MAP04B Features

- 4, 8, or 16 state soft–in–soft–out (SISO) maximum a posteriori (MAP) error control decoder and systematic recursive convolutional encoder
- Up to 2.5 Mbit/s decoding speed for all states
- Rate 1/2, 1/3, or 1/4 with optional punctured inputs
- Optional microprocessor interface
- 6–bit received data, 8–bit soft–in and soft–out data for information and parity bits for all rates
- 8–bit branch metric inputs for rate 1/2
- Continuous or block decoding (with or without tail)
- Programmable code polynomials
- Able to decode (n,n–4) cyclic block codes
- Minimum decoding depth (MDD) of 32 or 64
- Low decoding delay (from 66 to 257 bits)
- 17 programmable SNR's for optimum performance
- No external RAM required
- Asynchronous logic free design (except for global reset, no set or reset signals are used)
- Ideal for iterative decoding of parallel or serial concatenated "turbo" codes
- Available as BIT/MCS files for download into Xilinx XC4000 or Spartan field programmable gate arrays (FPGA), EDIF core, or Foundation schematics

Introduction

The MAP04B is a high speed maximum a posteriori (MAP) soft–in–soft–out (SISO) error control decoder with log–likelihood–ratio outputs for both the data and parity bits. A MAP decoder finds the most likely information bit to have been transmitted given a received noisy or distorted sequence, thus minimising the bit error rate (BER). This is unlike a Viterbi decoder which finds the most likely coded sequence. At low BERs, the MAP and Viterbi algorithms provide almost identical performance. However, at high BERs such as that experienced in iterative "turbo" decoders, MAP decoders can perform approximately 0.6 to 0.8 dB better than soft–output Viterbi decoder algorithms. A MAP decoder also inherently provides a soft output.

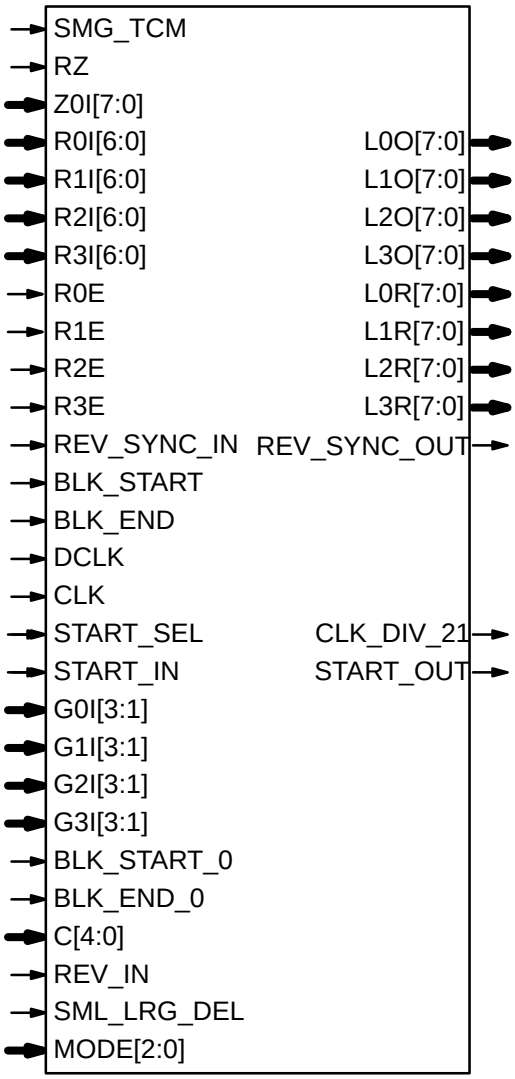


Figure 1: MAP04B schematic symbol.

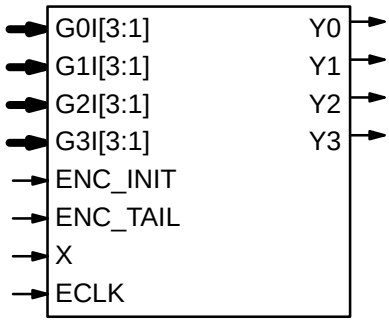


Figure 2: ENC04A schematic symbol.

Figures 1 and 2 show the schematic symbols for the MAP04B decoder and ENC04A encoder, respectively. These are the symbols that are used to compile various BIT files for download into Xilinx FPGA's. These symbols, along with an EDIF core for the ENC04A are freely available for Foundation schematics from [1]. Table 1 shows the various Xilinx parts currently supported. Support for other Xilinx parts are available on request. Consult the Xilinx data book for a list of devices, speeds, and packages currently available.

Table 1: Xilinx parts supported for MAP04B.

	Maximum Speed* (kbit/s)						
Xilinx Part	**–4**	**–3**	**–2**	**–1**	**–09**	**–08**	**–07**
XC4028EX	787 751	949 905	1526 1457	–	–	–	–
XC4036XLA	–	–	–	–	2018 1926	2268 2165	2588 2470

*Higher speed is for synchronous CLK. Lower speed is for asynchronous CLK.

Figure 3 shows the schematic symbol for the MIF04 microprocessor interface used in generating the BIT files. The interface can use the input latches/flip–flops and output buffers of Xilinx devices to minimise logic usage within the device. Up to 16 addressed inputs and 16 addressed outputs of user determined bus–width can be accessed. The bus–width used for the MAP04B is eight bits. Foundation symbols and schematics for the MIF04 are freely available from [1].

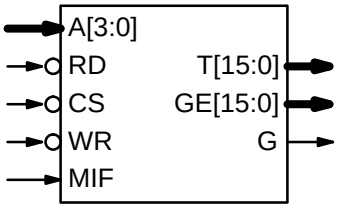


Figure 3: MIF04 schematic symbol.

Encoder

Figure 4 gives a block diagram of the 16 state systematic recursive encoder available in the ENC04A. The optional inputs are not shown. X is the data input and Y0 to Y3 are the coded outputs. GiIj = $g_i^j \in \{0, 1\}$, $0 \le i \le 3$, $1 \le j \le 3$, correspond to the code polynomial coefficients which are also used by the decoder. Data is clocked during the low to high transition of ECLK.

The encoder polynomials are defined as

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

where D is the delay operator and + indicates modulo–2 addition. It is usual practice to express the coefficients in octal notation, e.g., $g_0 = 23_8 = 10011_2 \equiv g_0(D) = 1 + D^3 + D^4$. This corresponds to G0I[3:1] = 100_2.

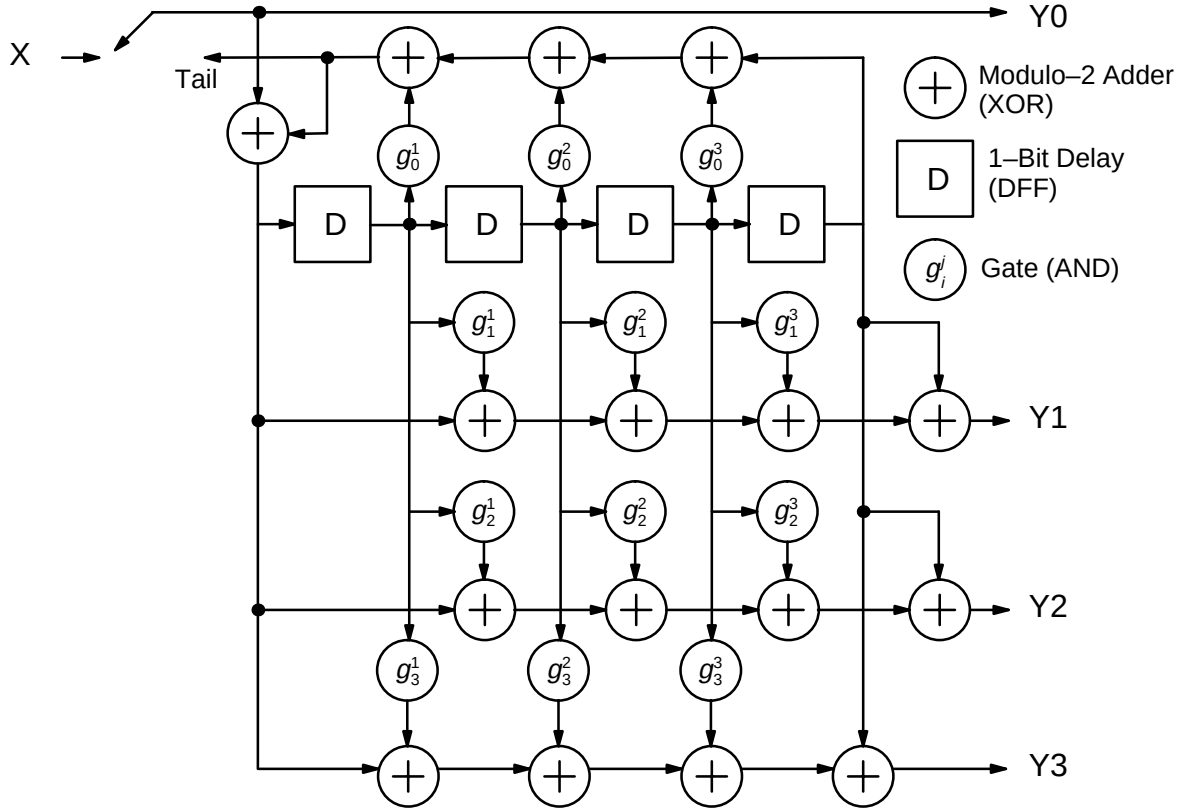


Figure 4: 16 state systematic recursive convolutional encoder.

