PCD04C Features

Turbo Decoder

- 16 state CCSDS compatible
- Rate 1/2 to 1/7
- Interleaver sizes from 1784 to 16056 bits
- Up to 342 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 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 (Optional)

- 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 9.9 Mbit/s (256 state) or 33.7 Mbit/s (64 state)
- 6–bit signed magnitude input data
- Estimated channel error output

- Available as VHDL core for AMD–Xilinx FPGAs under SignOnce IP License. ASIC, Intel/Altera, Lattice and Microsemi cores available on request.

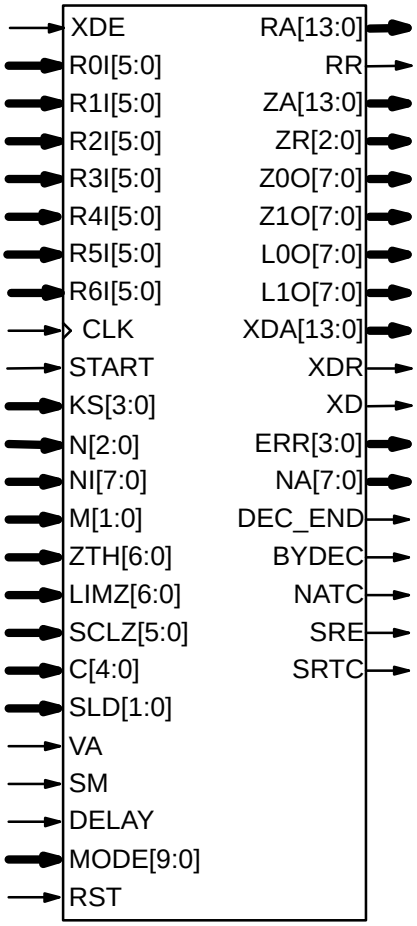

Figure 1: PCD04C schematic symbol.

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 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 $n(K+4)$ where the nominal code rate is $1/n$.

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. Max–log–MAP decoding increases speed by about 55% compared to log–MAP decoding.

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. The extrinsic information of both the data and first parity bits of the constituent code can also be output. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance.

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, minimis-

ing complexity. Maximum traceback lengths of 48 or 96 bits for 64 states or 60 or 120 bits for 256 states can be selected. 6–bit quantisation is used.

Figure 1 shows the schematic symbol for the PCD04C 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)	Turbo* Mbit/s	K=9 Mbit/s	K=7 Mbit/s
XC7S50–1	10.472	9.0	2.6	9.1
XC7S50–2	8.584	11.3	3.3	11.4
XC7A35T–1	10.534	9.2	2.7	9.3
XC7A35T–2	8.604	11.2	3.3	11.4
XC7A35T–3	7.661	12.6	3.7	12.8
XC7K70T–1	6.862	14.1	4.2	14.3
XC7K70T–2	5.556	17.4	5.2	17.7
XC7K70T–3	5.114	18.9	5.6	19.2
XCKU035–1	5.420	17.9	5.3	18.1
XCKU035–2	4.528	21.4	6.3	21.7
XCKU035–3	4.000	24.2	7.2	24.6
XCKU3P–1	3.751	25.8	7.7	26.2
XCKU3P–2	3.151	30.8	9.1	31.2
XCKU3P–3	2.916	33.3	9.9	33.7

*Large log–MAP, 5 iterations, no Z/L outputs, *K*=8920, SLD=1.

Table 2 shows the resources used for Kintex–7 devices The complexity for Virtex–5, Spartan–6, Virtex–6, 7–Series, UltraScale and UltraScale+ devices are similar to that for Kintex–7. The MODE[9:0] 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. An interleaver up to 16K in size is used, requiring one to eight 18KB Block RAMs.

Signal Descriptions

BYDEC Decoder Busy
C MAP Decoder Constant
0–9 (MODE[1] = 0)
0–17 (MODE[1] = 1)
CLK System Clock

Table 2: Resources used

Log MAP	Turbo Rates	Viterbi Rates	Z/L	SLD	6–Input LUTs
Max	1/2–1/7	–	No	0–1	4110
Small	1/2–1/7	–	No	0–1	5825
Large	1/2–1/7	–	No	0–1	6147
Small	1/2–1/3	–	No	0–1	4659
Small	1/2–1/7	–	Yes	0–1	6813
Small	1/2–1/7	1/2–1/4	No	0–1	6689
Small	1/2–1/7	–	No	0–2	6082

