



PCD04I Features

Turbo Decoder

- 16 state constituent decoder
- Rate 1/2, 1/3, 1/4 or 1/5
- Inmarsat compatible
- Data lengths from 1 to 4092, 6140, 12284 or 22524 bits
- External interleaver address table
- Up to 342 MHz internal clock (log-MAP)
- Up to 32.4 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
- 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 length from 1 to 32760 (256 state) or 32762 (64 state) bits
- Up to 7.9 Mbit/s (256 state) or 27.0 Mbit/s (64 state)
- 6-bit signed magnitude input data
- Estimated channel error output

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

Introduction

The PCD04I is a 16 state parallel concatenated error control turbo decoder. The interleaver address table is external to the core. Data lengths up to 4092, 6140, 12284 or 22524 bits can be implemented. Interleaver sizes are 4 bits greater, i.e., either 4096, 6144, 12288 or 22528 bits. Turbo code rates from 1/2 to 1/5 can be selected. The un-interleaved data is terminated with a tail. The data and this tail are interleaved. No tail is added to the interleaved data, implying that the end state of the interleaved data is unknown. The input block size is K . The interleaver size is $K+4$.

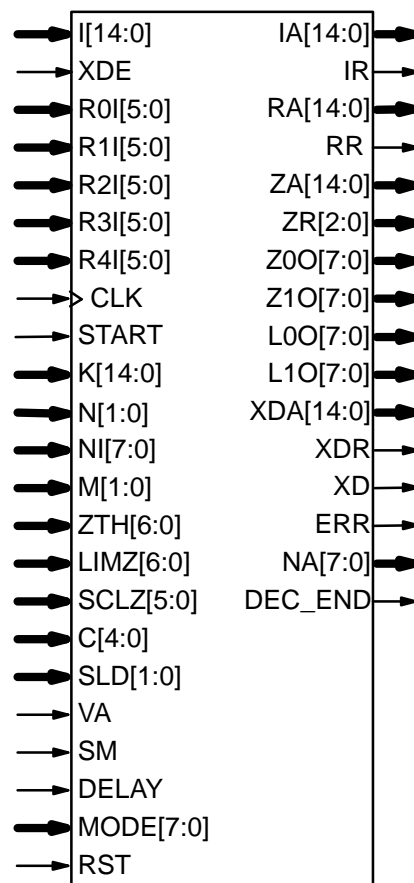


Figure 1: PCD04I schematic symbol.

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 PCD04I 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, 64, or 128. 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 parity bits of the constituent code are also output. The decoder is Inmarsat compatible.

The VA08V Viterbi decoder core can be used with the PCD04I 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, minimising 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.

The turbo decoder can achieve up to 32.4 Mbit/s with 5 iterations using a 342 MHz internal clock ($K = 5116$). Max-log-MAP decoding increases speed by about 50%. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance. The Viterbi decoder can achieve 7.9 Mbit/s with 256 states and 27.0 Mbit/s with 64 states with $K = 504$ and 506, respectively.

Figure 1 shows the schematic symbol for the PCD04I 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
XC7S50-1	10.517	9.0	2.2	7.5
XC7S50-2	8.568	11.0	2.7	9.2
XC7A35T-1	10.340	9.1	2.2	7.6
XC7A35T-2	8.480	11.2	2.7	9.3
XC7A35T-3	7.585	12.5	3.0	10.4
XC7K70T-1	6.795	13.9	3.4	11.6
XC7K70T-2	5.524	17.1	4.1	14.3
XC7K70T-3	5.132	18.5	4.5	15.4
XCKU035-1	5.531	17.1	4.1	14.2
XCKU035-2	4.687	20.2	4.9	16.8
XCKU035-3	4.025	23.5	5.7	19.6
XCKU3P-1	3.723	25.5	6.2	21.2
XCKU3P-2	3.170	29.9	7.3	24.9
XCKU3P-3	2.923	32.4	7.9	27.0

*Large log-MAP, 5 iterations, $K = 5116$, 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 Virtex-5. The MODE[7:0] inputs can be used to select various decoder implementations. The input/output memory is not included. Only one global clock is used. An interleaver up to 22K in size is used, requiring from 2 to 11 18KB Block RAMs. No other resources are used. Note that Zynq devices up to

XC7Z020 and from XC7Z030 use programmable logic equivalent to Artix-7 and Kintex-7 devices, respectively.

Table 2: Resources used

Log MAP	Turbo Rates	Viterbi Rates	SLD	6-Input LUTs
Max	1/3	–	0–1	3264
Small	1/3	–	0–1	5635
Large	1/3	–	0–1	6114
Small	1/3	–	0–2	5921
Small	1/2–1/3	1/2–1/3	0–1	6388
Small	1/2–1/5	1/2–1/4	0–1	6813

Signal Descriptions

C	MAP Decoder Constant 0–9 (MODE[1] = 0) 0–17 (MODE[1] = 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
I	Interleaver Address Input
IA	Interleaver Address Output
IR	Interleaver Address Ready
K	Data Length (1 – 4092, 6140, 12284 or 22524)
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 = rate 1/2 3 = rate 1/3 0 = rate 1/4 1 = rate 1/5 (turbo only)
NA	Half Iteration Number (0–255)
NI	Number of Half Iterations (0–255) NI = 2I–1 where I is number of iterations
R0I–R4I	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)
START	Decoder Start
VA	Viterbi Decoder Select
	0 = turbo decoder
	1 = Viterbi 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)

