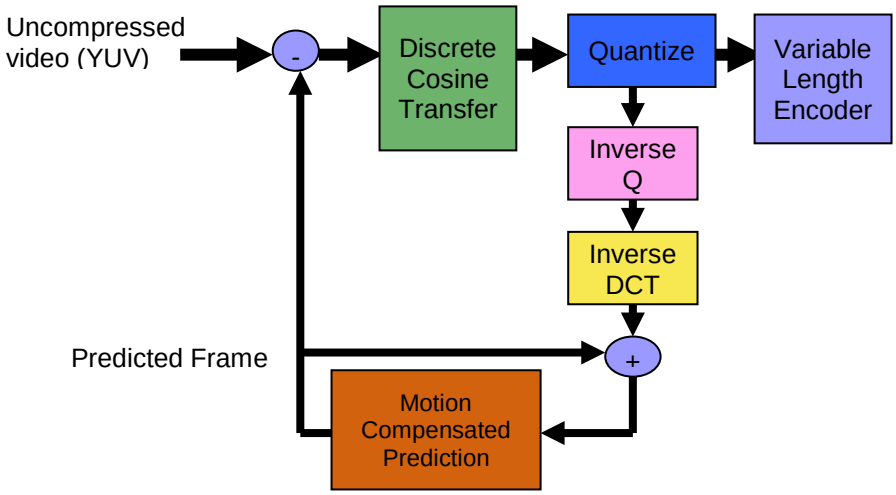

Fig.1. Block Diagram of MPEG Encoding

Psycho-visual redundancy refers to data between which the human eye cannot distinguish. In pre-processing the first step of compression transforms the data from the RGB (Red Green Blue) color space into YUV color space, which encodes a picture with the level of light (Y or luma component) and two color components (U and V or chroma). The luma is more important to visual perception, so each pixel's luma value is recorded. However, the chroma values can be compressed by sub-sampling at a 4:1 compression ratio without loss of quality.

Spatial redundancy refers to the similarity of a pixel to its neighbors. Each frame is encoded similarly to a JPEG still image using a DCT (discrete cosine transformation), which gives a DC term, and many other terms that are zero [2]. These zero terms aid the later stages of compression. The DCT algorithm is given by (1). This step typically requires the longest computation time, since computing each co-efficient requires the data from every pixel in the set.

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) cos \frac{(2x+1)u\pi}{2N} cos \frac{(2y+1)v\pi}{2N} \qquad C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} for\, u,v = 0 \\ 1\, otherwise \end{cases} \qquad (1)$$

The data enters a quantization phase, which is a lossy compression resulting in fewer bits storing the information. The data is fed into a VLC (variable length encoder) which traces through the pixels as shown in Figure 2, and groups them into words. Since many zero entries, represented by smaller dots in Figure 2, are clustered in the bottom right corner after the DCT and quantization stages, the string of zeros is coded as a single word to cut down on space.

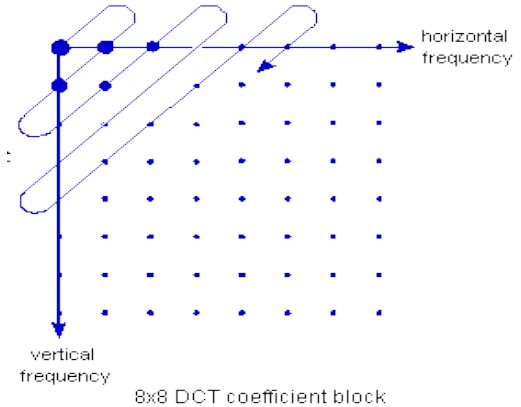

Fig. 2. Bit pattern of a variable length encoder

To take advantage of temporal redundancy, pixels are linked in time between frames. The MPEG standard defines three types of frames: I-Frames (intra-coded), P-frames (predicted) and B-frames (bidirectional predicted frames). I-frames have no additional compression, and represent a complete picture. P and B frames, which are created from previous P and I frames, encode only how the difference between their frame and some P or I reference frame. To create a P or B frame, the reference frame is recreated after quantization with an inverse quantization, and an inverse DCT, as shown in Figure 1. The result is then subtracted from the next incoming frame to determine how it is different from the reference. If the two frames are very similar, then much of the data entering after the feedback stage will be zero, resulting in significant compression in the VLC stage.

The MPEG bit stream is organized in the hierarchy shown in Figure 3. The lowest level of organization is the macroblock (MB). Each macroblock corresponds to a set of adjacent pixels in a frame. In MPEG-2, it is a standard size, and in MPEG-4, they can be dynamically sized. A 16x16 block will be assumed for this discussion. Each macroblock contains six 8x8 pixel matrices. Four store the luma values of the 16x16 block, the other two store sub-sampled chroma values requiring only one matrix each.

