PCD04D4 Features

Turbo Decoder

- 16 state DVB–RCS2 compatible
- Rate 1/2 or 1/3
- 40 to 4800 bit interleaver
- Up to 192 MHz internal clock
- Up to 136 Mbit/s with 5 decoder iterations
- 6–bit signed magnitude input data
- 4 parallel MAP decoders
- Optional log–MAP or max–log–MAP constituent decoder algorithms
- Up to 32 iterations in 1/2 iteration steps
- Optional power efficient early stopping
- Optional extrinsic information scaling and limiting
- Estimated channel error output
- Free simulation software

- Available as EDIF core and VHDL simulation core for Xilinx Virtex–II, Spartan–3, Virtex–4, Virtex–5, Virtex–6, Spartan–6 and 7–Series FPGAs under SignOnce IP License. Actel, Altera and Lattice FPGA VHDL cores available on request.
- Available as VHDL core for ASICs

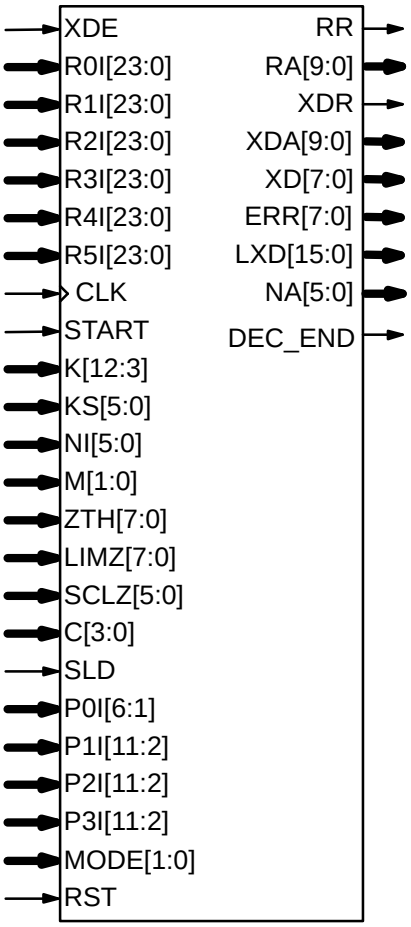

Figure 1: PCD04D4 schematic symbol.

Introduction

The PCD04D4 is a compatible DVB–RCS2 [1] error control decoder. DVB–RCS2 uses a 16 state rate 2/4 duo–binary tail–biting turbo code with an almost regular permutation (ARP) interleaver.

For DVB–RCS2, there are 24 interleaver sizes ranging from 112 to 4792 bits. Five parameters P, Q_0, Q_1, Q_2, and Q_3 are used by the interleaver. The decoder uses a simplified version of the interleaver with four parameters.

For DVB–RCS2 a code rate of 1/3 is specified. This code is punctured to obtain lower rates. The code uses a 16 state rate 2/4 systematic recursive convolutional tail–biting constituent code. Since a tail–biting code is used, there are no tail bits, increasing the bandwidth efficiency of the code.

Four MAP04D MAP decoder cores are used with the PCD04D4 core to iteratively decode the DVB–RCS2 turbo code. The log–MAP algorithm for maximum performance or the max–log–MAP algorithm for minimum complexity and highest speed can be selected. The extrinsic information can be optionally scaled and limited with each half iteration, improving performance with max–log–MAP decoding.

The reverse sliding block algorithm is used with sliding block lengths of L = 32 or 64. To reduce MAP decoder delay by approximately one half, the data is input and output in reversed blocks of L for $K/8 > L$, where K is the data length in bits. Six–bit quantisation is used for maximum performance.

The turbo decoder can achieve up to 136 Mbit/s with 5 iterations and max–log–MAP decoding using a 192 MHz internal clock (K = 4792). Log–MAP decoding decreases speed by about 30%. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance.

Figure 1 shows the schematic symbol for the PCD04D4 decoder. The EDIF core can be used with Xilinx Integrated Software Environment (ISE) or Vivado software to implement the core in Xilinx FPGA's. A VHDL simulation core is also provided. For other FPGA families and VHDL ASIC licenses, a VHDL core is provided.

Table 1 shows the resources used for various Virtex–4 and Virtex–5 devices. Resources for Virtex–II and Spartan–3 devices are similar to that for Virtex–4. Resources for Virtex–6, Spartan–6 and 7–Series devices are similar to that for Virtex–5. The MODE[1:0] inputs can be used to select various decoder implementations. The input/output memory is not included. Only one global clock is used. No other resources are used.

Table 1: Resources used.

Configuration	Virtex–4 LUTs	Virtex–5 LUTs	Block RAMS
Max–log–MAP	42,068	34,226	16
Log–MAP	58,293	49,277	16

Table 2 shows the performance achieved with various Xilinx parts. T_{cp} is the minimum clock period over recommended operating conditions. These performance figures may change due to device utilisation and configuration.

Table 2: Performance of Xilinx parts.

Xilinx Part	T_{cp} (ns)	f (MHz)	f_d (Mbit/s)
XC4VLX60–10	10.954	91.2	64.8
XC4VLX60–11	9.343	107.0	75.9
XC4VLX60–12	8.299	120.4	85.5
XC5VLX85–1	10.274	97.3	69.1
XC5VLX85–2	8.790	113.8	80.8
XC5VLX85–3	7.853	127.3	90.4
XC6VLX75T–1	8.775	114.0	80.9
XC6VLX75T–2	7.564	132.2	93.8
XC6VLX75T–3	6.796	147.1	104.4
XC7A100T–1	10.828	92.4	65.6
XC7A100T–2	8.876	112.7	80.0
XC7A100T–3	7.895	126.7	89.9
XC7K160T–1	6.925	144.4	102.5
XC7K160T–2	5.684	175.9	124.9
XC7K160T–3	5.188	192.8	136.8

Max–Log–MAP, 5 iterations, K = 4792, L = 32

Signal Descriptions

C MAP Decoder Constant (0–11)

CLK System Clock

DEC_END Decode End Signal

ERR Estimated Error

P0I–P3I Interleaver parameters (used when KS = 0).
P0I[6:1] = P div 2
P1I[11:2] = Q_1
P2I[11:2] = (Q_0P + Q_2) mod K/8
P3I[11:2] = (Q_0P + Q_3) mod K/8

K Data Length (K = 40–4800)
K[12:3] = K/8

KS KS Data Length Select
0 = select K, P0I–P3I
1,3* = length 304 (38 bytes)
2* = length 112 (14 bytes)
4* = length 472 (59 bytes)
5* = length 680 (85 bytes)
6 = length 768 (96 bytes)
7 = length 864 (108 bytes)
8* = length 920 (115 bytes)
9* = length 1040 (130 bytes)
10 = length 1152 (144 bytes)
11* = length 1400 (175 bytes)
12* = length 1552 (194 bytes)
13* = length 984 (123 bytes)
14 = length 1504 (188 bytes)
15 = length 2112 (264 bytes)
16* = length 2384 (298 bytes)
17* = length 2664 (333 bytes)
18* = length 2840 (355 bytes)
19 = length 3200 (400 bytes)
20 = length 3552 (444 bytes)
21* = length 4312 (539 bytes)
22* = length 4792 (599 bytes)
32,33 = length 800 (100 bytes)
34–36* = length 1360 (170 bytes)
37–39* = length 3504 (438 bytes)

* While DEC_END = 0, XD valid for even NA only.