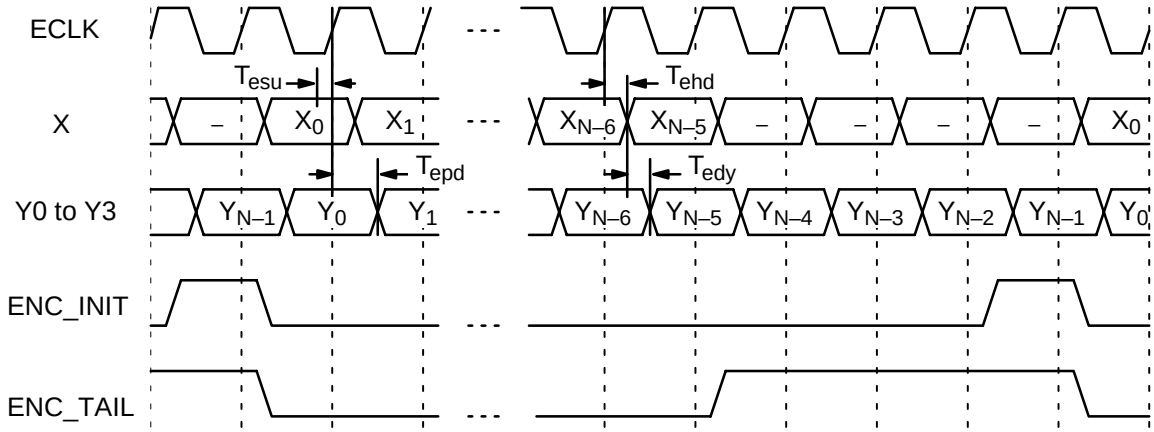


Figure 5: Encoder Timing

Figure 5 shows the timing diagram for encoding a block of data of length *N*, including four tail bits (the subscripts indicate the time index). The encoder starts and ends in state 0. The ENC_INIT signal is used to indicate the start of the block and initialises the encoder state to 0 when high and during a low to high transition of ECLK. The ENC_TAIL signal is used to force the final encoder state to zero after being held high for four ECLK cycles. While ENC_TAIL is held high, the multiplexer in Figure 4 is in the "Tail" position, i.e., the input data is not selected.

For continuous data operation ENC_INIT and ENC_TAIL should be held low.

The encoder (and decoder) can also be used for four and eight state codes. This is achieved by multiplying all code polynomials by $1+D^2$ ($1+D+D^2$ can also be used) for four state codes and $1+D$ for eight state codes. This converts the four and eight state code polynomials into 16 state code polynomials. Table 2 shows the binary code polynomial conversions for decoding four and eight state codes.

Example 1: Say a rate 1/2 eight state code has $g_0 = 15_8 = 1101_2$ and $g_1 = 17_8 = 1111_2$. Examining, the second two columns of Table 2, the equivalent 16 state code polynomials are $g_0 = 10111_2 = 27_8$ and $g_1 = 10001_2 = 21_8$. This corresponds to G0I[3:1] = 110_2 and G1I[3:1] = 000_2.

Table 2: Four and eight state code conversion

4 State	16 State	8 State	16 State
101	10001	1001	11011
111	11011	1011	11101
		1101	10111
		1111	10001

MAP Decoder

The MAP decoder is designed to be very flexible and can be operated in either continuous or block mode. Various options for reducing the decoder delay are also available.

Theory of Operation

The MAP decoding algorithm [3] finds the following likelihood ratio:

$$\lambda_k^0 = \frac{P_r(d_k = 1|R_1^N)}{P_r(d_k = 0|R_1^N)}, \qquad (2)$$

where d_k is the encoded data bit at time k (equivalent to X in Figure 4) and R_1^N is the received data from time 1 to N. The basic algorithm for an additive white Gaussian noise (AWGN) channel involves exponentials, multiply–add, and division operations. By taking the negative logarithm of (2), the exponentials disappear, the multiplies becomes additions, the divisions become subtractions, and the additions become the E operand defined below:

$$a \text{ E } b = \min(a, b) - f(|a - b|) \qquad (3)$$

where

$$f(z) = c\ln(1 + e^{-z/c}) \qquad (4)$$

and c is a constant dependent on the signal amplitude and noise variance. The function $f(z)$ is implemented as look up tables within the decoder. For the MAP04B, there are 17 integer values of c, from 0 to 17 via the C inputs. C should be equal to the closest integer to c. For example, if $c = 7.442$, then C[4:0] = $7 \equiv 00111_2$. Values of C greater than 17 are limited to 17. Due to quantisation effects, C = 1 is equivalent to C = 0.

When C = 0, $f(z) = 0$ and the sub–MAP (also known as max–log–MAP) algorithm is implemented [4]. The sub–MAP algorithm does not require knowledge of the signal to noise ratio (SNR),

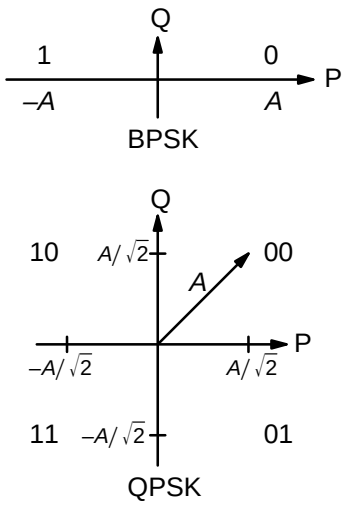


Figure 6: BPSK and QPSK signal sets.

but does not perform as well as the MAP algorithm.

Note that when a change in C is detected on a low–to–high transition of DCLK, the decoder takes up to 68 CLK cycles for the new look–up tables to be loaded. During this time, the lookup table outputs are forced to zero, equivalent to sub–MAP operation. After loading, the lookup table outputs are enabled. If a change in C is detected during loading, the whole operation starts again.

We define the log–likelihood ratio as

$$L_k^0 = -c\ln\lambda_k^0. \tag{5}$$

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

$$R_k^i = A((1-2y_k^i)/\sqrt{m} + n_k^i) \tag{6}$$

where A is the signal amplitude, $y_k^i \in \{0,1\}$, $i = 0$ to 3 correspond to the coded bits, $m = 1$ for BPSK or $m = 2$ for QPSK, and n_k^i is a Gaussian distributed random variable with zero mean and normalised variance σ^2. For a systematic code $y_k^0 = d_k$. Figure 6 shows the signal sets for BPSK and QPSK. We have

$$\sigma^2 = \left(2mR\frac{E_b}{N_0}\right)^{-1} \tag{7}$$

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

Since a zero is transmitted as $+A/\sqrt{m}$ and a one is transmitted as $-A/\sqrt{m}$ the sign bit of a noiseless R_k^0 in two's complement notation is equal to d_k. Similarly, (2) is formed so that the sign bit of L_k is equal to the estimate of d_k, i.e.,

$$\hat{d}_k = \begin{cases} 0 : L_k \geq 0 \\ 1 : L_k < 0 \end{cases}. \tag{8}$$

For optimal performance with BPSK or QPSK, the constant c should be adjusted such that

$$c = \sigma^2\sqrt{m}A/2. \tag{9}$$

Due to quantisation and limiting effects the value of A should also be adjusted according to the received signal to noise ratio. A program for calculating the optimum values of A and c is freely available from [2].

The value of A directly corresponds to the 6–bit signed magnitude inputs (described in more detail later). The 6–bit inputs have 63 quantisation regions with a central dead zone. The quantisation regions are labelled from –31 to +31. For example, one could have A = 15.7. This value of A lies in quantisation region 15 (which has a range between 15 and 16).

Example 2: Rate 1/3 BPSK code operating at E_b/N_0 = 0.3 dB. From (7) we have σ^2 = 1.39988. Assuming c = 11 we have from (9) that A = 15.7.

Example 3: Rate 1/2 QPSK code operating at E_b/N_0 = 0.8 dB. From (7) we have σ^2 = 0.41588. Assuming c = 7 we have from (9) that A = 23.8. Note that the amplitude in each dimension is $A/\sqrt{2}$ = 16.8.

