



PCD04C Features

Turbo Decoder

- 16 state constituent decoder
- Rate 1/2 to 1/7
- CCSDS compatible
- Interleaver sizes from 1784 to 16056 bits
- Up to 99 MHz internal clock
- Up to 9.8 Mbit/s with 5 decoder iterations
- 6-bit signed magnitude input data
- Log-MAP or max-log-MAP constituent decoder algorithms
- Up to 128 iterations in 1/2 iteration steps
- Power efficient early stopping
- Extrinsic information output with optional scaling and limiting
- Full estimated channel error output
- Free simulation software

Viterbi Decoder (Available Separately)

- 64 or 256 state (constraint length 7 or 9)
- Rate 1/2, 1/3 or 1/4
- Block lengths from 1784 to 16056
- Up to 2.9 Mbit/s (256 state) or 9.8 Mbit/s (64 state)
- 6-bit signed magnitude input data
- Estimated channel error output
- Available as EDIF core and VHDL simulation core for Xilinx Virtex-II Pro, Spartan-3, Virtex-4 and Virtex-5 FPGAs under SignOnce IP License. Actel and Lattice FPGA cores available on request.
- Available as VHDL core for ASICs
- Low cost university license also available

Introduction

The PCD04C 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. Turbo code rates from 1/2 to 1/7 can be selected. The un-interleaved 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 $n(K+4)$ where the nominal code rate is $1/n$.

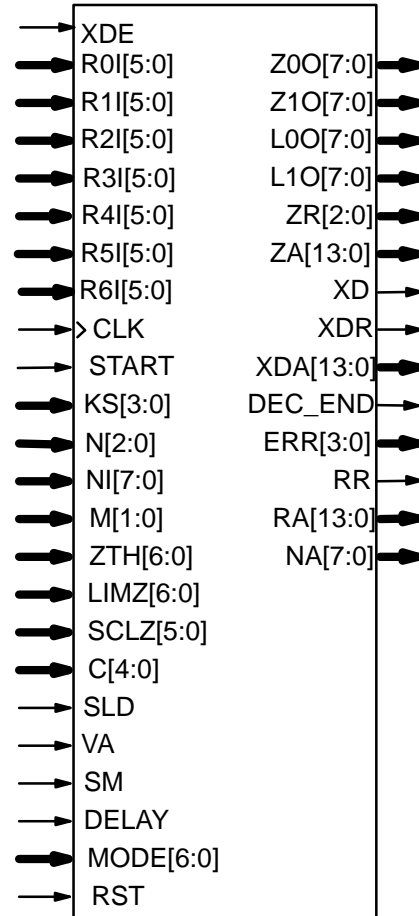


Figure 1: PCD04C schematic symbol.

The MAP04V MAP decoder core is used with the PCD04C 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 extrinsic information of both the data and first parity bits of the constituent code are also output.

The VA08V Viterbi decoder core can be used with the PCD04C core to decode 64 or 256 state rate 1/2 to 1/4 convolutional codes. The decoder

shares its traceback memory with the internal interleaver memory of the turbo decoder, minimizing complexity. Maximum traceback lengths of 48 or 96 bits for 64 states or 60 or 120 bits for 256 states can be selected. 4-bit quantisation is used.

The turbo decoder can achieve up to 9.8 Mbit/s with 5 iterations using a 99 MHz internal clock ($K = 16056$). Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance. The Viterbi decoder can achieve up to 2.9 Mbit/s with 256 states and 9.8 Mbit/s with 64 states.

Figure 1 shows the schematic symbol for the PCD04C decoder. This symbol is used to compile various BIT files for download into Xilinx FPGA's. 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)	Turbo* Mbit/s	K=9 Mbit/s	K=7 Mbit/s
XC2VP7-5	17.246	5.70	1.69	5.75
XC2VP7-6	15.359	6.40	1.89	6.45
XC2VP7-7	13.197	7.45	2.20	7.51
XC3S1000-4	23.336	4.21	1.24	4.24
XC3S1000-5	20.373	4.82	1.43	4.86
XC4VLX15-10	14.253	6.90	2.05	6.96
XC4VLX15-11	12.157	8.09	2.40	8.16
XC4VLX15-12	10.800	9.11	2.70	9.18
XC5VLX30-1	12.794	7.69	2.28	7.75
XC5VLX30-2	11.002	8.94	2.65	9.01
XC5VLX30-3	10.027	9.81	2.91	9.89

*small log-MAP, 5 iterations, no Z/L outputs, $K=16056$

Table 2 shows the number of slices used for Virtex-4 and Virtex-5 devices. The complexity for Virtex-II Pro and Spartan-3 devices are similar to that for Virtex-4. The MODE[6:0] 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. A 16K interleaver is used, requiring eight Block RAMs.

