PCD03T Features

Turbo Decoder

- 8 state TETRA–TEDS compatible
- Rate 1/2 or 1/3
- 2 to 6144 bit interleaver
- Up to 145 MHz internal clock
- Up to 13.8 Mbit/s with 5 decoder iterations
- 6–bit signed magnitude input data
- 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

Viterbi Decoder (Optional)

- 16 state (constraint length 5)
- Rate 1/4
- Data lengths from 2 to 8188 bits with tail termination of 4 zero data bits
- Up to 11.5 Mbit/s
- 6–bit signed magnitude input data
- Estimated channel error output

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

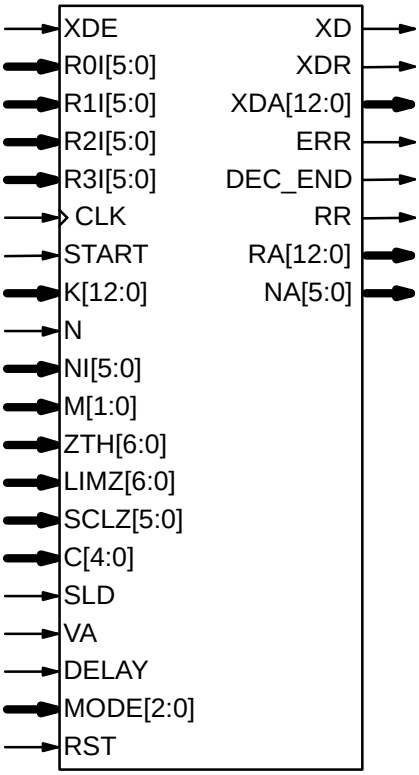

Figure 1: PCD03T schematic symbol.

Introduction

The PCD03T is a fully compatible TETRA (Terrestrial Trunked Radio) and TEDS (TETRA Enhanced Data Service) [1] error control turbo decoder. A constituent 8 state rate 1/2 systematic recursive convolutional code is used in the turbo encoder to form a rate 1/3 turbo code.

A symmetric interleaver is constructed from the quadratic permutation sequence $c(m) = (c(m–1)+m) \bmod S = m(m+1)/2 \bmod S$ where S is the smallest power of 2 greater than or equal to the data length K. This allows any interleaver lengths from 2 to 6144 bits to be used. The interleaver address table is calculated during the first half iteration of decoding (if K has changed) and stored in a 6Kx13 dual port RAM.

The non–interleaved data constituent encoder is terminated with a 6–bit tail. The interleaved data constituent encoder is terminated with a 3–bit tail, the data bits being punctured. This gives a total of nine tail bits.

The nominal code rate is 1/3. However, the PCD03T can also optionally decode the rate 1/2 turbo code specified in TETRA. For higher code rates, the decoder should be operated in rate 1/3 with input data being externally punctured.

The MAP03V MAP decoder core is used with the PCD03T 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. The sliding block algorithm is used with sliding block lengths of 32 or 64. Six–bit quantisation is used for maximum performance. The extrinsic information can

be scaled and limited with each half iteration, improving performance with max–log–MAP decoding.

The VA08V Viterbi decoder core is optionally used with the PCD03T core to decode the 16 state rate 1/4 TETRA convolutional code. The decoder shares its traceback memory with the internal interleaver memory of the turbo decoder, minimising complexity. Minimum traceback lengths of 32 or 64 bits can be selected. 6–bit quantisation is used.

The turbo decoder can achieve up to 13.8 Mbit/s with 5 iterations and log–MAP decoding using a 145 MHz internal clock (K = 5533). Max–log–MAP decoding increases speed by about 44%. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance. The Viterbi decoder can achieve 11.5 Mbit/s with K = 508.

Figure 1 shows the schematic symbol for the PCD03T decoder. The EDIF core can be used with Xilinx Integrated Software Environment (ISE) software to implement the core in Xilinx FPGA's. The VHDL core can be used in ASIC designs.

Table 1 shows the resources used for Virtex–4 and Virtex–5. Resources for Virtex–II and Spartan–3 are similar to that for Virtex–4. Resources for Virtex–6 and Spartan–6 are similar to that for Virtex–5. The MODE[2:0] 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. If not specified, the turbo decoder uses small–log–MAP.

