PCD03D Features

Turbo Decoder

- 8 state DVB–RCS and IEEE 802.16 WiMAX compatible
- Rate 1/3, 2/5, 1/2, 2/3, 3/4, 4/5, 5/6, 6/7, 7/8
- Automatic depuncturing
- 48 or 96 to 2048 or 5120 bit data length
- Up to 197 MHz internal clock
- Up to 35 Mbit/s with 5 decoder iterations
- 6–bit signed magnitude input data
- Optional log–MAP or max–log–MAP constituent decoder algorithms
- Up to 128 iterations in 1/2 iteration steps.
- Optional power efficient early stopping
- Optional extrinsic information scaling and limiting
- Estimated channel error output
- Decoded symbol log probability output
- DVB–RCS or IEEE 802.16 WiMAX implementation options with optional interleaver parameters
- Free simulation software

Viterbi Decoder (Optional)

- 64 state (constraint length 7)
- Rate 1/2, 2/3, 3/4, 5/6, 7/8
- Automatic depuncturing
- Data length from 2 to 2042 or 4090 bits with tail termination
- Optional tail biting decoding from 22 or 43 to 511 data bits
- Up to 19 Mbit/s
- 6–bit signed magnitude input data
- Estimated channel error output

- Available as EDIF core and VHDL simulation core for Xilinx FPGAs under SignOnce IP License. Actel, Altera and Lattice FPGA cores available on request.
- Available as VHDL core for ASICs

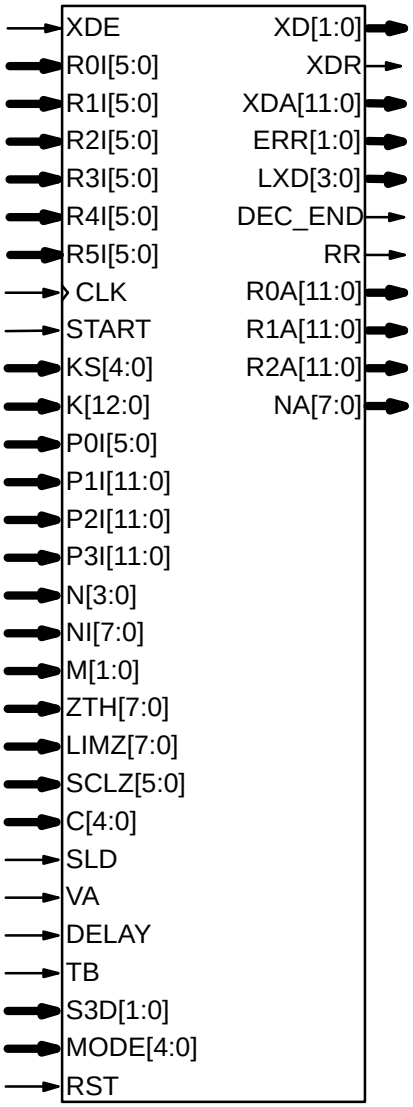

Figure 1: PCD03D schematic symbol.

Introduction

The PCD03D is a fully compatible DVB–RCS [1] and IEEE 802.16 WiMAX [2] error control decoder. Both codes use an eight state rate 2/4 systematic recursive convolutional tail–biting constituent code. Since a tail–biting code is used, there are no tail bits, increasing the bandwidth efficiency of the code. For DVB–RCS, block sizes from 96 to 1728 are used. For IEEE 802.16, block sizes range from 48 to 4800 bits.

The MAP03D MAP decoder core is used with the PCD03D core to iteratively decode the DVB–RCS or IEEE 802.16 WiMAX turbo code. The Log–MAP algorithm for maximum performance or the max–log–MAP algorithm for minimum complexity and maximum speed can be selected. The sliding block algorithm is used with sliding block lengths of L = 32 or 64. To reduce MAP decoder delay by approximately one half, the data is input

and output in reversed blocks of *L*. This results in increased decoder speed, especially for small block sizes. 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 interleavers for the standards are similar, using linear congruential equations that depend on four parameters, P0I to P3I. With IEEE 802.16 WiMAX, there are two different interleavers, depending on the system used; WirelessMAN–OFDM and WirelessMAN–OFDMA.

The VA08V Viterbi decoder core is used with the PCD03D core to decode the DVB–RCS or WiMAX convolutional code with or without tail–biting. The decoder shares its traceback memory with the internal interleaver memory of the turbo decoder, minimising complexity. Minimum traceback lengths of 64 or 128 bits can be selected. 6–bit quantisation is used.

The turbo decoder can achieve up to 35.9 Mbit/s with 5 iterations using an 197 MHz internal clock (*K* = 4800). Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance. The Viterbi decoder can achieve 19.2 Mbit/s with 64 states (*K* = 4800).

Figure 1 shows the schematic symbol for the PCD03D decoder. The EDIF core can be used with Xilinx Integrated Software Environment (ISE) software to implement the core in Xilinx FPGA's. The VHDL core can be used in ASIC designs.

Table 1 shows the performance achieved with various Xilinx parts. T_{cp} is the minimum clock period over recommended operating conditions. These performance figures may change due to device utilisation and configuration.

Table 2 shows the resources used for Virtex–4 and Virtex–5 devices. Virtex–2 and Spartan–3 devices have a similar complexity to Virtex–4. Virtex–6, Spartan–6, Virtex–7 and 7–Series devices have a similar complexity to Virtex–5. The MODE[4:0] 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. The RAMs refer to 18KB BlockRAMs.

Table 1: Performance of Xilinx parts.

Xilinx Part	T_{cp} (ns)	Turbo* Mbit/s	Viterbi Mbit/s
XC6SLX25–2	10.223	17.8	9.5
XC6SLX25–3	9.105	20.0	10.7
XC4VLX15–10	11.250	16.2	8.6
XC4VLX15–11	9.588	19.0	10.1
XC4VLX15–12	8.522	21.4	11.4
XC5VLX30–1	7.954	22.9	12.2
XC5VLX30–2	6.847	26.6	14.2
XC5VLX30–3	6.137	29.7	15.9
XC6VLX75T–1	6.904	26.4	14.1
XC6VLX75T–2	5.977	30.5	16.3
XC6VLX75T–3	5.425	33.6	17.9
XC7A100T–1	9.388	19.4	10.3
XC7A100T–2	7.736	23.5	12.5
XC7A100T–3	6.862	26.5	14.1
XC7K70T–1	6.798	26.8	14.3
XC7K70T–2	5.860	31.1	16.6
XC7K70T–3	5.076	35.9	19.2

*max–log–MAP, 5 iterations, *K* = 4800, *L* = 64

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 134 (TB = 0)
- 1 = delay 262 (TB = 0)
- 0 = delay 198 (TB = 1)
- 1 = delay 390 (TB = 1)

ERR Estimated Error

K Data Length (used when KS = 0)
- Turbo Decoder (VA = 0)
- Minimum Data Length:
- 48 (SLD = 0)
- 96 (SLD = 1)
- Maximum Data Length:
- 2048 (MODE4 = 0)
- 5120 (MODE4 = 1)
- Viterbi Decoder (VA = 1)
- Terminated Input (TB = 0)
- Minimum Data Length: 2
- Maximum Data Length:
- 2042 (MODE4 = 0)
- 4090 (MODE4 = 1)
- Tail Biting Input (TB = 1)
- Minimum Data Length:
- 22 (DELAY = 0)
- 43 (DELAY = 1)
- Maximum Data Length: 511

KS Data Length Select
- 0 = select K, P0I–P3I

Table 2: Resources used.