LIMZ Extrinsic Information Limit (1–193)

LXD Decoded symbol log probability (0–14)

M Early Stopping Mode
0 = no early stopping
1 = early stop at odd half iteration
2 = early stop at even half iteration
3 = early stop at any half iteration

MODE Implementation Mode (see Table 3)

NA Half Iteration Number (0–63)

NI Number of Half Iterations (0–63)
NI = 2I–1 where I is number of iterations

R0I Received Data (A)

R1I Received Data (B)

R2I Received Parity (Y_1)

R3I Received Parity (Y_2)

R4I Received Parity (W_1)
R5I Received Parity (W_2)
RA Received Data Address
RR Received Data Ready
RST Synchronous Reset
SCLZ Extrinsic Information Scale (1–32)
SLD Sliding window select
0 = small window (L = 32)
1 = large window (L = 64)
START Decoder Start
XD Decoded Data Byte
XDA Decoded Data Address
XDE Decoded Data Enable
XDR Decoded Data Ready
ZTH Early Stopping Threshold (1–255)

Table 3: MODE selection

Input	Description
MODE0	0 = max–log–MAP 1 = log–MAP
MODE1	0 = rate 1/2 1 = rate 1/3

Table 3 describes each of the MODE[1:0] inputs that are used to select various decoder implementations. Note that MODE[1:0] are “soft” inputs and should not be connected to input pins or logic. These inputs are designed to minimise decoder complexity for the configuration selected.

Note that the required size of each of the 16 internal interleaver memories is 150x28. This is implemented using 16 512x36 Xilinx BlockRAMs. Although the nominal maximum data length is 4800 bits, the decoder can actually decode up to 8128 bits for SLD = 0 or 8152 bits for SLD = 1.

Turbo Decoder Parameters

For optimal performance, the maximum a posteriori (MAP) [2] constituent decoder can be used which is dependent on the signal to noise ratio (SNR). Unlike other turbo decoders with sub-optimum soft–in–soft–in (SISO) decoders, using the MAP (or specifically the log–MAP [3]) algorithm can provide up to 0.5 dB coding gain at low SNRs. Log–MAP operation is implemented when MODE0 is high.

With binary phase shift keying (BPSK, m = 1) or quadrature phase shift keying (QPSK, m = 2) modulation (see Figure 2) the decoder constant C should be adjusted such that

$$C = A\sigma^2\sqrt{m}/2. \quad (1)$$

where A is the signal amplitude and σ^2 is the normalised noise variance given by

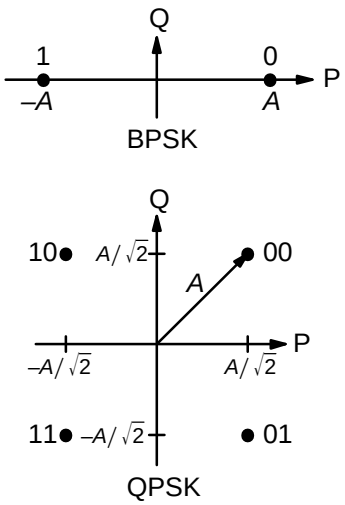

Figure 2: BPSK and QQPSK signal sets.

$$\sigma^2 = 1/(2mRE_b/N_0). \quad (2)$$

E_b/N_0 is the energy per bit to single sided noise density ratio, R = 1/n is the code rate, K is the data length, and n = 2 or 3. C should be rounded down to the nearest integer and limited to be no higher than 11. Max–log–MAP [3] operation occurs when C = 0. Due to quantisation effects, C = 1 is equivalent to C = 0. Thus C = 1 is internally rounded down to C = 0. Also, as C = 2 or 3 give poor performance due to quantisation effects, these values are internally rounded up to C = 4. Max–Log–MAP operation is implemented when MODE0 is low.

For each code (with a particular block size, rate and number of iterations), there will be a minimum E_b/N_0 where the maximum acceptable BER or FER is achieved. The value of C should be chosen for this E_b/N_0. This value of C can be kept constant for all E_b/N_0 values for this code. For higher values of E_b/N_0, there will be negligible degradation in performance, even though C will be higher than optimal [4]. For lower E_b/N_0 values, there could be up to a few tenths of a dB degradation, since C will be lower than optimal. However, this should not have much impact since the BER or FER will already be above the maximum acceptable level anyway.

For fading channels the value of A and σ^2 should be averaged across the block to determine the average value of C. Each received value r_k should then be scaled by $(A\sigma^2)/(A_k\sigma_k^2)$ where A_k and σ_k^2 are the amplitude and normalised variance of r_k. Note that this scaling should be performed for both the log–MAP and max–log–MAP algorithms for optimal performance.

The value of A directly corresponds to the 6–bit signed magnitude inputs (shown in Table 4). The

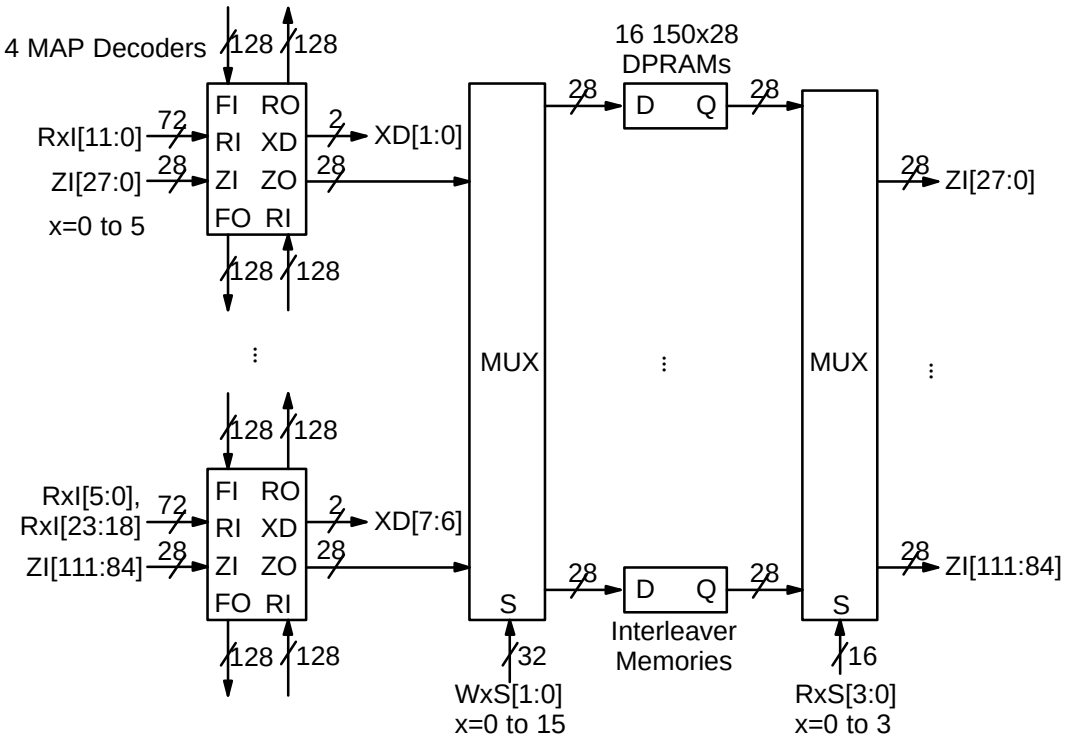

Figure 3: Simplified block diagram of PCD04D4 16 state turbo decoder.

