



SCD02C Features

Turbo Decoder

- 4 state CCSDS compatible serially concatenated convolutional code (SCCC) demapper and turbo decoder
- Code rates from 0.355 to 0.899
- Data lengths from 5758 to 43678 bits
- Interleaver sizes from 8640 to 65520 bits
- Optional QPSK, 8PSK, 16APSK, 32APSK and 64APSK demapping and deinterleaving with 10-bit inphase and quadrature two's complement input
- Optional demapped 8-bit sign magnitude log-likelihood-ratio (LLR) input
- Includes ping-pong input and output memories
- Optional log-MAP or max-log-MAP constituent decoder algorithms
- 60 parallel MAP decoders (1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40 or 120 parallel MAP decoders on request)
- Up to 356 MHz internal clock
- Up to 787 Mbit/s (146 Msym/s) with 10 decoder iterations
- Up to 64 iterations
- Programmable signal points option
- Optional power efficient early stopping
- Optional extrinsic information scaling and limiting
- Estimated channel error output
- Free simulation software
- Available as VHDL core for Xilinx FPGAs under SignOnce IP License. ASIC, Intel/Altera, Lattice and Microsemi/Actel cores available on request.

Introduction

The SCD02C is a fully compatible CCSDS SCCC [1] error control decoder. Nineteen different interleaver sizes from 8640 to 65520 bits are implemented. Twenty seven different code rates R from 0.355 to 0.899 can be selected, giving bandwidth efficiencies from 0.711 to 5.392 bit/sym, excluding header and pilot symbols.

For the code, both the outer and inner encoders use a four state rate 1/2 systematic recursive convolutional code. The outer rate 1/2 code is punctured to a rate of 2/3. The outer four bit tail is

punctured to three bits. The inner rate 1/2 code uses different forms of puncturing for the systematic and parity bits, with the four bit tail not punctured. The interleaver size is of size $I = 1.5K+3$, where K is the number of information bits, which ranges from 5758 to 43678.

The number of coded bits is $8100s$, where s is the number of bits per symbol, ranging from $s = 2$ for QPSK to 6 for 64APSK. To allow continuous decoding, a ping-pong symbol deinterleaver is used to buffer the input data to be decoded. For the turbo decoder, a ping-pong output memory is used to output the decoded data.

$N_p = 60$ MAP02C four state rate 1/2 MAP decoder cores are used with the SCD02CF core, to iteratively decode the SCCC turbo code. The suffix F for the core name indicates the use of 60 MAP decoders. Suffixes 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and G correspond to $N_p = 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40$ or 120, respectively.

The Log-MAP algorithm for maximum performance or the max-log-MAP algorithm for minimum complexity and highest speed can be selected. The extrinsic information can be optionally scaled and limited with each half iteration, improving performance with max-log-MAP decoding.

The reverse sliding window algorithm is used with sliding window lengths of $L = 32$ or 64 for $K_d/N_p > L$ and $L = K_d/N_p$ otherwise, where $K_d = I$ for inner decoding and $K_d = K+2 = 2I/3$ for outer decoding. Eight-bit quantisation is used for near optimum performance.

The turbo decoder can achieve up to 787 Mbit/s with 10 iterations and max-log-MAP decoding using a 356 MHz internal clock ($ACM = 27$, $N_p = 60$). Log-MAP decoding decreases speed by about 24%. Optional early stopping allows the decoder to greatly reduce power consumption with little degradation in performance.

Figure 1 shows the schematic symbol for the SCD02CF turbo decoder. The VHDL cores can be used with Xilinx Integrated Software Environment (ISE) or Vivado software to implement the core in Xilinx FPGA's. VHDL cores are available on request for ASIC, Intel/Altera, Lattice and Microsemi/Actel designs.

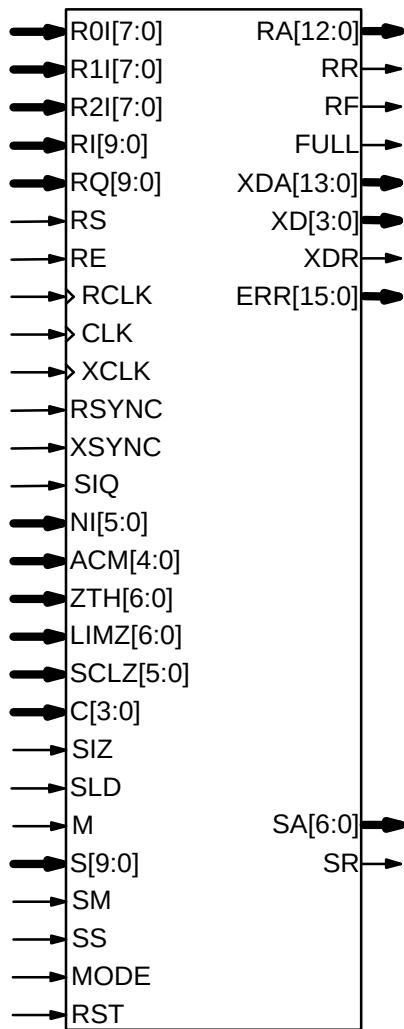


Figure 1: SCD02CF schematic symbol.

Table 1 shows the resources used for Kintex UltraScale+ devices. Resources for Virtex-5, Virtex-6, Spartan-6, 7-Series and UltraScale devices are similar to that for UltraScale+. The MODE input can be used to select various decoder implementations. Three clocks are used for the input symbols, turbo decoder and decoded data. No other resources are used.

Table 1: Resources used.

MAPs (Np)	LUTs Max-log-MAP	LUTs Log-MAP	18KB BRAMs
60	89,297	116,074	180

Table 2 shows the PCD02CF performance achieved with various Xilinx parts and $K = 43678$ ($ACM = 27$). 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 program-

mable logic equivalent to Artix-7 and Kintex-7 devices, respectively.

Table 2: Performance of Xilinx parts.

Xilinx Part	T_{cp} (ns)	f (MHz)	f_d (Mbit/s)
XC7A200T-1	9.042	110.6	244.7
XC7A200T-2	7.484	133.6	295.6
XC7A200T-3	6.609	151.3	334.8
XC7K160T-1	6.513	153.5	339.7
XC7K160T-2	5.466	182.9	404.8
XC7K160T-3	4.978	200.9	444.4
XCKU035-1	5.331	187.6	415.0
XCKU035-2	4.518	221.3	489.7
XCKU035-3	3.828	261.2	578.0
XCKU3P-1	3.536	282.8	625.7
XCKU3P-2	3.059	326.9	723.2
XCKU3P-3	2.809	356.0	787.6

MODE = 0, $l_d = 10$, ACM = 27, $L = 32$, $N_p = 60$

Signal Descriptions

ACM	Advanced Coding and Modulation select (1 to 27)
C	MAP Decoder Constant (0–11)
CLK	System Clock
ERR	Estimated LLR Channel Error (0 to 8100m)
FULL	Decoder Full (new data not accepted)
LIMZ	Extrinsic Information Limit (1–127).
M	Early Stopping Mode 0 = no early stopping 1 = early stopping
MODE	Implementation Mode 0 = Max-log-MAP 1 = Log-MAP
NI	Number of Iterations Minus One (0–63) $NI = l_d - 1$ where l_d is number of iterations
R0I	First Received LLR Data
R1I	Second Received LLR Data
R2I	Third Received LLR Data
RA	Received Data Address
RCLK	Received Data Clock
RE	Received Data Enable
RF	Received Data Finish
RI	Received Descrambled Inphase Data
RQ	Received Descrambled Quadrature Data
RR	Received Data Ready
RS	Received Data Start
RSYNC	RCLK and CLK equal
RST	Synchronous Reset
S	Symbol Value (10-bit)
SA	Symbol Address
SCLZ	Extrinsic Information Scale (1–32).

SIQ	Inphase and Quadrature Select 0 = No Demapping (LLR data input to R0I to R2I) 1 = Demapping (data input to RI and RQ)
SIZ	Internal SCLZ and LIMZ Select 0 = External SCLZ, LIMZ 1 = Internal SCLZ, LIMZ if MODE=0
SLD	Sliding window select 0 = small window ($L_{\max} = 32$) 1 = large window ($L_{\max} = 64$)
SM	Symbol Memory Select 0 = ROM 1 = RAM
SR	Symbol Ready
SS	Symbol Start
XCLK	Decoded Data Clock
XD	Decoded Data (4 bits, last symbol 2 bits)
XDA	Decoded Data Address
XDR	Decoded Data Ready
XSYNC	XCLK and CLK equal
ZTH	Early Stopping Threshold (1–127)

The MODE input is used to select various decoder implementations. Note that MODE is a “soft” input and should not be connected to input pins or logic. This input is designed to minimise decoder complexity for the configuration selected.

Table 3 lists the 27 different modulation and coding combinations selected by ACM. The information data length is given by K , the encoder interleaver size by I and the number of coded bits by N . The number of two dimensional modulated symbols is $N_s = 8100$. $R_{\text{eff}} = K/N_s$ is the bandwidth efficiency in bit/sym (frame marker, frame descriptor and pilot symbols not included). E_b/N_0 is the energy per bit to single sided noise density ratio for an additive white Gaussian noise (AWGN) channel at a bit error rate (BER) of 10^{-5} . E_s/N_0 is the energy per two dimensional symbol to single sided noise density ratio for an AWGN channel at a frame error rate (FER) of 10^{-3} . A is the no-noise average signal amplitude value which the input must be scaled to when demapping is used. SS is the selection parameter used to select the signal set. The value SH is the number of bits that the squared Euclidean distance of a received point is shifted to the right. This value is then divided by 2 and rounded to the nearest integer. C is the MAP decoder constant used for log-MAP decoding (SCLZ = 32). SCLZ and LIMZ are the extrinsic information scale factor and limiting factor used for max-log-MAP decoding ($C = 0$).

