Application Report
SPRABJ0—February 2011

Tuning VCP2 and TCP2 Bit Error Rate Performance

High-Performance and Multicore Processors

Jane Lu & Brighton Feng

Abstract

In most customer applications, a high level of decoding bit error rate (BER) performance is required. Since Convolutional codes and Turbo codes are widely used in wireless communication systems, TI DSPs integrate two high-performance embedded coprocessors (enhanced Viterbi decoder coprocessor and enhanced Turbo decoder coprocessor) that significantly speed up channel-decoding operations on-chip.

This application note presents the strategies and programming methodology to optimize VCP2 and TCP2 BER performance on TI DSPs. It is assumed the reader is familiar with Convolutional and Turbo coding theory.

Contents

List of Tables

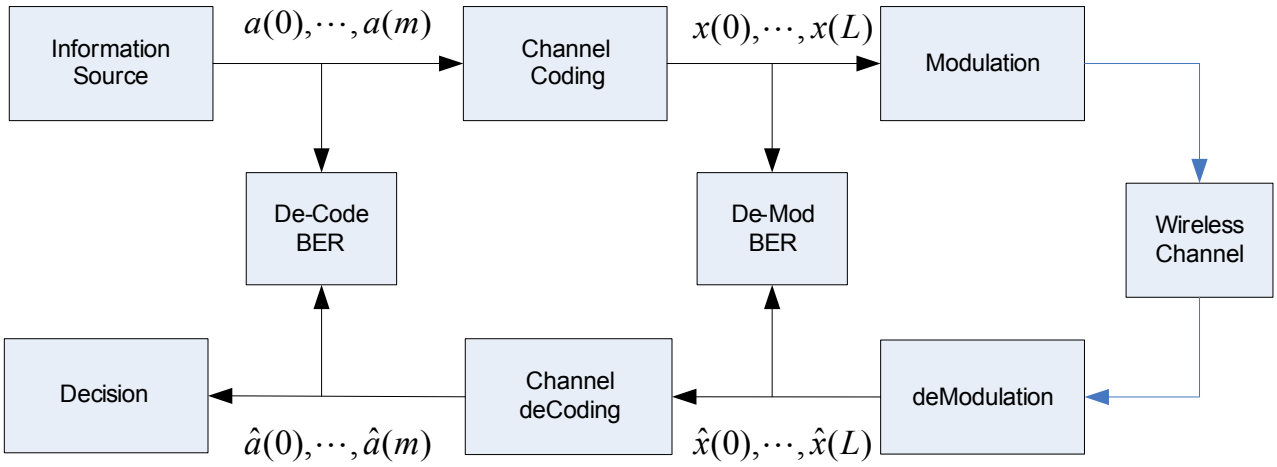
List of Figures

1 Introduction

Forward-error correction (FEC), also known as channel coding, is used to improve the reliability of a channel by adding redundant information to the data being transmitted. Convolutional coding and Turbo coding techniques are used widely in 2G and 3G wireless standards. The enhanced Viterbi coprocessor (VCP2) is a programmable peripheral used to decode convolutional codes. The enhanced Turbo coprocessor (TCP2) is a programmable peripheral used to decode Turbo codes. They are integrated into some Texas Instruments digital signal processors; refer to the particular device data sheet for compatibility information.

Figure 1 shows a simplified communication system model. As illustrated, the receiver's physical layer bit error performance is a system-level issue that includes both demodulation performance and decoding performance. The decoding performance is dependent on the demodulation output (performance). The theory of communication systems is not discussed here and the reader can find many materials in IEEE and other publications if interested. This document presents the strategies and programming methodology to make full use of the decoder coprocessors to meet the system BER performance goal.

Figure 1 BER Model of Communication System



2 VCP2 Decoder Performance

The Enhanced Viterbi Coprocessor (VCP2) is a programmable peripheral used to decode convolutional codes. The inputs into the coprocessor are 8-bit signed branch metrics, obtained by combining channel soft decisions. The outputs can be bit-packed hard decisions or 8-bit soft decisions. Communication between the DSP and the VCP2 is performed through a high performance DMA engine, the enhanced DMA (EDMA3). The user can refer to the VCP2 User Guide (SPRUE09) for detailed programming implementation.

VCP2 programmable parameters are as follows:

1. Constraint length K (5, 6, 7, 8, or 9)
2. Code rate (1/2, 1/3, or 1/4)
3. Polynomials
4. Frame length
5. Traceback mode
6. Initial conditions for state metric computation
7. Reliability length and convergence length
8. Yamamoto bit generation

Parameters 1-4 correspond directly to the parameters used for encoding and cannot be independently tuned at the receiver to obtain more performance margin. Parameters 5-8 are adjustable and may influence VCP2 decoding performance. In addition, the soft input data precision also will impact the decoding performance.

2.1 Traceback (TB) Mode

There are three traceback (TB) modes supported:

- tailed TB mode
- mixed TB mode
- convergent TB mode

For zero tailed convolutional encoding, tailed TB and mixed TB mode are used to decode the whole frame. Traceback operation should start from the zero state. Convergent TB mode can be used when you want to decode one portion of the frame.

For tail-biting convolutional encoding, the convergent TB mode of operation should be applied since there is no known end trellis state.

2.2 Initial Conditions for State Metric Computation

IMAXI, IMAXS, and IMINS are inputs to initialize the trellis prior to the start of executing the state metric unit. IMAXI should be set to the starting state of the encoder. IMAXS is the initial value for the starting state and IMINS is the initial value for all the other states. The difference between IMAXS and IMINS is important for BER performance.

