Reprint

Real-Time Disparity Information Compression in 3D Teleconferencing Systems

D. Papadimatos, T. Antonakopoulos and V. Makios

The 24th EUROMICRO Conference

VASTERAS, SWEDEN, AUGUST 1998

REAL-TIME DISPARITY INFORMATION COMPRESSION IN 3D TELECONFERENCING SYSTEMS

D. Papadimatos, T. Antonakopoulos, V. Makios
Department of Electrical Engineering and Computers Technology,
University of Patras, 26500 Patras
GREECE

Abstract

This paper presents a real-time entropy compression/decompression unit for disparity map information used in 3D teleconferencing systems. The disparity map data form a constant bit rate data stream which has to be transmitted through an ATM channel supporting lower data rates. The selection of the proper compression algorithm must be based on the durability of the regenerated data to various types of errors, generated mainly due to the limited available bandwidth. Initially we present the disparity map formats, why they should be coded losslessly, some well-known entropy algorithms and their performance in terms of compression rate and throughput. All algorithms are evaluated according to the application requirements which are: good compression rate, real-time implementability and fast convergence during the compression initialization phase. Feasible implementations are proposed for the algorithms selected using commercially available digital signal processors (DSPs) and field programmable gate arrays (FPGAs).

1. Introduction

The new generation of 3D telepresence videoconferencing systems aim to provide the user with an enhanced illusion of true contact. Various methods on achieving this with the use of stereoscopic cameras have highlighted the need for disparity estimation and image interpolation. In this work, we consider the case where disparity estimation is carried out on the transmitter side and interpolation is carried out on the receiver side. Thus for 3D image recreation, the transmission procedure becomes more complex, since a disparity stream must also be transmitted with the image data streams, but the most complex part of the system, the receiver, is greatly simplified. With the transmission of coded disparity map information with compressed video camera data, remote teleconferencing applications can be made feasible with reduced data rates. An example of such a system is currently being developed in the framework of the ACTS project 092 PANORAMA[1] [1] and is shown in Fig. 1.

In real-time applications like video and audio services, lossy algorithms have been chosen due to their superior compression rate. Such algorithms control their output bitrate by increasing or decreasing the compression rate. They take advantage of the redundancy that exists in video and audio raw data and of various characteristics of the human vision (video) or hearing (audio) to decrease the amount of information that needs to be transmitted. Unfortunately in the case of disparity map coding these algorithms cannot be used since such lossy coding of the disparity map would lead to large synthesis errors and even an incorrect three-dimensional impression [8]. For our application, we will consider redundancy reduction with the application of entropy coding algorithms. They achieve a smaller compression rate but the original stream can be recreated without errors. Unfortunately such algorithms have the disadvantage that the bitrate produced varies according to the compressibility of the input data stream. When this bitrate exceeds the capacity of the communications channel lossless coding is not possible. In the absence of extensive studies on the effect of disparity map errors on image quality, we chose a simple method of controlled data loss (CDL) during decreased compression efficiency. In CDL, the compression ratio is evaluated on a frame-to-frame basis. The output channel is divided into fixed-sized slots. Each compressed frame is allocated a time slot. If a compressed frame size is larger than its allocated slot, transmission continues into the next slot while the following frame is not transmitted. At the decoder side, a rejected frame is replaced by the previously transmitted frame.