At the next level, macroblocks are grouped together in slices. A slice contains one or more adjacent macroblocks. A slice can be a single row of the picture, or multiple, adjacent rows. Slices make up each frame of the video. Frames are grouped together in GOPs (group of pictures) based on the type of pictures they contain. A new GOP is made for each I-frame, and contains the P and B frames that use it as a reference.

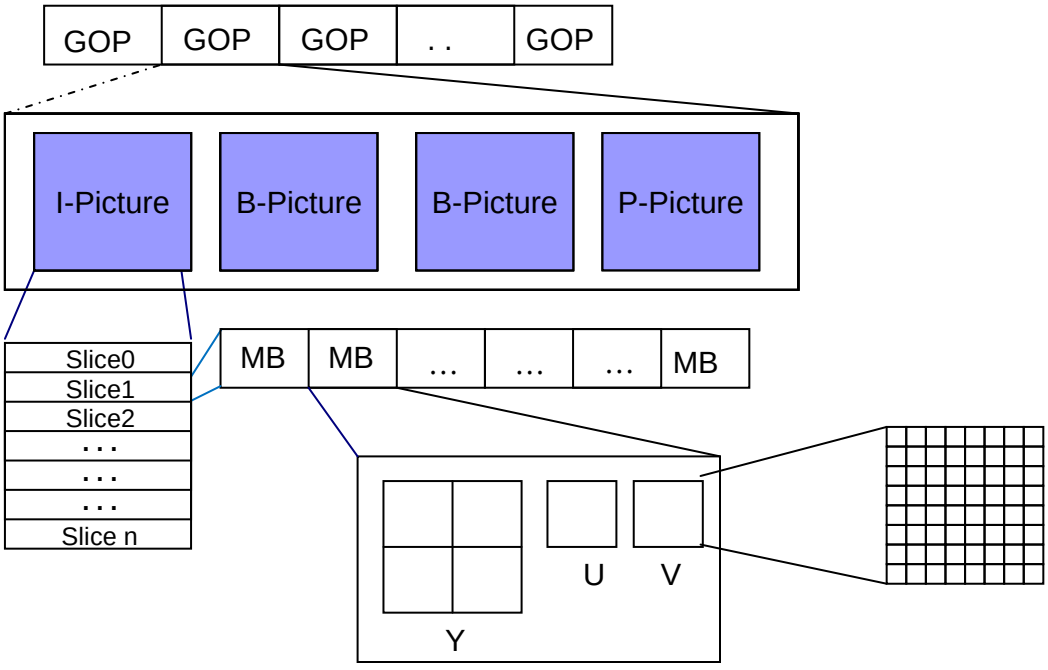

Fig.3 Storage Hierarchy for an MPEG file.

III. TILERA ARCHITECTURE

The Tilera chip connects 64 cores, each with its own processor, cache, and switching hardware, together in a square mesh [4]. In conventional multi-core architecture, information leaving and entering the chip had to pass through a single bus shared

by the different cores, causing an information bottle neck getting information on and off the chip. The mesh topology seeks to retain the speed gains of computing smaller pieces of a solution in parallel, while making loading data in and out of the processor more efficient.

Each tile processor has 72 kb in cache memory, 8kB in an L1 cache with 1 cycle latency, 64 kb in an L2 cache with 7 cycle latency[4]. The processors have a listed speed of 600 Hz to 1GHz.. The communications switch links each processor to its immediate neighbor, as shown in Figure 4. A tile can communicate with any other tile in the network by being routed through these switches with a delay of 1 clock cycle per tile move. Information can be routed simultaneously to and from tiles while the tiles are engaged in processing tasks. Any tile on the edge of the square network can be connected to memory or other I/O devices, so there are many more potential connections to reduce the information bottleneck for off-chip communications.

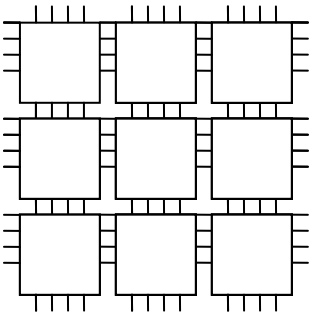

Fig. 4. Tile Interconnections

IV. Design Overview

The MPEG codec can be parallelized at any of the levels shown in the hierarchy in Figure 3. The most straightforward implementation is to divide the work evenly across each of the tiles, achieved most easily by dividing the macroblocks equally for each tile. This is equivalent to splitting each frame into equal slices, and giving one to each tile. The speed, ignoring routing costs, will scale with the number of tiles, as each tile has a smaller work-load.