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 70 (SM = 0) 0 = delay 72 (SM = 1) 1 = delay 134 (SM = 0) 1 = delay 136 (SM = 1)
ERR	Estimated Error
KS	Interleaver Size Select (0 to 8, Block Length $K = 1784(KS+1)$)
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)
L0O	Data Log-Likelihood Information
L1O	Parity Log-Likelihood Information
LIMZ	Extrinsic Information Limit (1-127)
N	Code Rate (2-7 turbo, 2-4 Viterbi)
NA	Half Iteration Number (0-255)
NI	Number of Half Iterations (0-255) $NI = 2I-1$ where I is number of iterations
R0I-R6I	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 136 1 = delay 264
SM	Viterbi Decoder Memory 0 = 64 state (constraint length 7) 1 = 256 state (constraint length 9)
START	Decoder Start
VA	Viterbi Decoder Select 0 = turbo decoder
XD	Decoded Data
XDA	Decoded Data and Z0A Address
XDE	Decoded Data Enable
XDR	Decoded Data Ready
Z0O	Data Extrinsic Information
Z1O	Parity Extrinsic Information
ZA	Z1O Address
ZR	Extrinsic Information Ready
ZTH	Early Stopping Threshold (1-127) 1 = Viterbi decoder

Table 3 describes each of the MODE[6:0] inputs that are used to select various decoder implementations. Note that MODE[6:0] 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

Table 2: Resources used

Configuration	Turbo Rates	Viterbi Rates	Z/L	Virtex-4 Slices	Virtex-5 Slices
Turbo (max-log-MAP)	1/2-1/7	-	No	3223	2688
Turbo (small log-MAP)	1/2-1/7	-	No	4055	3462
Turbo (large log-MAP)	1/2-1/7	-	No	4514	3411
Turbo (small log-MAP)	1/2-1/3	-	No	3229	2644
Turbo (small log-MAP)	1/2-1/7	-	Yes	4756	3727
Turbo (small log-MAP) and Viterbi	1/2-1/7	1/2-1/4	No	4671	3672

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 = rate 1/2-1/3 turbo 1 = rate 1/2-1/7 turbo
MODE3	0 = turbo decoder 1 = turbo and Viterbi decoder
MODE4	0 = rate 1/2-1/3 Viterbi 1 = rate 1/2-1/4 Viterbi
MODE[6:5]	0 = 4K Interleaver 1 = 8K Interleaver 2 = 12K Interleaver 3 = 16K Interleaver

For MODE[6:5] = 0, 1, 2 and 3, the decoder uses 2, 4, 6 and 8 BlockRAMs, respectively.

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

$$C = A\sigma^2 \sqrt{m} / 2. \tag{1}$$

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

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

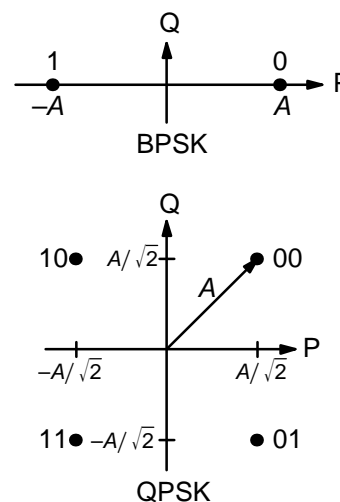


Figure 2: BPSK and QPSK signal sets.

E_b/N_0 is the energy per bit to single sided noise density ratio and $R = 1/(n+4/K)$, $n = 2-7$, $K = 1784(KS+1)$ is the code rate. 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 $C = 0$. Max-Log-MAP operation is also enabled when MODE0 is low.

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

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 $\sigma^2 = 1.39988$. Assuming $A = 16$ we have from (1) that $C = 11$ to the nearest integer.

Table 4: Quantisation for R0I, R1I and R2I.

Decimal	Binary	Range
31	011111	$30.5 \leftrightarrow \infty$
30	011110	$29.5 \leftrightarrow 30.5$
⋮	⋮	⋮
2	000010	$1.5 \leftrightarrow 2.5$
1	000001	$0.5 \leftrightarrow 1.5$
0	000000	$-0.5 \leftrightarrow 0.5$
32	100000	$-0.5 \leftrightarrow 0.5$
33	100001	$-1.5 \leftrightarrow -0.5$
34	100010	$-2.5 \leftrightarrow -1.5$
⋮	⋮	⋮
62	111110	$-30.5 \leftrightarrow -29.5$
63	111111	$-\infty \leftrightarrow -30.5$

Figure 3 gives a block diagram of the PCD04C 16 state turbo decoder. The number of turbo decoder half-iterations is given by NI , ranging from 0 to 255. $NI = 2I - 1$ where I is the number of iterations. This is equivalent to 0.5 to 128 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