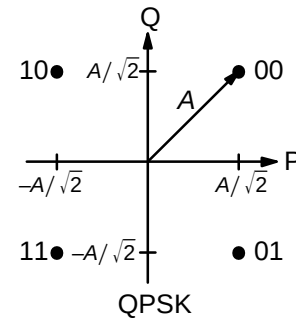


Figure 2: QPSK signal set.

Turbo Decoder Parameters

For optimal performance, the maximum a posteriori (MAP) [2] constituent decoder is used which is dependent on the signal to noise ratio (SNR). Unlike other turbo decoders with suboptimum soft-in-soft-in (SISO) decoders, using the MAP (or specifically the log-MAP [3]) algorithm can provide up to 0.5 dB coding gain at low SNRs. Log-MAP operation is enabled when MODE is high.

With quadrature phase shift keying (QPSK, $s = 2$) modulation (see Figure 2) the decoder constant C should be adjusted such that

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

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

$$\sigma^2 = 1/(4RE_b/N_0). \quad (2)$$

E_b/N_0 is the energy per bit to single sided noise density ratio and $R = K/N$. C should be rounded down to the nearest integer and limited to be no higher than 11. Max-log-MAP [3] operation occurs when $C = 0$. Due to quantisation effects, $C = 1$ is equivalent to $C = 0$. Thus $C = 1$ is internally rounded down to $C = 0$. Also, as $C = 2$ or 3 give poor performance due to quantisation effects, these values are internally rounded up to $C = 4$. Max-Log-MAP operation is also enabled when MODE is low.

For each code (with a particular block size, rate and number of iterations), there will be a minimum E_b/N_0 where the maximum acceptable BER or FER is achieved. The value of C should be chosen for this E_b/N_0 . This value of C can be kept constant for all E_b/N_0 values for this code. For higher values of E_b/N_0 , there will be negligible degradation in performance, even though C will be higher than optimal [4]. For lower E_b/N_0 values, there

could be up to a few tenths of a dB degradation, since C will be lower than optimal. However, this should not have much impact since the BER or FER will already be above the maximum acceptable level anyway.

The value of A directly corresponds to the 8-bit signed magnitude inputs (shown in Table 4). There are 255 quantisation regions with a central dead zone. They are labelled from -127 to +127.

Table 3: ACM Coding

ACM	Mod	K	I	E = S-2	N	R _{eff} (bit/sym)	E _b /N ₀ (dB)	E _s /N ₀ (dB)	A	SS	SH	SCLZI/LIMZ
1	QPSK	5758	8640	8640	16200	0.71086	0.98	-0.49	262	0	0	22/96
2	QPSK	6958	10440	10440	16200	0.85901	1.12	0.49	207	0	0	23/96
3	QPSK	8398	12600	11508	16200	1.03679	1.50	1.70	221	0	0	23/96
4	QPSK	9838	14760	12349	16200	1.21457	1.90	2.78	335	0	0	23/96
5	QPSK	11278	16920	13198	16200	1.39235	2.42	3.90	514	0	0	23/96
6	QPSK	13198	19800	14388	16200	1.62938	3.33	5.48	618	0	0	22/127
7	8PSK	11278	16920	16468	24300	1.39235	2.47	3.96	272	1	8	24/127
8	8PSK	13198	19800	15840	24300	1.62938	3.02	5.20	236	1	7	23/127
9	8PSK	14878	22320	18600	24300	1.83679	3.57	6.28	238	1	7	23/127
10	8PSK	17038	25560	19937	24300	2.10346	4.50	7.79	274	1	7	24/127
11	8PSK	19198	28800	21216	24300	2.37012	5.44	9.24	122	1	4	23/127
12	8PSK	21358	32040	22855	24300	2.63679	6.74	10.98	203	1	5	22/127
13	16APSK	19198	28800	24480	32400	2.37012	4.51	8.31	313	2	7	23/127
14	16APSK	21358	32040	25739	32400	2.63679	5.08	9.34	178	2	5	23/127
15	16APSK	23518	35280	27049	32400	2.90346	5.74	10.42	205	3	5	23/127
16	16APSK	25918	38880	28513	32400	3.19975	6.52	11.62	221	4	5	22/127
17	16APSK	28318	42480	29878	32400	3.49605	7.59	13.07	262	5	5	20/127
18	32APSK	25918	38880	31753	40500	3.19975	6.43	11.55	217	6	5	23/127
19	32APSK	28318	42480	33135	40500	3.49605	7.08	12.58	172	6	4	24/127
20	32APSK	30958	46440	34675	40500	3.82198	7.72	13.60	205	6	4	23/127
21	32APSK	33358	50040	36195	40500	4.11827	8.48	14.69	320	7	5	23/127
22	32APSK	35998	54000	37800	40500	4.44420	9.57	16.08	261	8	4	21/127
23	64APSK	33358	50040	39364	48600	4.11827	8.63	14.86	166	9	3	23/127
24	64APSK	35998	54000	41040	48600	4.44420	9.30	15.85	248	9	4	23/127
25	64APSK	38638	57960	42505	48600	4.77012	10.08	16.92	293	9	4	23/127
26	64APSK	41038	61560	43913	48600	5.06642	10.82	17.92	341	9	4	22/127
27	64APSK	43678	65520	45427	48600	5.39235	11.79	19.20	382	9	4	23/127

For example, one could have $A = 65.5$. The value of $A/\sqrt{2} = 46.32$ lies in quantisation region 46 (which has a range between 45.5 and 46.5).

Since most analogue to digital (A/D) converters do not have a central dead zone, a 9-bit A/D should be used and then converted to 8-bit as shown in the table. This allows maximum performance to be achieved. For signed magnitude inputs a decimal value of 128 has a magnitude of 0 (equivalent to -0).

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 3-bit received data $R0T[2:0]$, where $R0T[2]$ is the sign bit, we have $R0I[7:5] = R0T[2:0]$ and $R0I[4:0] = 16$ in decimal (10000 in binary). For punctured input data, all bits must be zero, e.g., $R1I[7:0] = 0$.

Example 1: Rate 0.35543 QPSK code operating at $E_b/N_0 = 0.3$ dB. From (2) we have $\sigma^2 = 0.65642$. Assuming $A = 13$ we have from (1) that $C = 6.034$ which is rounded down to 6.

Due to quantisation and limiting effects the value of A should also be adjusted according to the received signal to noise ratio. Where QPSK modulation is used in Table 3 with SIQ = 0, the value of A should be divided by $16\sqrt{2}$.

Table 4: Quantisation for R0I[7:0] to R2I[7:0]

Decimal	Binary	Range
127	01111111	$126.5 \leftrightarrow \infty$
126	01111110	$125.5 \leftrightarrow 126.5$
⋮	⋮	⋮
2	00000010	$1.5 \leftrightarrow 2.5$
1	00000001	$0.5 \leftrightarrow 1.5$
0	00000000	$-0.5 \leftrightarrow 0.5$
129	10000001	$-1.5 \leftrightarrow -0.5$
130	10000010	$-2.5 \leftrightarrow -1.5$
⋮	⋮	⋮
254	11111110	$-126.5 \leftrightarrow -125.5$
255	11111111	$-\infty \leftrightarrow -126.5$

Demapping

For quadrature phase shift keying (QPSK), eight phase shift keying (8PSK) and amplitude phase shift keying (APSK) the two-dimensional (2-D) received signal is modelled by the inphase r_j^I and quadrature r_j^Q received values at symbol index j

$$r_j^I = A(s_j^I + n_j^I) \quad (3)$$

$$r_j^Q = A(s_j^Q + n_j^Q) \quad (4)$$

where s_j^I and s_j^Q are the inphase and quadrature components of the transmitted signal which has an average energy of 1 and n_j^I and n_j^Q are the inphase inphase and quadrature AWGN, each with normalised variance

$$\sigma^2 = 1/(2R_{\text{eff}}E_b/N_0) \quad (5)$$

where s is the number of bits in each 2-D signal set. Note that the number of coded symbols $N_s = N/s$, where s is the number of modulated bits per 2-D symbol, e.g., $s = 2$ for QPSK, $s = 3$ for 8PSK, etc.

For the descrambled inphase RI and quadrature RQ sampled values, 10-bit two's complement quantisation is used. Table 5 gives the quantisation for RI and RQ.

For best performance with QPSK we recommend using the values of A given in Table 3. For all other signal sets, the value of A in Table 3 must be used, in order for the demapper to work correctly. These values of A were obtained experimentally in order to obtain the best performance.

Figures 3 and 4 show the signal sets within the 10-bit quantisation regions and recommended signal amplitude for 16APSK and 32APSK, using the values of A for ACM = 13 and 18, respectively.

Table 5: Quantisation for RI and RQ.

Decimal	Binary	Range
511	0111111111	$510.5 \leftrightarrow \infty$
510	0111111110	$509.5 \leftrightarrow 510.5$
⋮	⋮	⋮
2	0000000010	$1.5 \leftrightarrow 2.5$
1	0000000001	$0.5 \leftrightarrow 1.5$
0	0000000000	$-0.5 \leftrightarrow 0.5$
-1	1111111111	$-1.5 \leftrightarrow -0.5$
-2	1111111110	$-2.5 \leftrightarrow -1.5$
⋮	⋮	⋮
-511	1000000001	$-511.5 \leftrightarrow -510.5$
-512	1000000000	$-\infty \leftrightarrow -511.5$

Table 6 gives the signal set as selected by the internal parameter SS, as given in Table 3. The parameter $\gamma_i = R_{i+1}/R_1$ gives the ratio of the outer ring radius to the inner most ring radius. The signal set mappings and phases are given in [1].

