AN2785
Rev. 0, 7/2004

Using the Turbo Decode Coprocessor (TCOP) of the MSC8126 DSP Device

by Yasmin Oz, Oranit Machluf, Fabrice Aidan

The turbo decode coprocessor (TCOP) of the Freescale Semiconductor StarCore™-based MSC8126 device decodes high data rate input streams and imposes a minimum CPU time overhead. This application note describes the theory of turbo decoding, the TCOP implementation on the MSC8126 device, and TCOP programming procedures. It also provides performance graphs.

The main features of the TCOP are as follows:

- Full support of the 3GPP and CDMA2000 standards with 8-state constituent encoders and polynomials as defined by those standards.
- Encoding rate of 1/2,1/3,1/4 and 1/5 with programmable puncturing of parity symbols.
- Iterative decoding using the log MAP algorithm with MAX and MAX* approximations.
- Full-flexibility interleave function via a look-up-table.
- Flexible block size (1–32767 bits).
- Programmable number of iterations; host can stop processing after every MAP decoding at run time.
- Stop condition after $\Lambda_k(d_k)$ reaches a predefined threshold.
- Big- and little-endian support.

CONTENTS

1 Turbo Coding Theory and Background

In a typical communication scheme, information transmitted from a source passes through a noisy communication channel. The received information is likely to contain errors, so error control mechanisms are needed. In 1948, C.E. Shannon proved in his classical paper [1] that a communication channel can be made arbitrarily reliable by encoding the information so that a fixed fraction of the channel is used for redundant information.

Today, there is a variety of forward error correction (FEC) algorithms. All error correction codes add carefully designed information to the data before transmission and use this redundant information for detecting errors and reconstructing the corrupted data after transmission. A typical communications scheme is depicted in **Figure 1**.

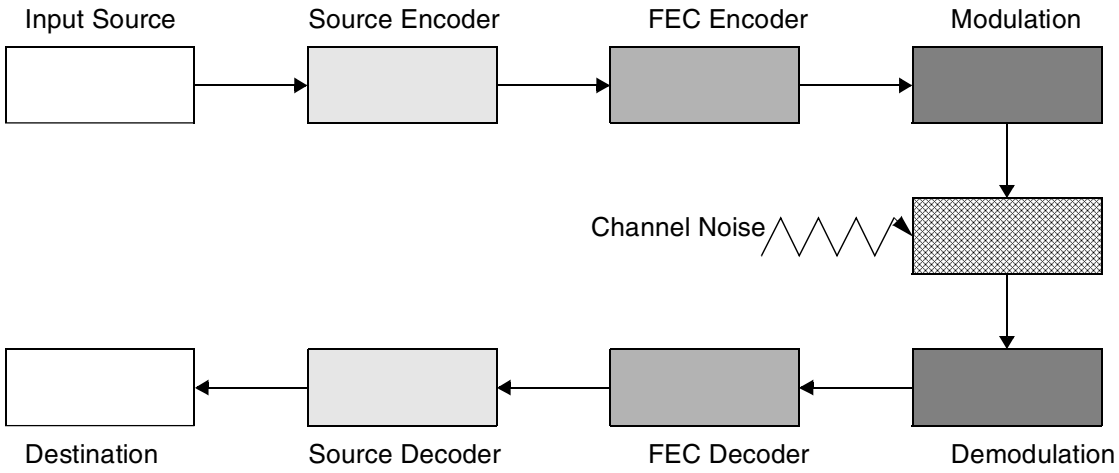

Figure 1. Typical Communication System

In 1993, C. Berrou, A. Glavieux and P. Thitimajshima introduced the first Turbo codes[2] that report results close to Shannon's theoretical lowest performance bound. Turbo codes are composed of two (or more) components that operate on different interleaved versions of the same information sequence. The decoding concept is to feed the soft decision outputs from one decoder into another decoder and to iterate this process several times to increase the reliability of the decoded sequence. This decision scheme is described in the next section.

1.1 Encoding

Encoding uses two or more constituent encoders. A popular turbo-code scheme is based on parallel concatenation of two recursive systematic convolutional (RSC) codes with interleaving in between them. The data is configured in blocks. The encoders are forced to a known state after the termination of the message block. In contrast to convolutional codes, the turbo encoders use a feedback loop to increase the number of high-weight code words. Mathematically, the encoder operation is equivalent to the division of two generator polynomials. **Figure 2** depicts a typical 8-state encoder composed of two parallel concatenated RSC encoders. The output of the encoder consists of the original bits and the parity bits.

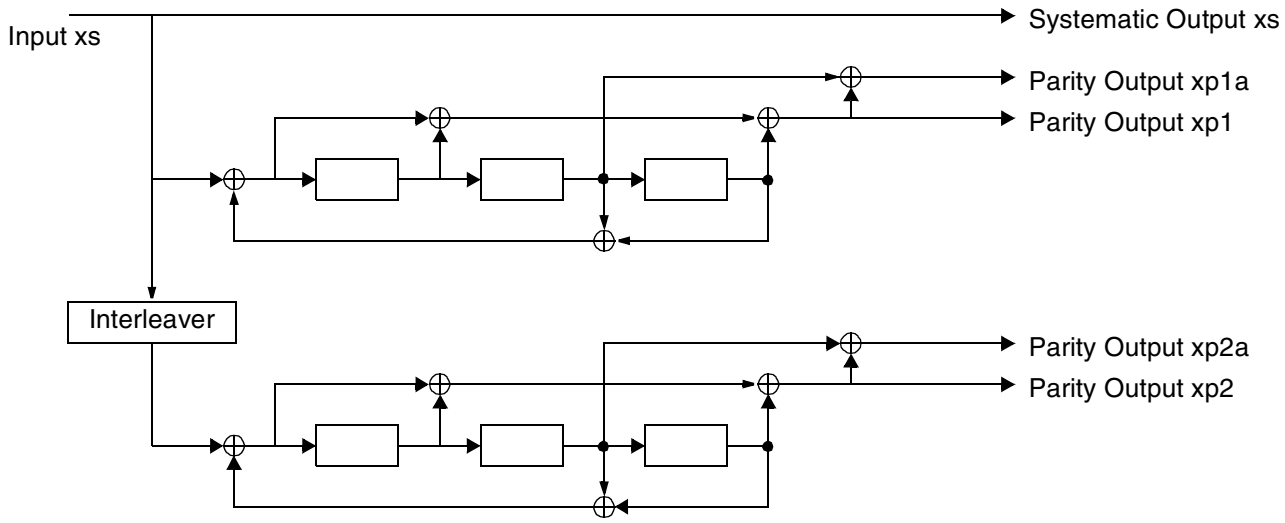

Figure 2. Turbo Coding Encoder

1.2 Decoding

Turbo decoding iteratively applies the MAP algorithm discussed in this section between two constituent decoders.

