LCD01D Features

LDPC Decoder

- Digital Video Broadcasting Second Generation (DVB–S2) and Extensions (DVB–S2X) compatible
- Nominal code rates from 1/5 to 9/10
- Data lengths from 2512 to 58192 bits
- Optional $\pi/2$ BPSK SF2, $\pi/2$ BPSK, QPSK, 8PSK, 8APSK, 16APSK, 32APSK, 64APSK, 128APSK and 256APSK demapping and deinterleaving with 11–bit inphase and quadrature two's complement input
- Optional interleaved or deinterleaved 8–bit log–likelihood–ratio (LLR) input
- Includes ping–pong input and output memories, BCH decoder and optional descrambler
- Scaled min–sum modified Gauss–Seidel low density parity check (LDPC) decoding algorithm
- Up to 323 MHz internal clock
- Up to 290 Mbit/s with 30 decoder iterations
- Up to 256 iterations
- Programmable signal points option
- Optional power efficient early stopping
- LDPC parity check and BCH error location polynomial degree output
- ~42,000 LUTs and 287 RAMB18s for Xilinx Virtex–5, Virtex–6, 7–Series, UltraScale and UltraScale+ FPGAs. ~53,000 ALUTs and 436 M10Ks for Altera Cyclone V.
- 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.

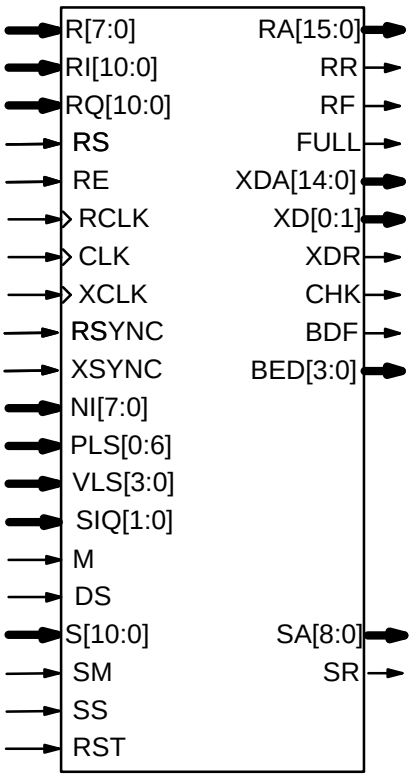

Figure 1: LCD01D schematic symbol.

Introduction

The LCD01D is a fully compatible Digital Video Broadcasting Second Generation (DVB–S2) [1] and Extensions (DVB–S2X) [2] error control decoder. It includes an optional demapper, LDPC decoder, BCH decoder and optional descrambler. Nominal LDPC code rates range from 1/5 to 9/10. Irregular repeat accumulate quasi–cyclic LDPC codes with circulant weights ranging from 1 to 4 (for the DVB–S2 codes [1]) and weight 1 (for the DVB–S2X codes[2]) are used. Circulant sizes for all codes is 360x360. There are from 5 to 140 circulant rows and either 45, 90 or 180 circulant columns in the parity check matrix.

The check node degree is irregular with one to five different values for each code. Due to the accumulate encoding method for the parity check bits, the first row has a check degree which is one less than the following 359 rows. The average check degree varies from 3.417 to 30.

The variable node degree is irregular with two to six different values for each code. Variable degrees range from 3 to 24 for the information bits. For the parity bits, due to the accumulate encoding method used, all column's have a variable degree of 2, except for the last column which has a degree of 1.

For DVB–S2 there are two sets of LDPC codes in the standard. These are Short and Normal which have code lengths of 16,200 and 64,800 bits, respectively. For DVB–S2X, there are additional Short and Normal codes, along with Medium codes which have coded lengths of 32,400 bits before puncturing. Some of these

codes are shortened and punctured for VL–SNR to obtain coded lengths of 14,976, 15,390, 30,780 or 61,560 bits. For DVB–S2 there are a total of 10 Short codes and 11 Normal codes. For DVB–S2X there are an additional seven Short codes, three Medium codes and 24 Normal codes. Five codes are used in both punctured and non–punctured form, giving a total of 60 different LDPC codes.

For DVB–S2 four different modulation schemes are used; QPSK, 8PSK, 16APSK and 32APSK [1]. For 16APSK and 32APSK there are six and five different versions, respectively, formed by varying the relative levels of the signal points. For DVB–S2X there are additional $\pi/2$ BPSK, $\pi/2$ BPSK Spreading Factor 2 (SF2), 8APSK (2 versions), 16APSK (9 versions), 32APSK (6 versions), 64APSK (4 versions), 128APSK (2 versions) and 256APSK (5 versions) modulation schemes used [2]. This gives a total of 43 different modulation schemes. Along with 27 different bit interleaver patterns, a total of 116 different modulation and coding schemes are used.

A modified Gauss–Seidel [3] iterative message passing algorithm is used in the decoder. In each clock cycle, 90 check nodes are decoded and the variable messages (VMs) updated. Each iteration requires $4qd_c$ clock cycles to calculate the updated VMs, where d_c is the average check degree value and q is the number of circulant rows. For calculation of the check messages (CMs), the scaled min–sum decoding algorithm [4] is used.

The LDPC decoder can achieve up to 290 Mbit/s with 30 iterations using a 323 MHz internal clock. Optional early stopping allows the decoder to reduce power consumption with no degradation in performance.

The decoder contains an input memory, VM memory, CM memory, VM sign memory, parameter memory and an output memory for the decoded data. The input and output memories are separated into two halves to allow ping–pong operation and maximum decoder speed. Separate clocks can be used for the input and output memories. As DVB–S2 LDPC codes can have circulant weights greater than one, several additional memories are also used.

An optional demapper and deinterleaver for the various modulation schemes is included. This allows 11–bit inphase and quadrature samples to be demapped into individual bit log–likelihood ratios (LLR) and deinterleaved. Alternatively, 8–bit LLR data can be input either interleaved or deinterleaved into the decoder.

A BCH decoder using the Berlekamp algorithm [5] is included to clean up any residual errors from the LDPC decoder. For DVB–S2 codes, the BCH decoder can correct up to either 8, 10 or 12 errors. For DVB–S2X, up to 12 errors can be corrected. For Short, Medium and Normal codes the corresponding Galois field is GF(2^{14}), GF(2^{15}) and GF(2^{16}), respectively.

Figure 1 shows the schematic symbol for the LCD01D decoder. The VHDL core can be used with Xilinx 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.

Table 1 shows the performance achieved with various Xilinx parts, 30 iterations and the slowest (rate 1/5 Short SF2 VL–SNR) and fastest (rate 9/10 Normal) codes. T_{cp} is the minimum clock period over recommended operating conditions. These performance figures may change due to device utilisation and configuration.

Table 1: Performance of Xilinx parts.

Xilinx Part	T_{cp} (ns)	f_d (Mbit/s)	
		Low	High
XC7A200T–1	8.805	19.3	101.9
XC7A200T–2	7.265	23.4	123.6
XC7A200T–3	6.407	26.5	140.1
XC7Z030–1	7.873	21.6	114.0
XC7Z030–2	6.385	26.6	140.6
XC7Z030–3	6.139	27.7	146.2
XC7K160T–1	6.613	25.7	135.7
XC7K160T–2	5.512	30.8	162.9
XC7K160T–3	5.005	33.9	179.4
XCKU035–1	5.271	32.2	170.3
XCKU035–2	4.489	37.9	200.0
XCKU035–3	3.915	43.4	229.3
XCKU3P–1	3.869	43.9	232.0
XCKU3P–2	3.382	50.3	265.5
XCKU3P–3	3.092	55.0	290.4

Signal Descriptions

BDF BCH Decoder Fail
BED BCH Error Location Polynomial Degree
CHK Parity Check
CLK System Clock
DS Descrambler Select
0 = No descrambling
1 = Descrambling enabled
FULL Decoder Full (new data not accepted)

M	Early Stopping Mode 0 = No early stopping 1 = Early stopping enabled
NI	Number of Iterations minus one (0–255) I = NI+1 where I is number of iterations
PLS	Physical Layer Signalling Select (2–124) ($b_0\,b_1\,b_2\,b_3\,b_4\,b_5\,b_6$). See Table 3
R	Received LLR Data (8–bit)
RA	Received Data Address
RCLK	Received Data Clock
RE	Received Data Enable
RF	Received Data Finish
RI	Received Inphase Data (11–bit)
RQ	Received Quadrature Data (11–bit)
RR	Received Data Ready
RS	Received Data Start
RST	Synchronous Reset
RSYNC	RCLK and CLK equal
S	Symbol Value (11–bit)
SA	Symbol Address
SIQ	Inphase and Quadrature Select 0 = No Demapping (deinterleaved data input to R) 1 = Demapping (data input to RI and RQ) 3 = No Demapping (interleaved data input to R)
SM	Symbol Memory Select 0 = ROM 1 = RAM
SR	Symbol Ready
SS	Symbol Start
VLS	Very Low SNR (VL–SNR) Select (0–8) See Table 3
XCLK	Decoded Data Clock
XD	Decoded Data (2–bit)
XDA	Decoded Data Address
XDR	Decoded Data Ready
XSYNC	XCLK and CLK equal

Table 3 lists the 116 different modulation and coding combinations selected by PLS and VLS. The information data length is given by K and the number of two dimensional modulated symbols is given by N_S. $R_{\text{eff}} = K/N_S$ is the bandwidth efficiency in bit/sym (synchronisation, PLS and VL–SNR 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, SC and SI are selection parameters used to select the signal set, LDPC code and symbol interleaver, respectively. 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. MS is the check message scale factor used by the LDPC decoder.

Table 4 lists the 60 different LDPC codes selected by SC. The coded data length after rate matching is given by N. The BCH error correction capability is given by t. The number of circulant rows is given by q. The minimum, maximum and average check degree values are given by $d_{c,\text{min}}$, $d_{c,\text{max}}$ and d_c, respectively. The frequency of each check degree value from $d_{c,\text{min}}$ to $d_{c,\text{max}}$ is given by $d_{c,\text{freq}}$. The parameter qd_c is the complexity of the LDPC code with f_d being the decoder speed when a CLK of frequency F_d = 100 MHz is used.

LDPC Decoder Parameters

For binary modulation, we model the received sample at time i as

$$r_i = A(s_i + n_i) \tag{1}$$

where A is the no–noise amplitude, s_i is the modulated signal with value +1 for coded bit $y_i = 0$ and –1 for $y_i = 1$, n_i is white Gaussian noise with normalised variance

$$\sigma^2 = 1/(2RE_b/N_0) \tag{2}$$

and $R = K/N$ is the code rate.

The value of A directly corresponds to the 8–bit two's complement inputs for R (shown in Table 2). The 8–bit inputs have 256 quantisation regions. The quantisation regions are labelled from –128 to 127. For example, one could have A = 15.7. This value of A lies in quantisation region 16 (which has a range between 15.5 and 16.5).

Table 2: Quantisation for received data R.