Table 6: Signal Sets

SS	Signal Set	γ_1	γ_2	γ_3
0	QPSK			
1	8PSK			
2	16APSK	3.15		
3	16APSK	2.85		
4	16APSK	2.75		
5	16APSK	2.60		
6	32APSK	2.84	5.27	
7	32APSK	2.72	4.87	
8	32APSK	2.54	4.33	
9	64APSK	2.73	4.52	6.31

To demap the received 8PSK or APSK signals into individual log-likelihood ratios (LLR) for each modulated bit, the optimum equation is

$$L_i = -\log \left(\frac{\sum_{h=0}^{S-1} h(i) \exp(-|s(h)-r|^2/c)}{\sum_{h=0}^{S-1} \bar{h}(i) \exp(-|s(h)-r|^2/c)} \right) \quad (6)$$

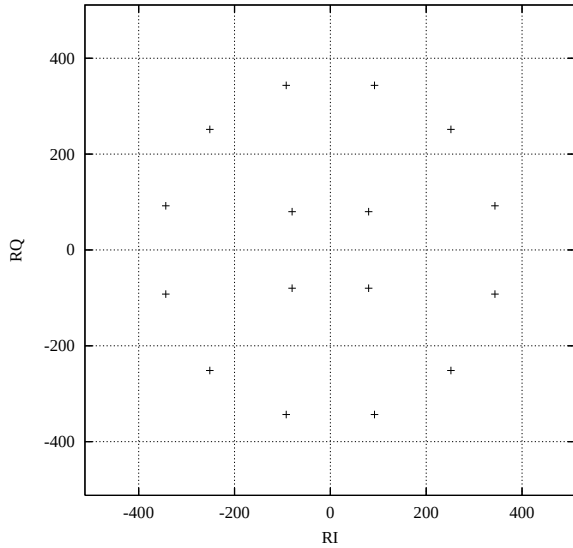


Figure 3: 16APSK ACM = 13 Signal Set

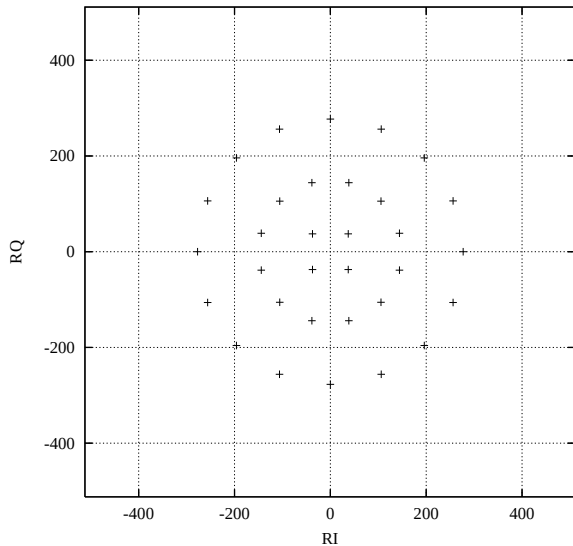


Figure 4: 32APSK ACM = 18 Signal Set

where i is the modulated bit index, $S = 2^s$ is the number of signal points, $h(i)$ is the i 'th bit of h in binary, $s(h)$ is the h 'th complex signal point, r is the received complex value and $c = 2A^2\sigma^2$. By using $\min^*(x, y) = \min(x, y) - c \ln(1 + \exp(-|x-y|/c))$ (7)

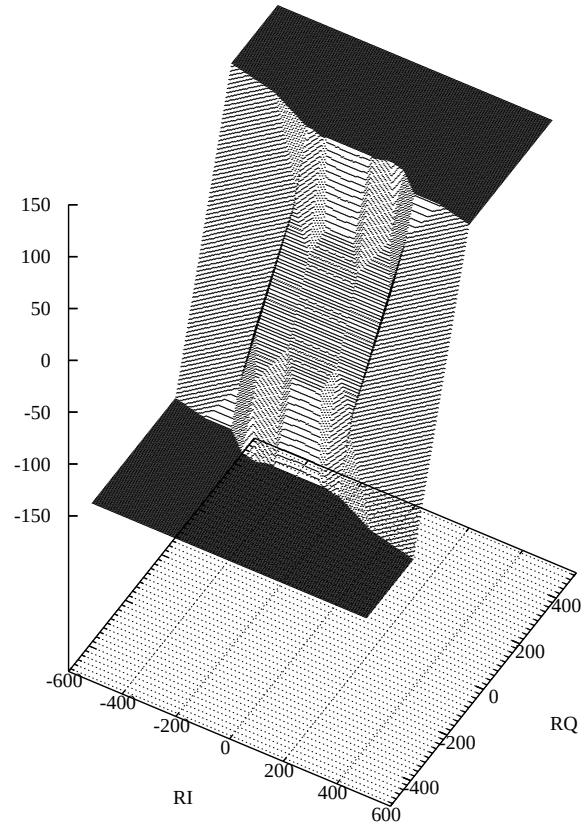
where the Jacobian logarithm is to base $e^{1/c}$, the LLR can be simplified to

$$L_i = \min_{h=0}^{S-1} |s(h)-r|^2/h(i) - \min_{h=0}^{S-1} |s(h)-r|^2/\overline{h(i)}. \quad (8)$$

As the signal to noise ratio is high for APSK modulation, the correction term in (7) becomes negligible. The LLR can thus be simplified to

$$L_i \approx \min_{h=0}^{S-1} |s(h)-r|^2/h(i) - \min_{h=0}^{S-1} |s(h)-r|^2/\overline{h(i)} \quad (9)$$

with little degradation in performance. The LLR is then scaled and rounded to the nearest integer. The result is an eight bit LLR which is deinterleaved for input to the decoder. Figure 5 gives an example plot of L_1 for 32APSK and ACM = 18. In this case, the LLR output was divided by $2^{SH+1} = 64$ and then rounded to the nearest integer.

Figure 5: L_1 for 32APSK ACM = 18

Demapping is selected with SIQ = 1 where data is input to RI[9:0] and RQ[9:0]. For SIQ = 0 and $N_p = 60$ MAP decoders, the demapped LLRs are input to R0I[7:0] to R2I[7:0]. Let $r_i(s-1), \dots, r_i(0)$ correspond to the LLRs of the received symbol at time i , where $r_i(s-1)$ corresponds to the LLR for the left most significant bit and $r_i(0)$ corresponds to the LLR for the right least significant bit. For QPSK and 8PSK each symbol is input in one clock cycle. For 16APSK, 32APSK and 64APSK, each symbol is input in two clock cycles. Table 7 gives the order in which the LLRs are input.

Table 7: LLR input for SIQ = 0.

s	1st clock cycle			2nd clock cycle		
	R2I	R1I	R0I	R2I	R1I	R0I
2	–	$r_i(1)$	$r_i(0)$	–	$r_{i+1}(1)$	$r_{i+1}(0)$
3	$r_i(2)$	$r_i(1)$	$r_i(0)$	$r_{i+1}(2)$	$r_{i+1}(1)$	$r_{i+1}(0)$
4	–	$r_i(3)$	$r_i(2)$	–	$r_i(1)$	$r_i(0)$
5	–	$r_i(4)$	$r_i(3)$	$r_i(2)$	$r_i(1)$	$r_i(0)$
6	$r_i(5)$	$r_i(4)$	$r_i(3)$	$r_i(2)$	$r_i(1)$	$r_i(0)$

Deinterleaving and Depuncturing

The LLRs for each symbol must be deinterleaved and depunctured. The interleaving operation consists of writing the N coded bits into length $N_s = N/s = 8100$ columns and then reading the length s rows [1]. The punctured systematic bits from the inner encoder are written first, followed by the punctured parity bits.

Deinterleaving is performed by writing the LLRs at the calculated interleaved address into the input RAM. With N_p parallel MAP decoders, the pseudo random nature of the puncturing used implies that $2N_p$ input memories are required, as each MAP decoder uses one memory for the systematic data and one memory for the parity data. The tail data is stored in a separate memory.

The depth of each memory is determined by the amount of puncturing of each code used. For $N_p = 60$, the maximum depth for systematic data is 786 for ACM = 27 and the maximum depth for parity data is 154 for ACM = 23. Ping-pong input memories are used where new LLRs are written to one half of the memory, while the other half is used by the turbo decoder. This implies that 60 2Kx8 memories are used for systematic data and 60 512x8 memories are used for parity data. The tail data is stored in a 4x16 memory. Table 8 gives the maximum systematic data depth w_d and parity data depth w_p for each N_p .

Table 8: Maximum input memory depths

N_p	Suf-fix	w_d	w_p	N_p	Suf-fix	w_d	w_p
1	1	45427	9232	12	9	3812	770
2	2	22740	4616	15	A	3054	616
3	3	15162	3078	20	B	2302	462
4	4	11390	2308	24	C	1926	385
5	5	9115	1847	30	D	1540	308
6	6	7582	1539	40	E	1174	231
8	7	5696	1154	60	F	786	154
10	8	4569	924	120	G	402	77

For writing there are 12 separate counters. Six counters (the first) correspond to the address in each memory for up to $s = 6$ LLR inputs. The other six counters (the second) are used to select one of the $2N_p$ input RAMs. Each first and second counter starts at the address where the first data is written. The first counter increments after each LLR. If the end of a RAM is reached, the first counter is reset and the second counter is incremented.

For reading, each MAP decoder has two counters. One counter increments for non-punctured data and holds for punctured data. After L clock cycles, the address is transferred to the second counter, which is used to read the input data. However, instead of incrementing, this counter decrements for punctured data, so that the received data can be read in reversed order, as required by the MAP decoder. Depth L memories are used to reverse in time the puncturing patterns to perform the decrement.

Decoder Speed

The number of turbo decoder iterations is determined by NI, ranging from 0 to 31. $NI = I_d - 1$ where I_d is the number of iterations. This is equivalent to 1 to 32 iterations. The decoder initially starts at iteration 0, increasing by one until NI is reached or an earlier time if early stopping is enabled.

The average decoder speed f_d is given by

$$f_d = \frac{F_c K}{I_d(5I/(3N_p) + 2L + 2L_o + 23) + L} \quad (10)$$

where F_c is the CLK frequency, K is the input data length, I_d is the number of iterations (equal to $NI+1$), I is the interleaver length, N_p is the number of parallel MAP decoders, L is the sliding window length (32 or 64), and L_o is the sliding window length for the outer decoder. For $N_p < 120$, $L_o = L$, since $(K+m)/N_p \geq 96$. For $N_p = 120$, $L_o = 32$ and 48 for SLD = 0 and 1, respectively.