6–bit inputs have 63 quantisation regions with a central dead zone. The quantisation regions are labelled from –31 to +31. For example, one could have A = 15.7. This value of A lies in quantisation region 16 (which has a range between 15.5 and 16.5).

Table 4: Quantisation for R0I[5:0], etc.

Decimal	Binary	Range
31	011111	30.5↔∞
30	011110	29.5↔30.5
⋮	⋮	⋮
2	000010	1.5↔2.5
1	000001	0.5↔1.5
0	000000	–0.5↔0.5
33	100001	–1.5↔–0.5
34	100010	–2.5↔–1.5
⋮	⋮	⋮
62	111110	–30.5↔–29.5
63	111111	–∞↔–30.5

Since most analogue to digital (A/D) converters do not have a central dead zone, a 7–bit A/D should be used and then converted to 6–bit as shown in the table. This allows maximum performance to be achieved.

For signed magnitude inputs a decimal value of 32 has a magnitude of 0 (equivalent to –0). External two's complement values will need to be converted to sign magnitude for input to the decoder. Note that for two's complement decimal 32 (integer –32), this needs to be limited to decimal 33 (integer –31).

For input data quantised to less than 6–bits, the data should be mapped into the most significant bit positions of the input, the next bit equal to 1 and the remaining least significant bits tied low. For example, for 3–bit received data R0T[2:0], where R0T[2] is the sign bit, we have R0I[5:3] = R0T[2:0] and R0I[2:0] = 4 in decimal (100 in binary). For punctured input data, all bits must be zero, e.g., R1I[5:0] = 0.

Due to quantisation and limiting effects the value of A should also be adjusted according to the received signal to noise ratio.

Example 1: Rate 1/3 BPSK code operating at E_b/N_0 = 0.3 dB. From (2) we have σ^2 = 1.39988. Assuming A = 9.0 we have from (1) that C = 6 rounded down to the nearest integer.

Figure 3 gives a simplified block diagram of the PCD04D4 16 state turbo decoder. The extrinsic information output for each of the four data symbols is given by four 9–bit values within the MAP decoder. To reduce the number of bits, two bits of ZO indicates the symbol with the smallest extrinsic information, with three eight bit values giving the scaled and limited extrinsic information of the other three symbols, minus the extrinsic information of the symbol with the smallest extrinsic

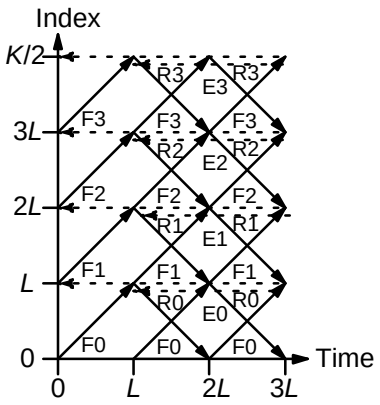

Figure 4: Timing diagram for $L = K/8$.

information. This gives increased performance compared to using the extrinsic information for each of the two decoded bits, which entails a loss of information. The remaining two bits of ZO are the hard decision values of the received data.

For even half iterations (odd NA), data is decoded for the interleaved received data. This means that data must be read in interleaved order. Due to the nature of the ARP interleaver, this would imply that the input memory (which is external to the decoder) would need to be implemented using 16 separate memories, the same as for the interleaver memory. To avoid this, we add a delayed version of the received data to the decoder extrinsic information for odd half iterations. For even half iterations, only the extrinsic information is output. This implies the input memory can be implemented using a single memory.

The above also explains the two hard decision bits in ZO. These bits are needed to determine the estimated error output ERR for even half iterations.

The sliding window algorithm used depends on $K/8$ and L. For $K/8 = L$, Figure 4 shows how the forward and reverse state metrics (SM) are calculated. The horizontal axis shows decoder time. The vertical axis the received symbol index. An arrow going up shows forward SMs for L symbols. An arrow going down shows reverse SMs for L symbols. Horizontal dashed arrows going backwards indicates SMs being passed between iterations. Forward SMs are indicated by an F and reverse SMs by R. Forward SMs that have been reversed in time are indicated by E. Decoded data is output when R and E are used together.

We see that in this case the RAMs are read in forward blocks of L. Decoded data is written in reversed blocks of L. Also, the SMs are allowed to settle for L symbols before being used. As L can be very small, e.g., $L = 14$ for $K = 112$, this is not sufficient to obtain reliable SMs. Thus, we pass

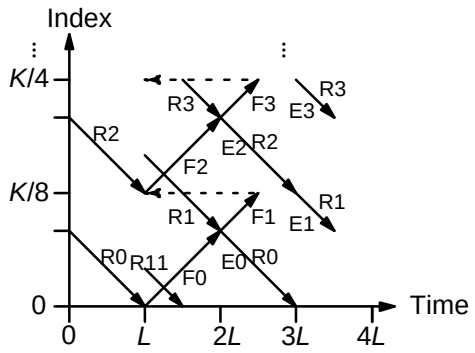

Figure 5: Timing diagram for $L < K/8 \leq 2L$.

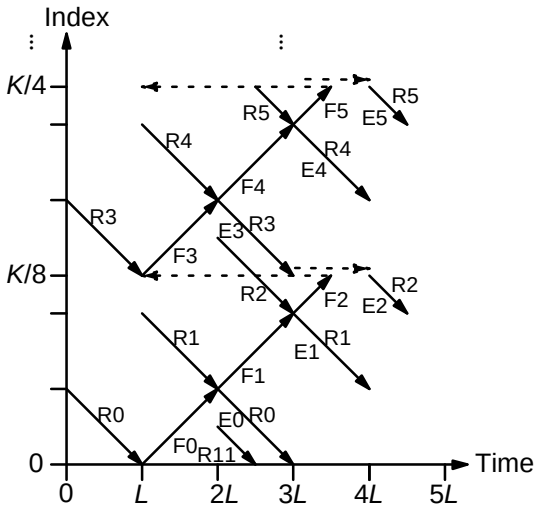

Figure 6: Timing diagram for $2L < K/8 \leq 3L$.

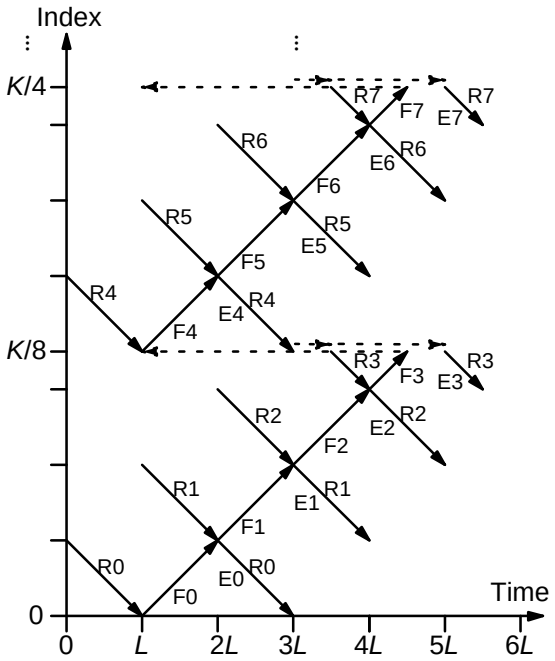

Figure 7: Timing diagram for $3L < K/8 \leq 4L$.

both forward and reverse SMs between iterations to help improve the reliability of the SMs.

