PCD04CH Features

Turbo Decoder

- 16 state CCSDS cmpatible
- Rate 1/2, 1/3, 1/4 or 1/6
- Interleaver sizes from 1784 to 16056 bits
- Includes optional automatic synchronisation to non–inverted or inverted sync marker, optional descrambler and ping–pong input memory
- Up to 343 MHz internal clock (log–MAP)
- Up to 33.3 Mbit/s with 5 decoder iterations
- 6–bit signed magnitude input data
- Log–MAP or max–log–MAP constituent decoder algorithms
- Up to 32 iterations in 1/2 iteration steps
- Power efficient early stopping
- Optional scaling and limiting of extrinsic information
- Full estimated channel error output
- Asynchronous logic free design
- Free simulation software
- Available as VHDL core for AMD–Xilinx FPGAs under SignOnce IP License. ASIC, Intel/Altera, Lattice and Microsemi cores available on request.

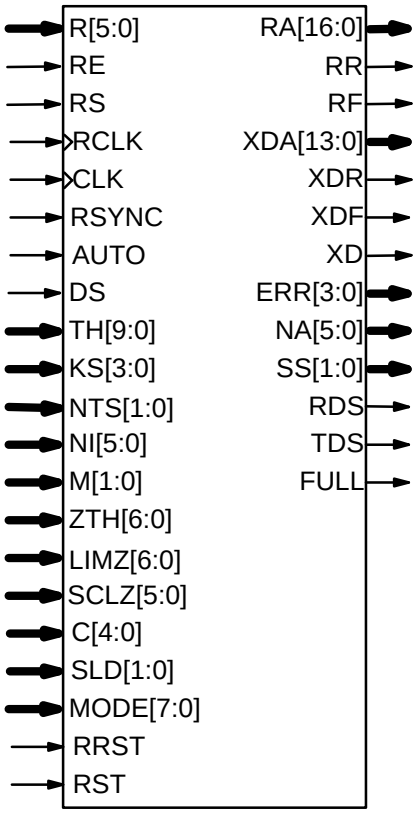

Figure 1: PCD04CH schematic symbol.

Introduction

The PCD04CH is a 16 state CCSDS [1] compatible parallel concatenated error control turbo decoder. Interleaver sizes from 1784 to 16056 bits in multiples of 1784 can be implemented. Nominal turbo code rates of $R = 1/n$ = 1/2, 1/3, 1/4 or 1/6 can be selected. The sequential data is terminated with a tail using both data and parity information. The interleaved data is terminated with a tail using parity data only. The input block and interleaver size is K. The number of coded bits is $N = (K+4)n$.

The decoder is able to automatically synchronise to a length $32n$ sync marker. After synchronisation the received data can be optionally descrambled and then decoded. The total received length is $(K+36)n$.

The MAP04V MAP decoder core is used with the PCD04CH core to iteratively decode the turbo code. The Log–MAP algorithm for maximum performance or the max–log–MAP algorithm for minimum complexity can be selected. Max–log–MAP decoding increases speed by about 45% compared to log–MAP.

The sliding block algorithm is used with sliding block lengths of 32, 64 or 128. 6–bit quantisation is used for near optimum performance. The extrinsic information can be scaled and limited with each half iteration, improving performance with max–log–MAP decoding. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance.

Figure 1 shows the schematic symbol for the PCD04CH decoder. The EDIF core can be used with Xilinx Foundation or Integrated Software Environment (ISE) software. The VHDL core can be used with Xilinx ISE or Vivado software. Custom VHDL cores can be used in ASIC designs.

Table 1 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 1: Performance of Xilinx parts.

Xilinx Part	T_{cp} (ns)	f_d* (Mbit/s)
XC7A35T–1	10.438	9.30
XC7A35T–2	8.513	11.40
XC7A35T–3	7.556	12.85
XC7S50–1	10.386	9.34
XC7S50–2	8.519	11.39
XC7K70T–1	6.887	14.09
XC7K70T–2	5.619	17.28
XC7K70T–3	5.106	19.01
XCKU035–1	5.632	17.24
XCKU035–2	4.770	20.35
XCKU035–3	4.064	23.89
XCKU3P–1	3.728	26.04
XCKU3P–2	3.174	30.59
XCKU3P–3	2.913	33.33

*large log–MAP, 5 iterations, *K*=8920, SLD=1.

Table 2 shows the resources used for Kintex–7 devices with SLD[1] = 0. LM stands for log–MAP. The complexity for Virtex–5, Virtex–6, Spartan–6, 7–Series, UltraScale and UltraScale+ devices are similar to that for Kintex–7. The MODE[7:0] inputs can be used to select various decoder implementations. An input RAM and interleaver RAM are also used requiring from 8 to 71 18KB BlockRAMs. No other resources are used.

Table 2: Resources used

Log MAP	Turbo Rates	SLD	LUTs
Max	1/2–1/6	0–2	5124
Small	1/2–1/6	0–2	6850
Large	1/2–1/6	0–2	7202
Small	1/2–1/3	0–2	5532
Small	1/2–1/6	0–1	6587

Signal Descriptions

AUTO Automatic Synchronisation Select
C MAP Decoder Constant
  0–9 (MODE[1] = 0)
  0–17 (MODE[1] = 1)
CLK System Clock
DS Descrambler Select
ERR Estimated Error
FULL Decoder Full (new data not accepted)
KS Data Length Select (0 to 8)
  0 = Length 1784 (CCSDS)
  1 = Length 3568 (CCSDS)
  2 = Length 7136 (CCSDS)
  3 = Length 8920 (CCSDS)
  4 = Length 5352
  5 = Length 10704
  6 = Length 12488
  7 = Length 14272
  8 = Length 16056
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 = 2*I*–1 where *I* is number of iterations
LIMZ Extrinsic Information Limit (1–127)
NTS Turbo Code Rate Select (0–3)
  0 = Rate 1/2
  1 = Rate 1/3
  2 = Rate 1/4
  3 = Rate 1/6
R Received Data
RA Received Data Address
RCLK Received Data Clock
RDS Received Data Start Internal
RE Received Data Enable
RF Received Data Finish
RR Received Data Ready
RRST Synchronous Reset for RCLK
RS Received Data Start
RST Synchronous Reset for CLK
RSYNC RCLK and CLK equal
SCLZ Extrinsic Information Scale (1–32)
SLD MAP Decoder Delay
  0 = delay 138
  1 = delay 266
  2 = delay 522
SS Synchronisation State (0–2)
  0 = Searching for sync marker
  1 = Synchronised, sync threshold met
  2 = Synchronised, sync threshold not met
TDS Turbo Decoder Start Internal
TH Synchronisation Threshold
XD Decoded Data
XDA Decoded Data Address
XDF Decoded Data Finish
XDR Decoded Data Ready
ZTH Early Stopping Threshold (1–127)

Table 3 describes each of the MODE[7:0] inputs that are used to select various decoder implementations. Note that RSYNC and MODE[7:0] are “soft” inputs and should not be connected to input pins or logic. These inuts are designed to

minimise decoder complexity for the configuration selected.