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)
NATC Half Iteration Number Terminal Count
NI Number of Half Iterations (0–255)
NI = 2*I*–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 138
1 = delay 266
2 = delay 522
SM Viterbi Decoder Memory
0 = 64 state (constraint length 7)
1 = 256 state (constraint length 9)
SRE Input RAM Enable
SRTC Input RAM Terminal Count
START Decoder Start
VA Viterbi Decoder Select
0 = turbo decoder
1 = Viterbi decoder
XD Decoded Data

XDA	Decoded Data, Z0O and L0O Address
XDE	Decoded Data Enable
XDR	Decoded Data Ready
Z0O	Data Extrinsic Information
Z1O	Parity Extrinsic Information
ZA	Z1O and L1O Address
ZR	Extrinsic Information Ready
ZTH	Early Stopping Threshold (1–127)

Table 3 describes each of the MODE[9:0] inputs that are used to select various decoder implementations. Note that MODE[9:0] 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 3: MODE selection

Input	Description
MODE[0]	0 = max–log–MAP 1 = log–MAP
MODE[1]	0 = small log–MAP (C[4] = 0) 1 = large log–MAP
MODE[2]	0 = rate 1/2–1/3 turbo 1 = rate 1/2–1/7 turbo
MODE[3]	0 = turbo decoder 1 = turbo and Viterbi decoder
MODE[4]	0 = rate 1/2–1/3 Viterbi 1 = rate 1/2–1/4 Viterbi
MODE[5]	0 = Z1O and L1O Disable 1 = Z1O and L1O Enable
MODE[8:6]	0 = 2K Interleaver (1x18KB) 1 = 4K Interleaver (2x18KB) 2 = 6K Interleaver (3x18KB) 3 = 8K Interleaver (4x18KB) 4 = 10K Interleaver (5x18KB) 5 = 12K Interleaver (6x18KB) 6 = 14K Interleaver (7x18KB) 7 = 16K Interleaver (8x18KB)
MODE[9]	0 = SLD ≤ 1 1 = SLD ≤ 2

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.

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

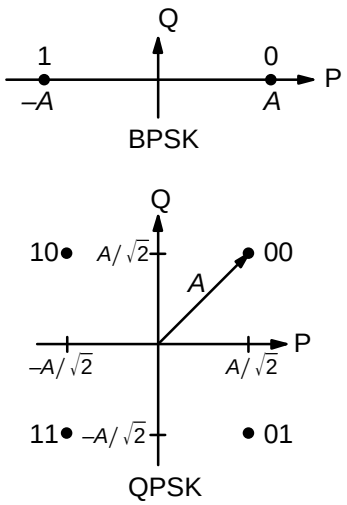

Figure 2: BPSK and QPSK signal sets.

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

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

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

E_b/N_0 is the energy per bit to single sided noise density ratio and $R = 1/(n(1+4/K))$, 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 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.

For fading channels each received value r_k at time k should be scaled by $(A_m\sigma_m^2)/(A_k\sigma_k^2)$ where A_k and σ_k^2 are the no–noise amplitude and normalised variance of r_k and m corresponds to the time index of the smallest σ_k^2. The value of C should be determined by A_m and σ_m^2. 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

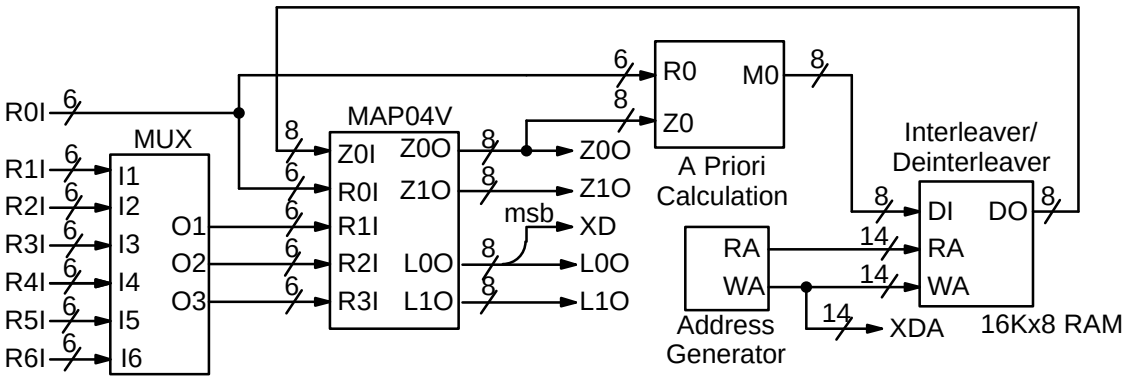

Figure 3: Simplified block diagram of PCD04C 16 state turbo decoder.

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

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.

Table 4: Quantisation for R0I to R6I.

