APPLYING ASYNCHRONOUS TECHNIQUES TO A VITERBI DECODER DESIGN

L. E. M. Brackenbury, M. Cumpstey, S. B. Furber, P. A. Riocreux

INTRODUCTION

Viterbi decoders are used for decoding convolutional forward error correction codes [1] in a large proportion of digital transmission systems including mobile phones and digital television. For portable applications, the battery size and lifetime is of commercial importance as is the size of the electronics. Therefore, a low power, area efficient implementation is of commercial interest as a means of both lowering costs and extending the battery operating time.

PREST and Powerpack have been collaborative projects, funded by the EU and EPSRC respectively, which have resulted in different Universities designing low power alternatives to an industry standard Viterbi Decoder [2]. The techniques involved for reducing power from current levels range from circuits and timing strategies to architectures and algorithms.

VITERBI ENCODING

The encoding operation performs a convolution and parity calculation on the input; the last k bits of the input stream are stored in a shift register and these are XORed with a (fixed) k-bit pattern and the parity of the resulting stream is the encoder output. Redundancy and forward error correction are obtained by using more than one pattern for the convolution so that the encoder produces multiple output bits for each input; in our case two output bits of three bits each are produced for each input bit using a 7-bit shift register. The rate of 1 input bit to two (encoded) output bits can be varied by puncturing, where output bit(s) from the encoder are omitted or in our case marked as invalid.

CONVENTIONAL DECODING

Although encoding is simple, decoding is computationally expensive leading to the use of dedicated hardware. In a synchronous system the decoder consists of a branch metric unit (BMU), a path metric unit (PMU) and a survivor memory unit (SMU). The bits transmitted by the encoder are received as soft bits by the BMU in the decoder. The BMU calculates the distance between the received bits and the four possible ideal symbols that could have been transmitted. These distances or branch metric weights are the likelihood, given what is known to have been received, that each of the ideal symbols was in fact what was transmitted.

The path metric unit (PMU) is the core of the Viterbi decoder. Here accumulative weight information relating to each possible encoder state (or node) is maintained. For each input to the decoder, the current score of each node in the PMU is added to the branch metric weight whose symbols indicate an incoming '0' and to the branch metric weight indicating an incoming '1' input. These indicate a new possible weight for the receiving nodes. However, since there are two branches from each node, this results in two possible scores being the next weight for a node. This is resolved by comparing the two weights into a node, with the lower being selected as the more likely route to the node; this (lower)

* Department of Computer Science, Manchester University, Oxford Road, Manchester M13 9PL

weight then becomes the new score for the node and the process repeats when the next input is received. In a synchronous system, the node arithmetic in the PMU is performed by add, compare and select elements.

The most likely route into each node is passed to the SMU which keeps a history, stretching back over many time slots, of the node winners. Only one bit per node need be stored per timeslot; this indicates if the winner route was along the upper or lower path into the node. Enough history is maintained in a set of RAMs, so that starting at any random point and tracing back through the state space, then the path eventually taken indicates the data that is most likely to have been transmitted by the Encoder at that time and it is this which is output.

ASYNCHRONOUS DESIGN

At Manchester University, the approach adopted in the Viterbi design is to use a self-timed (or asynchronous) timing strategy. This saves power through not having to generate or distribute a global clock. Instead, timing between blocks is performed by local handshake signals [3]. This enables an asynchronous system to only consume power when doing useful work and to have an idle power of near zero. Furthermore, there is an inherent advantage to asynchronous systems in that a system can switch almost instantaneously between the idle state and maximum activity; this is much more difficult to organise in a clocking system.

An asynchronous approach encourages designers to design in a modular manner. This means that sections can operate independently, concurrently and at their fastest natural rate. A further significant advantage of an asynchronous system over synchronous systems, which is of particular importance in digital transmission systems, is the low electromagnetic interference generated; this occurs because the circuit switching is spread in time rather than being concentrated around a clock edge.

Experience with asynchronous design shows that a synchronous design cannot be directly translated into a self-timed implementation; this is a result of the different levels of granularity required. Thus in designing an asynchronous Viterbi Decoder, the design needed to be considered from first principles.

SELF-TIMED VITERBI ARCHITECTURE

The asynchronous Viterbi decoder comprises a BMU, PMU and a HU (history unit which is called a SMU in synchronous systems), as shown in figure 1. The presence of the clock is purely to synchronise the decoder input and output with its external clocked environment. Communication within the decoder blocks is controlled by the Request-Acknowledge handshake pair which respectively signal that valid data is ready and that it has been accepted.