For the inner decoder, five cycles are required before MAP decoding can begin, due to the pipeline delay in calculating the punctured read addresses, reading the input memory, depuncturing and calculating the LLRs for input to the MAP decoder. For the first iteration, another L cycles are required to calculate the reversed start address. The MAP decoder has a delay of $2L+7$ cycles where seven cycles are from pipeline delay. By using the reverse sliding window algorithm, the delay is reduced from $4L+7$ cycles, increasing decoder speed. The output from the MAP decoder takes a further I/N_p cycles.

For the outer decoder, two cycles are required before MAP decoding can begin, due to reading the input memory and depuncturing for input to the MAP decoder. The MAP decoder has a delay of $2L_o + 7$ cycles. The output from the MAP decoder takes a further $2l/(3N_p) = (K+m)/N_p$ cycles.

Table 9 gives the decoder speed with $F_c = 100$ MHz, $l_d = 10$ and $N_p = 60$ for SLD = 0 and 1. Table 10 gives the decoder speed with $F_c = 100$ MHz, $l_d = 10$ and ACM = 1 with SLD = 1 and ACM = 27 with SLD = 0.

The symbol rate is given by

$$f_s = \frac{F_c N_s}{l_d(5l/(3N_p) + 2L + 2L_o + 23) + L} \quad (11)$$

where N_s is the number of symbols per codeword. For the SCCC standard this includes 8100 coded symbols with an optional 120 pilot symbols. This gives $N_s = 8100$ or 8220 with pilot symbols. Table 11 gives the maximum symbol rate with $F_c = 100$ MHz and $l_d = 10$ with ACM = 27, $N_s = 8100$ and $L = 64$ (Low) or ACM = 1, $N_s = 8220$ and $L = 32$ (High).

Table 9: Maximum decoder speed with $F_c = 100$ MHz, $l_d = 10$ and $N_p = 60$

ACM	f_d (Mbit/s)		ACM	f_d (Mbit/s)	
	$L=32$	$L=64$		$L=32$	$L=64$
1	146.1	109.6	15	207.4	185.9
2	156.6	120.9	16,18	210.0	189.8
3	166.6	132.2	17,19	212.2	193.2
4	174.4	141.5	20	214.4	196.5
5,7	180.7	149.3	21,23	216.0	199.1
6,8	187.4	158.0	22,24	217.6	201.6
9	192.2	164.3	25	219.0	203.9
10	197.2	171.2	26	220.1	205.7
11,13	201.2	176.9	27	221.2	207.5
12,14	204.5	181.7			

There are two decoder operation modes given by M. Mode M = 0 decodes a received block with a fixed number of iterations (given by NI). Mode M = 1 uses an early stopping algorithm. Early stopping is used to stop the decoder from iterating further once all the inner MAP decoder output systematic data LLR magnitudes have exceeded the threshold ZTH. After the threshold has been exceeded one extra outer decoding is performed.

Decoder Delay

The decoder delay can be separated into three parts. This is the input memory delay T_i , turbo decoder delay T_d and the output memory delay T_o . Each delay is equal to

Table 10: Maximum decoder speed with $F_c = 100$ MHz, $l_d = 10$, Low (ACM=1, SLD=1) and High (ACM=27, SLD=0)

N_p	f_d (Mbit/s)		N_p	f_d (Mbit/s)	
	Low	High		Low	High
1	3.9	4.0	12	38.8	47.2
2	7.7	8.0	15	46.2	58.8
3	11.3	11.9	20	57.3	77.8
4	14.8	15.9	24	65.0	92.8
5	18.2	19.9	30	75.2	115.1
6	21.4	23.8	40	89.2	151.4
8	27.6	31.6	60	109.6	221.2
10	33.4	39.4	120	154.2	410.4

Table 11: Maximum symbol speed with $F_c = 100$ MHz, $l_d = 10$, Low (ACM=27, SLD=1, $N_s = 8100$) and High (ACM=1, SLD=0, $N_s = 8220$)

N_p	f_s (MHz)		N_p	f_s (MHz)	
	Low	High		Low	High
1	0.7	5.6	12	8.6	60.7
2	1.5	11.2	15	10.7	73.8
3	2.2	16.6	20	14.1	94.0
4	2.9	21.9	24	16.8	109.0
5	3.7	27.1	30	20.6	129.6
6	4.4	32.2	40	26.9	159.9
8	5.8	42.1	60	38.5	208.5
10	7.2	51.6	120	67.8	299.8

$$T_i = (12S_q + 8100S_n + 2)T_r \quad (12)$$

$$T_d = \frac{(l_d(5l/(3N_p) + 2L + 2L_o + 23) + L + 1 - S_r)T_c}{1 - S_r} \quad (13)$$

$$T_o = (4 - 2S_x)T_x \quad (14)$$

where T_r is the RCLK period, $S_q = \text{SIQ}$, $S_n = 1$ for QPSK and 8PSK and 2 otherwise, $S_r = \text{RSYNC}$, T_c is the CLK period, T_x is the XCLK period and $S_x = \text{XSYNC}$.

The above delays assume that RE is high in the minimum time, no early stopping is used (M = 0) and that the decoder parameters do not change between decoded blocks. When decoding multiple blocks with varying code parameters, the delay is more difficult to predict.

When RSYNC is low the actual turbo decoder delay will vary from $T_d - T_c$ to T_d . Similarly, when XSYNC is low the actual output delay will vary from $T_o - T_x$ to T_o .

Demapper Operation

The demapper takes the RI and RQ values and calculates the approximate LLR values for each bit in the signal set. Since each ACM has a different value of A , excluding QPSK, there are a total of 21 signal sets that are stored in a lookup table (LUT). The complexity of the demapper is determined by the largest size signal set, which is 64APSK. In order to calculate six LLR values in two cycles with pipelining, 32 parallel squared Euclidean distance (SED) calculators are required. This allows the calculation of $2 \times 32 = 64$ SED values.

Since each 2-D signal point is represented by a 20-bit value, the LUT has a $32 \times 20 = 640$ bit output. The total number of signal points that need to be stored is 608. With ideal packing, the address space is $608/32 = 19$. However, to reduce complexity the 8PSK and 16APSK signal sets are stored once per address, instead of four and two times, respectively. This gives a total of 26 address locations. Two address locations of the depth 28 memory are used to store the program-mable signal set values.

Each 20-bit SED value is scaled and rounded to a 14-bit value using the value of SH. A total of 154 MIN circuits are used to find the six minimum distances (for the three LLRs) used to calculate the LLR. Each MIN has two 14-bit inputs, finding the minimum value of the inputs. By swapping some of the inputs and use of pipelining, the LLR calculation can be efficiently performed. The last six min values are subtracted and limited to give the LLR output.

For QPSK, RI and RQ are divided by 16 and rounded to the nearest integer.

Turbo Decoder Operation

Figure 6 gives a simplified block diagram of the SCD02CF four state turbo decoder with $N_p = 60$. The inner decoder data length is $W_i = I/N_p = 2W$, where $W = I/120$. Each memory is implemented using a number of smaller memories that cover the various code options. The input and output memories have two halves and use simple dual port RAMs with separate read and write clocks. One half is used for writing new data while simultaneously reading the previously written data. The input RAM writes data at RCLK and reads data at CLK. The output RAM writes data at CLK and

reads at XCLK. The turbo decoder operates at CLK.

The received or demapped input data are written 16 or 24 bits at a time into one half of the two input RAMs, one with $N_p \cdot 2w_d \times 8$ memories for the outer encoder data and one with $N_p \cdot 2w_p \times 8$ memories for the outer encoder parity. That is, we divide the data and parity LLRs into N_p memories each, each containing the non-punctured data of the inner encoded data. Due to the pseudo random puncturing used, each of these memories can have different depths, with the maximum values being w_d for data and w_p for parity.

For 64APSK, we have six address counters, one for each column in the symbol interleaver. For other signal sets, we use m counters, where the number of signal points is 2^m . Each of these counters has another selection counter, used to select one of the $2N_p$ memories. These counters are initialised with the locations of the first m symbols to be written into the RAMs. When the address counter reaches the maximum width given by the selection counter, the selection counter is incremented.

For $N_p = 60$, the maximum width of the data and parity RAMs is $w_d = 786$. This implies each of the m LLRs has a 10-bit address counter and a 7-bit selection counter. For the data RAMs all six columns can contain data, so six 10-bit address plus 1 bit for the input half are used. For the parity RAMs up to two columns contain parity and with $w_p = 154$, only two 8-bit address plus 1 bit for the input half are used.

For inner decoding, each of the $2N_p$ input RAMs has a separate counter to read the non-punctured data. Due to the MAP decoder algorithm used, this data is read in reverse order. If no puncturing is used the initial address is $L-1$, which counts down to 0 and then jumps to $2L-1$, repeating the decrement every cycle and jump every L cycles. With puncturing, a separate counter is used which increments for every non-punctured symbol. This is used to initialise the reverse counter every L cycles.

Thus, on the first iteration, we need to wait for L cycles before MAP decoding can start in order to initialise the reverse counters. After $W_m = L \lceil W_i/L \rceil$ cycles, the forward counters are reset and count for L cycles to calculate the initial values for the next iteration. Separate $L \times N_p$ RAMs are used to reverse in time the puncturing pattern for the read address counters. The reverse puncturing patterns are also used to puncture the re-

$j = 1$, we can see that if $\beta(k)$ is even ($m(k) = 0$), then the rotation is $d(k)$. For $\beta(k)$ odd ($m(k) = 1$), the rotation is $(1+d(k)) \bmod N_p$. In general, we have the rotation S for the outer encoder extrinsic information as

$$c(Wj+k) = (((j+m(k)) \bmod b) + d(k)) \bmod N_p. \quad (25)$$

That is, we have a cyclic rotation of $d(k)$ if $(j+m(k)) \bmod b = 0$ or $(1+d(k)) \bmod N_p$ if $(j+m(k)) \bmod b = 1$.

