High Speed Channel Coding Architectures for the Uncoordinated OR Channel

H. Chan[1], M. Griot[1], A. Vila Casado[1], R. Wesel[1], I. Verbauwhede[1,2]
{herwin, mgriot, avila, wesel, ingrid }@ee.ucla.edu

[1] Electrical Engineering Dept.
UC Los Angeles, USA

[2] Dept. ESAT/SCD-COSIC
K.U.Leuven, Belgium

ABSTRACT

Though it promises high bandwidths, the optical medium is not popular in local area networks. This is because current optical networks do not offer the ease of use and setup that an uncoordinated multiple access network such as Ethernet offers. By careful design and implementation of high speed channel coding architectures, we show that it is possible for optical networks to exhibit these desirable properties while maintaining high optical transmission rates. This paper presents an interleaver-division multiple access (IDMA) architecture implemented with a rate 1/20, 64-state Viterbi decoder and a word-based interleaver. These structures allowed us to achieve optical data rates of 2Gbps in FPGA implementation and 5.4Gbps for 0.18um ASIC implementation. The techniques presented can be adapted for other similar architectures.

1. Introduction

This paper describes the design of high speed signal processing architectures that are used to implement an uncoordinated access network in an OR channel. Specifically, an interleaver-division multiple access (IDMA) architecture implemented with a fully parallel Viterbi decoder and a word-indexed interleaver is discussed. To understand some of the design decisions, it is important to understand where such a channel exists and how we can use it to improve current networking standards.

Ethernet is a very popular standard for networking because of it's uncoordinated properties. Nodes can access the network at any time without informing any higher level authority. When two nodes happen to transmit at the same time, a collision is detected and the two nodes wait a random amount of time before attempting to re-transmit. In cases where there are many nodes in the system trying to transmit, Ethernet degrades rapidly [1]. Total useful network bandwidth is greatly reduced. In addition, the delay in which a message experiences before reaching the receiver is increased. This situation is caused by the inability of the system to transfer data when a collision occurs.

This need not be the case. In the CANbus network [2], collisions between data of several transmitters are used to determine the priority of the messages. Though CANbus only allows collisions in the header of a data transmission to determine priority, it illustrates that electrical signals can be used as an OR channel, and that data can be transmitted when a collision occurs.

An OR channel is a channel that behaves like an N-input OR gate, where N is the number of nodes transmitting simultaneously. If we assume on-off keying, if any node transmits a '1' data bit, all the receivers will see a '1' bit in the channel. If all nodes transmit a '0' bit, then the receivers will see a '0' in the channel. In the CANbus network, the '1' bit is called the dominant bit since its value hides the presence of any '0' bits. A passive optical star network can also be used as an OR channel. Physically, the dominant '1' bit is represented by the presence of light and the '0' bit is presented by the absence of light.

In our system, we assume the presence of collisions in the OR channel and implement novel channel codes to ensure that data is correctly transmitted. The nodes in the system are assumed to be uncoordinated and need not even be aware of the existence of other nodes in the system. When applied to the optical channel, our system is able to combine uncoordinated multiple access properties with optical bandwidth. There are no transmission delays and throughputs are guaranteed regardless of the number of simultaneously transmitting nodes.

This paper is organized as follows. Section 2 introduces the general channel coding architecture and discusses the implementation details of the Viterbi decoder and interleaver blocks specifically. Section 3 shows the speed and size of the result for both ASIC and FPGA targets and Section 4 draws some conclusions.

2. Channel Coding Architecture

We developed a channel coding architecture to demonstrate the feasibility of uncoordinated access in the optical channel at optical data rates (Figure 1). A rate 1/20 code has been designed to protect data from interference in an OR channel.
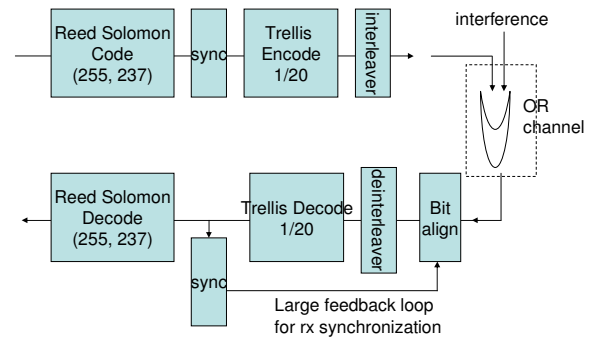

Figure 1. Channel coding architecture

During code design, the interference signal was assumed to have a random uniform distribution, therefore interleavers are used

after channel coding to randomize the position of the code bits. This combination allows us to recover data at a BER of 1e-5. A Reed Solomon block code is added at the back end to reduce the BER further to 1e-9. Since our target physical layer is the optical channel, data throughput is the main design criterion. The Viterbi decoder and interleaver blocks have been identified as the bottlenecks of the system and novel architectures are developed to mitigate their effects.