Table 1: Resources used.

Configuration	Virtex–4 Slices	Virtex–5 LUTs	Block RAMS
max–log–MAP	1495	2370	8
small–log–MAP	1932	3115	8
large–log–MAP	2172	3251	8
small–log–MAP & Viterbi	2489	3941	8

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)	Turbo* Mbit/s	Viterbi Mbit/s
XC3S400–4	21.378	4.46	3.71
XC3S400–5	18.664	5.11	4.25
XC6SLX25–2	17.640	5.41	4.50
XC6SLX25–3	13.437	7.10	5.91
XC4VLX15–10	13.402	7.12	5.92
XC4VLX15–11	11.429	8.35	6.94
XC4VLX15–12	10.160	9.39	7.81
XC5VLX30–1	10.653	8.96	7.45
XC5VLX30–2	9.151	10.43	8.67
XC5VLX30–3	8.153	11.70	9.73
XC6VLX75T–1	9.012	10.59	8.81
XC6VLX75T–2	7.700	12.39	10.31
XC6VLX75T–3	6.892	13.85	11.52

*small log–MAP, 5 iterations, K = 5533, SLD = 1

Signal Descriptions

C MAP Decoder Constant
0–9 (MODE1 = 0)
0–17 (MODE1 = 1)

CLK System Clock
DEC_END Decode End Signal
DELAY Viterbi Decoder Delay
0 = delay 68
1 = delay 132
ERR Estimated Error
K Interleaver Length (2–6144)
LIMZ Extrinsic Information Limit (1–127)
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)
N Turbo Code Rate
0 = rate 1/2
1 = rate 1/3
NA Half Iteration Number (0–63)
NI Number of Half Iterations (0–63)
NI = 2I–1 where I is number of iterations
R0I–R3I Received Data
RA Received Data Address
RR Received Data Ready
RST Synchronous Reset
SCLZ Extrinsic Information Scale (1–32)
SLD MAP Decoder Delay
0 = delay 137
1 = delay 265
START Decoder Start
VA Viterbi Decoder Select
0 = turbo decoder
1 = Viterbi decoder

XD	Decoded Data
XDA	Decoded Data Address
XDE	Decoded Data Enable
XDR	Decoded Data Ready
ZTH	Early Stopping Threshold (1–127)

Table 3: MODE selection

Input	Description
MODE0	0 = max–log–MAP 1 = log–MAP
MODE1	0 = small log–MAP (C4 = 0) 1 = large log–MAP
MODE2	0 = turbo decoder 1 = turbo and Viterbi decoder

Table 3 describes each of the MODE[2:0] inputs that are used to select various decoder implementations. Note that MODE[2:0] 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 TETRA has a maximum interleaver size of 5533 bits. However, a size of 6K is provided since the Xilinx BlockRAM step size is 2K with 8–bit words.

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 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

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

E_b/N_0 is the energy per bit to single sided noise density ratio, R = 1/(n+3n/K), K is the data length and n = 2–3. C should be rounded to the nearest integer and limited to be no higher than 17 with MODE1 high and 9 with MODE1 low. Max–log–MAP [3] operation occurs when C = 0. Due to quantisation effects, C = 1 is equivalent to C = 0.

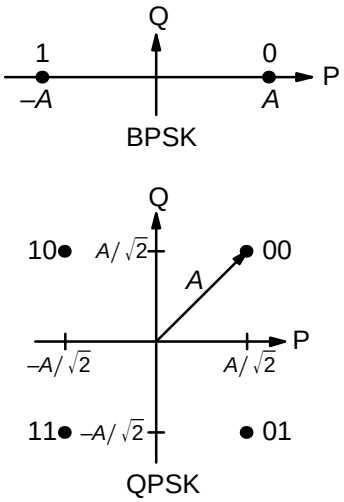

Figure 2: BPSK and QPSK signal sets.

Max–Log–MAP operation is also enabled 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 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).