Similarly, for the address we have

$$a(Wj+k) = W((j+m(k)) \bmod b) + \alpha(k). \quad (26)$$

If b is a power of 2, we can use $\beta(k)$ without modification, otherwise we store $m(k)$ and $d(k)$.

The MAP decoder produces the extrinsic information for both the data (ZD) and parity (ZP) bits. For inner decoding, we need to add ZD to the demapped data LLRs and then derotate the N_p values. The derotation value SD is equal to 0 if $c(i) = 0$, otherwise it is equal to $N_p - c(i)$. After derotation, the values are stored in one of the two interleaver memories determined by signals ZWEP and ZWED, which is the decoded value of $s(i) \bmod 2$.

To perform outer decoding, we have $NA0 = 1$. The parity information stored in the parity interleaver RAM is read in non-interleaved order. Signal PIP is used to depuncture the parity data, with every even symbol being punctured.

The data information is read from the data interleaver in non-interleaved order with $ZS = 0$ and $S = 0$. We also need $DP = 0$ as the received data LLRs were already added during inner decoding.

The data and parity a priori information is sent to the MAP decoders where they are decoded. After a delay of $2L+7$ cycles, the data and parity extrinsic information are stored in their respective interleaver memories. For the data information, the other data input to the output adder is forced low. The output cyler has $SD = 0$. The extrinsic data is written with $ZWED = 1$. The extrinsic parity is written with $ZWEP$ being high or low, for even or odd symbols, respectively.

The decoding process then repeats until the last iteration, where the N_p decoded bits (equal to the sign bits of the calculated LLRs) are stored in one half of the output memory. With $N_p = 60$, to maintain a sufficient decoding speed, the decoded data is read out four bits at a time in sequential order, except for the last output, where two bits are output since $K \bmod 4 = 2$.

As $W_o = 2W_i/3 = 2bW/3$ with $b = 2$, this implies that W_o is also divisible by four. Thus, we split the output memory into four $W_i/3 \times N_p$ memories (the total depth is $2W_o/4$ for simultaneous write and read) where data for address i is stored in memory

$i \bmod 4$. Ideally, size 182x60 memories should be used with $N_p = 60$. However, the simple dual port BlockRAM memories available with Xilinx have size 512x32. The input port is configured as 512x32, with the output port as 8Kx2. A total of eight 18K BlockRAM memories are used. All four memories are simultaneously read with four two input multiplexers used to output the four decoded bits.

MAP Decoder Operation

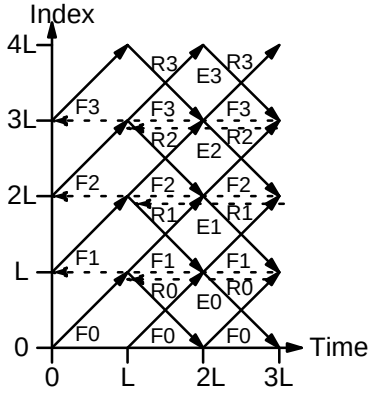
The MAP decoders are designed to accept input and output data in forward or reversed blocks of L . This reduces the MAP decoder delay by $2L$ (instead of $4L$ as in continuous MAP decoders).

The last forward and reverse state metrics (SM) are also stored in two separate registers for inner and outer decoder iterations. The stored forward SMs are used as the starting states in the following MAP decoder at the next full iteration. Similarly, the stored reverse SMs are used by the previous MAP decoder in the next full iteration. This is necessary as for small K , the sliding window size is small ($L = 48$ for $K+m = 5760$ and $N_p = 120$), implying that the SMs do not become reliable after starting from all zero values. By using the SMs from the previous iteration as the starting values, this gives a more reliable decoded output.

The sliding window algorithm used depends on L . Let W_d be the data length for each MAP decoder. We have $W_d = W_i = L/N_p$ for inner decoding and $W_d = W_o = (K+m)/N_p$ for outer decoding. If $W_d < 32(\text{SLD}+1)$ we have $L = W_d$ else $L = 32(\text{SLD}+1)$.

For $L = W_d$, Figure 7 shows how the forward and reverse state metrics (SM) are calculated with $N_p = 4$. For the SCCC code, this only occurs for $ACM = 1$, $N_p = 120$ and outer decoding where $L = 48$. The horizontal axis shows decoder time and the vertical axis shows the received symbol index. An arrow going up shows the forward SMs for L symbols. An arrow going down shows the reverse SMs for L symbols. A horizontal dashed arrow indicates SMs being passed between iterations. Forward SMs are indicated by an F and reverse SMs by R. Forward SMs that have been reversed in time are indicated by E. Decoded data is output when R and E are used together.

We see that in this case the RAMs are read in forward blocks of L . Decoded data is written in reversed blocks of L . Also, the SMs are allowed to settle for L symbols before being used. To help improve the reliability of the SMs, we pass both forward and reverse SMs between iterations. As the outer code starts and ends in state 0, F0 and RN-

Figure 7: Timing diagram for $N_p = 4$, $L = W_d$.

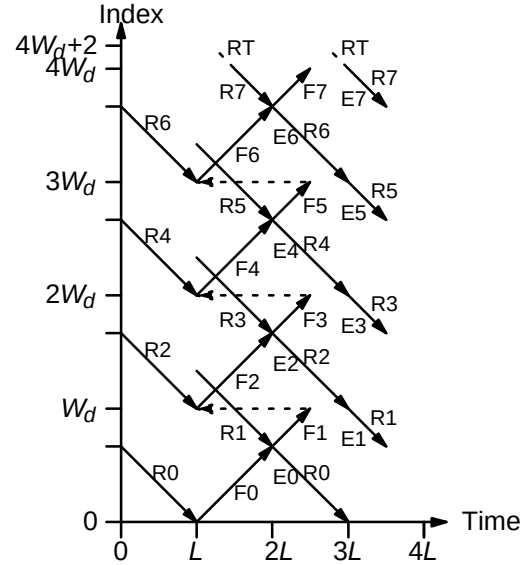
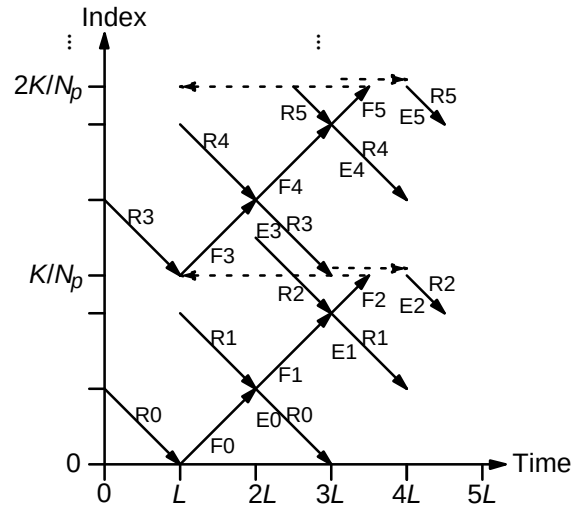
$p-1$ starts with $SM = 0$ for state 0 and $SM = 255$ (the maximum value with 8-bit SMs) for the other states.

Figures 8 to 10 show the timing diagram for various cases where $W_d > L$. Figures 9 to 10 only show the trellis diagrams for the first two MAP decoders. Unlike the previous case, we read the SMs in reverse blocks of L and output them in reverse order. We pass forward SMs between iterations so that the starting forward SMs become reliable. This is indicated by the dashed arrow going backwards in time. As the reverse state metrics are reliable when used to calculate the extrinsic information there is no need to pass reverse SMs between iterations. For MAP decoder N_p-1 , the reversed tail symbols RT (starting from state 0) are separately input for the inner decoder only, as shown in Figure 8. For the outer decoder, the tail symbols are included within R_{N_p-1} which also starts from state 0.

For $W_d > 2L$ the final SMs of the third block, for example R3 in Figure 9, are stored and then passed to the previous MAP decoder. This is indicated by the dashed arrow going forwards in time. For $L < W_d \leq 2L$ (Figure 8), the reverse SMs for the second reverse SM calculator are passed directly between MAP decoders.

For $L < W_d \leq 3L$ (Figures 8 and 9), the first reverse SM calculator initially uses input data from the next MAP decoder for the last input block in time. This method avoids additional multiplexers to select the first reverse SMs from the next MAP decoder. However, this technique can not be used for $W_d > 3L$ as the old extrinsic information will have been overwritten with new data. Instead, we use the previously stored reverse SMs as shown by the first forward arrow in Figure 10.

For $W_d > 4L$ the timing diagrams are similar to that for $3L < W_d \leq 4L$ (Figure 10). In all cases, we see that decoding time is equal to $W_d + 2L$. We also need to add an additional 7 clock cycles for pipe-

Figure 8: Timing diagram for $N_p = 4$, $L < W_d \leq 2L$ Figure 9: Timing diagram for $2L < W_d \leq 3L$.

line delay, to give a decoding time of $W_d + 2L + 7$ clock cycles for each half iteration. For inner and outer decoding an additional five and two clock cycles are required to read the input data.

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. 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. We recommend that 96 be used for ACM ≤ 5 and 127 for ACM > 5 .

