VA08VB 16–ACS Multi Function Viterbi Decoder

24 October 2023 (Version 1.01)

Product Specification

VA08VB Features

- 16, 32, 64 or 256 states (memory *m* = 4, 5, 6 or 8, constraint lengths 5, 6, 7 or 9) Viterbi decoder
- Up to 574 MHz internal clock
- Up to 95 Mbit/s for 16, 32, or 64 states or 31 Mbit/s with 256 states
- Rate 1/2, 1/3, 1/4 or 1/5 implementations (inputs can be punctured for higher rates)
- Optional or standard 3GPP/3GPP2 code polynomials
- Signed magnitude or two's complement input data
- Optional 6–bit input with 8 or 10 bit state metrics or 8–bit input with 10 or 12 bit state metrics
- Optional continuous, continuous terminated, block terminated or block tail–biting decoding
- 1 to 1024–*m* or 32768–*m* data bits for terminated or *m* to 511 data bits for tail biting
- Estimated channel bit error outputs
- Optional serial (continuous only) or parallel data input
- Optional automatic coded symbol synchronisation for rate 1/2 QPSK and rate 1/2 to 1/5 BPSK
- Xilinx: From 958 to 2204 LUTs and 1 to 5 18Kb BlockRAMs.
- Asynchronous logic free design
- Free simulation software
- Available as EDIF and VHDL core for Xilinx FPGAs under SignOnce IP License. ASIC, Intel/Altera, Lattice and Microsemi/Actel cores available on request.

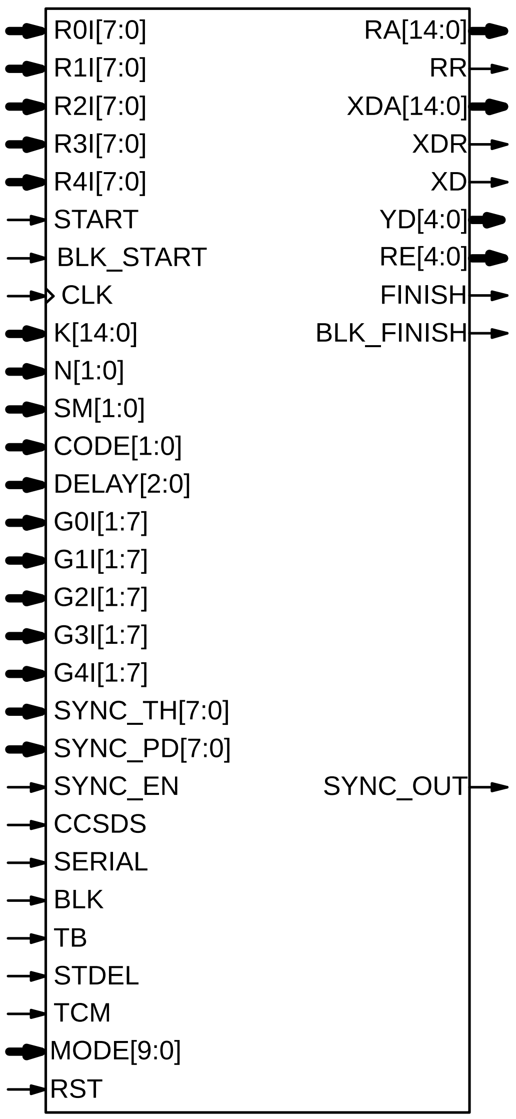

Figure 1: VA08VB schematic symbol.

Introduction

The VA08VB is a 16, 32, 64 or 256 state error control decoder using the maximum likelihood Viterbi algorithm [1]. The decoder can be used to decode 256 state 3GPP UMTS terminated codes [2], 256 state 3GPP2 1xEV–DV terminated codes [3], 64 state 3GPP LTE tail biting code [4], 64 state CCSDS continuous code [5] as well as other custom coding solutions.

The decoder can decode continuously (without or with terminated data) or in blocks using an external input memory (with terminated or tail–biting data). With continuous non–terminated operation, automatic synchronisation is available for rate 1/2 to 1/5 BPSK serial and rate 1/2 QPSK. Sign–magnitude or two's complement inputs can be used. Four quantisation options are available; 6–bit inputs with 8 or 10–bit internal state metrics (SM) or 8–bit inputs with 10 or 12 bit SM.

The VA08VB uses 16 add–compare–select (ACS) circuits in parallel up to four times for 16, 32 and 64 states and 16 times for 256 state convolutional codes. An internal configurable 2Kx8, 4Kx8, 6Kx8, 8Kx8 or 10Kx8 simple dual port synchronous RAM (implemented using up to five 2Kx8 18Kb BlockRAMs) is used to perform the traceback. In synchronous operation, six clock cycles are required per decoded bit for 16, 32, or 64 states or 18 clock cycles for 256 states. Asynchronous operation requires up to 7 or 19 clock cycles.

Figure 1 shows the schematic symbol for the VA08VB 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 (parallel input and no automatic synchronisation). T_{cp} is the minimum clock period over recommended operating conditions. These performance figures may change due to device utilisation and configuration. Note that Zynq devices up to XC7Z020 and from XC7Z030 use programmable logic equivalent to Artix–7 and Kintex–7 devices, respectively.

Table 1: Performance of Xilinx parts.*

Xilinx Part	T_{cp} (ns)	m=6* (Mbit/s)	m=8* (Mbit/s)
XC7S50–1	5.713	29.1	9.7
XC7S50–2	4.657	35.7	11.9
XC7A15T–1	5.696	29.2	9.7
XC7A15T–2	4.667	35.7	11.9
XC7A15T–3	4.124	40.4	13.4
XC7K70T–1	4.018	41.4	13.8
XC7K70T–2	3.380	49.3	16.4
XC7K70T–3	3.047	54.6	18.2
XCKU035–1	3.108	53.6	17.8
XCKU035–2	2.693	61.8	20.6
XCKU035–3	2.247	74.1	24.7
XCKU3P–1	2.270	73.4	24.4
XCKU3P–2	1.908	87.3	29.1
XCKU3P–3	1.741	95.7	31.9

*Synchronous operation

Signal Descriptions

BLK	Decoder Operation
	0 = Continuous
	1 = Block
BLK_FINISH	Block Finish (continuous only)
BLK_START	Block Start (continuous only)
CCSDS	Invert Sign Bit of R1I
CLK	System Clock
CODE	Code Select (see Table 3)
	0 = 3GPP Polynomials
	1 = 3GPP2 Polynomials
	2,3 = Use G0I to G4I
DELAY	Decoder Delay Select
	0 = 64+*m*
	1 = 128+*m*
	2 = 192+*m*
	3 = 256+*m*
	4 = 320+*m*
FINISH	Decoder Finish
G0I–G4I	Code Polynomial Input
K	Data Length
	TB = 0: 1 to 1024–*m* or 32768–*m*
	TB = 1: m to 511
MODE	Implementation Mode (see Table 2, connect to VCC or GND only)
N	Convolutional Code Rate
	2 = Rate 1/2
	3 = Rate 1/3
	0 = Rate 1/4
	1 = Rate 1/5
R0I–R4I	Received Data
RE	Estimated Symbol Error
RST	Synchronous Reset
SERIAL	0 = Parallel Input (R0I to R4I)
	1 = Serial Input (R0I only)
SM	Code State Select
	0 = 16 States (*m* = 4)
	1 = 32 States (*m* = 5)
	2 = 64 States (*m* = 6)
	3 = 256 States (*m* = 8)
START	Decoder Start
STDEL	Internal Start Delay (BLK = 1)
	0 = One Clock Cycle Delay
	1 = Two Clock Cycle Delay
SYNC_EN	Synchronisation Enable
SYNC_OUT	Synchronisation Output SIgnal
SYNC_PD	Synchronisation Period (1 to 255)
SYNC_TH	Synchronisation Threshold (1 to 255)
TB	Block Operation
	0 = Terminated
	1 = Tail Biting
TCM	Input Data Format
	0 = Sign Magnitude
	1 = Two’s Complement
XD	Decoded Data Output
YD	Decoded Symbol Output

Table 2 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. They should be connected to VCC or GND only. Connecting these pins to logic or input pins will result in excessive logic utilisation and decreased decoding speeds. These inputs are designed to minimise decoder complexity for the configuration selected.