Configuration	Turbo Rates	Viterbi Rates	Virtex–4 LUTs	Virtex–5 LUTs	Block RAMs
DVB–RCS Turbo (max–log–MAP)	1/3–7/8	–	6558	5986	2
DVB–RCS Turbo (max–log–MAP) and Viterbi	1/3–7/8	1/2–7/8	7699	6905	2
IEEE 802.16 WiMAX Turbo (max–log–MAP)	1/3–7/8	–	6733	6147	4
IEEE 802.16 WiMAX Turbo (small log–MAP)	1/3–7/8	–	8775	7608	4
IEEE 802.16 WiMAX Turbo (large log–MAP)	1/3–7/8	–	9930	7930	4

S3D = 0 (DVB–RCS)
1 = length 96 (12 bytes)
2 = length 128 (16 bytes)
3 = length 424 (53 bytes)
4 = length 440 (55 bytes)
5 = length 456 (57 bytes)
6 = length 848 (106 bytes)
7 = length 864 (108 bytes)
8 = length 880 (110 bytes)
9 = length 1696 (212 bytes)
10 = length 1712 (214 bytes)
11 = length 1728 (216 bytes)
12 = length 1504 (188 bytes)
S3D = 3 (WirelessMAN–OFDMA)
1 = length 48 (6 bytes)
2 = length 96 (12 bytes)
3 = length 144 (18 bytes)
4 = length 192 (24 bytes)
5 = length 240 (30 bytes)
6 = length 288 (36 bytes)
7 = length 384 (48 bytes)
8 = length 432 (54 bytes)
9 = length 480 (60 bytes)
10 = length 72 (9 bytes)
11 = length 216 (27 bytes)
12 = length 360 (45 bytes)
13 = length 960 (120 bytes)
14 = length 1920 (240 bytes)
15 = length 2880 (360 bytes)
16 = length 3840 (480 bytes)
17 = length 4800 (600 bytes)

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)

LIMZ Extrinsic Information Limit (1–193)

LXD Decoded symbol log probability (0–14)

N Code Rate
0 = rate 1/2
1 = rate 2/3
2 = rate 3/4
3 = rate 5/6
4 = rate 7/8
5 = rate 1/3 (turbo only)
6 = rate 2/5 (turbo only)
7 = rate 4/5 (turbo only)
8 = rate 6/7 (turbo only)

NA Half Iteration Number (0–255)

NI Number of Half Iterations (0–255)
NI = 2I–1 where I is number of iterations

P0I–P3I Interleaver parameters (used when KS = 0)

R0A Received Data (AB) Address

R1A Received Parity (Y_1Y_2) Address

R2A Received Parity (W_1W_2) Address

R0I Received Data (A)

R1I Received Data (B)

R2I Received Parity (Y_1)

R3I Received Parity (Y_2)

R4I Received Parity (W_1)

R5I Received Parity (W_2)

RR Received Data Ready

RST Synchronous Reset

S3D Code Select
0 = DVB–RCS
2 = IEEE 802.16 WirelessMAN–OFDM
3 = IEEE 802.16 WirelessMAN–OFDMA

SCLZ Extrinsic Information Scale (1–32)

SLD MAP decoder sliding window length
0 = length 32
1 = length 64

START Decoder Start

TB Tail Bite Select for Viterbi decoder
0 = Teminated with 6 symbol tail
1 = Tail Bite decoding

VA Viterbi Decoder Select
0 = turbo decoder
1 = Viterbi decoder

XD Decoded Data

XDA Decoded Data Address

XDE Decoded Data Enable

XDR Decoded Data Ready

ZTH Early Stopping Threshold (1–255)

Table 3 describes each of the MODE[4:0] inputs that are used to select various decoder implementations. Note that MODE[4:0] 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.

Turbo Decoder Parameters

For optimal performance, the maximum a posteriori (MAP) [3] 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 [4]) algorithm can provide up to 0.5 dB coding gain at low SNRs. Log–MAP operation is enabled when MODE0 is high.

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 to 7/8 1 = rate 1/3 to 7/8
MODE3	0 = turbo decoder 1 = turbo and Viterbi decoder
MODE4	0 = 1K Interleaver (DVB–RCS) 1 = 2.5K Int. (WiMAX)

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

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

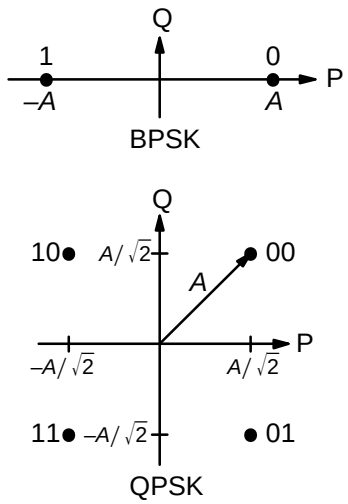

Figure 2: BPSK and QPSK signal sets.

E_b/N_0 is the energy per bit to single sided noise density ratio and $R = k/n$ 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 [4] 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.

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 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).

Table 4: Quantisation for R0I, R1I and R2I.

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

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.