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

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 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

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

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 1/7. For max–log–MAP decoding we recommend SCLZ = 23. The NA output can be used to

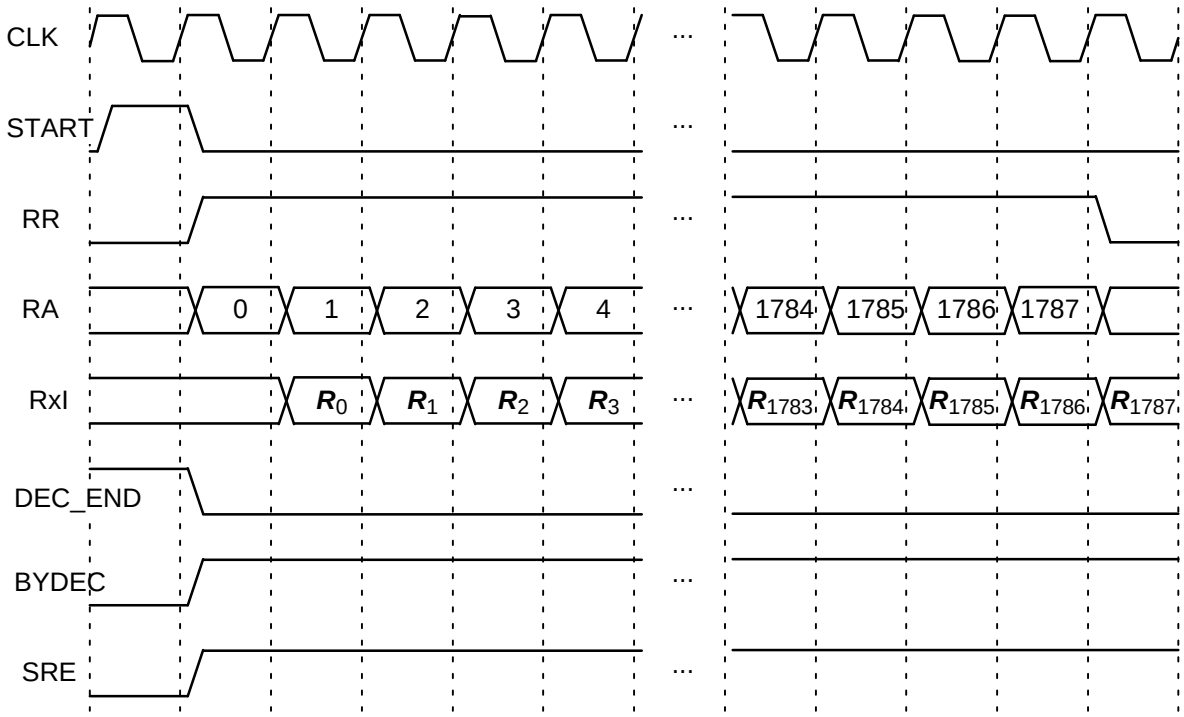

Figure 4: Turbo decoder input timing for first half iteration (K = 1784).

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)x6n, 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 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. Figures 4 and 5 illustrates the decoder timing for the first and last half iterations, respectively.

Signals SRE and SRTC are used to indicate when the input RAM is being used. SRE goes high after START goes high and goes low during the last half iteration after the last symbol has been read. SRTC goes high during the last half iteration while the last symbol is being read.

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[13:0] can be used as the write address to a buffer RAM. After the

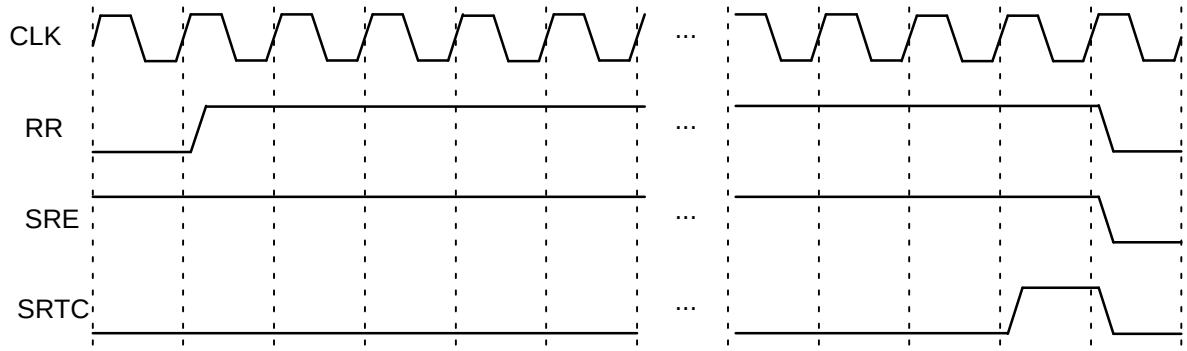

Figure 5: Turbo decoder input timing for last half iteration.

block has been written to the buffer RAM, the decoded block can be sequentially read from the buffer RAM.