The MODE[2] and MODE[5:3] inputs can be used to reduce the number of BlockRAMs used by the decoder. For MODE[2] = 1, either 1, 2, 3, 4 or 5 BlockRAMs are used for MODE[5:3] = 0, 1, 2, 3 or 4, respectively. For MODE[2] = 0, either 1 BlockRAM is used for MODE[5:3] = 0 to 3 or 2 BlockRAMs are used for MODE[5:3] = 4.

Table 2: MODE selection

Input	Description
MODE[1:0]	0 = Rate 1/2 1 = Rate 1/2 to 1/3 2 = Rate 1/2 or 1/4 3 = Rate 1/2 to 1/5
MODE[2]	0 = Memory 4 to 6 (SM $\leq$ 2) (1 or 2 BlockRAMs) 1 = Memory 4 to 6 and 8 (all SM) (1 to 5 BlockRAMs)
MODE[5:3]	0 = DELAY 0 1 = DELAY 0 to 1 2 = DELAY 0 to 2 3 = DELAY 0 to 3 4 = DELAY 0 to 4
MODE[7:6]	0 = 6–bit input, 8–bit SM 1 = 6–bit input, 10–bit SM 2 = 8–bit input, 10–bit SM 3 = 8–bit input, 12–bit SM
MODE[8]	0 = Data lengths to 1024–m bits 1 = Data lengths to 32768–m bits
MODE[9]	0 = Continuous non–terminated 1 = Continuous terminated or block

Code Selection

Figure 2 gives a block diagram of a 256 state (memory *m* = 8, constraint length 9) non–systematic encoder. X is the data input and Y0 to Y3 are the coded outputs. GiIj = $g_i^j \in \{0, 1\}$, $0 \leq i \leq 3$, $1 \leq j \leq 7$, correspond to the code polynomial coefficients which are used by the decoder.

The encoder polynomials are defined as

$$g_i(D) = 1 + g_i^1 D + g_i^2 D^2 + ... + g_i^7 D^7 + D^8 \quad (1)$$

where *D* is the delay operator and + indicates modulo–2 (exclusive OR) addition. It is usual practice to express the coefficients in octal notation, e.g., $g_0 = 561_8 = 101110001_2 \equiv g_0(D) = 1 + D^2 + D^3 + D^4 + D^8$. This corresponds to G0I[1:7] = 0111000_2.

When CODE[1] = 1, the code polynomials input to G0I[1:7] to G4I[1:7] are used. The input N[1:0] is also used to deselect the inputs for the various rates. That is, R2I[7:0] is internally grounded for rate 1/2, R3I[7:0] is internally grounded for rates 1/2 and 1/3 and R4I[7:0] is internally grounded for rates 1/2 to 1/4.

The 3GPP UMTS [2] and 3GPP2 [3] convolutional code standards are selected by CODE = 0 and 1, respectively. The codes are given in Table 3 in octal notation.

Figure 3 shows the 64 state (*m* = 6, constraint length 7) encoder. To decode 64 state encoded data, select SM = 2. The code polynomials are given by

Table 3: Convolutional Codes.

CODE [1:0]	SM [1:0]	N [1:0]	g0	g1	g2	g3	g4
0X	0	2	23	35	–	–	–
0X	0	3	25	33	37	–	–
0X	0	0	23	35	25	37	–
0X	0	1	37	35	31	27	25
0X	1	2	51	67	–	–	–
0X	1	3	51	67	75	–	–
0X	1	0	51	55	67	77	–
0X	1	1	77	73	71	55	45
00	2	2	133	171	–	–	–
00	2	3	133	171	165	–	–
0X	2	0	173	167	135	111	–
0X	2	1	175	171	155	127	113
00	3	2	561	753	–	–	–
00	3	3	557	663	711	–	–
0X	3	0	473	513	671	765	–
0X	3	1	755	651	637	561	453
01	2	2	171	133	–	–	–
01	2	3	171	133	165	–	–
01	3	2	753	561	–	–	–
01	3	3	557	663	711	–	–
1X	X	X	G0I	G1I	G2I	G3I	G4I

$$g_i(D) = 1 + g_i^1 D + g_i^2 D^2 + g_i^3 D^3 + g_i^4 D^4 + g_i^5 D^5 + D^6 \quad (2)$$

where GiI6 and GiI7 are equal to 0. For example, if g0 = 171, then G0I[1:7] = 1110000. Table 3 shows the 64 state codes selected for CODE = 0 or 1, corresponding to standard rate 1/2 and 1/3 convolutional codes. For rate 1/4 and 1/5, a code was selected from [6].

Similarly, we have

$$g_i(D) = 1 + g_i^1 D + g_i^2 D^2 + g_i^3 D^3 + g_i^4 D^4 + D^5 \quad (3)$$

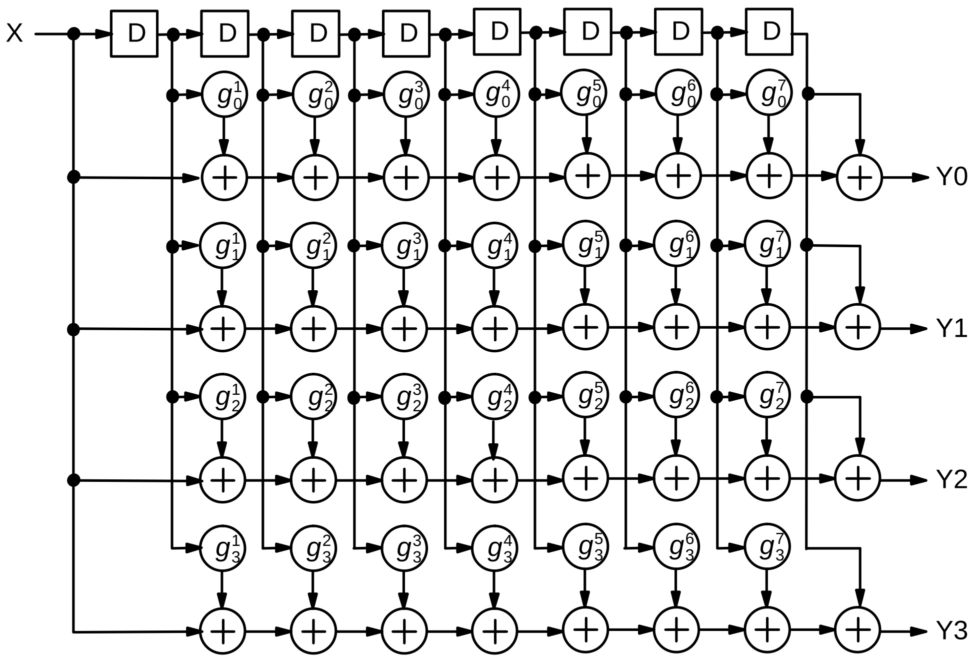

Figure 2: 256 state (*m* = 8, constraint length 9) rate 1/4 non–systematic convolutional encoder.