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

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 4.3 Mbit/s for $K = 1784$ and $L = 265$ to 4.9 Mbit/s for $K = 16056$ 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 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 be used to adjust decoder performance is SCLZ which ranges from 1 to 32. The extrinsic information is scaled by

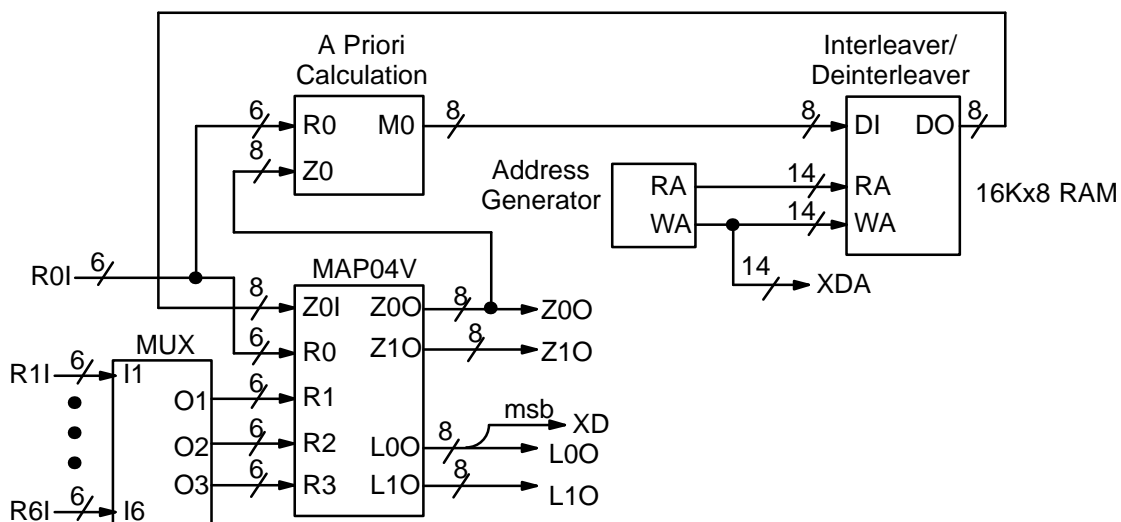


Figure 3: Simplified block diagram of PCD04C 16 state turbo decoder.

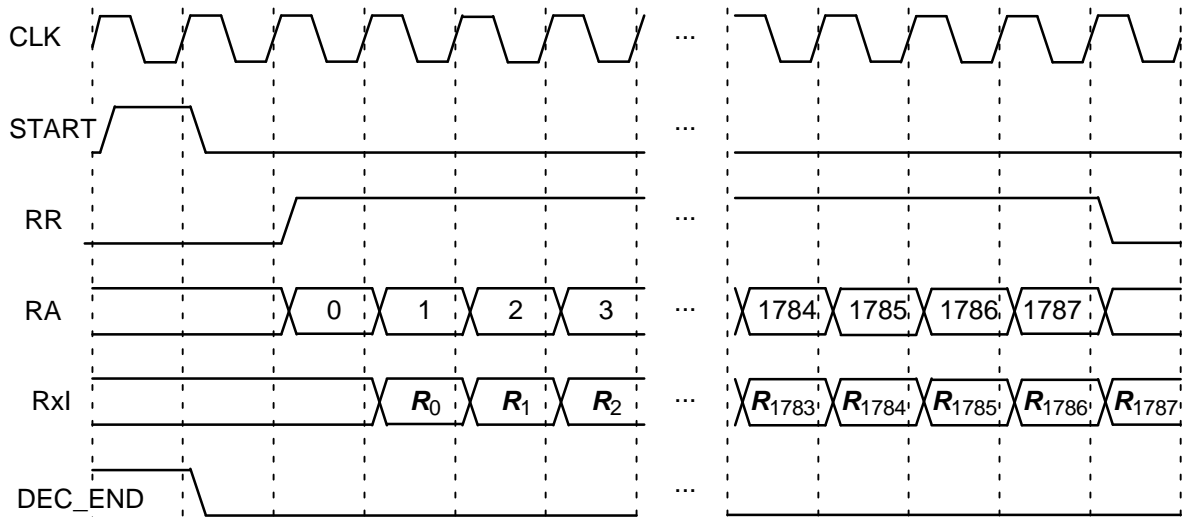


Figure 4: Turbo Decoder Input Timing ($K = 1784$).

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 $(K+4) \times 6n$, $n = 2$ to 7. Note that the CCSDS standard only specifies $n = 2, 3, 4$ and 6 and $K = 1784, 3568, 7136$ and 8920 corresponding to $KS = 0, 1, 3$ and 4, respectively. The interleaver parameters for $K = 16384$ is currently under study.