The problem of how to route the data assumes only one side of the chip is connected to the source of the data, shown on the left side of a 4x4 version and 2X2 version of the chip in Figure 5.

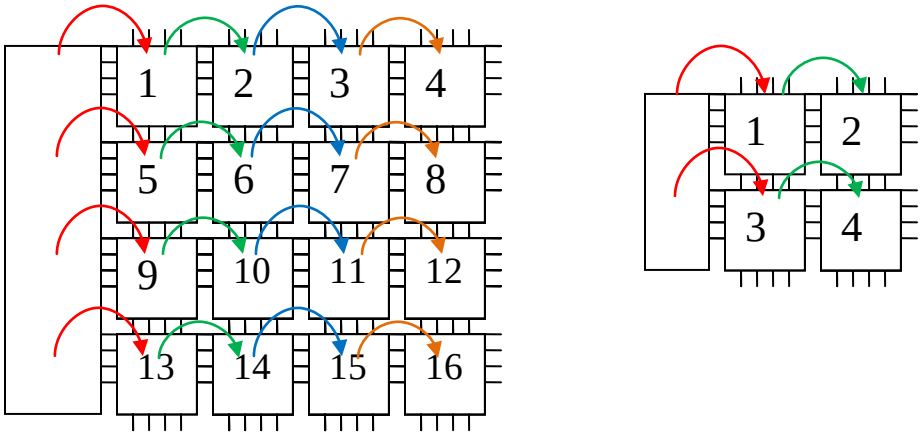

Fig. 5. Tile Layout and Routing for 16 and 4 Tile Chips

As shown in Figure 5, data can be routed directly from off chip to and from tiles 1, 5, 9, and 13, but only tile one can be written to or read from at a time. There is also a delay to move the uncompressed data to the other tiles for processing, and moving the compressed data back after processing is complete. Worst case delay in the 16 tile version is for tiles 4, 8, 12, and 16, which must go through 3 tile moves for the 16 tile version. For the four tile version, the worst case delay is for tiles 2 and 4, which must go through 1 tile move. For B tiles, worst case move delay will be $\sqrt{B}$ -1 moves. Parameters used in timing discussions are given in Table 2. If the parameter is static, a value is given assuming 2 bytes per uncompressed coefficient, and 15 bytes for an entire compressed matrix.

TABLE 2
Time Parameters and values

Symbol	Parameter	Value (if static)
Us	Size of an uncompressed macroblock	768 bytes
Cs	Size of a compressed macroblock	181 bytes
T1	Time to process one macroblock on one tile	175886 cycles
Intrachip Bandwidth	Time to move information between tiles	1 cycle/25 bytes [4]
Memory Bandwidth	Time to move information to edge tiles from off chip	1 cycle/250 bytes [4]
B	Number of tiles in chip	Variable
S	Size of frame in macroblocks	Variable
F	Frames in video being encoded	Variable

Load time, TL represents the time it takes for memory to be written into one of the edge tiles from off-chip, and is given by (2).

$$TL = \frac{S}{B} \times Us \times Memory\ Bandwidth \tag{2}$$

The time to move uncompressed data between tiles, Ts, is given for the worst case routing time. In the four tile case, the worst-case move is routing data from tile 1 to tile 2, or from tile 3 to tile 4. Ts(worst case) is given by (3).

$$Ts(wc) = Intrachip\ Bandwidth \times \frac{S}{B} \times Us \times \left(\sqrt{B} - 1\right) \tag{3}$$

"Block Processing" or Tb, is the time it takes a tile to complete the slice assigned to it for this frame after the appropriate data has been routed to it, and is given by (4).

$$Tb = [2(F-1)+1] \times \frac{S}{B} \times T1 \tag{4}$$

The time to move compressed data between tiles, Tr, is given for the worst case routing time. In the four tile case, the worst case routing is from tile 2 to tile 1, or tile 4 to tile 3. Tr is given by (5).

$$Tr(wc) = Intrachip\ Bandwidth \times \frac{S}{B} \times Cs \times \left(\sqrt{B} - 1\right) \tag{5}$$

The write time, Tw, is the time required to write compressed data off-chip from an edge tile, and is given by (6). As with the load time, it is assumed the off-chip source

$$Tw = \frac{S}{B} \times Cs \times Memory\ Bandwidth \tag{6}$$

The total time to compute a video is given by (7), and can be seen in the timing diagram for a four tile system, shown in Figure 6.

$$Total\ Time = \left(\sqrt{B} \times TL\right) + Ts(wc) + Tb + Tr(wc) + Tw \quad (7)$$