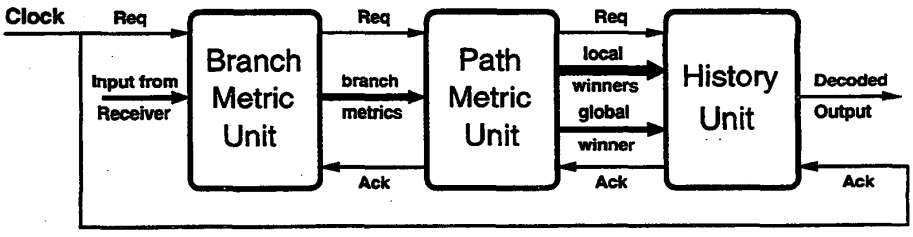

Figure 1. Top-Level Asynchronous Architecture

Like the synchronous systems, the BMU calculates a metric representing the distance between the received bits and the four possible ideal symbols that could have been transmitted. However, in the asyn-

chronous system, these weights are computed in the form of a number of events since this is the form of arithmetic used in the PMU.

Serial unary arithmetic has been adopted within the PMU. Here, all weights are held as a series of events in data-less FIFOs, with the data implicitly indicated by the state of the FIFO control elements. Two phase arithmetic is used so that the levels convey no meaning. However, an edge (or change of state from one control element to the next) indicates an event which is interpreted as a count of one. The basic unit of replication in the PMU is a node pair due to the nature of the butterfly interconnection network between node outputs and node inputs. In this particular system there are 32 node pairs i.e. 64 nodes. The logic for a node pair is illustrated in figure 2.

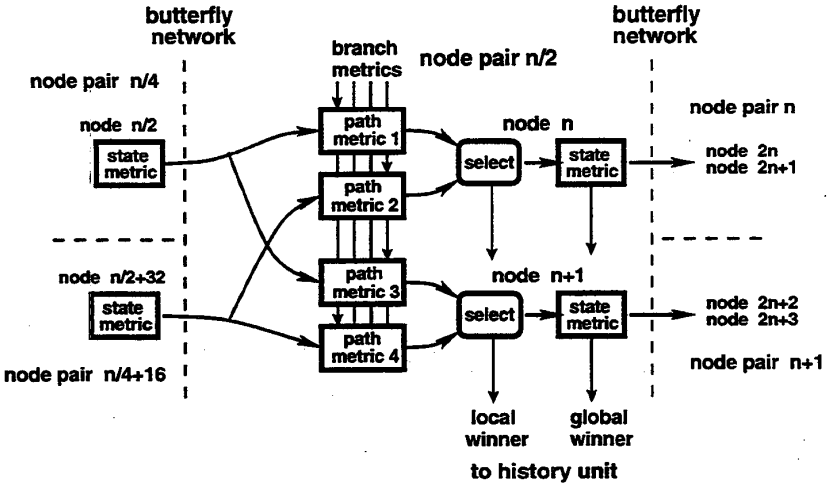

Figure 2. Asynchronous Node Pair

The path metric FIFOs are parallel loaded at the start of a timeslot with the branch weight metrics resulting from the input comparison; this overwrites any existing count in these FIFOs. The state metrics connected to these FIFOs across the butterfly interconnection are now serially transferred in to the path metric FIFOs, adding to the existing path metric weight and emptying the transmitting state metric FIFO. The use of serial unary arithmetic enables the compare-select operation to be very simply implemented with just a 2-input Muller C-gate, labelled select in figure 2. This gate produces an output event transition whenever an input transition occurs on both its inputs.

Once the state metric data has been transferred into the path metric FIFOs and the state metric FIFO to which these will transfer is empty, events from the two path metric FIFOs leading to an output node are paired by the select element and entered into the receiving state metric FIFO. This continues until the smaller of the two path metric FIFOs is empty, identifying it as the local winner for the node in that timeslot.

The use of scaling, weight capping and normalisation [4] where possible enables weights to be kept to seven or less. Furthermore, in most cases, the data received is correct and in this case, there will be a node with a zero state metric. In most conventional systems, an overall winning node with a zero score is not identified; usually the scores and number of nodes make this not viable. However, in the asynchronous system, a global winner in the form of a 6-bite node identity can be obtained relatively easily and this plus the local winner information is passed to the history unit.The PMU activity for the timeslot is then complete and the sequence repeats with the next set of branch metric weights.

The history unit (HU) stores both a 'good' path indicated by global winner node identities and the local winners at each node. This information is stored within the HU which holds similar information for the

last 64 time slots. Unusually, a single storage area only is required and this is implemented as an array of latches organised as a circular buffer. This sliding window scheme is illustrated in figure 3. A circulating token indicates which HU slice is the current head. This slice determines the next output from the Decoder and receives the new local and global winner information from the PMU. Once this has taken place, the slice's head token is passed to the next slice by its control unit, which now becomes the new head.