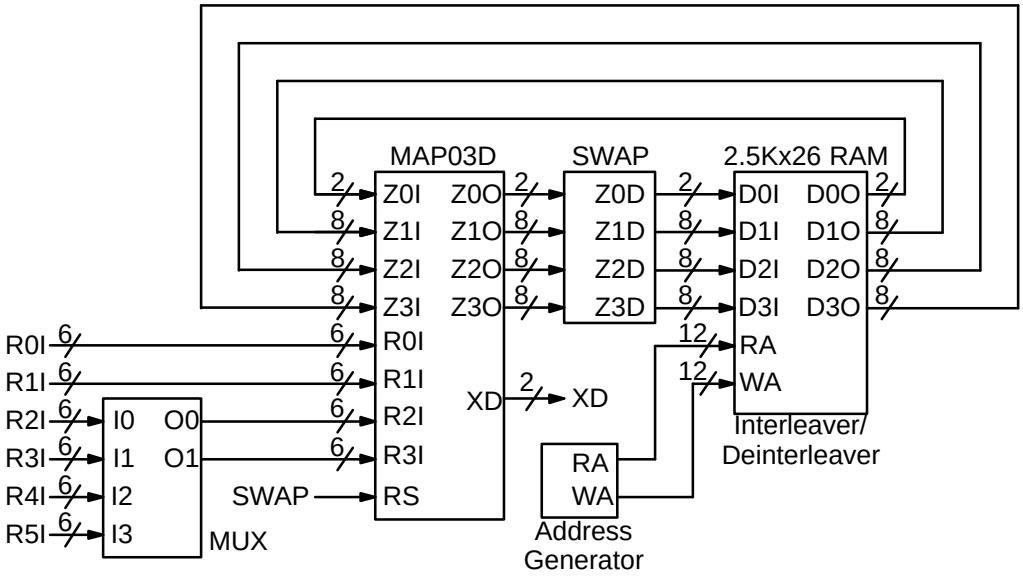

Figure 3: Simplified block diagram of PCD03D eight state turbo decoder.

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 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 PCD03D eight state turbo decoder. The extrinsic information output for each of the four data symbols is given by Z0O to Z3O from the MAP decoder. To reduce the number of bits, Z0O indicates the symbol with the smallest extrinsic information, with Z1O to Z3O giving the extrinsic information of the other three symbols, minus the extrinsic information of the symbol with the smallest extrinsic information. This should give increased performance compared to using the extrinsic information for each of the two decoded bits, which entails a loss of information.

The symbol SWAP is used to swap the symbols every other symbol when interleaving (or deinterleaving). This is as per the DVB–RCS and IEEE 802.16 WiMAX standard.

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

$$f_d = \frac{F_d K}{(NI + 1)(L(\lceil K/(2L) \rceil + 3) + 10) + 1} \quad (3)$$

where F_d is the CLK frequency and L is the MAP decoder sliding window length, either 32 or 64. For short block lengths, L = 32 should be used to increase decoder speed, while L = 64 should be used for larger block sizes and high code rates to increase performance. This parameter can be selected with the SLD input.

Note that the smallest value of K that can be used is 48 with L = 32 and 96 with L = 64. The largest value of K that can be used is 2048 with MODE4 = 0 and 5120 with MODE4 = 1.

The term $L\lceil K/(2L) \rceil$ is the length of data input to the MAP decoder, excluding L starting and L ending symbols. The rounding is due to using reversed blocks of L data. The remaining terms are L data used to initialise the state metrics in the MAP decoder, a MAP decoder delay of $2L$+9 and one clock cycle from reading extrinsic information from the synchronous interleaver RAM.

For example, if F_d = 100 MHz and I = 5 (NI = 9) the decoder speed ranges from 3.4 Mbit/s for K = 48 and L = 32 to 18.2 Mbit/s for K = 4800 and L = 64.

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 pre-

vent the decoder from making decisions too early, which improves decoder performance.

The limit factor LIMZ should vary between 1 and 193. Values greater than 193 could cause performance degradation due to limiting of branch metric values in the MAP decoder.

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 = 32. For max–log–MAP decoding we recommend SCLZ = 22. 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.

Interleaver parameters

The interleaving equations are of the form $i = (P_0 j + 1 + Q(j \bmod 4)) \bmod K/2$ where j varies from 0 to $K/2–1$. Table 5 gives the formulas for $Q(j)$ for the four interleaver types.

Table 5: Interleaver Parameters

j	*Q(j)*		
	DVB–RCS	**Wireless MAN–OFDM**	**Wireless MAN–OFDMA**
0	0	0	0
1	$K/4+P_1$	0	$K/4+P_1$
2	P_2	$K/8$	P_2
3	$K/4+P_3$	$K/4+P_1$	$K/4+P_3$

For DVB–RCS and WirelessMAN–OFDMA, the same interleaver is used, although the parameters P_0 to P_3 used for each scheme are different. Also, WirelessMAN–OFDMA has a longer maximum data length (4800 bits), compared to 1728 bits for DVB–RCS.

The parameters P_0 to P_3 depend on the block length K and which of the four schemes are used. These values are given in the two standards. P_0 is a prime number while P_1 to P_3 are even numbers.

When KS[4:0] = 0, the data length K is input to K[12:0] and the interleaver parameters P_0 to P_3 are input to P0I[5:0], and P1I[11:0] to P3I[11:0], respectively.

When KS[4:0] > 0, the internal data length selected by KS is used. Also, the internal interleaver parameters P_0 to P_3 for the data length and standard (either DVB–RCS or WirelessMAN–OFDMA) are used. The inputs K[12:0], P0I[5:0], and P1I[11:0] to P3I[11:0] are ignored.

For WirelessMAN–OFDM, $P_1 = 3K/8$, thus $(K/4+P_1) \bmod K/2 = K/8$. There are only two parameters K and P_0, which are input to K[12:0] and P0I[5:0]. The parameter inputs P1I[11:0] to P3I[11:0] are ignored. Note that if S3D = 2, the value of KS is also ignored. That is, K and P0I have to be externally input.

For WirelessMAN–OFDM, K depends on the number of subchannels and modulation used (QPSK, 16QAM, or 64QAM). The parameter P_0 depends on the modulation and the code rate (1/2, 2/3, or 3/4) used.

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 two or three synchronous read RAMs of size $K/2$x6n, n = 4 or 6.

The received data for A, B, Y_1, Y_2, W_1 and W_2 are input to R0I to R5I, respectively. Three output read addresses are used to read data from the input RAMs. Address R0A corresponds to R0I and R1I, address R1A corresponds to R2I and R3I and address R2A corresponds to R4I and R5I.

R0I and R1I require a separate address as the data is read in non–interleaved and interleaved order with each iteration. R1A and R2A are read in non–interleaved order. Normally, R1A and R2A are the same, except when internal depuncturing is used with rate 2/5. This rate is only used with DVB–RCS. Thus, R2I to R5I can use R1A for IEEE 802.16 WiMAX. Figure 4 shows how to interface the input RAMs to the decoder with IEEE 802.16 WiMAX.

The received data ready signal RR goes high to indicated the data to be read from the address given by R0A to R2A. Tables 6 and 7 illustrates the data length L_d for DVB–RCS and WirelessMAN–OFDM, respectively, stored for address 0 to $L_d–1$ for each of the received data. That is, there is no need to depuncture the data. The data can be stored as it is received from address 0 to $L_d–1$.