The received data ready signal RR goes high to indicated the data to be read from the address given by RA[13:0]. Table 5 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 X', 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), $g^2(D) = 1+D^2+D^4$ (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 then iteratively decodes the received data for $NI+1$ half iterations, rereading the received data for each half iteration for $K+4$ CLK cycles. The signal RR goes high for $K+4$ clock cycles while data is being output. Figure 4 illustrates the decoder timing where the data is input on the first half iteration.

If the START signal goes high while decoding, the decoder is reset and decoding starts anew. 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[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.

The bus ERR[3:0] is a channel error estimator output. For even NI , ERR0 is the exclusive OR (XOR) of XD and the sign bit of the input R0I. ERR[3:1] is the XOR of the parity bits of XD re-encoded with the sign bits of R3I, R2I and R1I. For odd NI , ERR0 is set to zero. This is because R0I is input in sequential order, not in the interleaved order of the output. ERR[3:1] is the same as for even NI . For rate 1/2 to 1/6, ERR[3:0] is punctured so that only encoded bits corresponding to actual

received data are compared with the re-encoded data.

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 ERR0 only. Thus, the decoder should be set to have an odd number (NI even) of half iterations.

Table 5: Input data format

Rate	Data	Output
1/2	R0I	X X
	R1I	Y1 Y1'
1/3	R0I	X
	R1I	Y1
	R2I	Y1'
1/4	R0I	X
	R1I	Y2
	R2I	Y3
	R3I	Y1'
1/5	R0I	X
	R1I	Y2
	R2I	Y3
	R3I	Y1'
	R4I	Y2'
1/6	R0I	X
	R1I	Y1
	R2I	Y2
	R3I	Y3
	R4I	Y1'
	R5I	Y3'
1/7	R0I	X
	R1I	Y1
	R2I	Y2
	R3I	Y3
	R4I	Y1'
	R5I	Y2'
	R6I	Y3'

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 $(N/2+1)(K+L+1)+1$.

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[3:0] is also output when XDE goes high. The outputs RA and RR are used to read the sign bits of R3I, R2I, R1I and R0I which are XORed with XD and the parity bits of XD re-encoded. The ERR[3:0] output is punctured according to the code rate.

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

The extrinsic (log-likelihood) information from the MAP decoder are output from Z0O[7:0] and Z1O[7:0] (L0O[7:0] and L1O[7:0]). The outputs Z0O and Z1O (L0O and L1O) corresponds to the data and parity, respectively, of the rate 1/2 MAP decoder. The information for both the data and tail bits are output and are in two's complement form.

L0O contains is the sum of R0I, the unchanged (not scaled or limited) Z0O for the current half iteration, and the scaled and limited Z0O from the previous half iteration. L1O is the sum of R1I or R2I and the unchanged Z1O for odd or even half iterations, respectively.

Z0O (L0O) is output every half iteration, using XDA as the write address and ZR0 is the ready address. For even half iterations (NA odd) Z0O (L0O) is interleaved. For odd half iterations (NA

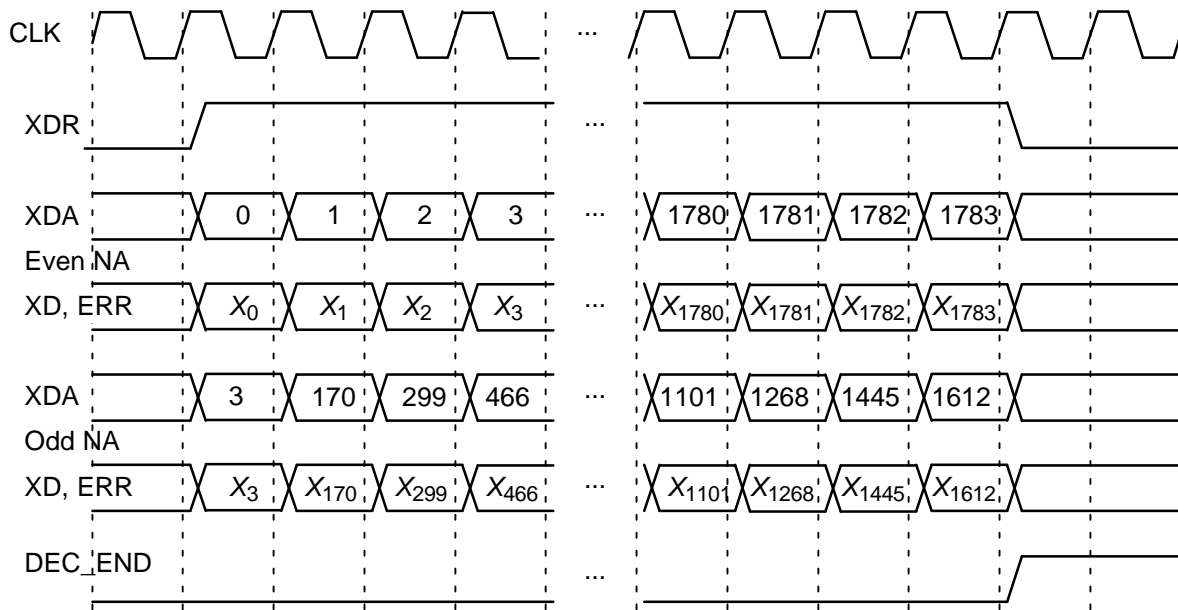


Figure 5: Turbo Decoder Output Timing ($K = 1784$).