Figures 5 to 7 show the timing diagram for various cases where $K/8 > L$. Only the trellis diagrams

for the first two MAP decoders are shown. Unlike the previous case, we read the SMs in reverse blocks of *L* and output them in reverse order. We pass forward SMs between iterations so that the starting forward SMs become reliable. This is indicated by the dashed arrow going backwards in time. As the reverse state metrics are reliable when used to calculate the extrinsic information there is no need to pass reverse SMs between iterations.

However, for $K/8 > 2L$ the final SMs of the third block, for example R3 in Figure 6, are stored and then passed to the previous MAP decoder. This is indicated by the dashed arrow going forwards in time. For $L < K/8 \leq 2L$ (Figure 5), the reverse SMs for the second reverse SM calculator are passed directly between MAP decoders.

For $L < K/8 \leq 3L$ (Figures 5 and 6), the first reverse SM calculator initially uses input data from the next MAP decoder for the last input block in time. This is why the RxI inputs shown in Figure 3 have an input width twice that of a single sample of six bits. This method avoids additional multiplexers to select the first reverse SMs from the next MAP decoder. However, this technique can not be used for $K/8 > 3L$ as the old extrinsic information will have been overwritten with new data. Instead, we use the previously stored reverse SMs as shown by the first forward arrow in Figure 7.

For $K/8 > 4L$ the timing diagrams are similar to that for $3L < K/8 \leq 4L$ (Figure 7). In all cases, we see that decoding time is equal to $K/8+2L$. We also need to add an additional 12 clock cycles for pipeline delay, to give a decoding time of $K/8+2L+12$ clock cycles for each half iteration. Two of the 12 clock cycles are for calculating the a priori information, eight clock cycles for the MAP decoder delay and two clock cycles for calculating the modified extrinsic information. One additional clock cycle is used to start the turbo decoder.

The number of turbo decoder half–iterations is given by *NI*, ranging from 0 to 63. $NI = 2I–1$ where *I* is the number of iterations. This is equivalent to 0.5 to 32 iterations. The decoder initially starts at half iteration NA = 0, increasing by one until *NI* is reached or at an earlier time if early stopping is enabled. The NA output can be used to select LIMZ and SCLZ values, which is useful for max–log–MAP decoding.

The turbo decoder speed f_d is given by

$$f_d = \frac{F_d K}{(NI + 1)(K/8 + 2L + 12) + 1} \qquad (3)$$

where F_d is the CLK frequency and *L* is the MAP decoder sliding window length. Table 5 gives the value of *L* depending on *K* and SLD. SLD = 0 can be used to increase decoder speed, while SLD = 1 should be used for high puncturing rates to increase performance.

Table 5: Sliding window length

K		*L*	
min	max	SLD=0	SLD=1
40	256	*K*/8 = 5..32	*K*/8 = 5..32
264	512	32	*K*/8 = 33..64
520	4800	32	64

For example, if F_d = 100 MHz and *I* = 5 (*NI* = 9) the decoder speed ranges from 20.7 Mbit/s for *K* = 112 and *L* = 14 to 71.0 Mbit/s for *K* = 4792 and *L* = 32.

An important parameter is LIMZ, the limit value for the extrinsic information. Extrinsic information is the “correction” term that the MAP decoder determines from the received data and *a priori* information. It is used used as *a priori* information for the next MAP decoding or half iteration. By limiting the correction term, we can prevent the decoder from making decisions too early, which improves decoder performance. The limit factor LIMZ should vary between 1 and 193. We recommend that 193 be used.

Another parameter that can used to adjust decoder performance is SCLZ which ranges from 1 to 32. The extrinsic information is scaled by SCLZ/32 followed by rounding to the nearest integer. Thus, when SCLZ = 32, no scaling is performed. For log–MAP decoding we recommend SCLZ = 29. For max–log–MAP decoding we recommend SCLZ = 23. The NA output can be used to adjust LIMZ and SCLZ with the number of iterations for optimum performance.

There are four decoder operation modes given by *M*. Mode *M* = 0 decodes a received block with a fixed number of iterations (given by *NI*). Modes 1 to 3 are various early stopping algorithms. Early stopping is used to stop the decoder from iterating further once it has estimated there are zero errors in the block. Mode 1 will stop decoding after an odd number of half–iterations. Mode 2 will stop decoding after an even number of half iterations. Mode 3 will stop after either an odd or even number of half iterations. Further details are given in the next section.

Interleaver parameters

The interleaving equation is given by

$$\pi(j) = (Pj + Q(j \bmod 4) + 3) \bmod K/2 \quad (4)$$

where j varies from 0 to K/2–1. Table 6 gives the formulas for $Q(j)$.

Table 6: Interleaver Parameters

j	*Q(j)*
0	0
1	$4Q_1$
2	$4Q_0P + 4Q_2$
3	$4Q_0P + 4Q_3$

The parameters P and Q_0 to Q_3 depend on the block length K. These values are given in the standard. P is an odd number while Q_0 to Q_3 can be odd or even numbers. To reduce interleaver complexity, we let $P(j) = Q(j) \bmod K/2$ for j = 1 to 3. We have that

$$Q(j) = D(j)K/2 + P(j) \quad (5)$$

where $D(j) = Q(j)$ div K/2. As four divides $Q(j)$ and K/2, four must also divide $P(j)$. That is

$$Q(j)/4 = D(j)K/8 + P(j)/4. \quad (6)$$

Since $P(j)/4 < K/8$ the decoder uses $P_j = Q(j)/4 \bmod K/8$ for j = 1 to 3 for the internal parameters. The term $Q(0)$ does not need to be externally input since it is always zero. We also let $P_0 = P$ div 2.

When KS[5:0] = 0, the byte length K/8 is input to K[12:3] and the interleaver parameters P_0 to P_3 are input to P0I[6:1] and P1I[11:2] to P3I[11:2], respectively. Internally, the two least significant bits PjI[1:0] = 0, $1 \le j \le 3$, and least significant bit P0I[0] = 1.

When KS[5:0] > 0, the internal data length selected by KS (equal to the Waveform ID) is used. Also, the internal interleaver parameters P_0 to P_3 for the data length from the standard are used. The inputs K[12:3], P0I[6:1], and P1I[11:2] to P3I[11:2] are ignored.

With four parallel MAP decoders, the interleaver addresses are given by

$$r(i,m) = \pi(i + mK/8) \bmod K/8 \quad (7)$$

$$s(i,m) = \pi(i + mK/8) \text{ div } K/8 \quad (8)$$

where $r(i,m)$ is the address of the depth K/8 RAM, $s(i,m)$ selects one of the four RAMs, i is the input RAM address from 0 to K/8–1, m is the input select address from 0 to 3 and $j = i + mK/8$ is the input address for the interleaver. We have that

$$r(i,m) = (P(i + mK/8) + Q((i + mK/8) \bmod 4) + 3) \bmod K/8$$
$$= (Pi + Q((i + mK/8) \bmod 4) + 3) \bmod K/8 \quad (9)$$

If one of the three following conditions

$$Q(j) \bmod K/8 = 0, \quad 1 \le j \le 3 \quad (10)$$

$$Q(1) \bmod K/8 = Q(3) \bmod K/8,$$
$$Q(2) \bmod K/8 = 0, \quad K/8 \bmod 2 = 0, \quad (11)$$

$$K/8 \bmod 4 = 0 \quad (12)$$

is satisfied, then $r(i,m) = \pi(i) \bmod K/8$. This implies that all four pairs of decoded bits will have the same write address. Thus, the decoded output XD[7:0] will be valid for all values of NA[5:0].