1.2.1 Maximum-a-Posteriori (MAP) Algorithm

The logical 1 and 0 elements are represented by values +1 and –1, respectively. We further denote the originally transmitted value as d and the received value as y. Because of noise, the received value is a continuous random variable with statistics dependant on the characteristics of the communication channel. The input into the detector is the received value (soft input) and the eventual output is either 1 or 0 per bit (hard decision). The decision process at the detector is based on the log-likelihood-ratio (LLR) defined in its general form as follows:

Equation 1

$$L(d|y) = \log\left[\frac{P(d=1|y)}{P(d=-1|y)}\right]$$

$P(d=\pm 1|y)$ denotes the probability that bit d is equal to +1 or –1, respectively, given received value y. The MAP algorithm determines the most likely information bit by calculating the LLR for each bit. A bit is decoded as 1 if its LLR is positive and as 0 otherwise. The magnitude of the LLR denotes the reliability of this hard decision.

The LLR of a bit d_k at time k is of a decoder with states s is formally given as follows:

Equation 2

$$L(d_k) = \log\left[\frac{\sum_{s}\lambda_k^{1,s}}{\sum_{s}\lambda_k^{-1,s}}\right]$$

where $\lambda_k^{i,s}$ is defined as the joint probability $\lambda_k^{i,s} = P(d_k = i, S_k = s|R_1^N)$ where $i = \pm 1$, S_k is the decoder state at time k.

R_1^N is the received soft input block sequence, and N denotes the bock length. The next step is to break up the received sequence into three parts (past, present and future), as shown in **Equation 3**.

Equation 3

$$\lambda_k^{i,m} = P\left(\underbrace{d_k = i, S_k = s}_{A}, \underbrace{R_1^{k-1}}_{B}, \underbrace{R_k}_{C}, \underbrace{R_{k+1}^{N}}_{D} \right)$$

Bayes' rule gives us **Equation 4**.

Equation 4

$$P(A|B,C,D) = \frac{P(A,B,C,D)}{P(B,C,D)} = \frac{P(B|A,C,D)P(A,C,D)}{P(B,C,D)}$$

Which yields the final result shown in **Equation 5**.

Equation 5

$$L(d_k) = \log\left[\frac{\sum_{(s,s'),\, d_k = 1} \alpha_{k-1}^{s} \beta_k^{s'} \gamma_k^{1,\, s \to s'}}{\sum_{(s,s''),\, d_k = -1} \alpha_{k-1}^{s} \beta_k^{s''} \gamma_k^{-1,\, s \to s''}} \right]$$

The summation is performed over all decoder states s' that are connected through a trellis branch at time $k-1$ to state s at time k, for both input bit 1 or 0.

The three factors α, β, γ are the forward state metric, the reverse state metric, and the branch metric, respectively, depicted graphically in **Figure 3**. Notice that the solid lines correspond to input –1 and the dashed lines to input 1.

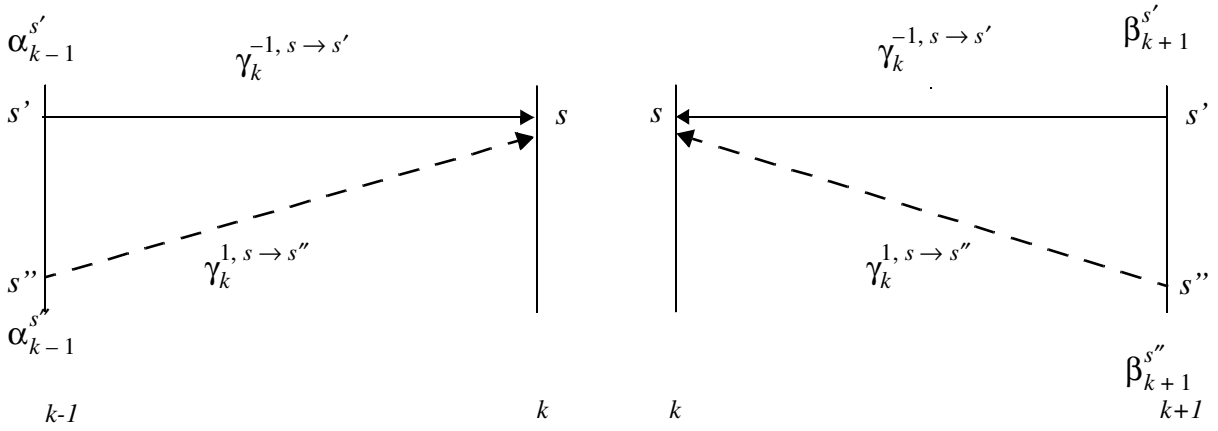

Figure 3. Forward and Reverse State Metrics

As **Figure 3** shows, the state metrics are given by **Equation 6**.