For zero tailed convolutional encoding, the delay line of the convolutional encoder is initialized with all zeros, so the initial state is the zero state. The final state is also driven to the zero state by appending an all-zero byte to the information block. Therefore, it is important to favor the zero state at the very beginning. This is done by setting the IMAXI = 0, which identifies that the zero state is the initial state, and also by setting

IMAXS to a large value (0x400 is recommended) and IMINS to a small value (0 is recommended), which correspond to the starting path metrics values for the favored (zero) state and other states, respectively. Traceback operation starts from the zero state.

For tail-biting convolutional encoding, since there is no known starting state, no state should be favored as an initial state. This is achieved by setting initial path (state) metrics for all states to the same value (IMINS = IMAXS = const; const = 0 is recommended).

2.3 Reliability Length and Convergence Length (R,C)

In the mixed TB mode and convergent TB mode, the frame is split into sliding windows. The VCP generates R decisions per each sliding window which has length of (R+C). The output decision is more reliable with larger C and R+C. The max (R+C) and C value is dependent on constraint length K and whether hard or soft output decisions are configured. (Refer to the VCP2 User Guide SPRUE09 for detailed information.) The choice of R and C will not largely affect the VCP processing delay since a larger portion of cycles is spent in state metric accumulation.

2.4 Yamamoto Bit Generation

The Yamamoto bit is used as a binary frame quality indicator. Initially, the Yamamoto bit (YAM bit in VCPOUT1) is set. If anywhere along the decoding path there was a point where the decision between two paths was less than a given threshold (YAMT field in VCPOUT1), the YAM bit yields a zero.

For example, even if the decoder's output has many errors, there is a low probability that it will still pass the CRC check. If the application uses only the CRC check as the frame quality indicator, the system may not be robust. In this case, the application can use the Yamamoto output bit to determine whether the VCP2 output is reliable.

2.5 Soft Input Data Precision

In the preceding section, it is mentioned that the input data quality is important for decoder performance. In particular, limited precision of the soft information coming from the demodulation module may be the performance limiting factor for the whole chain.

The VCP2 is implemented with fixed point processing and the internal state metric accumulation is based on 13-bit precision. The branch metrics can have the maximum input dynamic range of 7 bits + 1 sign bit [-128;+127]. In order to prevent calculation overflow during decoding, the final branch metrics maximum range should be as shown in Table 1. (Refer to the VCP2 User Guide SPRUE09 for detailed information.)

Table 1 VCP2 Soft Input Resolution

1/Rate	K	Data range [R-1, -R]	SoftInputResolution
2	5,6,7,8,9	[+63, -64]	7
3	5,6,7,8,9	[+42, -42]	6
4	5,6,7,8,9	[+31, -32]	6
End of Table 1			

The 16-bit fixed point data type is widely used in the communication demodulation algorithms in the uplink, whereas the 8-bit fixed point data type is used for VCP2 input branch matrix data. There is data scaling from 16-bit to 8-bit and the 8-bit soft input should meet the data range requirements in Table 1. In this section, three solutions are introduced and the related VCP2 BER simulation is performed on a typical use-case (AMR12.2 Kbps class B). The VCP2 BER simulation uses the same methodology introduced in SPRA794 [3.] However, QPSK modulation and 16-bit precision are applied instead.

Assume the decoder input sequence is: $\hat{x}(0), \hat{x}(1), \cdots \hat{x}(2N)$, where 2N is the frame length. The mean of this sequence is $\bar{x}$ and the variance is σ^2. The maximum value is $x_{\max}$. The 16-bit precision data and 8-bit precision data is defined separately as shown below:

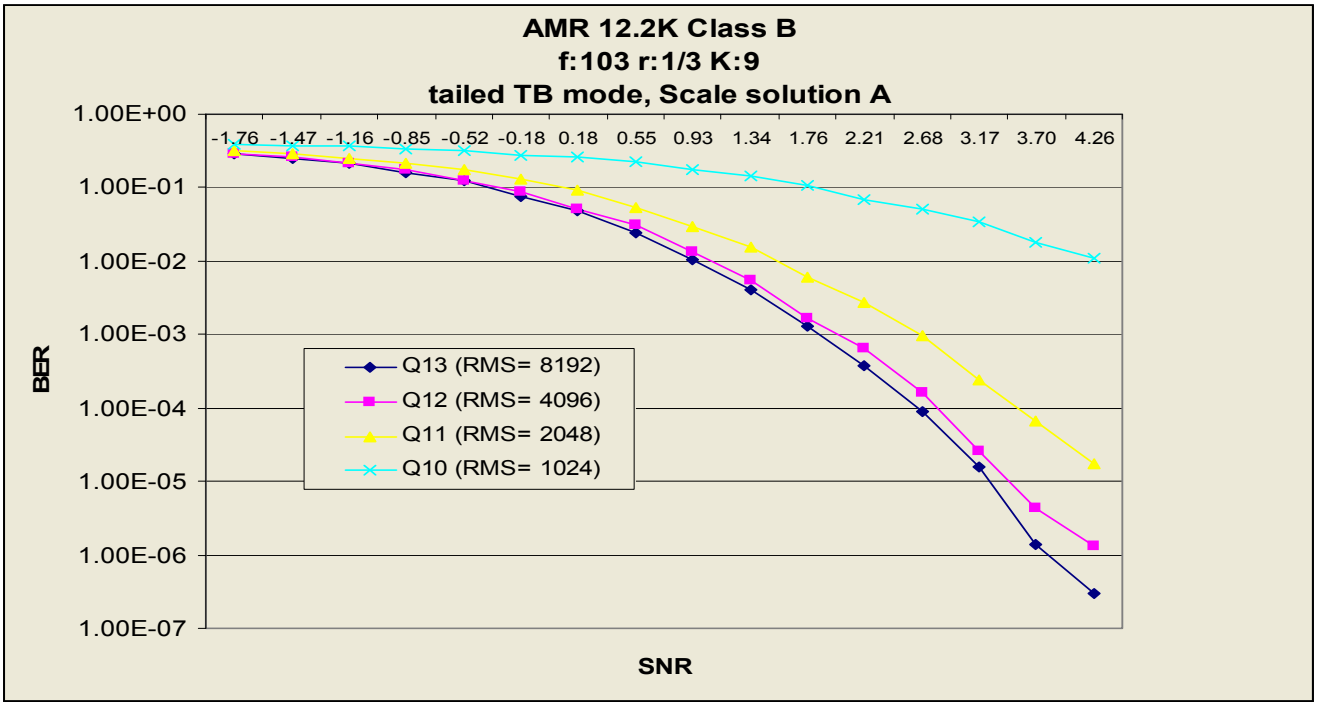
```
Short deModData[2N];
Byte  softInput[2N];
```

