TeleBench™ Version 1.1

Benchmark Name: Convolutional Encoder

Highlights

- **Benchmark encodes data for forward error correction, as seen in wireless communication systems.**
- **16 bit & 8 bit integer math and logic.**
- **Multiple data sets (3)**
- **Fits inside small L1 caches to allow focus on CPU-centric performance.**

Application

This benchmark performs a generic Convolutional Encoder algorithm.

Convolutional Encoding adds redundancy to a transmitted electromagnetic signal to support forward error correction at the receiver. A transmitted electromagnetic signal in a noisy environment can generate random bit errors on reception. By combining Convolutional Encoding at the transmitter with Viterbi Decoding at the receiver, these transmission errors can be corrected at the receiver, without requesting a retransmission.

This benchmark provides an indication of the potential performance of a microprocessor , when used to generate convolutional codes as used in forward error correction.

Benchmark Description

The Convolutional Encoding benchmark provides a generic algorithm for producing a sequence of BranchWords from DataByteSize number of serial input DataBits. The algorithm is generic because generating polynomials are passed parameters from the EEMBC Test Harness. The characteristics of the generating polynomials are unique for each data set, and are controlled by NumberCodeVectors, ConstraintLength, and CodeMatrix.

NumberCodeVectors indicates the number of generating polynomials. ConstraintLength is equal to one plus the number of delayed DataBit values required for the generating polynomials. CodeMatrix is an array of size ConstraintLength by NumberCodeVectors. The values in a column of the code matrix (zeros or ones) correspond to the current and delayed DataBit values, indicating which terms are present in the generating polynomial.

By using generating polynomials that are functions of current and previous input DataBits, the Convolutional Encoder generates a number of output BranchWords per DataBit equal to the NumberCodeVectors.

The EEMBC Test Harness can request one of the three generating polynomials listed below. In these equations, the notation “D4”, for example, means “the DataBit that occurred four bits prior to the current DataBit.” G0 and G1 are the output BranchWords. The “+” operation is implemented as a bitwise exclusive OR in the benchmark.

Generating Polynomials:

- Test case **xk5r2dt** -- ConstraintLength=5, NumberCodeVectors=2
  G0 = 1+D2+D3+D4 (octal 27)
  G1 = 1+D+D4 (octal 31)

- Test case **xk4r2dt** -- ConstrantLength=4, NumberCodeVectors=2
  G0 = 1+D1+D2+D3 (octal 17)
  G1 = 1+D2+D3 (octal 13)
- Test case **xk3r2dt** -- ConstraintLength=3, NumberCodeVectors=2
  G0 = 1+D1+D2 (octal 7)
  G1 = 1+D2 (octal 5)

Analysis of Computing Resources

The Convolutional Encoder performs 16-bit signed & 8-bit unsigned operations, bitwise exclusive-OR operations, and bytewise shifts. This benchmark comprises 20 lines of executable C-code. Data sets use a maximum of 512 DataBits per iteration.

Special Notes

1. All three convolutional encoder data sets must be run to obtain an EEMBC Telemark™ score.