Turbo Decoder Parameters

For optimal performance, the maximum a posteriori (MAP) [2] constituent decoder is used which is dependent on the signal to noise ratio (SNR). Unlike other turbo decoders with suboptimum 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 enabled when MODE[0] is high.

Table 3: MODE selection

Input	Description	
MODE[0]	0 = max–log–MAP 1 = log–MAP	
MODE[1]	0 = small log–MAP (C4 = 0) 1 = large log–MAP	
MODE[2]	0 = rate 1/2–1/3 1 = rate 1/2–1/6	
MODE[6:3]	18KB BlockRAMs	
	MODE[2] = 0	MODE[2] = 1
0 = $K \leq 1784$	5	8
1 = $K \leq 3568$	9	16
2 = $K \leq 5352$	14	24
3 = $K \leq 7136$	18	32
4 = $K \leq 8920$	23	40
5 = $K \leq 10704$	27	48
6 = $K \leq 12488$	32	56
7 = $K \leq 14272$	35	63
8 = $K \leq 16056$	40	71
MODE[7]	0 = SLD ≤ 1 1 = SLD ≤ 2	

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

$$\sigma^2 = \left(2mR_c\frac{E_b}{N_0}\right)^{-1}. \quad (2)$$

E_b/N_0 is the energy per bit to single sided noise density ratio and R_c = 1/(n(1+36/K)) is the code rate. C should be rounded to the nearest integer

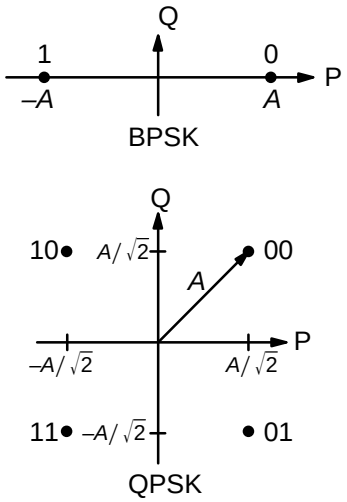

Figure 2: BPSK and QPSK signal sets.

and limited to be no higher than 17 with MODE[1] high and 9 with MODE[1] low. Max–log–MAP [3] operation occurs when C = 0. Due to quantisation effects, C = 1 is equivalent to C = 0. Max–Log–MAP operation is also enabled when MODE[0] is low.

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

The value of A directly corresponds to the 6–bit signed magnitude inputs (shown in Table 4). The 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 15 (which has a range between 15 and 16).

Table 4: Quantisation for R[5:0].

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
32	100000	–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

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 RT[2:0], where RT[2] is the sign bit, we have R[5:3] = RT[2:0] and R[2:0] = 4 in decimal (100 in binary). For punctured input data, all bits must be zero, e.g., R[5:0] = 0.

Example 1: Rate 1/3 K = 1784 BPSK code operating at E_b/N_0 = 0.3 dB. From (2) we have σ^2 = 1.428130. Assuming A = 16 we have from (1) that C = 11 to the nearest integer.

The number of turbo decoder half–iterations is give 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 turbo decoder speed f_d is given by

$$f_d = \frac{F_d}{(\text{NI} + 1)(1 + (L + M)/K)} \tag{3}$$

where F_d is the CLK frequency, L is the MAP decoder delay in bits (equal to either 138, 266 or 522), M = 0 for log–MAP and M = 1 for max–log–MAP decoding. The three delays indicate the sliding block length used in the MAP decoder, either 32, 64 or 128, respectively. For short block lengths L = 138 should be used to increase decoder speed, while L = 266 should be used for larger block sizes to increase performance. For highly punctured codes, for example a turbo code rate of rate 7/8, L = 522 should be used. This parameter can be selected with the SLD input.

For example, if F_d = 100 MHz, I = 5 (NI = 9), L = 266 and M = 0, the decoder speed ranges from 8.7 Mbit/s for K = 1784 to 9.8 Mbit/s for K = 16056.

An important parameter is LIMZ, the limit factors 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 127, although we recommend that 96 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. Thus, when SCLZ = 32, no scaling is performed. For log–MAP decoding we recommend SCLZ = 28 for rate 1/2 and 31 for rate 1/3 to 1/6. 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.

Synchronisation

The decoder includes an automatic synchronisation circuit to the length P = 32n sync marker. The received data R[5:0] is input to a programmable length correlator, with a maximum length of 192. The correlator calculates the correlation

$$C_i = \sum_{j=0}^{P-1} r_{i+j}\,(1 - 2s_j) \tag{4}$$

where s_j is the binary sync marker sequence and the received data is

$$r_i = A(1 - 2y_i + n_i) \tag{5}$$

where A is the no noise amplitude, y_i is the transmitted binary sequence and n_i is Gaussian noise with mean 0 and normalised variance σ^2.

If we assume that $y_i = s_i$ for $0 \le i \le P$–1, that is the sync marker is synchronised with the correlation circuit for i = 0, then C_0 has a Gaussian distribution with mean PA and variance $PA^2\sigma^2$. For the non–synchronised case we assume that y_i is a random variable equal to 0 or 1 with equal probability. Using the central limit theorem, this implies that C_i, $i > 0$, is an approximate Gaussian variable with mean 0 and variance $PA^2(\sigma^2+1)$ since the mean and variance of 1–2y_i is equal to 0 and 1, respectively.

The approximate optimum threshold c_t to decide whether a sequence is synchronised is determined when

$$p_{C_0}(c) = p_{C_i}(c) \tag{6}$$

where $p_{C_0}(c)$ is the synchronised probability distribution function (PDF) and $p_{C_i}(c)$ is the non–synchronised PDF. We have

$$p_{C_0}(c) = \frac{1}{\sqrt{2\pi P}A\sigma}\exp\left(\frac{-(c - PA)^2}{2PA^2\sigma^2}\right) \tag{7}$$

$$p_{C_i}(c) \approx \frac{1}{\sqrt{2\pi P(\sigma^2+1)}A}\exp\left(\frac{-c^2}{2PA^2(\sigma^2+1)}\right). \tag{8}$$

Solving (6), we obtain the threshold value

$$c_t = AP\left(\sigma^2 + 1 - \sqrt{(\sigma^2+1)\sigma^2(1 + \ln(1+1/\sigma^2)/P)}\right)$$