Equation 6

$$\alpha_k^s = \alpha_{k-1}^{s'} \gamma_k^{-1,\, s \to s'} + \alpha_{k-1}^{s''} \gamma_k^{1,\, s \to s''}$$

$$\beta_{k-1}^s = \beta_k^{s'} \gamma_k^{-1,\, s \to s'} + \beta_k^{s''} \gamma_k^{1,\, s \to s''}$$

Calculating the LLR is a scheme that resembles a Viterbi algorithm performed in two directions over a block of code bits. The state metrics are calculated recursively. Since the decoder state at the beginning and the end of the block is conventionally taken as state 0, we initialize the state metrics as follows.

$$\alpha_{k=0}^{s=0} = 1 \qquad \alpha_{k=0}^{s\neq 0} = 0 \qquad \text{and} \qquad \beta_{k=N}^{s=0} = 1 \qquad \beta_{k=N}^{s\neq 0} = 0$$

The first step in deriving the LLR is to derive all the branch metrics γ for the block. Then the state metrics α and β are calculated recursively and the products with the branch metrics are summed up. The branch metric γ depends on the channel characteristics. The normalized variables are defined as follows:

$$Y_{sk} \equiv \frac{2}{\sigma^2} y_{sk} \text{ and } Y_{pk} \equiv \frac{2}{\sigma^2} y_{pk}$$

Where y_{sk} denotes the actually received systematic bit and y_{pk} the received parity bit, we get an AWGN channel ([2]):

$$\gamma_k^{i,\, s\to s'} = (Pd_k = i) e^{\frac{1}{2}(Y_{sk} u_k^i + Y_{pk} v_k^{i,\, s\to s'})}$$

where u_k^i denote the originally transmitted systematic bit and $v_k^{i,\, s\to s'}$ the transmitted parity bit.

1.2.2 Log MAP Algorithm

The MAP algorithm is computationally very complex since it requires a series of multiplications. It is further very sensitive to numerical precision. Those problems are alleviated by performing the MAP algorithm in the log-domain. The LLR is given by **Equation 7**.

Equation 7

$$L(d_k) = \log\left[\frac{\sum_{(s,s'),\, d_k = 1} e^{\log\alpha_{k-1}^s} e^{\log\beta_k^{s'}} e^{\log\gamma_k^{1,\, s\to s'}}}{\sum_{(s,s''),\, d_k = -1} e^{\log\alpha_{k-1}^s} e^{\log\beta_k^{s''}} e^{\log\gamma_k^{-1,\, s\to s''}}}\right]$$

In the log-domain, a multiply operation becomes an add, and add becomes a log-add. The branch metric in the log domain is given in **Equation 8**:

Equation 8

$$\log\gamma_k^{i,\, s\to s'} \equiv \Gamma_k^{i,\, s\to s'} = \log(P(d_k = i)) + \frac{1}{2}(Y_{sk} u_k^i + Y_{pk} v_k^{i,\, s\to s'})$$

where $P(d_k = i)$ is the a-priori probability of a bit being equal to i at time k. The *a priori* LLR is defined in **Equation 9**:

Equation 9

$$\Lambda_k(d_k) \equiv \log\left[\frac{P(d_k = 1)}{P(d_k = -1)}\right]$$

The result is shown in **Equation 10**.

Equation 10

$$\Gamma_k^{i,\, s \to s'} = \underbrace{\log\left(\frac{1 + e^{-\frac{1}{2}\Lambda_k}}{1 + e^{-\Lambda_k}}\right)}_{C} + \frac{1}{2}(\Lambda_k d_k^i + Y_{sk} u_k^i + Y_{pk} v_k^{i,\, s \to s'})$$

C cancels out in the ratio of the final calculation of $L(d_k)$ and can therefore be omitted *a priori*. The calculation of the state metrics becomes **Equation 11**.

Equation 11

$$\log \alpha_k^s = \log\left(e^{\log \alpha_{k-1}^{s'} + \Gamma_k^{-1,\, s \to s'}} + e^{\log \alpha_{k-1}^{s''} + \Gamma_k^{1,\, s \to s''}}\right)$$

$$\log \beta_{k-1}^s = \log\left(e^{\log \beta_k^{s'} + \Gamma_k^{-1,\, s \to s'}} + e^{\log \beta_k^{s''} + \Gamma_k^{1,\, s \to s''}}\right)$$

This expression is a log-add operation of the general form shown in **Equation 12**.

Equation 12

$$\log(e^x + e^y) = max(x, y) + \log(1 - e^{|x-y|})$$

Thus, the log-add operation is a maximizing operation followed by a correction term that is a function of the absolute difference of x and y.

1.3 Iterative Decoding

The decoder is composed of two constituent decoders with an interleaver in between them. After the operation of the first decoder, the interleaver provides the second decoder with an additional weighted (soft) decision input. A deinterleaver captures the output of the second decoder and passes it as feedback information into the first decoder. Therefore, each decoder receives different redundant information on the same information sequence. The second decoder acts as a *second opinion* after it receives information from the decoder, and it in turn influences the decision of the first decoder until they reach a common decision.

In systematic code, it can be shown ([2]) that the LLR calculated by the MAP algorithm is equal to **Equation 13**.

Equation 13

$$L(d_k) = L_c(Y_{sk}) + \Lambda_k(d_k) + L_e(d_k)$$

where $L_c(Y_{sk})$ is the LLR due to the channel measurement and $\Lambda_k(d_k)$ is the *a priori* LLR. For an AWGN channel, we have $L_c(Y_{sk}) = Y_{sk}$. The term $L_e(d_k)$ is the extrinsic LLR that is the information gained in the current decoding iteration. The iterative process proceeds is as follows:

1. Calculate $L(d_k)$.

2. Subtract $L_c(Y_{sk})$ and $\Lambda_k(d_k)$ from $L(d_k)$ to obtain $L_e(d_k)$.
3. Feed $L_e(d_k)$ into the next decoder as $\Lambda_k(d_k)$.
4. Repeat steps 1–3 until the decision threshold is reached.

In the first iteration, since no *a priori* information is available, $\Lambda_k(d_k)$ is set to zero.The hard-decision bit is generated from the final $L(d_k)$, which is the sum of the channel LLR $L_c(Y_{sk})$, the extrinsic LLR from the first decoder, and the extrinsic LLR from the second decoder.

2 TCOP Implementation on the MSC8126

Turbo decoding highly taxes calculation resources. Most current DSP devices support only low data rates—for example, 384 kbps—which consume most of their computing power. The MSC8126 TCOP is designed to support all known standards. It connects to a shared area in the system memory. You must provide the data block and the interleave table for this system memory area. The TCOP decodes the block and returns the result to the same memory area for post processing. Each of the two MAP decoders generates the *a posteriori* information $\Lambda_k(d_k)$ that the other decoder uses as *a priori* information. The second MAP decoder, MAP2, operates on interleaved data; that is, it receives interleaved Y_{sk} and $\Lambda_k(d_k)$. Therefore the $\Lambda_k(d_k)$ should be interleaved before it is fed back into the first MAP decoder, MAP1. Since the MAP algorithm actually uses $\Lambda_k(d_k) + L_c(Y_{sk})$ and $L_c(Y_{sk}) = Y_{sk}$, we save one interleaver as depicted for code rate 1/3 in **Figure 4**.