Another parameter that can be used to adjust decoder performance is SCLZ which ranges from 1

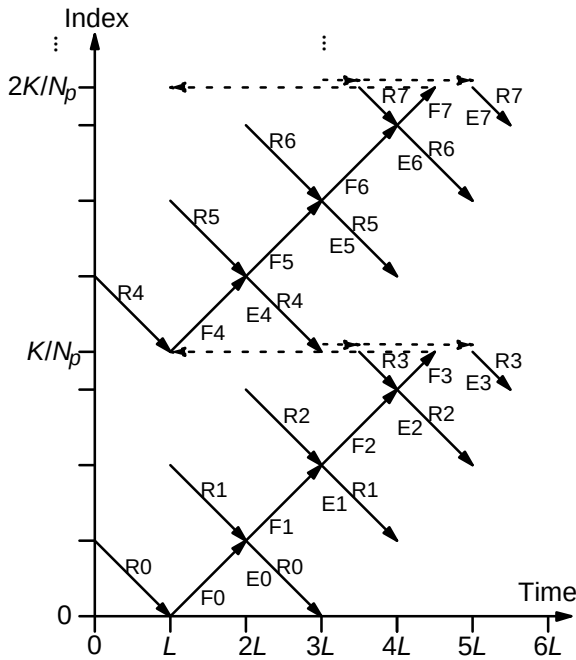


Figure 10: Timing diagram for $3L < W_d \leq 4L$.

to 32. The extrinsic information is scaled by $SCLZ/32$ followed by rounding to the nearest integer. Thus, when $SCLZ = 32$, no scaling is performed. For log-MAP decoding we recommend $SCLZ = 29$. For max-log-MAP decoding we recommend $SCLZ$ values around 23. For $MODE = 0$ (max-log-MAP decoding) and $SIZ = 1$, internal LUTs are used for $SCLZ$ and $LIMZ$ using the values given in Table 3.

Note that input SIZ is not internally pipelined. This parameter can only be changed until after the last received block has been decoded.

There are two decoder operation modes given by M . Mode $M = 0$ decodes a received block with a fixed number of iterations (given by NI). Mode $M = 1$ selects early stopping operation. This is used to stop the decoder from iterating further once it has estimated there are zero errors in the block. Further details are given in the next section.

Input Operation

The $FULL$ output indicates when the decoder can accept data. When high this indicates that new data must not be input to the decoder in the next clock cycle. That is, RS and RE must remain low while $FULL$ is high and any input data must be held.

If $FULL$ is low, the received data start RS signal is used to start the decoder. The received data enable input RE must also go high when RS goes high to read the first received data. RE can only go high once for each received input symbol. This means RE can only be high for $N_s = 8100$ clock cycles.

The received data ready output RR will go high to indicated that RE can now go high so as to read the next input. For QPSK and 8PSK with $N_p = 60$, RR will stay high until one clock cycle before the last data is input. For other signal sets with $N_p = 60$, RR goes high two clock cycles after RE goes high, except after the last RE going high. If RR is high and RE is low, RR will stay high in the next clock cycle. If RR and RE are both high RR will go low in the next cycle. RS and RR can be ORed together to form RE if the input data is stored in an external memory.

Valid data must be input one clock cycle after RE goes high. A received data address output $RA[12:0]$ is provided for reading received data from an external synchronous read input memory. For $SIQ = 1$, data read from the input memory must be held if RE goes low as shown in Figures 11 and 12.

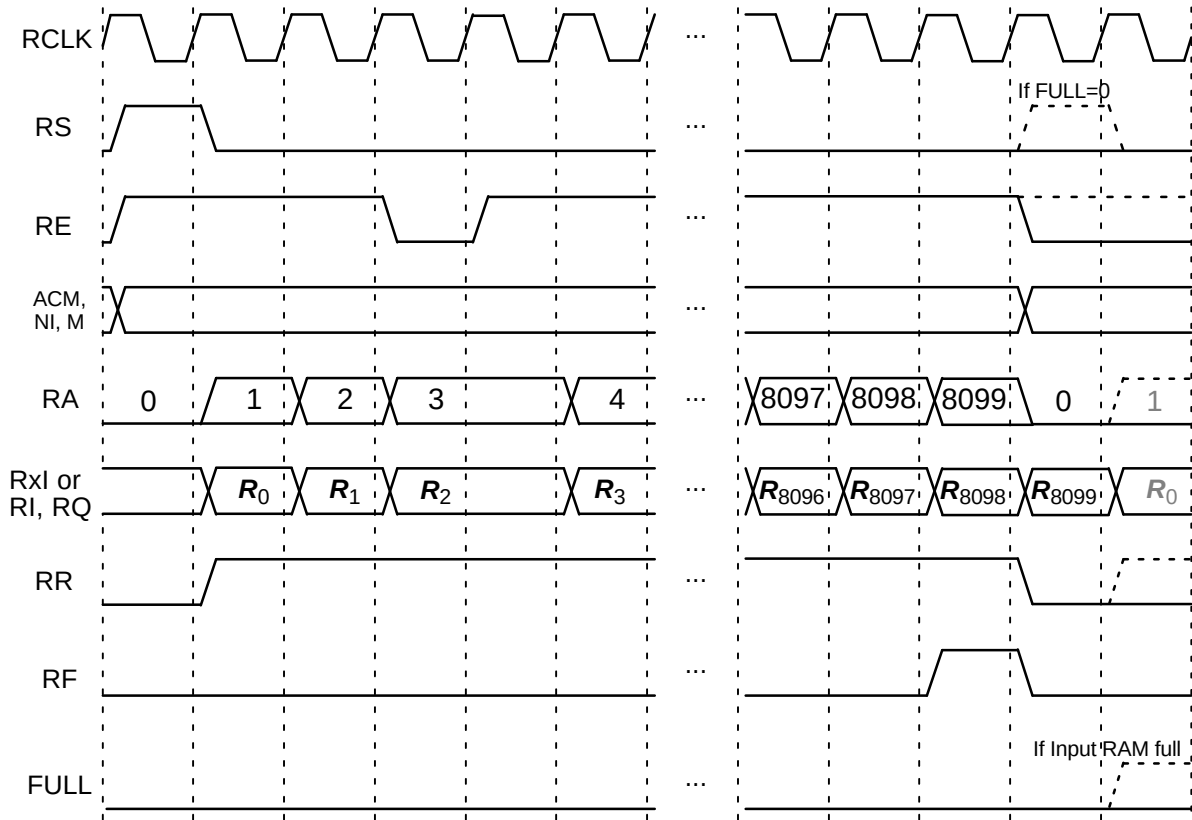
The received data finish output RF will go high at the end of each received block. For QPSK and 8PSK with $N_p = 60$, this occurs when $RA = N_s - 1$. If RE goes low at $RA = N_s - 1$, RF will remain high. RF will go low only after RE goes high. For other signal sets with $N_p = 60$, RF goes high for one clock cycle two clock cycles after RA changes from $N_s - 1$ to 0 at the end of the block.

If the other half of the Input RAM is available, $FULL$ will remain low, indicating that the next block may be input. If both halves of the RAM are full, then $FULL$ will go high. $FULL$ will not go low again until one of the halves of the RAM becomes available. If $FULL$ is low, RS and RE can go high.

Due to pipelining of the input data, $FULL$ can can go high while data is being input. This can occur 2 or 14 clock cycles after RS goes high for $SIQ = 0$ or 1, respectively. If this occurs, RE must be held low and the input data RxI or RI and RQ held while $FULL$ is high. When $FULL$ goes low, data can then be continue to be input.

Inputs $RxI[7:0]$, $RI[9:0]$, $RQ[9:0]$, RS , RE , $NI[5:0]$, $ACM[4:0]$ and M must be synchronous to $RCLK$. Outputs $RA[12:0]$, RR , RF and $FULL$ are synchronous to $RCLK$. Internal turbo decoding uses CLK . If $RCLK$ and CLK are equal to each other in both clock period and phase, then $RSYNC$ can equal 1. This reduces the decoder input time by one clock cycle. If $RCLK$ and CLK are not equal, then $RSYNC$ must equal 0. $RSYNC$ should not be connected to logic or input pins.

The input data can be input in any code order. That is, it is not necessary to wait for the decoder to output the last block of one code before changing to another code. If changing the code, the decoder parameters $NI[5:0]$, $ACM[4:0]$ and M must

Figure 11: Turbo decoder input timing for QPSK and 8PSK ($N_p = 60$).

stay constant from the time RS goes high to until after RF goes high. Note that input SIQ is not internally pipelined. This parameter can only be changed until after the last received block has been decoded.

Figure 11 illustrates the decoder input timing with QPSK or 8PSK and $N_p = 60$. For SIQ = 1, each received value R_j , $0 \leq j \leq N_s - 1$ represents a two-dimensional sample at symbol time j input to RI and RQ. For SIQ = 0, each received value R_j represents two (for QPSK) or three (for 8PSK) LLR samples input to R0I to R2I. RS is shown going high again for the case where FULL = 0.

Figures 12 illustrates the decoder input timing with APSK received data and $N_p = 60$. For SIQ = 0, each received value L_i , $0 \leq i \leq 2N_s - 1$ represents two or three LLR samples at time i input to R0I to R2I. The RCLK frequency must be at least two times the 2-D symbol rate. This is because the demapper calculates two or three LLRs for every RCLK cycle.

Figure 13 illustrates the decoder output timing. After a delay of T_0 clock cycles the signal XDR goes high for $\lceil K/4 \rceil - 1$ XCLK cycles while the block is output. The block is output in sequential order with address XDA[13:0].

Outputs XD[3:0], XDR, XDA[13:0] and ERR[15:0] are synchronous to XCLK. If XCLK and

CLK are equal to each other, i.e., they use the same clock signal, then XSYNC can equal 1. This reduces the decoder output time by two clock cycles. If XCLK and CLK are not equal, then XSYNC must equal 0. XSYNC should not be connected to logic or input pins. Note that for XDA = 0, XD[3] is the first decoded bit.

The output ERR is determined during the last iteration during inner decoding. A comparison between the hard decisions of the received data LLRs and the decoded data XD from the MAP decoders is made. If the magnitude of the received data is zero (including that for punctured data) then the comparison is also punctured.

The total of the number of bits different (varying from 0 to $E = S - 2$, where S is the total number of non-punctured data bits, including the tail) is counted and stored. If ERR = 0, this indicates that no channel errors were detected. A high value of ERR indicates a high channel error rate. The values for E are given in Table 3.