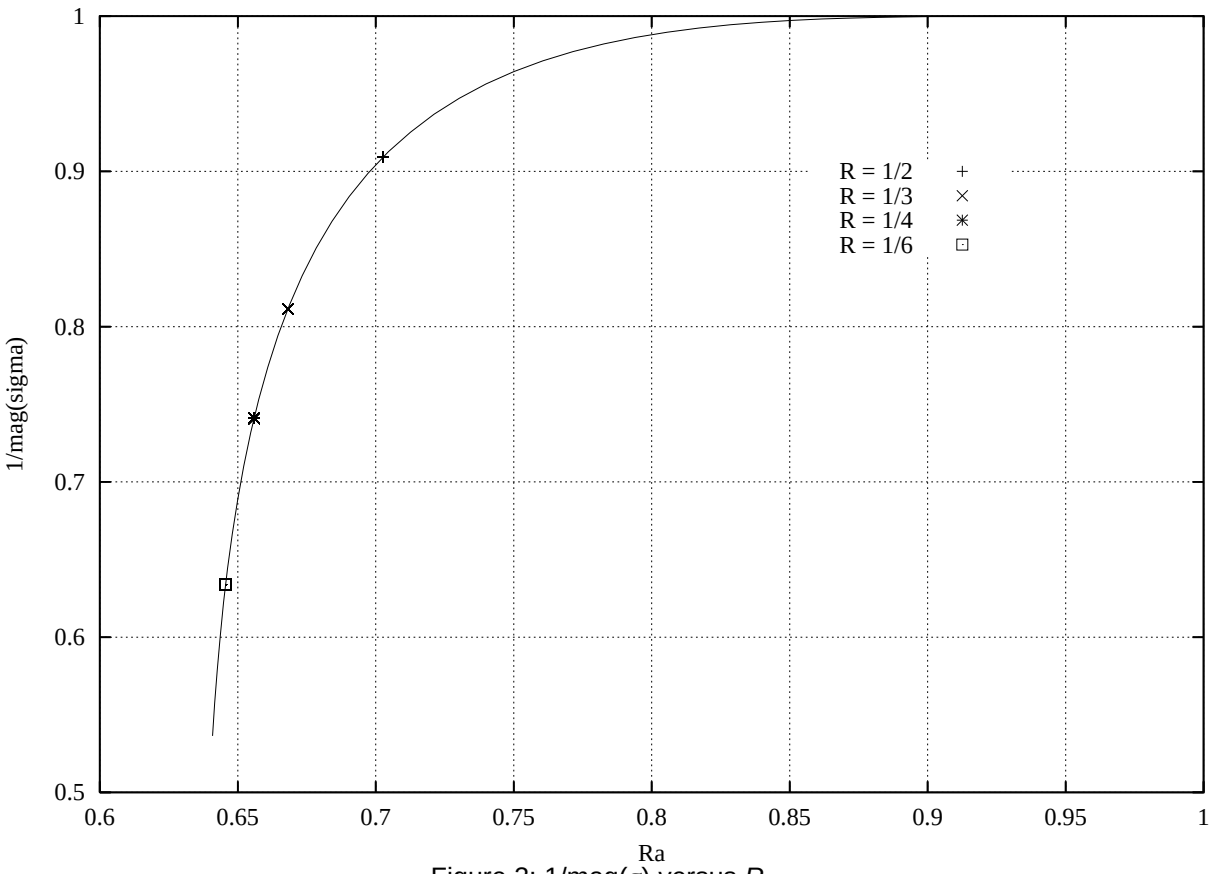

Figure 3: 1/mag(σ) versus R_a.

(9)

If we assume that A and σ^2 are not known, we can estimate these values by calculating the average of the square and magnitude of the input. That is

$$R_{rms}^2 = \frac{1}{T}\sum_{i=0}^{T-1} r_i^2 \tag{10}$$

and

$$R_{abs} = \frac{1}{T}\sum_{i=0}^{T-1} |r_i| \tag{11}$$

where $T > P$ is the length of a long sequence in which A and σ^2 can be assumed to be constant. For large T we have that

$$R_{rms}^2 \approx A^2(\sigma^2 + 1) \tag{12}$$

and

$$R_{abs} \approx A\,\mathrm{mag}(\sigma) \tag{13}$$

where

$$\mathrm{mag}(\sigma) = \sigma\sqrt{\frac{2}{\pi}}\exp\left(\frac{-1}{2\sigma^2}\right) + 1 - 2Q\left(\frac{1}{\sigma}\right) \tag{14}$$

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

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}}\exp\left(\frac{-t^2}{2}\right)dt. \tag{15}$$

By taking the ratio

$$R_a = \frac{R_{abs}^2}{R_{rms}^2} \approx \frac{\mathrm{mag}(\sigma)^2}{1+\sigma^2} \tag{16}$$

we obtain a function that is only dependent on σ^2. Thus, we can use a lookup table or curve fitting function that inputs R_a and outputs 1/mag(σ) and c_t/A. Using (13) we can calculate A by multiplying 1/mag(σ) with R_{abs} and c_t by multiplying c_t/A with A. If desired, (12) can be used to calculate σ^2, the E_b/N_0 of the channel using (2) and the MAP decoder constant C using (1).

Figures 3 and 4 plot 1/mag(σ) and c_t/A versus R_a, respectively. The points correspond to E_b/N_0 = 0.3, 0.6, 0.8 and 1.5 dB for R = 1/6, 1/4, 1/3 and 1/2, respectively.

Synchronisation Operation

Synchronisation has three states; *search* (SS = 0), *sync* (SS = 1) and *nosync* (SS = 2). In the *search* state a start signal is generated when $C_i > c_t$ (quantised as X > TH) and goes into the *sync* state. In the *sync* state, if X $\leq$ TH for the sync marker, the circuit goes into the *nosync* state, but still generates a start signal. If the following block still has X $\leq$ TH, the circuit goes back to the *search* state, otherwise it goes back to the *sync* state. That is, it takes two consecutive non–synchronisations to go back into the *search* state.