[1] This work has been performed in the framework of the European Commission ACTS Project AC092 PANORAMA -"Package for New Operational Autostereoscopic Multiview Systems and Applications"

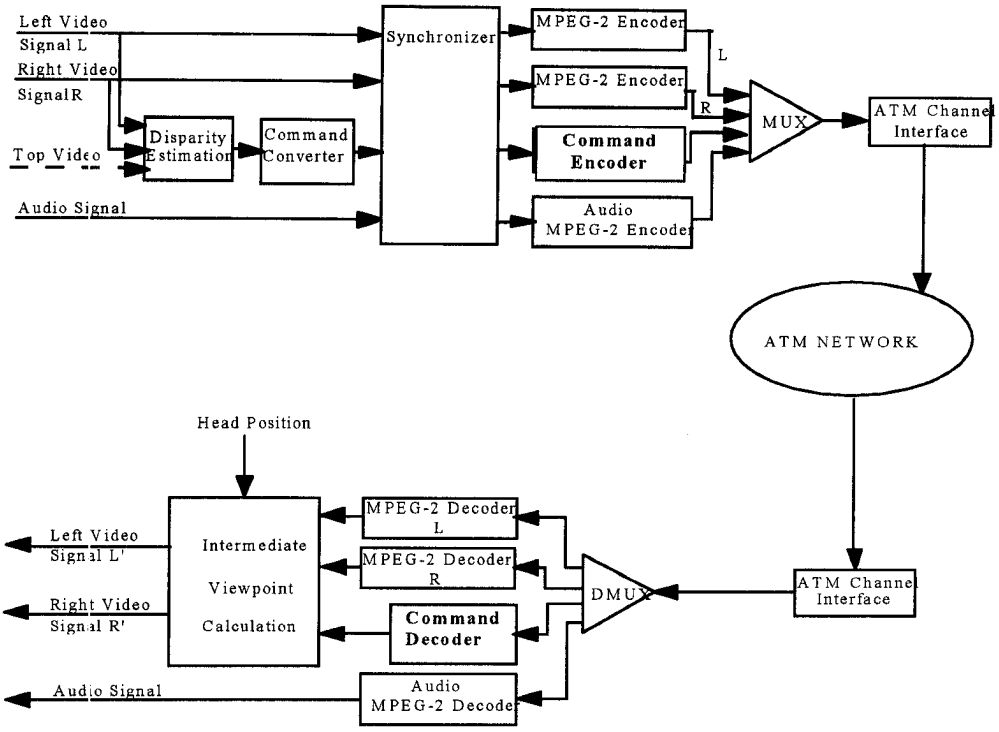

Figure 1. The 3D transmission hardware chain

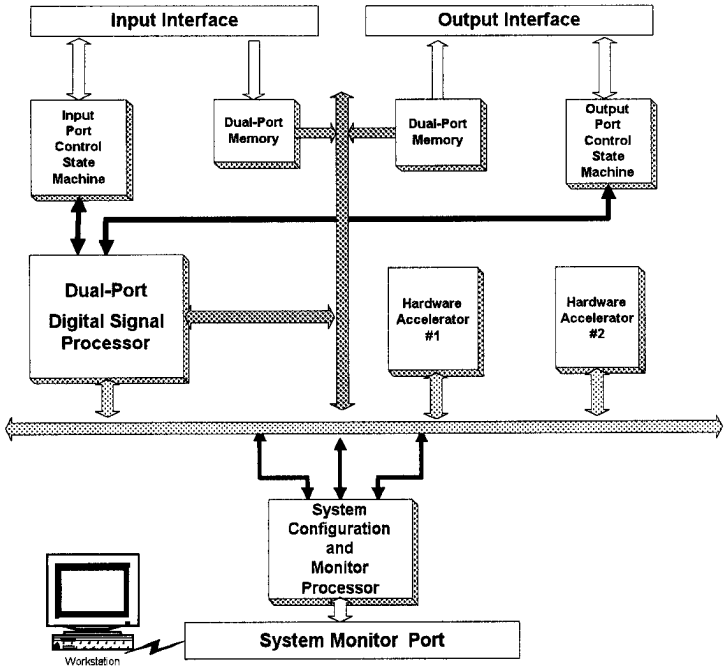

Figure 2. The Disparity Map compression/decompression engine architecture

In this paper we present the architecture and functionality of a lossless compression unit for disparity data and its decompression unit that is a part of the transmission chain of a 3D teleconferencing system. First we present a few characteristics of the data which must be coded; in our case the disparity map information from the disparity estimator unit. Secondly we evaluate the performance of various lossless compression algorithms in terms of compression ratio and complexity to determine their feasibility in terms of implementation with commercially available digital signal processors and existing re-programmable logic technologies. Both ad-hoc techniques and commercially available solutions were investigated. We also examine the performance of the various compression algorithms when CDL is used.