However, if all of the above conditions are not satisfied, then XD[7:0] will have different addresses for each pair of bits for odd NA[5:0] (even half iterations). Thus, XD[7:0] will only be valid for even NA[5:0] (odd half iterations) where non–interleaved decoding is performed.

Internal to the decoder, the property that $Q(j)$ is a multiple of four is used to split the interleaver memory into 16 separate memories is used. This allows correct read and write operations of the extrinsic information, regardless if the above conditions are satisfied.

Turbo Decoder Operation

After the START signal is sent, the decoder will read the received data at the CLK speed. It is assumed that the received data is stored in one synchronous read RAM of size (K/8)x24n, with n = 4 or 6 for rate 2/n decoding.

For input RiI[6j+5:6j][k], i = 0 to 5, j = 0 to 3, and k = 0 to K/8–1, the input data corresponds to code symbol 6(K/8)j+6k+i. The read address for input RiI is given by RA. The received data for A, B, Y_1, Y_2, W_1 and W_2 are input to R0I to R5I, respectively.

The received data ready signal RR goes high to indicate the data to be read from the address given by RA[9:0]. The parity check equations for the code are

$$(1 + D + D^2 + D^4)A + (1 + D^2 + D^4)B + (1 + D^3 + D^4)Y = 0 \quad (13)$$

$$(1 + D^2 + D^3 + D^4)A + (1 + D + D^2 + D^4)B + (1 + D^3 + D^4)W = 0 \quad (14)$$

The decoder then iteratively decodes the received data for NI+1 half iterations, rereading the received data for each half iteration for either $T_1 = 2L$ CLK cycles for K/8 = L or $T_2 = L((K/8–1) \text{ div } L)+L$ CLK cycles for K/8 > L. For K/8 $\le$ 3L, the signal RR goes high for either T_1 or T_2 CLK cycles while data is being output. For K/8 > 3L, RR is high for T_2–L CLK cycles and then goes high again at the (K/8)th CLK cycle for T_2–K/8 CLK cycles.

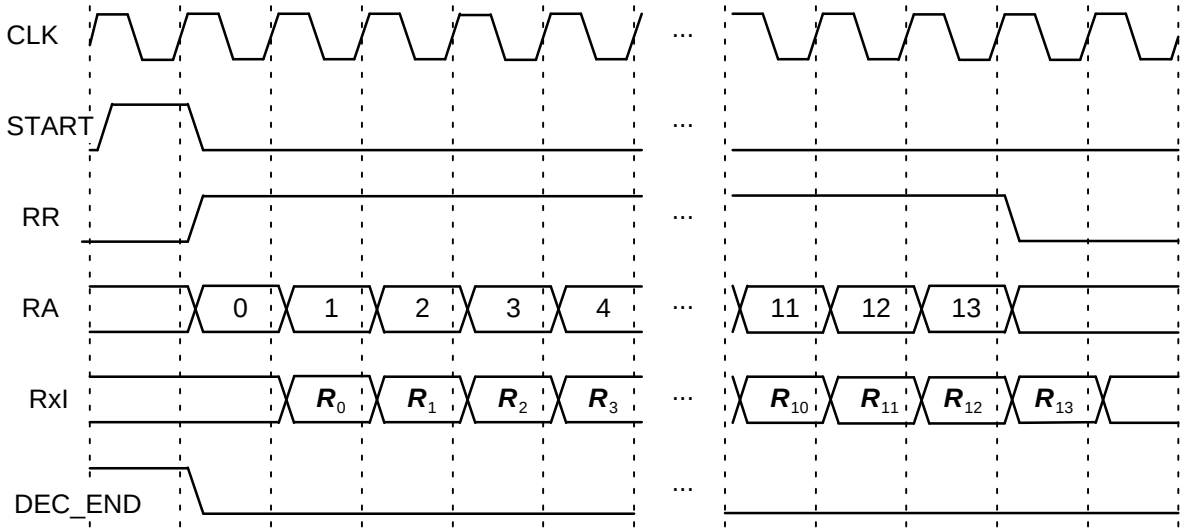

Figure 8: Turbo Decoder Input Timing (K = 112, L = 14, first half iteration).

Figure 8 illustrates the decoder timing where the data is input during the first half iteration. The input $\boldsymbol{R}_k$, for k = 0 to K/8–1 corresponds to input data RiI[23:0][k], i = 0 to 5.

Note that while DEC_END is low (decoding is being performed), the START signal is ignored, except for the last clock cycle before DEC_END goes high. If the code is being changed, then START must wait until DEC_END goes high, otherwise, it can go high in the last clock cycle.

A synchronous reset is also provided. All flip flops in the turbo decoder are reset during a low to high transition of CLK while RST is high.

The decoded data is output during the last half–iteration on XD[7:0]. That is, decoded data is output 8–bits every CLK cycle. For k = 0 to K/8–1 and j = 0 to 3 we have XD[2j] = $\hat{A}_{jK/8+k}$ and XD[2j+1] = $\hat{B}_{jK/8+k}$. The signal XDR goes high for K/8 CLK cycles while the block is output. If NI is even (odd half iterations), the block is output in reverse block sequential order. To dereverse the decoded data, the output XDA[9:0] needs to be used as the write address to a buffer RAM.

For NI odd (even half iterations), the block is output in reverse block interleaved order. To dereverse and deinterleave the block, the output XDA[9:0] is used as the write address to a buffer RAM. Note that this is only valid if one of the conditions in (10) to (12) are satisfied.

The bus ERR[7:0] is a channel error estimator output. It is the exclusive OR of XD[7:0] and the sign bits of R0I[23:0] and R01I[23:0], i.e., bits R0I[6j+5] and R1I[6j+5] for j = 0 to 3.

The DEC_END signal is low during decoding. At the end of decoding, DEC_END goes high. Figure 9 illustrates the decoder timing where data is output on the last half iteration. After startup, the maximum number of clock cycles for decoding is (NI+1)(K/8+2L+12).

During the last half iteration the decoded and error data are stored into the interleaver memory. This occurs correctly for all half iterations, unlike the XD output while DEC_END = 0. Once decoding has been completed, the input XDE can be used to sequentially clock the decoded and error data from from the interleaver memory (regardless of the number of iterations). XDE is disabled while the decoder is iterating. Figure 10 shows the decoder timing when XDE is used.

The early stopping algorithm uses the magnitude of the extrinsic information to determine when to stop. As the decoder iterates, the magnitudes generally increases in value as the decoder becomes more confident in its decision. By comparing the smallest magnitude of a block with threshold ZTH, we can decide when to stop. If the smallest magnitude is greater than ZTH, i.e., not equal or less than ZTH, the decoder will stop iterating if early stopping has been enabled.

Since the last half iteration is used to store the decoded data into the interleaver memory, the decoder performs an extra half iteration once the threshold has been exceeded.

Increasing ZTH will increase the average number of iterations and decrease the BER. Decreasing ZTH will decrease the average number of iterations and increase the BER. In general, higher values of SNR will decrease the number of iterations. A value of ZTH = 23 was found to give a good trade off between the average number of iterations and BER performance.