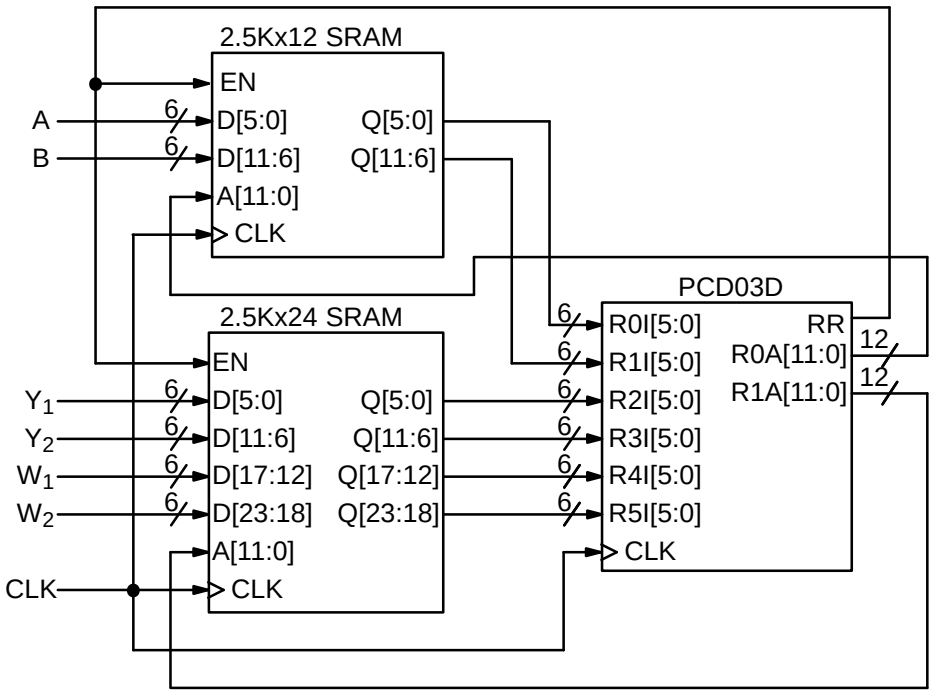

Figure 4: Interfacing Input RAMs to PCD03D for IEEE 802.16 WiMAX.

The decoder reads out the data at the correct time and automatically performs depuncturing. Note that automatic depuncturing can not be used with WirelessMAN–OFDMA, since puncturing occurs after channel interleaving.

Table 6: DVB–RCS Input data length

Data	Rate						
	1/3	2/5	1/2	2/3	3/4	4/5	6/7
R0I	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2
R1I	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2
R2I	*K*/2	*K*/2	*K*/2	*K*/4	*K*/6	*K*/8	*K*/12
R3I	*K*/2	*K*/2	*K*/2	*K*/4	*K*/6	*K*/8	*K*/12
R4I	*K*/2	*K*/4	0	0	0	0	0
R5I	*K*/2	*K*/4	0	0	0	0	0

Table 7: WirelessMAN–OFDM data length

Data	Rate					
	1/3	1/2	2/3	3/4	5/6	7/8
R0I	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2
R1I	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2	*K*/2
R2I	*K*/2	*K*/2	*K*/4	*K*/6	*K*/10	*K*/14
R3I	*K*/2	*K*/2	*K*/4	*K*/6	*K*/10	*K*/14
R4I	*K*/2	0	0	0	0	0
R5I	*K*/2	0	0	0	0	0

The code polynomials are $1+D+D^3$ (15 in octal) for the feedback branch, $1+D^2+D^3$ (13) for Y and $1+D^3$ (11) for W.

The decoder then iteratively decodes the received data for $NI+1$ half iterations, rereading the received data for each half iteration for $L\lceil K/(2L)\rceil + 2L$ clock cycles. The signal RR goes high for $L\lceil K/(2L)\rceil + 2L$ clock cycles while data is being output. Figure 5 illustrates the decoder timing with $K = 48$ and $L = 32$ (no puncturing) where the data is input on the first half iteration. Note that the first address read is 7 which equals $(L–1)$ mod $K/2$. The last address is 16 which equals $2L$ mod $K/2$.

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 $L\lceil K/(2L)\rceil$ clock 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. In both cases every sub–block of length L is time reversed. To output the block in the correct order, the output XDA[11:0] should be used as the write address to a buffer RAM. After the block has been written to the buffer RAM, the decoded block can be sequentially read from the buffer RAM.

The signal ERR is a channel error estimator output. It is the exclusive OR of XD and the sign bits of R0I and R1I appropriately delayed.

The DEC_END signal is low during decoding. At the end of decoding, DEC_END goes high. Figure 6 illustrates the decoder timing where data is

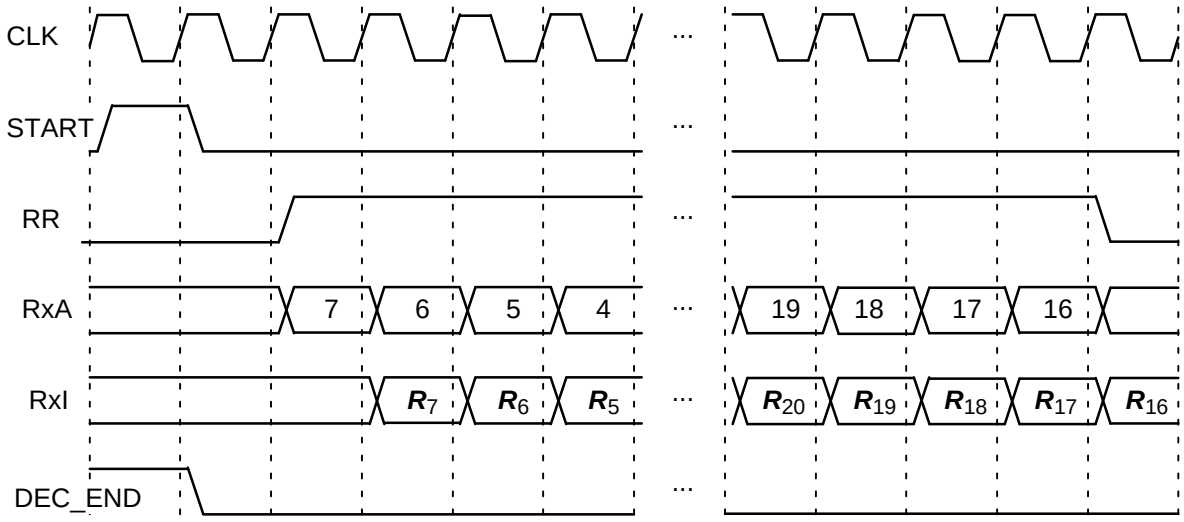

Figure 5: Turbo Decoder Input Timing (K = 48, L = 32, unpunctured).

output on the last half iteration. After startup, the maximum number of clock cycles for decoding is $(NI + 1)(L(\lceil K/(2L) \rceil + 3) + 10) + 1$.