even) Z0O (L0O) is not interleaved. Z0R is high for $K+4$ clock cycles every half iteration.

Z1O (L1O) is also output every half iteration, using ZA as the write address. Z1O (L1O) corresponds to the information for R1I and R2I for odd and even half iterations, respectively. The outputs ZR1 and ZR2 are the corresponding ready addresses. ZR1 and ZR2 goes high for $K+4$ clock cycles every odd and even half iteration, respectively.

Figure 7 illustrates how to connect Z0O (L0O) and Z1O (L1O) to three 16Kx8 memories. At the end of every decoding the memories will have stored the information for R0I, R1I and R2I.

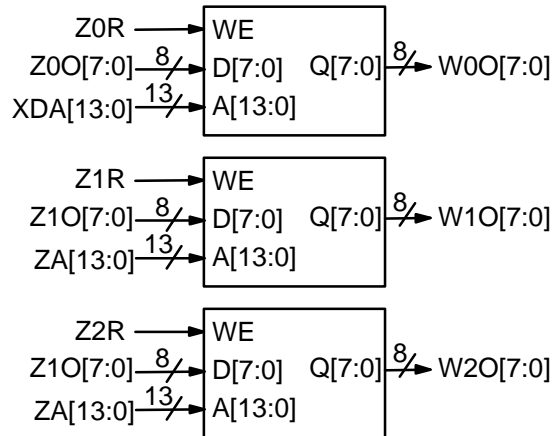


Figure 7: Output RAM for extrinsic information.

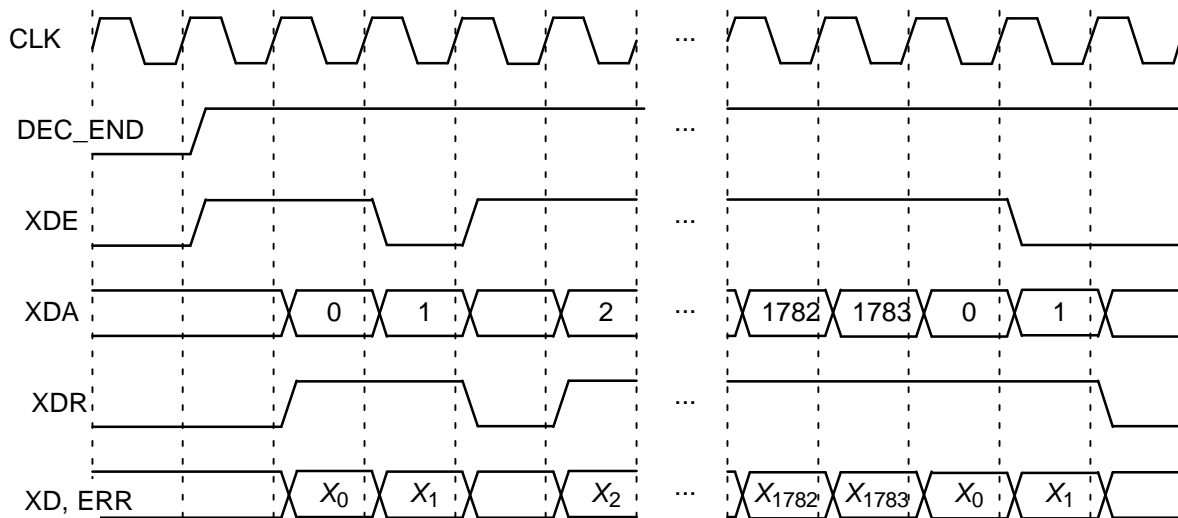


Figure 6: XDE Timing ($K = 1784$).

Simulation Software

Free software for simulating the PCD04C turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "pcd04csm request" in the subject header. The software uses an exact functional simulation of the PCD04C turbo decoder, including all quantisation and limiting effects.