Decimal	Binary	Range
127	01111111	126.5↔∞
126	01111110	125.5↔126.5
⋮	⋮	⋮
2	00000010	1.5↔2.5
1	00000001	0.5↔1.5
0	00000000	–0.5↔0.5
–1	11111111	–1.5↔–0.5
–2	11111110	–2.5↔–1.5
⋮	⋮	⋮
–127	10000001	–127.5↔–126.5
–128	10000000	–∞↔–127.5

Table 3: MODCOD Coding

PLS	MODCOD Name	K	N_S	R_{eff} (bit/sym)	E_b/N_0 (dB)	E_S/N_0 (dB)	A	SS	SC	SI	SH	MS
2	QPSK 1/4 Normal	16008	32400	0.49407	0.53	–2.47	239	0	10	–1	0	0.97
3	QPSK 1/4 Short	3072	8100	0.37926	0.80	–3.42	173	0	0	–1	0	0.97
4	QPSK 1/3 Normal	21408	32400	0.66074	0.51	–1.26	264	0	11	–1	0	0.97
5	QPSK 1/3 Short	5232	8100	0.64593	0.76	–1.14	230	0	1	–1	0	0.97
6	QPSK 2/5 Normal	25728	32400	0.79407	0.67	–0.31	218	0	12	–1	0	0.97
7	QPSK 2/5 Short	6312	8100	0.77926	0.93	–0.18	214	0	2	–1	0	0.97
8	QPSK 1/2 Normal	32208	32400	0.99407	1.05	1.03	266	0	13	–1	0	0.95
9	QPSK 1/2 Short	7032	8100	0.86815	1.18	0.56	350	0	3	–1	0	0.96
10	QPSK 3/5 Normal	38688	32400	1.19407	1.51	2.28	397	0	14	–1	0	0.79
11	QPSK 3/5 Short	9552	8100	1.17926	1.74	2.42	396	0	4	–1	0	0.79
12	QPSK 2/3 Normal	43040	32400	1.32840	1.81	3.06	760	0	15	–1	0	0.91
13	QPSK 2/3 Short	10632	8100	1.31259	2.05	3.20	743	0	5	–1	0	0.91
14	QPSK 3/4 Normal	48408	32400	1.49407	2.26	4.02	912	0	16	–1	0	0.88
15	QPSK 3/4 Short	11712	8100	1.44593	2.53	4.14	866	0	6	–1	0	0.90
16	QPSK 4/5 Normal	51648	32400	1.59407	2.61	4.65	1018	0	17	–1	0	0.86
17	QPSK 4/5 Short	12432	8100	1.53481	2.87	4.73	1017	0	7	–1	0	0.90
18	QPSK 5/6 Normal	53840	32400	1.66173	2.92	5.14	1059	0	18	–1	0	0.87
19	QPSK 5/6 Short	13152	8100	1.62370	3.17	5.28	1014	0	8	–1	0	0.88
20	QPSK 8/9 Normal	57472	32400	1.77383	3.66	6.17	1226	0	19	–1	0	0.87
21	QPSK 8/9 Short	14232	8100	1.75704	3.87	6.33	1142	0	9	–1	0	0.88
22	QPSK 9/10 Normal	58192	32400	1.79605	3.82	6.38	1328	0	20	–1	0	0.87
24	8PSK 3/5 Normal	38688	21600	1.79111	3.04	5.59	374	1	14	3	11	0.97
25	8PSK 3/5 Short	9552	5400	1.76889	3.28	5.73	264	1	4	3	10	0.94
26	8PSK 2/3 Normal	43040	21600	1.99259	3.56	6.56	310	1	15	0	10	0.95
27	8PSK 2/3 Short	10632	5400	1.96889	3.77	6.71	580	1	5	0	11	0.85
28	8PSK 3/4 Normal	48408	21600	2.24111	4.37	7.88	645	1	16	0	11	0.83
29	8PSK 3/4 Short	11712	5400	2.16889	4.67	8.02	346	1	6	0	9	0.88
30	8PSK 5/6 Normal	53840	21600	2.49259	5.32	9.30	555	1	18	0	10	0.79
31	8PSK 5/6 Short	13152	5400	2.43556	5.68	9.50	207	1	8	0	7	0.87
32	8PSK 8/9 Normal	57472	21600	2.66074	6.36	10.62	466	1	19	0	9	0.84
33	8PSK 8/9 Short	14232	5400	2.63556	6.58	10.79	482	1	9	0	9	0.84
34	8PSK 9/10 Normal	58192	21600	2.69407	6.57	10.89	345	1	20	0	8	0.84
36	16APSK 2/3 Normal	43040	16200	2.65679	4.70	8.96	353	2	15	0	9	0.90
37	16APSK 2/3 Short	10632	4050	2.62519	4.93	9.11	540	2	5	0	10	0.87
38	16APSK 3/4 Normal	48408	16200	2.98815	5.44	10.20	617	3	16	0	10	0.85
39	16APSK 3/4 Short	11712	4050	2.89185	5.73	10.34	321	3	6	0	8	0.88
40	16APSK 4/5 Normal	51648	16200	3.18815	5.96	11.00	485	4	17	0	9	0.83
41	16APSK 4/5 Short	12432	4050	3.06963	6.22	11.09	490	4	7	0	9	0.87
42	16APSK 5/6 Normal	53840	16200	3.32346	6.33	11.56	369	5	18	0	8	0.83
43	16APSK 5/6 Short	13152	4050	3.24741	6.64	11.75	176	5	8	0	6	0.87
44	16APSK 8/9 Normal	57472	16200	3.54765	7.31	12.82	297	6	19	0	7	0.85

Table 3: MODCOD Coding (cont.)

PLS/ VLS	MODCOD Name	K	N_S	R_{eff} (bit/ sym)	E_b/N_0 (dB)	E_S/N_0 (dB)	A	SS	SC	SI	SH	MS
45	16APSK 8/9 Short	14232	4050	3.51407	7.53	13.00	421	6	9	0	8	0.84
46	16APSK 9/10 Normal	58192	16200	3.59210	7.49	13.05	435	7	20	0	8	0.85
48	32APSK 3/4 Normal	48408	12960	3.73419	7.05	12.78	402	8	16	0	9	0.94
49	32APSK 3/4 Short	11712	3240	3.61481	7.43	13.01	584	8	6	0	9	0.90
50	32APSK 4/5 Normal	51648	12960	3.98519	7.67	13.67	583	9	17	0	10	0.94
51	32APSK 4/5 Short	12432	3240	3.83704	7.98	13.83	456	9	7	0	8	0.90
52	32APSK 5/6 Normal	53840	12960	4.15432	8.05	14.24	453	10	18	0	9	0.92
53	32APSK 5/6 Short	13152	3240	4.05926	8.40	14.46	485	10	8	0	8	0.88
54	32APSK 8/9 Normal	57472	12960	4.43457	9.08	15.56	583	11	19	0	8	0.85
55	32APSK 8/9 Short	14232	3240	4.39259	9.31	15.75	533	11	9	0	8	0.85
56	32APSK 9/10 Normal	58192	12960	4.49012	9.30	15.84	424	12	20	0	7	0.84
64/0	QPSK 2/9 Normal	14208	30780	0.46160	0.49	–2.81	191	0	36	–1	0	0.97
64/1	BPSK 1/5 Medium	5660	30780	0.18389	0.71	–6.57	192	–1	33	–1	0	0.97
64/2	BPSK 11/45 Medium	7740	30780	0.25146	0.58	–5.40	207	–1	34	–1	0	0.97
64/3	BPSK 1/3 Medium	10620	30780	0.34503	0.68	–3.91	280	–1	35	–1	0	0.97
64/4	BPSK–S 1/5 Short	2512	30780	0.08161	1.21	–9.66	196	–2	21	–1	0	0.97
64/7	BPSK–S 11/45 Short	3792	30780	0.12320	0.79	–8.31	201	–2	22	–1	0	0.97
65/5	BPSK 4/15 Short	4152	14976	0.27724	0.88	–4.74	169	–1	24	–1	0	0.96
65/6	BPSK 1/5 Short	3072	14976	0.20513	0.86	–6.03	189	–1	23	–1	0	0.97
65/8	BPSK 1/3 Short	5232	14976	0.34936	0.88	–3.68	259	–1	25	–1	0	0.96
66	QPSK 13/45 Normal	18528	32400	0.57185	0.51	–1.89	177	0	37	–1	0	0.97
67	QPSK 9/20 Normal	28968	32400	0.89407	0.81	0.33	233	0	38	–1	0	0.97
68	QPSK 11/20 Normal	35448	32400	1.09407	1.15	1.55	267	0	41	–1	0	0.95
69	8APSK 5/9–L Normal	35808	21600	1.65778	2.65	4.85	452	13	42	0	12	0.95
70	8APSK 26/45–L Normal	37248	21600	1.72444	2.92	5.27	429	14	43	0	12	0.95
71	8PSK 23/36 Normal	41208	21600	1.90778	3.42	6.22	243	1	47	0	9	0.79
72	8PSK 25/36 Normal	44808	21600	2.07444	3.92	7.08	432	1	51	2	11	0.93
73	8PSK 13/18 Normal	46608	21600	2.15778	4.17	7.51	604	1	53	2	11	0.83
74	16APSK 1/2–L Normal	32208	16200	1.98815	3.12	6.12	296	21	39	10	10	0.95
75	16APSK 8/15–L Normal	34368	16200	2.12148	3.50	6.75	270	21	40	6	10	0.94
76	16APSK 5/9–L Normal	35808	16200	2.21037	3.52	6.97	425	21	42	5	11	0.97
77	16APSK 26/45 Normal	37248	16200	2.29926	3.90	7.54	504	15	44	9	11	0.96
78	16APSK 3/5 Normal	38688	16200	2.38815	4.18	7.96	219	15	14	10	9	0.95
79	16APSK 3/5–L Normal	38688	16200	2.38815	3.73	7.53	175	22	45	0	8	0.97
80	16APSK 28/45 Normal	40128	16200	2.47704	4.28	8.22	342	16	46	7	10	0.91
81	16APSK 23/36 Normal	41208	16200	2.54370	4.37	8.44	264	17	47	8	9	0.95
82	16APSK 2/3–L Normal	43008	16200	2.65481	4.26	8.52	189	23	49	0	8	0.95
83	16APSK 25/36 Normal	44808	16200	2.76593	4.88	9.31	397	17	51	6	10	0.96
84	16APSK 13/18 Normal	46608	16200	2.87704	5.19	9.77	511	3	53	8	10	0.79
85	16APSK 7/9 Normal	50208	16200	3.09926	5.79	10.70	535	18	57	10	10	0.79
86	16APSK 77/90 Normal	55248	16200	3.41037	6.65	11.98	500	19	59	1	9	0.79

Table 3: MODCOD Coding (cont.)