During the last half iteration the decoded data XD, ERR and LXD are 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 7 shows the decoder timing when XDE is used.

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(SCLZ/32) was

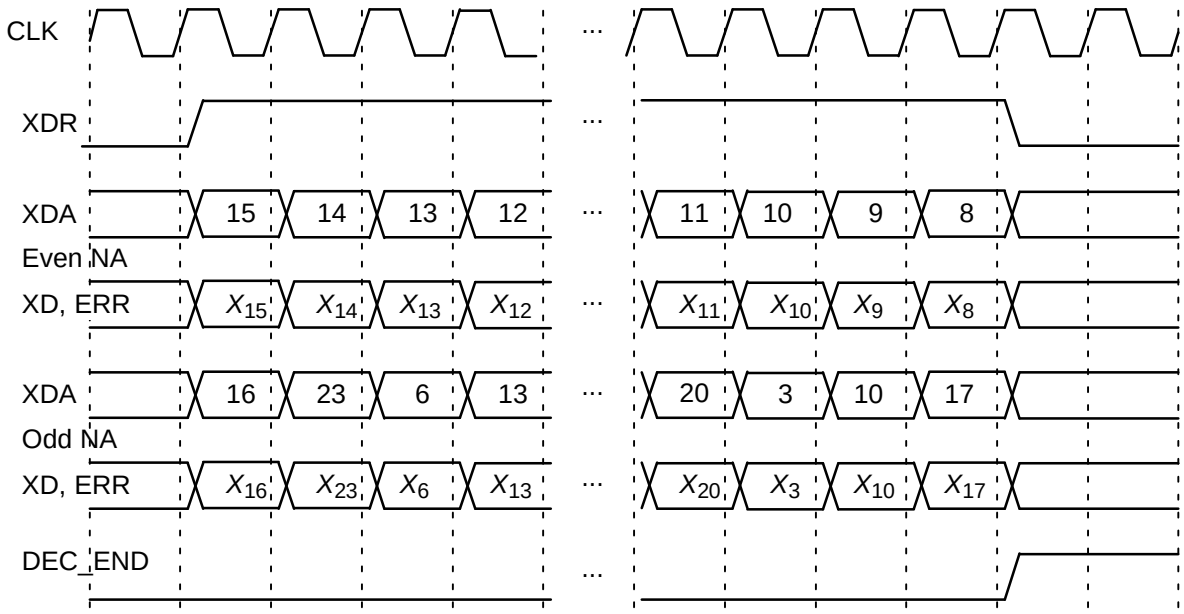

Figure 6: Turbo Decoder Output Timing (K = 48, L = 32, unpunctured, WirelessMAN–OFDMA).

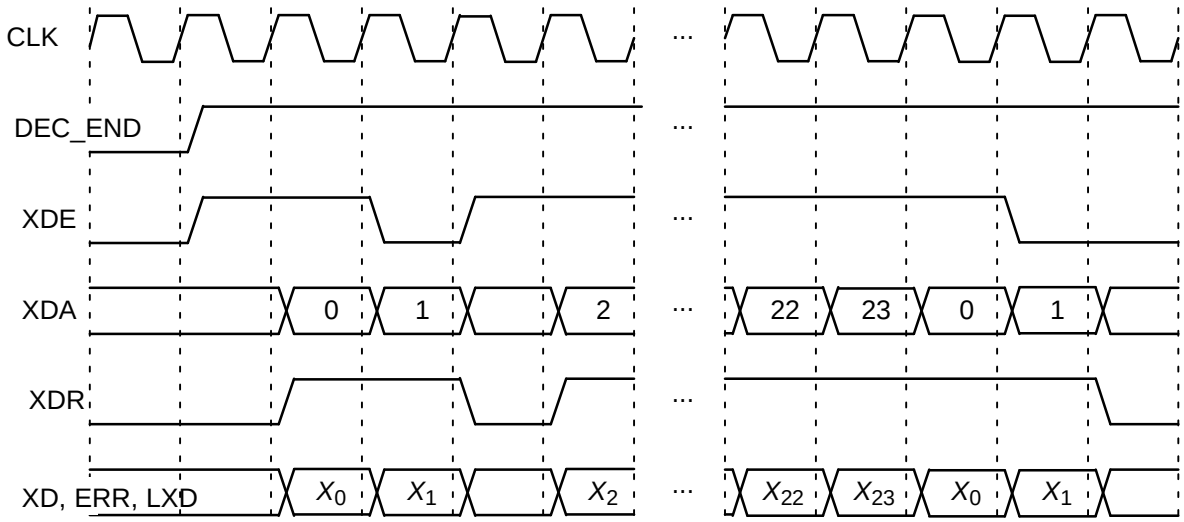

Figure 7: XDE Timing (K = 48).

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.

LXD Output

The output LXD is an estimate of the logarithm of the probability of the decoded symbol output. In the probability domain, this value ranges from 0.25 to 1 as there are four symbols. In the log domain, this corresponds to a range from 0 to 14.

The LXD output is formed by taking the min* operation (equivalent to the max* operation, but in the minus log domain) of the log–likelihood ratios of each of the four symbols, where the most likely symbol is used as the reference symbol. A fixed look–up table for the correction term in the min* operation is used to allow operation at high signal to noise ratio (otherwise, due to finite quantisation, all LXD values would equal zero).

The LXD output can be used to determine the reliability of a decoded output by summation of the $K/2$ LXD values. Thus, one can compare the reliability of the decoded data for different conditions, choosing the decoded data that has the highest summation of LXD values.

Due to the tail–biting and reversed blocks of L used in decoding, some output symbols are output twice, with the latter values being more reliable. Thus we recommend using the XDE input to output LXD after decoding is completed.

Simulation Software

Free software for simulating the PCD03D turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with “pcd03dsim request” in the subject header. The software uses an exact functional simulation of the PCD03D turbo decoder, including all quantisation and limiting effects.