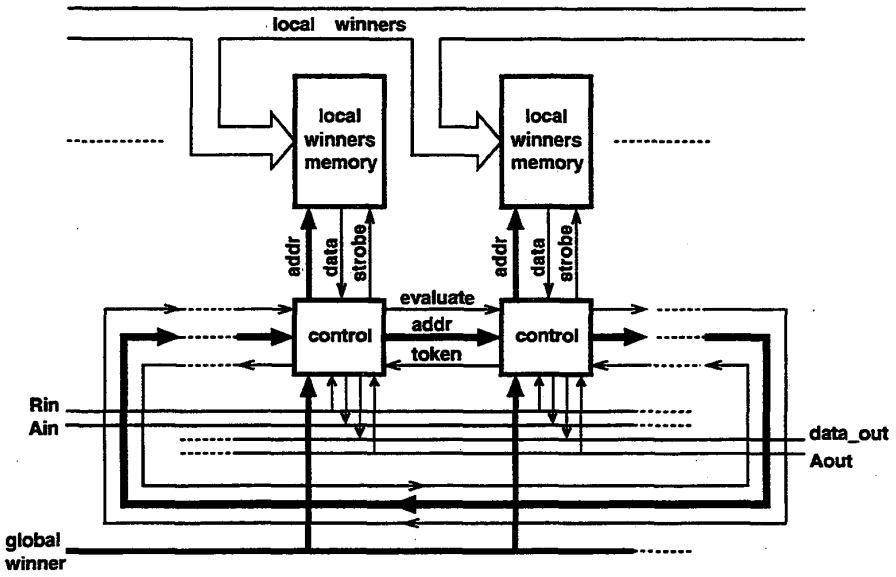

Figure 3. History Unit

Usually, the data received from the PMU contains no errors so that the current global winner is the child of the last winner. Since there is a simple arithmetic relationship between parent and child, the parent of the current winner is computed. This is then passed to the control of the preceding slice on the addr lines together with the evaluate signal. The parent slice checks the incoming addr bits against its global winner and if (as usual) they are identical then the global winner store still holds a good path; in this case no further tracing back is performed by the HU.

If the new overall winner is not related to the previous global winner because there is a data error, then the 'good' path is incorrect and the addr which accompanies the evaluate signal is written into the global winner store for the parent slice and also used to read out the local winner bit; this enables the parent of the new global winner of the parent slice to be determined and this is passed to the preceding timeslice with an evaluate signal. This process continues with a new 'good' path being constructed by reading back through the local winner memory. Usually, in performing this backtracing reconstruction, the new path converges with the existing global winners after a few timeslots have been searched and the backtrace process is then retired.

In an asynchronous system, the backtrace process only runs if required and then for only as long as required. It also runs completely independently of adding current information to the HU memory. Furthermore, the use of latches for the store instead of RAM (which is usually chosen) has enabled even greater concurrency of operation in the HU as multiple backtraces can be running independently and asynchronously of one another.

CONCLUSIONS

A Viterbi Decoder aimed at a low power implementation has been described. The use of an asynchronous timing strategy has resulted in a novel design. The use of scaling, weight capping and normalisa-

tion has enabled much design simplification in the node pair logic of the PMU in particular. Principally, the arithmetic here has been reduced from (typically) 8-bit parallel arithmetic to a serial-event bit stream. The HU should also yield a significant power saving since backtraces are only undertaken if necessary and terminate as soon as possible. The asynchronous framework particularly supports this as the handshakes enable a backtrace to be automatically entered or exited without an overhead or performance penalty. These considerations indicate that significant power savings on a conventional design should be realised. Now that the asynchronous design has been fabricated as well as the conventional industry-standard device and two other low-power synchronous decoders, testing will enable meaningful comparisons between the different strategies for low power design.

REFERENCES

1. A. J. Viterbi, 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', IEEE Transactions Information Theory, Vol.13, pp.260-269, April 1967.
2. George Abouyannis et al., 'Project PREST EP25242', European Low Power Initiative for Electronic System Design (ESDLPD) Third International Workshop, pp.5-49, July 2000.
3. I. E. Sutherland, 'Micropipelines', Communications of the ACM, Vol. 32, No. 6, pp.720-738, 1989.
4. G. C. Clark Jr. and J. B. Cain, 'Error-Correcting Coding for Digital Communication', Plenum, New York, 1981.

ACKNOWLEDGMENTS

The authors are grateful for the funding of this work by the EPSRC PowerPack project GR/L27930 and by the EU PREST project EP25242.