XAPP551 (1.0) February 14, 2005

Viterbi Decoder Block Decoding - Trellis Termination and Tail Biting

Authors: Bill Wilkie and Beth Cowie

Summary

Many digital communication standards employ Convolution Coding as a means of forward error correction (FEC). Data encoded in this way generally is decoded with a Viterbi decoder, which operates by constructing a *trellis* of state probabilities and branch metrics. The transmitted data is often terminated with a number of zeros to force the encoder back to the zero state. This allows the decoder to start decoding from a known state, however, extra symbols have to be transmitted over the channel.

Another termination technique is to ensure the trellis start and end states are identical. This technique is referred to as *tail biting* and has the advantage of not requiring any extra symbols to be transmitted. Tail biting is used in several popular communications standards, such as IEEE802.16. This application note explains how to use the Xilinx Viterbi Decoder LogiCORE™ module (version 5.0 or later) to implement both trellis termination and tail biting.

Viterbi Decoding

A description of the general Viterbi decoding algorithm is beyond the scope of this note. Refer to the Internet for available tutorial notes. The important concept to understand is that a trellis is constructed by computing the cost of being in each possible convolution encoder state at every symbol period. The data bit (0 or 1) most likely to have caused entry to each state is stored in a table. For example, if the encoder has a six register delay line (that is, constraint length 7), 2^6 bits are stored for each cycle. After a number of clock cycles, defined by the traceback length, the decoder traces back through the trellis, outputting the data bits for the most likely survivor path. This operation is referred to as *traceback*. Then these bits are passed through a last in, first out (LIFO) structure, so they are output in the order originally received. Usually the traceback length is set to be greater than six times the constraint length or greater than 10 times if the data is punctured.

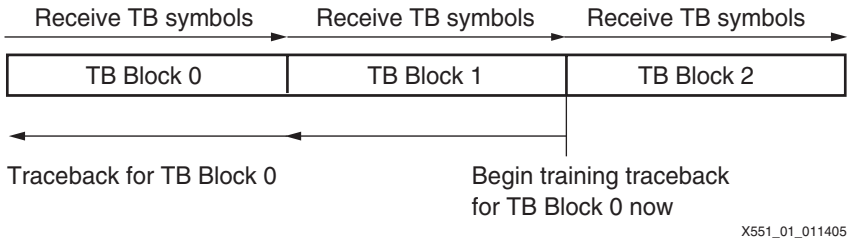

Figure 1: **Viterbi Decoder Traceback**

Figure 1 shows an example where the data bits causing entry to each state in the trellis are stored while TB Block 0 is received (TB is the traceback length). As more symbols are received, the data bits causing entry to each state in the trellis for Block 1 are stored. At this point, there are two traceback length's worth of data stored. The decoder now performs the traceback operation on Block 1, finding the most likely survivor path. The data bits for Block 1 are not output at this time. This operation determines the correct state to start the traceback through Block 0. In other words, Block 1 is used for training to start the traceback of Block 0 from the correct state. In reality, the upper and lower arrows in Figure 1 are skewed in time. The

traceback for a block occurs some time after the last symbol in the block is sampled. The arrows are positioned to show on which data the operation is acting. The blocks are sometimes referred to as *windows*, and the technique of processing one window at a time is referred to as the *sliding window* technique. In the same way, Block 2 is used as a training sequence to start the traceback of Block 1 from the correct state.

Viterbi Block Decoding

Convolution codes are not strictly block codes. The decoder operates on a continuous stream of incoming encoded data, splitting it into traceback lengths for processing. However, it is convenient to split the data into packets and regard each packet as a self-contained, independent block. Each packet can contain many traceback lengths of data. Generally, the encoder is forced to a known starting state and then forced to a known ending state after the last data bit is shifted in. The most frequently used methods to terminate the trellis in a Viterbi decoder are:

- trellis truncation
- trellis termination
- tail biting

Trellis Truncation

Trellis truncation is the simplest of all the methods. The encoder is reset to zero at the beginning of each packet. Then the data is shifted in. Nothing is done to force the encoder into a special termination state.

This method has the following advantages:

- It is simple to implement.
- The code rate is not affected, that is, the ratio of original data bits to transmitted bits remains the same. If the encoder code rate is R and N bits are input, then a total of N/R bits are transmitted.

The disadvantage of this method is that the bit error rate (BER) performance of the code is degraded because the decoder does not know from which state to start the last traceback and has no data to use as a training sequence to find the correct state. The distance properties of the convolution code are lost at the end of the packet.

Trellis Termination

Trellis termination is the most common technique. The encoder is reset to zero at the beginning of each packet as in trellis truncation. This operation generally happens automatically, as the previous packet ends in state 0. After all data bits are shifted into the encoder, another Z zero bits are shifted in, where Z is the number of register stages in the encoder shift register. This resets the encoder state back to zero. The final output symbols are generated as the Zth zero is on the input to the encoder shift register. On the next clock cycle, the encoder enters state 0 and no more symbols are transmitted for this packet.

Figure 2 shows this process for a constraint length 7 example. The encoder is initialized to all zeros, and the first symbol is formed when DATA_IN = d_0. The last symbol of the packet is formed when d_{N-1} is in the right-most register and the final zero is on DATA_IN. After one more shift, the encoder is returned to the all zero state. Note that the DATA_OUT symbols are not transmitted for this state. The decoder can begin traceback for the packet in the certain knowledge that state 0 is the correct state from which to start.

In trellis termination, both the start state and end state are known by the decoder. This method has the following advantages:

- It is fairly simple to implement.
- Unlike simple trellis truncation, the termination does not affect the error correction properties of the convolution code.

The disadvantage of this method is that extra bits have to be transmitted, reducing the code rate. If there were originally N data bits, then a total of (N+Z)/R bits are transmitted. The extra bits consume extra transmission time and slightly reduce the E_b/N_o for a given probability of error. Typically, the overall effect is insignificant except when N is very small.