After unzipping `pcd04csm.zip`, there should be `pcd04csm.exe` and `code.txt`. The file `code.txt` contains the parameters for running `pcd04csm`. These parameters are

<code>m</code>	Constituent code (CC) memory (2 to 4)
<code>nt</code>	Number of turbo code outputs (2 to 7)
<code>g0</code>	Divisor polynomial of CC in octal notation
<code>g1</code>	1st numerator polynomial of CC
<code>g2</code>	2nd numerator polynomial of CC
<code>g3</code>	3rd numerator polynomial of CC
<code>EbNomin</code>	Minimum E_b/N_0 (in dB)
<code>EbNomax</code>	Maximum E_b/N_0 (in dB)
<code>EbNoinc</code>	E_b/N_0 increment (in dB)
<code>optC</code>	Input scaling parameter (normally 0.65)
<code>ferrmax</code>	Number of frame errors to count
<code>NI</code>	Number of half iterations-1 (0 to 255)
<code>SLD</code>	MAP decoder delay select (0 or 1)
<code>LIMZ</code>	Extrinsic information limit (1 to 127)
<code>SCLZ</code>	Extrinsic information scale (1 to 32)
<code>M</code>	Stopping mode (0 to 3)=
<code>ZTH</code>	Extrinsic info. threshold (0 to 127)
<code>SI</code>	Select interleaver (0 or 1)
<code>KS</code>	Block length or Interleaver select (40-5114 for $S_I=0$ or 0-9 for $S_I=1$)
<code>q</code>	Number of quantisation bits (3 to 6)
<code>LOGMAP</code>	Log-MAP decoding (MODE0, 0 or 1)
<code>C4PIN</code>	Use five-bit C (MODE1, 0 or 1)
<code>state</code>	State file (0 to 2)
<code>s1</code>	Seed 1 (1 to 2147483562)
<code>s2</code>	Seed 2 (1 to 2147483398)
<code>read_x</code>	Use external information data (y or n)
<code>read_r</code>	Use external received data (y or n)
<code>out_dir</code>	Output directory
<code>in_dir</code>	Input directory
<code>C</code>	C (0 to 17)

Note that `g0`, `g1`, `g2` and `g3` are given in octal notation, e.g., `g0 = 23` $\equiv 10011_2 \equiv 1+D^3+D^4$. For the CCSDS standard, `m = 4`, `nt = 2, 3, 4` or `6`, `g0 = 23`, `g1 = 33`, `g2 = 25` and `g3 = 37`. The nominal turbo code rate is $1/nt$.

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

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

where σ^2 is the normalised noise variance given by (2) and $\text{mag}(\sigma)$ is the normalising magnitude resulting from an auto-gain control (AGC) circuit. We have

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

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

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

Although $\text{mag}(\sigma)$ is a complicated function, for high signal to ratio (SNR), $\text{mag}(\sigma) \approx 1$. For low SNR, $\text{mag}(\sigma) \approx \sigma \sqrt{2/\pi} \approx 0.798\sigma$. 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 $S_I = 0$ the 3GPP (UMTS) interleaver is used. This interleaver is valid from $K = 40$ to 5114. The block length is entered in `KS`. For $S_I = 1$ the CCSDS interleaver is used. The interleaver select value (0 to 9) is entered in `KS`.

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.

When the simulation is finished the output is given in, for example, file `k1784.dat`, where $K = 1784$. 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 bit error rate (`Pb`). Following this, the number of iterations, `na`, `berr`, `ferr`, and `Pb`, are given for each half iteration.

The following file was used to give the simulation results shown in Figure 8. Auto-stopping was used with up to 10 iterations. When iterating is stopped early, the `nasum` ($2 * \text{num} * \text{na}$), `berr` and `ferr` results at stopping are copied for each half iteration to the maximum iteration number. Thus, the $l = 10$ result is the performance one would measure with auto-stopping and $NI = 19$. The $l = 5$ curve shows the performance at 5 iterations with early stopping and $NI = 9$. Figure 9 shows the average number of iterations with E_b/N_0 .