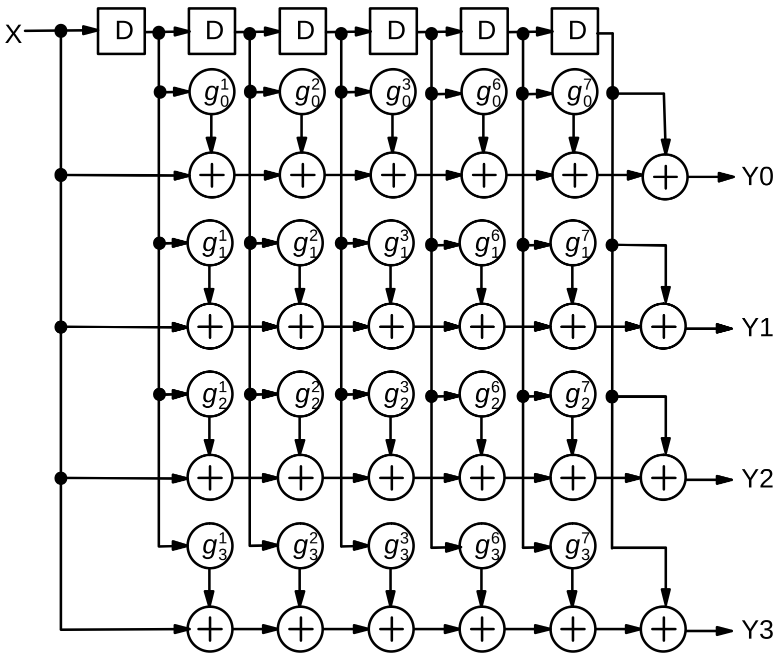

Figure 3: 64 state (*m* = 6, constraint length 7) rate 1/4 non–systematic convolutional encoder.

for 32 state codes (SM = 1, GiI5 to GiI7 equal to zero) and

$$g_i(D) = 1 + g_i^1 D + g_i^2 D^2 + g_i^3 D^3 + D^4 \qquad (4)$$

for 16 state codes (SM = 0, GiI4 to GiI7 equal to zero).

Viterbi Decoder

The Viterbi decoder is designed to be very flexible and can be operated in either continuous or block mode.

Theory of Operation

The Viterbi decoding algorithm [1] finds the most likely transmitted sequence given the received noisy sequence.

For binary phase shift keying (BPSK) or quadrature phase shift keying (QPSK) modulation the received signal is described by

$$R_k^i = A((1 - 2y_k^i)/\sqrt{b} + n_k^i) \qquad (5)$$

where A is the signal amplitude, $y_k^i \in \{0, 1\}$, i = 0 to 3 correspond to the coded bits, b = 1 for BPSK or b = 2 for QPSK, and n_k^i is a Gaussian distributed

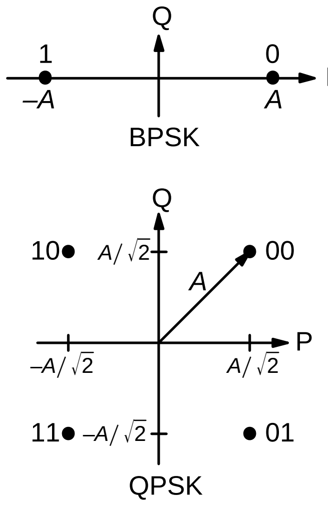

Figure 4: BPSK and QPSK signal sets.

random variable with zero mean and normalised variance σ^2. Figure 4 shows the signal sets for BPSK and QPSK. We have

$$\sigma^2 = \left(2bR\frac{E_b}{N_0}\right)^{-1} \tag{6}$$

where E_b/N_0 is the energy per bit to single sided noise density ratio and $R = k/n$ is the code rate (k is the number of information bits and n is the number of coded bits).

Since a zero is transmitted as $+A/\sqrt{b}$ and a one is transmitted as $-A/\sqrt{b}$ the sign bit of a noiseless R_k^0 in two's complement notation is equal to d_k.

Due to quantisation and limiting effects the value of A should also be adjusted according to the received signal to noise ratio. A program called *cmap* for calculating the optimum values of A is included with the cores.

The value of A directly corresponds to the 6 or 8–bit signed magnitude inputs (described in more detail later). For 6–bit inputs (using the six most significant bits of R0I[7:0] to R4I[7:0]) there are 63 quantisation regions with a central dead zone. The quantisation regions are labelled from –31 to +31. The 8–bit inputs have 255 quantisation regions with the regions labelled from –127 to +127. For example, with 8–bit inputs we could have A = 62.8. This value of A lies in quantisation region 63 (which has a range between 62.5 and 63.5).

Example 1: Rate 1/3 BPSK code operating at E_b/N_0 = 3 dB. From (6) we have σ^2 = 0.75178. Using cmap we have that A = 47.45.

Example 2: Rate 1/2 QPSK code operating at E_b/N_0 = 4 dB. From (6) we have σ^2 = 0.19905. Using cmap we have that A = 113.3. Note that the amplitude in each dimension is $A/\sqrt{2}$ = 80.12.

Decoder Operation

The VA08VB uses a finite traceback memory and is thus able to continuously decode data. The traceback depth is determined by DELAY and SM. Table 4 gives the minimum decoding depth, maximum decoding depth and decoder delay as a function of DELAY and SM.

The VA08VB uses 16 ACS circuits in parallel. Thus, 16 clock cycles are required to perform 256 ACS operations or four clock cycles for 64 ACS operations. For 16 and 32 states, four clock cycles are still used even though the ACS circuits only require one and two clock cycles, respectively. This allows the minimum traceback depth to remain the same for 16, 32 and 64 states. An additional two clock cycle overhead is also required.

Table 4: Decoding depth and delay.

SM	DELAY	Min Depth	Max Depth	Delay
0,1,2	0	33	48	68,69,70
0,1,2	1	65	96	132,133,134
0,1,2	2	97	144	196,197,198
0,1,2	3	129	192	260,261,262
0,1,2	4	161	240	324,325,326
3	0	57	60	72
3	1	113	120	136
3	2	169	180	200
3	3	225	240	264
3	4	281	300	328

Continuous Operation

For continuous operation (BLK = 0), the decoder uses a rising edge detector circuit at the START input to start decoding the received data. If the high period of the START input is greater than the CLK period, the decoder will start decoding. To detect the next rising transition, the START input must be low for a least one CLK period.

This allows the decoder to be operated in synchronous or asynchronous operation. Synchronous operation requires N_c = 6 clock cycles per decoded bit for 16, 32 or 64 states or N_c = 18 clock cycles per bit for 256 states. For asynchronous operation, the maximum decoder speed is

$$f_d = (N_c/F_d + T_{su} + T_{hd} + T_j)^{-1} \tag{7}$$

where F_d is the CLK frequency, T_{su} and T_{hd} are the setup and hold time of the START edge detector flip flop and T_j is the peak jitter or Doppler shift of