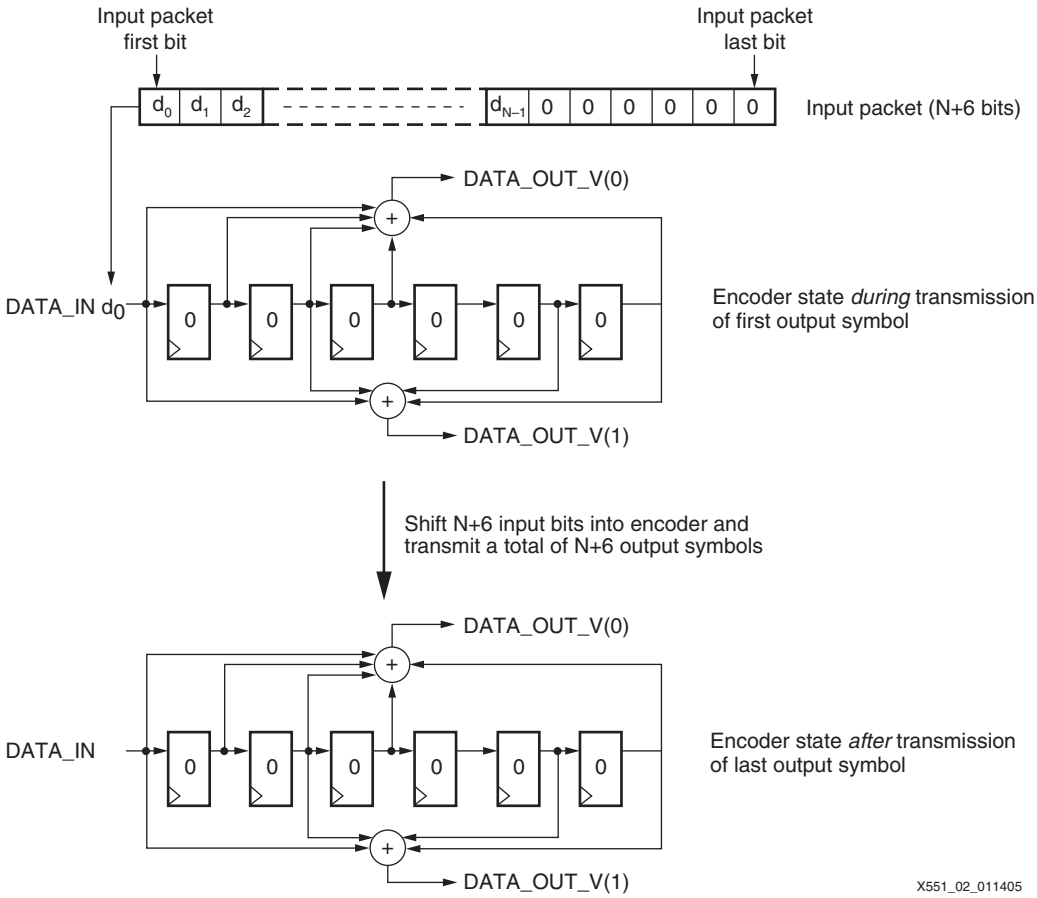

Figure 2: **Convolution Encoding for Trellis Zero Termination**

Tail Biting

Tail biting attempts to overcome the problem of having to transmit extra termination bits experienced by trellis termination. The last Z data bits of the packet are used to initialize the encoder shift register prior to transmission of the packet. No output symbols are transmitted during this encoder initialization, meaning that the encoder start state and end state for the packet are identical. It also implies that the entire packet must be available at the encoder before the first symbol is transmitted. Figure 3 shows this for a constraint length 7 example. The encoder is initialized to the last six bits of the packet, and the first symbol is formed when DATA_IN = d_0. The last symbol of the packet is formed when d_{N-1} is on DATA_IN. One more shift puts the encoder back into the initial state. The decoder can then begin traceback of the packet from that state.

Another technique is to initialize the encoder with the first Z data bits of the packet, again not transmitting any output symbols during this time. Then the remaining (N-Z) data bits are encoded and transmitted, followed by the first Z bits. This has the same effect of causing the start and end states to be identical. The advantage of this technique is that it does not require the entire packet before encoding starts, however, the bits are out of sequence at the receiver.

The tail-biting technique has the following advantages:

- The code rate is not affected, with a total of N/R bits being transmitted.
- Error correction properties of the convolution code are not affected.

This technique has the following disadvantages:

- Decoding latency is increased over trellis termination, because training is required to determine the correct start state and initial traceback state.
- Receiver complexity is slightly increased.