PLS	MODCOD Name	K	N_S	R_{eff} (bit/sym)	E_b/N_0 (dB)	E_S/N_0 (dB)	A	SS	SC	SI	SH	MS
87	32APSK 2/3–L Normal	43040	12960	3.32099	6.00	11.21	354	24	15	12	9	0.96
89	32APSK 32/45 Normal	45888	12960	3.54074	6.40	11.89	699	27	52	14	11	0.94
90	32APSK 11/15 Normal	47328	12960	3.65185	6.68	12.30	364	28	54	14	9	0.95
91	32APSK 7/9 Normal	50208	12960	3.87407	7.30	13.17	528	29	57	13	10	0.83
92	64APSK 32/45–L Normal	45888	10800	4.24889	7.81	14.09	169	30	52	18	6	0.94
93	64APSK 11/15 Normal	47328	10800	4.38222	8.52	14.93	499	33	54	25	9	0.95
95	64APSK 7/9 Normal	50208	10800	4.64889	8.84	15.52	414	31	58	17	8	0.95
97	64APSK 4/5 Normal	51648	10800	4.78222	9.16	15.95	540	31	17	16	9	0.92
99	64APSK 5/6 Normal	53840	10800	4.98519	9.61	16.59	303	32	18	21	7	0.93
100	128APSK 3/4 Normal	48408	9258	5.22878	10.68	17.87	507	34	56	22	8	0.95
101	128APSK 7/9 Normal	50208	9258	5.42320	11.34	18.67	698	35	57	20	9	0.91
102	256APSK 29/45–L Normal	41568	8100	5.13185	9.98	17.09	540	36	48	19	8	0.97
103	256APSK 2/3–L Normal	43008	8100	5.30963	10.25	17.51	510	39	49	0	8	0.97
104	256APSK 31/45–L Normal	44448	8100	5.48741	10.87	18.25	546	36	50	23	8	0.94
105	256APSK 32/45 Normal	45888	8100	5.66519	11.27	18.79	561	37	52	26	8	0.93
106	256APSK 11/45–L Normal	47328	8100	5.84296	11.28	18.95	446	40	55	0	7	0.97
107	256APSK 3/4 Normal	48408	8100	5.97630	11.93	19.69	481	38	56	24	7	0.95
108	QPSK 11/45 Short	3792	8100	0.46815	0.74	–2.57	180	0	26	–1	0	0.97
109	QPSK 4/15 Short	4152	8100	0.51259	0.80	–2.14	151	0	27	–1	0	0.97
110	QPSK 14/45 Short	4872	8100	0.60148	0.78	–1.44	186	0	28	–1	0	0.97
111	QPSK 7/15 Short	7392	8100	0.91259	1.20	0.75	198	0	29	–1	0	0.95
112	QPSK 8/15 Short	8472	8100	1.04593	1.44	1.59	212	0	30	–1	0	0.95
113	QPSK 32/45 Short	11352	8100	1.40148	2.13	3.59	613	0	32	–1	0	0.87
114	8PSK 7/15 Short	7392	5400	1.36889	2.44	3.79	242	1	29	2	10	0.96
115	8PSK 8/15 Short	8472	5400	1.56889	2.85	4.76	338	1	30	2	11	0.97
116	8PSK 26/45 Short	9192	5400	1.70222	3.12	5.42	384	1	31	2	11	0.97
117	8PSK 32/45 Short	11352	5400	2.10222	4.32	7.51	275	1	32	0	9	0.77
118	16APSK 7/15 Short	7392	4050	1.82519	3.46	6.03	155	20	29	3	8	0.96
119	16APSK 8/15 Short	8472	4050	2.09185	3.88	7.04	211	16	30	3	9	0.94
120	16APSK 26/45 Short	9192	4050	2.26963	4.16	7.70	445	15	31	4	11	0.94
121	16APSK 3/5 Short	9552	4050	2.35852	4.44	8.13	219	15	4	9	9	0.93
122	16APSK 32/45 Short	11352	4050	2.80296	5.33	9.78	532	3	32	0	11	0.92
123	32APSK 2/3 Short	10632	3240	3.28148	6.24	11.37	361	25	5	15	9	0.92
124	32APSK 32/45 Short	11352	3240	3.50370	6.72	12.13	503	26	32	11	10	0.92

Due to quantisation and limiting effects the value of A should be adjusted according to the received signal to noise ratio. Where QPSK modulation is used in Table 3, the value of A should be divided by $32\sqrt{2}$. For example, for PLS = 2, A = 239 when using RI and RQ. Dividing by $32\sqrt{2}$ gives A = 5.3 when data is input to R.

Demapping

For 90° binary shift keying (π/2 BPSK), 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

Table 4: Decoder Speeds, F_d = 100 MHz

SC	LDPC Code ID	K	N	t	q	$d_{c,\,min}$	$d_{c,\,max}$	$d_{c,\,freq}$	d_c	qd_c	f_d (Mbit/s)
0	1/4 Short	3072	16200	12	36	3	4	(1,3)/4	3.75	135	18.96
1	1/3 Short	5232	16200	12	30	5	5	(1)	5	150	29.06
2	2/5 Short	6312	16200	12	27	6	6	(1)	6	162	32.46
3	1/2 Short	7032	16200	12	25	4	7	(4,9,10,2)/25	5.4	135	43.40
4	3/5 Short	9552	16200	12	18	11	11	(1)	11	198	40.20
5	2/3 Short	10632	16200	12	15	10	10	(1)	10	150	59.05
6	3/4 Short	11712	16200	12	12	9	13	(1,3,4,3,1)/12	11	132	73.92
7	4/5 Short	12432	16200	12	10	11	13	(1,3,6)/10	12.5	125	82.86
8	5/6 Short	13152	16200	12	8	16	19	(4,1,1,2)/8	17.125	137	79.98
9	8/9 Short	14232	16200	12	5	27	27	(1)	27	135	87.83
10	1/4 Normal	16008	64800	12	135	4	4	(1)	4	540	24.70
11	1/3 Normal	21408	64800	12	120	5	5	(1)	5	600	29.73
12	2/5 Normal	25728	64800	12	108	6	6	(1)	6	648	33.08
13	1/2 Normal	32208	64800	12	90	7	7	(1)	7	630	42.60
14	3/5 Normal	38688	64800	12	72	11	11	(1)	11	792	40.71
15	2/3 Normal	43040	64800	10	60	10	10	(1)	10	600	59.77
16	3/4 Normal	48408	64800	12	45	14	14	(1)	14	630	64.03
17	4/5 Normal	51648	64800	12	36	18	18	(1)	18	648	66.42
18	5/6 Normal	53840	64800	10	30	22	22	(1)	22	660	67.98
19	8/9 Normal	57472	64800	8	20	27	27	(1)	27	540	88.69
20	9/10 Normal	58192	64800	8	18	30	30	(1)	30	540	89.80
21	1/5 Short SF2 VLS	2512	15390	12	36	2	4	(4,13,19)/36	3.417	123	17.01
22	11/45 Short SF2 VLS	3792	15390	12	34	4	4	(1)	4	136	23.23
23	1/5 Short VLS	3072	14976	12	36	3	4	(1,3)/4	3.75	135	18.96
24	4/15 Short VLS	4152	14976	12	33	4	5	(1,10)/11	4.909	162	21.36
25	1/3 Short VLS	5232	14976	12	30	5	5	(1)	5	150	29.06
26	11/45 Short	3792	16200	12	34	4	4	(1)	4	136	23.23
27	4/15 Short	4152	16200	12	33	4	5	(1,10)/11	4.909	162	21.35
28	14/45 Short	4872	16200	12	31	5	5	(1)	5	155	26.19
29	7/15 Short	7392	16200	12	24	8	9	(13,11)/24	8.458	203	30.34
30	8/15 Short	8472	16200	12	21	9	10	(1,20)/21	9.952	209	33.77
31	26/45 Short	9192	16200	12	19	10	10	(1)	10	190	40.31
32	32/45 Short	11352	16200	12	13	13	13	(1)	13	169	55.97
33	1/5 Medium VLS	5660	30780	12	72	3	4	(13,59)/72	3.819	275	17.15
34	11/45 Medium VLS	7740	30780	12	68	4	4	(1)	4	272	23.71
35	1/3 Medium VLS	10620	30780	12	60	5	5	(1)	5	300	29.50
36	2/9 Normal VLS	14208	61560	12	140	4	4	(1)	4	560	21.14
37	13/45 Normal	18528	64800	12	128	4	5	(3,29)/32	4.906	628	24.58
38	9/20 Normal	28968	64800	12	99	7	7	(1)	7	693	34.83
39	90/180 Normal	32208	64800	12	90	7	8	(1,8)/9	7.889	710	37.80
40	96/180 Normal	34368	64800	12	84	8	9	(5,23)/28	8.821	741	38.65

Table 4: Decoder Speeds, F_d = 100 MHz (cont.)

SC	LDPC Code ID	K	N	t	q	$d_{c,\min}$	$d_{c,\max}$	$d_{c,\text{freq}}$	d_c	qd_c	f_d (Mbit/s)
41	11/20 Normal	35448	64800	12	81	9	9	(1)	9	729	40.52
42	100/180 Normal	35808	64800	12	80	8	9	(1,7)/8	8.875	710	42.03
43	104/180 Normal	37248	64800	12	76	9	10	(3,35)/38	9.921	754	41.17
44	26/45 Normal	37248	64800	12	76	10	10	(1)	10	760	40.84
45	18/30 Normal	38688	64800	12	72	10	11	(1,71)/72	10.986	791	40.76
46	28/45 Normal	40128	64800	12	68	10	10	(1)	10	680	49.17
47	23/36 Normal	41208	64800	12	65	10	10	(1)	10	650	52.83
48	116/180 Normal	41568	64800	12	64	11	12	(17,47)/64	11.734	751	46.12
49	20/30 Normal	43008	64800	12	60	13	14	(59,1)/60	13.017	781	45.89
50	124/180 Normal	44448	64800	12	56	13	14	(25,31)/56	13.554	759	48.80
51	25/36 Normal	44808	64800	12	55	13	13	(1)	13	715	52.22
52	128/180 Normal	45888	64800	12	52	14	15	(27,25)/52	14.481	753	50.78
53	13/18 Normal	46608	64800	12	50	14	14	(1)	14	700	55.48
54	132/180 Normal	47328	64800	12	48	15	16	(11,37)/48	15.771	757	52.10
55	22/30 Normal	47328	64800	12	48	16	17	(41,7)/48	16.146	775	50.89
56	135/180 Normal	48408	64800	12	45	16	17	(2,1)/3	16.333	735	54.88
57	140/180 Normal	50208	64800	12	40	19	20	(3,5)/8	19.625	785	53.30
58	7/9 Normal	50208	64800	12	40	18	18	(1)	18	720	58.11
59	154/180 Normal	55248	64800	12	26	29	30	(21,5)/26	29.192	759	60.66

$$r_j^{\text{I}} = A(s_j^{\text{I}} + n_j^{\text{I}}) \tag{3}$$

$$r_j^{\text{Q}} = A(s_j^{\text{Q}} + n_j^{\text{Q}}) \tag{4}$$

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

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

where s is the number of bits in each 2–D signal set. Note that the number of coded symbols $N_S = \lceil N/s \rceil$, where s is the number of modulated bits per 2–D symbol, e.g., s = 0.5 for $\pi/2$ BPSK SF2, s = 1 for $\pi/2$ BPSK, s = 2 for QPSK, etc.

For the inphase RI and quadrature RQ sampled values, 11–bit two's complement quantisation is used. Table 5 gives the quantisation for RI and RQ.

For best performance with $\pi/2$ BPSK and 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.

Table 5: Quantisation for RI and RQ.