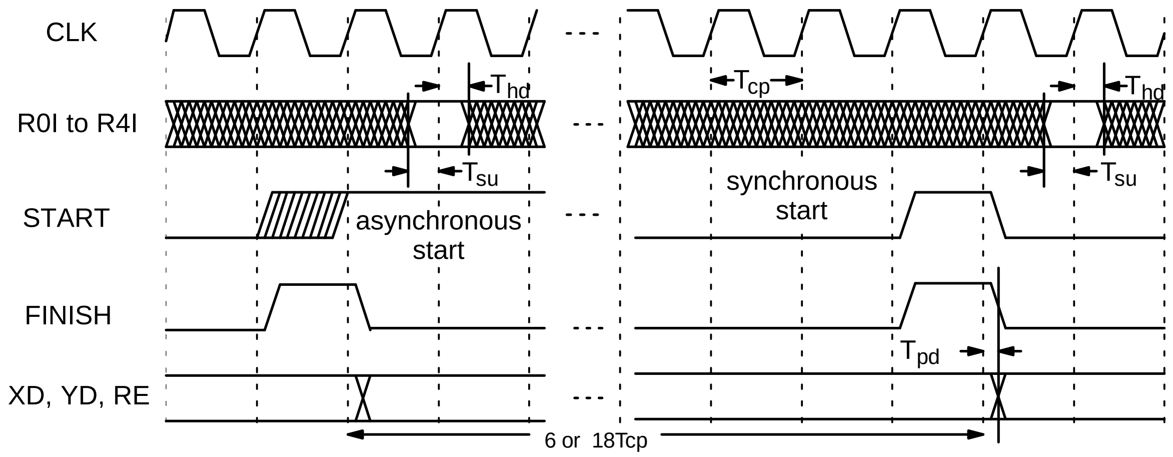

Figure 5: Continuous operation input and output timing (BLK = 0).

the START clock. For example, if F_d = 300 MHz, N_c = 6, T_{su} = 0.04 ns and T_{hd} = 0.14 ns for a Virtex–7 –3 device and T_j = 0, then f_d = 49.55 Mbit/s.

Figure 5 shows the relationship betwen the START input and R0I to R4I. In synchronous operation, these inputs must be valid from $2T_{cp}–T_{dsu}$ to $2T_{cp}+T_{dhd}$ after the rising edge of START (T_{cp}, T_{dsu}, and T_{dhd} are the decoder clock period, setup time, and hold time, respectively).

In asynchronous operation these signals must be valid from $T_{cp}–T_{dsu}$ to $2T_{cp}+T_{dhd}$ after the rising edge of START. Data must therefore change within one clock cycle after the rising edge of START.

The FINISH output goes high during the last clock cycle of the decoding operation. In continuous synchronous operation, the rising edges of START and FINISH should be coincident.

The decoded output XD, the re–encoded outputs YD[4:0] and estimated channel BER outputs RE[4:0] changes when FINISH goes low. RE[4:0] are obtained by exclusive ORing the appropriately delayed sign bit of the inputs with YD[4:0]. At low BER, these outputs can be used to give a good estimate of the channel BER.

Continuous Terminated Operation

For continuous terminated operation, operation is the same as for continuous, except that the input BLK_START is used to indicate the start of a terminated block with data length K that is input to K[14:0]. The number of coded symbols is $K+m$ and the number of coded bits is $(K+m)n$ for a rate $1/n$ convolutional code. BLK_START must be valid at the same time as for R0I to R4I. The START input must go high $K+m$+64(DELAY+1) times in order to output the decoded block.

Input K[14:0] must be valid for the entire duration of the input data for the terminated block. At the end of the block of input data, the next block can be immediately input with a new value of K if desired. If the value of K is not immediately known, a higher value can initially be input and then later changed to the correct value at least two symbols before the end of the received block.

Outputs RA[14:0] and XDA[14:0] change values after FINISH goes high. Output RR will go high two clock cycles after START goes high and one clock cycle after BLK_START goes high and will go low one clock cycle after RA[14:0] = $K+m–1$ and FINISH goes high. After 64(DELAY+1)+m inputs, XDR will go high for one clock cycle at the beginning of each decoded output bit. Output BLK_FINISH will go high for one clock cycle when XDA[14:0] = $K–1$ and FINISH goes high.

Figure 6 shows the required timing for asynchronous operation. The second block shows how K can change. Figures 7 and 8 shows the synchronous input data timing at the start and beginning of the block, respectively. Figures 9 and 10 shows the synchronous output data timing at the start and beginning of the block, respectively.

Block Operation

For block operation (BLK = 1), the received data is stored in an external synchronous read input memory. The START signal goes high only once to start decoding. The outputs RR and RA are used to read the received data. For rate 1/2 operation, R2I to R4I are not used. For rate 1/3 operation R3I and R4I are not used. For rate 1/4 operation R4I is not used. The output FINISH stays low until the last clock cycle of the last decoded bit.

The received data can be input either one clock cycle (STDEL = 0) or two clock cycles (STDEL = 1) after RR goes high.

For terminated codes (TB = 0), the decoder DELAY = 0 to 4 selects a delay equal to 64(DELAY+1)+m, where m is the encoder memory. For tail biting codes (TB = 1), the total delay

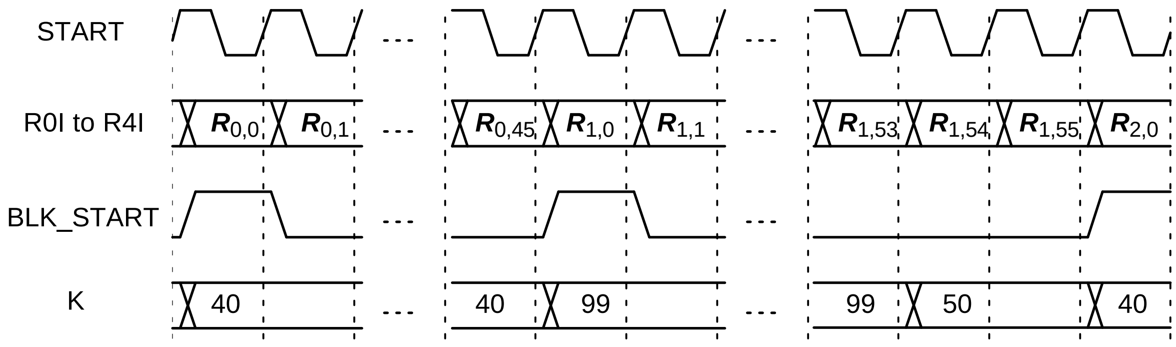

Figure 6: Continuous terminated operation asynchronous input (BLK = 0, m = 6).

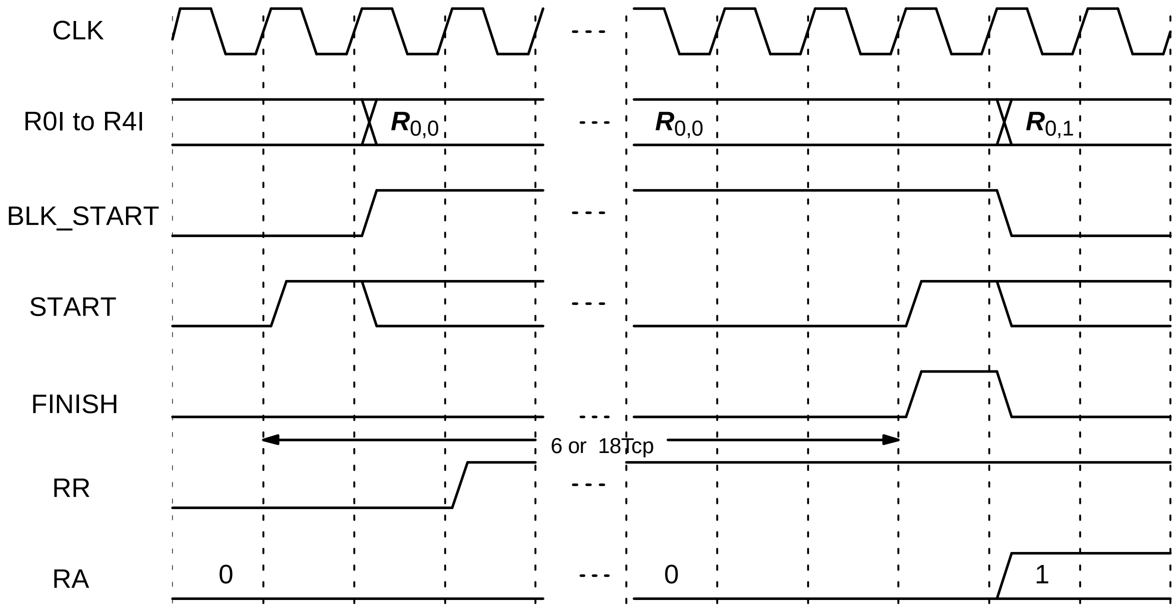

Figure 7: Continuous terminated operation input data start, synchronous input (BLK = 0).