Table 3 describes each of the MODE[7:0] inputs that are used to select various decoder implementations. Note that MODE[7:0] 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 (C4 = 0) 1 = large log-MAP
MODE[2]	0 = rate 1/2–1/3 turbo 1 = rate 1/2–1/5 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[6:5]	0 = 4K Interleaver (2x18KB) 1 = 6K Interleaver (3x18KB) 2 = 12K Interleaver (6x18KB) 3 = 22K Interleaver (11x18KB)
MODE[7]	0 = SLD ≤ 1 1 = SLD ≤ 2

Turbo Decoder Parameter

For optimal performance, the maximum a posteriori (MAP) [1] 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 [2]) algorithm can provide up to 0.5 dB coding gain at low SNRs.

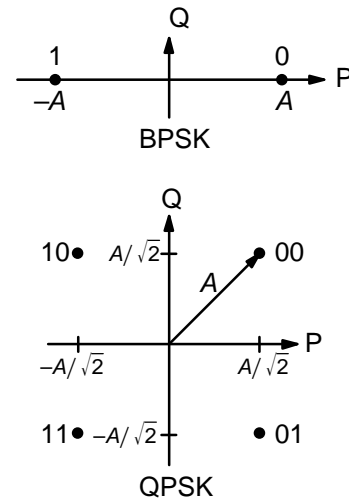


Figure 2: BPSK and QPSK signal sets.

Log-MAP operation is enabled when MODE[0] is high.

BPSK and QPSK Parameters

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 and $R = 1/(n(1+4/K))$ is the code rate where $n = 2$ to 5 and $K = 1$ to 22524. 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 [2] 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 be adjusted according to the received signal to noise ratio.

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).

Table 4: Quantisation for R0I, R1I and R2I.

Decimal	Binary	Range
31	011111	$30.5 \leftrightarrow \infty$
30	011110	$29.5 \leftrightarrow 30.5$
\vdots	\vdots	\vdots
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	$-2.5 \leftrightarrow -1.5$
\vdots	\vdots	\vdots
62	111110	$-30.5 \leftrightarrow -29.5$
63	111111	$-\infty \leftrightarrow -30.5$

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

For 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.

Example 1: Rate 1/3 BPSK code with $K = 5120$ operating at $E_b/N_0 = 0.3$ dB. From (2) we have $\sigma^2 = 1.40098$. Assuming $A = 9$ we have from (1) that $C = 6$ to the nearest integer.

16QAM Parameters

The Inmarsat standard has the following mapping in the inphase (I) and quadrature (Q) components:

I1	I0	I	Q1	Q0	Q
0	1	-3	0	1	-3
0	0	-1	0	0	-1
1	0	1	1	0	1
1	1	3	1	1	3

For a 2-D signal set consisting of (I,Q), I,Q = {-3,-1,1,3} the average energy is 10. This implies that the average signal amplitude is $\sqrt{10} = 3.16$.

Let the received 2-D signal be (x_k, y_k) where

$$x_k = s_k^I + \sqrt{10} n_k^I \quad (3)$$

$$y_k = s_k^Q + \sqrt{10} n_k^Q \quad (4)$$

where n_k^I and n_k^Q are the additive inphase and quadrature white Gaussian noise (AWGN), respectively, with zero mean and normalised variance σ^2 (2). We have $m = 4$ is the number of bits per symbol and $R = 1/2$ is the code rate (the Inmarsat code is punctured to give a rate of exactly a 1/2). We have multiplied n_k^I and n_k^Q by $\sqrt{10}$ in (3) and (4), respectively, to take into consideration that (2) is valid only if the signal set has been normalised to an energy of one.

In the Inmarsat code, I1 and Q1 are mapped with data bits and I0 and Q0 are mapped with parity bits (due to the extra puncturing four data bits are also mapped to I0 and Q0). We shall call the data bits d_k^I and d_k^Q and the parity bits p_k^I and p_k^Q . Since the in-phase and quadrature mapping are the same, we shall ignore the superscript I and Q notation in the following.

The likelihood ratio of the data bit d_k and the parity bit p_k are given by

$$r_k^d = \frac{P(d_k = 1|x_k)}{P(d_k = 0|x_k)} = \frac{\sum_{s=\{1,3\}} \exp\left(\frac{-1}{2\sigma^2 10} (x_k - s)^2\right)}{\sum_{s=\{-1,-3\}} \exp\left(\frac{-1}{2\sigma^2 10} (x_k - s)^2\right)} \quad (5)$$

$$r_k^p = \frac{P(p_k = 1|x_k)}{P(p_k = 0|x_k)} = \frac{\sum_{s=\{-3,3\}} \exp\left(\frac{-1}{2\sigma^2 10} (x_k - s)^2\right)}{\sum_{s=\{-1,1\}} \exp\left(\frac{-1}{2\sigma^2 10} (x_k - s)^2\right)} \quad (6)$$