Decimal	Binary	Range
1023	01111111111	1022.5↔∞
1022	01111111110	1021.5↔1022.5
⋮	⋮	⋮
2	00000000010	1.5↔2.5
1	00000000001	0.5↔1.5
0	00000000000	–0.5↔0.5
–1	11111111111	–1.5↔–0.5
–2	11111111110	–2.5↔–1.5
⋮	⋮	⋮
–1023	10000000001	–1023.5↔–1022.5
–1024	10000000000	–∞↔–1023.5

Figures 2 and 3 show the signal sets within the 11–bit quantisation regions and recommended signal amplitude for 16APSK and 32APSK, respectively, using the values of A for 3/4 Normal. Tables 6 and 7 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. For 128APSK, the five inner rings have 16 points each, while the outer ring has 48 points. For

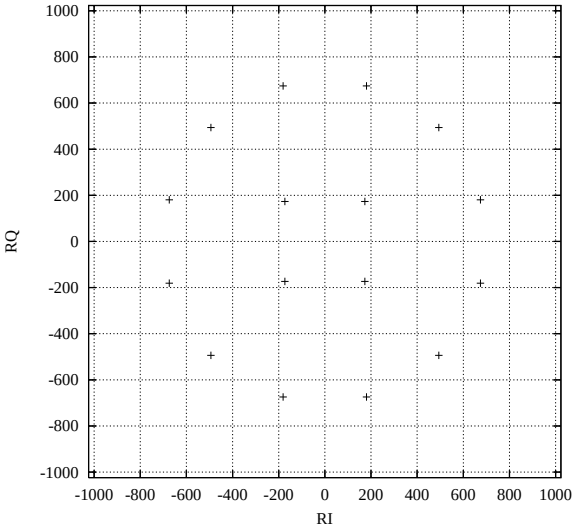

Figure 2: 16APSK 3/4 Normal Signal Set

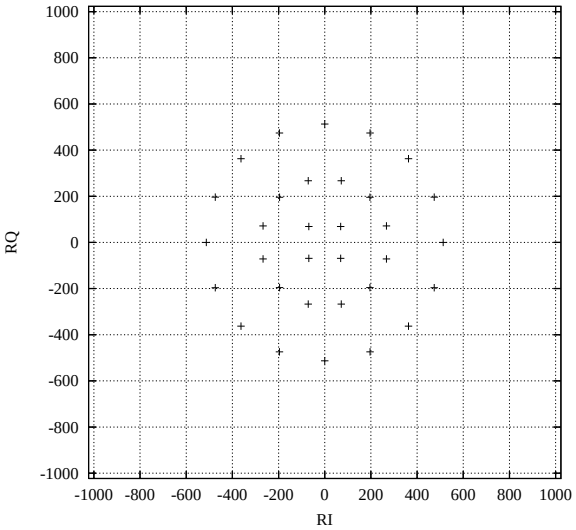

Figure 3: 32APSK 3/4 Normal Signal Set

256APSK the first three signal sets have eight rings with 32 points each, while the last two signal sets have 64 rings with 4 points each. The signal set mappings and phases are given in [1,2].

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

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

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

$$\min^*(x,y) = \min(x,y) - c\ln(1+\exp(-|x-y|/c)) \quad (7)$$

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

Table 6: Signal Sets (π/2 BPSK to 64APSK)

SS	Signal Set	$\gamma_1\ \gamma_2\ \gamma_3$
–2	π/2 BPSK SF2	
–1	π/2 BPSK	
0	QPSK	
1	8PSK	
2	4+12APSK	3.15
3	4+12APSK	2.85
4	4+12APSK	2.75
5	4+12APSK	2.7
6	4+12APSK	2.6
7	4+12APSK	2.57
8	4+12+16APSK	2.84 5.27
9	4+12+16APSK	2.72 4.87
10	4+12+16APSK	2.64 4.64
11	4+12+16APSK	2.54 4.33
12	4+12+16APSK	2.53 4.30
13	2+4+2APSK	5.32 6.8
14	2+4+2APSK	6.39 8.0
15	4+12APSK	3.7
16	4+12APSK	3.5
17	4+12APSK	3.1
18	4+12APSK	3.6
19	4+12APSK	3.2
20	4+12APSK	3.32
21	8+8APSK	2.19
22	8+8APSK	2.42588
23	8+8APSK	2.30268
24	4+12+16rbAPSK	2.85 5.55
25	4+12+16rbAPSK	2.84 5.54
26	4+12+16rbAPSK	2.84 5.26
27	4+8+4+16APSK	2.6 2.99 5.6
28	4+8+4+16APSK	2.6 2.86 5.6
29	4+8+4+16APSK	2.8 3.08 5.6
30	16+16+16+16APSK	1.88 2.72 3.95
31	8+16+20+20APSK	2.2 3.6 5.2
32	8+16+20+20APSK	2.2 3.5 5.0
33	4+12+20+28APSK	2.4 4.3 7.0

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

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

Table 7: Signal Sets (128APSK and 256APSK)

SS	Signal Set	$\gamma_1\ \gamma_2\ \gamma_3\ \gamma_4\ \gamma_5$ ...
34	128APSK	1.715 2.118 2.681 2.75 3.819
35	128APSK	1.715 2.118 2.681 2.75 3.733
36	256APSK	1.791 2.405 2.980 3.569 4.235 5.078 6.536
37	256APSK	1.794 2.409 2.986 3.579 4.045 4.6 5.4
38	256APSK	1.794 2.409 2.986 3.579 4.045 4.5 5.2
39	256APSK	1.0007 1.1139 1.1141 1.9047 1.9130 2.0370 2.0394 2.4847 2.4877 2.4979 2.5029 2.7591 2.7773 2.8729 2.8751 3.7000 3.7022 3.7413 3.7474 3.8669 3.8852 4.0000 4.0053 4.6242 4.6373 4.6419 4.6582 4.6782 4.7009 4.7085 4.7129 5.7859 5.7896 5.7911 5.8085 5.8224 5.8375 5.8456 5.8456 6.6511 6.6726 6.7349 6.7973 6.8619 6.8972 6.9239 6.9369 8.0998 8.1249 8.1707 8.2249 8.2739 8.3121 8.3360 8.3461 10.0114 10.0381 10.0841 10.1353 10.1818 10.2231 10.2481 10.2560
40	256APSK	1.0015 2.2665 2.2729 2.3085 2.3230 2.7410 2.7474 3.8768 3.9017 3.9343 3.9401 3.9442 3.9449 4.3762 4.4062 5.3246 5.3258 5.3279 5.3523 5.5064 5.5903 5.8131 5.8632 6.6476 6.6521 6.6753 6.7587 6.9021 7.0170 7.1466 7.1896 8.0365 8.0610 8.1206 8.2136 8.3269 8.4283 8.4996 8.5368 9.5550 9.5960 9.6715 9.7606 9.8496 9.9209 9.9619 9.9854 11.4273 11.4697 11.5411 11.6187 11.6893 11.7418 11.7719 11.7869 13.8441 13.8840 13.9490 14.0182 14.0777 14.1205 14.1450 14.1563

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

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 deinterinterleaved for input to the decoder. Figure 4 gives an example plot of L_3 for 32APSK 3/4 Normal. In this case, the LLR output was divided by $2^{SH+1} = 1024$ and then rounded to the nearest integer.

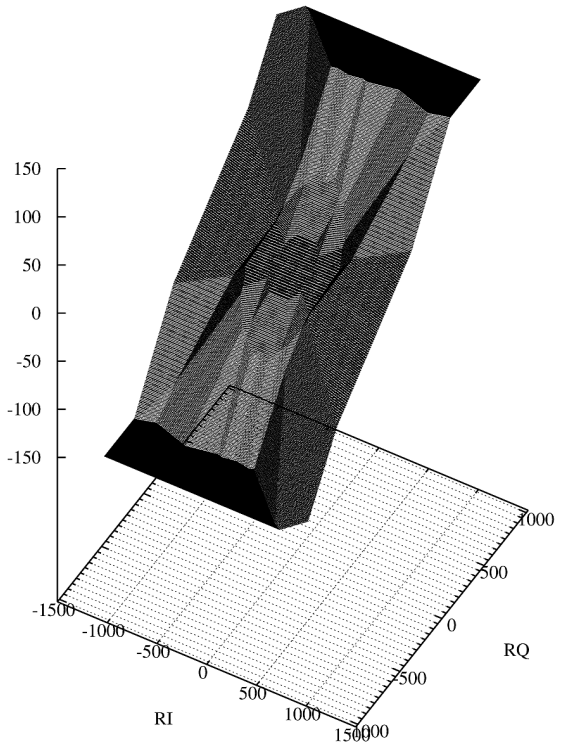

Figure 4: L_3 for 32APSK 3/4 Normal

Deinterleaving

For 8PSK and APSK modulation, the descrambled LLRs must be deinterleaved. The interleaving operation consists of writing the N coded bits into length $N_S = \lceil N/s \rceil$ columns and then reading the length s rows in a bit interleaved order which depends on the code used [1,2].

Table 8: Signal Set Bit Interleaver Patterns

SI	Pattern	SI	Pattern
–1	No bit interleaving	13	4 0 2 1 3
0	0 1 2 3 4 5 6 7	14	4 0 3 1 2
1	0 3 2 1	15	4 1 2 3 0
2	1 0 2	16	1 2 4 0 5 3
3	2 1 0 3	17	2 0 1 5 4 3
4	2 1 3 0	18	3 0 5 2 1 4
5	2 3 0 1	19	4 0 3 7 2 1 5 6
6	2 3 1 0	20	4 1 3 0 2 5 6
7	3 0 1 2	21	4 2 1 0 5 3
8	3 0 2 1	22	4 2 5 0 3 1 6
9	3 2 0 1	23	4 6 3 2 0 5 7 1
10	3 2 1 0	24	5 0 7 4 3 6 1 2
11	1 0 4 2 3	25	5 2 0 1 4 3
12	2 1 4 3 0	26	7 5 6 4 2 3 0 1

Deinterleaving is performed by writing the LLRs at the calculated interleaved address into the input RAM. For $\pi/2$ BPSK and QPSK modulation, no interleaving is performed. Table 8 gives the signal set bit interleaver patterns selected by SI, as listed in Table 3 for each of the codes. Note

that only the first s pattern values are used for each signal set.

Demapping and deinterleaving is selected with SIQ = 1. Data is input to RI[10:0] and RQ[10:0]. This applies to $\pi/2$ BPSK SF2, $\pi/2$ BPSK, QPSK, 8PSK and APSK modulation. For SIQ = 0, the deinterleaved LLRs are directly input to R[7:0] of the decoder. For SIQ = 3, the interleaved LLRs are input to R[7:0], with deinterleaving performed by the decoder. Note that for $\pi/2$ BPSK SF2, $\pi/2$ BPSK and QPSK, no deinterleaving is performed with SIQ = 3, but demapping into single LLR values for each coded bit must be performed external to the decoder, since data is input to R[7:0].

Decoder Speed

The number of LDPC decoder iterations is determined by NI, ranging from 0 to 255. NI = I–1 where I is the number of iterations. This is equivalent to 1 to 256 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 LDPC average decoder speed f_d is given by

$$f_d = \frac{F_d K}{4Iqd_c + 4} \tag{10}$$

where F_d is the CLK frequency, K is the input data length, I is the number of iterations (equal to NI+1), q is the number of circulant rows and d_c is the average check degree value. Table 4 gives the performance of the LDPC codes with F_d = 100 MHz and I = 30.

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 parity checks have been satisfied in the block. After the parity checks have been satisfied one extra iteration is performed.

Decoder Delay

The decoder delay can be separated into three parts. This is the input memory delay T_i, LDPC decoder delay T_l and the output memory delay T_o (which includes the BCH decoder delay). Each delay is equal to

$$T_i = (21S_q + N + 1)T_r \tag{11}$$

$$T_l = (26S_m + 4Iqd_c + 5 - S_r)T_c \tag{12}$$

$$T_o = (N_b/2 + 4 + d_b - 2S_x)T_x \tag{13}$$

where T_r is the RCLK period, S_q = SIQ[0], S_m = 1 if MS changes between received blocks, otherwise S_m = 0, S_r = RSYNC, T_c is the CLK period, T_x is the XCLK period, d_b = 247 is the BCH decoder delay, S_x = XSYNC, $N_b = K+mt$ is the BCH code length, K is the data length, t the number of errors the BCH decoder can correct and m is the Galois field primitive polynomial degree (14, 15 and 16 for Short, Medium and Normal codes, respectively).

