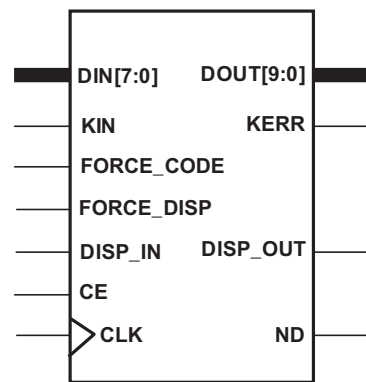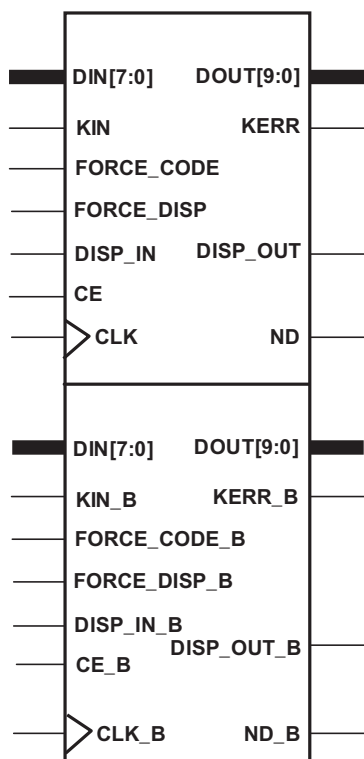# 8b/10b Encoder v5.0

Features

- Drop-in module for Virtex™, Virtex-E, Virtex-II,™ Virtex-II Pro™, Virtex-4™, Spartan™-II, Spartan-IIE, and Spartan-3 FPGAs

- Encoding of 8-bit bytes into 10-bit symbols

- Encoding of 12 special (*K*) characters in addition to the 256 possible byte combinations

- Encoder tracks *running disparity*, ensuring that consecutive symbols have correct disparity

- Choice of a LUT-based implementation, which uses FPGA slices, or a block memory-based implementation, that uses one of the dedicated on-chip block memory blocks

- Optional KERR output indicates whether an invalid special character has been requested

- Optional FORCE_CODE input generates a preselected code symbol regardless of state of DIN and KIN inputs (Virtex-II family only)

- Optional FORCE_DISP input allows Encoder's running disparity to be overridden with an external input

- Optional DISP_OUT output tracks the current running disparity of the Encoder

- Optional CE input to enable or stall the Encoder

- Optional ND (new data) output indicates DOUT has a new encoded symbol on its output

- Block RAM implementation can take secondary (*B*) Encoder with almost no additional resource

- Incorporates Xilinx Smart-IP™ technology for utmost parameterization and optimum implementation

- For use with v6.2i and later of the Xilinx CORE Generator™ system

XIP2035

Figure 1: **Core Schematic Symbol**

XIP2036

Figure 2: **Dual Encoder Schematic Symbol**

Functional Description

The 8b/10b Encoder core implements the full code set proposed by A.X. Widmer and P.A. Franaszek[1]. The code specifies the encoding of an 8-bit byte (256 unique data words) and an additional 12 special (or *K*) characters into a 10-bit symbol, hence the 8b/10b designation.

The characteristics of the code scheme make it ideally suited for high-speed local area networks, computer links, or any serial data link. The code scheme is DC-balanced, which is of particular benefit for active gain, threshold setting and equalization of optical receivers. The code insures a limited run length, no more than 5 consecutive ones or zeros, and a guaranteed transition density, which permits clock recovery from the data stream. The special (K) characters are useful as packet delimiters. A subset of them, referred to as commas, are unique in that their bit pattern never occurs in a string of serialized data symbols and hence can be used to determine symbol boundaries at the receiving end. Additional rules embedded in the code design allow many errors to be detected at the receiving end. The combination of these features allows the receiving end of an encoded 8b/10b data stream to extract the bit rate clock, to determine symbol (and packet) boundaries, and to detect most transmission errors. This is all done with a comparatively low overhead of 25 percent (each 10-bit symbol contains 8 bits of information) versus, for example, a Manchester code with its 100 percent overhead. Because of its many features, the code has been used in the physical layer (PHY) of a number of current and emerging standards, including Fibre Channel, Gigabit Ethernet, and Rapid I/O, to name a few.