Notice that we have a division by 10 in the exponential so as to normalise the signal set energy to one. Taking the log-likelihood ratios, we have

$$R_k^d = -\log_e r_k^d = [B(x_k - 1)^2 \text{ E } B(x_k - 3)^2] - [B(x_k + 1)^2 \text{ E } B(x_k + 3)^2] \quad (7)$$

$$R_k^p = -\log_e r_k^p = [B(x_k - 3)^2 \text{ E } B(x_k + 3)^2] - [B(x_k - 1)^2 \text{ E } B(x_k + 1)^2] \quad (8)$$

where

$$a \text{ E } b = \min(a, b) - C \ln(1 + e^{-|a-b|/C}) \quad (9)$$

and $C = 1/\ln \epsilon = \log_e e$ with

$$B = -\log_e e^{-1/(2\sigma^2 10)} = 0.05C/\sigma^2. \quad (10)$$

Using the fact that $(a + b) \text{ E } (a + c) = a + [b \text{ E } c]$ we can simplify (7) and (8) to

$$R_k^d = [Ax_k/2 \text{ E } A] - [(Ax_k/2 + A) \text{ E } 0] - Ax_k \quad (11)$$

$$R_k^p = ([3Ax_k/2 \text{ E } 0] - [Ax_k/2 \text{ E } 0] - Ax_k/2 + A) \quad (12)$$

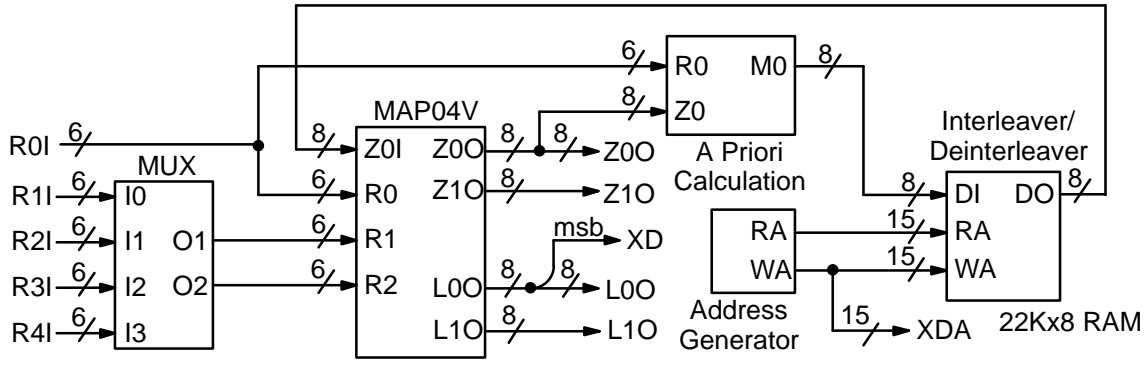


Figure 3: Simplified block diagram of PCD04I 16 state turbo decoder.

where

$$A = 8B = 0.4C/\sigma^2. \quad (13)$$

Note that Ax_k represents the received value in quantised form. For example, the received quantised form could be from -31 to 31 , with the transmitted signal set points being $\{-24, -8, 8, 24\}$ implying that $A = 8$. Note that A does not have to be an integer. Knowing the normalised variance σ^2 and A we can calculate C to be

$$C = 2.5A\sigma^2. \quad (14)$$

For large values of Ax_k , R_k^d is approximately equal to $-Ax_k$. Since a transmitted signal can equal $3Ax_k$ with no noise, then R_k^d can vary from $3Ax_k$ to $-3Ax_k$.

The data log-likelihood ratios R_k^d should be input to R0I[5:0] and the parity log-likelihood ratios R_k^p should be input to either R1I[5:0] or R2I[5:0], depending on whether the parity information belongs to non-interleaved or interleaved data, respectively. If a parity input has been punctured then 0 should be input to either R1I[5:0] or R2I[5:0].

Example 2: Let $E_b/N_0 = 4.0$ dB and $A = 8$. This implies $\sigma^2 = 0.0995$ and $C = 2$ to the nearest integer.

For a fading channel where the amplitude A_k varies with time, we should substitute A with A_k in (11) and (12) and calculate C from

$$C = \frac{5\sigma^2}{K} \sum_{k=0}^{K/2-1} A_k \quad (15)$$

where K is the number of data bits.

Other Parameters

Figure 3 gives a block diagram of the PCD04I 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