Table 5: 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	1/7	R0I	X
	R2I	Y3		R1I	Y1
	R3I	Y1′		R2I	Y2
1/5	R0I	X		R3I	Y3
	R1I	Y2		R4I	Y1′
	R2I	Y3		R5I	Y2′
	R3I	Y1′		R6I	Y3′
	R4I	Y2′			

The bus ERR[3:0] is a channel error estimator output. For even NA[7:0], 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 of the sign bits of R1I (rates 1/2 to 1/7), R2I (rates 1/4 to 1/7) and R3I (rates 1/6 and 1/7). 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[7:0], 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 (rates 1/4 and 1/5), R4I (rates 1/5 and 1/6), R5I (rates 1/6 and 1/7) and R6I (rate 1/7). 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.

The DEC_END signal is low during decoding. At the end of decoding, DEC_END goes high. Signal BYDEC goes high after START goes high and goes low during the last half iteration while the last decoded bit is output. SIgnal NATC goes high during the last half iteration while the last decoded bit is output. 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.

To enable operation with different clocks, the outputs DEC_END, BYDEC, NATC, SRE and SRTC are all registered.

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 the interleaver memory (regardless of the number of iterations). XDE is disabled while the decoder is iterating. Figure 7 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 R0I and R1I (rates 1/2 to 1/7), R2I (rates 1/4 to 1/7) and R3I (rates 1/6 and 1/7) which are XORed with XD and the parity bits of XD re–encoded. The ERR[3:1] outputs are punctured according to the first constituent encoder puncturing pattern.

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

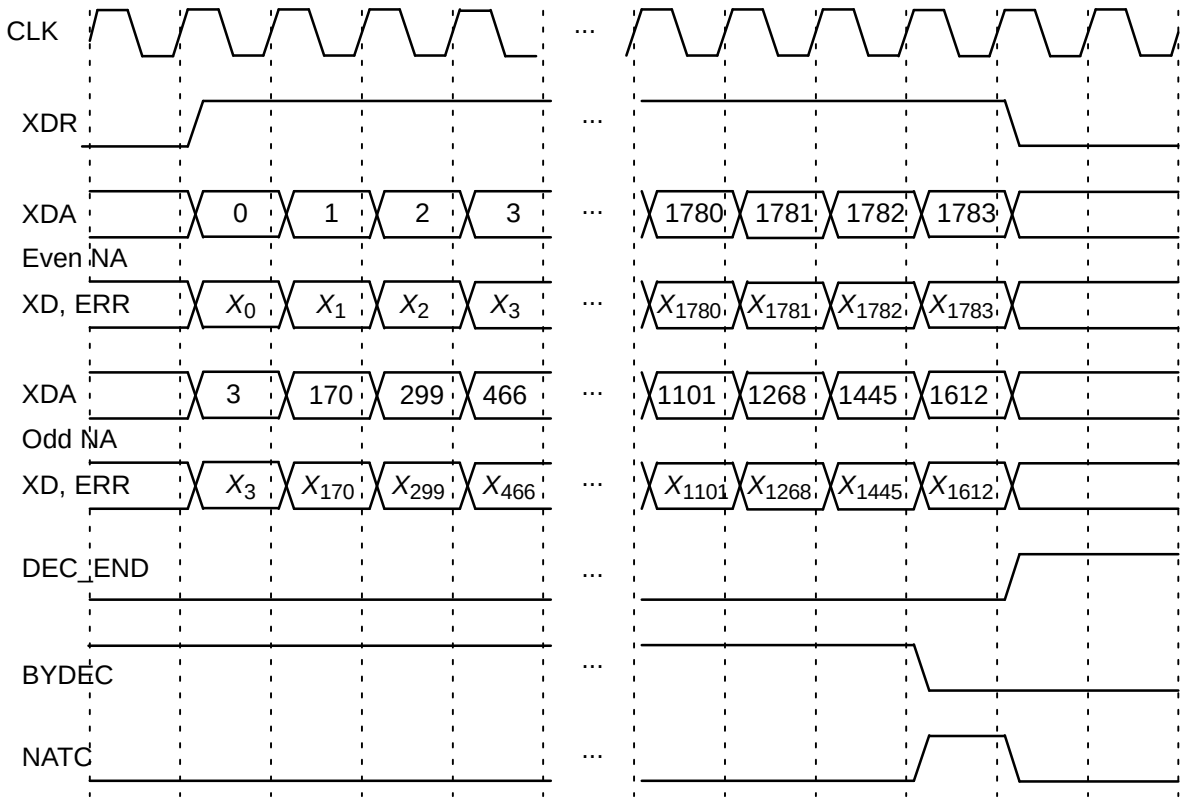

Figure 6: Turbo Decoder Output Timing (K = 1784).