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

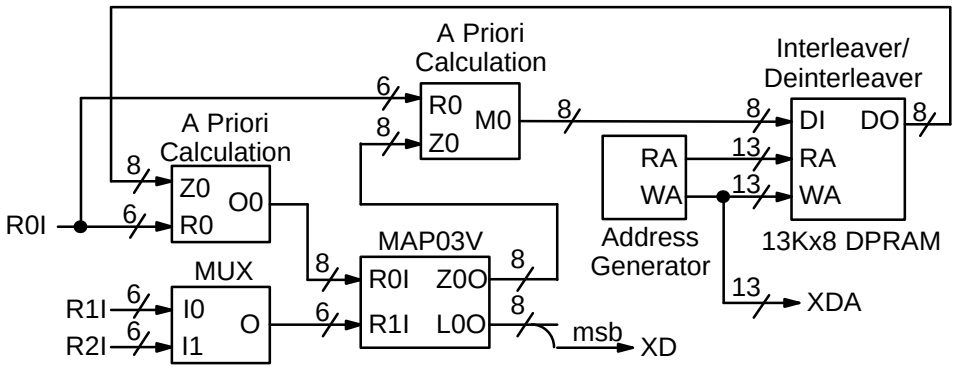

Figure 3: Simplified block diagram of PCD03T eight state turbo decoder.

shown in the table. This allows maximum performance to be achieved.

Table 4: Quantisation for R0I, R1I and R2I.

Signed Magnitude		Two's Complement		Range
Dec.	**Binary**	**Dec.**	**Binary**	
31	011111	31	011111	30.5↔∞
30	011110	30	011110	29.5↔30.5
⋮	⋮	⋮	⋮	⋮
2	000010	2	000010	1.5↔2.5
1	000001	1	000001	0.5↔1.5
0	000000	0	000000	–0.5↔0.5
33	100001	63	111111	–1.5↔–0.5
34	100010	62	111110	–2.5↔–1.5
⋮	⋮	⋮	⋮	⋮
62	111110	34	100010	–30.5↔–29.5
63	111111	33	100001	–∞↔–30.5

For signed magnitude inputs a decimal value of 32 has a magnitude of 0 (equivalent to –0). For two's complement inputs, if 32 is input, this will be internally limited to 33 (equivalent to –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* = 16 we have from (1) that *C* = 11 to the nearest integer.

Figure 3 gives a block diagram of the PCD03T eight state turbo decoder. The number of turbo decoder half–iterations is given by *NI*, ranging from 0 to 63. *NI* = 2*I*–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 an earlier time if early stopping is enabled. The NA output can be used to select LIMZ and SCLZ values, especially for max–log–MAP decoding.

The turbo decoder speed f_d is given by

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

where F_d is the CLK frequency and *L* is the MAP decoder delay in bits (equal to either 137 or 265). The two delays indicate the sliding block length used in the MAP decoder, either 32 or 64, respectively. For short block lengths *L* = 137 should be used to increase decoder speed, while *L* = 265 should be used for larger block sizes to increase performance. This parameter can be selected with the SLD input.

For example, if F_d = 50 MHz and *I* = 5 (*NI* = 9) the decoder speed ranges from 1.7 Mbit/s for *K* = 73 and *L* = 137 to 4.8 Mbit/s for *K* = 5533 and *L* = 265.