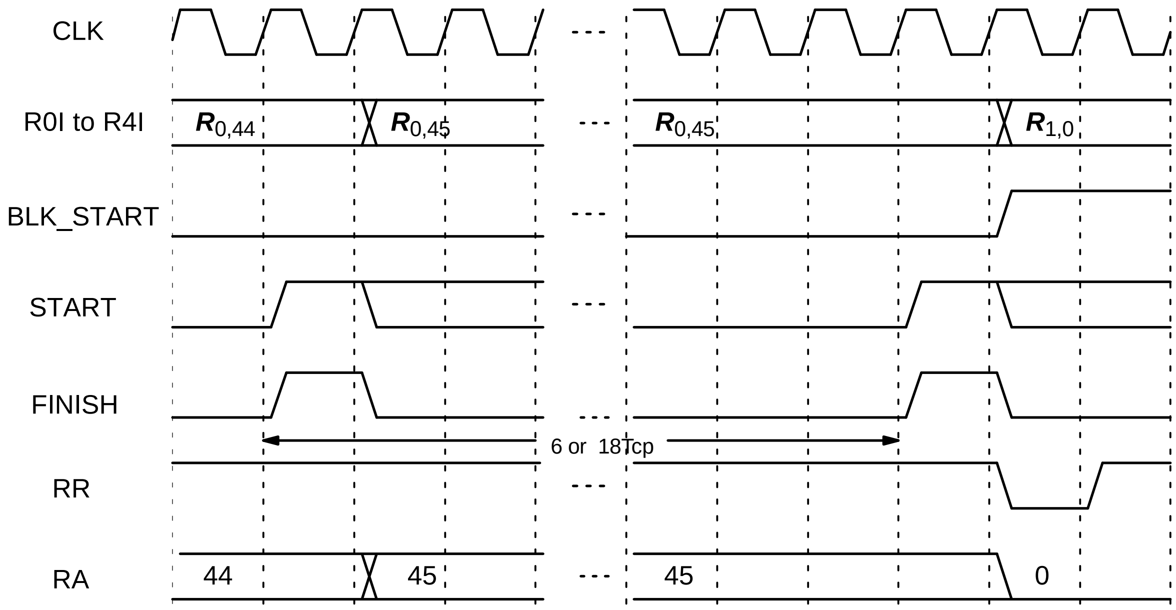

Figure 8: Continuous terminated operation input data end, synchronous input (BLK = 0, K = 40).

is 128(DELAY+1)+m. This assumes that the encoder start state is equal to the last m bits of the data block.

For TB = 0, the decoder first inputs the received data from address 0 to K–1, where K is the information data length which can vary from 1 to

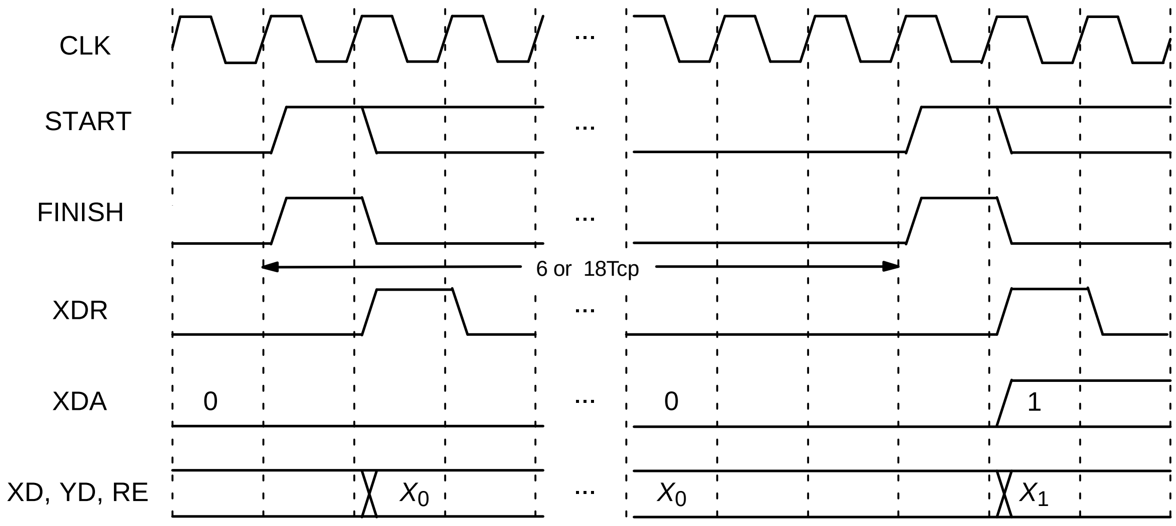

Figure 9: Continuous terminated operation output data start, synchronous input (BLK = 0).

1024–*m* or 32768–*m* bits. The tail is then input from address *K* to *K*+*m*–1. 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.

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

Due to the modulus operation, the data lengths available for tail biting are restricted for use in the decoder. Data lengths from *m* to 511 bits can be used.

Figures 11 and 12 shows the Viterbi decoder input timing for terminated and tail–biting codes, respectively. Either one or four clock cycles are used to start decoding (for terminated and tail biting codes, respectively), with each decoded bit taking 6 or 18 clock cycles. Figure 13 shows the Viterbi decoder output timing.

The decoding speed is given by

$$f_d = \frac{F_d}{N_c(1 + D/K) + S/K} \tag{8}$$

where F_d is the internal clock speed, N_c is the number of decoder clock cycles (6 or 18), *D* is the total decoding delay equal to 64(TB+1)(DELAY +1)+*m*, and *S* = 2+3TB+STDEL is the start delay.

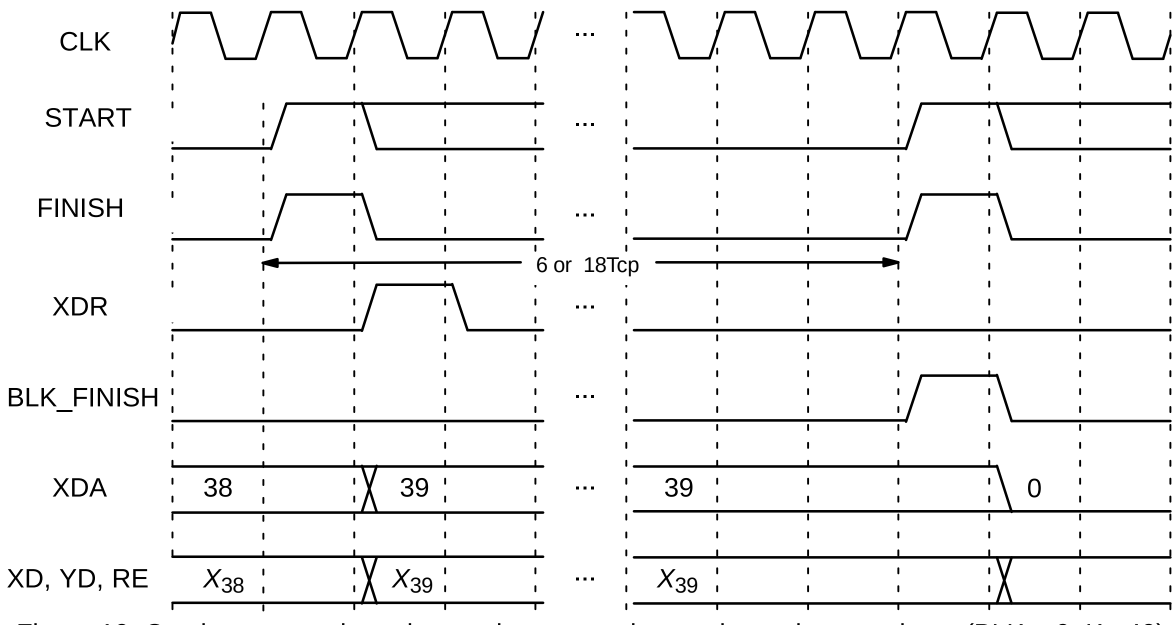

Figure 10: Continuous terminated operation output data end, synchronous input (BLK = 0, K= 40).