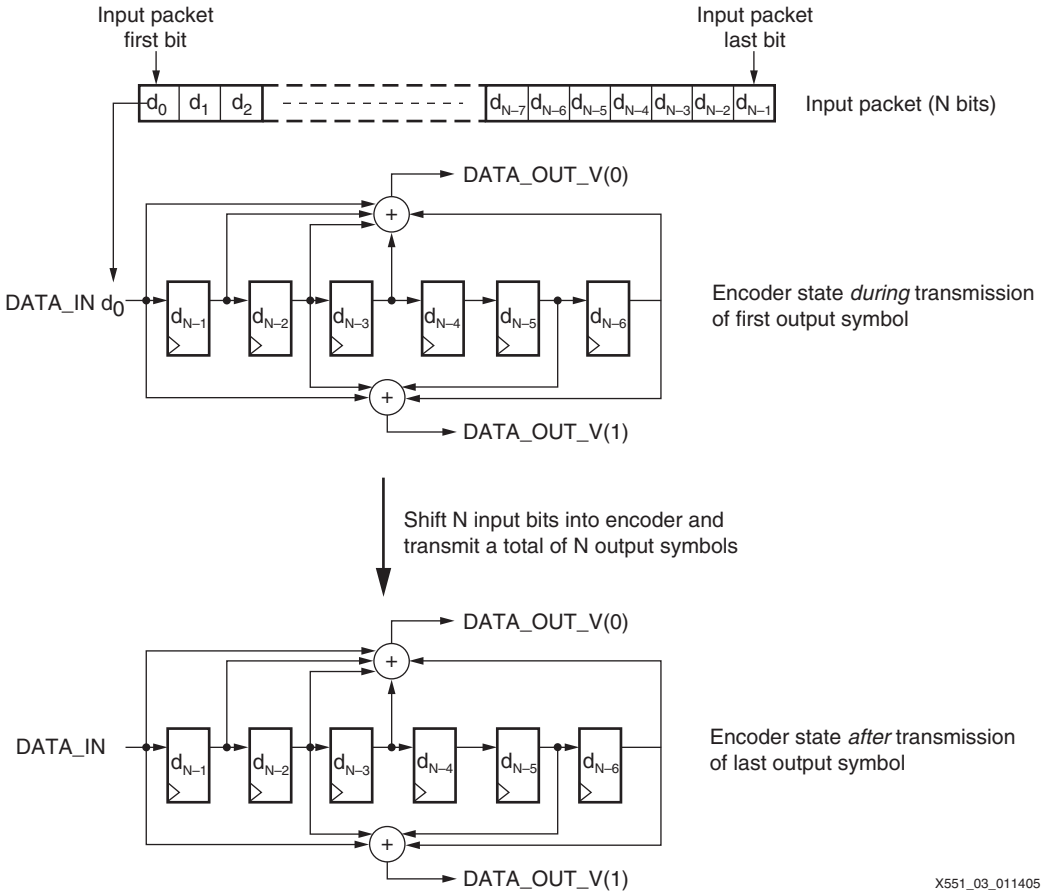

Figure 3: **Convolution Encoding for Tail-Biting Termination**

Implementation

All the termination methods can be implemented with the standard Xilinx Viterbi Decoder LogiCORE module.

Trellis Truncation Implementation

The Viterbi decoder decodes in the normal way, using a sliding window, as shown in Figure 1, page 1. The trellis construction should start from state 0. The decoder can be forced to give state 0 the lowest cost at the start of trellis construction (see "Trellis Termination Implementation," page 6 for more information).

Dummy data or data for the next packet is clocked in after the end of the packet to provide a training sequence for the last block. This training sequence causes the traceback for the last block to start in a state that might not be the correct state.

One alternative is to not bother with a training sequence for the last block and start the traceback immediately from the state with the lowest cost. The core can be configured in this

way by selecting *Direct Traceback* in the core GUI. Select the *Best State* option to ensure the traceback begins from the lowest cost state. This option gives the best chance for correct decoding, however, the results are not as good as when a full training sequence is available. If traceback is started from the wrong state, some BER degradation occurs at the end of a packet. Note that the Best State decoding logic increases the core size.

When using the Direct Traceback technique (see Figure 4), the TB_BLOCK input is used to signal the position of the final TB block of data within a packet. This block is traced back without a prior training sequence to determine the traceback start state. Traceback of the final TB block begins immediately after TB_BLOCK is deasserted.

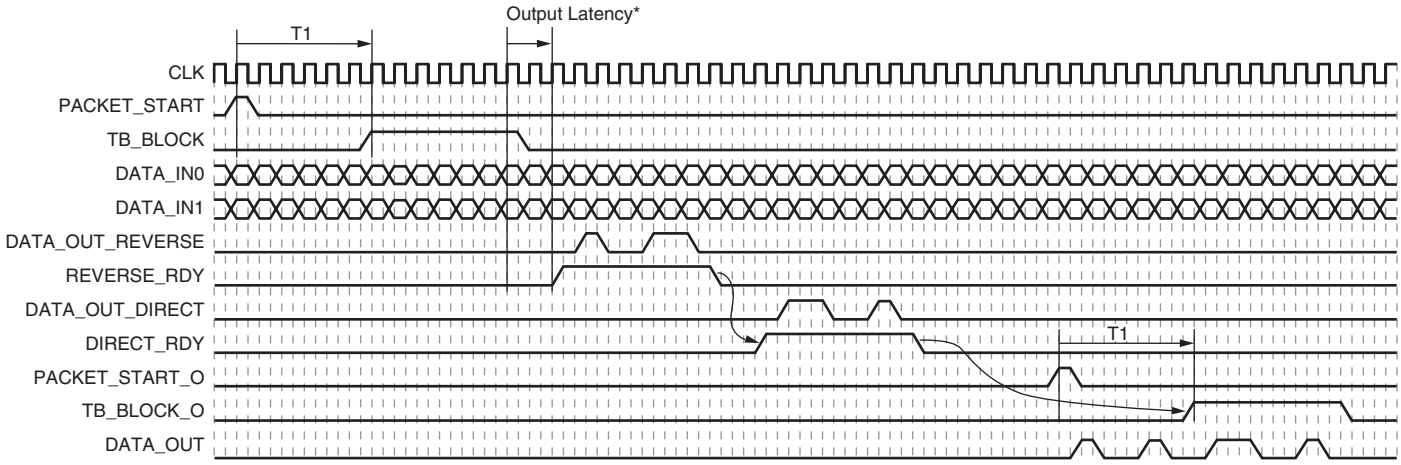

Figure 4: **Packet Handling**

In this example, PACKET_START is used to signal the start of the packet. Use PACKET_START when you want to force the trellis construction to start from a state other than the one the previous block ended in, as is the case with trellis truncation. See "Trellis Termination Implementation," page 6 for more details on PACKET_START.

TB_BLOCK can be asserted at any time. The T1 delay in Figure 4 can be any number of clock cycles and does not have to be an integer number of traceback lengths. The T1 delay is 0 clock cycles when there is only one traceback block in the packet. Typically the TB_BLOCK pulse has a duration of the number of remaining clock cycles when the packet length is divided by the traceback length. If this remainder is zero, then TB_BLOCK should be the same number of cycles as the traceback length. The TB_BLOCK pulse can be any duration, within the limits defined in the Viterbi Decoder LogiCORE data sheet (DS247). It does not matter if the pulse is less than the normally required traceback length, because a training sequence is not used to define the start state of the traceback in this case. TB_BLOCK is used in the same way with trellis termination and tail biting.