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

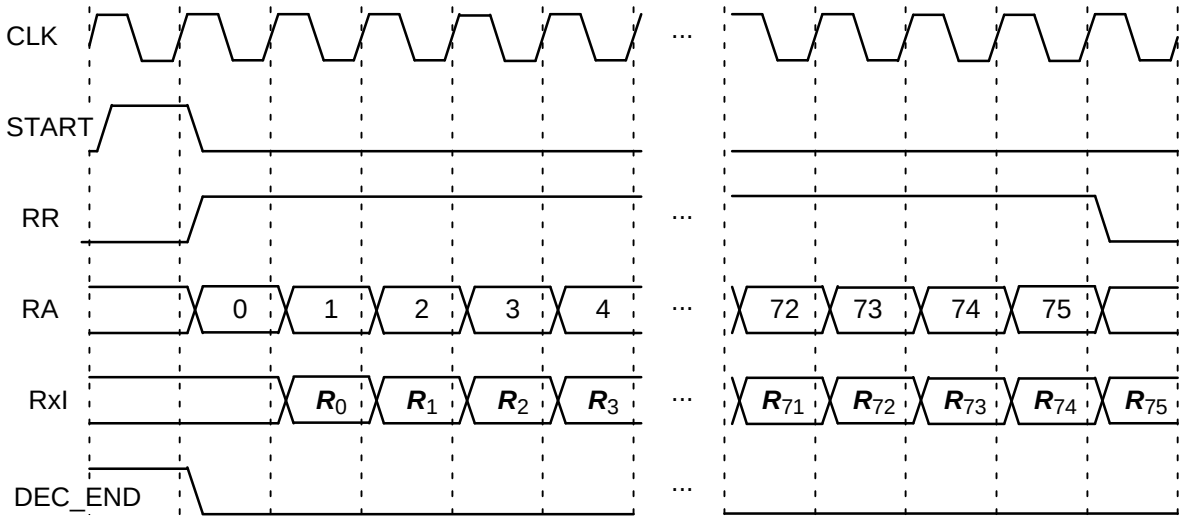

Figure 4: Turbo Decoder Input Timing (K = 73).

SCLZ/32. 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.

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 a synchronous read RAM of size (K+3)x6n, n = 2 to 3. The received data ready signal RR goes high to indicated the data to be read from the address given by RA[12:0]. Tables 5 illustrates which data is stored for address 0 to K+2. The entries for the table indicate which encoded data output is selected, X and Y1 for the first encoder and X′ and Y1′ for the second encoder. The code polynomials are $g^0(D) = 1+D^2+D^3$ (13 in octal) and $g^1(D) = 1+D+D^3$ (15). For rate 1/2 the data is punctured, which is why two entries are shown.

The decoder then iteratively decodes the received data for NI+1 half iterations, rereading the received data for each half iteration for K+3 CLK cycles. The signal RR goes high for K+3 clock cycles while data is being output. Figure 4 illustrates the decoder timing where the data is input on the first half iteration.

Table 5: Input data format

Rate	Data	Main
1/2	R0I	X X
	R1I	Y1 Y1′
1/3	R0I	X
	R1I	Y1
	R2I	Y1′

Note that the START signal is ignored during decoding, except for the last clock cycle before DEC_END goes high. 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 block is output during the last half–iteration. The signal XDR goes high for K CLK cycles while the block is output. 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[12:0] 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.

The signal ERR is a channel error estimator output. For even NI, it is the exclusive OR of XD and the sign bit of the input R0I. For odd NI, it is the exclusive OR of XD re–encoded to give the first parity bit and the sign bit of the input corresponding to Y1′ (this is because R0I is input in sequential order, not in the interleaved order of the