For high SNR operation early stopping can lead to significantly reduced power consumption,

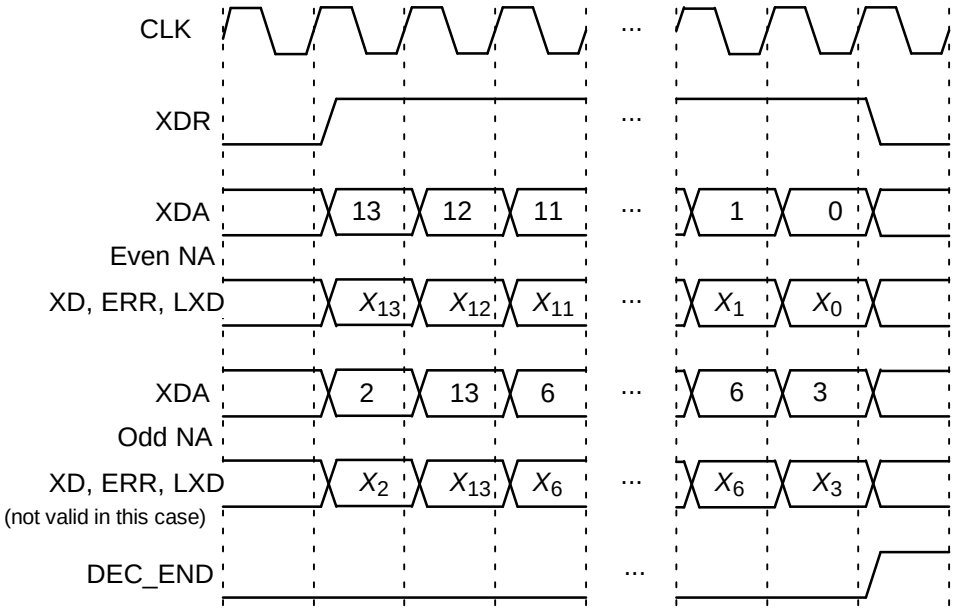

Figure 9: Turbo Decoder Output Timing (K = 112, L = 14).

since most blocks will be decoded in one or two iterations.

LXD Output

The output LXD[15:0] is an estimate of the logarithm of the probability of the four decoded symbol outputs (four bits for each symbol). In the probability domain, this value ranges from 0.25 to 1 as there are four symbols. In the log domain, this corresponds to a range from 0 to 14.

The LXD output is formed by taking the min* operation (equivalent to the max* operation, but in the minus log domain) of the log–likelihood ratios of each of the four symbols, where the most likely symbol is used as the reference symbol. A fixed look–up table for the correction term in the min* operation is used to allow operation at high signal to noise ratio (otherwise, due to finite quantisation, all LXD values would equal zero).

The LXD output can be used to determine the reliability of a decoded output by summation of the K/2 LXD values. Thus, one can compare the reliability of the decoded data for different conditions, choosing the decoded data that has the highest summation of LXD values.

Simulation Software

Free software for simulating the PCD04D4 turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with “pcd04d4sim request” in the subject header. The software uses an exact functional simulation of

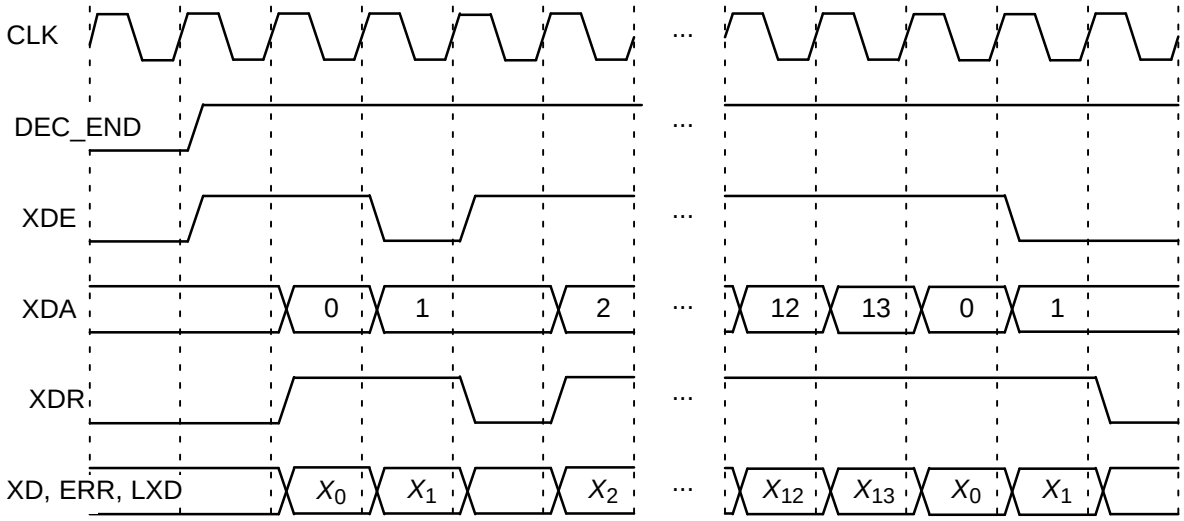

Figure 10: XDE Timing (K = 112).

the PCD04D4 turbo decoder, including all quantisation and limiting effects.

After unzipping pcd04d4sim.zip, there should be pcd04d4sim.exe and code.txt. The file code.txt contains the parameters for running pcd04d4sim. These parameters are

`kt`	No. of data bits (1)
`nt`	No. of coded bits (2 or 3)
`m`	Encoder memory (2 to 4)
`g0`	1st divisor polynomial of CC in octal
`g1`	2nd divisor polynomial of CC
`g2`	1st numerator polynomial of CC
`g3`	2nd numerator polynomial of CC
`q`	Number of quantisation bits (1 to 6)
`EbNomin`	Minimum E_b/N_0 (in dB)
`EbNomax`	Maximum E_b/N_0 (in dB)
`EbNoinc`	E_b/N_0 increment (in dB)
`optC`	Input scaling parameter (normally 0.5)
`ferrmax`	Number of frame errors to count
`Pfmin`	Minimum frame error rate (FER)
`Pbmin`	Minimum bit error rate (BER)
`NI`	Number of half iterations–1 (0 to 63)
`SLD`	MAP decoder delay select (0 or 1)
`LIMZ`	Extrinsic information limit (1 to 193)
`SCLZ`	Extrinsic information scale (1 to 32)
`M`	Stopping mode (0 to 4)
`ZTH`	Extrinsic info. threshold (0 to 255)
`KS`	Data length select (–1 to 22, 32 to 39)
`K`	Block length (40 to 8128)
`P0`	1st interleaver parameter (odd number)
`P1`	2nd input interleaver parameter
`P2`	3rd input interleaver parameter
`P3`	4th input interleaver parameter
`Q0`	1st DVB–RCS2 interleaver parameter
`Q1`	2nd DVB–RCS2 interleaver parameter
`Q2`	3rd DVB–RCS2 interleaver parameter
`Q3`	4th DVB–RCS2 interleaver parameter
`LOGMAP`	Log–MAP decoding (MODE0, 0 or 1)
`enter_C`	Enter external C (y or n)
`C`	C (0 to 11)
`state`	State file (0 to 2)
`s1`	Seed 1 (1 to 2147483562)
`s2`	Seed 2 (1 to 2147483398)
`out_screen`	Output data to screen (y or n)
`read_x`	Use external information data (y or n)
`read_r`	Use external received data (y or n)
`out_dir`	Output directory
`in_dir`	Input directory

The parameter `optC` is used to determine the "optimum" values of A and C. The "optimum" value of A is

$$A = \frac{\text{optC}(2^{q-1} - 1)}{\text{mag}(\sigma)} \quad (15)$$

where σ^2 is the normalised noise variance given by (2) and mag(σ) is the normalising magnitude resulting from an auto–gain control (AGC) circuit. We have

$$\text{mag}(\sigma) = \sigma\sqrt{\frac{2}{\pi}}\exp\left(\frac{-1}{2\sigma^2}\right) + 1 - 2Q\left(\frac{1}{\sigma}\right) \quad (16)$$

where $Q(x)$ is the error function given by

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}}\exp\left(\frac{-t^2}{2}\right)dt. \quad (17)$$

Although mag(σ) is a complicated function, for high signal to ratio (SNR), mag(σ) $\simeq$ 1. For low SNR, mag(σ) $\simeq \sigma\sqrt{2/\pi} \simeq 0.798\sigma$. That is, an AGC circuit for high SNR has an amplitude close to the real amplitude of the received signal. At lower SNR, the noise increases the estimated amplitude, since an AGC circuit averages the received signal amplitude.