After unzipping pcd03dsim.zip, there should be pcd03dsim.exe and code.txt. The file code.txt contains the parameters for running pcd03dsim. These parameters are

`m`	Constituent code (CC) memory (2 to 4)
`nt`	Number of turbo code outputs (2 to 8)
`kt`	Number of turbo code inputs (1 to 7)
`g0`	1st divisor polynomial of CC in octal
`g1`	2nd divisor polynomial of CC
`g2`	1st numerator polynomial of CC
`g3`	2nd numerator polynomial of CC
`EbNomin`	Minimum E_b/N_0 (in dB)
`EbNomax`	Maximum E_b/N_0 (in dB)
`EbNoinc`	E_b/N_0 increment (in dB)
`optC`	Input scaling parameter (normally 0.65)
`ferrmax`	Number of frame errors to count
`Pfmin`	Minimum frame error rate (FER)
`Pbmin`	Minimum bit error rate (BER)
`NI`	Number of half iterations–1 (0 to 255)
`SLD`	MAP decoder delay select (0 or 1)
`LIMZ`	Extrinsic information limit (1 to 193)
`SCLZ`	Extrinsic information scale (1 to 32)
`M`	Stopping mode (0 to 4)
`ZTH`	Extrinsic info. threshold (0 to 255)
`q`	Number of quantisation bits (3 to 6)
`LOGMAP`	Log–MAP decoding (MODE0, 0 or 1)
`C4PIN`	Use five–bit C (MODE1, 0 or 1)
`enter_C`	Enter external C (y or n)
`C`	External C (0 to 17)
`S3D`	Standard select (0, 2 or 3)
`KS`	DVB–RCS data length select (0 to 11)

K	Block length (96 to 1728 for DVB–RCS or 48 to 4800 for IEEE 802.16 WiMAX)
P0I	1st interleaver parameter (prime number)
P1I	2nd interleaver parameter (even number)
P2I	3rd interleaver parameter (even number)
P3I	4th interleaver parameter (even number)
state	State file (0 to 2)
s1	Seed 1 (1 to 2147483562)
s2	Seed 2 (1 to 2147483398)
read_x	Use external information data (y or n)
read_r	Use external received data (y or n)
out_dir	Output directory
in_dir	Input directory

Note that g0, g1, g2, and g3 are given in octal notation, e.g., g0 = 13 $\equiv$ 1011_2 $\equiv$ $1+D^2+D^3$. The polynomials are for the encoder in recursive systematic shift register form [5]. For the DVB–RCS and IEEE 802.16 WiMAX, the equivalent code is m = 3, g0 = 4, g1 = 6, g2 = 3 and g4 = 5. The turbo code rate is kt/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 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 low SNR, mag(σ) $\simeq$ $\sigma\sqrt{2/\pi}$ $\simeq$ 0.798σ. 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.

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 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. An external value of C can be input by setting enter_C to y.

Table 8 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 DVB–RCS, K = 424, 10 iterations (NI = 19), M = 3, ZTH = 23, LIMZ = 192 and log–MAP decoding. Using these parameters for higher E_b/N_0 values should result is very little performance degradation. For max–log–MAP, the same optC values were used as for log–MAP, with SCLZ equal to 23 for rates 1/3 and 2/5 and 22 for rates 1/2 to 6/7.

Table 8: Simulation parameters

R	E_b/N_0 (dB)	optC	A	C	SCLZ	BER 10^{-2}
1/3	0.40	0.46	11.36	8	28	2.75
2/5	0.50	0.47	12.20	7	32	3.05
1/2	0.75	0.50	13.74	6	32	3.22
2/3	1.60	0.52	15.29	3	27	2.68
3/4	2.00	0.52	15.58	3	32	2.92
4/5	2.25	0.52	15.71	3	32	2.77
5/6	2.50	0.58	17.62	3	32	2.46
6/7	2.60	0.58	17.67	3	32	2.97

When the simulation is finished the output is given in, for example, file k424.dat, where K = 424. 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, the number of iterations, 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 8, which also shows the performance for other code rates and max–log–MAP. Auto–stopping was used with up to 10 iterations. When iterating is stopped early, the nasum (2*num*na), berr and ferr results at stopping are copied for each half iteration to the

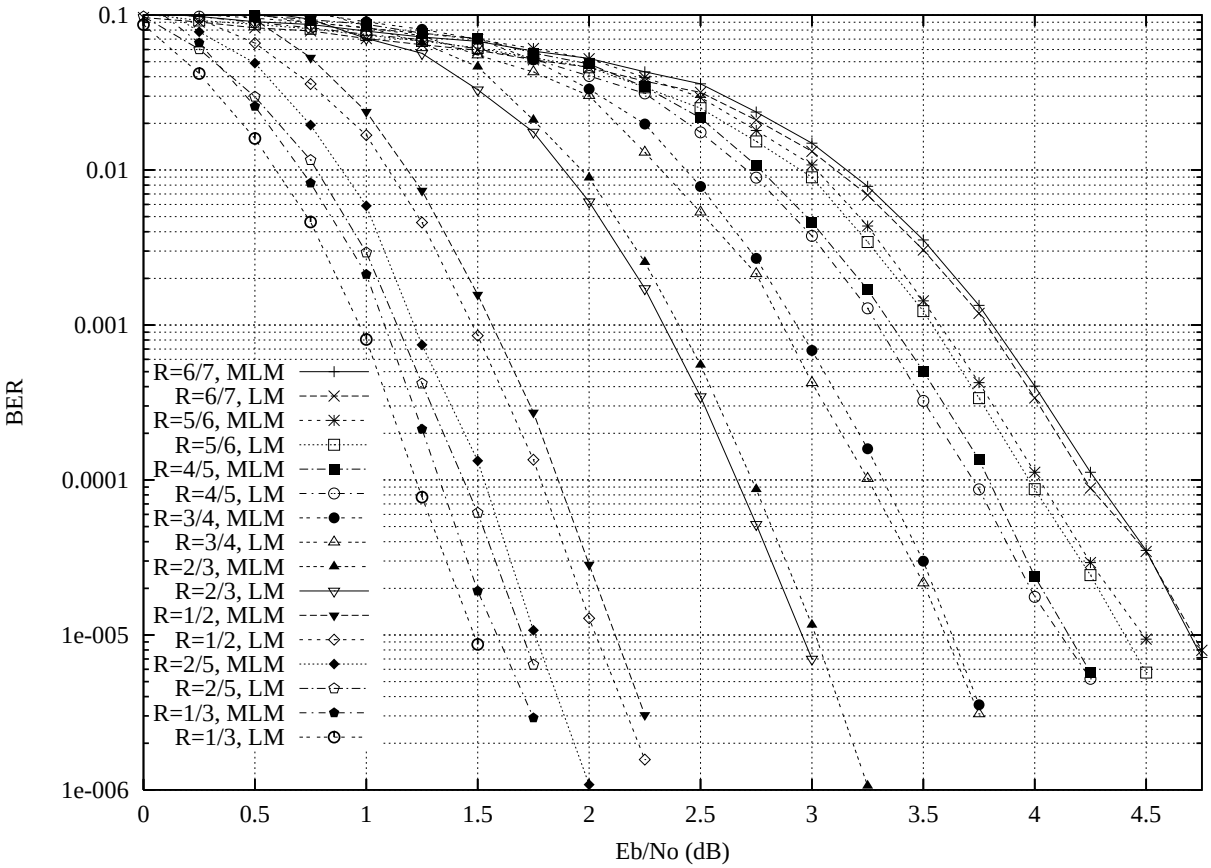

Figure 8: Performance with up to 10 iterations, block size 424, DVB–RCS and auto–stopping.