2. System Overview

The main aim of the lossless compression unit is to reduce the bitrate required to transmit the disparity map data through an ATM network. The disparity data are encapsulated in a CCIR recommendation 601/656 format with a total rate equal to 27 Mbytes/sec but the raw data rate is 5.184 Mbps, as it is explained in the next paragraph. The CBR channel allocated for the disparity data has a fixed capacity of between 2 Mbps and 4 Mbps, which is determined during the call set-up. Lossless compression is used due to the nature of the data while the introduced delay per frame will also be constant. Thus, if the corresponding video codecs also introduce constant delay, synchronization is possible at the ATM multiplexer. The decompression unit accepts the coded disparity bitstream and regenerates the original disparity map bitstream. The intermediate view calculation unit uses as input two video channels and the disparity map information. It contains a synchronization unit which synchronizes the three channels. The timing difference must be smaller than one frame. In order to keep synchronization the introduced delay must also be kept constant. The disparity map is generated by the disparity estimator which worked on two video bit-streams [5]. It produces two unidirectional vector maps for each video frame (left-to-right image disparity map and right-to-left disparity map).

In the PANORAMA project the disparity vectors are transmitted in the form of a chain map [5], [4]. The disparity estimator generates two unidirectional vector fields. The vector fields are sparse fields i.e. are subsampled four times in the vertical direction. Disparity information derived from CCIR-601 video was transmitted every fourth scanline. The transformation is described in [4] but in general, when teleconferencing scenes are involved, vectors from both maps are used according to their position. Vectors from the right-to-left map are used for the left part of the image while vectors from the left-to-right map are used for the right part.

Table 1. Entropy of various typical teleconferencing sequences (bits/symbol)

	MAN	MAN 4 times subsampled	WOMAN
8-bit symbols	5.478	5.404	5.239
16-bit symbols	8.668	8.596	8.2553
24-bit symbols	11.0216	10.7164	10.4536

For other scenes one of the two disparity vector maps are used. Thus for CCIR-601 video the resulting data rate of the chain map is 5.184 Mbits/sec.

Due to the bi-directional consistency check, the two maps are highly redundant so the chain map can transfer most of the information contained in both disparity vector maps [4]. The chain map has low inherent redundancy [5] but the original disparity vector maps have some redundancy due to the limited disparity range and filtering in the final stages of processing [4].

Entropy coders in general do not have a absolutely defined compression rate. It is not even necessary for them to be able to compress all types of data. Their target is to reduce the redundancy inherent in most bitstreams. An approximate metric of this redundancy is given by the entropy of the input bitstreams. So in order to gain an insight in the amount of compression feasible we first analyzed test sequences in terms of their entropy. The entropy is calculated using the following equation:

$$H = -\sum p(symbol)\log_2[p(symbol)]$$

The entropy gives the theoretical minimum number of bits required to code the input data stream. The entropy of the chain maps of various typical teleconferencing scenes were measured and results are shown in Table 1. We note that the longer the symbols are, the fewer bits are needed to code them. So compression algorithms which can exploit intersymbol redundancy should fare better than those that utilize lone symbol probability distribution alone.