Disparity

A number of terms in this datasheet are taken from the original *IBM Journal* paper, and a thorough understanding of the terminology is critical to a successful application of the core. Most important is the concept of *disparity*.

The disparity of any block of data is defined as the difference between the number of ones and zeros in the block.

Positive and negative refer to an excess of 1s over 0s, or 0s over 1s, respectively. Each encoded symbol can be considered to be a block. The code scheme guarantees that an encoded symbol's disparity is always either 0 (five ones, five zeros), +2 (six ones four zeros) or -2 (four ones, six zeros). Some byte inputs will have more that one potential symbol encoding with the encoded symbol pattern determined by the "running disparity." Running disparity is simply a record of disparity for the aggregate of all the previously encoded symbols. For packet-based networking applications, the running disparity is typically tracked from the start of a packet. The code scheme stipulates that the running disparity at the end of any symbol (block) is always +1 or -1. To insure that this rule is maintained, the Encoder will track the current running disparity. If the currently encoded byte produces a symbol of zero disparity, the running disparity remains unchanged. When the input byte produces a nonzero disparity symbol, the Encoder will encode the data such that the running disparity is swapped, for example, [+1 + (-2) = -1] or [-1 +(+2) = +1]. See Table 1 for an example of the two possible encoded symbol patterns for the D31.1 data symbol.

The code scheme actually partitions the input byte into 5-bit and 3-bit sub-blocks, which in turn are encoded into 6- and 4-bit blocks respectively. The original nomenclature defines the symbols in terms of these sub-blocks. The five input bits are defined as A, B, C, D and E (A is LSB) and the 3-bit block is F, G and H (F is LSB). A prefix of D or K is used to distinguish between data and special characters respectively. For example, D31.1 is a data symbol with all ones on the 5-bit block (11111) and a single one as the LSB of the 3-bit block (100) (see Table 1). Note that the 5-bit sub-block precedes the 3-bit sub-block and the ordering (LSB to MSB) is ABCDE_FGH. The encoded sub-blocks are described with lowercase letters a, b, c, d, e, i and f, g, h, j respectively. The digital weighting of the individual bits of a data byte to be encoded is somewhat arbitrary. However, the limited number of special characters (K symbols) makes the bit ordering for special character generation critical. See Table 2 for the appropriate state of the DIN input when generating special characters.

Implementation Guidelines

A number of details must be adhered to when using the encoder in a serial link—if these guidelines are not followed, the implementation of the link will not match the original function of the code scheme.

- When serializing encoded symbols, DOUT[0] should be transmitted first, DOUT[9] transmitted last. This will produce a serial stream of abcdei_fghj as stipulated in the original paper.

- The Encoder uses the same convention as the Fibre Channel specification in which D[0] is the least significant bit of the input byte. This doesn't have any implications for data words, but care must be taken when generating special characters. See Table 2 when determining correct input combinations for special character generation.

- If more than one encoder is used to encode larger words into multiple symbols, their disparity inputs and outputs must be chained together. The RUN_DISP output of the first Encoder drives the DISP_IN input of the second Encoder with the final Encoder's RUN_DISP output connected back to the DISP_IN of the first Encoder.

Table 1: **Example Encoding of D31.1 for Both Running Disparity (RD) Cases**

	DIN[7:0]								RD (prior)	DOUT[9:0]										RD (after)
	7	6	5	4	3	2	1	0		9	8	7	6	5	4	3	2	1	0	
	H	G	F	E	D	C	B	A		j	h	g	f	i	e	d	c	b	a	
D31.1	0	0	1	1	1	1	1	1	+1	1	0	0	1	0	0	1	0	1	0	-1
D31.1	0	0	1	1	1	1	1	1	-1	1	0	0	1	1	1	0	1	0	1	+1