The decoder continues to output decoded data with the normal latency on DATA_OUT. If Direct Traceback is enabled, the decoder also outputs the data for the last block of the packet on DATA_OUT_REVERSE. This last block of data is output a small number of clock cycles after TB_BLOCK is deasserted, and provides data with the lowest possible latency. This is useful if there is only a single block in a packet because you can ignore DATA_OUT and take output data from DATA_OUT_REVERSE or DATA_OUT_DIRECT.

Overall latency can be reduced using DATA_OUT_REVERSE even if there are several blocks in a packet. In this case, DATA_OUT_REVERSE can be buffered in the appropriate locations of a dual-port RAM while the data for the earlier blocks (on DATA_OUT) is being written to the same RAM. This means that all decoding has completed as soon as the last symbol for the second last block on DATA_OUT is written to the RAM. Thus the overall latency can be reduced by TB cycles.

Because data on DATA_OUT_REVERSE is not passed through the LIFO, it comes out in reverse order. It is immediately followed by the correctly ordered data on DATA_OUT_DIRECT. In some applications, it might not matter that the data comes out in reverse and DATA_OUT_REVERSE can be used to give extremely low latency. The data for the final block still comes out on DATA_OUT after the normal latency, indicated by TB_BLOCK_O. This data is identical to the final block data output earlier on DATA_OUT_DIRECT. If latency of the final block is not an issue, then it is simplest to always read the decoded data from the DATA_OUT port and ignore DATA_OUT_REVERSE and DATA_OUT_DIRECT.

Trellis Termination Implementation

The packet starts and finishes in state 0. It is desirable to force the decoder to begin the trellis construction from state 0. Nothing special needs to be done if the previous packet received by the decoder left state 0 as the most likely state. However, this cannot be guaranteed.

Figure 5 shows the BER degradation that occurs when the trellis construction relies on the previous packet ending in state 0 versus *forcing* the new packet to start from state 0. This constraint length 7 example has 32-symbol packets and uses Direct Traceback. The degradation at lower signal-to-noise ratios (SNRs) is approximately 0.5 dB.

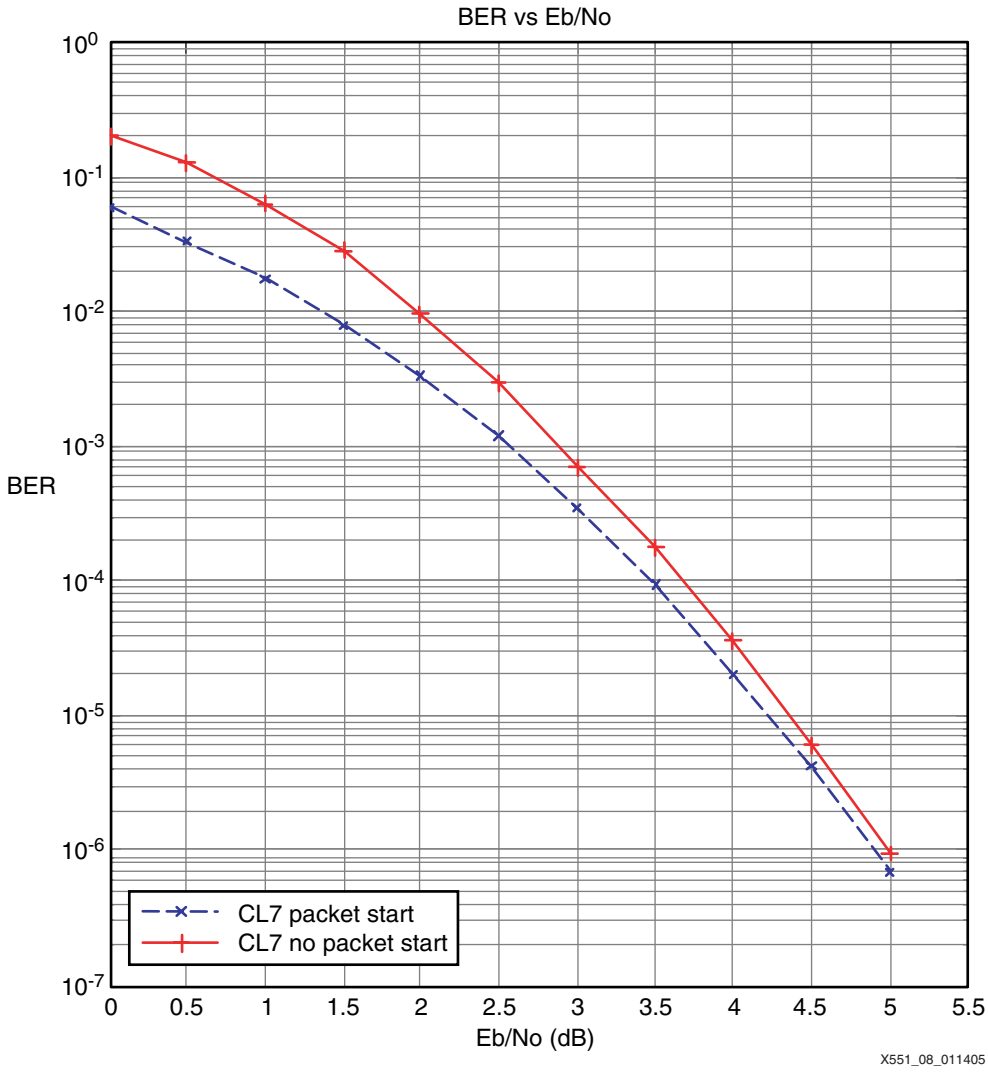

Figure 5: **Effect of Incorrect Trellis Construction in Trellis Termination**

The decoder can be forced to begin trellis construction from state 0 in one of two ways:

1. Feed in a number of strong zeros prior to the first real data in the packet. A minimum of Z zero symbols needs to be sampled to force state 0 to have the lowest cost in the decoder as illustrated in Figure 6. It does not matter where the decoder is in a traceback block when the zeros are fed in. The zeros between the packets still force the decoder to the correct state prior to the start of the real packet data. When these zeros are sampled, the core's BLOCK_IN input can be asserted. BLOCK_OUT then goes to a logic High when the data corresponding to the input zeros is output. This signal can be used as a flag to ignore the output data. BLOCK_OUT is simply BLOCK_IN delayed by the decoding latency.