A priori information corresponding to the information bit can also be input to the MAP decoder. This input corresponds to

$$Z_k^0 = -c\ln\left(\frac{P_r(d_k=1)}{P_r(d_k=0)}\right), \tag{10}$$

and is Z0I in the MAP04B. The output L_k^0 (corresponding to L0O) can be expressed as

$$L_k^0 = Z_k^0 + R_k^0 + Z_k^{0'}, \tag{11}$$

where $Z_k^{0'}$ is the extrinsic information.

The MAP04B also provides a log ratio output for the parity bits y_k^j, $1 \leq j \leq 3$ (equivalent to Yj in Figure 4). We define this as

$$L_k^j = -c\ln\frac{P_r(c_k=j|R_1^N)}{P_r(c_k=0|R_1^N)} \tag{12}$$

which corresponds to LjO. Similarly to L_k^0, we can express L_k^j as

$$L_k^j = Z_k^j + R_k^j + Z_k^{j\prime}, \tag{13}$$

where Z_k^j is the *a priori* information for y_k^j, R_k^j is the received noisy parity symbol (RjI), and $Z_k^{1\prime}$ is the extrinsic information for y_k^1.

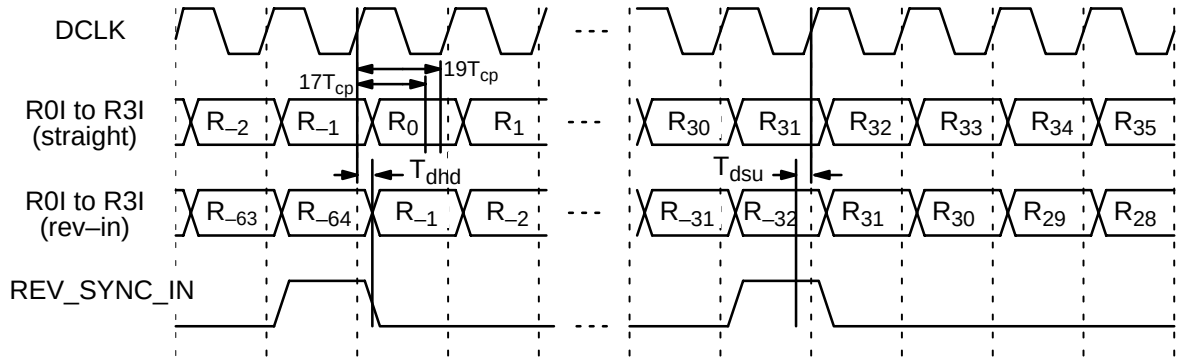


Figure 8: Decoder Input Timing (MDD = 32)

Decoder Operation

The MAP04B is a sliding block decoder [5] and is thus able to continuously decode data (just like a Viterbi decoder). When SML_LRG_DEL is low or high the minimum decoding depth (MDD) is 32 or 64, respectively. The decoding delay is 129 or 257 bits, respectively. For best performance an MDD of 64 should be chosen, especially when punctured codes are used. For minimum delay, MDD of 32 can be chosen.

To reduce the decoding delay, the decoded output is available in time reversed blocks of MDD. Choosing the L0R to L3R decoded outputs (instead of L0O to L3O) reduces the delay to 97 and 193 bits for MDD of 32 and 64, respectively. The L0O to L3O outputs are the L0R to L3R outputs time reversed in blocks of MDD. The REV_SYNC_OUT signal indicates the end of the block in time. Figure 7 illustrates the output timing.

The delay can be further reduced by inputting data (R0I to R3I, Z0I, R0E to R3E, BLK_ START, and BLK_END) in time reversed blocks of MDD. With time reversed inputs and outputs the delay is reduced to 66 and 130 bits for MDD of 32 and 64, respectively. Table 3 summarises the various delay options.

The REV_IN input when high indicates that the input data is reversed and the REV_SYNC_IN signal is used to indicate the end of the block in time. Figure 8 illustrates the input timing. The REV_SYNC_IN signal can still be used even if REV_IN is low. This allows synchronisation of the REV_SYNC_OUT output signal.

For straight–in operation, R0I to R3I, R0E to R3E, Z0I, BLK_ START, and BLK_END must be valid between 17 and 19 CLK cycles after the low–to–high transition of DCLK. For reverse–in operation, these inputs must be valid during the low–to–high transition of DCLK.

The outputs L0O to L3O, L0R to L3R, and REV_SYNC_OUT change value from five to seven CLK cycles after the low–to–high transition of DCLK.

Table 3: Decoder Delay Options (in bits)

MDD	straight	rev–in	rev–out	rev–in/ rev–out
32	129	98	97	66
64	257	194	193	130

Block Operation

The decoder is also able to decode blocks of data with the same low decoder delay as in continuous mode. The static inputs BLK_START_0 and BLK_END_0 when high indicate whether the block starts or ends in state 0. When low the decoder assumes that the block starts or ends in an unknown state. The signals BLK_START and BLK_END indicate the start and end of the block in time. When these signals go high the forward

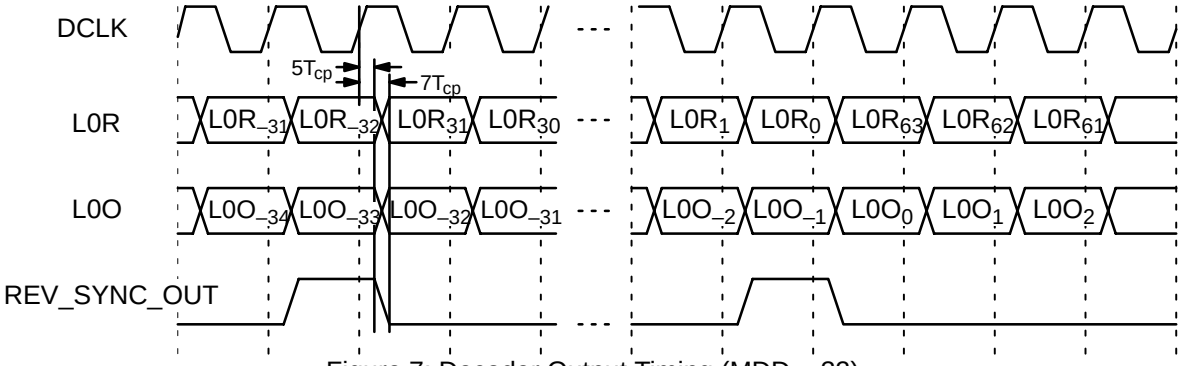


Figure 7: Decoder Output Timing (MDD = 32)

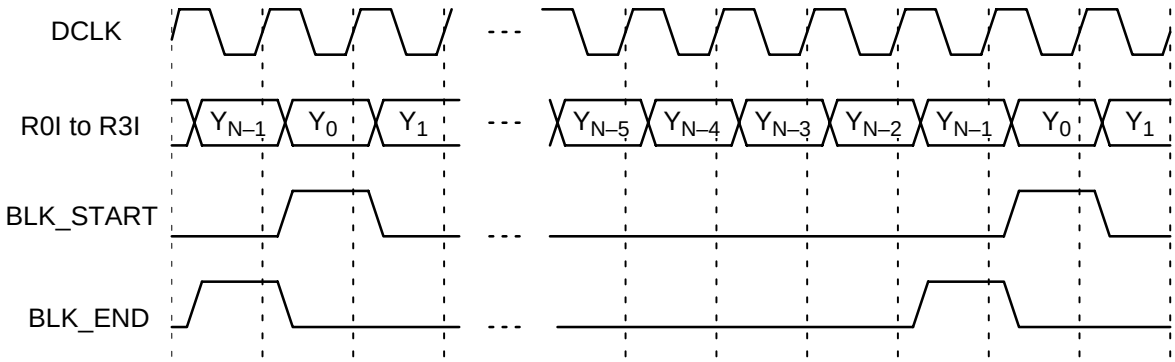


Figure 9: Decoder Block Timing

and reverse state metrics of the MAP algorithm are initialised to their appropriate starting values. Figure 9 illustrates the timing for block encoded data. Note that reverse inputs and outputs can also be used with block encoded data so as to reduce the decoder delay.

Block codes with four parity bits can also be decoded. For these codes Y1 to Y3 are not used and R1I to R3I are set to 0. The divisor polynomial g_0 should be a primitive polynomial such as $g_0 = 1 + X^3 + X^4$ (G0I[3:1] = 100). Block encoding and decoding is performed as normal (the encoder and decoder states must be made to both start and end in state zero). The (15,11) BCH code is one block code that can be near–optimally decoded. Shorter or longer block lengths are also possible.