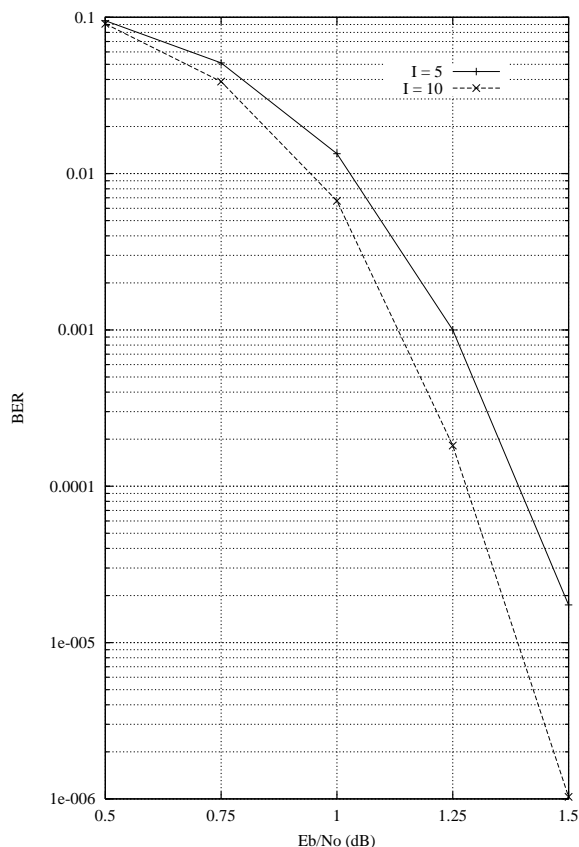


Figure 8: Performance with block size 1784 and auto-stopping (ZTH = 23).

```
{m nt g0 g1 g2 g3}
4 2 23 33 25 37
{EbNomin EbNomax EbNoinc optC ferrmax}
0.50 2.50 0.25 0.65 64
{NI SLD LIMZ SCLZ M ZTH SI}
19 1 96 32 3 23 1
{KS q LOGMAP C4PIN}
0 6 1 1
{state s1 s2}
0 12345 67890
{read_x read_r out_dir in_dir C}
n n output input 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`

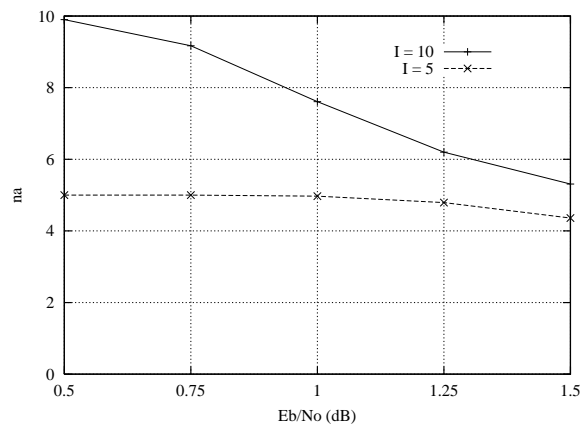


Figure 9: Average number of iterations with block size 1784 and auto-stopping (ZTH = 23).

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.

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 6-bit

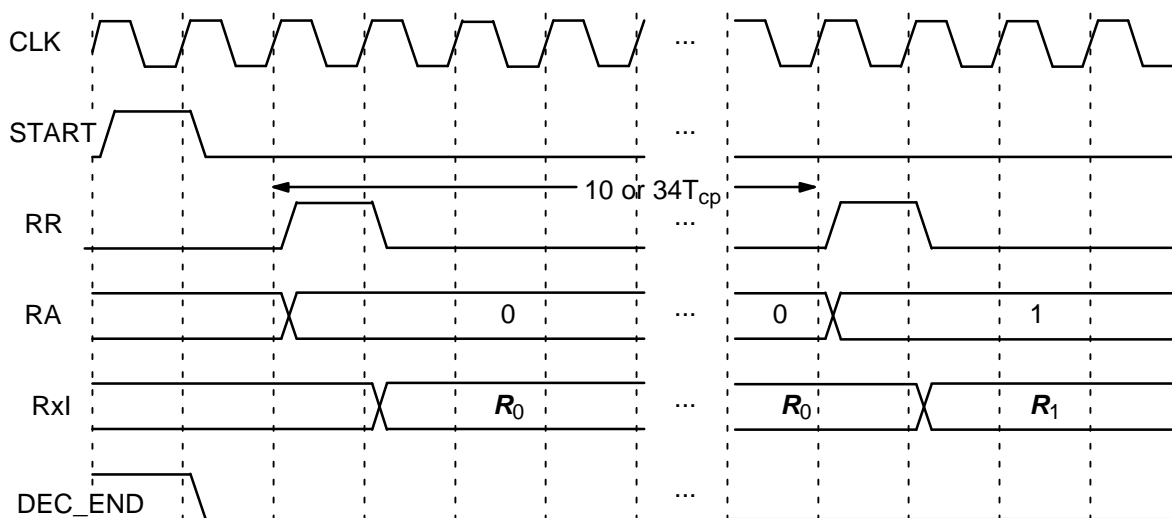


Figure 10: Viterbi Decoder Input Timing.

quantised received data. For rate 1/2 operation, R2I to R6I are not used. For rate 1/3 operation R3I to R6I are not used. For rate 1/4 operation R4I to R6I are not used.

The input SM selects 64 states (constraint length 7) when low and 256 states (constraint length 9) when high. The input DELAY when low selects either a delay of 70 or 72 (for 64 or 256 states). When high a delay of 134 or 136 (for 64 and 256 states) is selected. Table 6 shows the codes selected with the number of states and code rate.

Table 6: Convolutional Codes.

SM	N	G0I	G1I	G2I	G3I
0	2	171	133	–	–
0	3	171	133	165	–
0	4	173	167	135	111
1	2	753	561	–	–
1	3	557	663	711	–
1	4	765	671	513	473

The CCSDS standard only specifies the rate 1/2 64 state convolutional code with G1I inverted. This inversion can be simply performed by placing an inverter before the R1I5 input.

The decoder first inputs the received data from address 0 to K-1. The tail is then input from address K to K+5 for 64 states and K+7 for 256 states. 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[3:0] is the XOR of the sign bits of R3I, R2I, R1I and R0I with the corresponding re-encoded decoded output bits. This allows an estimate of the channel BER.

Figure 10 shows the Viterbi decoder input timing. Two clock cycles are used to start decoding, with each decoded bit taking 10 clock cycles for 64 states or 34 clock cycles with 256 states.

Figure 11 shows the Viterbi decoder output timing. The input XDE is not used either during or after Viterbi decoding.

The decoding speed is given by

$$f_d = \frac{F_d}{N_c(1 + D/K) + 2/K} \tag{7}$$