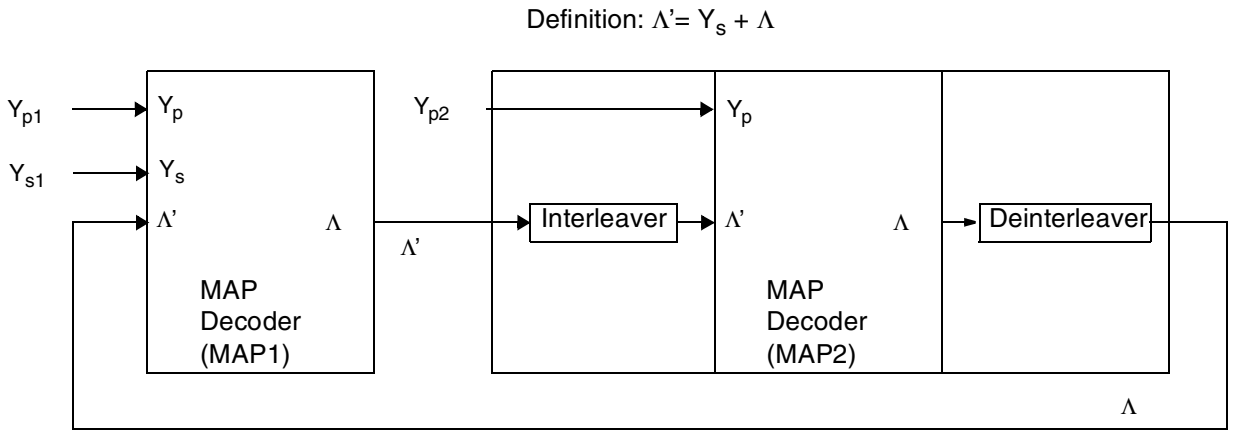

Figure 4. TCOP Decoder Implementation

2.1 3GPP and CDMA2000 Standards

The main differences between the 3GPP and CDMA2000 standards are the code rate, the puncturing scheme, and the resulting tail bits required to return the decoder to state 0. In both cases, there are eight encoder states. The 3GPP block size ranges up to about 5000 bits, and the CDMA2000 block size ranges up to about 20000 bits. For 3GPP, the required code rate is 1/3 with encoder polynomial 15/13. CDMA2000 supports code rates 1/2, 1/3, and 1/4 with encoder polynomial 15/13 for code rates 1/2 and 1/3 and polynomial 17/13 for code rate 1/4.

2.2 Addition in the Log Domain: MAX and MAX*

The TCOP performs its computations in the log-domain, which implies that addition operations become log-add operations as described at the end of **Section 1.2.2**, *Log MAP Algorithm,* on page 5. The log-add operation is approximated by various schemes. As a first estimate, we simply omit the correction term $\log(1 - e^{|x-y|})$ to yield **Equation 14**.

Equation 14

$$\log(e^x + e^y) \approx max(x, y)$$

A correction that is a function of the absolute difference of x and y can be added to the MAX operation. The refined MAX operation is denoted as the MAX* operation. The TCOP uses a linear function to approximate the correction term shown in **Equation 15**.

Equation 15

$$\log(e^x + e^y) \approx max^*(x, y) \equiv max(x, y) + \frac{1}{2}(C - |x - y|)$$

The constant parameter C depends on the signal-to-noise (SNR) ratio and is programmable in the CF field of the Correction Factor Register (CFR), which is described in the TCOP chapter of the *MSC8126 Reference Manual*.

2.3 Dual-Pass Scheme

The MAP algorithm requires soft-input for-output (SISO) decoding for each bit position. However, direct implementation of SISO decoding heavily consumes memory because an entire recursion must be stored. The number of bits required per recursion is $N \times M$ bits/state where N is the block size and M is the number of decoder states. For example, $M = 8$ and 13 bits/state requires 2 Mbits in CDMA2000 and 0.5 Mbits in 3GPP.

The dual-pass approach reduces the memory demands by processing the data in two passes:

- *Pass 1*. Calculates the β values but stores them only at checkpoints. The block of N bits is divided into sub-blocks of L bits. The β values are calculated backwards for every bit of the N-bit wide block, but a β value is stored in the system memory only every L bits.
- *Pass 2*. Each sub-block is processed separately backwards. The β value at the beginning of the sub-block is loaded and the β, and γ are calculated backwards and stored in the TCOP internal memory. Finally, the α and Λ values are calculated forwards to the beginning of the sub-block and the Λ values are stored in the system memory.

The dual pass procedure is graphically depicted in **Figure 5** and **Figure 6**, for $M = 8$.