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 LDPC decoder delay will vary from $T_l - T_c$ to T_l. 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 MODCOD has a different value of A, there are a total of 31 DVB–S2 and 46 DVB–S2X 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 256APSK. In order to calculate eight LLR values in eight cycles with pipelining, 32 parallel squared Euclidean distance (SED) calculators are required. This allows the calculation of 8×32 = 256 SED values.

Since each 2–D signal point is represented by a 22–bit value, the LUT has a 32×22 = 704 bit output. The total number of signal points that need to be stored is 3216. With ideal packing, the address space is $\lceil 3216/32 \rceil$ = 101. However, to reduce complexity the 16APSK signal sets are stored once per address (instead of twice) and the 8PSK and 8APSK signal sets are stored twice per address (instead of four times). This gives a total of 120 address locations. The last eight address locations of the depth 128 memory are used to store the programmable signal set values.

Each 22–bit SED value is scaled and rounded to a 15–bit value using the value of SH. A total of 127 SWAPMIN circuits are used to find the two minimum distances (for 0 or 1) used to calculate the LLR. Each SWAPMIN has four 15–bit inputs, finding two minimum values. By swapping two of the inputs and extensive use of pipelining, the LLR calculation can be efficiently performed. The last two min values are subtracted and limited to give the LLR output.

For QPSK, RI and RQ are parallel to serial converted, divided by 32 and rounded to the nearest integer. For $\pi/2$ BPSK, RI and RQ are added or

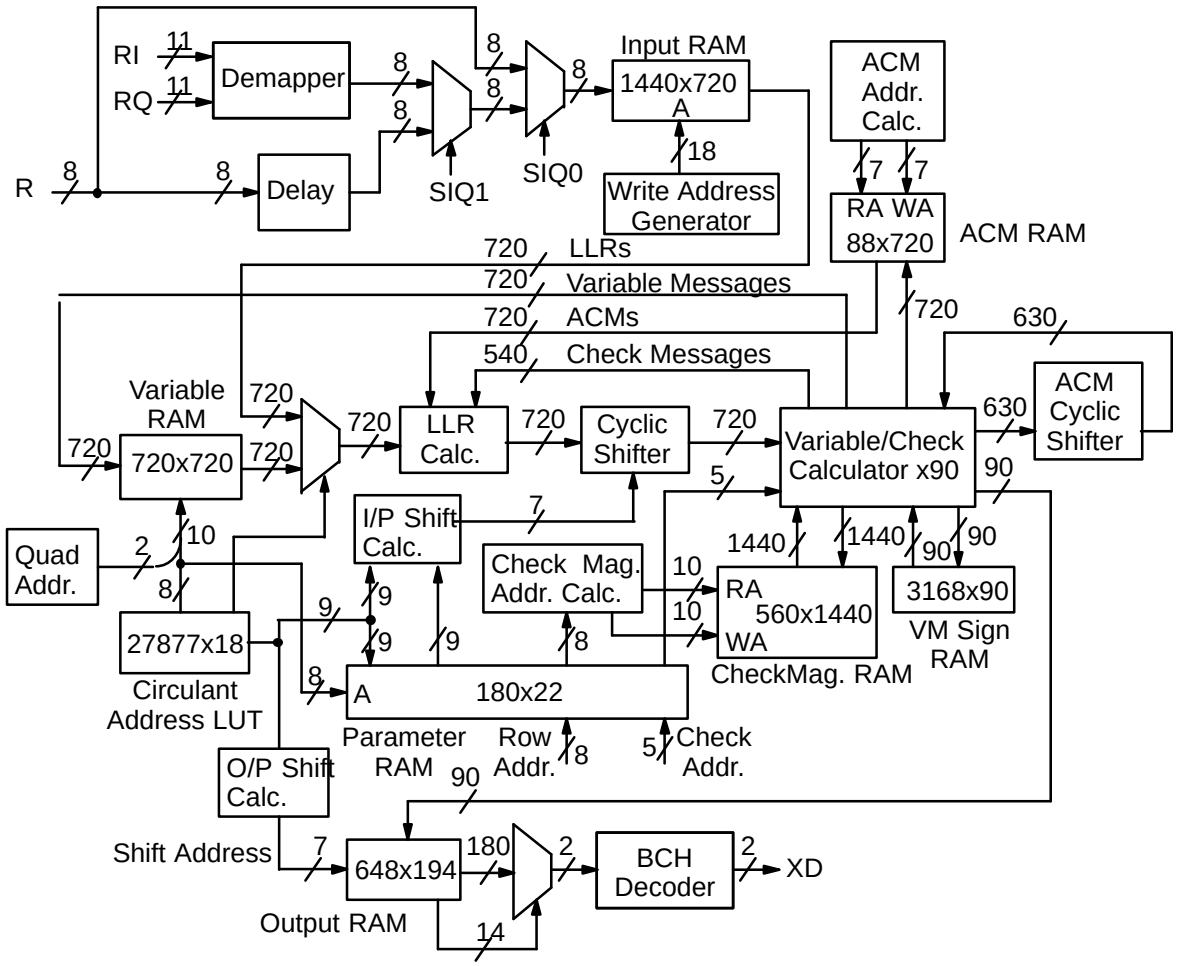

Figure 5: Simplified block diagram of LCD01D decoder.

subtracted together, divided by 64 and rounded to the nearest integer. For $\pi/2$ BPSK SF2 the two consecutive LLRs are added together, divided by 128 and rounded to the nearest integer.

LDPC Decoder Operation

Figure 5 gives a simplified block diagram of the LCD01D decoder. Each memory is implemented using a number of smaller memories that cover the various code options.

The received or demapped input data are written 8–bits at a time into one half of the Input RAM. The other half of the memory has $M/4 = 90$ 8–bit input data read into the LDPC decoder during the first iteration in each clock cycle. Input data corresponding to punctured bit positions are made equal to zero after reading.

The Circulant Address LUT is used to address the Input and Variable RAMs, reading the circulants for each parity check equation. The Circulant Address LUT is also used to rotate the 90 LLRs using a cyclic shifter. As the new variable messages (VMs) are written in cyclic shifted order to reduce complexity, the current rotation is stored in the Parameter RAM. The next time that this circulant is read, the previously stored shift is subtracted from the current shift to derotate the previous shift.

As the circulant weight can be greater than one for the DVB–S2 codes, this introduces additional complexity. A depth 88 memory is used to store the additional check messages (ACM) from circulants with weight greater than one. Other memories and logic are used to calculate the read and write addresses for this memory.

In each iteration, for each circulant row i, $0 \le i \le q-1$, the VMs and ACMs are read for $4d_c(i)$ clock cycles and after a delay of $d = 4$ clock cycles, the new VMs are written for $4d_c(i)$ clock cycles, where $d_c(i)$ can vary from $d_{c,\min}$ to $d_{c,\max}$. At the same time as the new VMs are written, the VMs and ACMs (if available) for the next circulants are read. The order in which the VMs and ACMs are read and written is carefully arranged so that the VMs and ACMs for each circulant are read after they are written.

There are 90 parallel check/variable circuits. Each circuit serially inputs one of the 90 LLRs for $4d_c(i)$ clock cycles. During this time, the previously

stored VM sign bits are read for each of the LLR inputs.

The minimum magnitude, next minimum magnitude, VM parity and the address of the minimum magnitude value is read once at the beginning of the calculation. If the address matches the $d_c(i)$ clock address, then the next minimum magnitude is output, otherwise the minimum magnitude is output.

The VM parity, VM sign bit and selected magnitude are combined to form a two's complement CM which is subtracted from the LLR to form an 8–bit VM. The $d_c(i)$ signs of the LLRs are serially exclusive–ORed (added modulo–2) to form the LLR parity check. The magnitude of the VMs (limited to 5–bits each) are used to serially find the minimum magnitude and next minimum magnitude using two comparator circuits. The $d_c(i)$ signs of the VMs are serially exclusive–ORed together to calculate the VM parity.

The magnitudes are scaled and rounded down by values ranging from MS = 0.77 to 0.97, depending on the signal set and LDPC code used. This avoids over optimistic values which reduces performance. Small programmable lookup tables (LUTs) are used to perform the scaling. If MS changes between received blocks, it takes 32 clock cycles to program the LUTs. However, due to the pipeline delay, the decoding times only increases by 26 clock cycles.

The two scaled magnitudes, VM parity and the address of the minimum magnitude are stored in the Check Magnitude RAM. This requires 16 bits each for each of the check circuits. The VM sign bits are stored in the VM Sign RAM.

If the circulant has a weight w greater than one, the compressed check messages are decompressed, rotated to match the rotation of the first rotation used to calculate the VM, added with previously calculated ACMs, and after $d_c(i)+w+1$ clock cycles written to the ACM.

After the VMs have been read, the previously stored compressed CM values needs to be added to form the LLR. The VM parity is XORed with the sign bit of the VM to produce the sign bit of the CM. This is combined with the magnitudes to produce a two's complement CM. This CM is then added to the VM to produce the new LLR. If there is also a previously calculated ACM, this is also added to the LLR. The LLR is also limited in value so that the maximum positive value is 127 and the maximum negative value is –128.

For the Check Magnitude RAM, there are three sets of read and write operations. There is one clock cycle to read the previous iteration CMs and one clock cycle to write the new CMs. This has to be done at the same time as reading $d_c(i)$ CMs for calculation of the new LLRs. However there are only $d_c(i)$ clock cycles to perform this.

Let c be the number of circulants that have the same column address between rows. As $c \geq 1$ for accumulate type LDPC codes, this means that while writing the new CMs, the new CMs can be passed directly to the LLR calculator via a multiplexer. This operation is also performed for any common circulants between rows, since the new CMs have not yet been written into memory.

For the dual read operation, a single read of the same circulant is performed. We use the property that accumulate type LDPC codes have a variable degree of 2 for the check bits. That is, we read the circulant that is common with the next row at address $d_c(i)–1$. This is used to calculate the CM to be added. The compressed CMs are then delayed and held for calculation of the CMs to be subtracted for the next row.

To ensure that the new VMs are written after the old VMs have been read, we must have $c \leq 4d_c(i)–d$. Fortunately, this property is satisfied for all codes.

On the last iteration decoded data are stored in one half of the Output RAM at CLK, along with the current shift address. Address and multiplexer select generation are then used to read the Output RAM at XCLK so as to select the decoded output in the correct order.

BCH Decoder Operation

The BCH decoder uses the Berlekamp decoding algorithm [5]. To minimise the BCH decoder memory and delay, the decoder inputs two bits from the output memory at a time. This dual–bit stream is divided in $N_b/2$ clock cycles by the BCH polynomials given in [1,2] and the remainder calculated, to give the polynomials $b_i(X)$, $1 \leq i \leq t$ of degree m, with $14 \leq m \leq 16$ depending on the code used. The output memory then outputs the first K bits in $K/2$ clock cycles for a second time. The dual–bit stream is delayed and has any errors corrected.

To calculate the error location polynomial $\sigma(X)$ of degree t, one Galois field (GF) squarer, two GF multiplier/adders and one GF adder (XOR) are used. Each multiplier/adder has the output of an adder as one of the inputs to a multiplier. A multiplier/adder is used to calculate the odd syndromes $S_{2i-1} = b_i(\alpha^{2i-1})$. The circuit for $b_1(X)$ is used to calculate α^{2i-1}. The squarer is used to calculate the even syndromes $S_{2i} = S_i^2$. One pair of syndromes S_{2i-1} and S_{2i} are calculated during each iteration

of the Berlekamp algorithm and stored in a 23x16 syndrome memory.