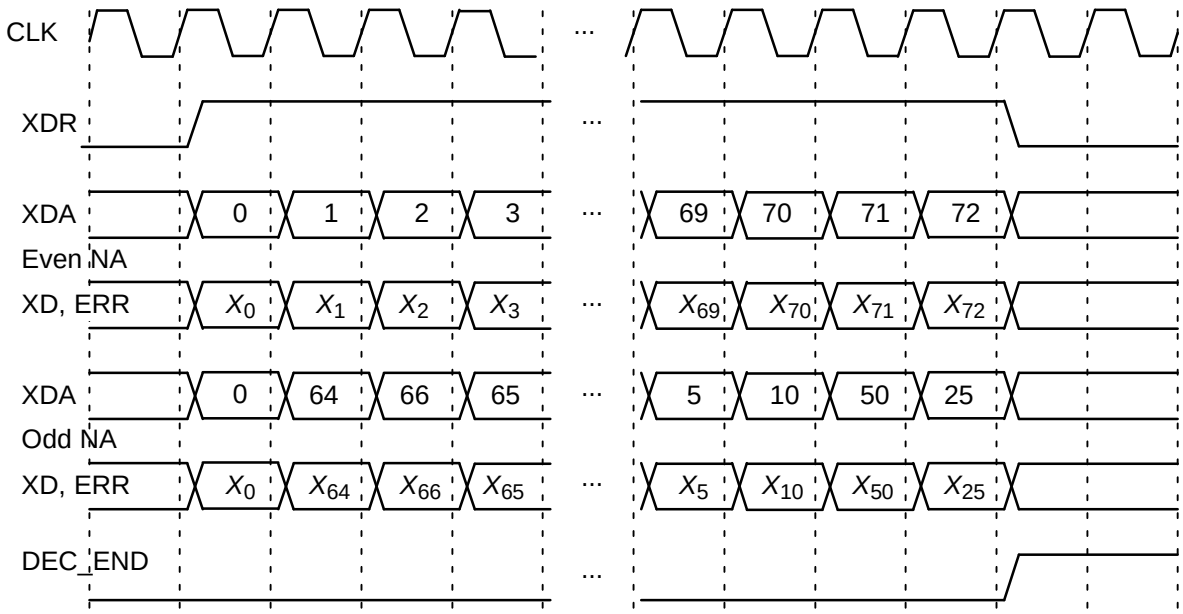

Figure 5: Turbo Decoder Output Timing (K = 73).

output). If the output of the MAP decoder has zero errors, then this gives an approximation of the channel bit error rate (BER). Since Y1′ is punctured for rate 1/2, the number of bits counted is $\lfloor K/2 \rfloor$.

The DEC_END signal is low during decoding. At the end of decoding, DEC_END goes high. Figure 5 illustrates the decoder timing where data is output on the last half iteration. After startup, the maximum number of clock cycles for decoding is $(NI+1)(K+L)$.

During the last half iteration the decoded data is stored into the interleaver memory. Once decoding has been completed, the input XDE can be used to sequentially clock the decoded data from from the interleaver memory (regardless of the number of iterations). XDE is disabled while the decoder is iterating. Figure 6 shows the decoder timing when XDE is used.

The output ERR is also output when XDE goes high. The outputs RA and RR are used to read the sign bit of R0I which is exclusive–ORed with XD to give ERR.

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

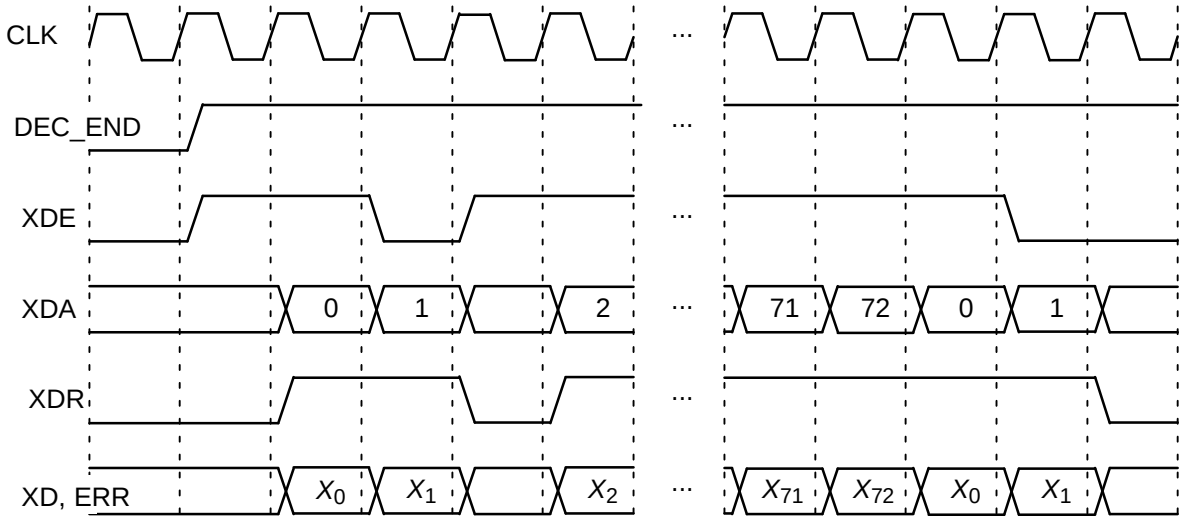

Figure 6: XDE Timing (K = 73).

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

Simulation Software

Free software for simulating the PCD03T turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "pcd03tsim request" in the subject header. The software uses an exact functional simulation of the PCD03T turbo decoder, including all quantisation and limiting effects.