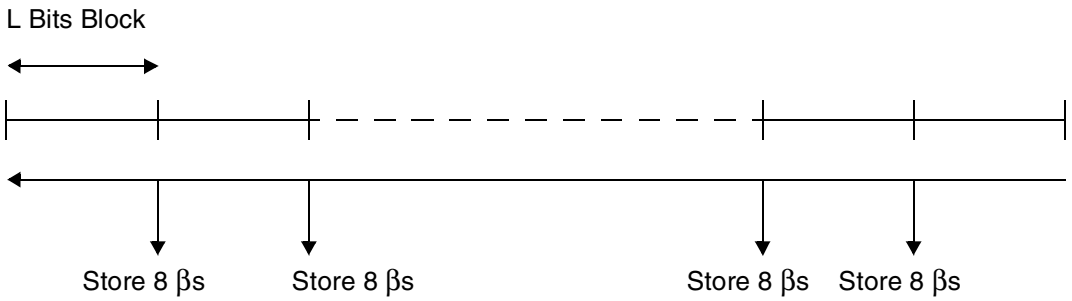

Figure 5. Pass 1

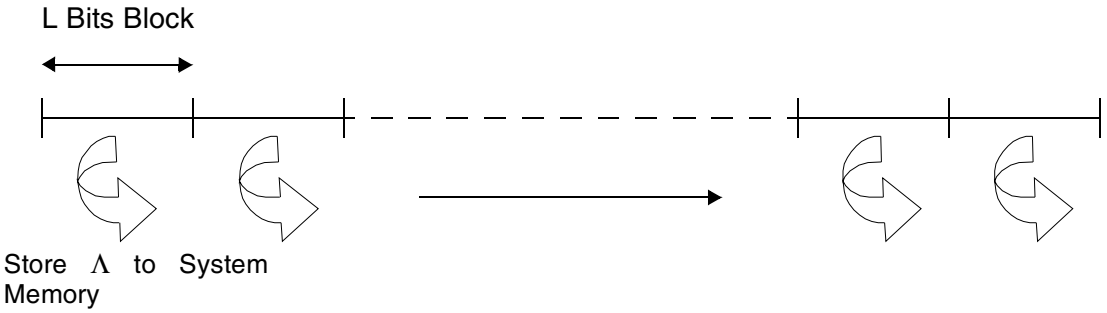

Figure 6. Pass 2

The size L of the sub-blocks in the TCOP is 64 bits. The dual-pass scheme is exact and thus optimal. There is no performance degradation due to activities such as windowing.

2.4 Stopping Condition

A stopping condition allows the number of iterations to be adapted to the quality of the input frame. There are two stopping conditions for early exit of the iterative decoding procedure:

1. A predetermined number of iterations.
2. A threshold condition on Λ is reached.

The threshold stop condition is met for a MAP decoder if there is at most a predefined number, written in the Stop Condition Register SCR[LTC] field, of Λ soft values that lie below a threshold, written in the SCR[LTV] field.

2.5 Input/Output Format

Both systematic and parity data are stored as symbols in the system memory. The symbol format used in the TCOP is 8-bits wide signed fractional numbers in two's complement representation. Depicted below are Y inputs in the s3.4 format.

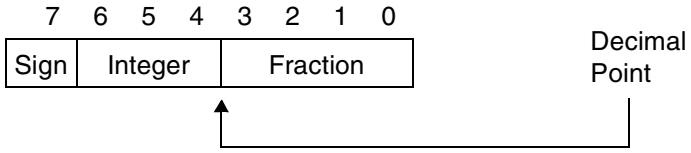

Figure 7. Y Inputs Fixed-Point Format

The output Λ has a similar format as the symbol format, but with the integer range extended by additional 3 bits.

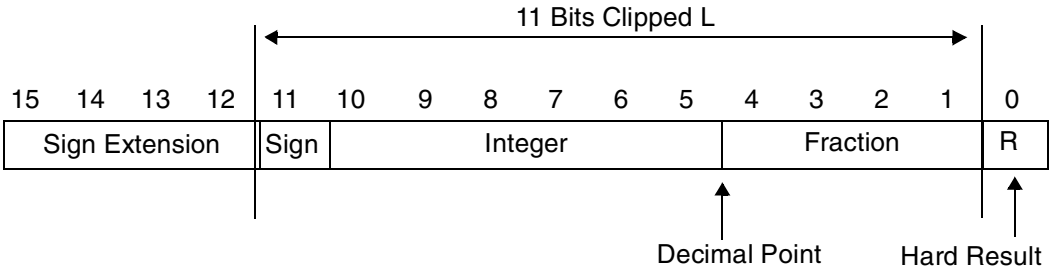

Figure 8. Λ Outputs Fixed-Point Format

The systematic input block Y_s is composed of the individual symbols Y_{sk}, which are the normalized received values. The block resides in the system memory, and the Y-Systematic Base Address (YSBA) register points to it.

The Y-Parity 1 Base Address (YP1BA) and Y-Parity 1A Base Address (YP1ABA) registers point to parity blocks Y_{p1} and Y_{p1a} (for code rate 1/4 only). Similarly, the YP2BA and YP2ABA registers point to parity blocks Y_{p2} and Y_{p2a}.

The outputs of the TCOP are N 16-bits words in the system memory, and the Lambda Base Address (LBA) register points to them. The bit in the LSB position of each word is the output hard-decision bit, and the remaining 15 bits are the corresponding extrinsic value.

Both systematic and parity data are stored in the system memory. Since all symbols are 8-bits wide, the memory required for the input data is N bytes divided by the code rate. The memory required for the TCOP output is $2N$ bytes.

Further, $\frac{N}{4}$ + 48 bytes are required for the dual-pass scheme, and $2N$ bytes are required for interleaving.

2.6 Operating the TCOP

To operate the TCOP, perform the following steps:

1. Set up the addresses and data.

   Program the base addresses for data buffers, Λ values, interleave table, and intermediate β (registers YSBA, YP1BA, YP2BA, YP1ABA, YP2ABA, IOBA and BIBA, respectively, as defined in [4]) to their desired values. Then prepare the data buffers Y_s, Y_{p1} and Y_{p2} (plus Y_{p1a} and Y_{p2a} for code rate 1/4 only). The initial α and β values are calculated and written to the system memory, and the N initial Λ values are set to zero in the LBA area.

2. Set up the control parameters.

   The basic control parameters are block size, coding rate, number of iterations, and interrupt enable after decoding completes. These parameters are programmed in the TCOP control registers TCR1 and TCR2. If the TCR1[MAX] bit is set, the log-add operation proceeds according to the MAX* scheme. The constant C as introduced in **Section 2.2** must be programmed in the Correction Factor Register CFR[CF] bit field. The maximum number of MAPs is programmed in the TCR2[MAXMAP] bit field. If the TCR2[SC] bit is set to enable a threshold stop condition, the decoder stops when the SCR[LTC] absolute values of the output Λ are less than the value specified in the threshold SCR[LTV] bit field.

3. Activate the TCOP by setting the TCOP enable TCR1[TEN] bit.

4. To terminate the TCOP when decoding is finished, wait for an interrupt if interrupts are enabled or read the TSR[END] bit.

Finally, the hard-decision bit must be extracted from the LBA area during quick post processing. For details on TCOP programming and an assembly code driver, see [5].

3 Performance Results

The TCOP offers a linear approximation of the log-add operation.The constant parameter C depends on the SNR. The format of C must be identical to the symbol format presented in **Section 2.5**.