The simplest scaling method, defined as solution A, directly extracts the upper resolution bits from the 16-bit data. However, this solution may lose too much precision and degrade the bit error performance. The following pseudo code shows the basic procedure. The **softInputResolution** variable is the number of bits used to represent an input value and is determined by Table 1.

```
Int *in, *out;
Int temp1,temp2, shr_amnt;

In = (int *)deModData;
Out = (int *)softInput;

shr_amnt = 16 - softInputResolution;
    for(i=0; i< N; i+=2, j++)
    {
        temp1 = _shr2(in[i], shr_amnt);
        temp2 = _shr2(in[i+1], shr_amnt);
        out[j] = _packl4(temp1, temp2);
    }
```

Figure 2 shows the AMR 12.2 Kbps class B BER performance in an AWGN channel based on solution A scaling. The average RMS (Root Mean Square) per QPSK symbol is changed from 1024 to 8192 and it is shown that the RMS with Q13 has the better performance than does the RMS with Q10. Here, Q13 means the average RMS per QPSK symbol is 2^13, Q12 means 2^12, and so on.

Figure 2 BER Performance of Scale Solution A



In most applications, it is hard to guarantee that the demodulation output is distributed in a specific RMS range after complex demodulation algorithm processing. An improved method, defined as solution B, is to normalize the input soft bits using the maximum value x_{max}. The procedure steps are as follows:

1. Search for the maximum value x_{max}
2. Calculate the scale factor R / x_{max} and normalize the input soft bits

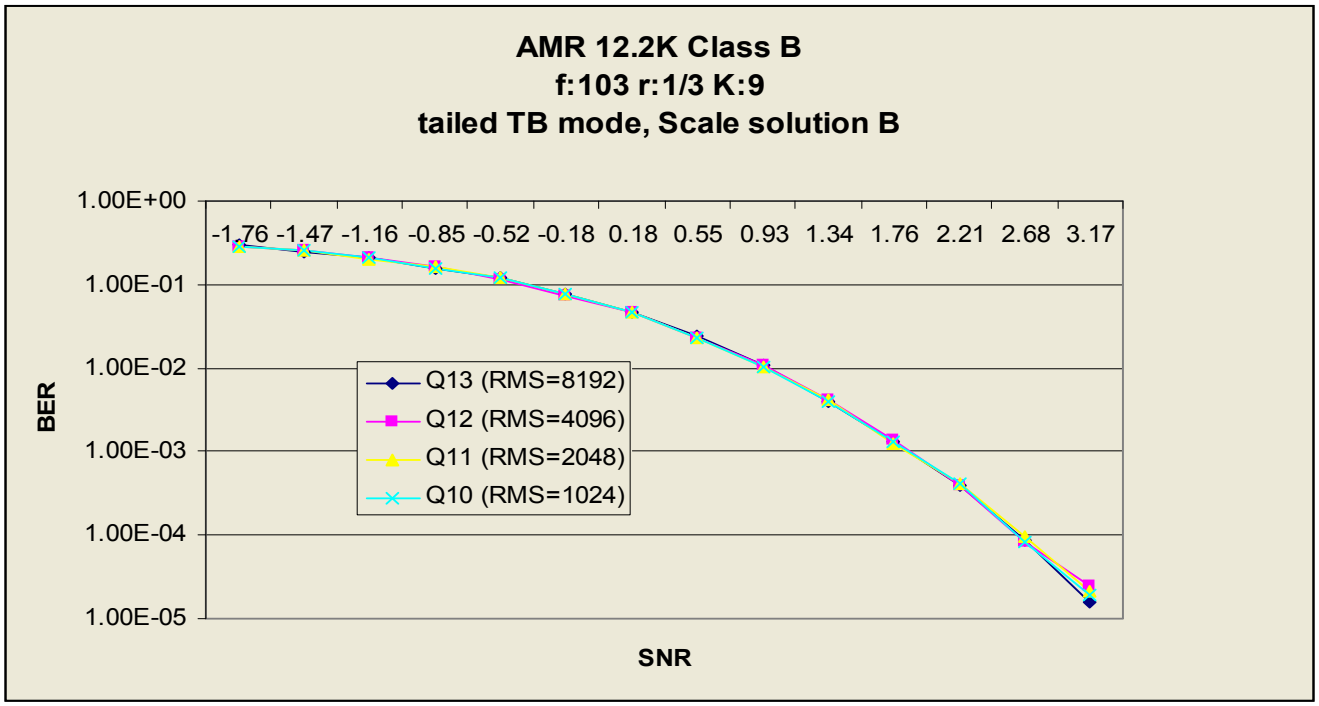
An efficient option is using **_norm()** to calculate the minimum signed bits and then scale the sequence. The pseudo code is shown below. The **softInputResolution** variable is the number of bits used to represent an input value and determined by Table 1.

```
Int *in, *out;
Int temp1,temp2,shr_amnt,minBits;
long long temp;
Int temp_a1, temp_a2;

In = (int *)deModData;
Out = (int *)softInput;

temp1 = 0x7fff7fff;
    for (i=0; i<N; i++)
        {
          temp = _mpy2ll(in[i], 0x00010001);     // Convert 16-bit data to 32-bit
                                                 // for _norm intrinsic
          temp_a1 = _norm(_hill(temp));
          temp_a2 = _norm(_loll(temp));
          temp1 = _min2(temp1, _pack2(temp_a1, temp_a2));
        }
minBits =( _min2(temp1, _packlh2(temp1,temp1)))&0xffff;

/* calculate the saturation level */
shr_amnt = 32- minBits - softInputResolution;

    for(i=0; i< N; i+=2, j++)
    {
```

```
    temp1 = _shr2(in[i], shr_amnt);
    temp2 = _shr2(in[i+1], shr_amnt);
    out[j] = _packl4(temp1, temp2);
}
```