2.1 Trellis Encoder

To protect data in the OR channel, we designed a non-linear binary trellis code that uses 20-bit codewords and contains 64 states. The design of the encoder consists of a 6 bit register used to address memory that contains all the 128 possible codewords. Each clock cycle a new data bit is shifted into the register and a new 20 bit codeword is produced. Details of the trellis code design procedure are presented in [3].

2.2 Viterbi Decoder

In DSP implementations, the Viterbi decoder focused on the acceleration of a single branch metric calculation and careful memory management for storing the results. This means for decoding a single code word, several clock cycles (depending on the number of states in the trellis code) are needed. Hardware architectures such as [4, 5] has focused on area efficient architectures. For common wireless applications, such as 802.11b and 802.16a, [6] describes a fast parallel hardware implementation that decodes at 160Mbps on a FPGA.

For our non-linear trellis code that uses 20-bit codewords and contains 64 states, a traceback length of 35 is used in the Viterbi decoder. The technique used to design the decoder is to parallelize and pipeline all operations as much as possible. Care was taken to find structures where feedback paths are as short as possible. The overall architecture of the Viterbi decoder is shown in Figure 2.

Figure 2. Viterbi decoder architecture

The Viterbi decoder can be divided into several different stages, each of these stages will be discussed individually in detail:
- calculation of metric
- accumulation and selection of metric
- finding of minimum path
- subtraction of accumulated result

2.2.1 Branch Metric
Because the Viterbi decoder is being designed for the OR channel, the branch metric used is different from traditional designs. In an

OR channel, it is impossible to receive a 0 bit when a 1 bit has been transmitted by any of the nodes. Because of this, in the comparison between the received codeword and branch codeword, if any of the received bits is 0 when a 1 is expected the branch metric is set to a maximum value of 20. Errors in which a 1 is received when a 0 is expected are summed together to give the branch metric in the normal case. The logic used to implement this function is shown in Figure 3. In our 64 state codes, 128 branch metrics are calculated in parallel; the logic used to calculate this function makes up one stage in our pipeline.

Figure 3. Calculation of path metric

2.2.2 Accumulation and Selection
There are two possible branches that lead to each of the 64 state nodes. The path with the smallest path metric (which is an accumulation of past branch metrics) is chosen as the most likely path that was taken to reach the node. Path metric calculation is performed by adding the path metric of the source nodes to their respective branch metrics. The two sums are then compared and the path with the lowest metric is selected. Sixty-four of these calculations are performed in parallel and constitutes a single stage in our pipeline. Further pipelining of this stage is impossible since the calculation of the path metric involves a feedback path from previous path metric calculations. Figure 3 shows the implementation of this function.

2.2.3 Finding Minimum Path
The most likely bit that was transmitted is the bit at the head of the path with the lowest path metric. At each cycle, 64 path metrics are calculated and their respective paths are accumulated. A sorting network is used to select the path with the smallest accumulated metric. A minimum time sorting network based on Batcher's odd-even merging algorithm [7] is used. This is a recursive algorithm that sorts a group of unordered numbers (Figure 4) and contains the following three steps:
1. Divide the numbers in to two groups
2. Sort the two groups of numbers separately
3. odd-even merge the two groups of numbers.

Figure 4. Block diagram of sorting network

The algorithm can be completely implement using only two input comparators. For the sorting of n numbers, the number of comparators grows in $O(n \log n^2)$. The delay through the network is $\begin{pmatrix} 1 + \log n \\ 2 \end{pmatrix}$. For our system of 64 states, this translates to 543 comparators with a delay of 21 comparators. However, since there are no feedback paths in the sorting algorithm, the architecture can be fully pipelined to achieve very fast throughputs.

2.2.4 Subtraction of accumulated result
The minimum path metric is fed back to the Viterbi decoder and subtracted from all 64 accumulated path metrics. This is to ensure that the register values do not overflow. The sorting network used to find the minimum path is heavily pipelined, so the value used is several cycles behind the values that are currently calculated. This delay in the results translates to larger possible accumulated path metrics which may necessitate the use of larger operators (like adders) with may slow down the system. Therefore, care was taken to pipeline the sorting network only to the degree that is necessary to avoid unnecessary increases in hardware and possible increases in critical path delays. The sorting network in our design is pipelined to have 6 cycles of delay.