where F_d is the internal clock speed, N_c is the number of decoder clock cycles (10 or 34) and D is the Viterbi decoder delay in bits. For example, if $K = 1784$, $D = 134$ (SM = 0, DELAY = 1), $N_c = 10$ (SM = 0) and $F_d = 50$ MHz, decoding speed is 4.65 Mbit/s.

Ordering Information

- SW-PCD04C-SOS (SignOnce Site License)
- SW-PCD04C-SOP (SignOnce Project License)
- SW-PCD04C-VHD (VHDL ASIC License)
- SW-PCD04C-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 not include the Viterbi decoder which must be ordered separately (see the VA08V data sheet). The University license is only available to tertiary educational institutions such as universities and colleges and is limited to n instan-

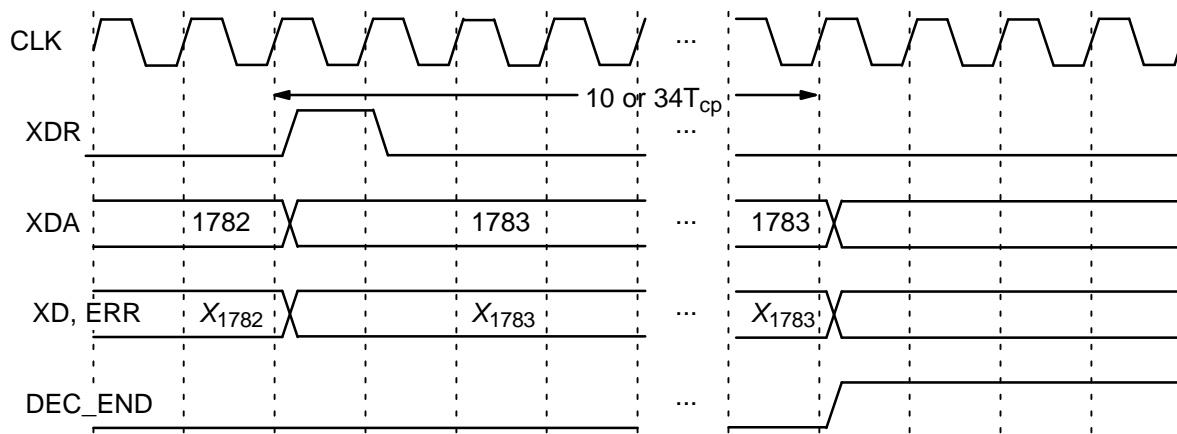


Figure 11: Viterbi Decoder Output Timing (K = 1784).

tations 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] Consultive Committee for Space Data Systems, "Recommendation for space data system standards: TM Synchronization and channel coding," CCSDS 131.0-B-1, Blue Book, Sep. 2003.
- [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 does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its copyrights or any rights of others. *Small World Communications* reserves the right to make changes, at any time, in order to improve performance, function or design and to supply the best product possible. *Small World Communications* will not assume re-

sponsibility for the use of any circuitry described herein. *Small World Communications* does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. *Small World Communications* assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. *Small World Communications* will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

© 2003–2008 *Small World Communications*. All Rights Reserved. XILINX is a registered trademark of Xilinx, Inc. All XC-prefix product designations, Virtex and Xilinx Foundation Series are trademarks of Xilinx, Inc. All other trademarks and registered trademarks are the property of their respective owners.

Supply of this IP core does not convey a license nor imply any right to use turbo code patents owned by France Telecom, GET or TDF. Please contact France Telecom for information about turbo codes licensing program at the following address: France Telecom R&D – VAT/Turbo-codes, 38 rue du Général Leclerc, 92794 Issy Moulineaux Cedex 9, France.

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