Figure 3 shows the solution B scaling BER performance. There is now no obvious difference among the different RMS values.

Figure 3 BER Performance of Scale Solution B



Since the data normalization is based on the maximum data, this solution also has a precision limiting issue when the data is coupled with large noise and the average data is much less than x_{max}. The performance degrades especially with the low SNR in a fading channel.

A third method, referred to as solution C, is introduced to compensate for the large noise burst case. The idea is to calculate the average number of sign bits for the input data to determine a reasonable normalization value. For example, for a 32-bit data value, there are 13 sign bits for 0x00077777 and 13 sign bits for 0xFFFA7777. Assuming the number of sign bits for a 32-bit input value is s, one option is to choose the normalization value as 2^(32 + K - $\bar{x}$), where K is an experimental value typically in the range of {2–4} and $\bar{x}$ is the average number of sign bits s. The intrinsic **_norm( )** can be used to calculate s as an efficient option. The pseudo code is shown below. The **softInputResolution** variable is the number of bits used to represent an input value and is determined by Table 1.

```
Int *in, *out;
Int temp1,temp2,shr_amnt,minBits;
long long temp;
Int temp_a1, temp_a2;
int  maxpos;
unsigned int maxpos2;
unsigned int maxneg2;
Int tempA, tempB;

In = (int *)deModData;
```

```
Out = (int *)softInput;

for (i=0; i< N; i++)
{
    temp = _mpy2ll(in[i], 0x00010001);  // Convert 16-bit data to 32-bit
                                        // for _norm intrinsic
    temp_a1 += _norm(_hill(temp));
    temp_a2 += _norm(_loll(temp));
}
    /* calculate the average value and add 2bit margin */
    average = ((temp_a1 +temp_a2 + (N<<1) -1)/(N<<1));
    average = (average >2)? (average-2):0;

    /* calculate the saturation level */
    shr_amnt = 32-average - softInputResolution;

    maxpos = (1<<(32-average-1))-1;
    maxpos2 = _pack2(maxpos,maxpos);
    maxneg2 = _sub2(0,maxpos2);

        for(i=0; i< N; i+=2, j++)
        {
            tempA = _min2(in[i], maxpos2);
            tempB = _min2(in[i+1], maxpos2);
            temp1 = _shr2(_max2(tempA,maxneg2), shr_amnt);
            temp2 = _shr2(_max2(tempB,maxneg2), shr_amnt);
            out[j] = _packl4(temp1, temp2);
        }
```

Figure 4 shows the solution C scaling BER performance. There is no obvious difference among the different RMS values. Figure 5 shows the BER performance comparison between solution B and C with Q13. The performance difference between solution B and solution C is small since the corrupted data is in a normal distribution.