After unzipping pcd03tsim.zip, there should be pcd03tsim.exe and code.txt. The file code.txt contains the parameters for running pcd03tsim. These parameters are

`m` Constituent code (CC) memory (2 to 4)
`nt` Number of turbo code outputs (2 to 3)
`g0` Divisor polynomial of CC in octal notation
`g1` 1st numerator polynomial of CC
`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 0.65)
`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 127)
`SCLZ` Extrinsic information scale (1 to 32)
`M` Stopping mode (0 to 4)
`ZTH` Extrinsic info. threshold (0 to 127)
`K` Block length (2 to 6144)
`q` Number of quantisation bits (1 to 6)
`LOGMAP` Log–MAP decoding (MODE0, 0 or 1)
`C4PIN` Use five–bit *C* (MODE1, 0 or 1)
`state` State file (0 to 2)
`s1` Seed 1 (1 to 2147483562)
`s2` Seed 2 (1 to 2147483398)
`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
`out_screen` Output data to screen (y or n)
`C` *C* (0 to 17)

Note that g0 and g1 are given in octal notation, e.g., g0 = 13 $\equiv 1011_2 \equiv 1+D^2+D^3$. For the TETRA standard, m = 3, nt = 2, g0 = 13 and g1 = 15. The nominal turbo code rate is 1/nt.

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.

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

$$A = \frac{\text{optC}(2^{q-1} - 1)}{\text{mag}(\sigma)} \tag{4}$$

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

$$\text{mag}(\sigma) = \sigma\sqrt{\frac{2}{\pi}}\exp\left(\frac{-1}{2\sigma^2}\right) + 1 - 2Q\left(\frac{1}{\sigma}\right) \tag{5}$$

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

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}}\exp\left(\frac{-t^2}{2}\right)dt. \tag{6}$$

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 the value of *C* given by (1) to the nearest integer. If LOGMAP = MODE0 = 0 then *C* is forced to 0. If LOGMAP = 1 and C4PIN = MODE1 = 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.

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

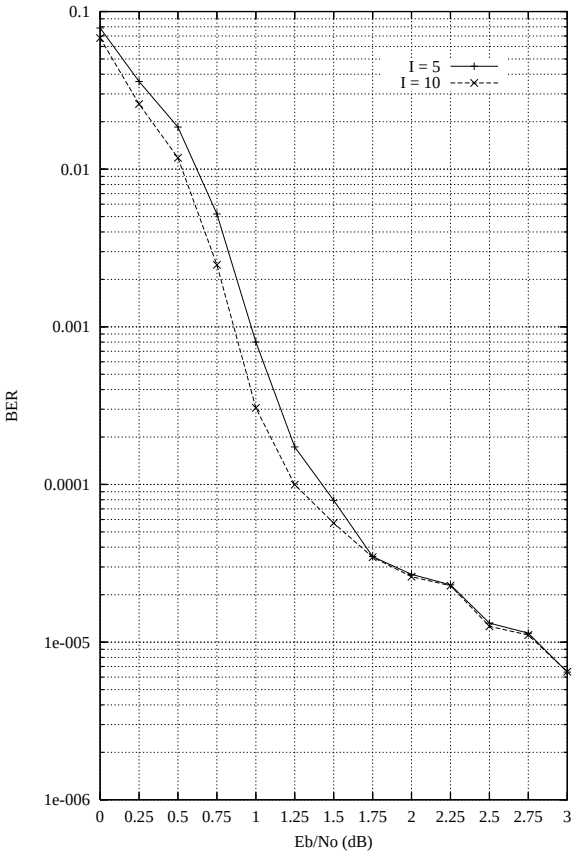

Figure 7: Performance with block size 597 and auto–stopping (ZTH = 23).

to `ferrmax`. If `ferrmax` = 0, then only one frame is simulated.