Table 2: **DIN, KOUT, and DOUT for Valid Special Character Decoding**

	KIN	DIN[7:0]								DIN Unsigned (7 as MSB)
	K	H	G	F	E	D	C	B	A	
K28.0	1	0	0	0	1	1	1	0	0	28
K28.1	1	0	0	1	1	1	1	0	0	60
K28.2	1	0	1	0	1	1	1	0	0	92
K28.3	1	0	1	1	1	1	1	0	0	124
K28.4	1	1	0	0	1	1	1	0	0	156
K28.5	1	1	0	1	1	1	1	0	0	188
K28.6	1	1	1	0	1	1	1	0	0	220
K28.7	1	1	1	1	1	1	1	0	0	252
K23.7	1	1	1	1	1	0	1	1	1	247
K27.7	1	1	1	1	1	1	0	1	1	251
K29.7	1	1	1	1	1	1	1	0	1	253
K30.7	1	1	1	1	1	1	1	1	0	254

Figure 3 shows one possible implementation of a dual encoder, capable of encoding 16 bits into 20 bits. Be aware of the synchronous nature of both encoders and their DISP_IN and DISP_OUT ports. In order for each encoder's input disparity (DISP_IN) to be dependent on the output disparity (DISP_OUT) of the other, the two encoders would have to be clocked alternately to each other by either inverting the input clock to one encoder or by toggling CE. A faster implementation could be derived with more sophisticated logic.
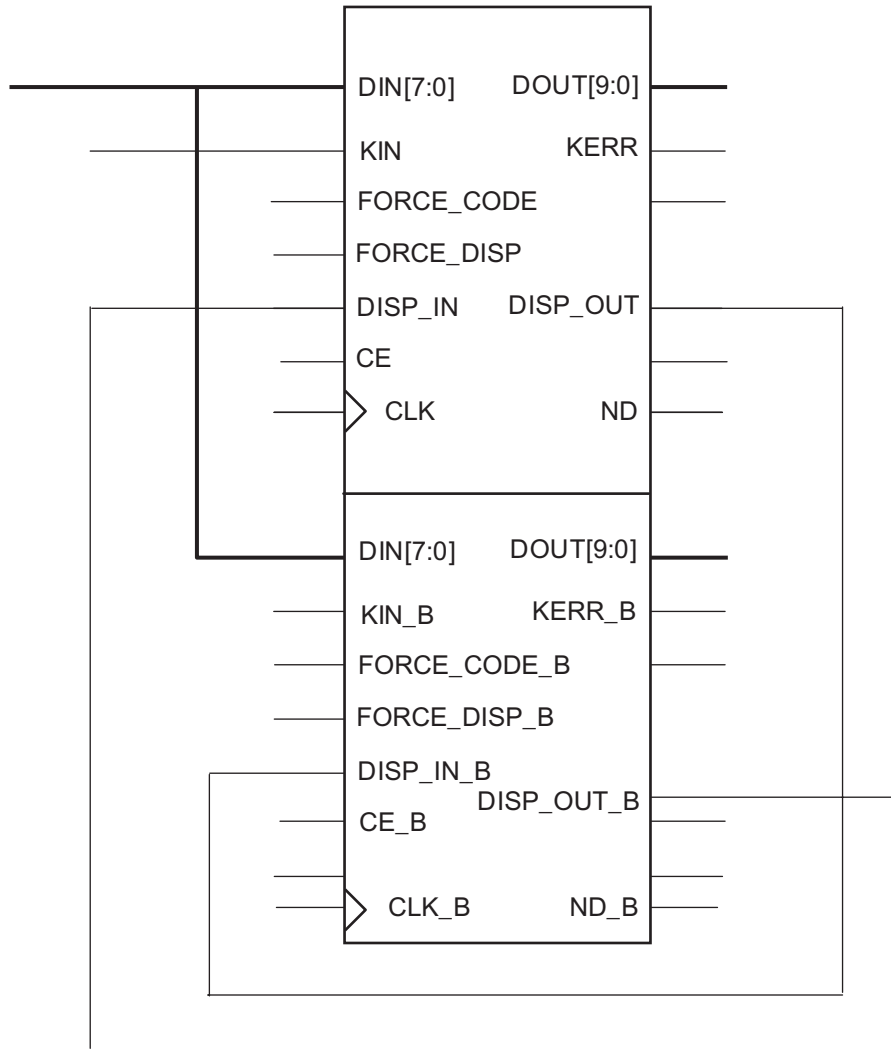
XIP2037

Figure 3: **Example of Chaining Two Encoders**

Pinout

Signal names are shown in Figures 1 and 2 and described in Table 3 and in the text that follows the table.