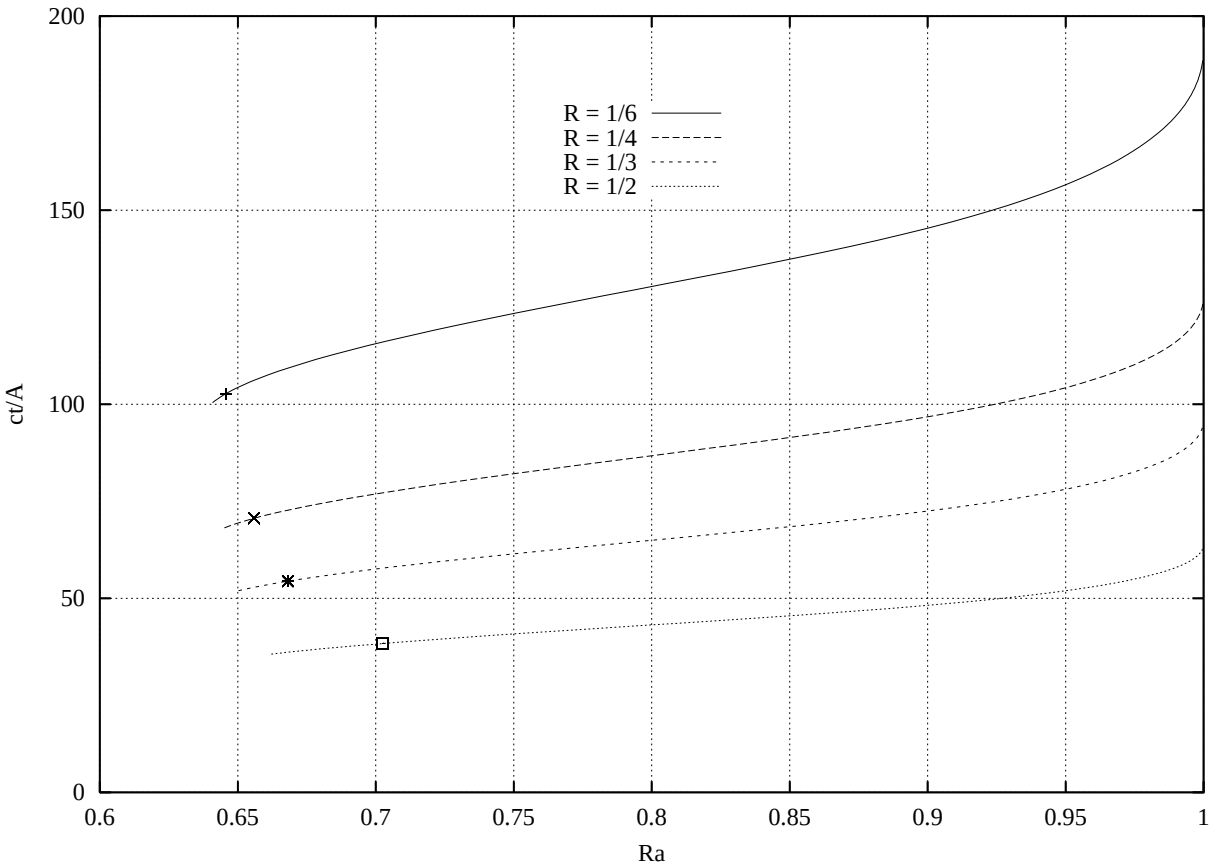

Figure 4: c_t/A versus R_a.

The decoder uses the anti–correlation value to detect and then correct for binary phase shift keying (BPSK) data that has been rotated by 180° (thus inverting the received data). That is, if the correlation value X > TH or X < –TH correlation is detected. If X < –TH, then the received data is inverted.

For rate 1/4 and 1/6 the sync markers are given by SM4 = (SM2,$\overline{\text{SM2}}$) and SM6 = (SM3,$\overline{\text{SM3}}$), respectively, where SM2 and SM3 are the sync markers for rate 1/2 and 1/3, respectively. This can cause synchronisation problems if the threshold is not correctly chosen.

For example, at infinite SNR we have $A = A_{\text{opt}}$ = 31optC. From the optC values given in Table 7, the recommended values of A_{opt} are 10.85, 10.85, 11.47 and 13.02 for rate 1/6, 1/4, 1/3 and 1/2, respectively. For rate 1/4, the sync marker correlation is $32n[[A]/8] = 128$, where $[x]$ rounds x to the nearest integer, $P = 32n = 128$ is the sync marker length, A = 10.85 and $[[A]/8] = 1$ is the quantised symbol correlation value.

Since SM4 = (SM2,$\overline{\text{SM2}}$), the first 64 symbols of the sync marker will be anti–correlated with the last 64 symbols of the sync marker. Assuming a correlation of zero for the other 64 symbols, the correlation is X = –1×64 = –64. Thus, values of TH less than 64 can cause a false early correlation to be detected from a negative correlation.

A solution to correct this problem is to use a higher value for TH when SS = 0 and the optimal value of TH when SS > 0. When SS > 0, the correlation X is only sampled at the expected time, so the problem with anti–correlation is avoided.

For no noise and $A = A_{\text{opt}}$, suggested values for TH are $0.75\times32n[[A]/8]-1$ and $0.5\times32n[[A]/8]-1$ for SS = 0 and SS > 0, respectively. That is, the TH values are

Table 5: TH values for $A = A_{\text{opt}}$ and no noise.

Rate	A_{opt}	$[[A_{\text{opt}}]/8]$	TH SS = 0	TH SS > 0
1/2	13.02	2	95	63
1/3	11.47	1	71	47
1/4	10.85	1	95	63
1/6	10.85	1	143	95

With noise and smaller values of A, different values of TH need to be used. For example, with rate 1/4, A = 8.19 and E_b/N_0 = 0.6 dB, the optimal value of TH = 71 should be used for SS > 0. For SS = 0, the no noise full correct correlation and incorrect anti–correlation values are $B = 32nA/8$ =

$4nA$ = 131.04 and C = –0.5×32nA/8 = –2nA = –65.52. Taking the average of the magnitudes we have TH = [(B–C)/2]–1 = [(4nA+2nA)/2]–1 = [3nA]–1 = 97 for SS = 0.

Since rate 1/2 and 1/3 don't have symmetric sync markers, the same optimum value of TH can be used for SS = 0 and SS > 0.

Input Scaling

To obtain the best performance from the decoder it is important that the input data be correctly scaled. We have

$$\mathrm{R[5:0]} = [r_i\, A_{\mathrm{opt}}/R_{\mathrm{abs}}] \tag{17}$$

where [x] rounds x to the nearest integer. Similarly, the values of c_t and C also need to be scaled and rounded.

$$\mathrm{TH[9:0]} = [c_t\, A_{\mathrm{opt}}/(8R_{\mathrm{abs}})] \tag{18}$$

$$\mathrm{C[4:0]} = [C\, A_{\mathrm{opt}}/R_{\mathrm{abs}}]. \tag{19}$$

In order to reduce the correlator complexity, the input R[5:0] is rounded to the three most significant bits. This implies c_t should be divided by eight, as shown in (18). The turbo decoder still uses the full 6–bit input.

At infinite SNR, (18) would give TH = [PA_{opt}/8] = 138 for rate 1/3. However, A = 11.47 would get rounded down to 11 and 11/8 = 1.375 would get rounded down to 1. Thus, the actual correlation would be 96, which would always be below the threshold! Thus, we have

$$\begin{aligned}\mathrm{TH[9:0]} = \min([c_t\, A_{\mathrm{opt}}/(8R_{\mathrm{abs}})],\\ P[[A_{\mathrm{opt}}]/8] - 1).\end{aligned} \tag{20}$$

Note that after rounding to the nearest integer, the input must be limited so that the maximum magnitude is not greater than 31, as shown in Table 4. If the demodulator automatic gain control (AGC) circuit normalises to a fixed absolute voltage (instead of power), then fixing this voltage to A_{opt} means that no scaling is required. However, rounding and limiting still need to be performed.

Descrambling

After synchronisation, the received data must be descrambled. The descrambler select DS input can be used to enable (DS=1) or disable (DS=0) descrambling.

The descrambler generator is an 8–bit right shift register with initial contents of c = 255 = 377_8. The descrambler is defined by the polynomial [1]

$$h(x) = x^8 + x^7 + x^5 + x^3 + 1 \tag{21}$$

where addition is modulo 2. However, the descrambler is implemented as

$$\begin{aligned}e(D) &= (D^8 h(D^{-1}) + 1)e(D)\\ &= (D + D^3 + D^5 + D^8)e(D)\end{aligned} \tag{22}$$

where $e(D) = e_0 + e_1 D + e_2 D^2 + ... + e_{N-1} D^{N-1}$ is the scrambled sequence and D is the delay operator. When e_i = 1 the received data input is inverted, otherwise the input is unchanged.