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.

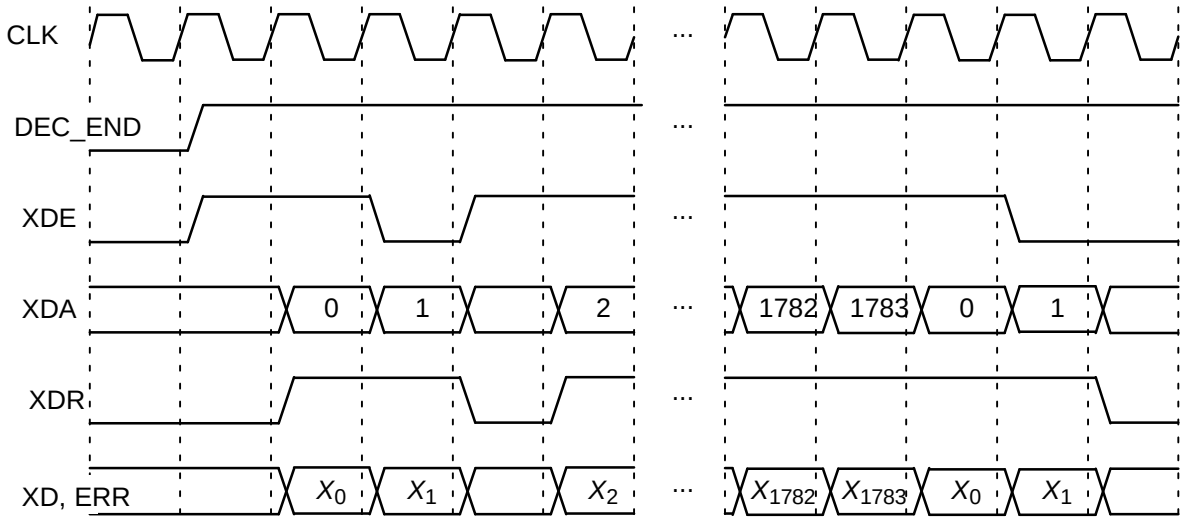

Figure 7: XDE Timing (K = 1784).

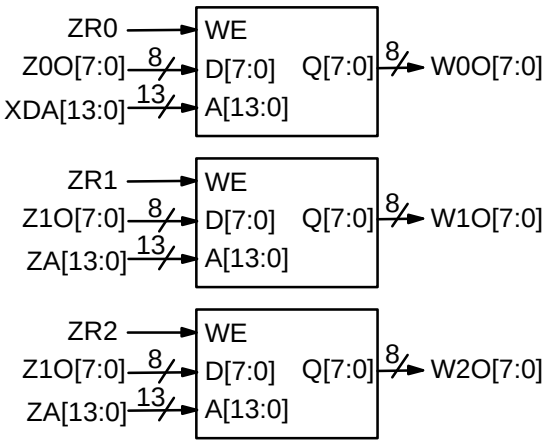

Figure 8: Output RAM for extrinsic information.

As the first constituent code is stronger than the second constitiuent 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.

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. For rate 1/3, 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 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 8 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.

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 "pcd04csim 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 pcd04csim.zip, there should be pcd04csim.exe and code.txt. The file code.txt contains the parameters for running pcd04csim. These parameters are

`m`	Constituent code (CC) memory (2 to 4)
`nt`	Number of turbo code outputs (2 to 7)
`g0`	Divisor polynomial of CC in octal notation
`g1`	1st numerator polynomial of CC
`g2`	2nd numerator polynomial of CC
`g3`	3rd 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 (0.0 to 1.0)
`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 255)
`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)
`SI`	Select interleaver (0 or 1)
`KS`	Block length or Interleaver select (17–16384 for `SI`=0 or 0–9 for `SI`=1)
`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)
`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

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 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 mag(σ) is the normalising magnitude resulting from an auto–gain control (AGC) circuit. We have

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

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 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 6 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 6: Simulation parameters

R	E_b/N_0 (dB)	optC	A	C	SCLZ	BER 10^{-2}
1/6	–0.35	0.35	6.55	11	31	3.31
1/4	0.0	0.35	8.19	7	31	2.52
1/3	0.2	0.37	9.02	6	31	2.79
1/2	0.8	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) is below or equal to `Pfmin` or the BER is below or equal to `Pbmin`. Each simulation point continues until the number of frame errors is equal to `ferrmax`. If `ferrmax` = 0, then only one frame is simulated.