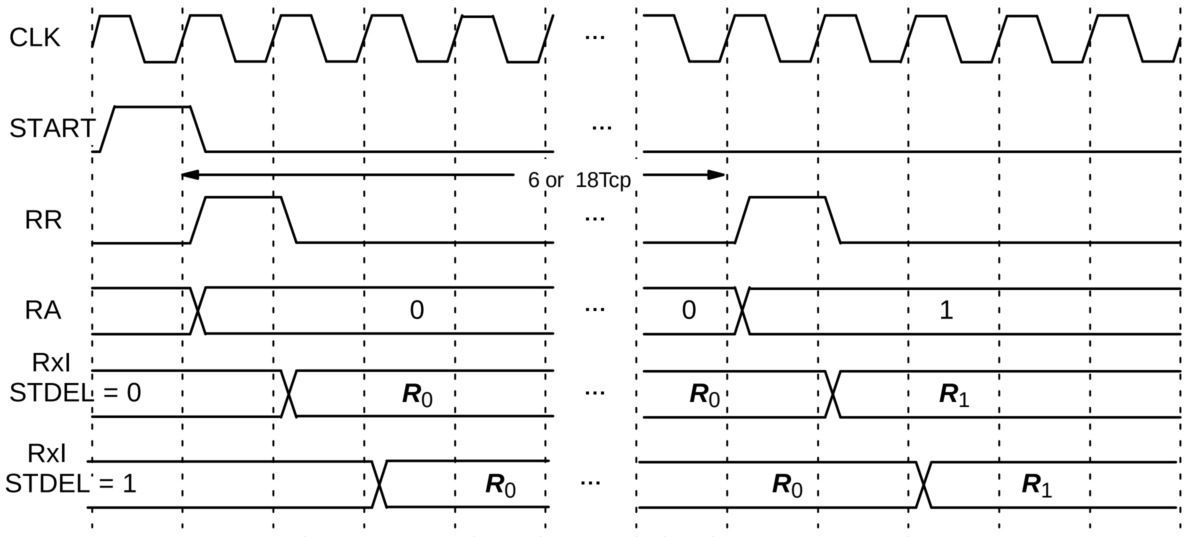

Figure 11: Terminated Input Timing (BLK = 1, TB = 0).

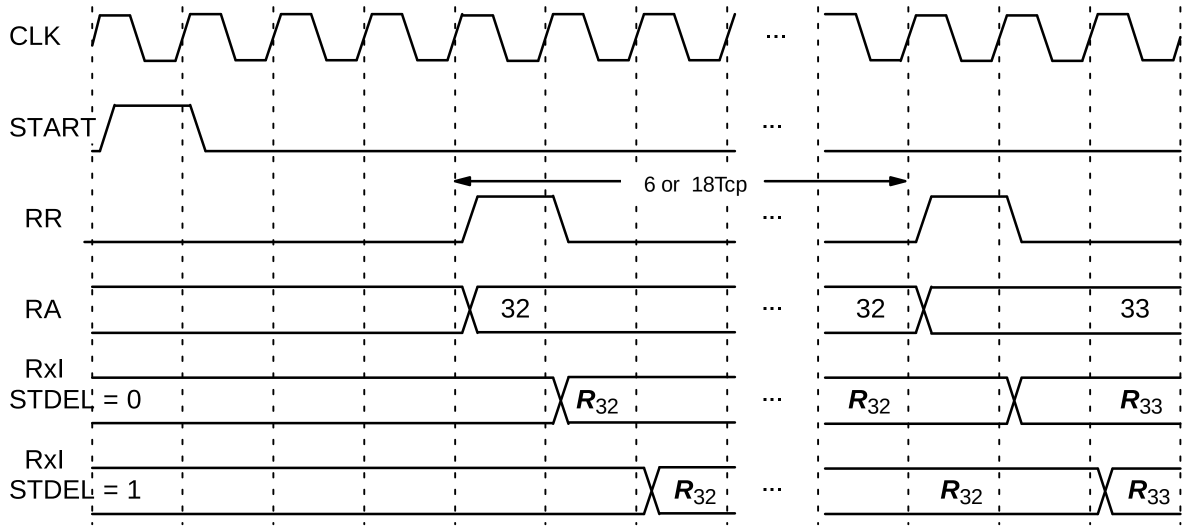

Figure 12: Tail Biting Input Timing (K = 48, DELAY = 0, BLK = 1, TB = 1).

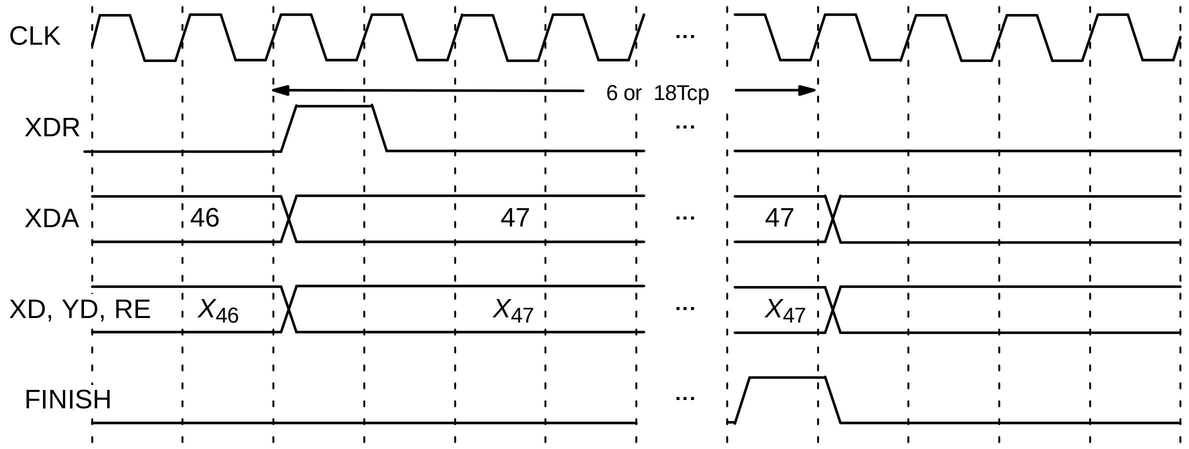

Figure 13: Viterbi Decoder Output Timing (K = 48, BLK = 1).

For example, if TB = 0, STDEL = 0, DELAY = 1, K = 192, SM = 3 (m = 8 and N_c=18) and F_d = 200 MHz then D = 136, S = 2 and the decoding speed is 6.50 Mbit/s.