For the "optimum" A, we round down the value of C given by (1) to the nearest integer. If LOGMAP = MODE0 = 0 then C is forced to 0. For LOGMAP = 1, if C is greater than 11, C is limited to 11. If C = 1, C is rounded down to 0. If C = 2 or 3, C is rounded up to 4. An external value of C can be input by setting `enter_C` to `y`.

For KS > 0, internal interleaver parameters as specified by the standard are used. For KS = 0, the parameters P = P0 and $P(1)$ = P1 to $P(3)$ = P3 are used. For KS < 0, P = P0 and Q_0 = Q0 to Q_3 = Q3 are used. The software then calculates $P(1)$ to $P(3)$ for use by the interleaver.

The simulation will increase E_b/N_0 (in dB) in `EbNoinc` increments from `EbNomin` until `EbNomax` is reached or the frame error rate (FER) is below or equal to `Pfmin` or the bit error rate (BER) is below or equal to `Pbmin`. Each simulation point continues until the number of frame errors is equal to `ferrmax`. If `ferrmax` = 0, then only one frame is simulated.

An optional Genie aided stopping mode can be selected by setting M = 4. This will stop the decoder from further iterations when the Genie has detected there are no errors compared to the transmitted data. This allows a lower performance bound to be simulated, allowing fast simulations for various configurations at low bit error rates. This option is not available in the decoder core.

When the simulation is finished the output is given in, for example, file k472.dat, where K = 472. The first line gives the E_b/N_0 (`Eb/No`), the number of frames (`num`), the number of bit errors in the frame (`err`), the total number of bit errors (`berr`), the total number of frame errors (`ferr`), the aver-

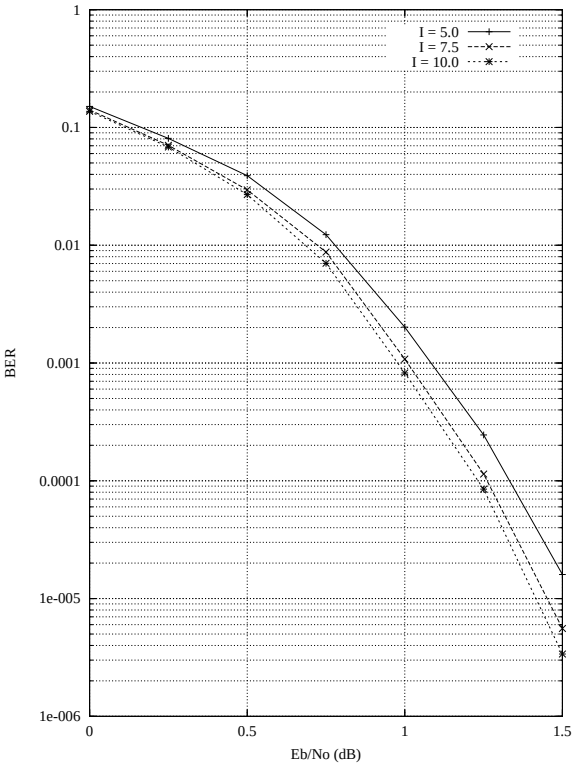

Figure 11: Rate 1/3 BER performance with *K* = 472, max–log–MAP and auto–stopping.

age number of iterations (`na`), the average BER (`Pb`) and the average FER (`Pf`). Following this, the number of iterations, `na`, `berr`, `ferr`, `Pb`, and `Pf` are given for each half iteration.

The following file was used to give the simulation results shown in Figure 11 for *K* = 472 (KS = 4) and max–log–MAP decoding. Auto–stopping was used with up to 10 iterations. When iterating is stopped early, the `nasum` (`2*num*na`), `berr` and `ferr` results at stopping are copied for each half iteration to the maximum iteration number. Figure 12 shows the average number of iterations with E_b/N_0.

```
{kt nt m g0 g1 g2 g3}
 1  3  4 17 15  3  6
{q EbNomin EbNomax EbNoinc optC}
 6 0.00    1.50    0.25    0.45
{ferrmax Pfmin Pbmin}
 128     1e-99 1e-5
{NI  SLD  LIMZ  SCLZ  M  ZTH}
 19  0    193   21    3  23
{KS K   P0 P1 P2 P3 Q0 Q1 Q2 Q3}
 4  112 9  8  48 16  2  2  8  0
{LOGMAP enter_C C}
 0      n       6
{state s1    s2    out_screen}
 0     12345 67890 y
{read_x read_r out_dir in_dir}
 n      n      output  input
```

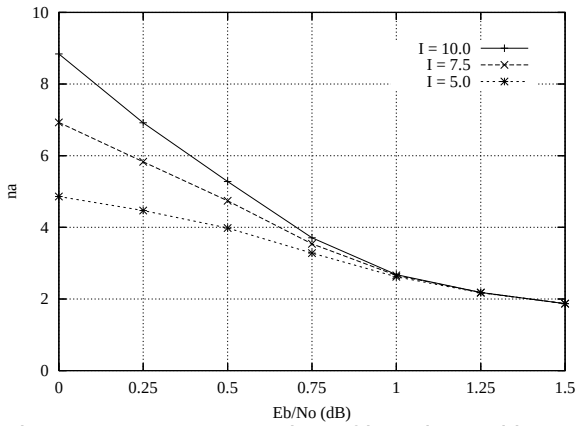

Figure 12: Average number of iterations with *R* = 1/3, *K* = 472, max–log–MAP and auto–stopping.

Figure 13 shows the performance of the turbo decoder for various block sizes, log–MAP and max–log–MAP decoding. For max–log–MAP decoding, optC = 0.5 and SCLZ = 22 were used for *K* = 112 and 304. For *K* = 472 and 680, optC = 0.51 and SCLZ = 21 were used. Typically, log–MAP gains an additional 0.07 to 0.1 dB coding gain at a BER of 10^{-5} compared to max–log–MAP.

Table 7 gives the parameters optC, A, C and SCLZ that were found to give the best performance for various code rates at a bit error rate (BER) of around 3×10^{-2} for 10 iterations (NI = 19), *R* = 1/2, M = 3, ZTH = 23, LIMZ = 193 and log–MAP decoding. Using these parameters for higher E_b/N_0 values should result is very little performance degradation.

Table 7: Simulation parameters

K	E_b/N_0 (dB)	optC	A	C	SCLZ	BER 10^{-2}
112	1.40	0.50	14.08	5	31	3.02
304	1.10	0.45	12.54	4	31	2.43
472	0.97	0.45	12.47	5	32	2.96
680	0.90	0.44	12.16	4	32	2.38

The `state` input can be used to continue the simulation after the simulation has been stopped, e.g., by the program being closed or your computer crashing. For normal simulations, `state` = 0. While the program is running, the simulation state is alternatively written into state1.dat and state2.dat. Two state files are used in case the program stops while writing data into one file. To continue the simulation after the program is stopped follow these instructions:
1) Copy the state files state1.dat and state2.dat. This ensures you can restart the program if a mistake is made in configuring code.txt.