$$f_d = \frac{F_d}{(NI + 1)(1 + (L + M)/K)} \quad (16)$$

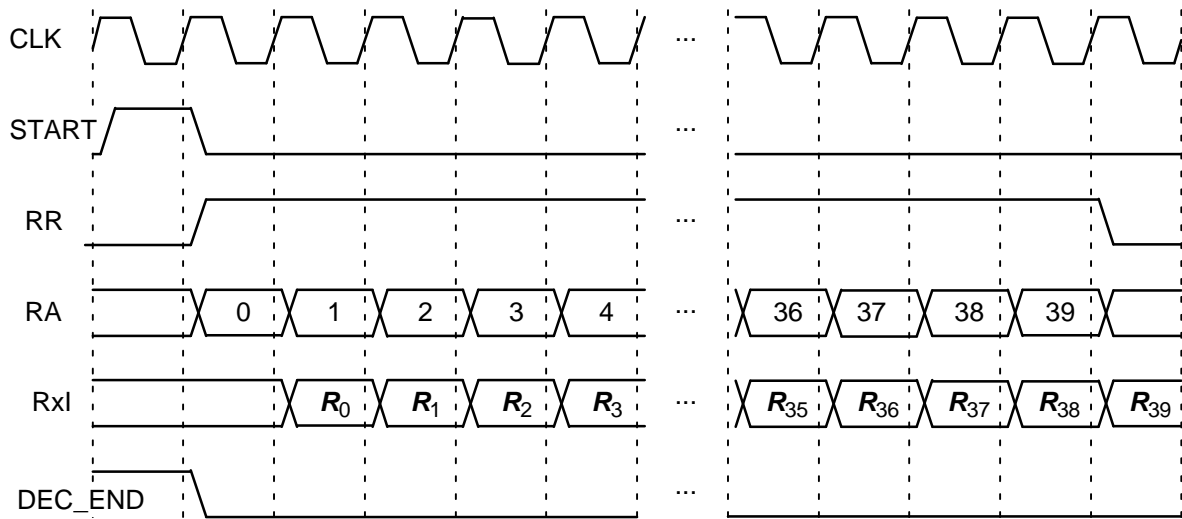
where F_d is the CLK frequency and L is the MAP decoder delay in bits (equal to either 138, 266, or 522), $M = 4$ for log-MAP and $M = 5$ for max-log-MAP decoding. The three delays indicate the sliding block length used in the MAP decoder, either 32, 64, or 128, respectively. For short block lengths $L = 138$ should be used to increase decoder speed, while $L = 266$ should be used for larger block sizes to increase performance. For high rate punctured turbo codes, $L = 522$ should be used. This parameter can be selected with the SLD input.

For example, if $F_d = 100$ MHz, $I = 5$ ($NI = 9$) and $M = 4$ (log-MAP decoding), the decoder speed ranges from 2.1 Mbit/s for $K = 40$ and $L = 138$ to 9.4 Mbit/s for $K = 5116$ and $L = 266$.

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 $SCLZ/32$. 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

Figure 4: Turbo Decoder Input Timing ($K = 36$).

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 N). 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 5. The received data ready signal RR goes high to indicate the data to be read from the address given by RA[14:0]. 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 and Y2 for the first encoder and X', Y1' and Y2' for the second encoder. The code polynomials are $g^0(D) = 1+D^3+D^4$ (23 in octal), $g^1(D) = 1+D+D^2+D^4$ (35) and $g^2(D) = 1+D+D^2+D^3+D^4$ (37). For rate 1/2 and 1/4 the data and tail are punctured, which is why two entries are shown.

The decoder then iteratively decodes the received data for $N/1$ 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 illus-

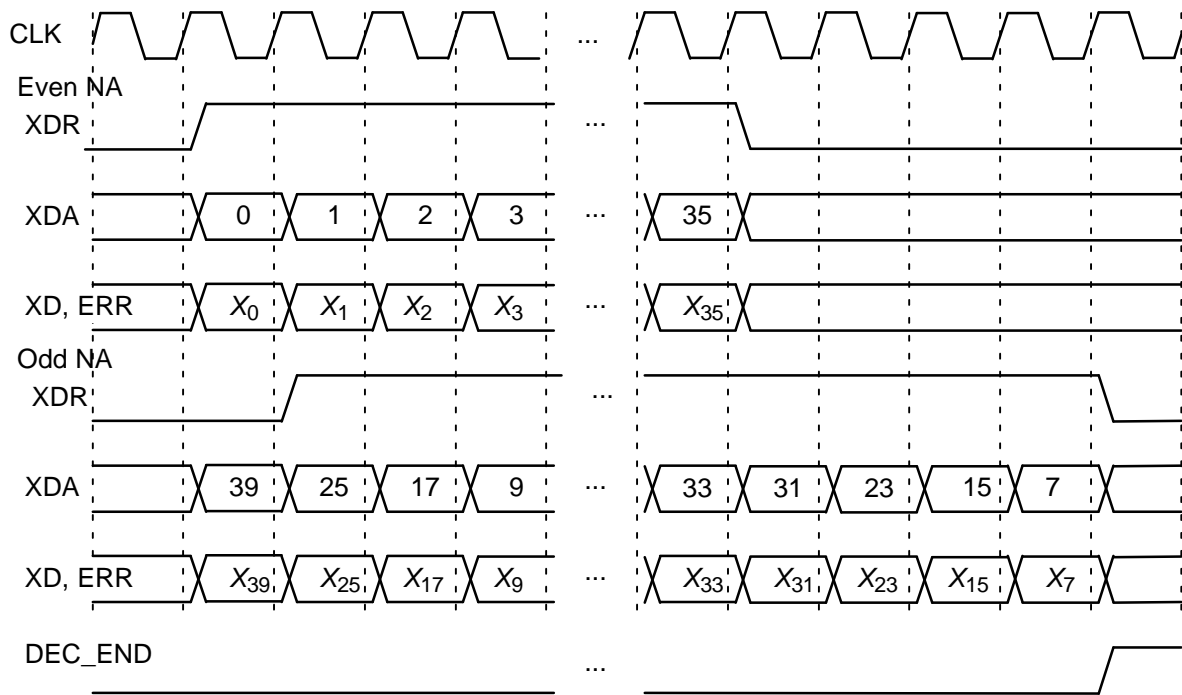
trates the decoder timing where the data is input on the first half iteration.