   It is not necessary to insert zeros prior to the first packet after a reset as construction of the first trellis always begins from state 0.

   One advantage of this method is that it also takes care of forcing the traceback to state 0 at the end of a packet. However, there are other ways of doing this task, as described on page 8.

2. Use the PACKET_START input to force the trellis construction to start at state 0 or any other state as defined by the PS_STATE input. This input forces the cost for the start state to be very low and the cost of all the other states to be very high. The advantage of this method is that it does not waste any clock cycles while dummy zeros are sampled, as in Method 1. It also avoids the awkward timing and control logic that might be required to insert dummy zeros. Refer to Figure 4, page 5 for the PACKET_START timing.

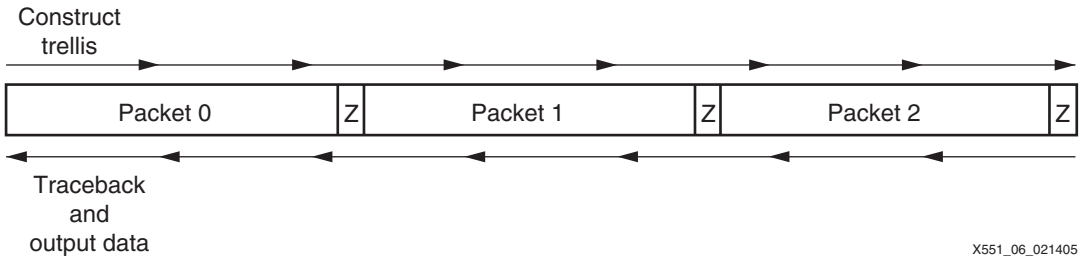

Figure 6: **Forcing Trellis Construction Start State to Zero by Zero Insertion**

As mentioned at the beginning of this section, it is possible to decode normally without forcing state 0 to have the lowest cost at the start of a packet. This might cause a BER degradation due to the reduced likelihood of correcting an error at the start of a packet. The degradation is noticeable if the channel is noisy when the tail bits for the previous packet are received and contain errors. Then it is likely that state 0 does not have a significantly lower cost than the other states as trellis construction for the next packet begins.

Having taken care of the zero forcing at the start of trellis construction, you must ensure that the traceback begins at state 0. The decoder begins traceback from the state with the lowest cost at the end of the training sequence traceback. If this state can be guaranteed to be zero, then nothing needs to be done. This will be the case when the next packet begins from state 0 and is decoded without error, because the first traceback block of the next packet forms the training sequence for the last traceback block of the current packet. For example, if the last block of Packet 0 in Figure 6 is being decoded, the first block of Packet 1 is used as the training sequence. If the traceback through this first block ends in state 0 then the starting state for the traceback through the last block of Packet 0 will be state 0. Typically this sequence cannot be guaranteed and the traceback must be forced to start from state 0.

Figure 7 shows the BER degradation that occurs when the traceback relies on the next packet trellis starting in state 0 versus *forcing* the traceback to start from state 0. This constraint length 7 example has 96-symbol packets. The degradation at lower SNRs is approximately 0.25 dB.

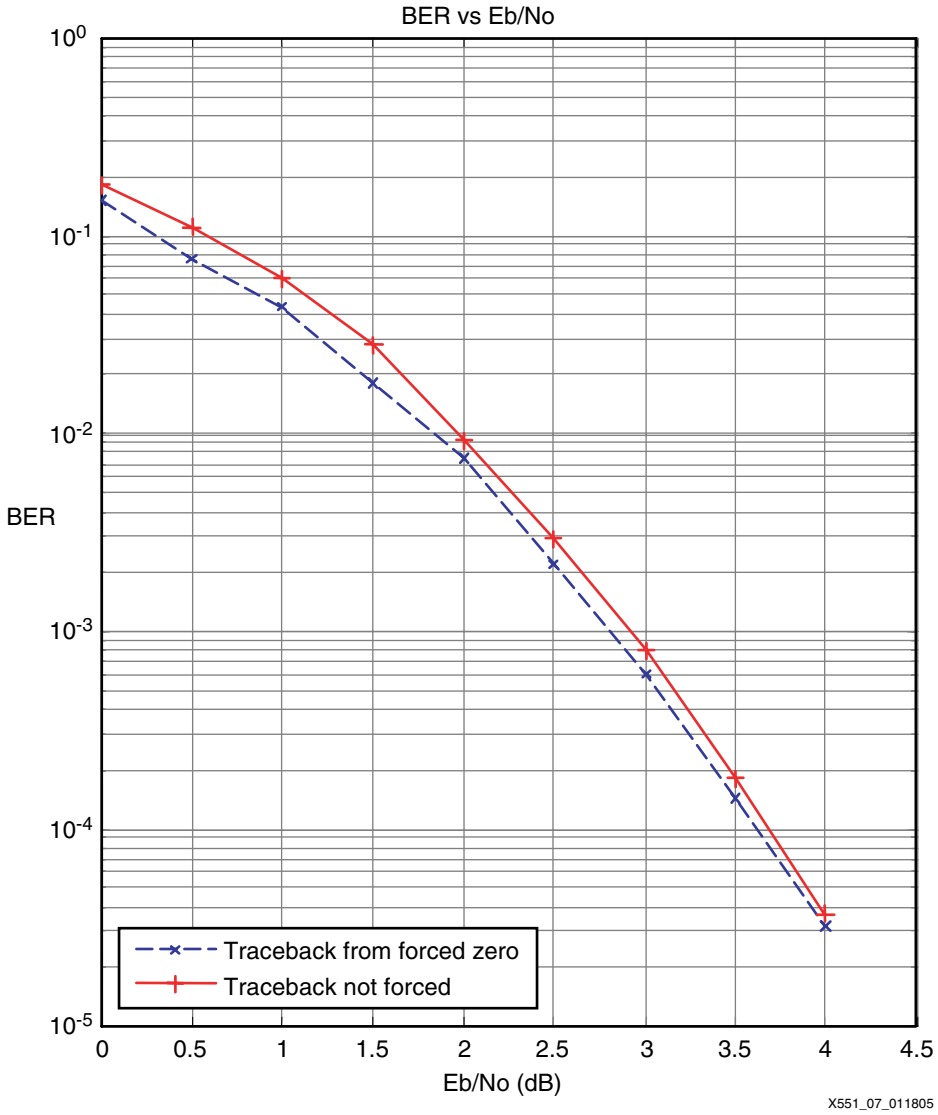

Figure 7: **Effect of Incorrect Traceback Start State in Trellis Termination**

The traceback of a packet can be forced to start from state 0 in one of two ways:

1. Strong zeros can be inserted between packets in exactly the same way as in the trellis construction method described in Method 1 on page 7. Again, as long as Z or more zeros are inserted, traceback can be guaranteed to begin from state 0. As shown in Figure 6, the zeros do not have to align with the start of the traceback block. All that matters is that they immediately follow a packet.

   This technique assumes that the dummy zeros are immediately followed by the next packet. If this is the last packet and there is no more data, more dummy input symbols (any value) should be sampled to flush out the decoder traceback pipeline.

2. Direct Traceback from state 0 can be used for the last block in the packet.

   If latency is important, then the processing delay on the last block can be reduced by selecting this feature. This does away with the training sequence for the last block and forces the traceback to begin from state 0, saving traceback length clock cycles.

   This timing is described in "Trellis Truncation Implementation," page 4 and is shown in Figure 4, page 5. The Direct Traceback is identical in this case, with the exception that we are tracing back from state 0 rather than the Best State.

   For punctured data, the best possible BER results are obtained if the training sequences for all the other blocks in the packet are started from the Best State. See the Viterbi Decoder LogiCORE data sheet for more details on when to use the Best State logic. It is possible to trace back from the best state for these blocks while still tracing back from state 0 for the last block.

   There is no BER degradation with Direct Traceback compared to using a training sequence, because we know for certain that the traceback starts from state 0.

   It is also possible to perform Direct Traceback from a non-zero, user-defined state. In this case, the state value is sampled on the TB_STATE input at the same time as the last data symbol of the last TB block is sampled.

Figure 8 shows an example with a single small packet, as can be found in the IEEE802.11a standard. In this example, the packet contains only a single traceback length, and the PACKET_START signal is used to force the trellis construction to begin from state 0. As mentioned above, this can also be achieved by padding with strong zeros prior to the packet. While TB_BLOCK is High, 24 symbols are sampled. When TB_BLOCK is deasserted, the decoder begins traceback from state 0 and outputs the packet data in reverse order on DATA_OUT_REVERSE. This standard requires that the 24-bit header packet is decoded as quickly as possible. Using DATA_OUT_REVERSE is the fastest way to meet this requirement. Data follows in non-reversed order on DATA_OUT_DIRECT and DATA_OUT, as described in the example in Figure 4, page 5.

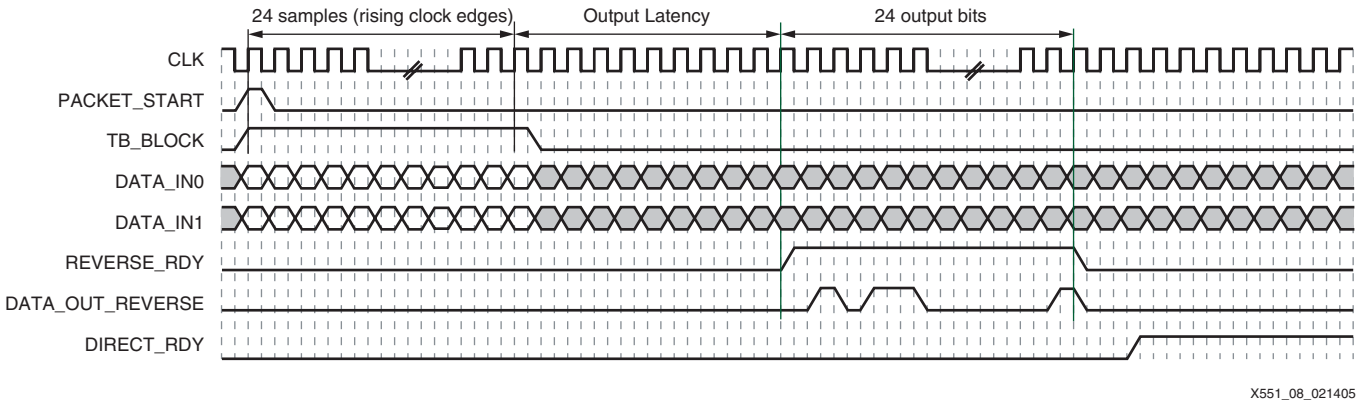

Figure 8: **Trellis Termination with a Single Small Packet**

Tail-Biting Implementation

In most tail-biting applications, the encoder state is initialized to the last Z bits of the data packet prior to any transmission. Thus the encoder has the same starting and ending states for a packet. For optimal decoding, the decoder needs to start the trellis construction from this state. If not done, some BER degradation occurs due to the reduced likelihood of correcting an error at the start of a packet. The IEEE802.16d standard uses this form of tail biting.

If there are several traceback lengths within a packet, one technique is to decode the first TB block at the end of the packet. Assuming the packet terminates in the correct state, then the trellis construction for TB block 0 begins in the correct start state, as illustrated in Figure 9. This example assumes a packet contains (N + 1) traceback lengths.