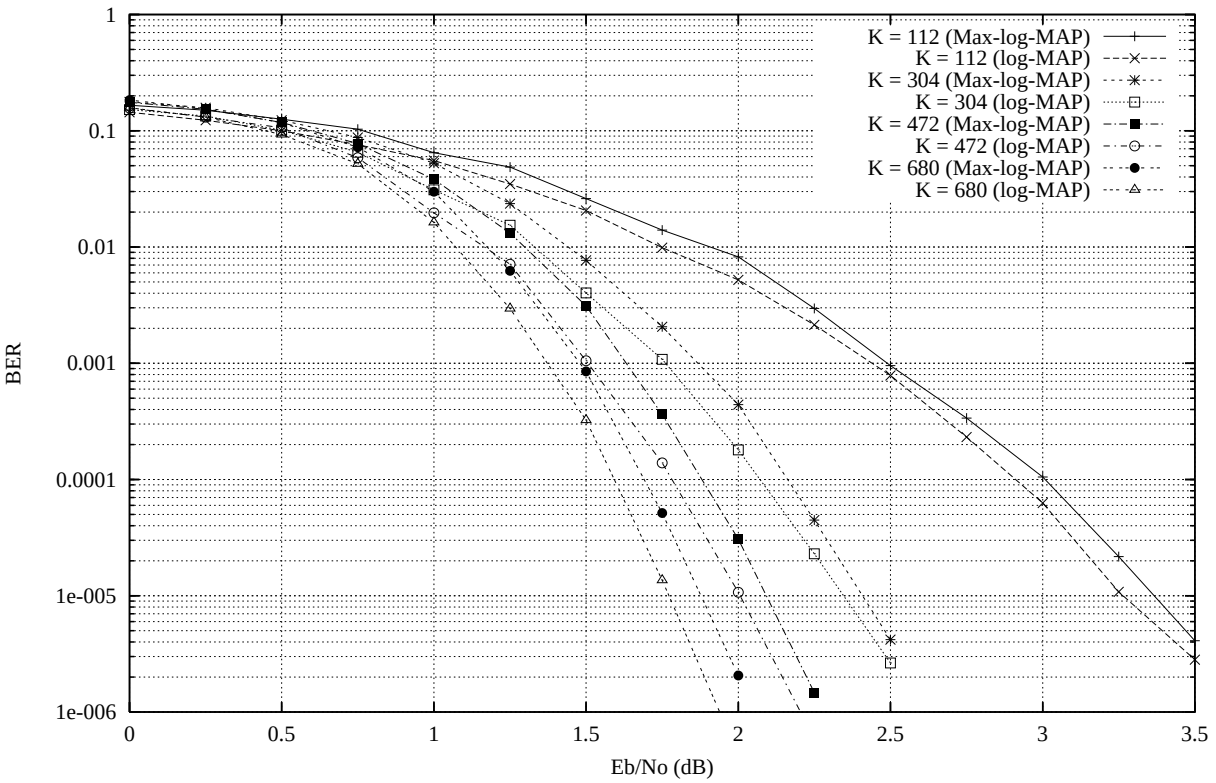

Figure 13: Rate 1/2 performance with various block sizes, auto–stopping (ZTH = 23), log–MAP and max–log–MAP.

2) Examine the state files and choose one that isn't corrupted.
3) Change the state parameter to 1 if state1.dat is used or 2 if state2.dat is used.
4) Restart the simulation. The output will be appended to the existing k(K).dat file.
5) After the simulation has been completed, make sure that `state` is changed back to 0.

The software can also be used to encode and decode external data. To encode a block `x_(K).dat` in the directory given by `in_dir`, set `read_x` to `y`, e.g., `x_472.dat` in directory `input` (each line contains one bit of data). The data is input in the order A(0) B(0) ... A(K/2–1) B(K/2–1).

The encoded stream `y_(K).dat` will be output to the directory given by `out_dir`, e.g., `y_472.dat` to directory `output`. The encoded data is output in the order A(0) B(0) ... A(K/2–1) B(K/2–1) $Y_1(0)$ $Y_2(0)$ ... $Y_1(K/2–1)$ $Y_2(K/2–1)$ $W_1(0)$ $W_2(0)$ ... $W_1(K/2–1)$ $W_2(K/2–1)$.

To decode data, place the received block of data in file `r_(K).dat` in directory `in_dir` and set `read_r` to `y`. The decoded data is output to `xd_(K).dat` in directory `out_dir`. The file `r_(K).dat` has in each line R[i,j], i = 0 to 3 or 5 from j = 0 to *K*/2–1, e.g., the first three lines of rate 1/2 data could be

```
-31   1 -25  27
-31  12  -4   9
 11  31  31   2
```

The input data is of the form

R[i,j] = A*(1–2*Y[i,j]+N[i,j])

where A is the signal amplitude, Y[i,j] is the coded bit, and N[i,j] is white Gaussian noise with zero mean and normalised variance σ^2. The magnitude of R[i,j] should be rounded to the nearest integer and be no greater than 2^{q-1}–1. If `read_r` = `y`, then *C* is externally input via `c`.

Ordering Information

SW–PCD04D4–SOS (SignOnce Site License)
SW–PCD04D4–SOP (SignOnce Project License)
SW–PCD04D4–VHD (VHDL ASIC License)

All licenses include EDIF and VHDL cores. The VHDL cores can only be used for simulation in the SignOnce and University licenses. The University license is only available to tertiary educational institutions such as universities and colleges and is limited to n instantiations of the core.

The SignOnce and ASIC licenses allows unlimited instantiations.

Note that *Small World Communications* only provides software and does not provide the actual devices themselves. Please contact *Small World Communications* for a quote.

References

[1] EBU–UER and DVB, "Digital video broadcasting second generation interactive satellite system (DVB–RCS2) Part 2: Lower layers for satellite standard," ETSI EN 301 545–2 V1.1.1, Jan. 2012.

[2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT–20, pp. 284–287, Mar. 1974.

[3] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub–optimal MAP decoding algorithms operating in the log domain," *ICC'95*, Seattle, WA, USA, pp. 1009–1013, June 1995.

[4] M. C. Reed and J. A. Asenstorfer, "A novel variance estimator for turbo–code decoding," *Int. Conf. on Telecommun.*, Melbourne, Australia, pp. 173–178, Apr. 1997.

Small World Communications, 6 First Avenue, Payneham South SA 5070, Australia.
info@sworld.com.au ph. +61 8 8332 0319
http://www.sworld.com.au fax +61 8 7117 1416

Version History

- 0.00 22 January 2015. Preliminary product specification.
- 1.00 2 February 2015. First official release. Added performance curves and parameters.
- 1.01 11 February 2015. Added option to input $P(1)$ to $P(3)$ for BER simulation software. Corrected description of Q0 to Q3 parameters. Added description of x_(K).dat and y_(K).dat files. Corrected r_(K).dat description. Added `enter_c` description.
- 1.02 22 February 2015. Updated Table 1.
- 1.03 22 April 2015. Deleted large–log–MAP option. Changed range of C values for small–log–MAP (now called log–MAP). Improved log–MAP performance. Simplified internal RA generation.
- 1.04 30 May 2015. Minor corrections.
- 1.05 12 December 2018. Clarified XD[7:0] output.