An optional Genie aided stopping mode can be selected by setting M = 4. This will stop the decoder from further iterations when the Genie has detected there are no errors compared to the transmitted data. This allows a lower performance bound to be simulated, allowing fast simulations for various configurations at low bit error rates.

For `SI` = 0 the 3GPP2 (cdma2000) interleaver is used. This interleaver is valid from K = 17 to 16384. The block length is entered in `KS`. For SI = 1 the CCSDS interleaver is used. The interleaver select value (0 to 9) is entered in `KS`.

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 (`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.

```
{m nt g0 g1 g2 g3}
 4 2  23 33 25 37
{EbNomin EbNomax EbNoinc optC}
 0.8     1.5     0.1     0.42
{ferrmax Pfmin Pbmin}
 256     1e-99 1e-5
{NI  SLD LIMZ  SCLZ  M  ZTH  SI}
 19  1   96    28    1  23   1
{KS  q  LOGMAP  C4PIN  enter_C  C}
 0   6  1       1      y        5
{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, e.g., by the program being closed or your computer crashing. For normal simulations, `state` = 0.

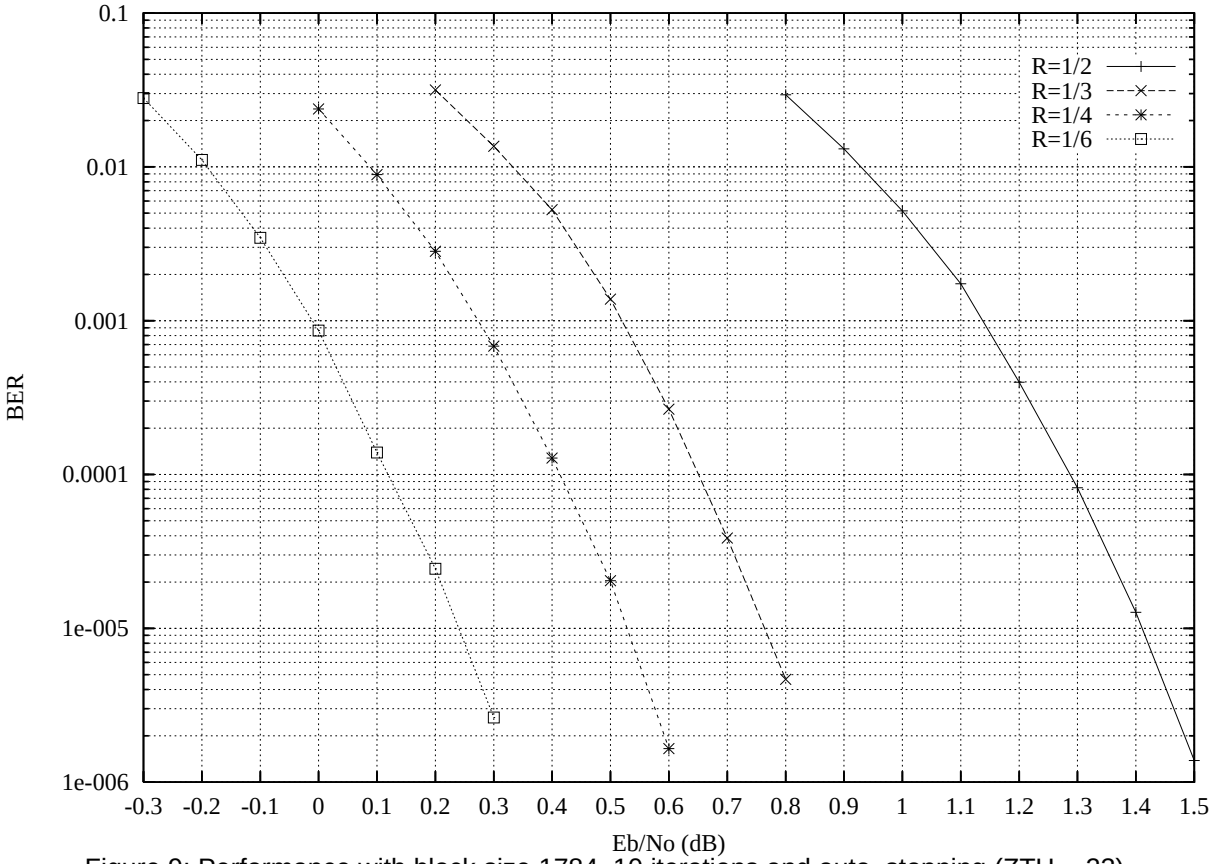

Figure 9: Performance with block size 1784, 10 iterations and auto–stopping (ZTH = 23).

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.

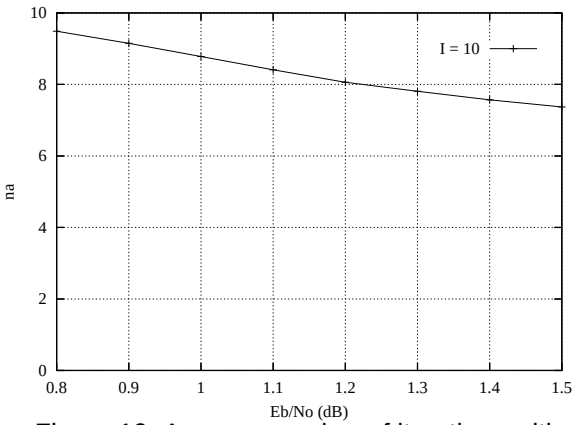

Figure 10: Average number of iterations with block size 1784 and auto–stopping (ZTH = 23).

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`.

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

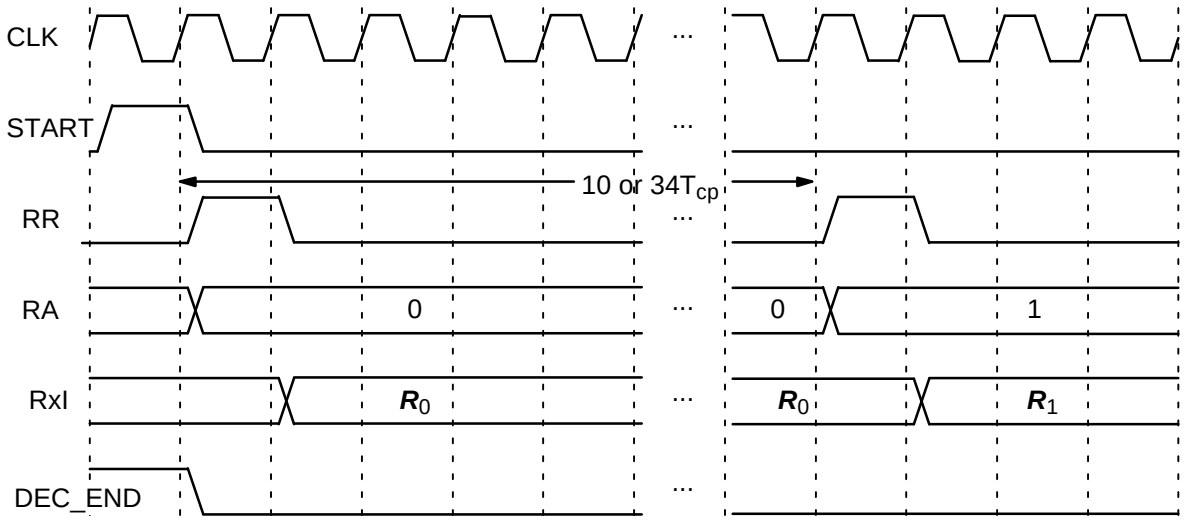

Figure 11: Viterbi Decoder Input Timing.

$$R[i,j] = A*(1–2*Y[i,j]+N[i,j])$$

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`.

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 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. Note that for correct operatation, MODE[8:6] must be greater than zero.