Input Data Format

The decoder internally uses 6 or 8–bit signed magnitude quantisation for R0I to R4I. External data can be input in either sign–magnitude (TCM = 0) or two's complement (TCM = 1) format. Two's complement inputs equal to –32 or –128 are internally limited to –31 or –127, for 6 or 8–bit quantisation, respectively.

Tables 5 and 6 shows the 8–bit quantisation ranges for sign magnitude and two's complement inputs, respectively. 6–bit quantisation is similar, with ranges from –31 to 31 for signed magnitude and –32 to 31 for two's complement.

Table 5: 8–bit Sign Magnitude Quantisation

Int–eger	Dec–imal	Binary	Range (Min)	Range (Max)
127	127	01111111	126.5	∞
126	126	01111110	125.5	126.5
⋮	⋮	⋮	⋮	⋮
2	2	00000010	1.5	2.5
1	1	00000001	0.5	1.5
0	0	00000000	–0.5	0.5
0	128	10000000	–0.5	0.5
–1	129	10000001	–1.5	–0.5
–2	130	10000010	–2.5	–1.5
⋮	⋮	⋮	⋮	⋮
–126	254	11111110	–126.5	–125.5
–127	255	11111111	–∞	–126.5

Table 6: 8–bit Two's Complement Quantisation

Int–eger	Dec–imal	Binary	Range (Min)	Range (Max)
127	127	01111111	126.5	∞
126	126	01111110	125.5	126.5
⋮	⋮	⋮	⋮	⋮
2	2	00000010	1.5	2.5
1	1	00000001	0.5	1.5
0	0	00000000	–0.5	0.5
–1	255	11111111	–1.5	–0.5
–2	254	11111110	–2.5	–1.5
⋮	⋮	⋮	⋮	⋮
–126	130	10000010	–126.5	–125.5
–127	129	10000001	–127.5	–126.5
–128	128	10000000	–∞	–127.5

Note that for sign magnitude, 0 and 32 (6–bit) or 0 and 128 (8–bit) indicate the central dead zone and have the same range. If the analog to digital to digital (A/D) converter does not have a central dead zone we recommend that 7–bit or 9–bit A/Ds are used with the output converted to 6–bit or 8–bit so that the appropriate ranges are obtained.

For input data quantised to less than 8–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 6–bit received data R0T[5:0], where R0T[5] is the sign bit, we have R0I[7:2] = R0T[5:0] and R0I[1:0] = 2 in decimal (10 in binary). If MODE[6] = 0, then the two least signficiant bits for each input shoud be tied low, e.g., R0I[1:0] = R1I[1:0] = 0 for rate 1/2.

Punctured Code Operation

Manual puncturing can be performed by forcing all bits of R0I[7:0] to R4I[7:0] low. For example, rate 2/3 can be obtained by puncturing a rate 1/2 code with puncturing patterns of 11 for R0I and 10 for R1I. That is, R0I is not punctured, while R1I is forced low every other decoded bit.

Serial Operation

When the SERIAL input is high, the decoder uses an internal serial to parallel converter to convert serially received data, for example in BPSK modulation, into parallel received data. The received clock should be input to START. This clock must not be divided down and must be equal to the received binary symbol rate. The received data must be valid from one to two CLK cycles after the rising edge of START and be input to R0I only.

The data corresponding to Y0 to Y4 is assumed to be received in this order. For example for rate 1/2, the data for Y0 is received first, followed by the data for Y1.

Note that FINISH only goes high at the end of each received code symbol. This consists of *n* received data symbols for a rate 1/*n* code. Due to the serial to parallel operation, the decoder delay increases by *n*–1 received data periods. If desired, the decoder can also be operated using continuous terminated data.

The design will also work with serial input data and continuous terminated data. BLK_START should go high for one coded bit at the start of the block. Note that if the decoder is still decoding data, the next BLK_START be must be after *n* coded bits, otherwise the first coded symbol will be corrupted. The following symbols will be correct.

Automatic Synchronisation

When BLK = 0 and SYNC_EN = 1, automatic synchronisation to the coded symbol is enabled. This counts the number of state metric normalisations within the decoder. If the count exceeds the synchronisation threshold (SYNC_TH) before the end of the synchronisation period (SYNC_PD), SYNC_OUT will go high for one code symbol period. The normalisation counter is then disabled for one SYNC_PD, to allow the decoder to settle to its new synchronisation state. A new count is then started. If the threshold is not exceeded at the end of the SYNC_PD, the normalisation and period counters are reset, and a new count is started.

When SYNC_EN is high, SYNC_OUT is internally used by the decoder to change the synchronisation state. For serial operation, the code symbol period is increased by one received data period. This is performed only once, and causes the serial to parallel conversion to load one received data period later. With rate 1/2 Gray mapped QPSK operation, the received data is rotated by 0 or 90°, depending on the synchronisation state (0 or 1).

Note that if SYNC_EN is low, SYNC_OUT is not disabled (however, the internal synchronisation state is not allowed to change). This allows SYNC_OUT to be externally used to control the synchronisation state of an external synchronisation circuit.

With rate 1/2 operation at an E_b/N_0 of 4.2 dB (corresponding to a BER of 8.3×10^{-6} with DELAY = 1), the state metric normalisation rate is about 6.7×10^{-3} with 8–bit input data. When the decoder is out is sync though, the normalisation rate increases to about 4.7×10^{-2}. Thus, with a SYNC_PD of 128, a SYNC_TH of $128 \times 4.7 \times 10^{-2} = 6$ should provide a robust threshold. The average count when in sync is less than one. This should ensure that the decoder does not lose sync when it is in sync and that it quickly synchronises when out of sync.

CCSDS Operation

The VA08VB core can also be used to decode the CCSDS [5] rate 1/2 64 state convolutional code. The CCSDS code is selected with CODE = 1, SM = 2, and N = 2. The CCSDS code inverts Y1 to aid with demodulation. When the CCSDS input to VA08VB is high, the sign bit of the received data corresponding to Y1 is also inverted, allowing the decoder to decode CCSDS transmitted data.

Note that there is no differential encoder used for the CCSDS encoder. Since the code used is 180° rotationally invariant (either for BPSK or Gray mapped QPSK), this implies the synchronisation circuit can not detect 180° rotations. This implies that the decoded data could be inverted due to a 180° rotation. This will need to be externally detected, so that non–inverted data is used.

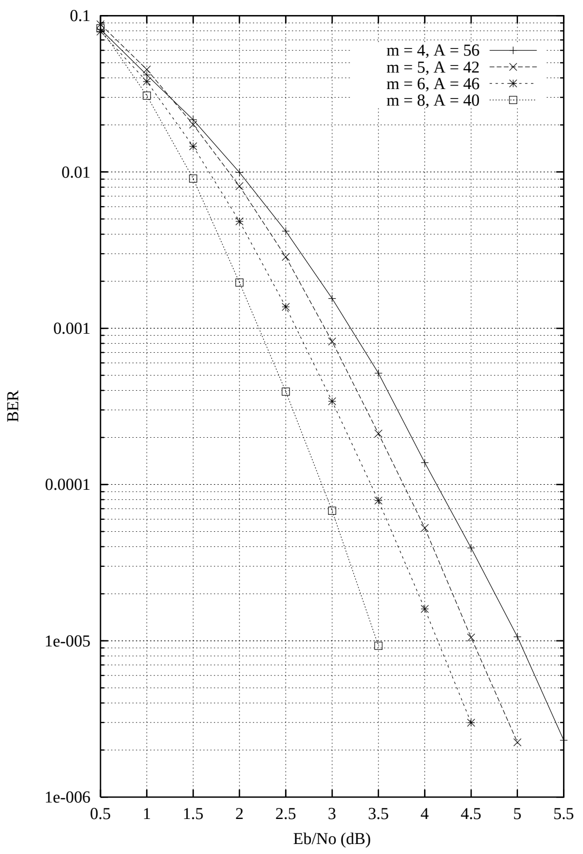

Figure 14: Rate 1/2 BER performance.