C can be estimated from the Taylor expansion of $log(e^x + e^y)$ as:

$$C(SNR) = 2log(2)10^{\frac{SNR}{10}}$$

Figure 9 shows the results obtained by looking for the optimal constant and running 1000 blocks of 320 bits. The format is s3.4. The circle shows the theoretical result.

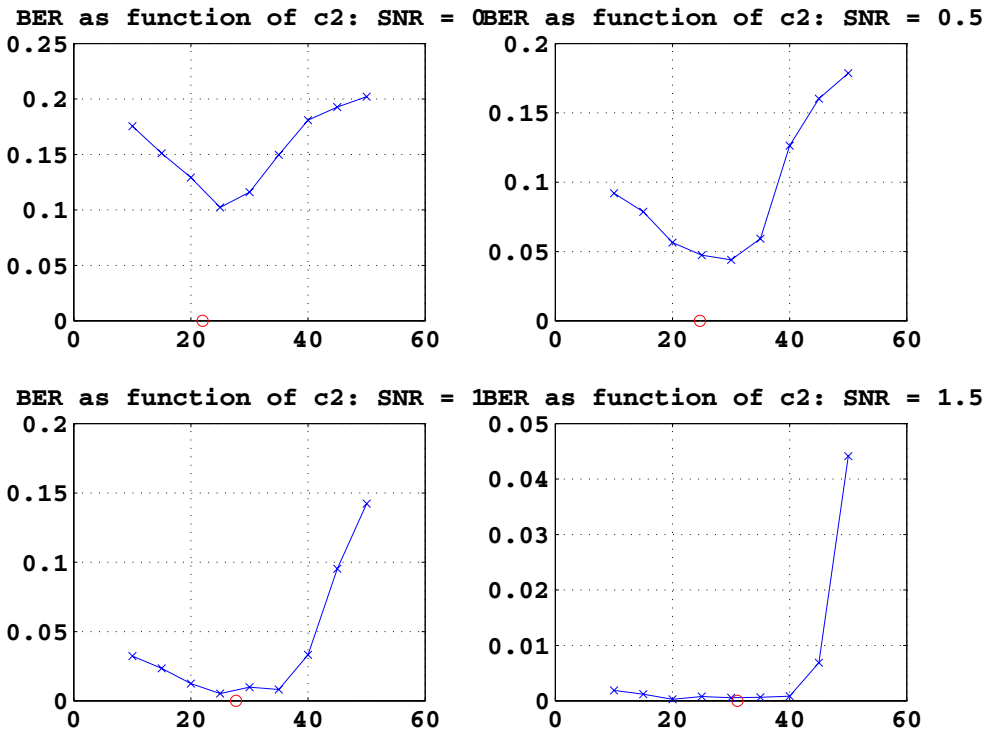

Figure 9. Bit Error Rate (BER) as Function of C for MAX* Metric

As **Figure 9** shows, the actual optimal values of C are higher than the theoretical estimate. The following expression is a good approximation of C:

$$C(SNR) \approx 1.6875 + 0.25SNR$$

In the s3.4 format, the expression is as follows:

$$C(SNR) \approx 27 + 4SNR$$

3.1 MAX Versus MAX* Approximation

This section reports the results of the bit error rate (BER) for a channel with Gaussian noise for block sizes 320, 1400, and 3840. Each simulation was run for 10000 blocks. **Figure 10** and **Figure 11** show the convergence of the BER of a 320-bit block, for 16 MAPs (= 8 full iterations). The SNR-dependence of C used in the MAX* metric was as presented earlier. The graphs show that applying the MAX* metric yields a lower BER for all SNR compared to the BER obtained with the MAX metric. The convergence rate is even faster.