Figure 4 BER Performance of Scale Solution C

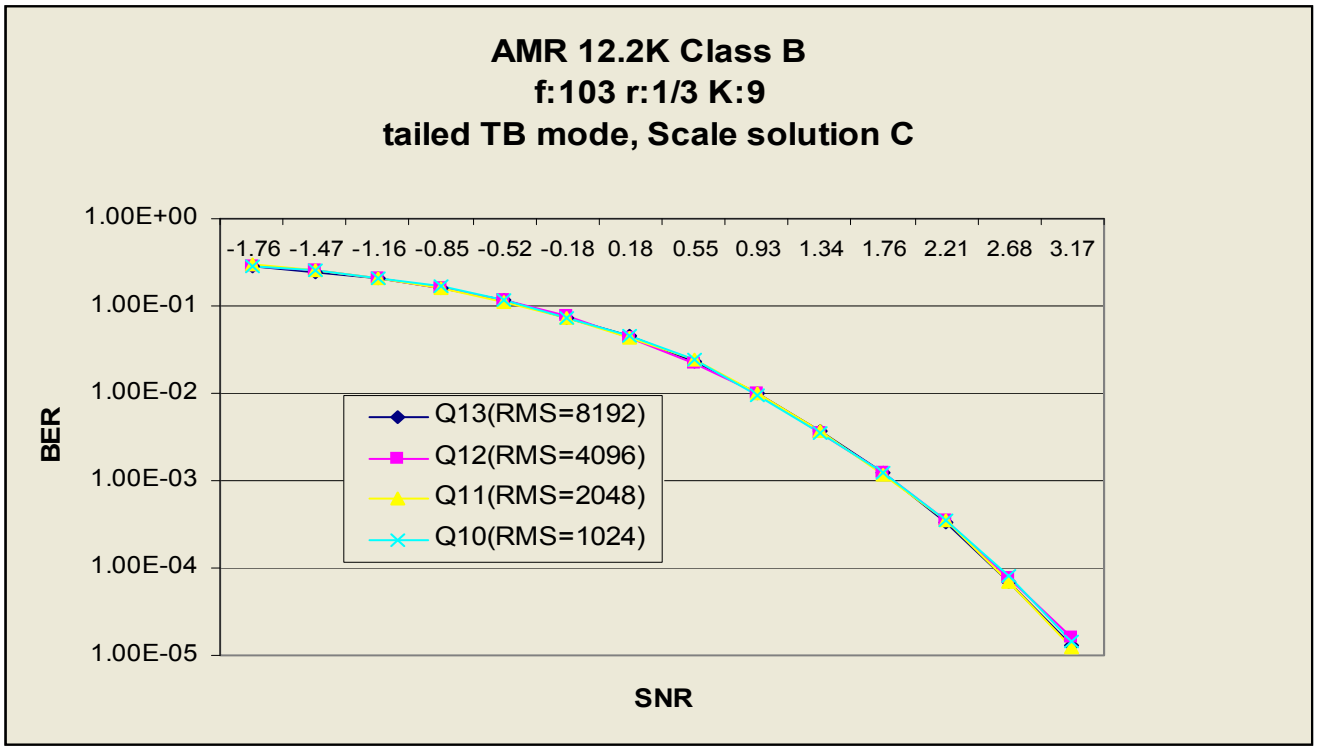
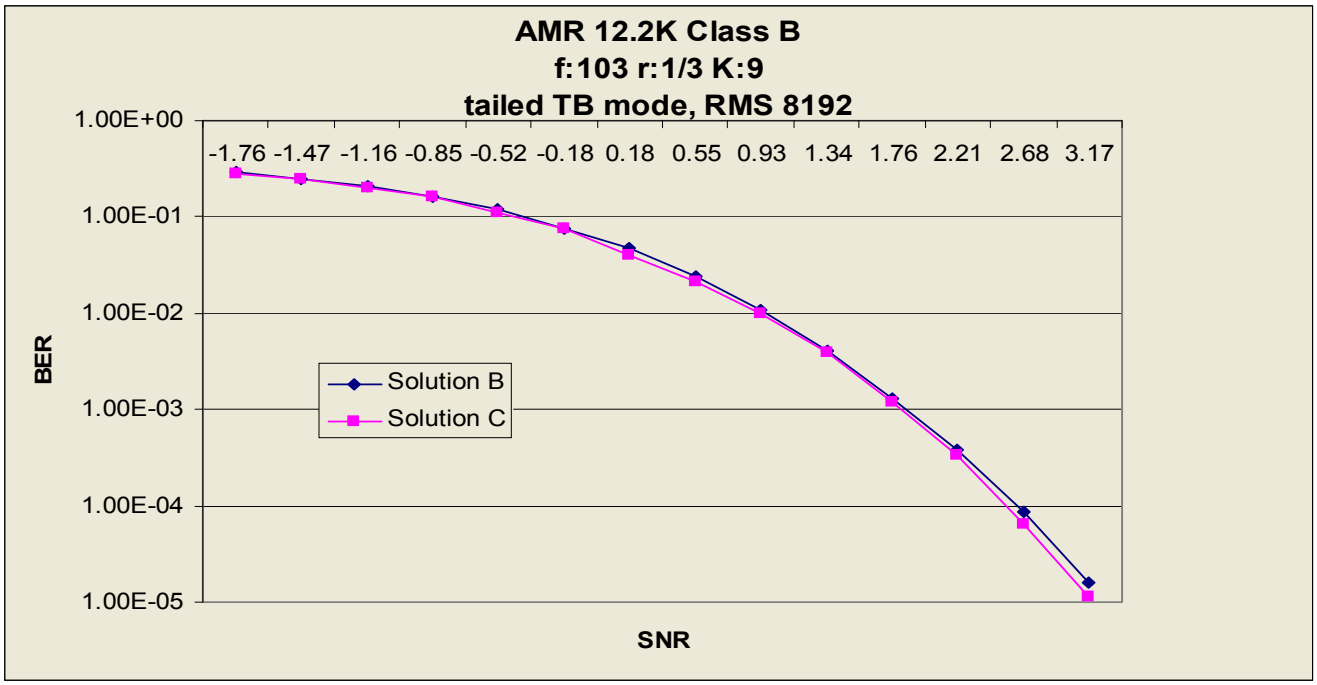


Figure 5 BER Performance Comparison



There is no universal method which always has better BER performance in all decoding systems. The user should perform some investigation to decide which method is best for the application case by case.