When the simulation is finished the output is given in, for example, file k597.dat, where $K = 597$. Only frames that are in error are stored in the output file. 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 average number of iterations (`na`), and 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 7. 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. Thus, the $I = 10$ result is the performance one would measure with auto–stopping and $NI = 19$. The $I = 5$ curve shows the performance at 5 iterations with early stopping and $NI = 9$. Figure 8 shows the average number of iterations with E_b/N_0.

```
{m nt g0 g1}
 3 3  13 15
```

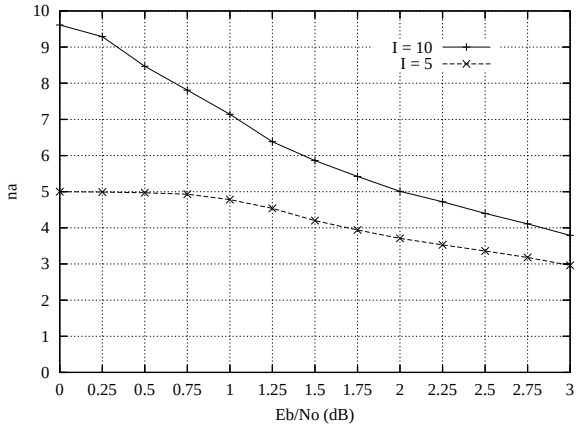

Figure 8: Average number of iterations with block size 597 and auto–stopping (ZTH = 23).

```
{EbNomin EbNomax EbNoinc optC ferrmax
Pfmin Pbmin}
 0.00    3.00     0.25    0.45 64
1e-99 1e-5
{NI SLD LIMZ  SCLZ  M  ZTH}
 19 1   96     29    1  23
{K   q LOGMAP C4PIN}
 597 6 1      1
{state s1    s2}
 0     12345 67890
{read_x  read_r out_dir in_dir
out_screen C}
 n        n      output  input
y          12
```

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.
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_597.dat` in directory `input` (each line contains one bit of data). The encoded

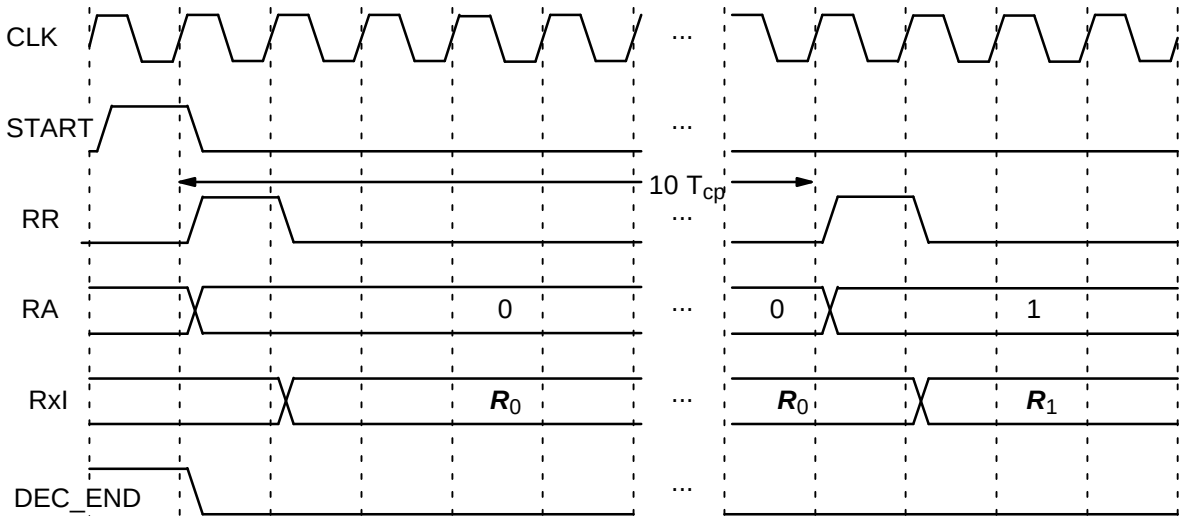

Figure 9: Viterbi Decoder Input Timing.

stream `Y_(K).dat` will be output to the directory given by `out_dir`, e.g., `Y_597.dat` to directory `output`.

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 nt–1 from j = 0 to *K*+*m*–1, e.g., for nt = 3 the first three lines could be

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

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`.

Viterbi Decoder Operation

The Viterbi decoder is operated in a similar way to the turbo decoder. The START signal is used to start decoding, using RR and RA to read the received data. All inputs R0I to R3I are used to obtain a code rate of 1/4. Higher code rates are obtained by externally puncturing the data.