It can be shown that the scrambler can be easily implemented in software as

```
e := 25; {initialise register}
for i := 0 to N–1 do
   e0 := e and 1;
   e := (e shr 1) xor e0*149;
   yt[i] := y[i] xor e0;
end for;
```

where `i` is the time index, `y[i]` is the encoded bit, `yt[i]` is the scrambled bit and `e0` is the least significant bit of the register `e`. The feedback term is equal to $f = 2^7 h(0.5)–0.5 = 149$ where + is now integer addition. To calculate the initial value g we have

$$g_j = c_j + \sum_{i=0}^{j-1} f_{j-1-i}\, c_i \tag{23}$$

where (f_7,...,f_0), (g_7,...,g_0) and (c_7,...,c_0) are the binary representations of f, g and c, respectively and addition is modulo 2. This gives g = 25.

Turbo Decoder Operation

Figure 5 gives a simplified block diagram of the PCD04CH turbo decoder. After synchronisation has been performed, the received 6–bit input data is optionally descrambled, serial to parallel converted to 6n bits and written into one half of the Input RAM. The other half of the memory has 6n bit input data read into the turbo decoder during each half iteration.

Since SM4 = (SM2,$\overline{\mathrm{SM2}}$) and SM6 = (SM3, $\overline{\mathrm{SM3}}$), to generate the rate 1/4 correlation the length 64 correlation output for rate 1/2 is subtracted from the correlation output delayed by 64 clock cycles. Similarly, to generate the rate 1/6 correlation the length 96 correlation output for rate 1/3 is subtracted from the correlation output delayed by 96 clock cycles. This halves the correlator complexity compared to using a length 192 correlator for rate 1/6, at the expense of an additional delay circuit.

Table 6 illustrates which data is stored for address 0 to K–1 for the main data and K to K+3 for the tail. The entries for the table indicate which encoded data output is selected, X, Y1, Y2 and Y3 for the first encoder and Y1′, Y2′ and Y3′ for the second encoder. The code polynomials are $g^0(D)$ = 1+D^3+D^4 (23 in octal), $g^1(D)$ = 1+D+D^3+D^4 (33),

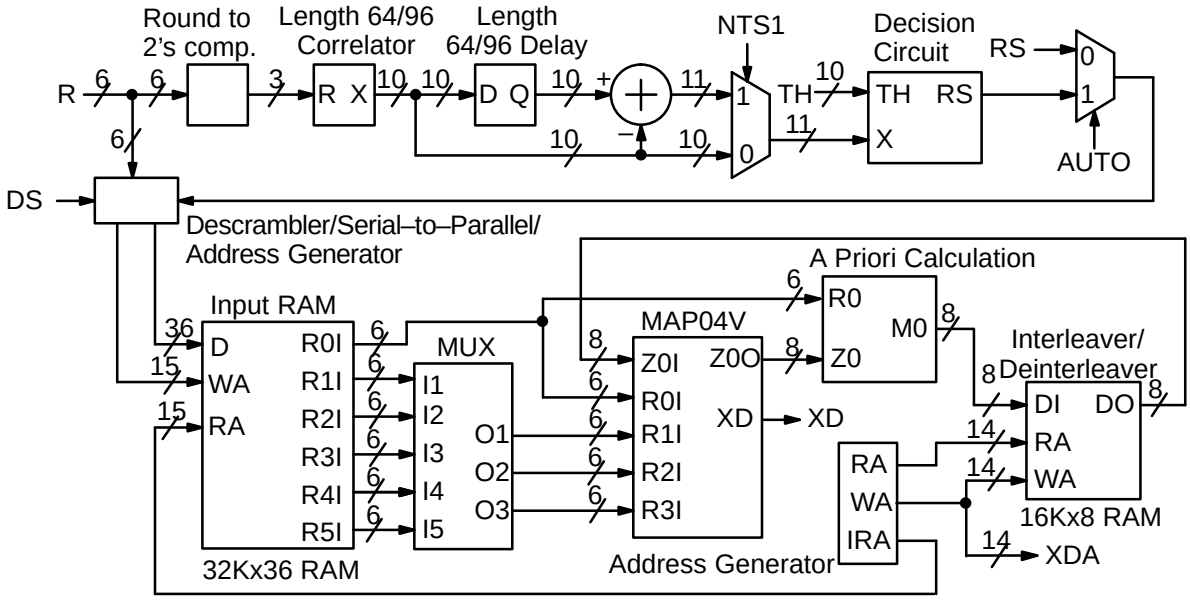

Figure 5: Simplified block diagram of PCD04CH turbo decoder.

$g^2(D) = 1+D^2+D^4$ (25) and $g^3(D) = 1+D+D^2+D^3+D^4$ (37). For rate 1/2 the data and tail are punctured, which is why two entries are shown.

The decoder iteratively decodes the received data for NI+1 half iterations, rereading the received data for each half iteration for K+4 CLK cycles.

The decoded block is output during the last half–iteration. The signal XDR goes high for K CLK cycles while the block is output, with XDF going high for the last decoded bit. If NI is even, the block is output in sequential order. For NI odd, the block is output in interleaved order. To deinterleave the block, the output XDA[13:0] can be used as the write address to a buffer RAM. After the block has been written to the buffer RAM, the decoded block can be sequentially read from the buffer RAM.

Table 6: Input data format

Rate	Data	Output	Rate	Data	Output
1/2	R0I	X X	1/6	R0I	X
	R1I	Y1 Y1′		R1I	Y1
1/3	R0I	X		R2I	Y2
	R1I	Y1		R3I	Y3
	R2I	Y1′		R4I	Y1′
1/4	R0I	X		R5I	Y3′
	R1I	Y2			
	R2I	Y3			
	R3I	Y1′			

The bus ERR[3:0] is a channel error estimator output. For even NA[5:0], ERR[0] is the exclusive OR (XOR) of XD and the sign bit of the input R0I. For ERR[3:1], the parity bits of XD re–encoded are XORed with the sign bits of R1I (rates 1/2 to 1/6), R2I (rates 1/4 and 1/6) and R3I (rate 1/6). These bits are punctured according to the first constituent encoder puncturing pattern. Note that the outputs correspond to the encoder output bit positions. For example, for rate 1/4, the error bits for inputs R1I and R2I are mapped to ERR[2] and ERR[3], which correspond to encoded bits Y2 and Y3.

For odd NA[5:0], ERR[0] is set to zero. This is because R0I is input in sequential order, not in the interleaved order of the output. For ERR[3:1], the parity bits of XD re–encoded are XORed with the sign bits of R1I (rate 1/2), R2I (rate 1/3), R3I (rate 1/4), R4I (rate 1/6) and R5I (rate 1/6). The bits are punctured according to the second constituent encoder puncturing pattern.

If the output of the MAP decoder has zero errors, then this gives an approximation of the channel bit error rate (BER) or to test that the turbo encoder is working correctly. Due to error propagation with the re–encoded parity bits, channel BER estimation is best performed with ERR[0] only. Thus, the decoder should be set to have an odd number (NI even) of half iterations.

Figure 6 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+L+1)+1.