3. Lossless Compression Algorithm Evaluation

3.1. Algorithm selection

The algorithm selection process involved the evaluation of the following parameters:

- Achieved compression rate (coding efficiency)
- Algorithm complexity.
- Latency

We decided to evaluate the performance on the three categories of lossless compression algorithms: Huffman coding, Arithmetic coding and Lempel-Ziv coding. The evaluation was mainly based on the first two parameters. The input bitstream was divided into constant sized blocks. These blocks contain disparity data which correspond to a frame of video data and are called disparity map frames. The size of these frames was such that the above algorithms were able to adapt and achieve reasonable compression rate. On the other hand by controlling how often these algorithms reset their adaptation mechanisms the introduced latency could be bounded and recovery from transmission errors could be carried out on the reset boundaries.

The test disparity maps were generated under realistic teleconferencing conditions using corresponding disparity estimator software. Candidates from each category were tested with the simulated disparity map sequences to evaluate achieved compression rate.

The following candidates were evaluated:

- Static and Adaptive Huffman algorithms [3],[6]
- Arithmetic Codes [2]
- Algorithms belonging to LZ77 group [9],[6].
- Algorithms belonging to LZ78 group [10].

Huffman algorithms achieve compression by assigning short codewords to input symbols which are more probable. This implies that there must be a probability model of the input source which drives the Huffman encoder. In the case of the static Huffman algorithm, the symbol probabilities of fixed sized blocks of input data are extracted and then these are used to encode the data. The size of the blocks is equal to a frame of disparity map data so that decoding could proceed on frame boundaries. The compressed bitstream is composed of the encoding tables and the encoded disparity map bitstream. On the other hand the adaptive Huffman algorithm doesn't transmit the encoding tables. These tables are modified in tandem by both the compression and decompression algorithms so each has the same copy. Transmission of the encoding tables is avoided at the expense of constant probability table maintenance which reduces throughput.

Arithmetic codes are more efficient than Huffman codes in terms of compression ratio but few implementations achieved high data rates. To achieve good compression rate, the arithmetic coders are driven by sophisticated probability models. These models make predictions based on the previous context, i.e. previous byte sequences. Unfortunately every possible context had to be kept in memory restraining context length to three bytes. Adaptation is also a problem. The performance penalty due to this housekeeping became quite obvious during the algorithm evaluation. In [2] two different methods are described: the prediction by partial matching (PPM) algorithm and Dynamic Markov Coding (DMC). In our evaluation we used a variant of the first optimized for DSP operation.

Lempel-Ziv compression schemes belong to a class of algorithms which view the input stream as a concatenation of fixed or variable length strings of symbols. A mapping procedure assigns unique codes to sequences of these symbols which are found in the input stream. These algorithms achieve compression by substituting sequences which have previously appeared at the input with pointers to these occurrences. The data objects used to store the mapping between the previously occurring sequences and their corresponding codes are called dictionaries. The management of the dictionary structure differentiate the various existing algorithms. Available Lempel-Ziv algorithms generally belong to two groups corresponding to the algorithms described in [9] and [10]. In our evaluation we chose a representative from the LZ-77 group and two from the LZ-78 group. Generally algorithms from the first group implement sliding window dictionaries so they can be considered locally adaptive. This allows fast adaptation of the algorithm to the input bit-stream. On the other hand, the second group algorithms adapt more gradually but are able to adapt their dictionary to the whole bit-stream. A summary of the adaptation characteristics of the evaluated algorithms is given in Table 2.

3.2. Evaluation Platform

In order to carry out evaluation under conditions which resemble real-time operation a system architecture based on a fast DSP and re-programmable logic was used. The architecture, which is shown in Fig. 2, is composed of four basic modules: the input state machine, the DP-DSP with hardware accelerators, the output state machine and system configuration.

The input and output state machine take care of the interface functions with all other modules. In compression mode, the input interface filter the disparity map data from the CCIR 601/656 format, while in decompression mode it accepts compressed data and passes them to the DSP for decompression. During evaluation of compression algorithms, the input interface was connected to a CCIR 601/656 test vector generator which stored disparity map data. The DP-DSP with the accompanying hardware accelerators implement the various compression algorithms.