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 7 shows the codes selected with the number of states and code rate.

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.

Table 7: 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 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 11 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 12 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 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 = 100 MHz, decoding speed is 9.3 Mbit/s.

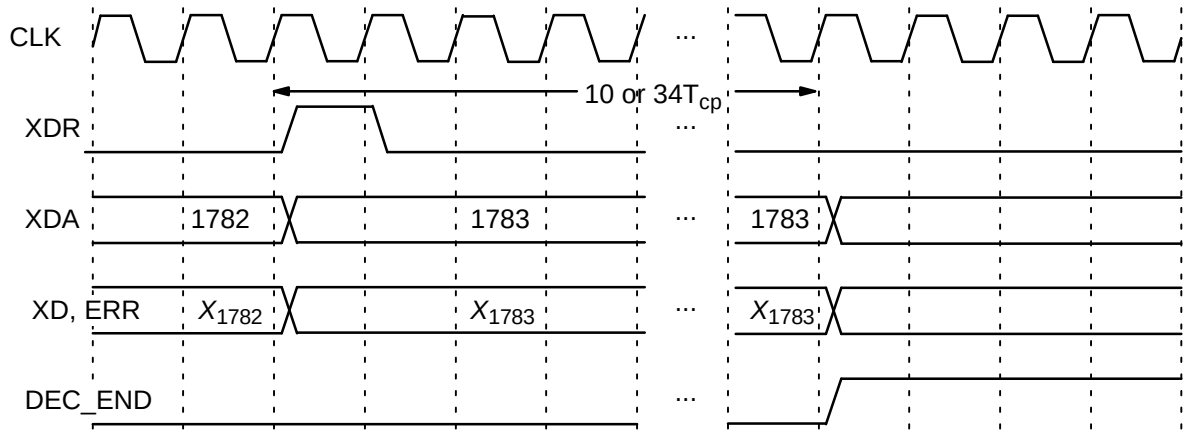

Figure 12: Viterbi Decoder Output Timing (K = 1784).

Ordering Information

SW–PCD04C–SOS (SignOnce Site License)
SW–PCD04C–SOP (SignOnce Project License)
SW–PCD04C–VHD (VHDL ASIC License)

All licenses include Xilinx VHDL cores. The above licenses do not include the Viterbi decoder which must be ordered separately (see the VA08V data sheet). 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

- 1.0 27 Jun. 2003. First release.
- 1.2 15 Aug. 2003. Improved decoder speed. Added average number of half iterations for I = 5 in Figure 10. Corrected decoder delay in Viterbi decoder example.
- 1.3 18 Jan. 2005. Added Spartan–3 performane and complexity. Updated Virtex–E and Virtex–II performance. Corrected KS input range.
- 1.31 24 May 2005. Added Virtex–II Pro and Virtex–4 performance and complexity.

- 1.32 10 Jun. 2005. Updated description of using external input data for simulation software.
- 1.40 21 July 2008. Added MODE[7] input. Changed ERR output to ERR[3:0]. Added Virtex–5 complexity and performance. Deleted Virtex–E and Virtex–II performance and complexity. Improved Virtex–4 performance. Corrected code rate *R* equation and data length *K* range. Updated CCSDS reference.
- 1.43 4 Oct. 2010. Deleted Virtex–II Pro performance. Updated Virtex–5 performance. Added Spartan–6 and Virtex–6 performance. Added description for quantisation less than six bits.