During each of the t iterations, $0 \le \mu \le t-1$, the inverse of the discrepancy $d_{\mu-1}^{-1} = d_{\mu-1}^{2} d_{\mu-1}^{4} \ldots d_{\mu-1}^{2^{m-1}}$ is calculated using a squarer and multiplier in the previous iteration. If the value of $\rho < \mu$, as defined in [5], is updated, the register that stores d_{ρ}^{-1} is updated with $d_{\mu-1}^{-1}$. The d_{ρ}^{-1} register is initialised with a value of 1. The value is d_{ρ}^{-1} is multiplied by d_{μ} to calculate the next partial error location polynomial $\sigma^{(\mu+1)}(X)$ and next discrepancy $d_{\mu+1}$ from $\sigma^{(\mu)}(X)$, $\sigma^{(\rho)}(X)$ and S_i, $2\mu+3-l_{\mu+1} \le i \le 2\mu+3$, were l_{μ} is the degree of $\sigma^{(\mu)}(X)$. The calculation of $\sigma^{(\mu+1)}(X)$ and $d_{\mu+1}$ is performed alternately using two multiplier/adders and an adder with $\sigma^{(\mu)}(X)$ and $\sigma^{(\rho)}(X)$ having been previously stored in memory. If ρ is not updated then $\sigma^{(\mu+1)}(X)$ is written over $\sigma^{(\mu)}(X)$. Otherwise, $\sigma^{(\mu)}(X)$ becomes $\sigma^{(\rho)}(X)$ with $\sigma^{(\mu+1)}(X)$ being written over $\sigma^{(\rho)}(X)$.

To reduce the decoding time, registers are used to keep track of the non–zero values in $\sigma^{(\mu)}(X)$ and $\sigma^{(\rho)}(X)$. In this way, each iteration requires only m clock cycles for $\mu = 0$ (the first iteration), $m+\mu-1$ clock cycles for $1 \le \mu \le t-2$, $2t-1$ clock cycles for $\mu = t-1$ and 3 clock cycles of pipeline delay. This gives a BCH decoder delay of

$$d_b = (t - 1)(m + t/2 - 1) + t + 4. \qquad (14)$$

To keep a constant delay, we use the maximum values of $m = 16$ and $t = 12$ to give $d_b = 247$. For $t < 12$ or $m < 16$, the operation of the decoder is modified so as to correctly calculate $\sigma(X) = \sigma^{(t)}(X)$.

For the last iteration, the calculation of $d_{\mu+1}$ is modified so that S_i is replaced with $T_i = \alpha^{(2^m - N_b)i}$, $1 \le i \le t$, so that each coefficient of $\sigma(X)$ becomes $\sigma_i T_i$. This is because the codes have been shortened so that $N_b < 2^m-1$. For each of the 60 different LDPC codes (some of them having the same N_b and m) a lookup table is used to store $\alpha^{2^m - N_b}$. While $b_i(X)$ is being calculated, a GF multiplier is used to calculate T_i, with the values being stored in a 12x16 RAM.

The final step is the Chien search unit. The t registers are initialised with $R_i = \sigma_i T_i$, multiplied by α^i and then again with α^i. After the dual–bit LDPC decoded data is delayed by 247 clock cycles, a dual bit correction term is XORed with the delayed data to correct the first two bits. The registers are then updated with $R_i\alpha^{2i}$, repeating for $K/2$ clock cycles to correct all the errors.

The value of the BCH error location polynomial degree is output to BED[3:0]. This indicates the number of errors the BCH decoder has corrected. If the degree of $\sigma(X)$ exceeds t, no error correction is attempted, with BED[3:0] = 0 with the BCH decoder fail signal BDF going high. Note that very few error patterns of weight greater than t can be detected using BDF. Most uncorrectable error patterns will not be detected.

Descrambling

After BCH decoding, the decoded data must be descrambled. The descrambler select DS input can be used to enable (DS=1) or disable (DS=0) descrambling.

The descrambler generator is a 15–bit right shift register with initial contents of 45200_8. The output is $e(D) = D^{14}e(D) + D^{15}e(D) = e_0 + e_1D + e_2D^2 + \ldots + e_{K-1}D^{K-1}$. When $e_i = 1$ the BCH decoder output is inverted, otherwise the output is unchanged.

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.

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. For SIQ = 0 this means RE can only be high for N clock cycles. For SIQ = 1 and 3, RE goes high for a total of N_S 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 SIQ = 0, RR will stay high until one clock cycle before the last data is input. For SIQ = 1 and 3, RR goes high s clock cycles after RE goes high, except after the last RE going high. RR will then stay high until after RE goes low. 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[15:0] is provided for reading received data from an external synchronous read input memory. Data read from the input memory must be held if RE goes low as shown in Figure 6 for SIQ = 0.

The received data finish output RF will go high at the end of each received block. For SIQ = 0, this occurs when RA = N–1. If RE goes low at RA = N–1, RF will remain high. RF will go low only after RE goes high. For SIQ = 1 and 3, RF goes high for one clock cycle s–1 clock cycles after RA changes from N_S–1 to 0 at the end of the block.

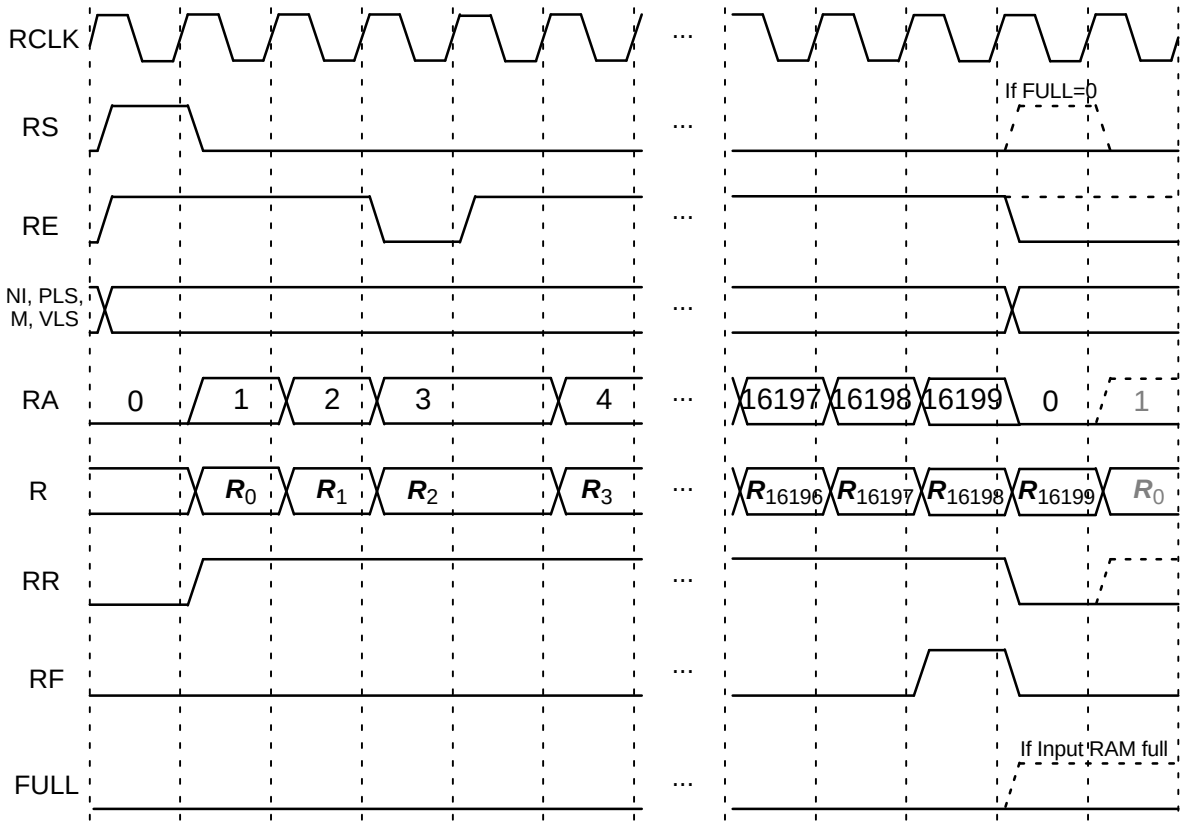

Figure 6: LDPC decoder input timing (SIQ = 0, PLS = 3, VLS = 0).

Note that for 128APSK, the last s–1 = 6 symbols has RR going high s–1 = 6 clock cycles after RE going high, except after the last RE going high. RF goes high for one clock cycle s–2 = 5 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 for 1 or 21 clock cycles after RS goes high for SIQ = 0 or 1 and 3, respectively. If this occurs, RE must be held low and the input data R or RI and RQ held while FULL is high. When FULL goes low, data can then be continue to be input.

Inputs R[7:0], RI[10:0], RQ[10:0], RS, RE, NI[7:0], PLS[0:6], VLS[3:0], DS and M must be synchronous to RCLK. Outputs RA[15:0], RR, RF and FULL are synchronous to RCLK. Internal LDPC 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. Note that RA[15] = 0 for SIQ > 0.

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[7:0], PLS[0:6], VLS[3:0], DS and M must stay constant from the time RS goes high to until after RF goes high. Note that input SIQ[0] is not internally pipelined. This parameter can only be changed until after the last received block has been decoded.

Figure 6 illustrates the decoder input timing with SIQ = 0. Each received sample $\boldsymbol{R}_i$, $0 \le i \le N$–1 represents an 8–bit sample at time i. RS is shown going high again for the case where FULL = 0.

Figures 7 and 8 illustrates the decoder input timing with QPSK and 16APSK received data, respectively. For SIQ = 1, each received value $\boldsymbol{R}_j$, $0 \le j \le N_S$–1 represents a two–dimensional sample at symbol time j input to RI and RQ. For SIQ = 3, each received value $\boldsymbol{L}_i$, $0 \le i \le N$–1 represents a LLR sample at bit time i input to R. The RCLK frequency must be at least $\lceil s \rceil$ times the 2–D symbol rate. This is because the demapper calculates one LLR for every RCLK cycle, except for $\pi/2$ BPSK SF2, where one LLR is calculated every two RCLK cycles. Note that each demapped input