The early stopping algorithm uses the magnitude of the extrinsic information to determine when to stop. As the decoder iterates, the magni-

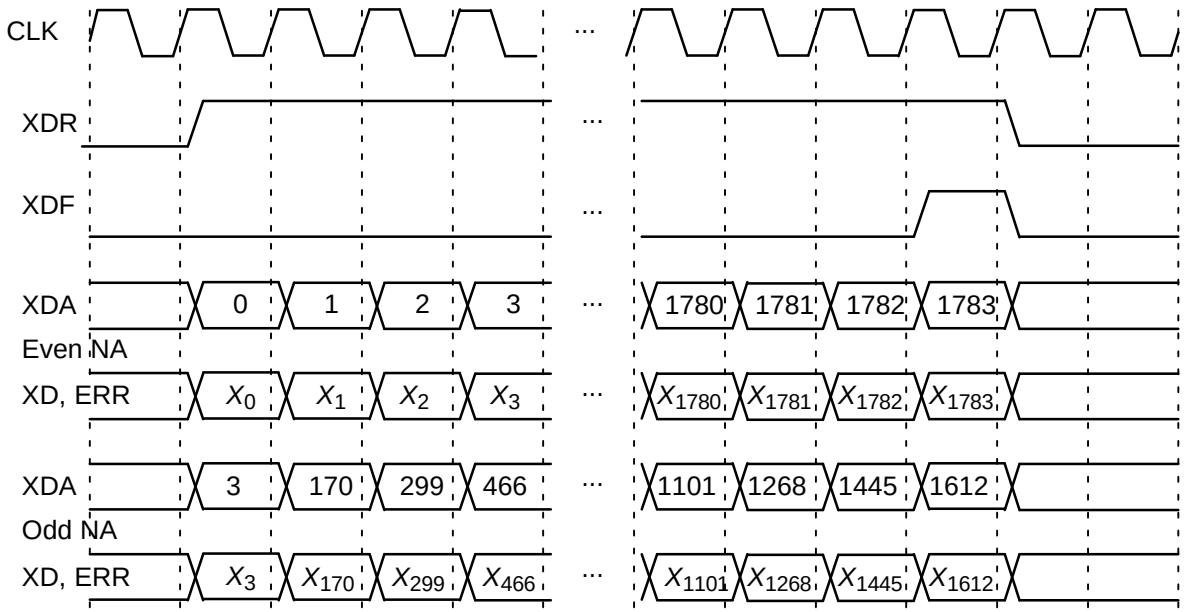

Figure 6: Turbo Decoder Output Timing (K = 1784).

tudes 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.

As the stopping decision can only be decided after a half iteration has been completed, 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. 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, since most blocks will be decoded in one or two iterations.

As the first constituent code is stronger than the second constituent code either by having a lower code rate or more parity bits in the tail, better performance is achieve by selecting M = 1, that is, stopping during odd half iterations.

Input Operation

If automatic synchronisation is enabled (AUTO = 1) then data is input continuously into R[5:0]. The synchronisation circuit will decide when to generate an internal start signal. The external RS input should be held low. The RE input can be used to hold the received data if desired, otherwise, RE should be held high. Figure 7 shows the input timing. As the start of the input data is not known, signals RA[16:0], RR and RF are kept low.

The FULL output indicates when the decoder can accept data. When high this indicates that new data must not be input to the decoder in the next clock cycle. That is, RS and RE must remain low while FULL is high. FULL going high during automatic synchronisation indicates that data is being input too fast for the decoder to decode in time.

With manual synchronisation (AUTO = 0), if FULL is low, the received data start RS signal is used to start the decoder. RS should go high at the start of the turbo encoded block, that is, after the sync marker has ended.

The received data enable input RE must also go high when RS goes high to input the first re-

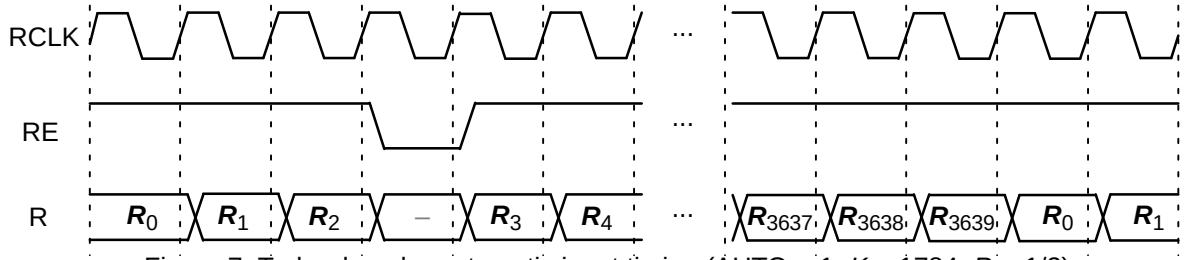

Figure 7: Turbo decoder automatic input timing (AUTO = 1, K = 1784, R = 1/2).

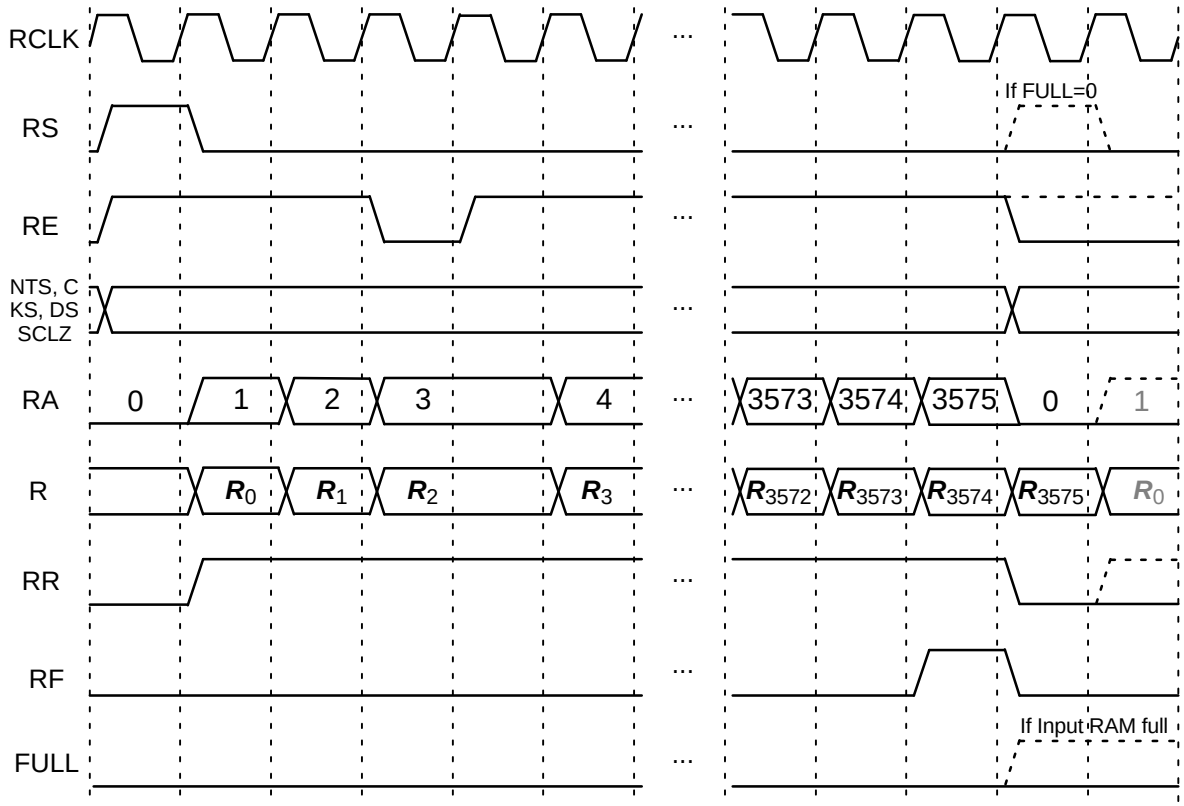

Figure 8: Turbo decoder manual input timing (AUTO = 0, *K* = 1784, *R* = 1/2).

ceived data. RE can only go high once for each received input symbol. This means RE can only be high for *N* clock cycles for each received block.