Clock Operation

When START_SEL = 0 the CLK input is an external clock provided to the decoder that must be at least 22 times greater in frequency than DCLK, the received data clock. CLK can be asynchronous to DCLK. For example, if CLK = 45 MHz the maximum value of DCLK is 2.0 MHz. DCLK can vary from 0 to its maximum value in frequency. The minimum low or high period of DCLK must be greater than the CLK period.

When START_SEL is high, CLK is 21 times the frequency of DCLK. Figure 10 shows how a phase locked loop (PLL) can be used to generate CLK. The phase detector must be able to synchronise the rising edges of CLK and DCLK. The minimum DCLK high or low period is the same as for ECLK (see Switching Characteristics section).

Figure 11 shows the timing of the various signals. For example, if DCLK is 2.048 MHz, the PLL must be able to generate a rising edge synchronous CLK signal with a frequency of 43.008 MHz. The START_OUT signal from a decoder with a PLL can be used to drive the START_IN inputs of other decoders without a PLL.

Note: The high time of CLK must not exceed one millisecond for XC4000E devices. Holding CLK high for more than 1 ms could result in excessive current or damage to the FPGA.

Data Formats

When RZ is low, the seven bit received data R0I to R3I can be in signed magnitude or two's complement format. In signed magnitude format (SMG_TCM low), the most significant bit (i.e., RiI6, i = 0 to 3) corresponds to the sign bit and the remaining bits represent the magnitude. When SMG_TCM is high the input data is in two's complement format. Table 4 gives a partial listing of the range represented by each quantised value.

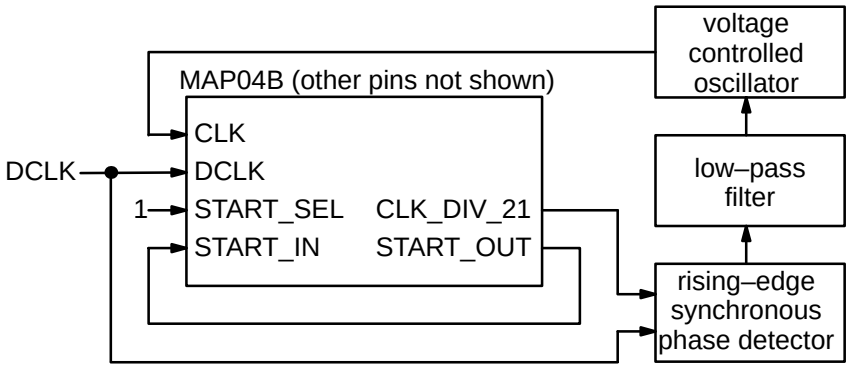


Figure 10: Phase locked loop operation.

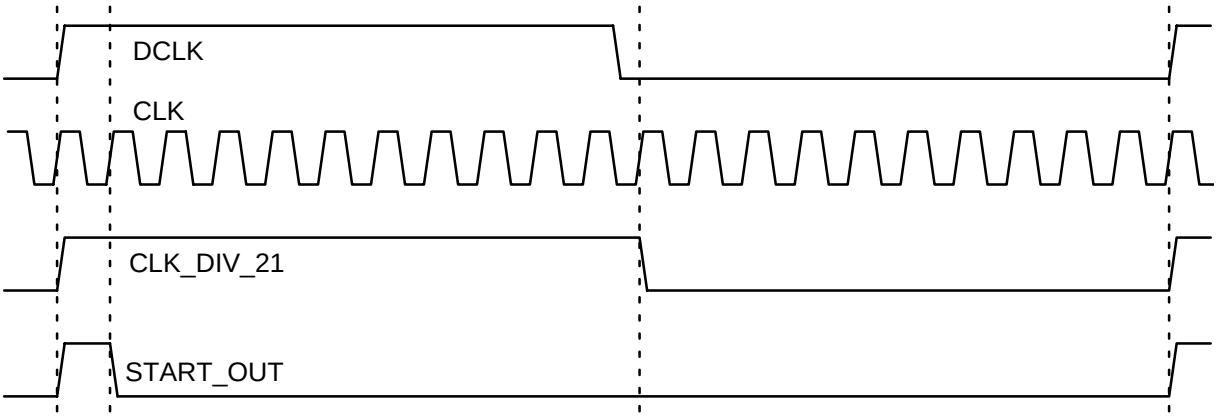


Figure 11: Phase locked loop timing.

Internally, the decoder converts the seven bit R0I to R3I values to six bit signed magnitude quantisation with a central dead zone. Table 5 shows the six bit internal quantisation ranges. For external six bit signed magnitude data with central dead zone quantisation, the data is input to RiI[6:1] with RiI0 = 0. For six bit two's complement data with central dead quantisation, the data is input to RiI[6:1] with RiI0 = RiI6.

When RZ is high RiE,RiI[6:0] for i = 1 to 3 are equal to eight bit two's complement inputs with notation Z1I[7:0] to Z3I[7:0]. The eight bit Z0I to Z3I inputs, L0O to L3O, and L0R to L3R outputs are in two's complement arithmetic. Table 6 gives a partial listing of the range represented by each quantised value.

Internally, Z0I is added to R0I to produce an eight bit signed magnitude value. When RZ is low, the seven bit R1I to R3I inputs are converted to six bit signed magnitude values with a central dead zone. When RZ is high, the eight bit Z1I to Z3I inputs are converted to eight bit signed magnitude values. This allows the decoder to iteratively decode serially concatenated convolutional codes. In this case, the inner decoder has RZ low and the outer decoder has RZ high. For iteratively decoding parallel concatenated codes, RZ is low for both the inner and outer decoders.

For rate 1/2 codes and MODE = 7 (see next section), RiE,RiI[6:0] for i = 0 to 3 are equal to eight bit branch metric (BM) inputs with notation BM0I[7:0] to BM3I[7:0]. RZ, SMG_TCM, and Z0I inputs are not used. The BM inputs correspond to the BM's for symbols $2y^1 + y^0$, e.g., the BM for $y^1 = 1$ and $y^0 = 0$ corresponds to BM2I. The BM's range in value from 0 to 255. The lower the BM value, the more likely the symbol. To maximise performance, the smallest BM should be subtracted from all four BM's (implying that the smallest BM will then be equal to zero).

The branch metric inputs can be used to decode signal sets other than BPSK and QPSK, e.g., rate 1/2 16QAM. For these signal sets, the branch metrics are not linear to the received signal. Non–linear computations or look–up tables can be used to calculate the branch metrics for these signal sets.

Table 4: Quantisation for R0I to R3I.

Sign Mag.	Two's Comp	Range
63	63	63↔∞
62	62	62↔63
⋮	⋮	⋮
1	1	1↔2
0	0	0↔1
64	127	–1↔0
65	126	–2↔–1
⋮	⋮	⋮
126	65	–63↔–62
127	64	–∞↔–63

Table 5: Internal quantisation for R0I to R3I.

Decimal	Binary	Range
31	011111	30.5↔∞
30	011110	29.5↔30.5
⋮	⋮	⋮
2	000010	1.5↔2.5
1	000001	0.5↔1.5
0	000000	–0.5↔0.5
32	100000	–0.5↔0.5
33	100001	–1.5↔–0.5
34	100010	–2.5↔–1.5
⋮	⋮	⋮
62	111110	–30.5↔–29.5
63	111111	–∞↔–30.5

Table 6: Quantisation for Z0I to Z3I, L0O to L3O, and L0R to L3R.

Decimal	Binary	Range
127	01111111	127↔∞
126	01111110	126↔127
⋮	⋮	⋮
1	00000001	1↔2
0	00000000	0↔1
255	11111111	–1↔0
254	11111110	–2↔–1
⋮	⋮	⋮
129	10000001	–127↔–126
128	10000000	–∞↔–127

Punctured Code Operation

When RZ is low, the input signals R0E to R3E are used to enable the received data inputs R0I to R3I, respectively. Data is erased when R0E to R3E are low for use in punctured code decoding. Manual puncturing can be performed by forcing R0I[5:0] to R3I[5:0] low.

Example 4: Rate 1/2 code punctured to rate 2/3. In this case the systematic bit is not punctured (R0E high) while the parity bit is punctured every second bit (R1E alternates between high and low). R1E is equal to DCLK divided by two. The phase of R1E needs to match the received data so that when R1E is low, the parity information is punctured.

Mode Selection

To minimise the decoder complexity, the MODE[2:0] inputs can be used with the schematic symbols to select only those rates and input types that are expected to be used. Table 7 lists the ten possible modes of operation.

For modes 0 to 7, MODE[2:0] is directly equal to the mode number, e.g., for mode 4, MODE[2:0] = 100_2. These inputs should be connected to internal VCC and GND supplies. Connecting the inputs to pads will result in excessive configurable logic block (CLB) usage and decreased decoder speed. Mode 2 and mode 4 are equivalent to the previously available MAP04 and MAP04A, respectively.