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 X
	R1I	Y1 Y1
	R2I	Y2 Y1'
	R3I	Y2' Y2'
1/5	R0I	X
	R1I	Y1
	R2I	Y2
	R3I	Y1'
	R4I	Y2'

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 $N/1$ is even, the block is output in sequential order. For $N/1$ odd, the block is output in interleaved order. To deinterleave the block, the output XDA[14:0] 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.

Figure 5: Turbo Decoder Output Timing ($K = 36$).

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 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 and 1/4, 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. Note that for even half iterations (odd NA), XDR only goes high when $XDA < K$. After startup, the maximum number of clock cycles for decoding is $(NI+1)(K+L+5)+1$.

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 is also output when XDE goes high. The outputs RA and RR are used to read the

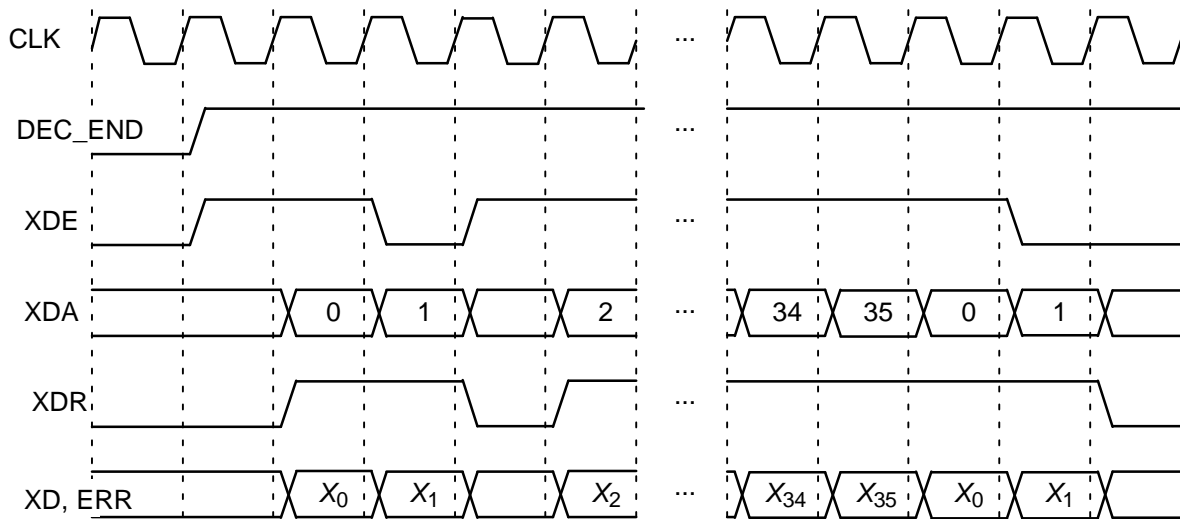
sign bit of R0I which is exclusive-ORed with XD to give ERR.

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 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.

Figure 6: XDE Timing ($K = 36$).

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 even) $Z0O$ ($L0O$) is not interleaved. $ZR0$ 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 $5K \times 8$ memories. At the end of every decoding the memories will have stored the information for $R0I$, $R1I$ and $R2I$.

The interleaver address table needs to be stored in an external synchronous ROM. $IA[14:0]$ is the input address to the ROM with $I[14:0]$ being the output of the ROM. The signal IR indicates when IA is valid. Figure 8 shows the timing for the ROM.