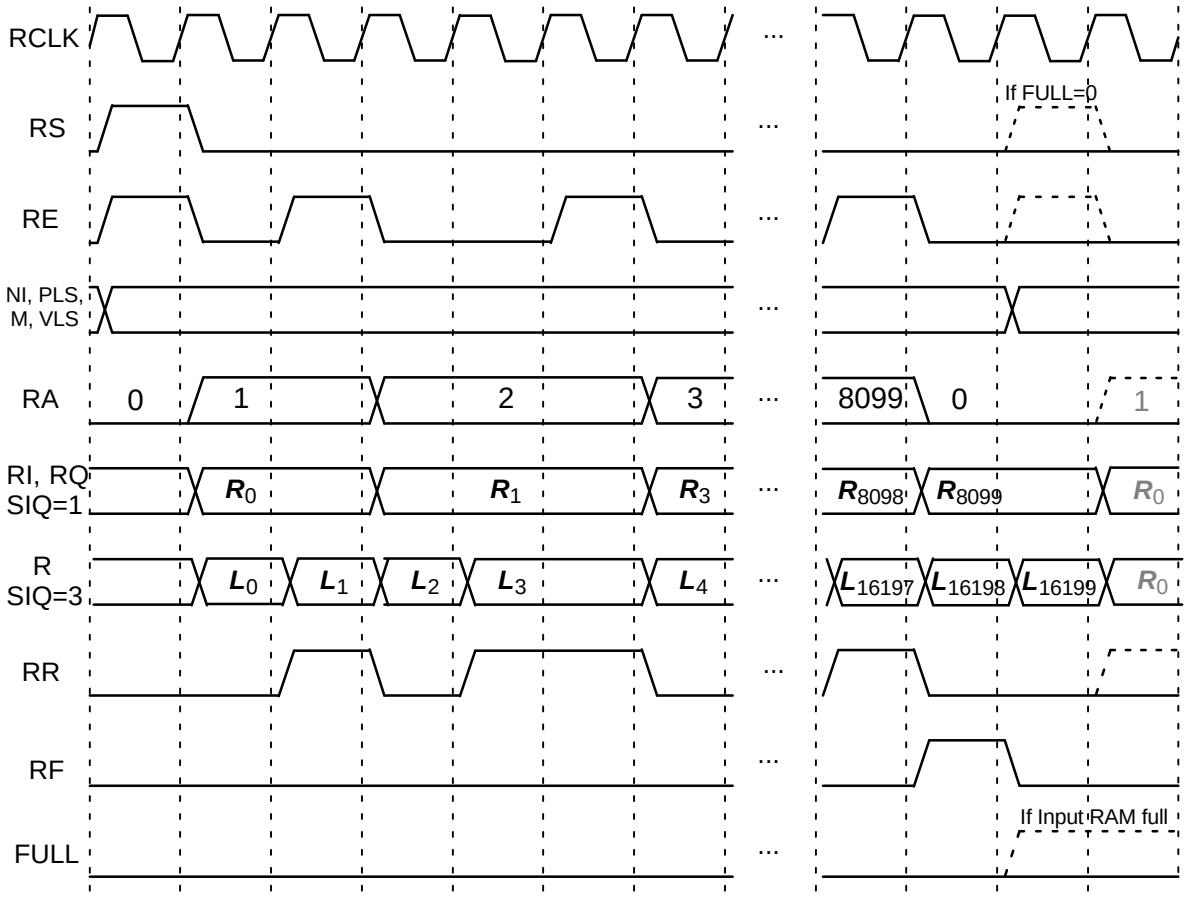

Figure 7: LDPC decoder input timing for QPSK (SIQ = 1 or 3, PLS = 3, VLS = 0).

for $\pi/2$ BPSK SF2 must be repeated, i.e., the input must be $\boldsymbol{L}_0,\boldsymbol{L}_0,\boldsymbol{L}_1,\boldsymbol{L}_1,\ldots,\boldsymbol{L}_{N-1},\boldsymbol{L}_{N-1}$.

Figure 9 illustrates the BCH decoder output timing. The LDPC decoded block is output twice from one half of the Output RAM. After a delay of T_O clock cycles of the second output, the BCH decoder corrects up to t errors in the LDPC decoded block. The signal XDR goes high for $K/2-1$ XCLK cycles while the block is output. The block is output in sequential order with address XDA[14:0].

Outputs XD[0:1], XDR, XDA[14:0] and CHK 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[0] is the first decoded bit.

The early stopping algorithm uses the LLR parity checks to determine when to stop. If all the parity checks are satisfied, the decoder will continue for one more iteration and then stop decoding. Early stopping is selected with M = 1. If M = 0, all I iterations are performed. For high SNR operation early stopping can lead to significantly reduced power consumption, since most blocks will be decoded with a small number of iterations.

Parity Check Output

The CHK output provides an indication if the parity checks were satisfied during the last iteration. This output is valid while XDR is high. Note that the check is performed before the last LLR calculation. It is not performed on the decoded output. If CHK is high this indicates the parity checks were satisfied, indicating that there are probably no errors in the decoded data. If CHK is low, this indicates the checks were not satisfied and there are probably errors in the decoded data.

Computer simulations show that the probability of a missed detection, that is the proportion of frames that have errors where CHK = 1 (checks satisfied), is very low. This means that if CHK = 1 it is very likely that there are no errors in the decoder data.

However, the probability of false detection, that is the proportion of frames that have no errors where CHK = 0 (checks not satisfied), can be high. For a low number of iterations or low SNR, the probability is very close to one. That is, nearly all frames that have no errors are falsely detected to

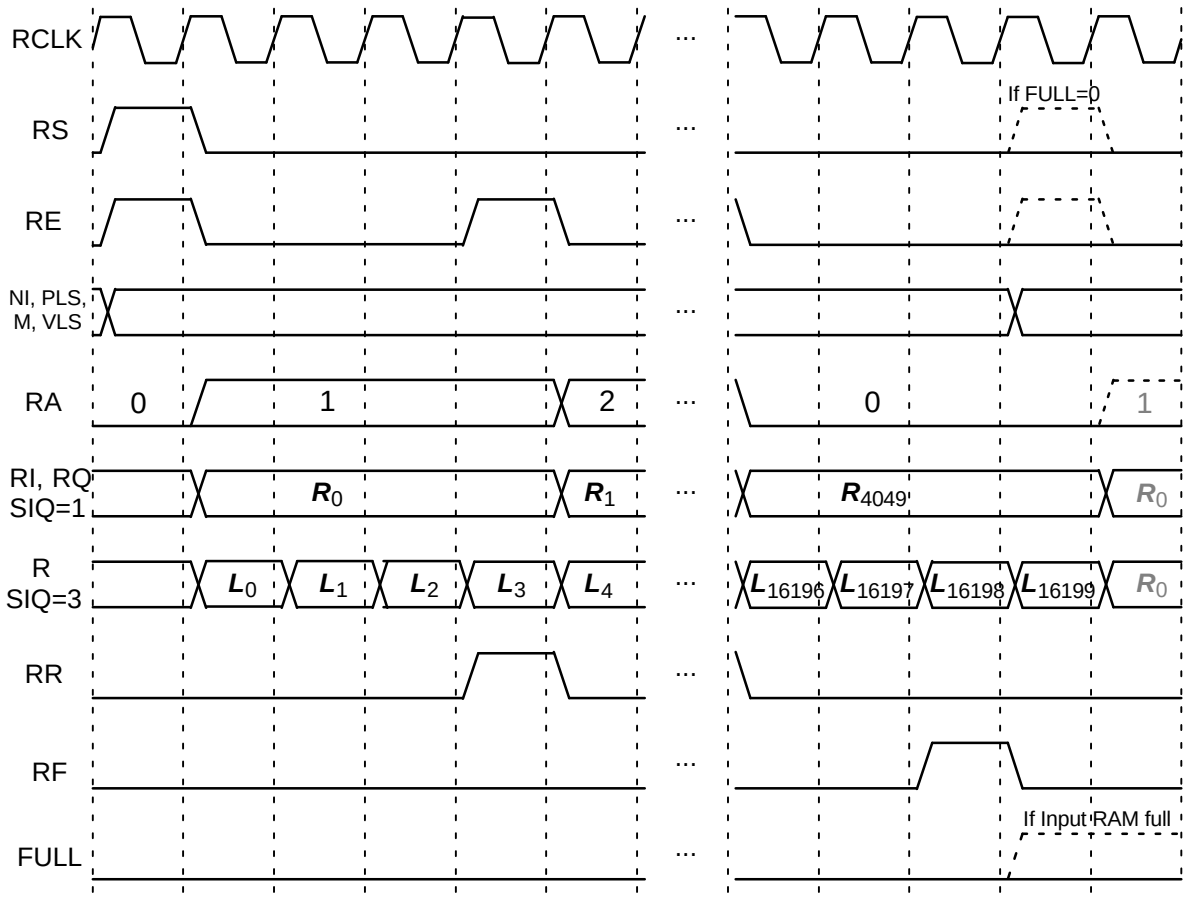

Figure 8: LDPC decoder input timing for 16APSK (SIQ = 1 or 3, PLS = 37, VLS=0).

have errors. As the number of iterations increases and the SNR increases the probability decreases to zero.

APSK Signal Set Programming

Each amplitude level in the APSK signal set can change amplitude and phase due to the effect of non–linear amplifiers. Thus, the LCD01D 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[8:0] will count from 0 to 2*S*–1, where *S* is the number of signal points. SR will go high for 2*S*–1 clock cycles. The symbol data is input in alternating 11–bit two's complement values to S[10:0], with the inphase value first followed by the quadrature value. The external LUT or RAM should be

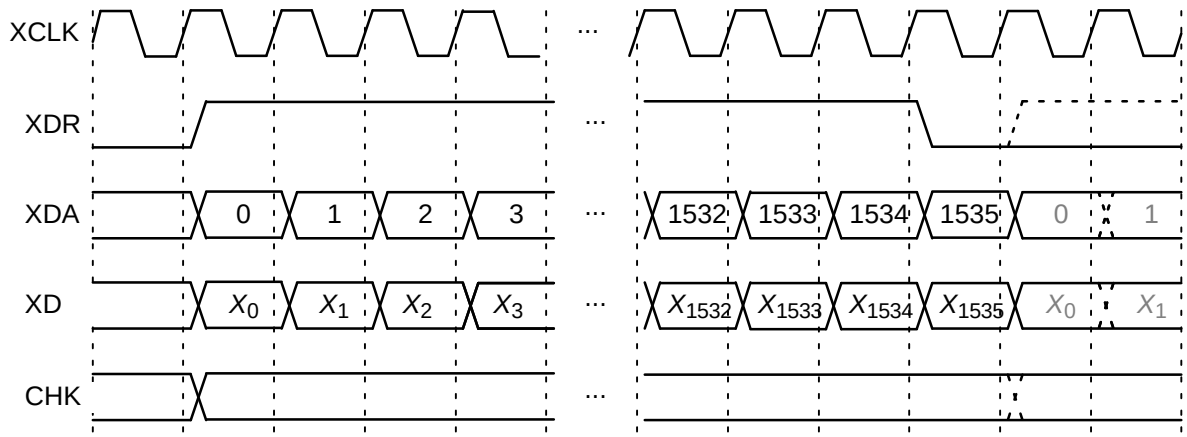

Figure 9: BCH decoder output timing (PLS = 3, VLS = 0).

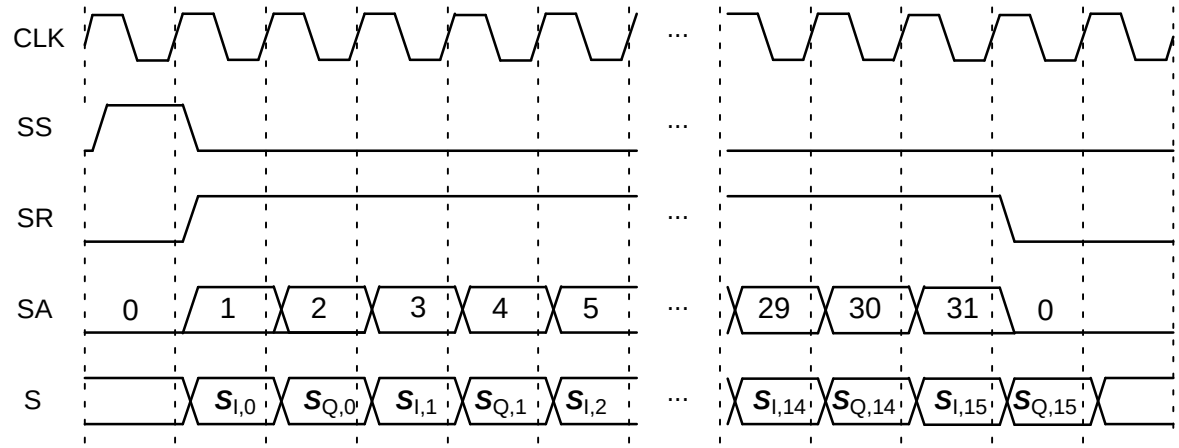

Figure 10: 16APSK signal set programming (PLS = 37, VLS = 0).

read synchronously. Figure 10 shows the programming sequence for 16APSK.