The received data ready output RR will go high to indicated that RE can now go high so as to read the next input. RR will stay high until one clock cycle before the last data is input. RS and RR can be ORed together to form RE if the input data is stored in an external memory.

Valid data must be input one clock cycle after RE goes high. A received data address output RA[16:0] is provided for reading received data from an external synchronous read input memory. Data read from the input memory must be held if RE goes low as shown in Figure 8.

The received data finish output RF will go high at the end of each received block. This occurs when RA = *N*–1. If RE goes low at RA = *N*–1, RF will remain high. RF will go low only after RE goes high.

Inputs R[5:0], RS, RE, DS, KS[3:0], NTS[1:0], C[4:0] and SCLZ[5:0] must be synchronous to RCLK. Outputs RA[16:0], RR, RF and FULL are synchronous to RCLK. Internal turbo decoding uses CLK. Inputs NI[5:0], M[1:0], ZTH[6:0], LIMZ[6:0], and SLD[1:0] are synchronous to CLK. If RCLK and CLK are equal to each other in both clock period and phase, then RSYNC can equal 1. This reduces the decoder input time by one clock cycle. If RCLK and CLK are not equal, then RSYNC must equal 0. RSYNC should not be connected to logic or input pins.

The input data can be input in any code order. That is, it is not necessary to wait for the decoder to output the last block of one code before changing to another code. If changing the code, the decoder parameters must stay constant from the time RS goes high to until after RF goes high.

Figure 8 illustrates the decoder input timing. Each received sample $\boldsymbol{R}_i$, $0 \le i \le N-1$ represents an 6–bit sample at time *i*. RS is shown going high again for the case where FULL = 0.

Outputs RDS and TDS are internal signals synchronous to RCLK used to indicate the start of writing received data to the input memory and to start the turbo decoder, respectively. Note that the actual received data start signal used by the decoder is $\overline{\text{AUTO}}$.RS+AUTO.RDS.RE where . and + indicate logical AND and OR, respectively.

RDS will go high for one RCLK cycle if a sync marker is detected and SS = 0, or a sync marker is detected at the end of writing data to the input memory, or SS = 1 at the end of writing data to the input memory. TDS will go high for one RCLK cycle if there is input data in the input RAM and the turbo decoder is not decoding.

Simulation Software

Free software for simulating the PCD04CH turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "pcd04chsim request" in the subject header. The software uses an exact functional simulation of the PCD04CH turbo decoder, including all quantisation and limiting effects.

After unzipping pcd04chsim.zip, there should be pcd04chsim.exe and code.txt. The file code.txt contains the parameters for running pcd04chsim. These parameters are

`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 (0.0 to 1.0)
`ferrmax`	Number of frame errors to count
`Pfmin`	Minimum frame error rate (FER)
`Pbmin`	Minimum bit error rate (BER)
`KS`	Data length select (0 to 9)
`NT`	Number of turbo code outputs (2 to 6)
`q`	Number of quantisation bits (1 to 6)
`LOGMAP`	Log–MAP decoding (MODE[0], 0 or 1)
`C4PIN`	Use five–bit *C* (MODE[1], 0 or 1)
`enter_C`	Enter external *C* (y or n)
`C`	External *C* (0 to 17)
`NI`	Number of half iterations–1 (0 to 63)
`SLD`	MAP decoder delay select (0 to 2)
`LIMZ`	Extrinsic information limit (1 to 127)
`SCLZ`	Extrinsic information scale (1 to 32)
`M`	Stopping mode (0 to 4)
`ZTH`	Extrinsic info. threshold (0 to 127)
`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 data length is selected by `KS` (0 to 9). The code rate is selected by the number of code bits `NT` (2 to 6).

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

$$A = \frac{\text{optC}(2^{q-1} - 1)}{\text{mag}(\sigma)} \qquad (24)$$

where σ^2 is the normalised noise variance given by (2) and mag(σ) is given by (14), simulating the effect of an AGC circuit that normalises the received amplitude to a fixed voltage.

Although mag(σ) is a complicated function, for high signal to ratio (SNR), mag(σ) $\simeq$ 1. For very 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 the value of *C* given by (1) to the nearest integer. If LOGMAP = MODE[0] = 0 then *C* is forced to 0. If LOGMAP = 1 and C4PIN = MODE[1] = 0, *C* is limited to a maximum value of 9. If LOGMAP = 1 and C4PIN = 1, *C* is limited to a maximum value of 17. An external value of *C* can be input by setting `enter_C` to `y`.

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), M = 1, ZTH = 23, LIMZ = 96 and large log–MAP decoding. Using these parameters for higher E_b/N_0 values should result is very little performance degradation.

Table 7: Simulation parameters

R	E_b/N_0 (dB)	optC	A	C	SCLZ	BER 10^{-2}
1/6	–0.27	0.35	6.55	11	31	3.31
1/4	0.08	0.35	8.19	7	31	2.52
1/3	0.28	0.37	9.02	6	31	2.79
1/2	0.88	0.42	11.55	5	28	2.90

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) and the BER are both below or equal to `Pfmin` and `Pbmin`, respectively. 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.