3 TCP2 Decoder Performance

The Enhanced Turbo Coprocessor (TCP2) is a programmable peripheral for decoding of IS2000/3GPP Turbo codes. In both cases, 8-state constituent encoders are used. The differences are in code rate, puncturing scheme, and handling of tail bits. The coprocessor operates either as a complete Turbo decoder running multiple iterations (stand-alone processing mode), or it can operate as a single maximum a posteriori (MAP) decoder running a single iteration each time it is used (shared processing mode). In the stand-alone processing mode, the inputs into the TCP2 are channel soft decisions for systematic and parity bits and the 16-bit interleaver index table. The outputs are hard decisions. In the shared processing mode, the inputs are channel soft decisions for systematic and parity bits and a priori information for systematic bits, and the outputs are extrinsic information for systematic bits.

The TCP2 programmable parameters are:

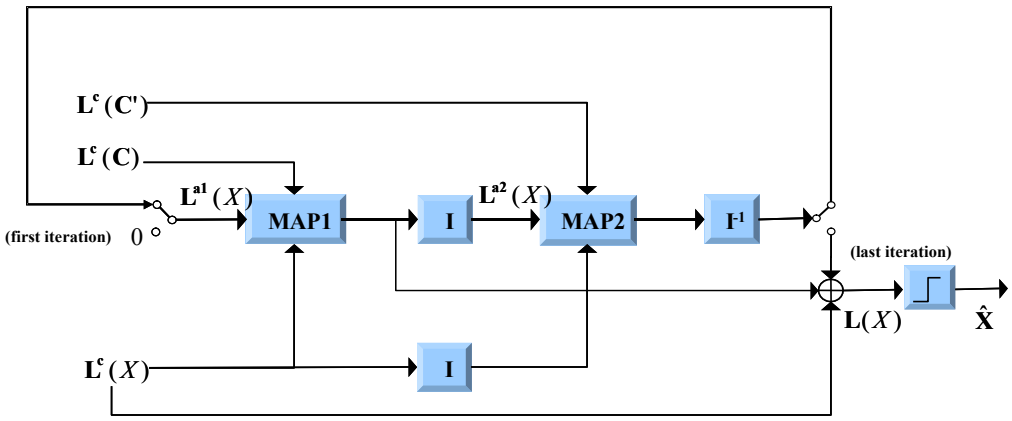
1. Code rate (1/2, 1/3, 1/4 or 1/5)
2. Frame length
3. Decoding algorithm: max*-log-MAP and max-log-MAP
4. Decoding mode: Standalone (SA) or Shared Processing (SP)
5. Maximum number of iterations and early stopping rules

Parameters 1–2 correspond directly to the parameters used for encoding and cannot be independently tuned at the receiver to obtain more performance margin. Parameters 3–5 are adjustable and may influence TCP2 decoding performance.

3.1 Decoding Algorithm

The decoding of Turbo codes is based on an iterative structure constructed from two MAP decoders. The Turbo decoder architecture is shown in Figure 6.

Figure 6 Turbo Decoder Architecture



The algorithm used as a basis for MAP decoding is the Bahl Cocke Jelinek Raviv (BCJR) algorithm first presented in [5.]; it operates in the log-likelihood ratio (LLR) domain which is defined as:

$$L(u) \triangleq \log\left(\frac{P(u=1)}{P(u=0)}\right)$$

Each MAP decoder has three inputs:

$L^c(X)$: the channel LLR for systematic bits

$L^c(C)$ or $L^c(C')$: the channel LLR for parity bits

$L^a(X)$: the prior LLR for systematic bits, i.e., $L^{a1}(X)$ is for MAP1, and $L^{a2}(X)$ is for MAP2.

Each MAP decoder has one output:

$L^e(X)$: the extrinsic LLR for systematic bits

As shown in Figure 6, the prior LLR is calculated by extrinsic LLR. For example, $L^{a1}(X)$ is the prior information for MAP decoder 1, and it is derived from MAP decoder 2 extrinsic information $L^e_{21}(X)$. Similarly, $L^{a2}(X)$ is the prior information for MAP decoder 2, and it is derived from MAP decoder 1 extrinsic information $L^e_{12}(X)$. The $L^{a1}(X)$ is zero assuming $P(u=1)$ equals $P(u=0)$ for the first round iteration.

After n iterations, the $L^e_{21}(X)$ and $L^e_{12}(X)$ tend to be stable and the total LLR of MAP decoder output $L(x_k)$ can be calculated by equation (3-1). The final decision will be made based on the sign of $L(x_k)$, i.e., $\tilde{x}_k = sign[L(x_k)]$.

$$L(x_k) = L^c(x_k) + L^e_{21}(x_k) + L^e_{12}(x_k) \qquad (3\text{-}1)$$

Most of the computation involved in equation (3-1) is based on the logarithm of a sum of exponentials. Such an expression can be exactly computed, two terms at a time, using the Jacobian logarithm [5.]:

$$Max^*(x_1, x_2) = Log(e^{x_1} + e^{x_2}) = Max(x_1, x_2) + Log(1 + e^{-|x_1 - x_2|})$$