maximum iteration number. Figure 9 shows the average number of iterations with E_b/N_0.

```
{m nt kt g0 g1 g2 g3}
 3  2  1  4  6  3  5
{EbNomin EbNomax EbNoinc optC}
 0.00    2.00    0.25    0.5
{ferrmax Pfmin Pbmin}
 64      1e-99 1e-5
{NI SLD LIMZ SCLZ M ZTH}
 19 1   192  32   3 23
{q LOGMAP C4PIN  enter_C  C}
 6 1      1      n        6
{S3D KS K   P0I P1I P2I P3I}
 0   3  424 13  106 108 2
{state s1    s2}
 0     12345 67890
{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. 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.

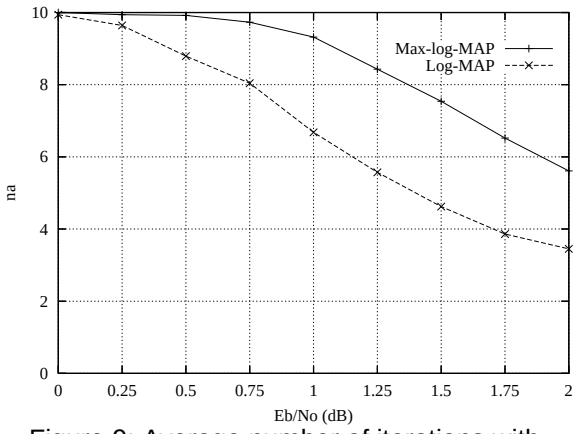

Figure 9: Average number of iterations with up to 10 iterations, block size 424, DVB–RCS, rate 1/2 and auto–stopping.

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_456.dat` in directory `input` (each line contains one bit of data). The data is input in the order A(0) B(0) ... A(K/2–1) B(K/2–1).

The encoded stream `y_(K).dat` will be output to the directory given by `out_dir`, e.g., `y_456.dat` to directory `output`. For rate k/n < 1/2, the encoded data is output in the order A(0) B(0) ... A(K/2–1) B(K/2–1) $Y_1(0)$ $Y_2(0)$ ... $Y_1(K/2–1)$ $Y_2(K/2–1)$ $W_1(0)$ $W_2(0)$ ... $W_1(K/(2k)–1)$ $W_2(K/(2k)–1)$. For rate k/n ≥ 1/2 the order is A(0) B(0) ... A(K/2–1) B(K/2–1) $Y_1(0)$ $Y_2(0)$ ... $Y_1(K/(2k)–1)$ $Y_2(K/(2k)–1)$.

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`. The file `r_(K).dat` has in each line R[i,j], i = 0 to 3 or 5 from j = 0 to *K*/2–1 with non–punctured parity data entered at their correct symbol positions, e.g., the first three lines of rate 2/3 punctured data could be