For mode 8, MODE2 should be connected to an input pad while MODE1 and MODE0 should be connected to VCC. MODE2 can then be used to select between R inputs with $n = 4$ (MODE2 low) and BM inputs with $n = 2$ (MODE2 high).

For mode 9, MODE0 should be connected to an input pad while MODE1 and MODE2 should be connected to VCC. MODE0 can then be used to select between R or Z inputs with $n = 4$ (MODE0 low) and BM inputs with $n = 2$ (MODE0 high).

For R inputs, only L0O and L0R are output. For R,Z inputs all the outputs are provided corresponding to the number of inputs. Table 8 shows the various inputs and outputs that are available for BIT/MCS files for the ten different modes. Both the encoder and decoder use the same G inputs.

CLB(L) indicates the number of CLB's used for both the encoder and decoder with MDD = 32 only. CLB(H) indicates the number of CLB's with MDD = 32 or 64.

Table 7: Mode selection

Mode	*n*	Input	CLB(L)	CLB(H)
0	1	R	687	871
1	2	R	731	942
2	3	R	774	1011
3	4	R	827	1088
4	2	R,Z	848	1076
5	3	R,Z	1018	1286
6	4	R,Z	1175	1487
7	2	BM	771	1058
8	4,2	R,BM	911	1194
9	4,2	R,Z,BM	1217	1539

Microprocessor Interface

The MIF04 microprocessor interface allows a microprocessor or digital signal processor to write and read data to a Xilinx FPGA using a data and address bus. To maximise flexibility the MIF04 consists of only the address generation logic. The various input latches/flip–flops and output buffers must be added separately. To minimise CLB usage, we have incorporated the input latches/flip–flops and output buffers into the input/output blocks (IOB) of the Xilinx device.

Note that RD, CS, and WR are active low signals. The address bus A[3:0] shown in Figure 3 is used to select one of 16 inputs or one of 16 outputs. The address selected enables one of the GE[15:0] outputs from the MIF04 if CS is low and MIF is high. This allows data to be written to the selected latches when WR goes low (the G output of MIF04 is the inverse of the WR input). Figure 12 shows an example of how R0E is connected.

If CS and RD are both low and MIF is high the address selected enables one of the T[15:0] outputs from the MIF04. This allows the selected data to be read from the selected output buffers. When CS and MIF are both high all the inputs and outputs are disabled. Figure 12 shows an example of how Y0 is connected. Figure 13 shows the timing for the MIF04.

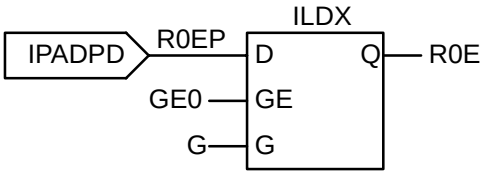
Table 8: Inputs and outputs selected for BIT/MCS files for different modes

Inputs and Outputs	Mode									
	0	1	2	3	4	5	6	7	8	9
BLK_END, BLK_END_0, BLK_START, BLK_START_0, CLK	●	●	●	●	●	●	●	●	●	●
CLK_DIV_21, C[4:0], DCLK, ECLK, ENC_INIT, ENC_TAIL	●	●	●	●	●	●	●	●	●	●
RESET, REV_IN, REV_SYNC_IN, REV_SYNC_OUT	●	●	●	●	●	●	●	●	●	●
SML_LRG_DEL*, START_IN, START_OUT, START_SEL, X	●	●	●	●	●	●	●	●	●	●
DCLKM, ECLKM, A[3:0], RD, CS, WR, MIF	●	●	●	●	●	●	●	●	●	●
G0I[3:1], R0E, R0I[6:0], Y0	●	●	●	●	●	●	●	●	●	●
G1I[3:1], R1E, R1I[6:0], Y1		●	●	●	●	●	●	●	●	●
G2I[3:1], R2E, R2I[6:0], Y2			●	●		●	●		●	●
G3I[3:1], R3E, R3I[6:0], Y3				●			●		●	●
L0O[7:0], L0R[7:0]	●	●	●	●	●	●	●	●	●	●
L1O[7:0], L1R[7:0]					●	●	●			●
L2O[7:0], L2R[7:0]						●	●			●
L3O[7:0], L3R[7:0]							●			●
MODE0/MODE2									●	●
RZ					●	●	●			●
SMG_TCM, Z0I[7:0]	●	●	●	●	●	●	●		●	●

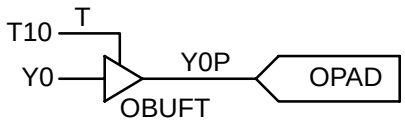
* Not connected for MAP0BmL BIT files (MDD = 32)

If the microprocessor interface is not used then the MIF input should be low. This causes all the input latches to be transparent and all the output buffers to be enabled. The separate inputs and outputs can then be used. When MIF is high the microprocessor interface is enabled. The various inputs and outputs must then be externally connected together to form the data bus.

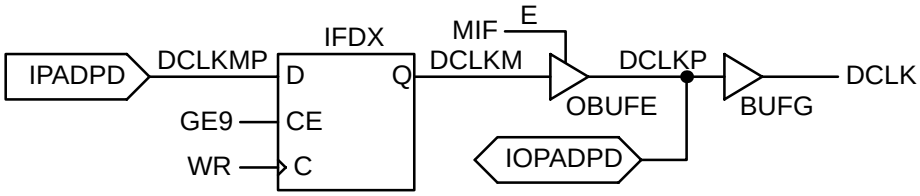
Edge sensitive inputs such as DCLK and ECLK cannot use latches. This is because the data could be changing before the low to high transition of WR (or high to low transition of G),



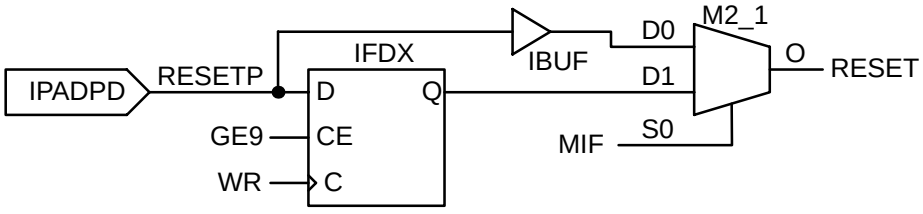
Normal Input Configuration



Output Configuration



Edge Sensitive Input Configuration



Asynchronous Input Configuration

Figure 12: Example input/output configurations for MIF04.

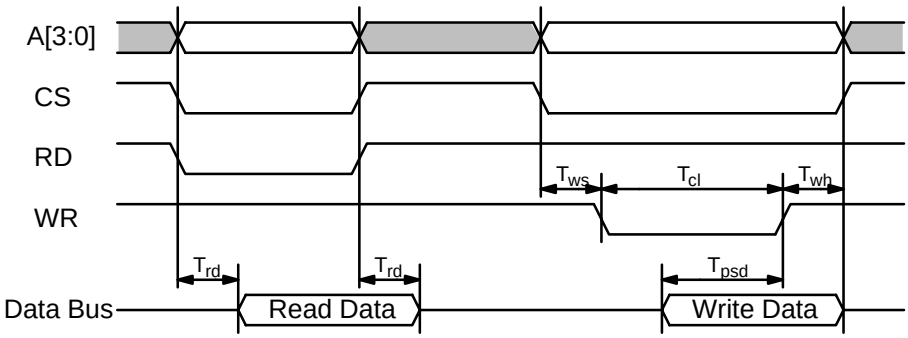


Figure 13: Microprocessor interface timing.

causing false clocking. The circuit shown in the Figure 12 shows how DCLK and similarly ECLK should be connected. When MIF is low, the DCLK and ECLK signals should be fed to the DCLK and ECLK pads. When MIF is high, the DCLKM and ECLKM pads should be connected to the data bus and the DCLK and ECLK pads be left unconnected. The input flip–flops for the DCLKM and ECLKM pads ensure that the internal DCLK and ECLK signals only change during the low to high transition of WR.

Since RESET is an asynchronous input, it also needs to be clocked during the low to high transition of WR. However, RESET does not need a dedicated global buffer like the clock signals, thus there is no need to add an additional pad. Figure 12 shows how RESET should be connected.

Table 9 shows the address locations for the various inputs and output signals. Note that the order of signals for each address is arbitrary, although we recommend users follow the order given (from most significant data bit to the left to least significant data bit to the right).