Input SM = 0 can be used to select the internal ROM with the signal set positions and bit interleaver patterns (BIP) corresponding to [1,2]. 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 PLS is changed, the signal set for the new code must be programmed.

Internally, the demapper only uses the BIP defined by SI = 0, i.e., 0 1 2 3 4 5 6 7. For other BIPs, the signal set must be remapped. This is performed for all the signal sets stored in the ROM. For example, 8PSK has the mapping 0 4 6 2 3 7 5 1 with BIP 0 1 2 [1]. For BIP 2 1 0, the mapping should be changed to 0 1 3 2 6 7 5 4.

Simulation Software

Free software for simulating the LCD01D LDPC decoder in additive white Gaussian noise (AWGN) or with external data is available by sending an email to info@sworld.com.au with "lcd01dsim request" in the subject header. The software uses an exact functional simulation of the LCD01D LDPC decoder, including all quantisation and limiting effects.

After unzipping lcd01dsim.zip, there should be lcd01dsim.exe, code.txt, ldpc.txt, apsk.txt, modcod.txt and bitint.txt. The files ldpc.txt, apsk.txt, modcod.txt and bitint.txt should not normally be edited, as they contain the LDPC code, APSK modulation, MODCOD and BIP parameters, respectively. The file code.txt contains the parameters for running lcd01dsim. 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
`ferrmax`	Number of frame errors to count
`Pfmin`	Minimum frame error rate (FER)
`Pbmin`	Minimum bit error rate (BER)
`SIQ`	Input data select (0, 1 or 3)
`PLS`	Physical Layer Signalling select (0–124)
`VLS`	Very Low SNR select (0 to 8)
`SS`	Signal set select (–2 to 40)
`SC`	LDPC code Select (0 to 59)
`SI`	Bit Interleaver Pattern select (–1 to 26)
`SH`	Signal set SED shift (6 to 12)
`MS`	Message scale (0.75 to 1.00)
`NI`	Number of iterations–1 (0 to 255)
`M`	Stopping mode (0 or 1)
`SM`	External signal set select (0 or 1)
`DS`	Descrambler select (0 or 1)
`state`	State file (0 to 2)
`s1`	Seed 1 (1 to 2147483562)
`s2`	Seed 2 (1 to 2147483398)
`out_dir`	Output directory
`in_dir`	Input directory
`read_x`	Use external information data (y or n)
`read_r`	Use external received data (y or n)

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.

For `PLS` = 0, the values for `A`, `SC`, `SI`, `SH` and `MS` given in code.txt are used. Otherwise, the values given in modcod.txt are used. The value for *t* is 12, except for SC = 15 and 18 where *t* = 10 and SC = 19 and 20 where *t* = 8.

When the simulation is finished the output is given in file c(*id*).dat for `PLS` = 0 or p(*id*).dat for `PLS` > 0, where "(*id*)" is replaced with `SC` for `PLS` = 0,

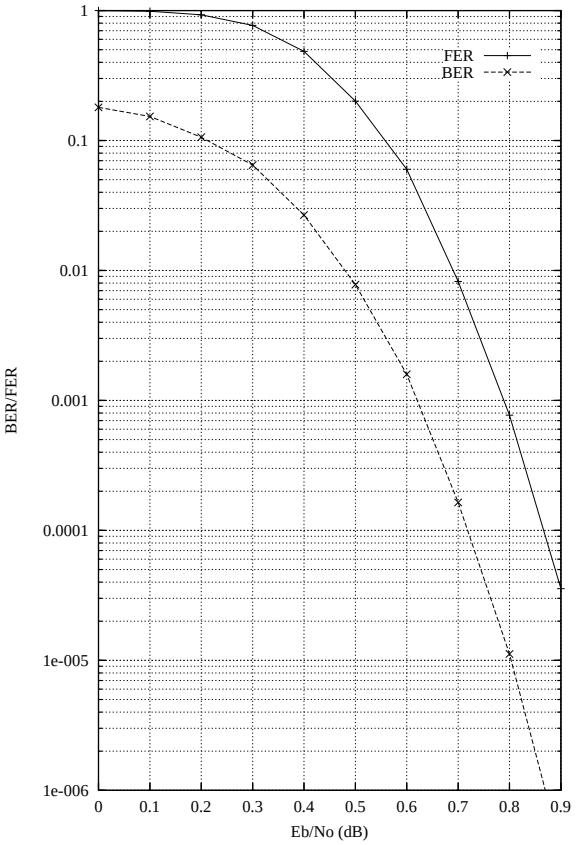

Figure 11: Rate 1/4 Short QPSK BER/FER performance with auto–stopping and 30 iterations.

with PLS for PLS > 0 and PLS ≠ 64 or 65, or with PLSVLS for PLS = 64 or 65. For example c01.dat for PLS = 0 and SC = 1, p002.dat for PLS = 2 and p641.dat for PLS = 64 and VLS = 1.

The first line gives the E_b/N_0 (Eb/No), the number of frames (num), the number of bit errors (err), the 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 *I*, na, berr, ferr, Pb, Pf, number of missed detections (miss), number of false detections (fd), missed detection rate (Pmiss) and false detection rate (Pfd) are given for each iteration.

The following file was used to give the BER and FER for PLS = 3 in Figure 11. Auto–stopping was used. When iterating is stopped early, the nasum (num*na), berr, ferr, miss and fd results at stopping are copied for each iteration to the maximum iteration number. Thus, the *I* = 30 result is the performance one would measure with auto–stopping and *NI* = 29.

```
{EbNomin EbNomax EbNoinc A}
 0.0     2.0     0.1     192
{ferrmax Pfmin Pbmin}
 256     1e-99 1e-5
{SIQ PLS VLS SS SC SI SH MS}
 1   3   0   0   0 -1 11 0.97
{NI M SM DS}
 29 1 0  0
{state s1    s2}
 0     12345 67890
{out_dir  in_dir  read_x  read_r}
 dat      input   n       n
```

The state input can be used to continue the simulation after the simulation has been stopped, e.g., by the program being closed or your computer crashing. For normal simulations, state = 0. While the program is running, the simulation state is alternatively written into state1.dat and state2.dat. Two state files are used in case the program stops while writing data into one file. To continue the simulation after the program is stopped follow these instructions:

1) Copy the state files state1.dat and state2.dat. This ensures you can restart the program if a mistake is made in configuring code.txt.
2) Examine the state files and choose one that isn't corrupted.
3) Change the state parameter to 1 if state1.dat is used or 2 if state2.dat is used.
4) Restart the simulation. The output will be appended to the existing c(*id*).dat or p(*id*).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_(*id*).dat in the directory given by in_dir, set read_x to y, e.g., x_003.dat in directory input (each line contains one bit of data). The encoded stream y_003.dat will be output to the directory given by out_dir. If SIQ > 0, the modulated symbol output will be given in ys_003.dat.

To decode data, place the received block of data in file r_(*id*).dat in directory in_dir and set read_r to y. The decoded data is output to xd_(*id*).dat in directory out_dir. For SIQ = 0 or 3, r_(*id*).dat has in each line R[*i*], *i* = 0 to *N*–1 in decimal form, e.g., the first three lines could be

```
–25
 9
 31
```

Note that for SIQ = 3 and $\pi/2$ BPSK SF2, the file must be of length 2*N*, with each line being repeated.

For SIQ = 1, r_(*id*).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_(*id*).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 PLS = 25 (8PSK with BIP 2 1 0) are

187 187
0 264
–264 0

LDPC Code Parameters

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

k	Nominal data length
n	Nominal code length
kl	LDPC data length
nl	LDPC code length
Xs	Number of shortened bits
P	Puncturing period
Xp	Number of punctured bits
M	Circulant size
q	Number of circulant rows

The second set of parameters correspond to the variable degree values specified for the data

nv	Number of different variable degrees
dv(i)	Variable degree, i = 0 to nv–1
ndv(i)	Number of degree dv(i) circulant columns

The third set of parameters gives the addresses of the parity bit accumulators, obtained from [1,2]. The first 21 codes correspond to DVB–S2 [1]. The remaining 39 codes correspond to DVB–S2X [2]. Additional LDPC codes can be added to the end of the file if desired, starting at code SC = 60.

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	Implementation 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

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.

Note that for SS = 22 and 23 (8+8APSK) and SS = 39 and 40 (256APSK), the rectangular coordinates have been converted to polar coordinates as used by the other signal sets.

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 = 41.

Bit Interleaver Pattern Parameters

The BIP parameters are given in file bitint.txt. Each row contains the various BIPs. Additional BIPs can be added to the end of this file if desired, starting at SI = 27.

MODCOD Parameters

The MODCOD and other decoder parameters are given in file modcod.txt. Each row contains the following information.

PLS	PLS code value divided by 2
VLS	VL–SNR code value
Canonical Name	Canonical MODCOD name
SS	Signal set select
SC	LDPC code select
SI	Bit interleaver pattern select
A	Average signal amplitude
SH	Signal set SED shift value
MS	Check message scale value

Note that for the Canonical Name, any spaces in the name must be replaced with an under score. Additional MODCOD codes can be added, but the PLS and VLS values must be in numerical increasing order.

The VLS code value is obtained from the Walsh–Hadamard sequences used for each code [2]. We have

$$w = vH \tag{15}$$

where $w = (w_0 \ldots w_{15})$ is the binary representation of the Walsh–Hadamard sequence, with $0 \rightarrow +1$ and $1 \rightarrow -1$, $v = (v_0 \ldots v_4)$, with v_i = VLS[*i*] for $i = 0$ to 3 and $v_4 = 0$ for VLS < 9, otherwise $v_4 = 1$, and

$$H = \begin{bmatrix} 0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0 \\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix} \quad (16)$$

where additions are performed modulo 2. For example, the dummy sequence would correspond to VLS = 9 with w = (0101010110101010).

Ordering Information

SW–LCD01D–SOS (SignOnce Site License)
SW–LCD01D–SOP (SignOnce Project License)
SW–LCD01D–VHD (VHDL ASIC License)

All licenses include Xilinx VHDL cores. The SignOnce and ASIC 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] ETSI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 1: DVB–S2," ETSI EN 302 307–1 V1.4.1, Nov. 2014.

[2] ETSI, "Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications; Part 2: DVB–S2 Extensions (DVB–S2X)," ETSI EN 302 307–2 V1.1.1, Feb. 2015.

[3] E. Yeo, P. Pakzad, B. Nikolic and V. Anantharam, "High throughput low–density parity–check decoder architectures," *Global Telecommun. Conf.*, San Antonio, USA, vol. 5, pp. 3019–3024, Nov. 2001.

[4] J. Chen and M. P. C. Fossorier, "Near optimum universal belief propagation based decoding of low–density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406–414, Mar. 2002.

[5] S. Lin and D. J. Costello, Jr., "Error control coding," 2nd Ed., Pearson Education, Upper Saddle River, 2004.

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

Revision History

- v1.00 19 Apr 2021. First release.
- v1.01 20 Apr 2021. Added description for PLS = 0 and t for code.txt.
- v1.02 21 Apr 2021. Corrected (12) and added description of MS programming time.
- v1.03 14 Sep 2021. Added E_b/N_0 and E_s/N_0 values to Table 3.
- v1.04 1 Dec. 2021. Corrected LDPC Decoder Operation.