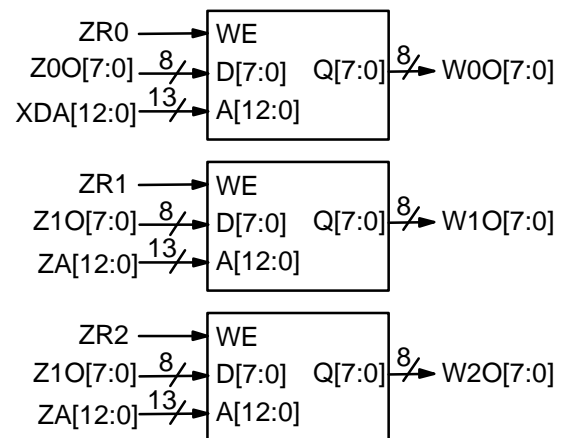


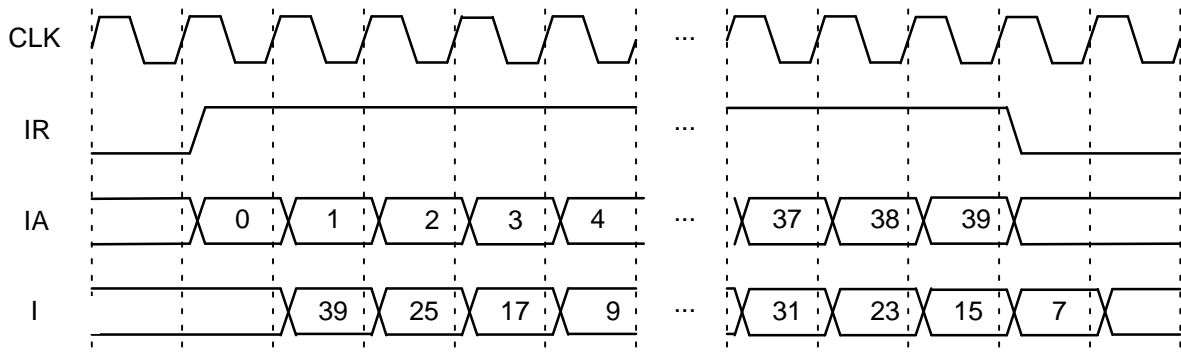
Figure 7: Output RAM for extrinsic information.

Simulation Software

Free software for simulating the PCD04I turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "pcd04isim request" in the subject header. The software uses an exact functional simulation of the PCD04I turbo decoder, including all quantisation and limiting effects.

After unzipping `pcd04isim.zip`, there should be `pcd04isim.exe` and `code.txt`. The file `code.txt` contains the parameters for running `pcd04isim`. These parameters are

m	Constituent code (CC) memory (2 to 4)
nt	Number of turbo code outputs (2 to 5)
g0	Divisor polynomial of CC in octal notation
g1	1st numerator polynomial of CC
g2	2nd numerator polynomial of CC
EbNomin	Minimum E_b/N_0 (in dB)

Figure 8: Interleaver address timing ($K = 36$).

<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>Pfmin</code>	Minimum frame error rate (FER)
<code>Pbmin</code>	Minimum bit error rate (BER)
<code>NI</code>	Number of half iterations-1 (0 to 255)
<code>SLD</code>	MAP decoder delay select (0 to 2)
<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 4)
<code>ZTH</code>	Extrinsic info. threshold (0 to 127)
<code>SI</code>	Select interleaver (0 or 1)
<code>K</code>	Block length
<code>q</code>	Number of quantisation bits (1 to 6)
<code>LOGMAP</code>	Log-MAP decoding (MODE[0], 0 or 1)
<code>C4PIN</code>	Use five-bit C (MODE[1], 0 or 1)
<code>enter_C</code>	Enter external C (y or n)
<code>C</code>	External C (0 to 17)
<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

Note that g_0 , g_1 and g_2 are given in octal notation, e.g., $g_0 = 23 \equiv 10011_2 \equiv 1 + D^3 + D^4$. For the Inmarsat standard, $m = 4$, $nt = 2$, $g_0 = 23$ and $g_1 = 35$ (g_2 is not used). 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 for BPSK or QPSK is

$$A = \frac{\text{optC}(2^{q-1} - 1)}{\text{mag}(\sigma)} \quad (17)$$

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

$$\text{mag}(\sigma) = \sigma \sqrt{\frac{2}{\pi}} \exp\left(\frac{-1}{2\sigma^2}\right) + 1 - 2Q\left(\frac{1}{\sigma}\right) \quad (18)$$

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

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) dt. \quad (19)$$

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 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 .

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, an internal 3GPP2 interleaver is used. This interleaver is valid from $K = 17$ to 32,764. For `SI` = 1, an external interleaver is input from directory `in_dir`. The file is called $K.\text{dat}$, where K is the data length, e.g., $36.\text{dat}$. The interleaver length is $K+4$. The interleaver file must be

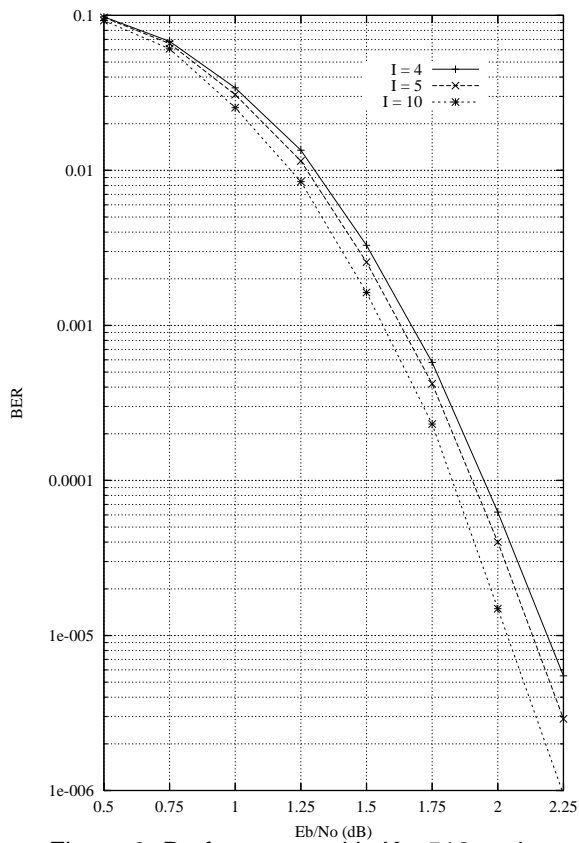


Figure 9: Performance with $K = 512$ and auto-stopping ($ZTH = 23$).