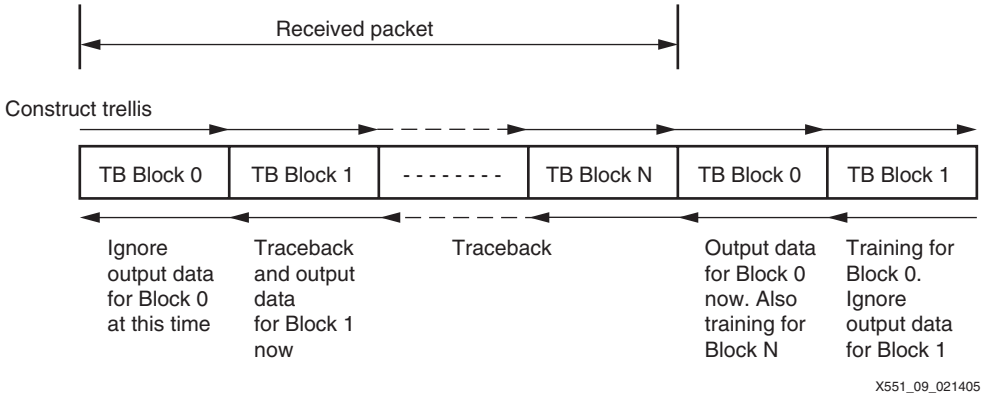

Figure 9: **Decoding for Tail Biting**

TB Block 0 is used at the start of the packet only to ensure the trellis construction for TB Block 1 starts from the correct state. Ignore the output data for TB Block 0 at this time because it might be incorrect due to not knowing from which state to begin the trellis construction. There is no point using the PACKET_START input. If we did know from which state to begin trellis construction, we would use PACKET_START, input the state on PS_STATE, and use the data output for TB Block 0 at this time without re-inserting TB Block 0 at the end of the packet.

The re-insertion of Block 0 at the end also provides the correct training sequence to start the traceback of Block N from the correct state.

This method clearly adds extra decoding latency for Block 0.

The blocks other than the start and end blocks are decoded normally. The end state of one block automatically provides the start state for the next.

Another technique is to input TB Block N first, ignoring the output data. This technique gives the correct start state for the trellis construction of TB Block 0. This block is followed by blocks 0 to N. TB Block 0 is input again at the end to provide a training sequence for decoding TB Block N. The advantage of this technique is that all the data comes out of the decoder in the correct order. The disadvantage is that you have to wait until the entire packet is received before starting to decode.

The core's Direct Traceback feature can be used to perform the decoding without the need to re-insert TB Block 1 at the end, as in Figure 9. This technique is shown in Figure 10.

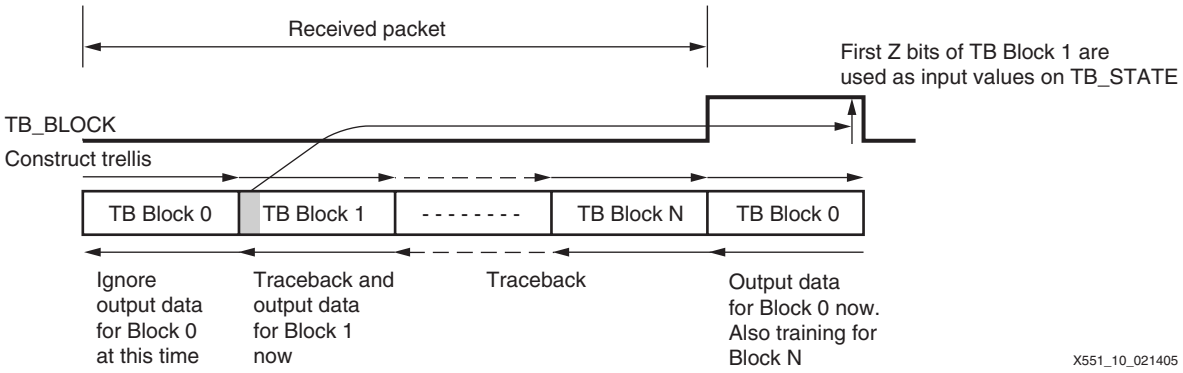

Figure 10: **Decoding for Tail Biting using Direct Traceback**

In this case, TB Block 0 is still used initially to obtain the correct starting state for the trellis construction during TB Block 1 and the first output data for TB Block 0 is ignored. When the data for TB Block 1 appears on DATA_OUT, the first Z bits can be saved in a register, giving the correct traceback start state to begin the decoding of TB Block 0. Thus TB Block 1 does not need to be re-inserted at the end of the packet to determine the start state for the TB Block 0

traceback. Simply apply the saved start state on the TB_STATE input at the same time that the last symbol of TB BLOCK 0 is sampled on DATA_IN (the timing is shown in Figure 11). In this example, the constraint length is 7 and Z is 6. The first six bits of TB Block 1 are decoded as 110000 binary, giving the state the encoder shift register was in immediately after TB Block 0 and hence the correct state with which to begin the Direct Traceback of TB Block 0. Input a value of 3 (000011 binary) on TB_STATE at the end of the TB_BLOCK pulse. The traceback of TB Block 0 then begins from the correct state (3) without requiring a training sequence. Not all clock cycles are shown in Figure 11. In reality, the blocks and latencies are longer than shown.

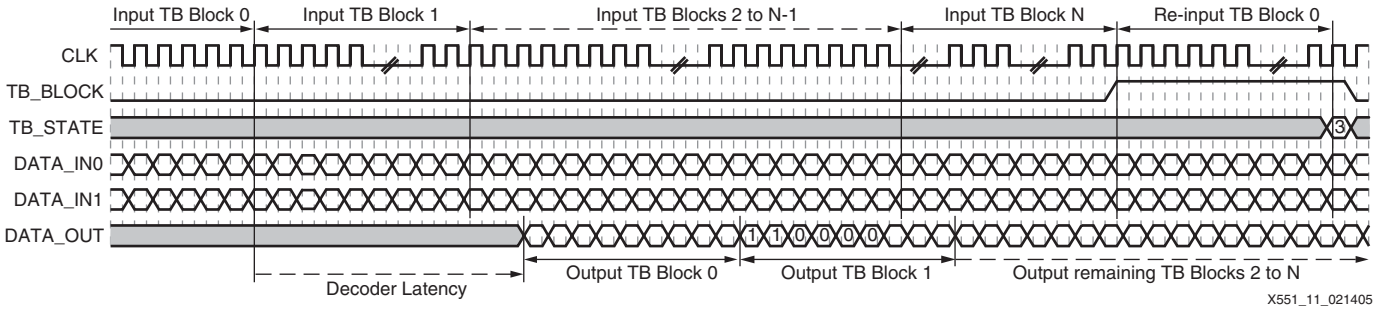

Figure 11: **Timing for Direct Traceback Tail-Biting Technique**

This technique relies on being able to decode the first Z bits of TB Block 1 before the state value is required for the TB_STATE input. There might not be sufficient time to perform this operation if there is only a small number of TB Blocks in a packet. In this case, the method shown in Figure 9, page 10 can be used.