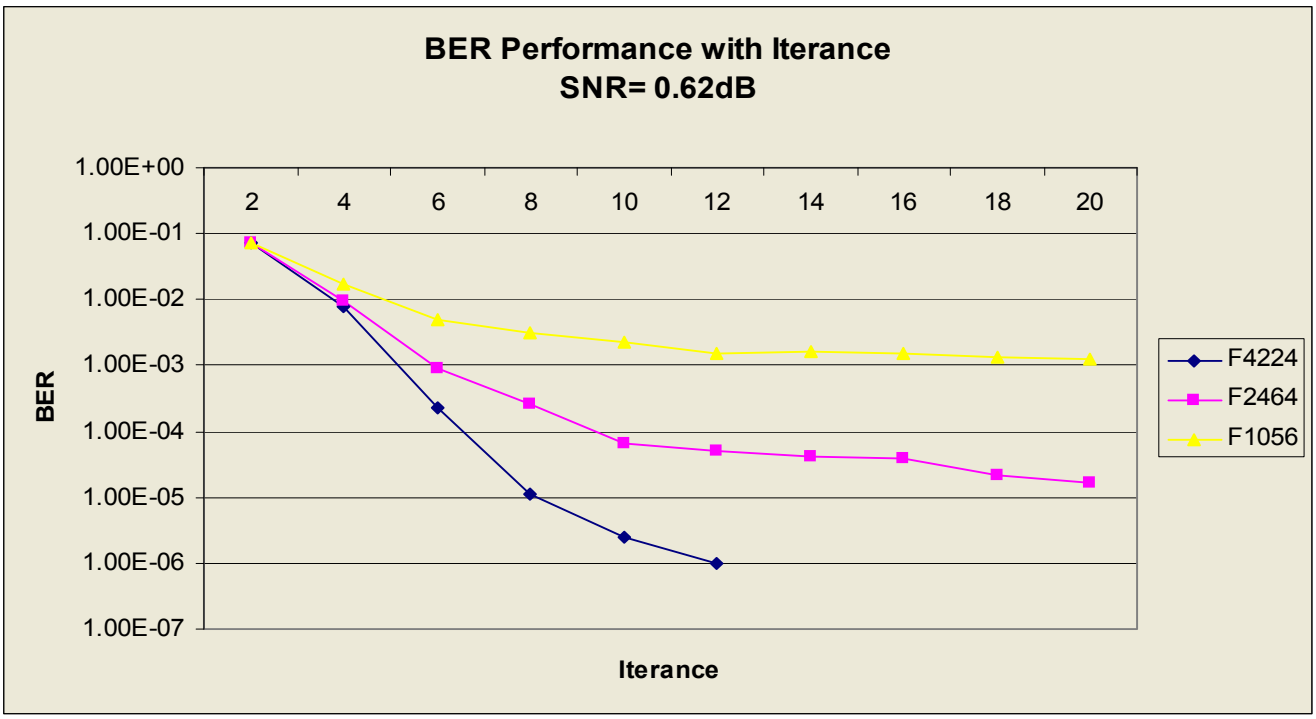
$Log(1 + e^{-|x_1 - x_2|})$ is known as the correction term. If the correction term is omitted and only max term is used to compute $L^e_{12}(X)$ and $L^e_{21}(X)$, we obtain the so-called max-log-MAP approximation. If the correction term is approximated using a small lock-up table, we obtain the max*-log-MAP approximation.

TCP2 offers both max*-log-MAP and max-log-MAP implementations. This can be selected on a frame-by-frame basis. The max-log-MAP implementation does not require the input LLRs (log-likelihood ratios) to scale with a factor $-\frac{2E_s}{\sigma^2}$ which is inversely proportional to noise variance, and is therefore more robust in situations where SNR cannot be accurately estimated[6.][7.]. Normally, the $-\frac{2E_s}{\sigma^2}$ value can be gotten either by calculating training sequence or by using some statistic measurement in the communication system. The method of $-\frac{2E_s}{\sigma^2}$ calculation is not covered in this document.

There are 16 extrinsic scaling factors that can be used for the max-log-MAP algorithm. These factors are defined in TCPIC12~TCPIC15 registers and the value can range from 0.0–1.0. As far as what is the best extrinsic scaling factor to configure, it will actually vary based on the variance. The experienced value is in the range of 0.7–0.75. It is recommended that the user evaluate the value based on the application.

The BER performance of both max*-log-MAP and max-log-MAP on TCP2 are shown in Figure 7. The testing was based on an AWGN channel. SNR stands for the theoretical value $10Log\left(\frac{E_s}{2\sigma^2}\right)$, and the estimated_SNR means the SNR value calculated by estimating E_s and σ^2. The test result shows the max*-log-MAP performance with the theoretical SNR value has the best performance. But max*-log-MAP performance becomes worse if the SNR estimation is not accurate. Figure 7 also shows max-log-MAP has better performance than Max*-log-Map when the gap between SNR and SNR_estimation is 1.2db. It is recommended that max-log-MAP be used if the SNR estimation is not very accurate.

Figure 7 Turbo Decoder Algorithm Performance Comparison



3.2 Decoding Mode

The TCP2 has two fundamental modes: standalone (SA) and shared processing (SP).

In SA mode, the TCP2 iterates a given number of times and outputs hard decisions. SA mode is typically used for frame sizes smaller or equal to 20730.