The early stopping algorithm uses the magnitude of the extrinsic information during inner decoding 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

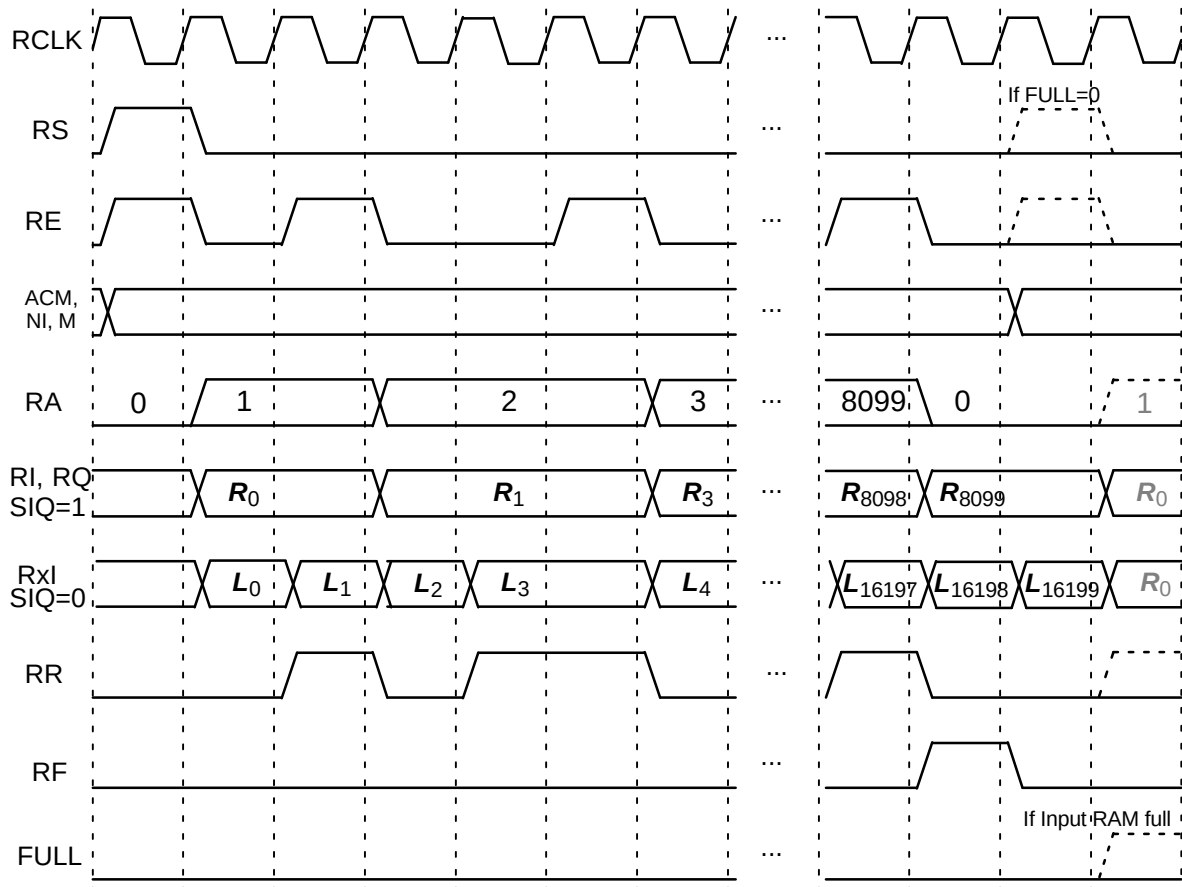


Figure 12: LDPC decoder input timing for 16APSK, 32APSK and 64APSK ($N_p = 60$).

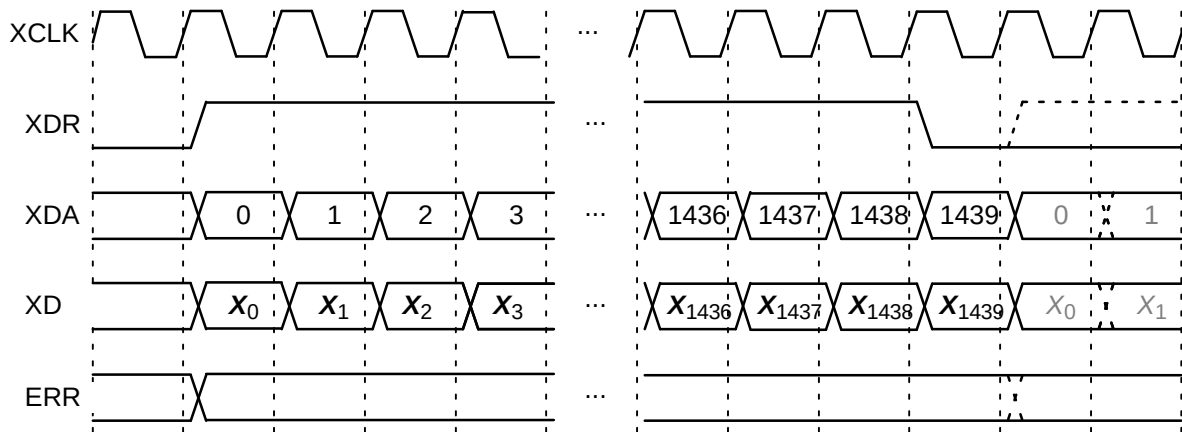


Figure 13: Turbo decoder output timing (ACM = 0, $N_p = 60$).

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.

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.

APSK Signal Set Programming

Each amplitude level in the APSK signal set can change amplitude and phase due to the effect

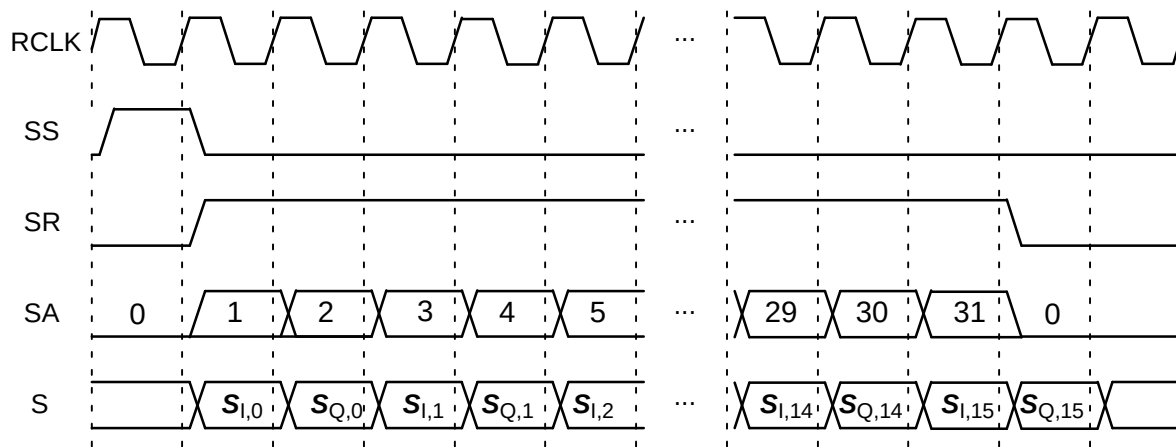


Figure 14: 16APSK signal set programming.

of non-linear amplifiers. Thus, the SCD02C core is designed so that the signal set points can be programmed before the demapper can be used.

To start the programming sequence, SS should go high for one clock cycle. The address SA[7:0] will count from 0 to $2S-1$, where S is the number of signal points. SR will go high for $2S-1$ clock cycles. The symbol data is input in alternating 10-bit two's complement values to S[9:0], with the inphase value first followed by the quadrature value. The external LUT or RAM should be read synchronously. Figure 14 shows the programming sequence for 16APSK.

Input SM = 0 can be used to select the internal ROM with the signal set positions corresponding to [1]. With SM = 1, the programmed RAM values are used. Note that SM is ignored during programming and that only one signal set can be stored. If ACM is changed, the signal set for the new code must be programmed.

Simulation Software

Free software for simulating the SCD02C turbo decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "scd02csim request" in the subject header. The software uses an exact functional simulation of the SCD02C turbo decoder, including all quantisation and limiting effects.

After unzipping `scd02csim.zip`, there should be `scd02csim.exe`, `code.txt`, `acm.txt`, `apsk.txt`, `int.txt` and `pp.txt`. The files `acm.txt`, `apsk.txt`, `int.txt` and `pp.txt` should not normally be edited, as they contain the SCCC code parameters, APSK modulation, interleaver and data puncturing parameters, respectively. The file `code.txt` contains the parameters for running `scd02csim`. These parameters are

EbNomin	Minimum E_b/N_0 (in dB)
EbNomax	Maximum E_b/N_0 (in dB)
EbNoinc	E_b/N_0 increment (in dB)
A	Modulation amplitude if SIQ = 1 and SM = 0
ferrmax	Number of frame errors to count
Pfmin	Minimum frame error rate (FER)
Pbmin	Minimum bit error rate (BER)
NP	Number of MAP decoders (1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60 or 120)
SIQ	Input data select (0 or 1)
ACM	Advanced coding and modulation select (0 to 27)
SH	Signal set SED shift if SIQ = 1 and SM = 0 (3 to 8)
SIZ	Select internal SCLZ, LIMZ if MODE = 0 (0 or 1)
NI	Number of iterations-1 (0 to 63)
LIMZ	Extrinsic information limit (1 to 127)
SCLZ	Extrinsic information scale (1 to 32)
ZTH	Extrinsic info. threshold (0 to 127)
M	Stopping mode (0 to 2)
SM	External signal set select (0 or 1)
SLD	MAP decoder delay select (0 or 1)
LOGMAP	Log-MAP decoding (MODE, 0 or 1)
C	C (0 to 11)
state	State file (0 to 2)
s1	Seed 1 (1 to 2147483562)
s2	Seed 2 (1 to 2147483398)
out_screen	Output data to screen (y or n)
read_x	Use external information data (y or n)
read_r	Use external received data (y or n)
out_dir	Output directory
in_dir	Input directory

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 bit error rate (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.