- 1.45 28 Oct. 2010. Improved Virtex–4 and Virtex–5 complexity. Corrected XDA description.
- 1.46 15 Jan. 2011. Added version history. Added Genie aided eariy stopping for simulation software.
- 1.47 2 Feb. 2011. Updated BER simulation software to allow external C input and BER and FER minimum values. Added simulation parameters table. Updated recommended SCLZ and M values. Changed optional BER simulation software interleaver from UMTS to 3GPP2.
- 1.48 2 Mar. 2011. Updated BER simulation curves to include rate 1/3, 1/4 and 1/6 results.
- 1.49 9 Jun. 2011. Changed SLD input to SLD[1:0]. Changed MAP decoder delay *L* values so as to simplify decoding speed equation. Corrected fading channel information. Updated Figure 3.
- 1.50 11 Jan. 2013. Clarified explanation of ERR[3:0] outputs.
- 1.51 25 July 2017. Deleted Spartan–3, Spartan–6 and Virtex–4 performance. Added Zynq–7, Artix–7 and Kintex–7 performance. Updated decoder speed and resources used.
- 1.52 16 Oct. 2017. Updated decoder speed and resources used. Updated BER and number of iterations performance. Startup delay reduced by one clock cycle.
- 1.53 24 Oct. 2017. Updated BER and number of iterations performance.
- 1.54 9 Nov. 2017. Added `out_screen` to code.txt.
- 1.55 14 Dec. 2017. Corrected code.txt M value.
- 1.56 28 Dec. 2017. Updated complexity with Viterbi decoder.
- 1.57 5 Jan. 2018. Added “(log–MAP)” to Features internal clock. Rearranged schematic symbol outputs.
- 1.58 22 Apr. 2020. Updated CCSDS reference.
- 1.59 8 Oct. 2021. Added BYDEC, NATC, SRE and SRTC outputs. Changed MODE[7] to MODE[5]. Added MODE[8] input. Deleted Virtex–5, Virtex–6 and Zynq–7 performance. Added Spartan–7, Ultrascale and Ultrascale+ performance. Updated LUT complexity.
- 1.60 8 Aug. 2022. Added MODE[9] pin. Updated decoder speed and complexity.