Table 9: Microprocessor Interface Address Table

Address	Inputs	Outputs
0	R0E, R0I[6:0]	L0O[7:0]
1	R1E, R1I[6:0]	L1O[7:0]
2	R2E, R2I[6:0]	L2O[7:0]
3	R3E, R3I[6:0]	L3O[7:0]
4	Z0I[7:0]	L0R[7:0]
5	G1I[3:1], G0I[3:1]	L1R[7:0]
6	G3I[3:1], G2I[3:1]	L2R[7:0]
7	MODE0/MODE2, RZ, SML_LRG_DEL, SMG_TCM, START_SEL, BLK_START_0, BLK_END_0, REV_IN	L3R[7:0]
8	REV_SYNC_IN, BLK_START, BLK_END, C[4:0]	REV_SYNC_OUT
9	RESET, ECLKM, DCLKM	
10	ENC_INIT, ENC_TAIL, X	Y[3:0]

Example 1

In this section we give an example of how the MAP04B can be used as a continuous rate 1/2 QPSK encoder and decoder. Note that MAP04B does not perform any synchronisation. This needs to be performed external to the chip.

A simple synchronisation circuit could monitor the output magnitude of the decoder output. The average magnitude will be higher in the synchronised state compared to the unsynchronised states.

Figure 14 shows how the MAP04B1H (1H indicates the mode) can be configured for continuous rate 1/2 QPSK operation. The code used is $g_0 = 23_8$ and $g_1 = 33_8$ (which is not rotationally invariant). Note that unconnected inputs are pulled down to ground. Since the code is not invariant to 180° phase rotations, differential encoding and decoding are not used.

The demodulator output is assumed to be in two's complement form. The L0R output is used for the Synch State Monitor due to the smaller decoder delay. An MDD of 64 is used for best performance. The information and parity bits are assumed to be equally likely implying Z0I = 0. With a 50 MHz asynchronous CLK the maximum decoder speed is 2.2 Mbit/s.

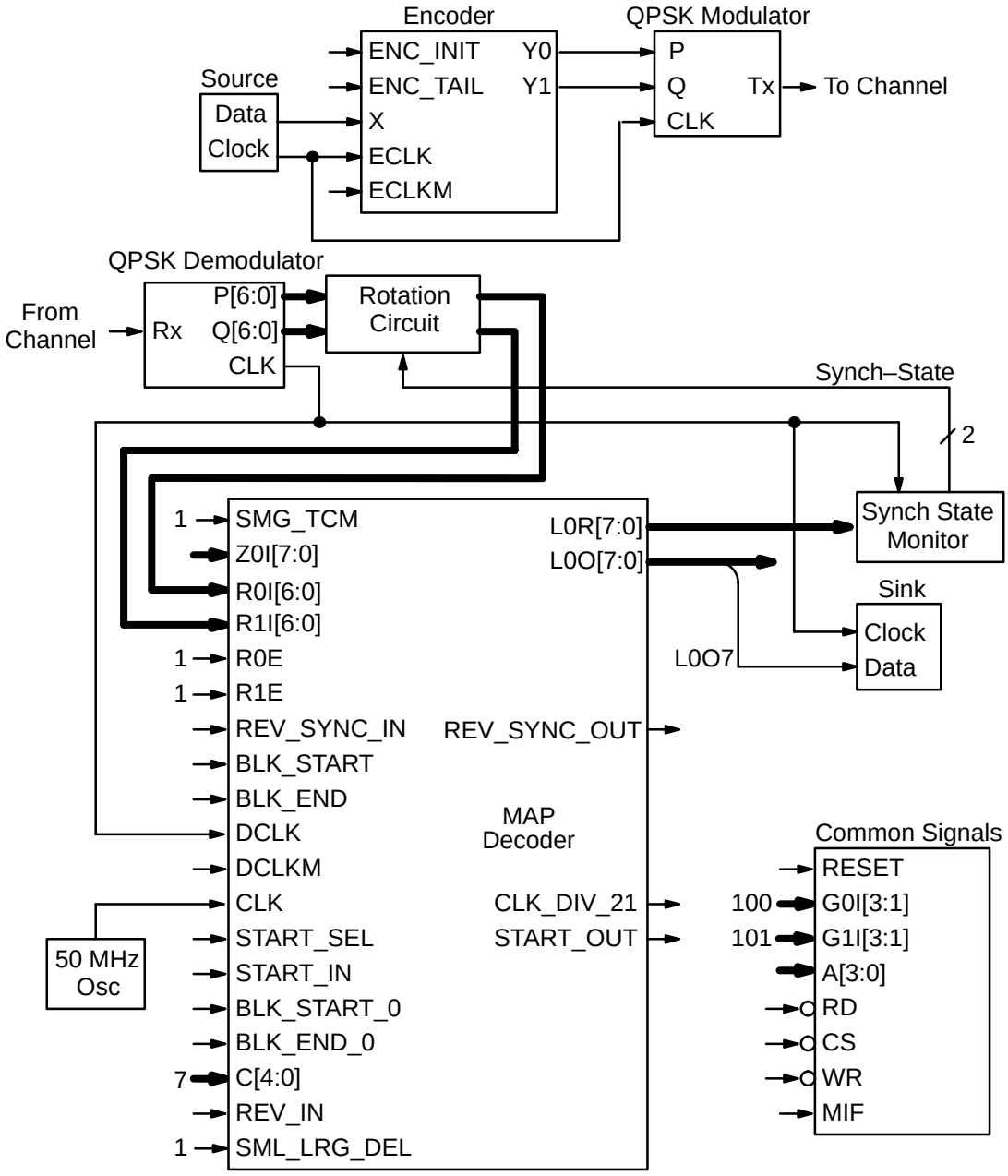


Figure 14: Block diagram of rate 1/2 QPSK codec.

Example 2

In this section we give an example of how the MAP04B1H can interface with a microprocessor. Figure 15 shows how this is achieved. The eight bit data bus is labelled DATA[7:0].

Test Circuit

The following circuit, using the MAP04B1H as an example, can be used to test the decoder for proper operation. The contents for the look–up tables (which contain the test vectors) and the expected decoder outputs are provided with our products. A discrete logic OR gate should be used to avoid double clocking when START goes high. TCLK is the test clock and is equal in frequency to DCLK. Follow these steps for the test:

1. START = low. This clears the counter and forces DCLK high for the start of the test.
2. Download BIT file into Xilinx FPGA. This initialises the flip–flops (FF) and memory in the decoder. This step has to be performed at the beginning of every test.
3. START = high. The counter starts and DCLK is enabled. The logic analyser should also start recording when START goes high. The first 256 decoded outputs can then be compared with the expected output.

BIT and MCS Files

The BIT file is what is downloaded into the FPGA on startup. The Xilinx data book explains how this can be achieved. The MCS files can be used to program serial PROMs which can then be used to program the FPGA on powerup.

For the BIT files, the encoder and decoder are reset when RESET (not shown in schematic symbols) goes high. All the inputs for the encoder and decoder have pulldown resistors.

Configuration

Each Xilinx chip can be preconfigured for certain options. This section gives the options that