an ASCII file, with an interleaver address in each line, e.g.,

```
39
25
17
```

When the simulation is finished the output is given in, for example, file `k512.dat`, where $K = 512$. 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`), the average bit error rate (`Pb`) and the average frame error rate (`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 9. Auto-stopping was used with up to 10 iterations. To set up the simulation we let `optC = 0.35`, `enter_C = n` and adjusted E_b/N_0 to 1.0 dB to give `Pb = 0.0337`. We then adjusted `optC` to 0.41 which gave the lowest `Pb = 0.0241`, `C = 5` and `A = 11.36`. The values of `optC = 0.41` and `C = 5` are then used in the simulation.

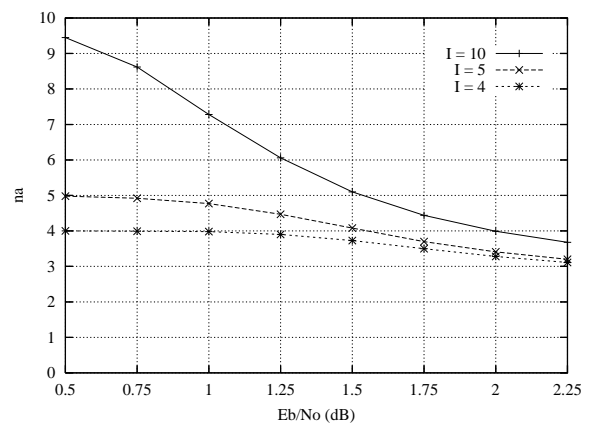


Figure 10: Average number of iterations with $K = 512$ and auto-stopping ($ZTH = 23$).

```
{m nt g0 g1 g2}
4 2 23 35 37
{EbNomin EbNomax EbNoinc optC}
0.50 2.50 0.25 0.41
{ferrmax Pfmin Pbmin}
256 1e-99 1e-5
{NI SLD LIMZ SCLZ M ZTH SI}
19 1 96 32 3 23 0
{K q LOGMAP C4PIN enter_C C}
512 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
```

When iterating is stopped early, the `nasum` ($2 \times \text{num} \times \text{na}$), `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 10 shows the average number of iterations with E_b/N_0 .

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., $X_{512}.dat$ 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., $Y_{512}.dat$ 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

$$R[i,j] = A \cdot (1 - 2 \cdot Y[i,j] + N[i,j])$$

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

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 6: Convolutional Codes.

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

The output ERR is the exclusive-OR of the sign bit of ROI with the corresponding re-encoded decoded output bit. 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

$$f_d = \frac{F_d}{N_c(1 + D/K) + 1/K} \quad (20)$$

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 $K = 504$, $D = 136$ (SM = 1, DELAY = 1), $N_c = 34$ (SM = 1) and $F_d = 100$ MHz, decoding speed is 2.3 Mbit/s.

Ordering Information

SW-PCD04I-SOS (SignOnce Site License)
 SW-PCD04I-SOP (SignOnce Project License)
 SW-PCD04I-VHD (VHDL ASIC License)

All licenses include Xilinx EDIF and VHDL cores. The EDIF core can be used for Virtex-2, Spartan-3 and Virtex-4 with Foundation or ISE software. The VHDL core can be used for Virtex-5, Spartan-6, Virtex-6, 7-Series, UltraScale and UltraScale+ with ISE or Vivado software.

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.

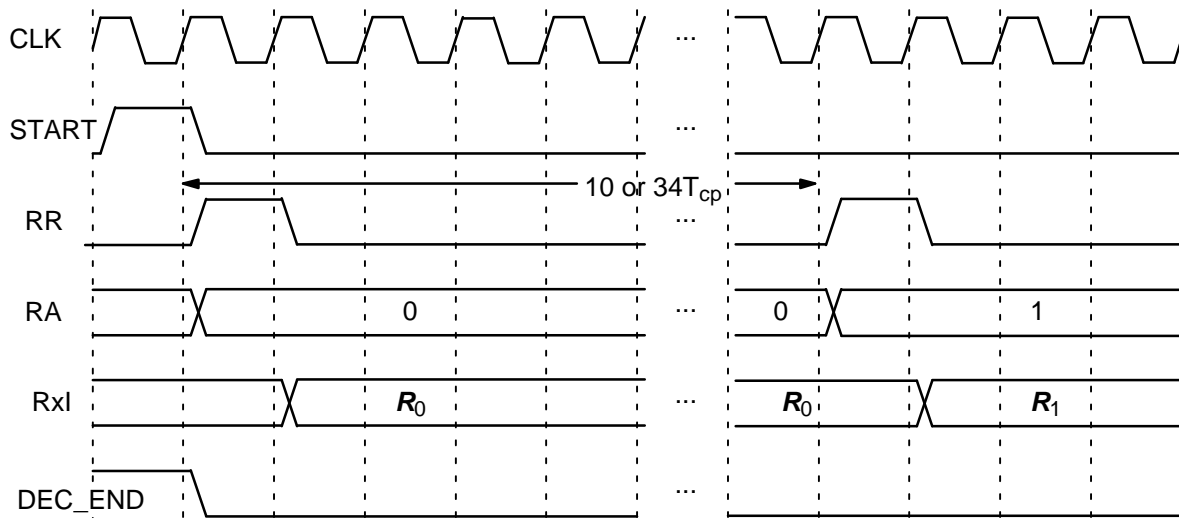


Figure 11: Viterbi Decoder Input Timing.