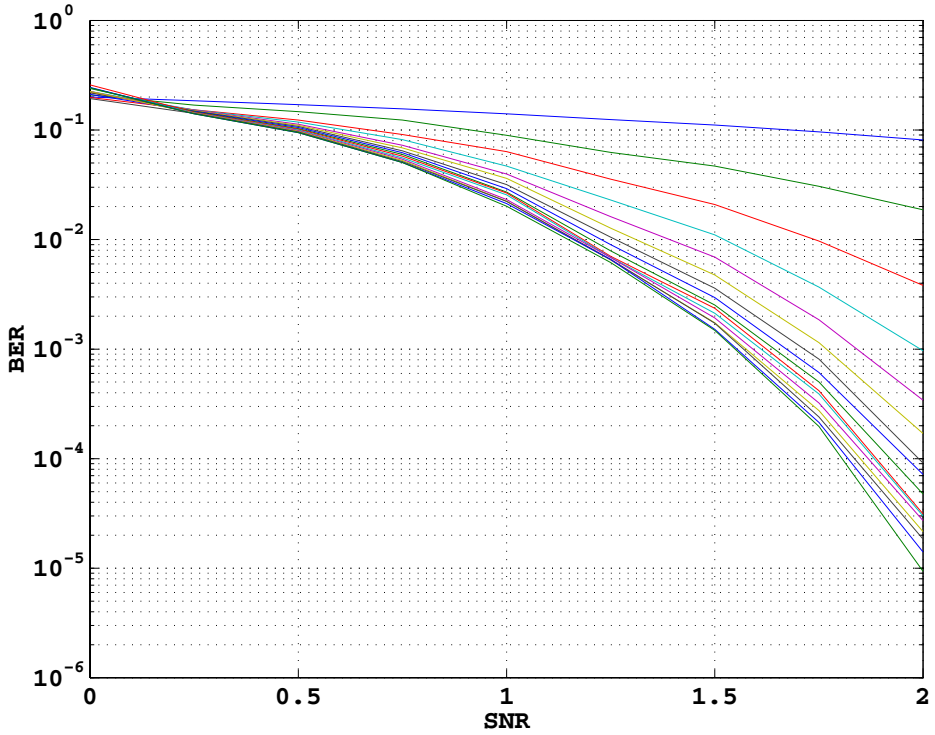

Figure 10. Convergence of BER from 1 to 16 MAPs: Block Size = 320, MAX Metric

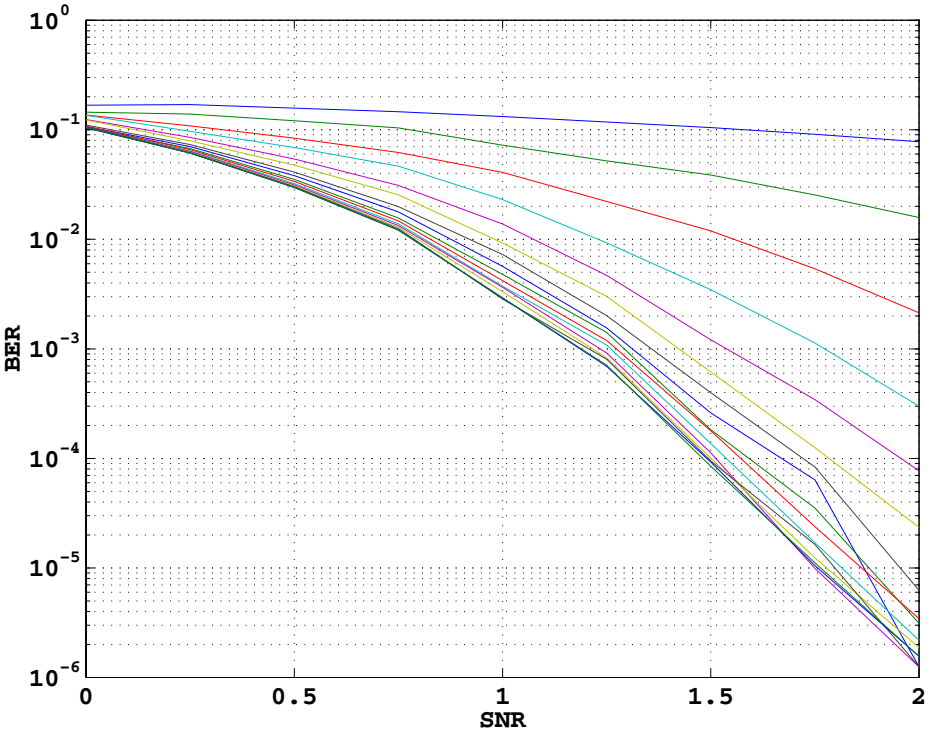

Figure 11. Convergence of BER from 1 to 16 MAPs: Block Size = 320, MAX* Metric

Figure 12 depicts the BER after eight full iterations using the MAX and the MAX* metric for block sizes of 320, 1400, and 3840 bits. Again, the MAX* metric out-performs the MAX metric by over 0.5 dB. The dashed line represents the MAX metric, and the solid line represents the MAX* metric.

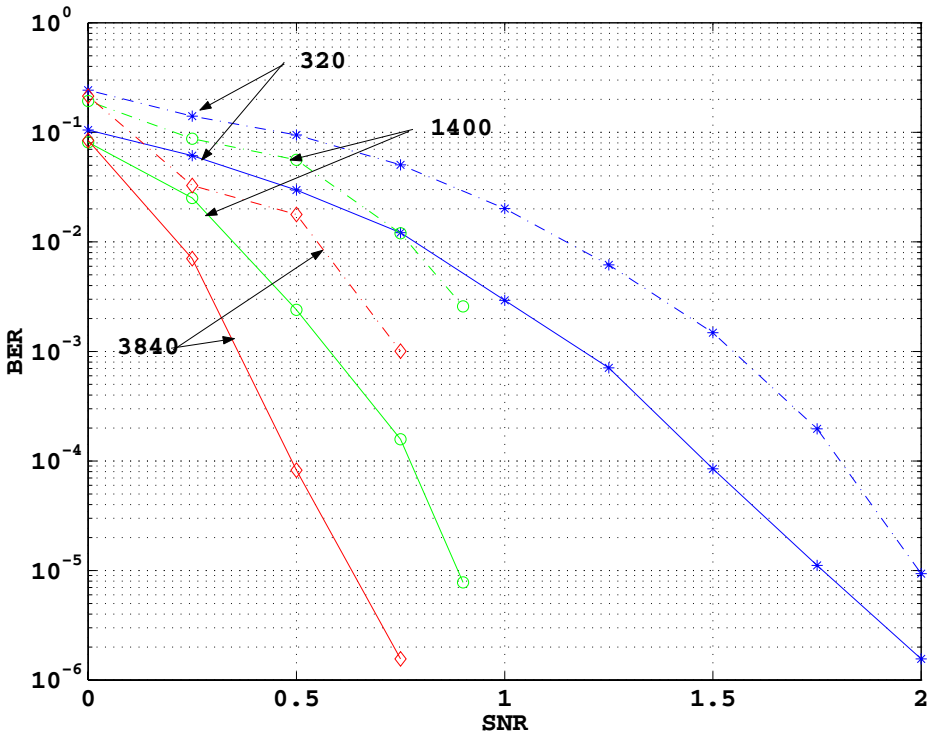

Figure 12. BER at Eight iterations (16 MAPs).

3.2 Stopping Condition

If the stop condition is enabled, the decoder is stopped if at most SCR.LTC values among the magnitudes of the output Λ are less than SCR.LTV. The format of SCR.LTV must be identical to the symbol format.

The values for SCR.LTV and SCR.LTC depend on the desired output BER. Let us denote SCR.LTV as $|\Lambda_T|$ and the fraction of the block (that is, SCR.TLC/TCR1.BLKSZ) which has $|\Lambda|$ less than $|\Lambda_T|$ as p.

Larger $|\Lambda_T|$ associated with smaller p give better results but require more iterations. Smaller $|\Lambda_T|$ associated with larger p require fewer iteration but may give poorer results in terms of Bit Error Rates (BER).

Figure 13 depicts the BER as a function of the SNR for different $|\Lambda_T|$ and p. **Figure 14** depicts the number of MAPs as a function of the SNR for the same $|\Lambda_T|$ and p values. As the graphs show, for a correct choice of the parameters the number of iterations decreases rapidly, saving power consumption and processing delay, for an acceptable performance degradation.