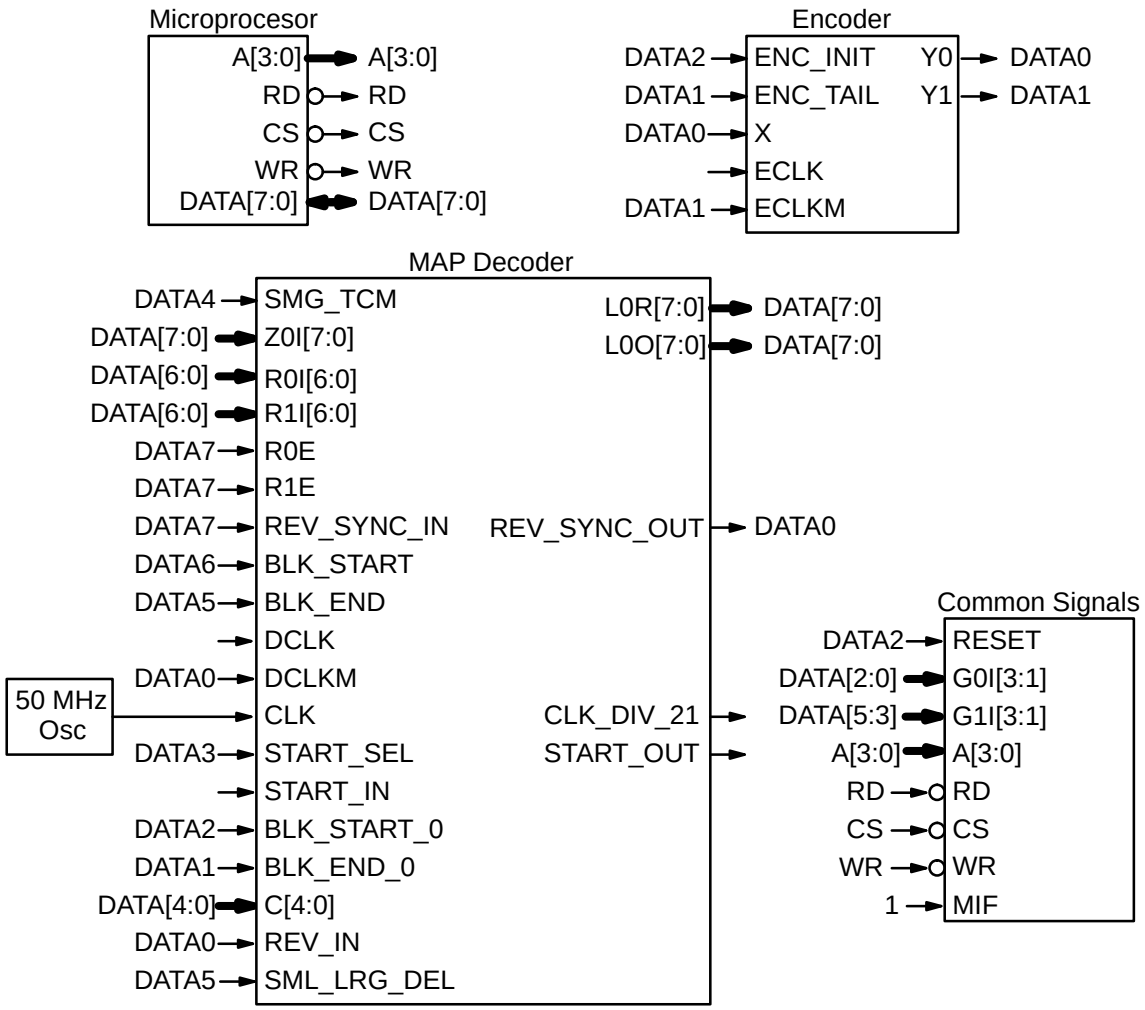


Figure 15: Block diagram of microprocessor interfacing to MAP04B.

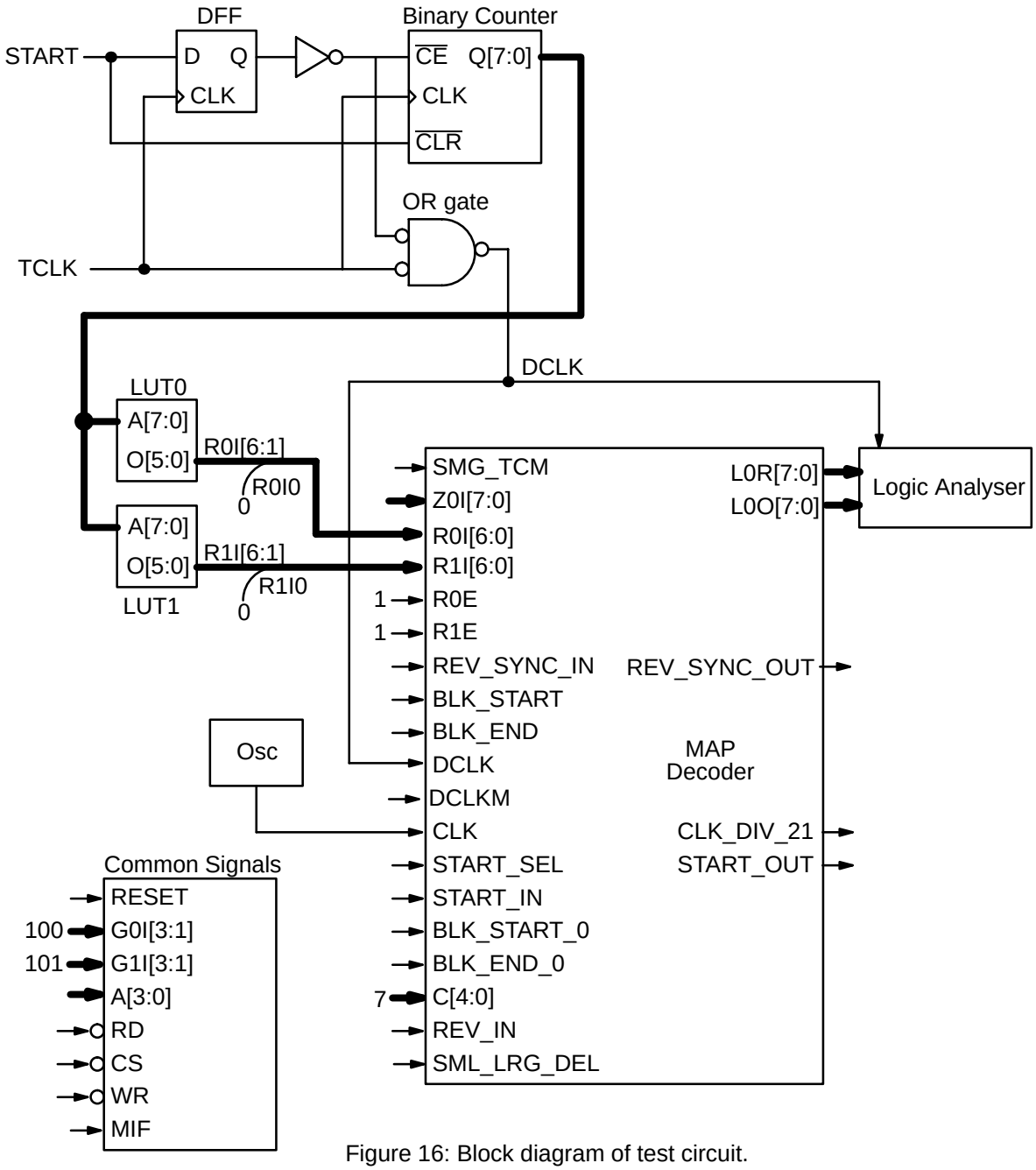


Figure 16: Block diagram of test circuit.

were selected. Other options are also available. Please consult the Xilinx data book for configuration options.

Configuration

- Configuration Rate: Slow
- Threshold Levels (XC4000E and XC4000EX only) –
  Inputs: TTL
  Outputs: CMOS
- Configuration Pins –
  TDO: Float
  M0: Float
  M1: Float
  M2: Float
  DONE: Pull–up
- Perform CRC during configuration: Yes
- Produce ASCII configuration file: No
- 5 V Tolerant I/Os (XC4000XLA and XC4000XV only): Yes

Startup

- Startup clock: CCLK
- Synchronise start–up to DONE input pin: No
- Output Events –
  Done: C1
  Enable Outputs: C3
  Release Set/Reset: C4

Readback

- Clock: CCLK
- Enable bitstream verification and in–circuit hardware debugging: Yes
- Abort readback when TRIG goes inactive: No

Tie

- Tie unused interconnect: No

Advanced

- XC4000EX, XV, XL options – Configuration address lines: 18

Decoder Performance

Figure 17 plots BER versus E_b/N_0 of the MAP04B in continuous decoding operation at code rate 1/2 with code polynomials $g_0 = 37_8$ and $g_1 = 21_8$. A BPSK amplitude A of 16 was used. The value of c used at each operating point was the optimum value quantised to the nearest integer. The "software MAP" curve is a computer simulation of the MAP decoding algorithm with no quantisation. As can be seen, the MAP04B achieves near optimum performance with an implementation loss in the hundredths of a dB.

Notice that the sub–MAP decoder has almost identical performance at low BER. However, at high BER, where iterative decoders initially operate, the MAP algorithm performs significantly better than the sub–MAP algorithm.

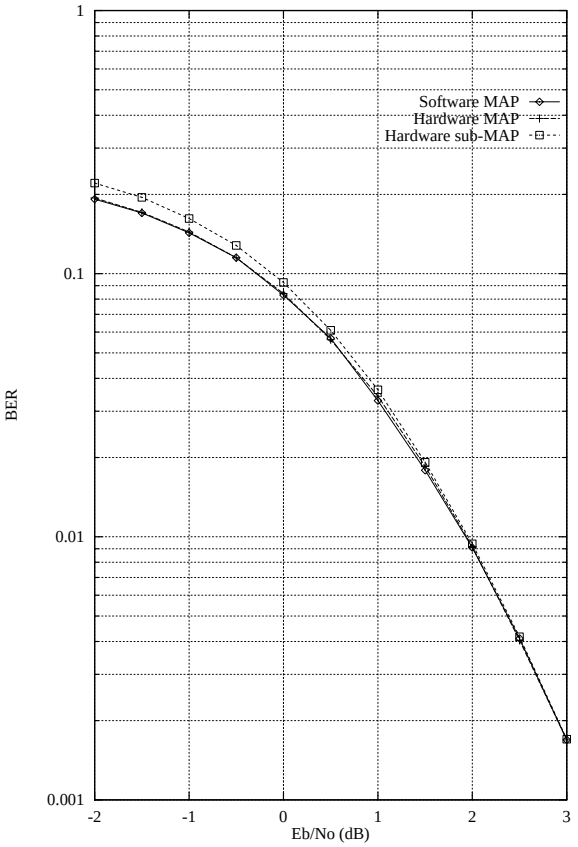


Figure 17: Rate 1/2 performance.