References

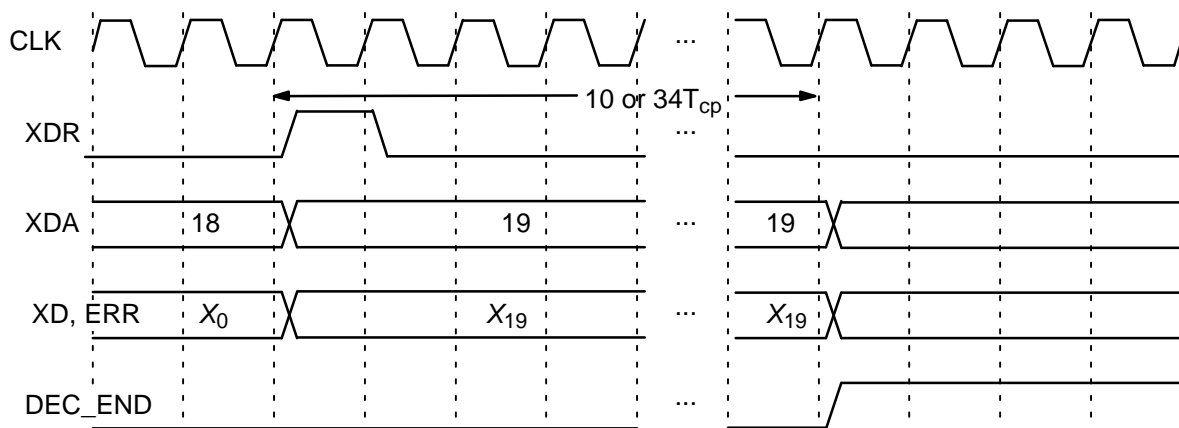
- [1] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 284–287, Mar. 1974.
- [2] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating 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.

© 2003–2022 *Small World Communications*. All Rights Reserved. Xilinx, Spartan, Virtex, Artix, Kintex, Zynq, 7-Series, UltraScale, UltraScale+ and all XC-prefix product designations are trademarks of AMD Xilinx, Inc. All other trademarks are the property of their respective owners.

Small World Communications, 6 First Avenue, Payneham South SA 5070, Australia.

Figure 12: Viterbi Decoder Output Timing ($K = 20$).

info@sworld.com.au ph. +61 8 8332 0319
http://www.sworld.com.au fax +61 8 7117 1416

Revision History

- v0.1 28 Feb. 2003. Preliminary product specification.
- v1.0 21 Mar. 2003. First official release. Deleted simulation curves with $K = 200, 500$ and 1000 . Updated simulation curves with $K = 600$.
- v1.12 12 June 2003. Added L0O and L1O outputs.
- v1.2 13 Aug. 2003. Updated performance, resources and simulation curves.
- v1.31 18 Jan 2005. Added Spartan-3 performance and resources used. Updated Virtex-E and Virtex-II performance and resources used. Added 16QAM demapping description.
- v1.32 26 May 2005. Added Virtex-II Pro and Virtex-4 performance.
- v1.33 10 June 2005. Added file name length received data format for simulation.
- v1.34 22 June 2005. Changed 5K and 21K interleaver sizes to 6K and 22K. Deleted Virtex-E and Virtex-II performance. Deleted Virtex-E resources used.
- v1.35 2 Nov. 2007. Changed SLD input to SLD[1:0]. Updated resources used.
- v1.36 15 Oct. 2010. Deleted Virtex-II Pro performance and resources used. Added Virtex-5, Virtex-6 and Spartan-6 performance. Updated Virtex-4 resources used. Added Virtex-5 resources used.
- v1.37 21 Oct. 2010. Updated resources used.
- v1.38 25 June 2014. Deleted university license. Deleted Spartan-3 performance. Added Artix-7, Kintex-7 and Zync performance.
- v1.39 11 Jan. 2018. Updated performance and resources used. Deleted Spartan-6 and Virtex-4 performance and resources used. Reduced startup delay by one clock cycle. Added one clock cycle per half iteration for max-log-MAP decoding. Rearranged schematic symbol outputs. Added revision history prior to v1.37. Deleted Xilinx EDIF and VHDL simulation cores. Added Xilinx VHDL core. Updated simulation performance. Changed 20K interleaver option to 12K.
- v1.40 26 Aug. 2022. Added MODE[7] input. Updated decoder performance and complexity. Deleted Virtex-5, Virtex-6 and Zync performance. Added Spartan-7, Kintex Ultrascale and Kintex Ultrascale+ performance.
- v1.41 8 Nov. 2022. Corrected code rate R and decoder delay for SLD.