If SIQ = 1 and SM = 0, the values for A and SH given in acm.txt are used. Otherwise, the values given in code.txt are used. If SIZ = 1 and LOGMAP = 0, the values of SCLZ and LIMZ given in acm.txt are used. Otherwise, the values given in code.txt are used.

When the simulation is finished the output is given in, for example, file sim01.dat, where ACM = 1. 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 BER (Pb) and the average FER (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 15 for ACM = 1, $N_p = 60$ and max-log-MAP decoding. Auto-stopping was used with up to 10 iterations. When iterating is stopped early, the nasum (num*na), berr and ferr results at stopping are copied for each iteration to the maximum iteration number. Figure 16 shows the average number of iterations with E_b/N_0 .

```
{EbNomin EbNomax EbNoinc A}
0.0 1.00 0.1 256
{ferrmax Pfmin Pbmin}
128 1e-99 1e-5
{NP SIQ SM ACM SH}
60 1 0 1 5
{NI SIZ SCLZ LIMZ ZTH M}
9 1 23 96 23 1
{SLD LOGMAP C}
1 0 9
{state s1 s2 out_screen}
0 12345 67890 y
{read_x read_r out_dir in_dir}
n n output input
```

Genie aided early stopping (where the Genie knows the transmitted data) can be employed by selecting M = 2 in code.txt. This option is not available in the core.

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

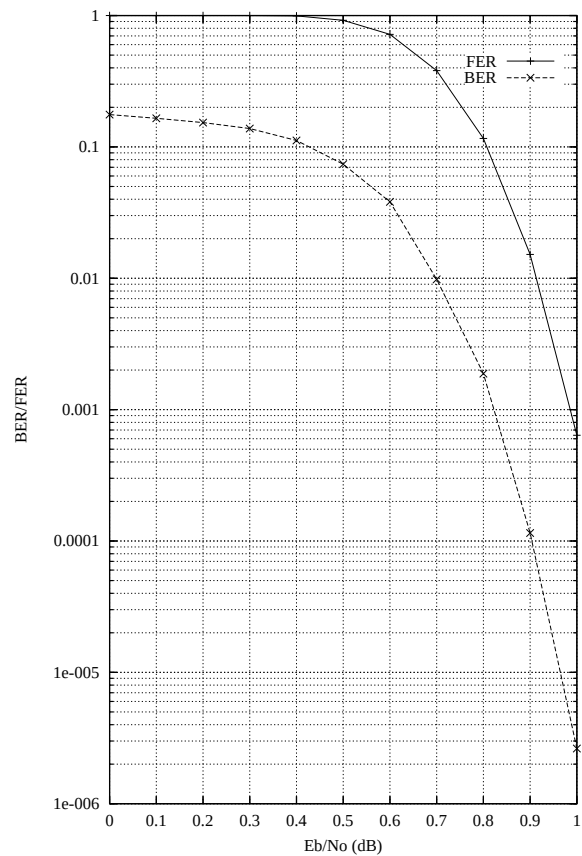


Figure 15: BER performance with ACM = 1, $N_p = 60$, max-log-MAP and auto-stopping.

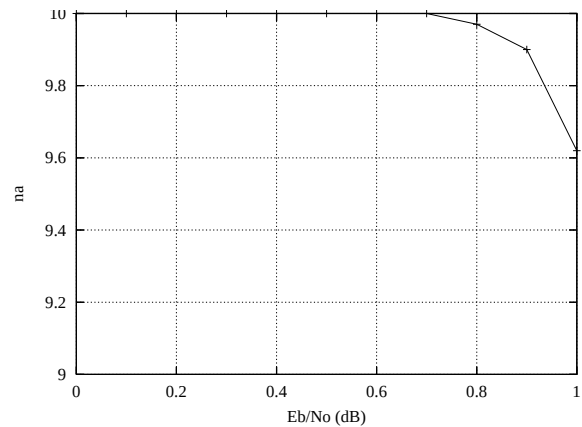


Figure 16: Average no. of iterations with ACM = 1, $N_p = 60$, max-log-MAP and auto-stopping.

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 ap-

pended to the existing `sim(ACM).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_(ACM).dat` in the directory given by `in_dir`, set `ferrmax = 0` and `read_x` to `y`, e.g., `x_01.dat` in directory input (each line contains one bit of data). The binary encoded stream after symbol interleaving `y_(ACM).dat` will be output to the directory given by `out_dir`, e.g., `y_01.dat` to directory output. `y_(ACM)` has N lines with one bit per line.

If `SIQ = 1`, `s_(ACM)` and `m_(ACM)` are also output with N_s lines each. `s_(ACM)` has the decimal value of the s -bit modulated symbol in each line. `m_(ACM)` has the inphase and quadrature values in each line where the signal set energy has been normalised to 1.

To decode data, place the received block of data in file `r_(ACM).dat` in directory `in_dir` and set `read_r` to `y`. The decoded data is output to `xd_(ACM).dat` in directory `out_dir`. If `SIQ = 0`, `r_(ACM).dat` has `R[i]`, $i = 0$ to $N-1$, e.g., the first three lines could be

```
-31
1
-25
```

where `R[0] = -31`, `R[1] = 1` and `R[2] = -25`. For QPSK the input data is of the form $R[i] = (1-2Y[i]+N[i])A/\sqrt{2}$ where A is the signal amplitude, $Y[i]$ is the coded bit, and $N[i]$ is white Gaussian noise with zero mean and normalised variance σ^2 . The magnitude of `R[i]` should be rounded to the nearest integer and be no greater than 127. If `read_r = y`, then `C` is externally input via `C`.

For `SIQ = 1`, `r_(ACM).dat` has in each line `RI[j]` and `RQ[j]`, $j = 0$ to N_s-1 in decimal form, e.g., the first three lines could be

```
175 -140
155 97
89 106
```

For `SM = 1`, `s_(ACM).dat` has in each line `SI[j]` and `SQ[j]`, $j = 0$ to $S-1$ in decimal form, e.g., the first three lines for `ACM = 7` (8PSK) could be

```
93 93
0 132
-93 0
```

SCCC Code Parameters

The SCCC code parameters are given in file `acm.txt`. Information in brackets {...} is a description of the data below. The first set of parameters are

ACM	Advanced coding and modulation select
K	Information data length
Ssur	Number of inner data non-punctured bits per 300 bit segment
Delta	Number of inner parity punctured bits
SS	Signal set select
A	Average signal amplitude
SH	Signal set SED shift value
SCLZ	Extrinsic information scale value
LIMZ	Extrinsic information limit value

APSK Modulation Parameters

The APSK modulation parameters are given in file `apsk.txt`. The first set of parameters are

NS	Number of signal set variations
L	Number of different signal amplitudes
Name	Signal set name

The next two sets of parameters for level $i = 1$ to L are

n_i	Number of points for level i
t_i	Initial phase in degrees for level i

n_i has one row, while t_i has NS rows, one row for each variation. The next NS rows of parameters for level $j = 2$ to L are r_j/r_1 , the amplitude ratio for each signal set variation.

The last set of parameters are the actual mappings. There are L rows, each row giving the decimal representation of the n_i points in each level. The first value in each row corresponds to the initial phase. The remaining points in each level are formed in anti-clockwise order.

If desired, the amplitude ratios and initial phase can be adjusted to reproduce the effects of a non-linear amplifier on the signal set. Other signal sets can be added to the end of the file, starting at `SS = 10`.

Interleaver Parameters

There are 19 different interleavers, the parameters of which are stored in file `int.txt`. Each interleaver has two rows. The first and second rows store $\alpha(k)$ and $\beta(k)$ for $0 \leq k \leq W-1$. The value of K is used to select the appropriate interleaver.

Inner Data Puncturing Parameters

A depth 300 lookup table is used to determine the puncturing pattern for inner encoded data. File pp.txt is used to store the positions of the punctured data in each 300 bit segment. Depending on ACM, the first 300–Ssur rows of pp.txt are selected. The SCCC standard provides the first 100 positions. As the smallest value of Ssur is 208, only the first 92 positions are selected.

Ordering Information

SW–SCD02C–SOS (SignOnce Site License)
 SW–SCD02C–SOP (SignOnce Project License)
 SW–SCD02C–VHD (VHDL ASIC License)

All licenses include Xilinx VHDL cores. All licenses allows unlimited instantiations.

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

References

- [1] Consultative Committee for Space Data Systems, "Recommendation for space data system standards: Flexible advanced coding and modulation scheme for high rate telemetry applications," CCSDS 131.2–B–1, Blue Book, Mar. 2012.
- [2] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT–20, pp. 284–287, Mar. 1974.
- [3] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," *ICC'95*, Seattle, WA, USA, pp. 1009–1013, June 1995.
- [4] M. C. Reed and J. A. Asenstorfer, "A novel variance estimator for turbo-code decoding," *Int. Conf. on Telecommun.*, Melbourne, Australia, pp. 173–178, Apr. 1997.

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 responsibility 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.

© 2025–2026 *Small World Communications*. All Rights Reserved. Xilinx, Spartan, Virtex, 7-Series, Zynq, Artix, Kintex, UltraScale and UltraScale+ are registered trademarks and all XC-prefix product designations are trademarks of Advanced Micro Devices, Inc. and Xilinx, Inc. All other trademarks and registered trademarks are the property of their respective owners.

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

- 0.00 24 Jan. 2025. Preliminary product specification.
- 0.01 19 Feb. 2026. Added two clock cycles per iteration for decoding time. Added decoder speed and complexity without demapper for Kintex UltraScale+.
- 0.02 10 Jul. 2026. Added demapper. Updated speed and complexity for Kintex UltraScale+. Added 16APSK, 32APSK and LLR figures. Updated simulation program parameters.
- 0.03 5 Oct. 2026. Added SIZ input and internal LIMZ LUT. Updated SCLZ for ACM = 8 and LIMZ for ACM > 5. Added E_b/N_0 and E_s/N_0 performance for ACM < 27. Added ACM = 1 performance figures.
- 1.00 12 Oct. 2026. First release. Updated SCLZ and added performance for ACM = 27.