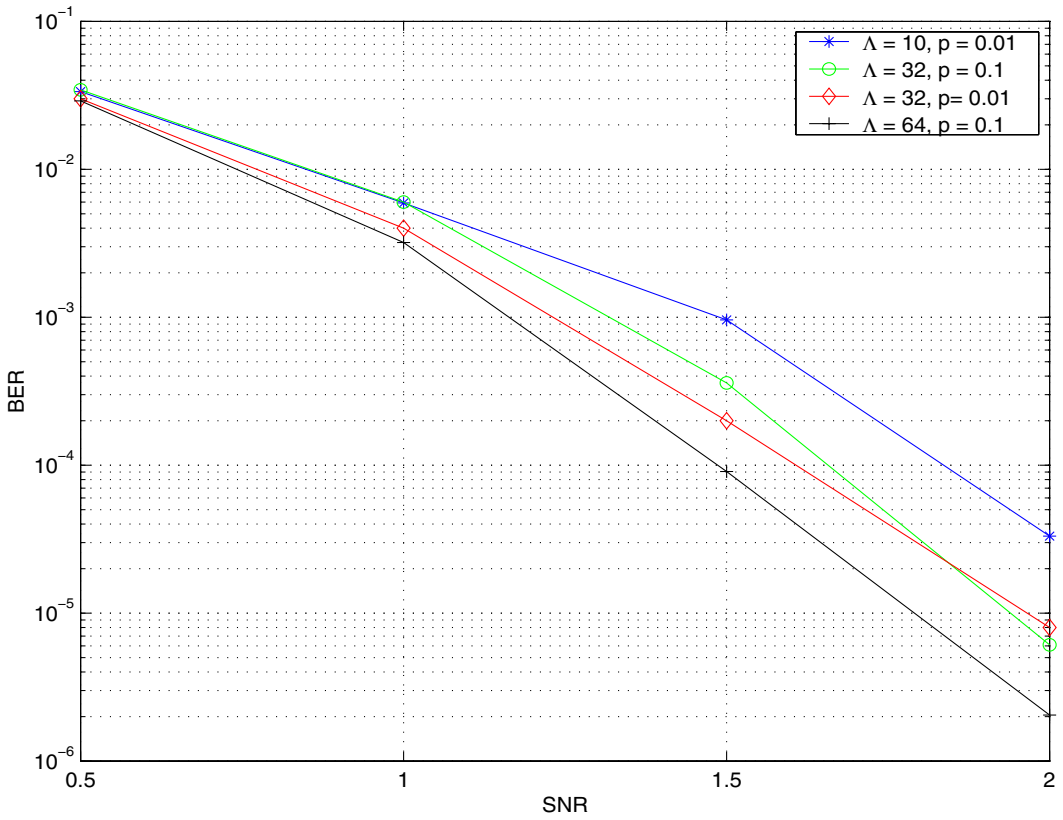

Figure 13. Influence of Stop Condition Parameters on BER Performance

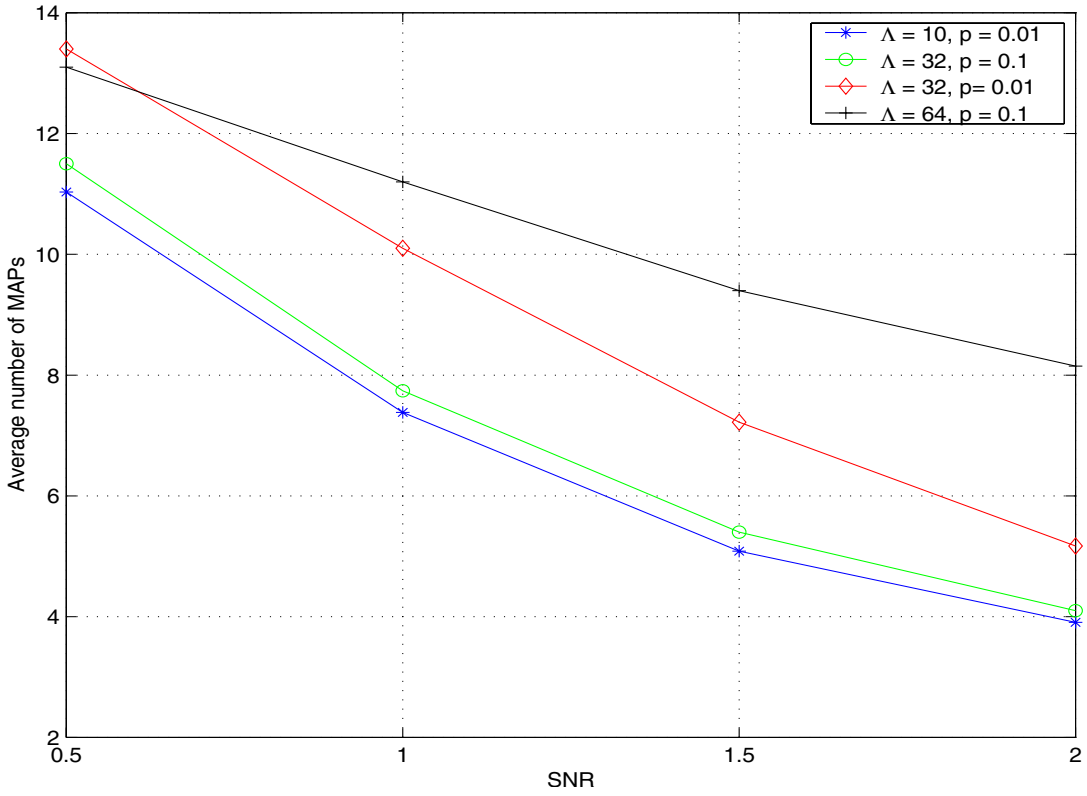

Figure 14. Influence of Stop Condition Parameters on Number of MAPs

4 References

[1] C. E. Shannon, “A Mathematical Theory of Communication,” *Bell System Tech. Journal*, vol. 27,1948, pp. 379–423 and pp. 623–656.

[2] C. Berrou, A. Glavieux and P. Thitimajshima, *Proc. IEEE International Conference Communications*, 1993, pp. 1064–1070.

[3] B. Sklar, *Digital Communications: Fundamentals and Applications,* Prentice Hall, 2001.

[4] *MSC8126 Reference Manual* (MSC8126RM), Freescale Semiconductor, 2004.

[5] *MSC8126 User’s Guide* (MSC8126UG), Freescale Semiconductor, 2004.

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor
Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
Technical Information Center
3-20-1, Minami-Azabu. Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
http://www.freescale.com

AN2785
Rev. 0
7/2004

freescale™ semiconductor