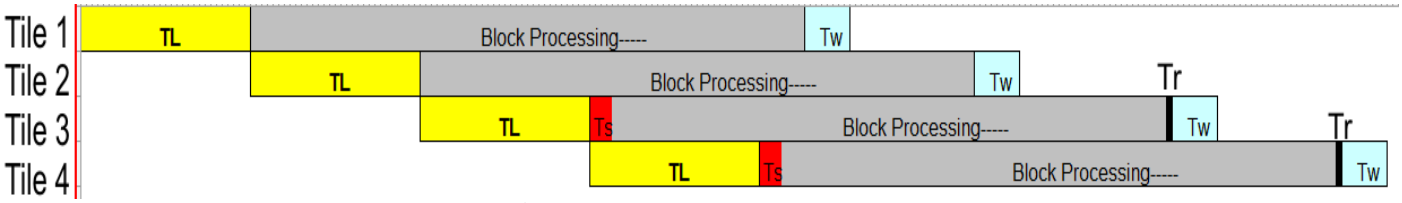

Fig. 6. Timing of a Four Tile System (block processing time not to scale).

V. Performance Analysis

At 32 frames a second, an hour long video is $1.2*10^5$ frames. Analysis was carried out to $2*10^5$ frames. Data tables are included in the appendix.

The time it takes to compute scales linearly with time and number of frames F, as shown in Figure 7. The Tb term scales linearly with frame number, and is by far the dominate term in (1). Each term in (1) also scales linearly by the number of blocks in the chip B.

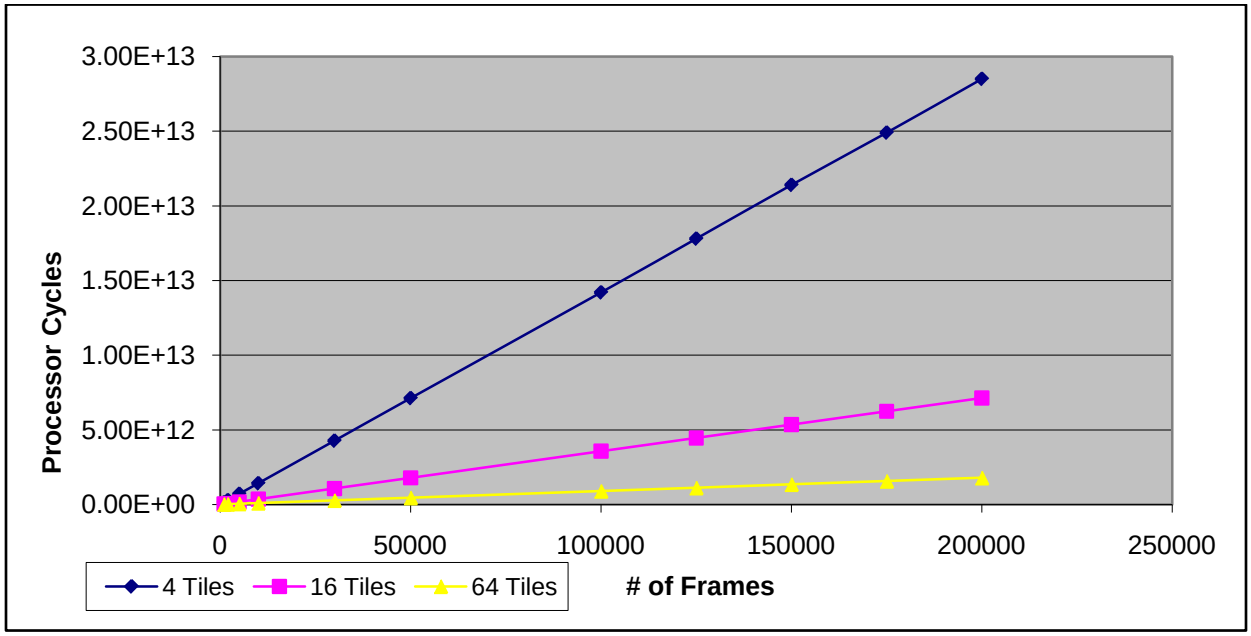

Fig. 7. The Effect of Tile Number and Frame Number on Encoding Speed in a Mid-level Video

The time to encode the video also scales with the size of the video, which determines the numbers of blocks each tile must operate on. When the number of tiles is held stable, as shown in Figure 8, the amount of encoding each tile must perform increases, and the Tb time scales linearly with the size of the video S. Since the Tb factor dominates the total computation time, the total time still scales linearly over the number of frames.