When the simulation is finished the output is given in, for example, file k1784.dat, where *K* = 1784. For each simulation point 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 frame errors (`ferr`), the average number of iterations (`na`), the average bit error rate

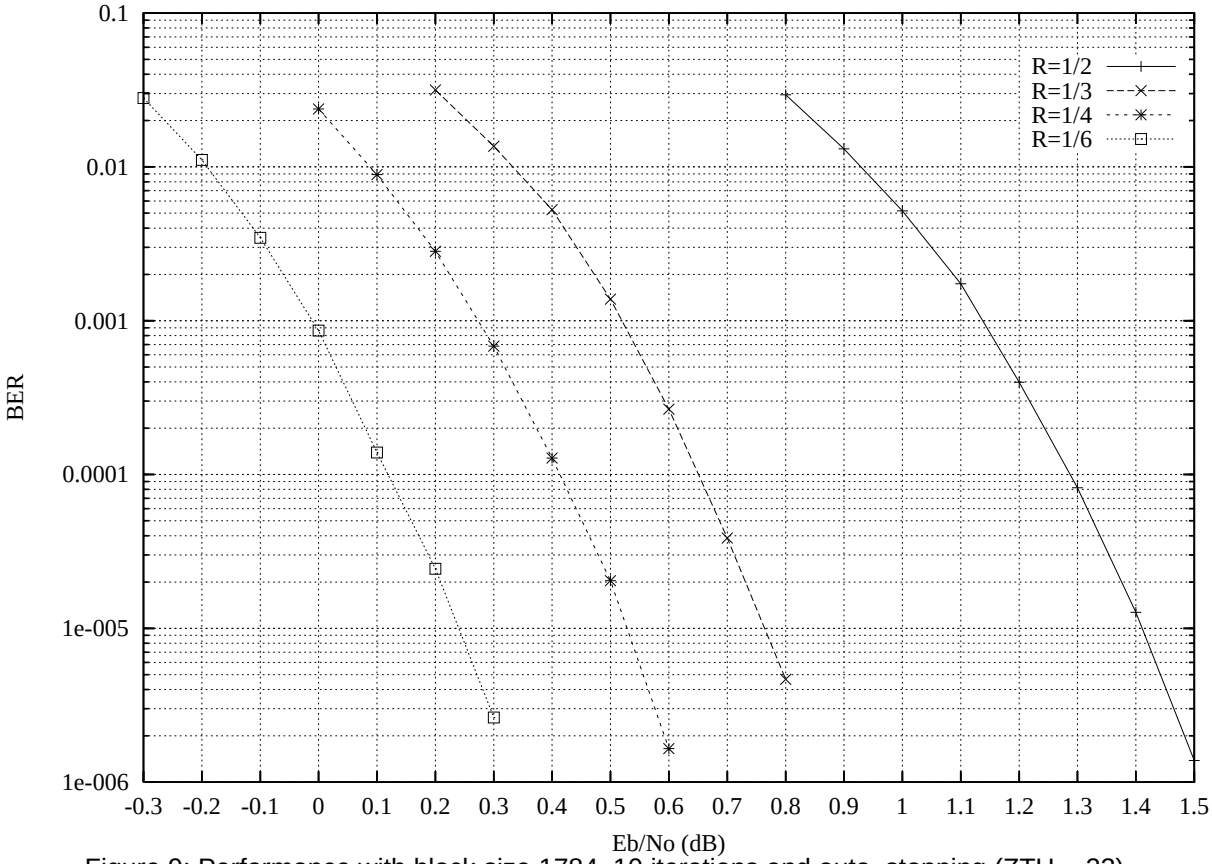

Figure 9: Performance with block size 1784, 10 iterations and auto–stopping (ZTH = 23).

(`Pb`) and the average frame error rate (`Pf`). Following this, `na`, `berr`, `ferr`, `Pb` and `Pf` are given for each half iteration.

The following file was used to give the rate 1/2 simulation results shown in Figure 9. For $P_b \leq 10^{-4}$, `ferrmax` = 64. Auto–stopping was used with a maximum of 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. This allows the performance to be obtained for each iteration number. Figure 10 shows the average number of iterations with E_b/N_0 for rate 1/2.

```
{EbNomin EbNomax EbNoinc optC}
 0.8     1.5     0.1     0.42
{ferrmax Pfmin Pbmin}
 256     1.00  1e-5
{KS NT  q  LOGMAP  C4PIN  enter_C  C}
 0  2   6  1       1      y        5
{NI  SLD LIMZ  SCLZ  M  ZTH}
 19  1   96    28    2  23
{state s1    s2     out_screen}
 0     12345 67890  y
{read_x  read_r  out_dir  in_dir}
 n       n       output   input
```

The `state` input can be used to continue the simulation after the simulation has been stopped,

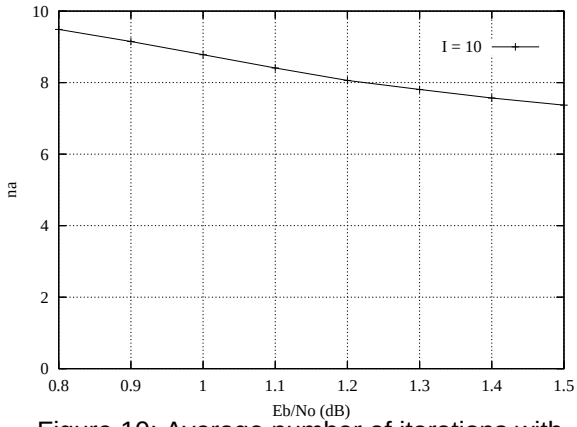

Figure 10: Average number of iterations with block size 1784 and auto–stopping (ZTH = 23).

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 k(K).dat and the state files state1.dat and state2.dat. This ensures you can restart the program if a mistake is made in configuring code.txt.

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_1784.dat` in directory `input` (each line contains one bit of data). The encoded stream `y_(K).dat` will be output to the directory given by `out_dir`, e.g., `y_1784.dat` to directory `output`. The encoded data includes the sync marker and optional scrambling.

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`. `r_(K).dat` has in each line R[i,j], i = 0 to n–1 from j = 0 to *K*+3, e.g., for n = 3 the first three lines could be

```
–31   1 –25
 –31  12   9
  11  31  31
```

Note that the sync marker should not be included. 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 31. If `read_r` = `y`, then *C* is externally input via `c`.

Ordering Information

SW–PCD04CH–SOS (SignOnce Site License)
SW–PCD04CH–SOP (SignOnce Project License)
SW–PCD04CH–VHD (VHDL ASIC License)

All licenses include Xilinx VHDL cores. The SignOnce and ASIC licenses allows unlimited instantiations and free updates for one year.

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] Consultative Committee for Space Data Systems, "Recommendation for space data system standards: TM Synchronization and channel coding," CCSDS 131.0–B–3, Blue Book, Sep. 2017.

[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.

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 30 Apr 2021. Preliminary product specification.
  0.01 10 May 2021. Added SS[1:0] output. Corrected Example 1.
  0.02 12 May 2021. Updated FULL output and TH input.

- 0.03 31 May 2021. Changed NI width from 8 to 6 bits and MODE width from 5 to 7 bits.
- 1.00 8 July 2021. First release. Corrected NA width from 8 to 6 bits and RA width from 14 to 17 bits. Added resources used and timing performance.
- 1.01 4 May 2022. Added RRST input and RF output. Updated Table 2.
- 1.02 16 May 2022. Added RDS and TDS outputs.
- 1.03 24 May 2022. Added Synchronisation Operation section and input timing figure for AUTO = 1.
- 1.04 25 July 2022. Updated decoder speed and complexity. Simplified correlation circuit.
- 1.05 17 Aug. 2022. Added MODE[7] input. Updated decoder speed and complexity.
- 1.06 23 Oct. 2023. Corrected E_b/N_0 values in Table 7.
- 1.07 19 Dec. 2023. Added description of RA[16:0], RR and RF output for AUTO = 1.
- 1.08 28 Dec. 2023. Updated TH[9:0] calculation on pages 6 and 7.