Table 2. Adaptation characteristics of evaluated compression algorithms

Algorithm name	Adaptation Heuristic
Huffman	Two pass per frame. One to build symbol probability model and second to encode data
Adaptive Huffman	Probability table modified on the fly. Probability tables not transmitted
Arithmetic Ord. 0	Each byte coded independently of context
Arithmetic Ord. 1	1 byte context
Arithmetic Ord. 2	2 byte context
Arithmetic Ord. 3	3 byte context
LZ Sliding Dictionary	4096 entry sliding window dictionary
LZ Dynamic Dictionary	Fixed Codeword length = 12 bits. 4096 entry dictionary. No adaptation possible when the dictionary is full
LZ Dynamic Dictionary	Variable Codeword length = 9 - 15 bits. 32 K entry dictionary

The DSP used was the Motorola 96002 working at 40MHz. The processor was augmented with 128 Kbytes SRAM. Reprogrammable logic was used to implement various functions which could not implemented as software on the DSP. The monitor processor was connected to a PC which carried out statistics gathering functions such as compression rate and throughput unobtrusively.

3.3. Evaluation Results

We evaluated existing algorithms which were written in C code by porting them to the DSP. At first only some general optimizations were carried out on the algorithms to take advantage of various features of the processor such as parallel loading, floating point functions etc. Some initial compression results for the MAN sequence are shown in Table 3. The compression ratio expresses the reduction in size of disparity map sequences. Compression and decompression data throughput are measured relatively to the rate achieved by the basic Huffman coding algorithm. On the DSP an average rate of 2.5 Mbps was achieved.

No algorithm could attain the required throughput using its software implementation. Even with the fast DSP, the algorithm complexity cannot support the disparity data rates. Major optimizations on the most promising algorithms had to be carried out to achieve the throughput requirement. The Huffman algorithms had high throughput but compression efficiency was very low. On the other hand the arithmetic codes achieved the best compression efficiency but throughput was an order of magnitude lower even though the code was optimized for the DSP.

These results show that the LZ class combined reasonable compression rates with high throughput. The difference between compression ratio between the arithmetic coding routines and the LZ routines did not justify optimizing the arithmetic coding routines for speed. Since the compression efficiency of arithmetic codes was not much greater while computing resources required were of an order of magnitude larger, we decided to continue with the LZ group. Compression rates were slightly lower than those achieved by the LZ78 algorithms we tested.

Thus we undertook a complexity study of two LZ candidates to identity the blocks which needed to be adapted so that the system specifications could be met and how these adaptations affected the algorithms characteristics especially compression ratio.

The first candidate belonged to the LZ78 group which we called L78/DSP. It was a dictionary-based scheme where matching byte sequences were encoded and transmitted. A hash index was used to search the dictionary for the longest match. The code was rewritten in DSP96002 code and a four-fold improvement was seen in throughput without change in compression ratio. The throughput and compression ratio variation between sequential dictionary resets is plotted for the MAN sequence in Fig. 3. It is noted that in the encoder the dictionary search method incurs the greatest overhead. Even so we noted that the compression ratio was not affected by the resetting of the dictionary. This implies that with these sequences temporal redundancies are concentrated only between adjacent frames. In order to increase throughput the size of the dictionary could be reduced so that the hash collisions are lessened. Hardware implementations of the LZW algorithm [7] use this method to increase throughput.