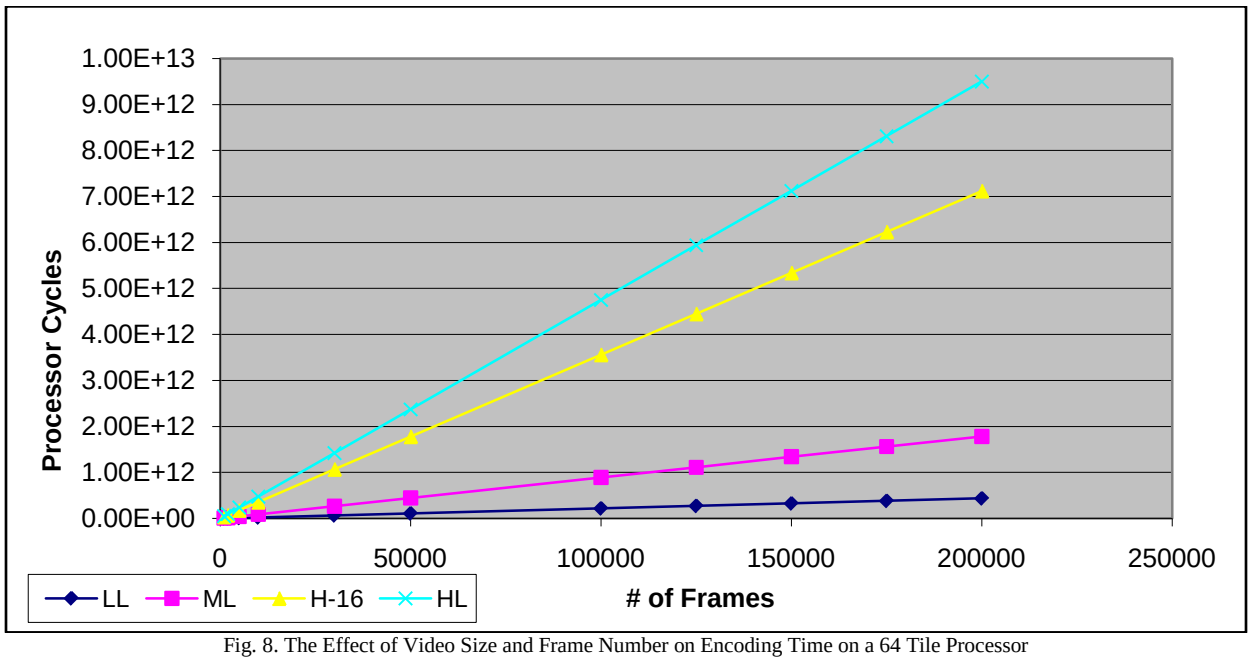

Fig. 8. The Effect of Video Size and Frame Number on Encoding Time on a 64 Tile Processor

When the encoding time is normalized for the % TL, as shown in Figure 9, there is little difference between 4, 16, and 64 tiles. TL is inversely related to the number of tiles. More data must be moved for fewer tiles, increasing the load time.