Other inputs

The RST input when high synchronously forces all flip–flops low. This is useful for VHDL simulations where flip–flops are initially in an unknown state. The decoded output will be unknown until the unknown data in RAM is flushed out. The length of the unknown output data should be equal to the decoder delay.

Simulation Software

Free software for simulating the VA08VB Viterbi decoder in additive white Gaussian noise (AWGN) is available by sending an email to info@sworld.com.au with “va08vbsim request” in the subject header. The software uses an exact functional simulation of the VA08VB Viterbi decoder, including all quantisation and limiting effects.

Figures 14 to 17 shows the 8–bit AWGN performance with binary modulation obtained for rate 1/2, 1/3, 1/4 and 1/5 convolutional codes decoded by the VA08VB Viterbi decoder with continuous decoding, CODE = 0 and DELAY = 3.

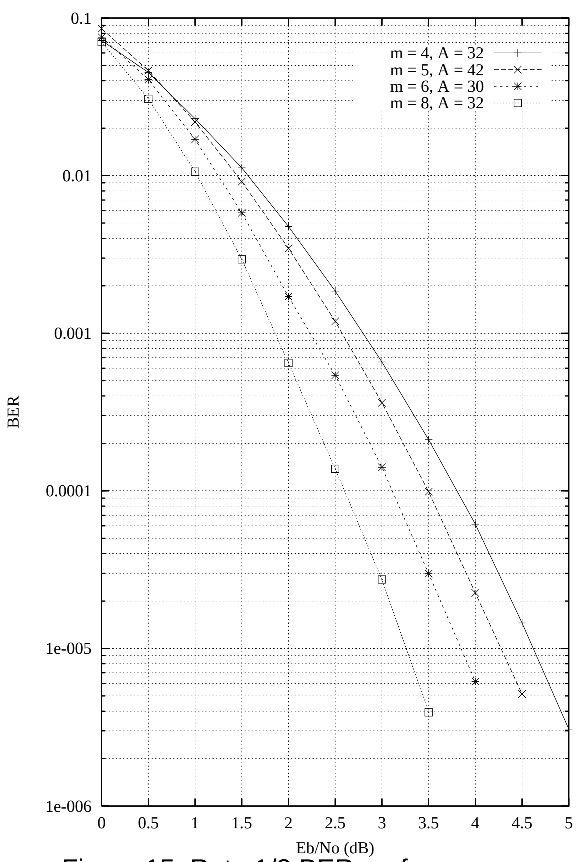

Figure 15: Rate 1/3 BER performance.

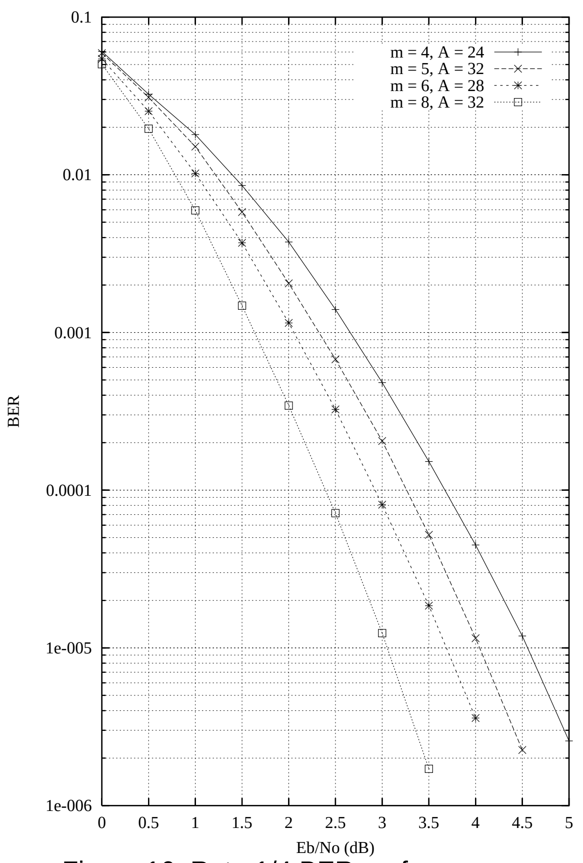

Figure 16: Rate 1/4 BER performance.

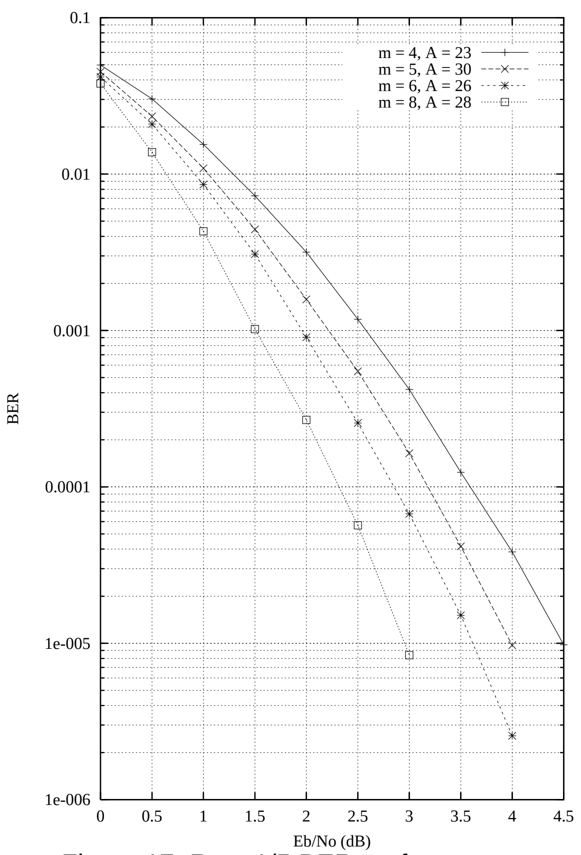

Figure 17: Rate 1/5 BER performance.

Ordering Information

SW–VA08VB–SOS (SignOnce Site License)
SW–VA08VB–SOP (SignOnce Project License)
SW–VA08VB–VHD (VHDL ASIC License)

All licenses include EDIF and VHDL cores. The SignOnce and ASIC licenses allows unlimited instantiations. 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.

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] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Trans. Inform. Theory*, vol. IT–13, pp. 260–269, Apr. 1967.

[2] Third Generation Partnership Project (3GPP), “Universal mobile telecommunications system (UMTS); Multiplexing and channel coding (FDD),” 3GPP TS 25.212 V5.2.0 Release 5, Sep. 2002.

[3] Third Generation Partnership Project 2 (3GPP2), “Physical layer standard for cdma2000 spread spectrum systems, Release D,” 3GPP2 C.S0002–D Version 2.0, Sep. 2005.

[4] Third Generation Partnership Project (3GPP), “Evolved universal terrestrial radio access (E–UTRA); Multiplexing and channel coding,” 3GPP TS 36.212 V8.1.0 Release 8, Nov. 2007.

[5] Consultive Committee for Space Data Systems, “Recommendation for space data system standards: TM Synchronization and channel coding,” CCSDS 131.0–B–1, Blue Book, Sep. 2003.

[6] P. J. Lee, “Further results on rate 1/*N* convolutional code constructions with minimum required SNR criterion,” *IEEE Trans. on Commun*., vol. COM–34, pp. 395–399, Apr. 1986.

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 6 Jun. 2023. First release.
- 1.01 24 Oct. 2023. Updated description of [7].