This could be used to identify the number of frames required to get a good compromise between achieved compression ratio and propagation of errors between compressed blocks. These algorithms do not explicitly transmit their dictionaries. The decoding algorithm builds it up from the decoded bitstream. Unfortunately this requires that both

Table 3. MAN disparity map sequence compression results

	Comp. Ratio (%)	Comp. Rate 1=2.5 Mbps	Decom. Rate
Huffman	32	1	1.09
Adaptive Huffman	32.2	0.53	0.63
Arithmetic Ord. 0	34.2	0.21	0.1
Arithmetic Ord. 1	56.8	0.07	0.06
Arithmetic Ord. 2	58.1	0.09	0.1
Arithmetic Ord. 3	59	0.1	0.1
LZ Sliding Dictionary	50.8	0.15	1.74
LZ Dynamic Dictionary Fixed Codeword length = 12 bits	45.2	1.17	1.45
LZ Dynamic Dictionary Variable Codeword length 9 - 15 bits	52.6	1	1.45

Table 4. WOMAN disparity map sequence compression results

	Compr. Ratio (%)
Huffman	34.9
Adaptive Huffman	35.1
Arithmetic Ord. 0	36
Arithmetic Ord. 1	61
Arithmetic Ord. 2	64.8
Arithmetic Ord. 3	68.9
LZ Sliding Dictionary	56
LZ Dynamic Dictionary Fixed Codeword length = 12 bits	44
LZ Dynamic Dictionary Variable Codeword length 9 - 15 bits	62

dictionaries are built synchronously and errors corrupt the decoding of subsequent symbols. By resetting the dictionary at constant intervals the indefinite propagation of the error is avoided. Also during decreased compression efficiency some blocks can be selectively discarded without affecting the decoding of following blocks. The effect of the frequency of dictionary resets on the achieved compression ratio is shown in Fig. 4. These results indicate that most redundancy exists in the spatial neighborhood of each frame. The best results were attained when the reset interval didn't exceed two disparity frames.

Due to the nature of the building of the dictionary complex heuristics were needed to implement adaptation once it was filled. Various exist in the literature even though they improve compression by a little but they don't lend themselves to easy implementations. Simply discarding the previous dictionary and starting a new gave comparable compression results.

On the other hand LZ77 algorithms by virtue of their "sliding window" dictionary achieved much better adaptation features to locally concentrated redundancy. The dictionary used a least recently used (LRU) update heuristic where the oldest symbol was removed to make space for the newest symbol. The second algorithm we tested belonged to this group. Compression ratio was reasonable and decompression was very fast but compression throughput was very low. This was due to the search method employed. Typically if this search algorithm is implemented in software then if n symbols are contained in the dictionary then the time needed to search the dictionary is $O(n)$ [2]. In order to find the optimum dictionary size we carried out various tests on the available sequences. Compression rate results are plotted in Fig. 5. We found that optimum compression could be achieved with a very small dictionary. This dictionary only needed to retain data which corresponded to one or two previous video scanlines. Thus by constraining the dictionary the encoder processing rate improved considerably. The required decompression rate was easily attainable with a simple software implementation on the DSP.

The decompression unit could be implemented with a software implementation of the LZ77 algorithm on the DSP. Still the required compression throughput was not attained with the use of software, so we decided to explore the possibility of using a hardware accelerator to implement the compression algorithm. This work is described in the next section.

4. Implementation of the LZFastSearch encoding algorithm

The algorithm belongs to the LZ77 group of algorithms. As such it substitutes sequences which have appeared previously with a pointer to the corresponding position and length of the matching sequence. These are transmitted when they need fewer bits to transmit than the bytes they replace. A sliding window heuristic is used to update the