2.3 Interleaver
Interleavers, which permute the order of data bits, are commonly used to randomize the data stream to improve the performance of error correcting codes. Much work in this area has been focused on the design of interleavers in conjunction with convolutional and turbo codes. Interleaver specifications are found in the IEEE 802.16a, 802.11a/g and 802.11n standards. The interleavers found in these systems are different from the ones in which we design in two significant ways:
1. The interleaver pattern is static for a particular standard; in our system, they are used to protect a transmission from interference in an OR channel.
2. The supported data rates are in the order of 100Mbps; optical channels support data rates that are at least an order of magnitude faster.

In our system, each transmitter uses a unique interleaver pattern. This pattern is chosen from a set of patterns determined at design time to have good cross correlation properties. The role of the interleaver in our system is similar to its role in an IDMA system. In that system [8], interleavers are used to distinguish nodes in a wireless CDMA system and increase channel capacity. The interleaver design, therefore, must be flexible enough to accommodate a family of permutation sequences that work well together. Interleaver design for IDMA has been examined [9], however, the focus has been on performance efficiency rather than high-speed implementation. As a consequence, those results are unable to support our high data rates.

A de-interleaver is used at the receiver to recover the initial sequence. Its architecture is the same as the interleaver architecture; the permutation sequences, however, are run in reverse order to recover the original uninterleaved signal.

For our 1600-bit interleaver implementation, we adopted a randomized write-by-row, read-by-column scheme. As seen in Figure 5, data can be broken into square blocks of 400 bits. Each of the 20 rows and columns are indexed. Groups of 20 incoming bits are written to a randomly indexed row. When the data block is filled, the bits are read out of the block one column at a time in a random order.
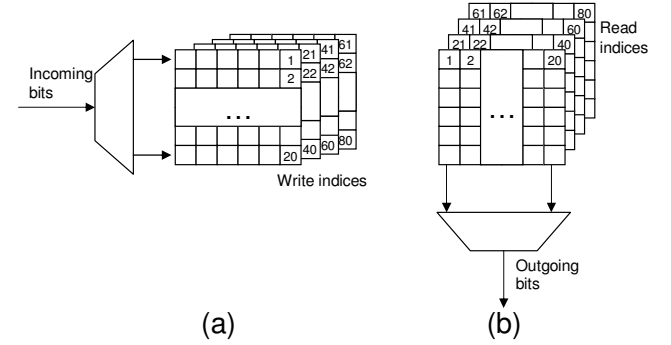

Figure 5: Indexed write-by-row, read-by-column interleaver

The 400-bit-square block forms the basic unit of our interleaver design. In order to produce the necessary randomness, four of such blocks were used in our final implementation. Like the indexing within the blocks, the inputs and outputs of the four blocks are accessed independently and randomly.

This scheme provides us with enough randomness to operate on the optical channel. In our interleaver design, 4 square blocks of 400 bits are used, giving us a total of 80 indexed locations. This corresponds to a design space of $(80!)^2 > 1e+237$ possible permutation sequences to choose from. For the desired channel rate of 2 Gbps, using the 20-bit wordsize of our trellis code, our target operating frequency for the interleaver is 100MHz.

2.4 Reed Solomon Code
When the Trellis decoder block makes an error, the errors usually come in a burst of a few bits at a time. A Reed Solomon (RS) code is a block code that operates on bytes at a time. This makes it a very good choice to correct the residual errors and bring the final BER to below 1e-9. A standard (255,237) RS code was selected.

Since timing is not critical in this block, a standard architecture design is used [10]. The data rate at the output of the NL-TCM decoder is 100Mbps. Since the Reed Solomon code operates on data blocks of 255 bytes (2040 bits), the time budget for the RS decoder is 20.4us. We clocked the module at 50MHz, and in the worse case the decoding operation takes 856 cycles (17.1us) to complete.

3. Implementation Results
The coding structures were implemented in both FPGA and ASIC targets and results are discussed in the following sections.

3.1 FPGA
The system blocks were implemented on the VirtexII-Pro FPGA from Xilinx. Table 1 summarizes the size various blocks in the design. The critical period is given for the transmitter and receiver.

Table 1: Size and speed of transmitter and receiver blocks implemented on the VirtexII-Pro FPGA

	Size (slices)	Critical period (ns)
Transmitter		
Reed Solomon encode	189	5.3
NL-TCM encode	34	3.4
Interleaver	3387	7.7
Receiver		
Reed Solomon decode	3686	9.0
NL-TCM decode (Viterbi)	10504	10.3
Interleaver	3387	7.7

The transmitter is implemented on the VirtexII Pro XC2VP20 FPGA which is contains 9,230 slices of logic. Each transmitter design occupies 40% of the available area. The receiver is a significantly larger design and is implemented on the XC2VP50 which has a capacity of 23,616 slices. The receiver design occupies 70% of the available area. The maximum throughput for an FPGA implementation is 2.0Gbps channel speed and is limited by the Viterbi decoding block.