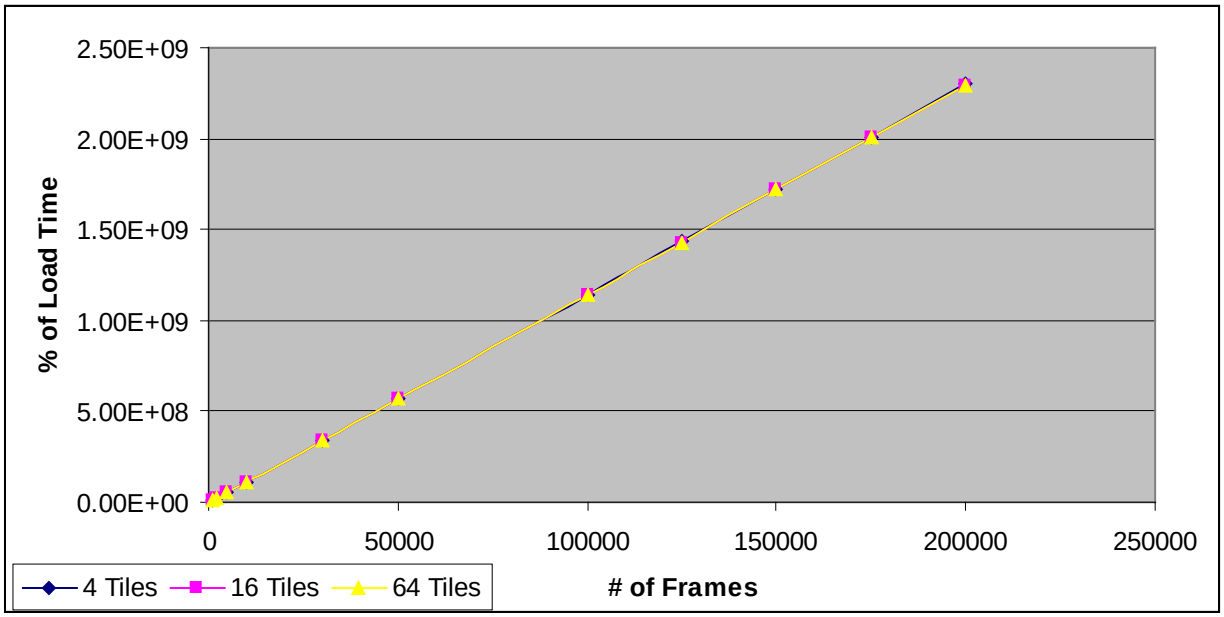

Fig. 9. The Effect of Tile Number and Frame Number on Encoding Time as % of Load Time on a Mid-level Video

As with the load time TL, the worst case shift time Ts is larger for small values of B, and smaller for big values of B. The effect of normalizing for Ts is not as profound of that of TL, because Ts does not scale perfectly with the amount of data being moved. Ts also must move the data across increasing distances for increasing B. The encoding time, as a percent of the worst case shift time, is shown in Figure 10.

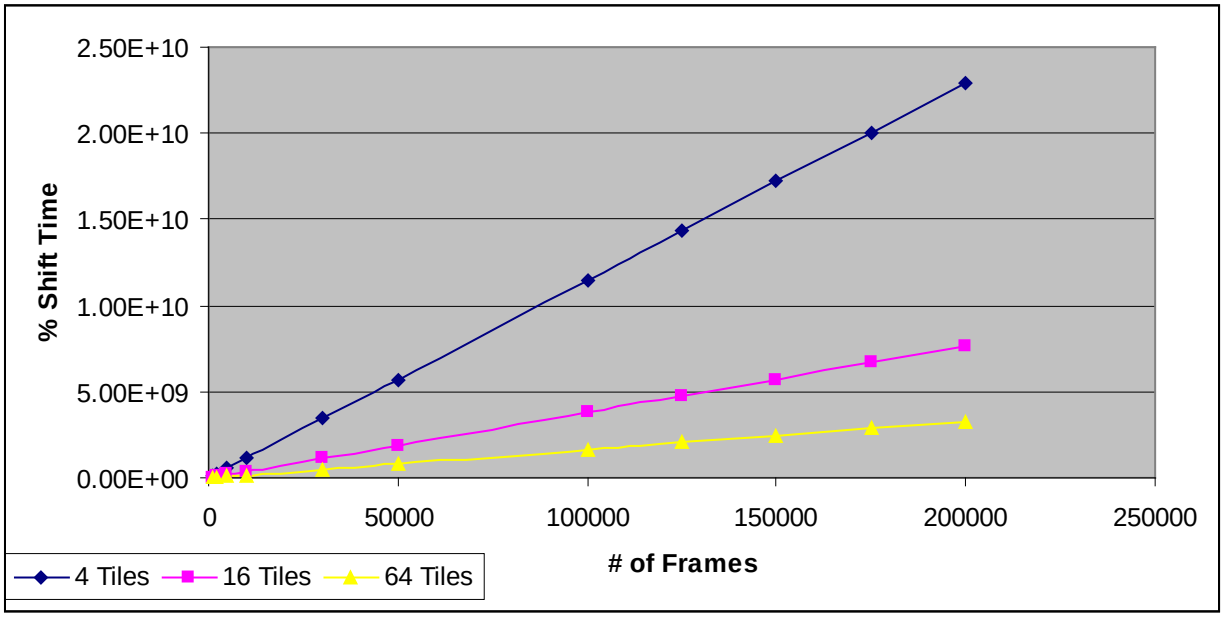

Fig. 10. The Effect of Tile Number and Frame Number on Encoding Time as % of Shift Time on a Mid-level Video

Figure 11 shows the effects of normalizing for Ts + TL. It shows a mix between the moderate clustering for that of Ts, and the absolute normalizing effect of TL.