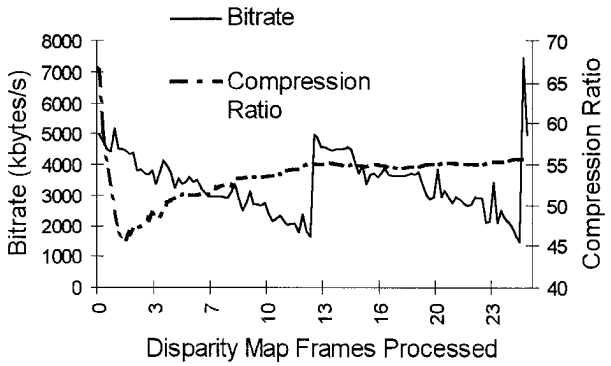

Figure 3. Variation of throughput and compression ratio on MAN sequence

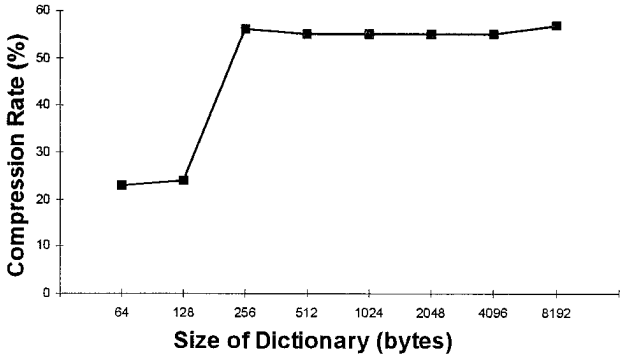

Figure 5. Compression performance of LZ77 algorithm on WOMAN sequence

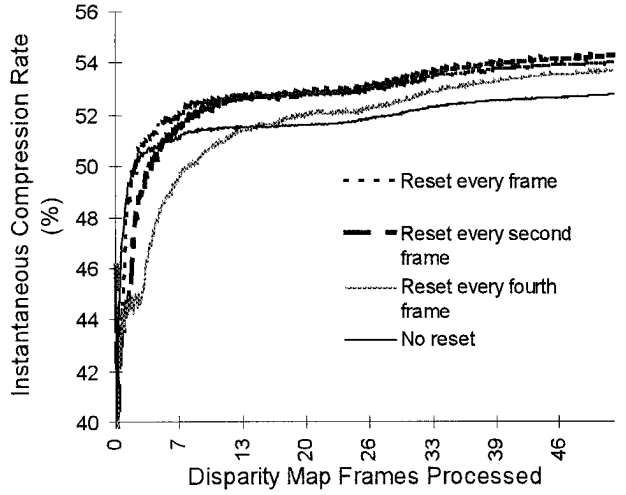

Figure 4. Compression performance of LZ78/DSP algorithm for MAN sequence using different dictionary reset periods

dictionary.

Due to the redundancy characteristics of the input bitstream, only data corresponding to the previous one or two scanlines need to be searched. Thus the dictionary can be implemented as a digital delay line. Each element of the delay line is composed of the comparison and carry-over cells. The carry-over cells are connected to a priority encoder which gives the relative position of the match. The length counter gives the length of the matching sequence.The comparison cells compare the present input with their stored value. The results are transferred to the carry-over cell. At the end of the comparison the stored value is shifted to a neighboring comparison cell. The width of the input ports of the comparison cell can be variable in width although we chose eight bit width for compatibility with the input bitstream. Also choosing wider widths reduced the number of carry-over cells required for a constant delay line size.

The carry-over cells keep information from the previous comparison and modify the result of the comparison cell before transferring it to the priority encoder. These cells "remember" previously occurring sequences. Their ability to "remember" depends on existence of matches. Each time an attempt is made to encode the longest possible sequence. If a match doesn't occur then the resulting code on the longest match found is output, the carry-over cell is cleared and a new cycle is begun.

The processing of each input byte is divided into four phases:

- Comparison of input with stored values in comparison cells
- Match evaluation with previous context enabled in carry-over cells
- Match evaluation with previous context disabled
- Storage of current match result for use with next processing cycle.

The algorithm has been implemented using FPGAs, so processing time is bounded and depends mainly on the combinatorial delays of the priority encoder. An 128-byte length delay line was implemented in an Xilinx 4013 using architectural features such as on-chip RAM. Throughput of around 5 Mbytes/sec was achieved.