If the packet contains only a single traceback length, then the above technique is effectively the same as passing Block 0 through the decoder three times: once to ensure the correct start state for the trellis construction, once to construct the trellis, and once to perform the correct training so the traceback begins from the correct state (see Figure 12). The Direct Traceback technique cannot be used in this case, because you cannot obtain the traceback start state to apply to TB_STATE.

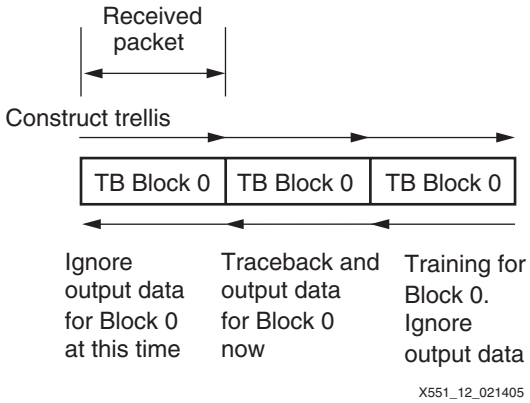

Figure 12: **Single Block Packet**

Block 0 *must* be passed through three times. If it passes through only twice, BER degradation is substantial, resulting in almost all packets decoding incorrectly. Figure 13 shows a small 32-symbol, constraint length 3 example. The curves for tail biting, implemented by passing the block through the decoder three times and a standard continuous stream (non-tail biting) decoder are almost identical. The curve obtained when the block is only passed through twice shows considerably worse BER performance. For larger constraint lengths, this performance is even worse because the likelihood of choosing the correct starting state by chance is reduced.

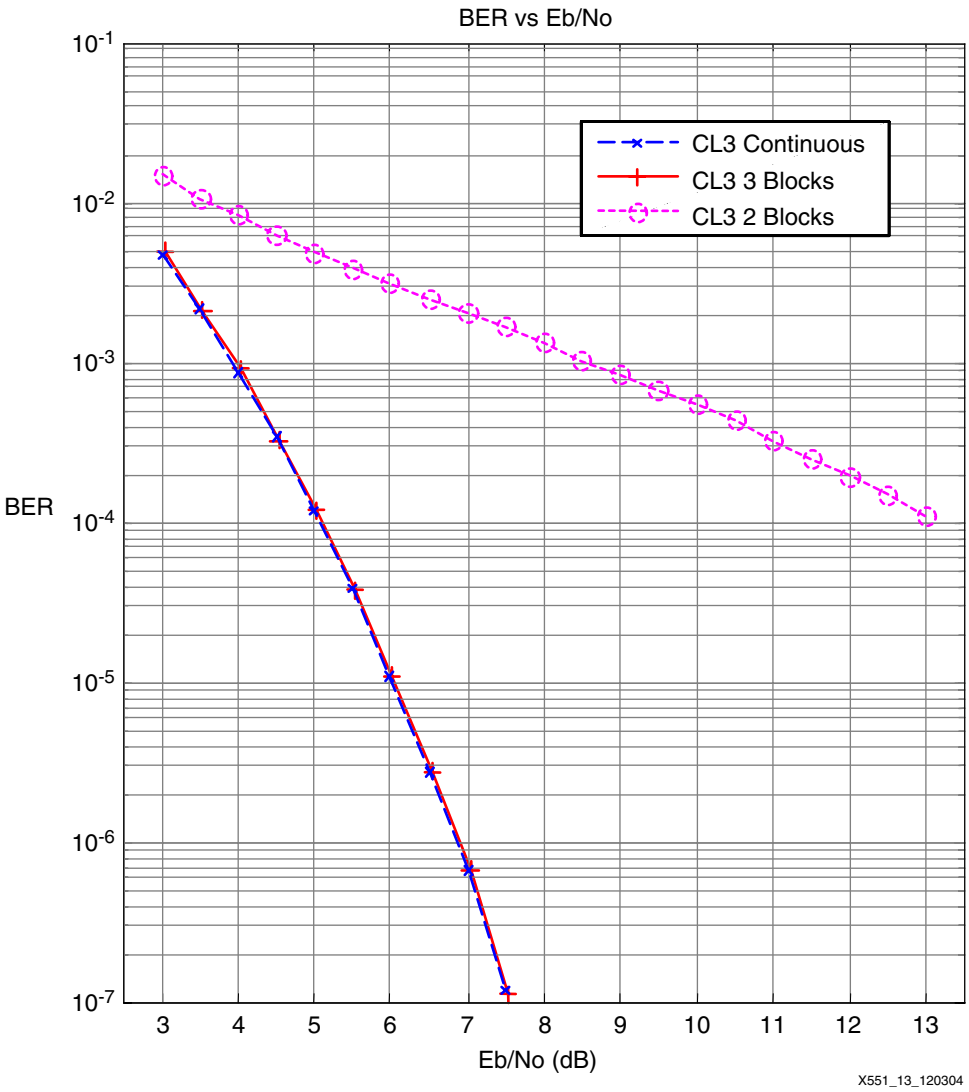

Figure 13: **Effect of Incorrect Trellis Construction in Tail Biting**

Conclusion

This application note describes how trellis termination and tail biting can be implemented using the Xilinx Viterbi Decoder LogiCORE module (v5.0 or later).

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/14/05	1.0	Initial Xilinx release.