3.2 ASIC

The system blocks were synthesized using Synopsis Design Compiler. The TSMC 0.18um CMOS standard cell library with conservative wire load model was used. Table 2 summarizes the size various blocks in the design. The critical period is given for the transmitter and receiver.

Table 2: Size and speed of transmitter and receiver blocks synthesized in 0.18um ASIC

	Size (Kgates)	Critical period (ns)
Transmitter		
Reed Solomon encode	3	1.6
NL-TCM encode	9	1.3
Interleaver	39	3.7
Receiver		
Reed Solomon decode	29	10.3
NL-TCM decode (Viterbi)	239	1.4
Interleaver	39	3.7

For the ASIC implementation, the receiver is limited by the Reed Solomon decoder block to a channel speed of 4.6Gbps. The implementation of this block, however, is not optimized and there have been published results showing decoders that can handle channel rates of up to 50Gbps. If one of these implementations is used, the maximum throughput for an ASIC implementation is 5.4Gbps channel speed.

It is interesting to note that the interleaver is the bottleneck in the ASIC while the Viterbi decoder is the bottleneck for FPGA. This is due to increased routing delays within blocks that take up a large area in the FPGA. Since the Viterbi decoder is at least 3 times larger than the interleaver block, the effect is more pronounced.

4. Conclusions

An uncoordinated multiple access system using the optical channel was introduced in this paper. This was accomplished by careful design of a non-linear trellis code combined with interleavers to differentiate the nodes in the system. To support the high optical data rates, new coding architectures were developed that took advantage of data parallelization and pipelining techniques.

In addition to fully parallelizing the Viterbi architecture, care was taken to minimize computational feedback paths in the system. The feedback paths that existed were kept small. This allowed us to pipeline the design to give the minimum critical path and therefore the maximum throughput. In the design of the interleaver, data was processed using 20-bit words. A novel write-by-row, read-by-column scheme allows fast processing of bits while maintaining the necessary freedom to choose good permutation patterns.

The work done on uncoordinated access in optical networks seems to be a very attractive idea for LANs that require high bandwidth. Our implementation shows that current off the shelf technology can be feasibly used to build such a system. The architectures that we introduce in this paper can support throughputs of up to 2.0 Gbps for FPGAs and 5.4 Gbps for 0.18um standard cell ASICs. In addition, the techniques presented here can be used in other high speed channel coding designs.

Acknowledgements

References

[1]   D. R. Boggs, J.C. Mogul, and C.A. Kent, "Measured Capacity of an Ethernet: Myths and Reality," ACM SIGCOMM '88 Symposium on Communications Architectures and Protocols, pp. 222-234.

[2]   CANbus specification http://www.semiconductors.bosch.de/de/20/can/index.asp

[3]   M. Griot, A. Vila Casado, W. Y. Weng, H. Chan, J. Basak, E. Yablonovitch, I. Verbauwhede, B. Jalali, and R. D. Wesel, "Interleaver-Division Multiple Access on the OR Channel," First Annual Workshop on Information Theory and its Applications, San Diego, CA, February 2006.

[4]   Y. Zhu and M. Benaissa, "A Novel ACS Scheme for Area-Efficient Viterbi Decoders," 2003.

[5]   M. Guo, O. Ahmad, M. Swamy, and C. Wang, "A Low-Power Systolic Array-Based Adaptive Viterbi Decoder and its FPGA Implementation," 2003.

[6]   A. Abdul Shakoor, V. Szwarc, and T. Kwasniewski, "High Speed Viterbi Decoder for W-LAN and Broadband Applications," 2004.

[7]   D. Knuth, The Art of Computer Programming Vol. 3 Sorting and Searching, Addison-Wesley, Reading, MA, 1973, pp.229-232.

[8]   L. Ping, L. Liu, and W. K. Leung, "A Simple Approach to Near-Optimal Multiuser Detection: Interleaver-Division Multiple Access," IEEE Wireless Communications and Networking Conference, pp. 391-396, 2003.

[9]   I. Pupeza, A. Kavcic, and L. Ping, "Efficient Generation of Interleavers for IDMA," accepted IEEE International Conference on Communications 2006.

[10] based on code found at http://www.humanistic.org/~hendrik/