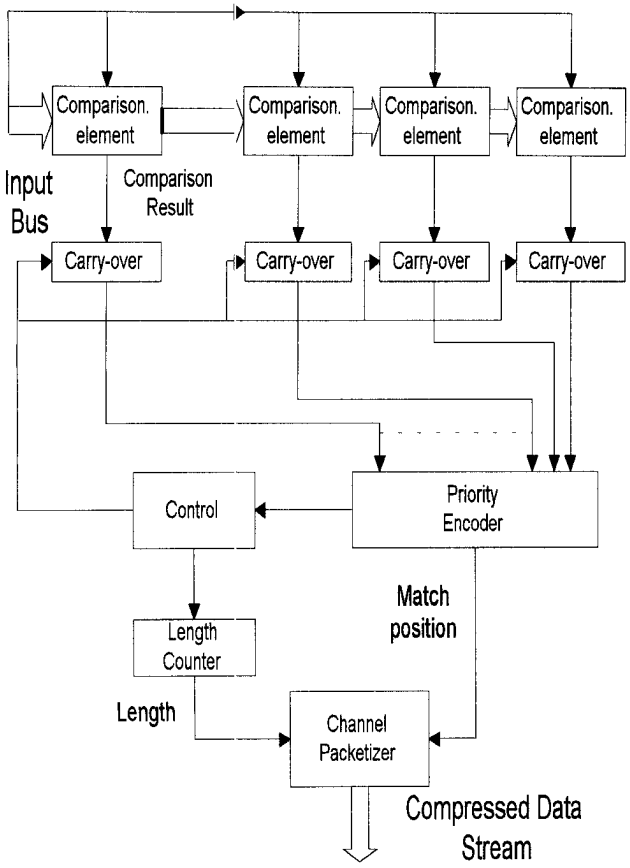

Figure 6. LZFastSearch encoding algorithm block diagram

5. Conclusion

In this paper we presented an analysis of available lossless compression algorithms and their feasibility for compressing real-time disparity data in terms of compression ratio and implementation complexity. We justify the choice of LZ algorithms based on the compression performance and complexity on simulation on real-time disparity sequences.

Various characteristics of the disparity bit-stream were given and the algorithms were modified to take advantage of those characteristics. A hardware architecture based on DSP and re-programmable logic hardware accelerators is presented and utilized for the evaluation of candidate compression algorithms. The compression algorithms reduce the data rate by a factor of two but don't satisfy the throughput requirement. An algorithm from the LZ77 and LZ78 are chosen for optimization. The first algorithm chosen gave good compression performance and high throughput on the chosen hardware base. The second algorithm gave faster adaptation performance to the localized redundancy of the disparity map and provided somewhat better compression performance and high decompression rate. The compression throughput requirement was achieved through a straightforward implementation on an FPGA.

References

[1] *ACTS Project AC092 PANORAMA: Package for New Operational Autostereoscopic Multiview Systems and Applications- Technical Annex*, 1995.

[2] T. Bell, J. G. Cleary, and I. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1990.

[3] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, September 1952.

[4] J. Ohm, K. Grueneberg, E. Izquirdo, M. Karl, E. Hendriks, P. Redert, D. Kalivas, and D. Papadimatos. A real-time hardware system for stereoscopic videoconferencing with viewpoint adaptation. *Image Communication, Special Issue on 3D technology*, January 1998.

[5] A. Redert and E. Hendriks. Disparity map coding for 3D teleconferencing applications. *Proceedings of the SPIE conference on Visual Communications and Image Processing (VCIP)*, 3024:369–379, San Jose, CA, USA, April 1997.

[6] J. A. Storer. *Data Compression*. Computer Science Press, Rockville, Maryland, USA, 1988.

[7] T. A. Welsh. A technique for high-performance data compression. *IEEE Computer*, 55(6):8–19, June 1984.

[8] M. Ziegler. *Region-Based Analysis and Coding of Stereoscopic Video*. Ph.D thesis, Akademischer Verlag Muenchen, Germany, 1997.

[9] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IT-23(3):337–343, May 1977.

[10] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, IT-24(5):530–536, September 1978.