The decoder complexity is fixed at 16 states (constraint length 5). The input DELAY = 0 and 1 selects a delay of 68 and 132. The code used in the TETRA–TEDS standard is g_0 = 31, g_1 = 27, g_2 = 35 and g_4 = 33 in octal notation

The decoder inputs the received data from address 0 to *K*+3, where the last four received data corresponds to the tail. After a decoding delay, the decoded data is output to XD. XDR goes high for one clock cycle at the beginning of each decoded bit. XDA goes from address 0 to *K*–1 as the decoded data is output.

The output ERR is the exclusive–OR of the sign bit of R0I with the corresponding re–encoded decoded output bit. This allows an estimate of the channel BER.

Figure 9 shows the Viterbi decoder input timing. Two clock cycles are used to start decoding, with each decoded bit taking 10 clock cycles.

Figure 10 shows the Viterbi decoder output timing. The input XDE is not used either during or after Viterbi decoding.

The decoding speed is given by

$$f_d = \frac{F_d}{N_c(1 + D/K) + 1/K} \qquad (7)$$

where F_d is the internal clock speed, N_c is the number of decoder clock cycles (10) and *D* is the Viterbi decoder delay in bits. For example, if *K* = 508, *D* = 132 (DELAY = 1), N_c = 10 and F_d = 50 MHz, decoding speed is 3.97 Mbit/s.

Ordering Information

SW–PCD03T–SOS (SignOnce Site License)
SW–PCD03T–SOP (SignOnce Project License)
SW–PCD03T–VHD (VHDL ASIC License)
SW–PCD03T–UNI–n (University License)

All licenses include EDIF and VHDL cores. The VHDL cores can only be used for simulation in the SignOnce and University licenses. The above licenses do no include the Viterbi decoder which must be ordered separately (see the VA08V

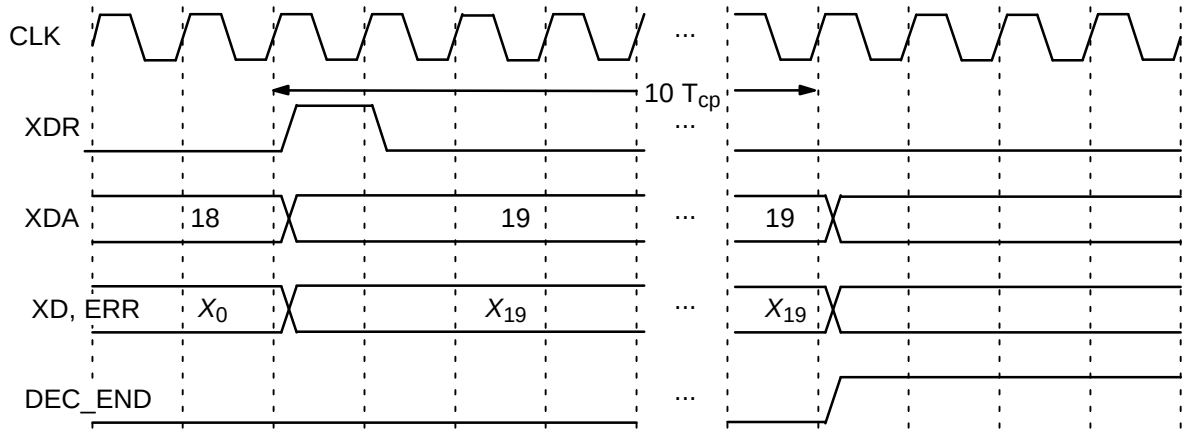

Figure 10: Viterbi Decoder Output Timing (K = 20).

data sheet). 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] ETSI, "Terrestrial Trunked Radio (TETRA); Voice plus Data (V+D); Part 2: Air Interface (AI)," ETSI EN 300 392–2 V3.4.0, Mar. 2010.

[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 8332 3177

Version History

- 0.00 10 January 2011. First release of preliminary product specification
- 1.00 17 January 2011. First official release. Added simulation result figures and Genie aided early stopping for BER simulation software.
- 1.01 7 March 2011. Updated BER simulation results.