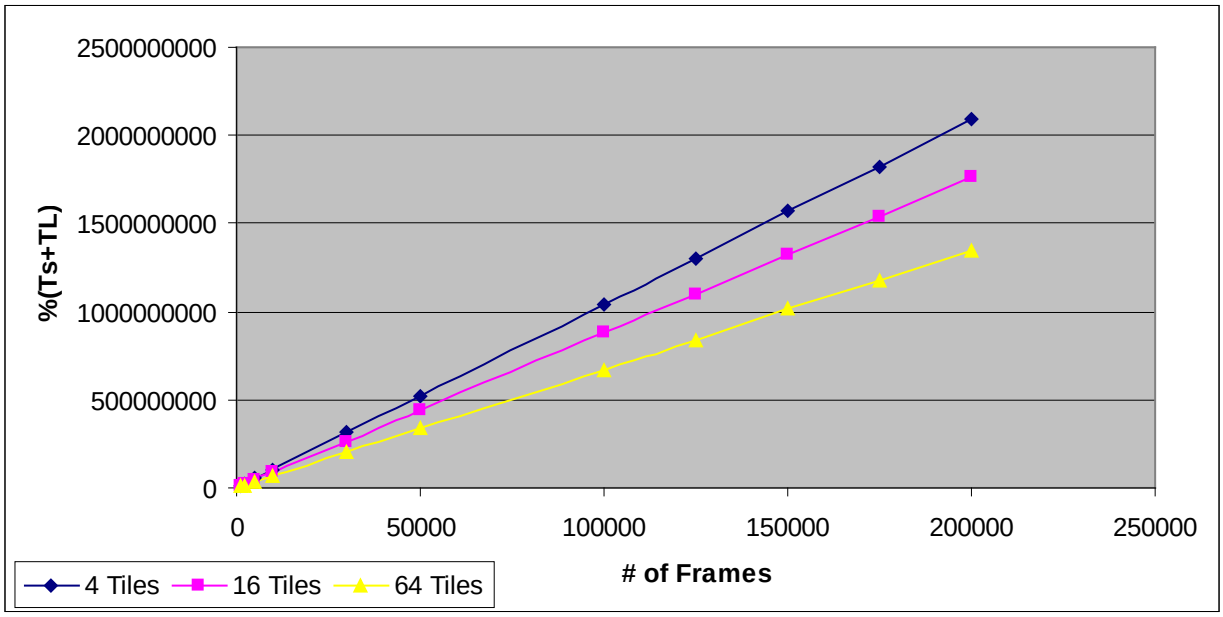

Fig. 11. The Effect of Tile Number and Frame Number on Encoding Time as % of (TL+Ts) on a Mid-level Video

To add baseline performance data to compute the speed-up of this algorithm, a theoretical running on a one tile system is made, assuming that a single tile could hold all the necessary data to run in this algorithm. The speed-ups do not vary significantly for the number of frames over the examined range of 1000-200,000 frames, so only one value is given. Table 3 shows speed-ups in encoding time that scale with the number of processors.

TABLE 3
Speed Up for B Tiles vs. Theoretical 1 Tile Computation

Tiles	Speed Up (t1/tB)
4	3.986
16	15.989
64	63.992

VI. DISCUSSION/CONCLUSIONS

The model that splits the work among the tiles equally scales linearly with the number of tiles over different video sizes and lengths. Even at the large end of the MPEG standard, and for 100 min videos, the Tb term dominates the timing and preserves the linear behavior of the model. The model assumes enough memory on-chip to store an uncompressed and compressed slice, no matter the size, when there are only 72 KB available [4]. This is enough to store all implementations listed in this paper, but makes the model unrealistic to apply to a single tile case, as was used in finding the speed-up results. The model also assumes memory that switching and routing are never interfered with. While the Tilera has 5 independent networks [4], routing may overlap on chips with a large number of tiles, causing speed losses that this model does not account for.

While the Tb timing was the dominant factor in this model, a more realistic look at moving data across the tiles may find that switching time for a macroblock becomes comparable to the encoding time for a macroblock. In this case, the algorithm may be better served by choosing different sized slices, and assigning ones with more macroblocks closer to the chips edge, and giving slices with fewer blocks to tiles with long routing paths.

ACKNOWLEDGMENT

W. Flohr thanks Ed Richter and Mark Franklin for their time and help on this project.

APPENDIX: *DATA TABLES*

TABLE A1.
The Effect of Tile Number and Frame Number on Encoding Speed in a Mid-level Video
Figure 7 Data

Frames	4 Tiles (cycles)	16 Tiles (cycles)	64 Tiles (cycles)
1000	1.42E+11	3.56E+10	8.90E+09
2000	2.85E+11	7.12E+10	1.78E+10
5000	7.12E+11	1.78E+11	4.45E+10
10000	1.42E+12	3.56E+11	8.90E+10
30000	4.27E+12	1.07E+12	2.67E+11
50000	7.12E+12	1.78E+12	4.45E+11
100000	1.42E+13	3.56E+12	8.90E+11
125000	1.78E+13	4.45E+12	1.11E+12
150000	2.14E+13	5.34E+12	1.34E+12
175000	2.49E+13	6.23E+12	1.56E+12
200000	2.85E+13	7.12E+12	1.78E+12