```
–31   1 –25  27
–31  12
 11  31  31   2
```

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, but with VA = 1. The START signal is used to start decoding, using RR and R0A to read the received data. Since only rate 1/2 or higher (with puncturing) is used, R2I to R5I are not used.

To allow longer block lengths, R1A is the same as R0A. This allows the turbo decoder input RAM for R2I and R3I to be used. For example the input RAMs for DVB–RCS are only 1K in width. If R0A is only used, this limits the input data length to 1018 bits (six bits are used by the tail). By using the input RAM for R2I and R3I we can extend the data length to 2042 bits. A multiplexer will be needed at the input to R0I and R1I, so that the appropriate input RAM is selected depending on R0A[10] delayed by one clock cycle.

The code used is the rate 1/2 64 state convolutional code with G0I = 171 and G1I = 133 (in octal) corresponding to R0I and R1I. For TB = 0, the input DELAY = 0 or 1 selects a delay of 134 or 262, respectively. For TB = 1, the input DELAY = 0 or 1 selects a delay of 198 or 390, respectively. For very high code rates such as 7/8, the longer delay is necessary to avoid severe performance degradation.

For TB = 0 and non–punctured rate 1/2 operation, the decoder inputs the received data from address 0 to *K*+5, where *K* can vary from 2 to 2042 for MODE[4] = 0 or 2 to 4090 for MODE[4] = 1. After a decoding delay, the decoded data is output to XD[0]. 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.

For TB = 1 and non–punctured operation, the input sequence is more complicated. For DELAY = 0, the data is input for 64 symbols from address (–64 mod *K*) to *K*–1, for *K* symbols from address 0 to *K*–1, and then for 64 symbols from address 0 to ((*K*+63) mod *K*). For DELAY = 1, the data is input for 128 symbols from address (–128 mod *K*) to *K*–1, for *K* symbols from address 0 to *K*–1, and then for 128 symbols from address 0 to ((*K*+127) mod *K*). The decoder automatically calculates the correct address for R0A, so the data only needs to be stored from address 0 to *K*–1.

Due to the modulus operation, the data lengths avalable for tail biting are restricted for use in the decoder. For DELAY = 0, data lengths from 22 to 511 bits can be used. For DELAY = 1, data lengths from 43 to 511 bits can be used.

For rates greater than 1/2 with N between 1 and 4, the decoder will automatically calculate the correct address for both terminated and tail biting codes and depuncture the received data. Table 9 shows the puncturing patterns used. X corresponds to G0I and Y to G1I. The transmitted data is C0 and C1, which corresponds to the received data R0I and R1I, respectively.

The output ERR[1:0] is the exclusive–OR of the sign bit of R0I and R1I with the corresponding re–encoded decoded output bit. This allows an estimate of the channel BER. Note that ERR is not

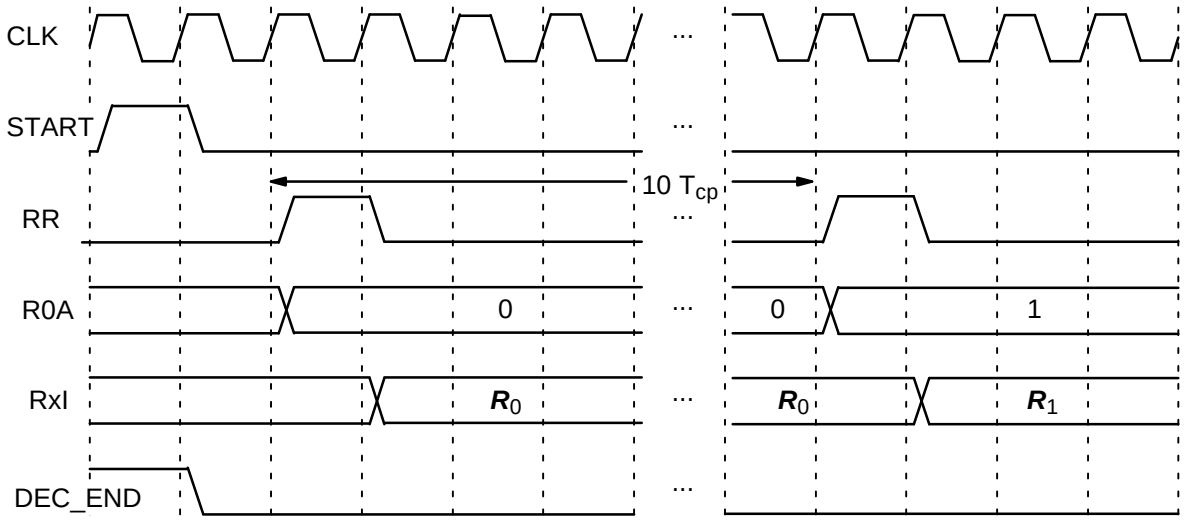

Figure 10: Viterbi Decoder Input Timing.

punctured.

Table 9: Convolutional Codes.

Rate	Pattern	Sequence
1/2	X: 1 Y: 1	C0: X_1 C1: Y_1
2/3	X: 1 0 Y: 1 1	C0: X_1 Y_2 Y_3 C1: Y_1 X_3 Y_4
3/4	X: 1 0 1 Y: 1 1 0	C0: X_1 Y_2 C1: Y_1 X_3
5/6	X: 1 0 1 0 1 Y: 1 1 0 1 0	C0: X_1 Y_2 Y_4 C1: Y_1 X_3 X_5
7/8	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0	C0: X_1 Y_2 Y_4 Y_6 C1: Y_1 Y_3 X_5 X_7

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

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

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

where F_d is the internal clock speed, N_c = 10 is the number of decoder clock cycles and D is the Viterbi decoder delay in bits. For example, if TB= 0, K = 504, D = 134 (DELAY = 0), and F_d = 100 MHz, decoding speed is 7.8 Mbit/s.

Ordering Information

SW–PCD03D–SOS (SignOnce Site License)
SW–PCD03D–SOP (SignOnce Project License)
SW–PCD03D–VHD (VHDL ASIC License)

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

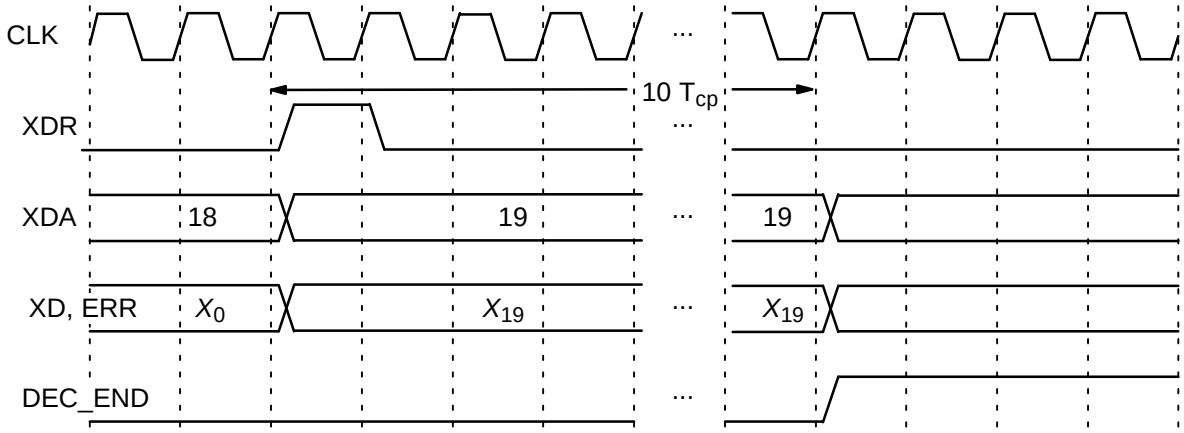

Figure 11: Viterbi Decoder Output Timing (K = 20).

Note that *Small World Communications* only provides software and does not provide the actual devices themselves. Please contact *Small World Communications* for a quote.

References

[1] ETSI, "Digital video broadcasting (DVB); Interaction channel for satellite distribution systems," EN 301 790 V1.5.1, May 2009.

[2] IEEE, "Standard for air interface for broadband wireless access systems," IEEE Std 802.16–2012, 17 Aug. 2012.

[3] 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.

[4] 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.

[5] W. Xiang and S. S. Pietrobon, "A new class of parallel data convolutional codes," *IEEE Australian Commun. Theory Workshop*, Brisbane, Australia, pp. 78–82, Feb. 2005.

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

- 1.00 21 June 2007 First official release.
- 1.01 17 September 2010 Added Virtex–5 performance.
- 1.02 21 October 2010 Deleted Virtex–II Pro performance. Improved Virtex–5 performance. Added Virtex–6 and Spartan–6 performance.
- 1.03 6 December 2010 Corrected Max–log–MAP performance in Figures 8 and 9. Added version history.
- 1.04 17 December 2010. Added ZRH[7:0] output.
- 1.05 27 January 2011. Corrected Viterbi decoder data length and delay. Added description for extending data length for Viterbi decoding. Added Genie aided early stopping mode for BER simulation software. Updated Virtex–4 and Virtex–5 complexity.
- 1.06 20 May 2011. Corrected KS description and fading channel scaling.
- 1.07 30 September 2011. Added tail–biting option for Viterbi decoder. Updated BER simulation program parameters.
- 1.08 18 November 2011. Updated BER simulation results. Corrected and updated standard references. Deleted references to WirelessMAN–SCa as no longer in standard. Deleted Spartan–3 and Virtex–4 performance. Improved Virtex–5, Virtex–6 and Spartan–6 performance. Added Kintex–7 performance. Deleted Virtex–4 resources. Updated Virtex–5 resources.
- 1.09 23 December 2011. Changed ZRH output to LXD.
- 1.10 30 December 2011. Added Spartan–3 and Virtex–4 performance and Virtex–4 complexity. Updated Virtex–5 complexity.
- 1.11 30 August 2012. Corrected format for punctured received data in file `r_(K).dat`.
- 1.12 3 June 2015. Corrected WirelessMAN–OFDM interleaver. Added Artix–7 performance. Updated Table 2.