Switching Characteristics

The following switching characteristics reflect worse–case values over the recommended operating conditions. The values are expressed in units of nanoseconds. All values are preliminary. See Decoder Operation section for decoder delay after DCLK and other decoder setup before DCLK requirements. See Clock Operation section for DCLK period and DCLK low or high period.

MAP04B1H–XC4028EX–2PG299C		**–4**		**–3**		**–2**	
Description	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
Encoder setup before ECLK	T_{esu}	60.2		49.2		36.7	
Encoder hold after ECLK	T_{ehd}	0		0		0	
Encoder delay after ECLK	T_{epd}		72.3		60.1		45.4
Encoder delay	T_{edy}		100		82.8		60.4
ECLK period	T_{ecp}	27.6		23.0		20.5	
Decoder setup before DCLK	T_{dsu}	105		86.6		73.5	
Decoder hold after DCLK	T_{dhd}	0		0		0	
CLK period	T_{cp}	60.5		50.2		31.2	
CLK low, ECLK low or high	T_{clh}	3.5		3.0		3.0	
CLK high	T_{wps}	5.5		4.5		4.5	
MIF read delay	T_{rd}		58.2		48.3		27.4
MIF write setup time	T_{ws}	55.7		46.0		24.9	
MIF write period	T_{cl}	3.5		3.0		3.0	
MIF write hold time	T_{wh}	0		0		0	
MIF write data setup time	T_{psd}	8.0		6.8		6.8	

MAP04B4H–XC4036XLA–07HQ160C		**–09**		**–08**		**–07**	
Description	**Symbol**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
Encoder setup before ECLK	T_{esu}	24.8		22.3		19.6	
Encoder hold after ECLK	T_{ehd}	0		0		0	
Encoder delay after ECLK	T_{epd}		34.5		30.8		27.3
Encoder delay	T_{edy}		43.3		38.8		34.3
ECLK period	T_{ecp}	15.6		13.9		12.3	
Decoder setup before DCLK	T_{dsu}	51.7		46.3		41.1	
Decoder hold after DCLK	T_{dhd}	0		0		0	
CLK period	T_{cp}	23.6		21.0		18.4	
CLK low, ECLK low or high	T_{clh}	2.2		1.9		1.7	
CLK high	T_{wps}	3.4		3.0		2.7	
MIF read delay	T_{rd}		26.4		23.6		20.7
MIF write setup time	T_{ws}	21.9		19.5		17.4	
MIF write period	T_{cl}	2.2		1.9		1.7	
MIF write hold time	T_{wh}	0		0		0	
MIF write data setup time	T_{psd}	4.9		4.5		4.0	

Pinouts

Pad Name	HQ160	HQ240	PG299
A0	P156	P236	F5
A1	P152	P230	E2
A2	P12	P16	D8
A3	P157	P237	E4
BLK_END	P96	P146	X12
BLK_END_0	P28	P44	D13
BLK_START	P22	P32	B11
BLK_START_0	P139	P209	L3
CLK	P2	P2	D4
CLK_DIV_21	P155	P233	C1
C0	P73	P113	U19
C1	P75	P115	R16
C2	P69	P105	R19
C3	P86	P126	U16
C4	P67	P103	P19
CS	P153	P231	F3
DCLKM	P36	P56	D16
DCLK	P43	P63	B19
ECLKM	P33	P53	B17
ECLK	P37	P57	C17
ENC_INIT	P47	P67	D18
ENC_TAIL	P46	P66	E17
G0I1	P116	P176	U5
G0I2	P126	P186	T4
G0I3	P115	P175	T6
G1I1	P65	P95	L17
G1I2	P56	P86	J20
G1I3	P45	P65	F16
G2I1	P62	P92	L19
G2I2	P55	P79	H18
G2I3	P64	P94	L16
L0O0	P109	P163	W7
L0O1	P97	P147	U11
L0O2	P108	P162	V8
L0O3	P104	P154	T10
L0O4	P107	P160	X7
L0O5	P113	P173	U6
L0O6	P106	P159	W8
L0O7	P125	P185	R5
L0R0	P127	P187	U3
L0R1	P135	P203	M3
L0R2	P129	P191	U1
L0R3	P133	P198	P3
L0R4	P138	P208	L4
L0R5	P128	P188	V1
L0R6	P134	P202	N1
L0R7	P132	P197	N4
L1O0	P71	P111	P16
L1O1	P49	P69	D19
L1O2	P44	P64	C19
L1O3	P50	P70	E18
L1O4	P25	P41	A14
L1O5	P31	P51	E14
L1O6	P27	P43	B14
L1O7	P30	P50	C15
L1R0	P23	P33	C11
L1R1	P90	P132	U14
L1R2	P89	P131	X17
L1R3	P72	P112	V20
L1R4	P136	P206	L5
L1R5	P112	P172	X3
L1R6	P130	P192	P4
L1R7	P111	P171	T7
MIF	P24	P34	E11
R0E	P124	P184	V2
R0I0	P14	P18	B7
R0I1	P95	P142	X14
R0I2	P98	P148	V11
R0I3	P92	P138	W14
R0I4	P114	P174	V5
R0I5	P105	P155	U10
R0I6	P99	P149	W11
R1E	P147	P220	H1
R1I0	P11	P15	B6
R1I1	P137	P207	M1
R1I2	P146	P216	K4
R1I3	P149	P225	H4
R1I4	P144	P214	K3
R1I5	P143	P213	K2
R1I6	P17	P27	D10
R2E	P57	P87	K17
R2I0			
R2I1	P34	P54	B18
R2I2	P35	P55	E15
R2I3	P63	P93	L18
R2I4	P58	P88	K18
R2I5	P32	P52	C16
R2I6	P148	P221	J3
RD	P8	P8	E7
RESET	P78	P118	X20
REV_IN	P150	P226	F2
REV_SYNC_IN	P145	P215	K5
REV_SYNC_OUT	P140	P210	L2
RZ	P18	P28	C10
SMG_TCM	P26	P42	C13
SML_LRG_DEL	P15	P25	E10
START_IN	P68	P104	N17
START_OUT	P154	P232	G5
START_SEL	P66	P102	N18
WR	P159	P239	C3
X	P54	P78	G19
Y0	P53	P77	G18
Y1	P48	P68	C20
Y2	P52	P76	H17
Z0I0	P83	P123	W19

Pad Name	HQ160	HQ240	PG299
Z0I1	P76	P116	T17
Z0I2	P93	P139	V13
Z0I3	P74	P114	V19
Z0I4	P87	P129	W17
Z0I5	P103	P153	V10
Z0I6	P88	P130	V16
Z0I7	P94	P141	T12

Please consult the Xilinx data book for power and configuration pinouts.

Packages

UCF No.	Mode	Xilinx Part No.
MAP04B01	1H	XC4028EX–2PG299C
MAP04B04	4H	XC4036XLA–07HQ160C

Other Xilinx parts and modes are also available. All pinouts are upward compatible. See the Xilinx data book for pinouts of other packages. Other pinouts can be ordered from *Small World Communications*.

Ordering Information

SW–MAP04B–EDN for EDIF core
SW–MAP04B–SCH for Foundation schematics
SW–MAP04Bmd–p for BIT file
m = mode (0 to 9)
d = L (MDD = 32) or H (MDD = 32 or 64)
p = Xilinx part no. (e.g., XC4036XLA–07HQ160C)

Please indicate how many instantiations you wish to license. An instantiation is considered to be an integrated circuit that uses or is derived from our software in the device's programming or manufacture. Also, *Small World Communications* only provides software and does not provide the actual devices themselves. Note that license costs per instantiation decrease with increasing number of instantiations. Please contact *Small World Communications* for a quote on the number of instantiations and type of license you require.

References

[1] Small World Communications, "Sworld Foundation symbols and sample EDIF files," Aug. 1999.
www.sworld.com.au/software/sworld.zip

[2] Small World Communications, "MAP decoder constant program," Jan. 1999.
www.sworld.com.au/software/cmap.zip

[3] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT–20, pp. 284–287, Mar. 1974.

[4] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub–optimal MAP decoding algorithms operating in the log domain," *ICC'95*, Seattle, WA, USA, pp. 1009–1013, June 1995.

[5] S. S. Pietrobon, "Implementation and performance of a turbo/MAP decoder," *Int. J. Satellite Commun.*, vol. 16, pp. 23–46, Jan.–Feb. 1998.

Small World Communications, 6 First Avenue, Payneham South SA 5070, Australia.
products@sworld.com.au ph. +61 8 8332 0319
http://www.sworld.com.au fax +61 8 8332 3177