TABLE A2.
The Effect of Video Size and Frame Number on Encoding Time on a 64 Tile Processor

Figure 8 Data

Frames	LL (cycles)	ML (cycles)	H-16 (cycles)	HL (cycles)
1000	2.18E+09	8.90E+09	3.56E+10	4.75E+10
2000	4.35E+09	1.78E+10	7.12E+10	9.50E+10
5000	1.09E+10	4.45E+10	1.78E+11	2.37E+11
10000	2.18E+10	8.90E+10	3.56E+11	4.75E+11
30000	6.53E+10	2.67E+11	1.07E+12	1.42E+12
50000	1.09E+11	4.45E+11	1.78E+12	2.37E+12
100000	2.18E+11	8.90E+11	3.56E+12	4.75E+12
125000	2.72E+11	1.11E+12	4.45E+12	5.94E+12
150000	3.26E+11	1.34E+12	5.34E+12	7.12E+12
175000	3.81E+11	1.56E+12	6.23E+12	8.31E+12
200000	4.35E+11	1.78E+12	7.12E+12	9.50E+12

TABLE A3.
The Effect of Tile Number and Frame Number on Encoding as a % of Load Time on a Mid-Level Video

Figure 9 Data

Frames	4 Tiles (%TL)	16 Tiles (%TL)	64 Tiles (%TL)
1000	1.15E+07	1.14E+07	1.14E+07
2000	2.30E+07	2.29E+07	2.29E+07
5000	5.74E+07	5.72E+07	5.72E+07
10000	1.15E+08	1.14E+08	1.14E+08
30000	3.44E+08	3.44E+08	3.43E+08
50000	5.74E+08	5.72E+08	5.72E+08
100000	1.15E+09	1.14E+09	1.14E+09
125000	1.44E+09	1.43E+09	1.43E+09
150000	1.73E+09	1.72E+09	1.72E+09
175000	2.01E+09	2.00E+09	2.01E+09
200000	2.30E+09	2.29E+09	2.29E+09

TABLE A4:
The Effect of Tile Number and Frame Number on Encoding Time as % of Shift Time on a Mid-level Video

Figure 10 Data

Frames	4 Tiles (%Ts)	16 Tiles (%Ts)	64 Tiles (%Ts)
1000	1.14E+08	3.82E+07	1.64E+07
2000	2.29E+08	7.63E+07	3.27E+07
5000	5.72E+08	1.91E+08	8.18E+07
10000	1.14E+09	3.82E+08	1.64E+08
30000	3.43E+09	1.15E+09	4.91E+08
50000	5.72E+09	1.91E+09	8.18E+08
100000	1.14E+10	3.82E+09	1.64E+09
125000	1.43E+10	4.77E+09	2.04E+09
150000	1.72E+10	5.72E+09	2.46E+09
175000	2.00E+10	6.68E+09	2.87E+09
200000	2.29E+10	7.63E+09	3.27E+09

TABLE A5
The Effect of Tile Number and Frame Number on Encoding Time as % of (TL+Ts) on a Mid-level Video

Figure 11 Data

Frames	4 Tiles (%Ts + TL)	16 Tiles (%Ts + TL)	64 Tiles (%Ts + TL)
1000	10407383	8805343	6730595
2000	20888058	17610685	13461189
5000	52183498	44026713	33652974
10000	1.04E+08	88053426	67305947
30000	3.13E+08	2.65E+08	2.02E+08
50000	5.22E+08	4.4E+08	3.37E+08
100000	1.04E+09	8.81E+08	6.73E+08
125000	1.3E+09	1.1E+09	8.39E+08
150000	1.57E+09	1.32E+09	1.01E+09
175000	1.82E+09	1.54E+09	1.18E+09
200000	2.09E+09	1.76E+09	1.35E+09

REFERENCES

[1] Tilera Corporation. “Tile Processor Architecture- Technology Brief.” Available: http://www.tilera.com/pdf/ProductBrief_TileArchitecture_Web_v4.pdf, 2008.

[2] MPEG working group. “Coding of Motion Pictures and Audio.” Available: http://www.chiariglione.org/mpeg/, 2008.

[3] Watkinson, John. (2001). *The MPEG Handbook (pp. 4)*. Woburn, MA: Focal Press.

[4] Agarwal, Anant. “The Tile Processor: A 64-Core Multicore for Embedded Processing” Available: http://www.tilera.com ,2007.