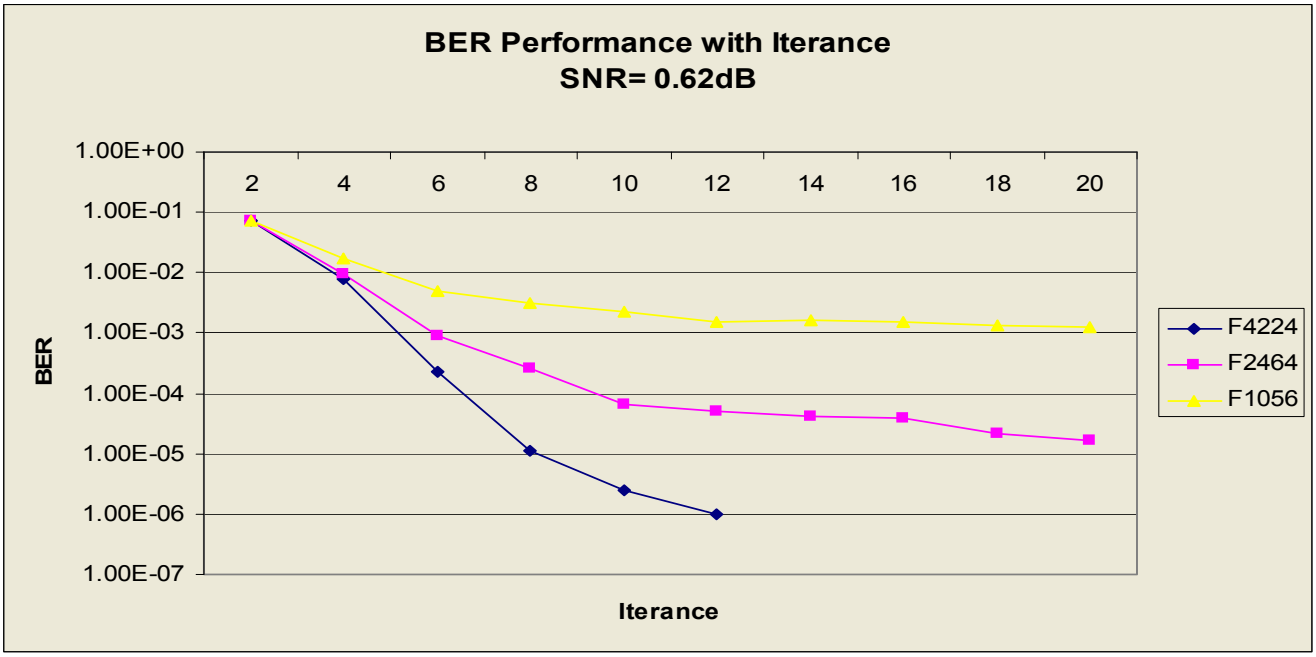
In SP mode, the TCP2 executes a single MAP decode and outputs extrinsic information, which is a (5,2) number, that is, SIIII.FF (where S = sign bit, I = integer, F = fractional bit) and is right-justified with 0 in the most-significant-bit position. The DSP core must perform the input data de-multiplexing, interleaving, de-interleaving operations, hard decision calculation, and any stopping criteria algorithm. SP mode must be used for frames strictly larger than 20730.

3.3 Iterations Rules

Turbo decoders are iterative decoders. The number of iterations is either a deterministic number, or it depends on a test performed on the turbo decoder output after each iteration.

In the first case, the deterministic number can be up to 32 iterations. Figure 8 shows the BER performance with different numbers of iterations. The three curves shown are for frame lengths of 4224, 2464, and 1056. The plot shows that the BER performance increases with number of iterations up to a point.

Figure 8 Turbo Decoder Algorithm Performance Comparison



In the second case, there are early stopping criteria: CRC passed or SNR threshold passed. In this case, the boundary conditions can be programmed, that is, the minimum and the maximum number of iterations that should be performed.

3.4 Soft Input Data Precision

Since the TCP2 decoder is a fixed-point accelerator, there are two kinds of scaling for the input data preparation:

1. SNR scaling. It is the nature of MAX*-LOG-MAP algorithm to scale input LLR. It is needed for either fixed point processing or float point processing.
2. Precision Scaling only for fixed point processing, this scaling is to make sure the data precision meets the requirements and better decoder performance can be achieved.

The TCP2 input data corresponds to channel log-likelihood ratios (LLRs) that should be in 6-bit precision. For the MAX-LOG-MAP algorithm, since it does not require scaling the input LLRs with the factor $-\frac{2E_s}{\sigma^2}$, we can just make sure the LLR is in 6-bit precision. For MAX*-LOG-MAP algorithm, the LLR is multiplied with the factor $-\frac{2E_s}{\sigma^2}$ and the multiplication result should meet the 6-bit precision requirement. Assuming a 16-bit fixed point data type is used in the communication demodulation algorithm, the data must be scaled from 16-bit to 6-bit. The recommended scaling method can be maximum value based or mean value based. It is similar to the VCP2 input data precision scaling. Please refer to the solution B and solution C presented in Section 2.5.

4 Conclusion

Tuning the TCP2 or VCP2 decoder performance is related to several programming parameters. There is no single configuration set which can guarantee the best BER performance in all decoding systems. It is recommended that user perform investigation based on these programming parameters and decide which method is best for the specific application.

5 References

1. TMS320TCI648x DSP Turbo-Decoder Coprocessor 2 (TCP2) User Guide (SPRUE10)
2. TMS320TCI648x DSP Viterbi-Decoder Coprocessor 2 (VCP2) User Guide (SPRUE09)
3. TMS320C6416 Coprocessors and Bit Error Rates (SPRA974)
4. TMS320TCI648x TCP2 Channel Density (SPRAAG3)
5. L.R.Bahl et. al., “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate”, IEEE Transactions on Information Theory, March 1974, p. 284287.
6. P. Robertson, P. Hoeher, E. Villebrun, “Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo Decoding,” European Trans. on Telecommunications, vol. 8, pp.119125, Mar./Apr. 1997.
7. A Worm, P Hoeher, N Wehn, “Turbo-Decoding Without SNR Estimation”, Page 1. IEEE Communications Letters, VOL. 4, NO. 6, JUNE 2000
8. TCP2 to TCP3 Migration Guide for KeyStone Devices (SPRABH9)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com	Wireless	www.ti.com/wireless-apps
RF/IF and ZigBee® Solutions	www.ti.com/lprf		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265