Table 3: **Encoder Input and Output Ports**

I/O Pin Name	Direction	Description
CLK [CLK_B]	Input	**Clock:** clock input, all Encoder inputs are sampled and all outputs are synchronous to the rising edge of the CLK input. CLK_B is the clock input for the optional secondary B Encoder.
CE [CE_B] (optional)	Input	**Clock Enable:** gates the clock input if the optional CE input port is present. The CE input must be Active (High) or the Encoder will not change state in response to a CLK input. CE_B is the clock enable input for the optional secondary B Encoder.
DIN<7:0> DIN_B<7:0>	Input	**Data Input:** DIN is the input byte to be encoded. DIN_B is the input byte to the B Encoder.
KIN [KIN_B]	Input	**Command Input:** The KIN input controls whether DIN is encoded as data (KIN=0) or, if KIN is Active (logic 1), DIN will be encoded as a special character (if defined). See Table 2 (KIN=1).
FORCE_DISP [FORCE_DISP_B] (optional)	Input	**Force Disparity:** FORCE_DISP, when Active (logic 1), overrides the current running disparity with the external DISP_IN input. FORCE_DISP_B controls the disparity function for the B Encoder.
DISP_IN [DISP_IN_B] (optional)	Input	**Disparity Input:** DISP_IN sets the running disparity for the current input to be encoded if FORCE_DISP is Active (logic 1) for a positive running disparity, logic 0 for a negative running disparity. If FORCE_DISP is inactive, DISP_IN has no effect and the Encoder's internal running disparity will be used to encode the input data. DISP_IN_B has the same function for the B Encoder.
FORCE_CODE [FORCE_CODE_B] (optional)	Input	**Force Code:** FORCE_CODE will drive the Encoder into a predefined initialization state. Except for the CLK input, all other inputs are don't care when FORCE_CODE is Active. FORCE_CODE_B has the same function for the B Encoder.
DOUT<9:0> DOUT_B<9:0>	Output	**Data Output:** DOUT is the encoded 10-bit symbol. DOUT_B is the encoded 10-bit symbol for the B Encoder.
DISP_OUT [DISP_OUT_B] (optional)	Output	**Disparity Output:** DISP_OUT tracks the Encoder's running disparity. If DISP_OUT is a logic 1, the next symbol will be encoded with a positive starting running disparity; if it is a logic 0 the next symbol will be encoded with a negative starting running disparity. DISP_OUT_B performs the same function for the B Encoder. See detailed description of DISP_OUT below for more details.
KERR KERR_B (optional)	Output	**Command Error:** KERR intended for debug, this output will become Active (logic 1) if KIN was Active and DIN did not map to a defined special character (see Table 2). KERR_B performs the same function for the B Encoder.
ND ND_B (optional)	Output	**New Data:** ND can be used to indicated to downstream devices that the DOUT port has a new data symbol ready for transmission. (ND is simply a register version of the CE input.) ND_B performs the same function for the B Encoder.

Clock - CLK [CLK_B]

The Encoder is fully synchronous to its appropriate clock port (CLK_B for the B Encoder). All Encoder input ports have their setup time referenced to the rising edge of the CLK (or CLK_B) input. All Encoder outputs are also synchronous to their respective CLK input. Clock inputs are rising edge Active by default, to make the Encoder(s) respond to the falling edge of a system clock insert an inverter between the system clock and the Encoder's CLK input.

Clock Enable - CE [CE_B]

CE, optional input, can be used to gate the clock input to the Encoder. If the Encoder has a CE port, the clock input will be gated by the CE input and transitions on the clock port will have no effect unless the associated CE input is Active (logic 1). The CE_B pin gates the CLK_B input to the B Encoder in the same manner.

Data-Input Bus - DIN<7:0> [DIN_B<7:0>]

DIN<7:0> the DIN bus provides the data bytes to be encoded into code symbols. When KIN is inactive, a data symbol will be produced. If KIN is active, a special character will be encoded, provided the DIN input maps to a defined special character (see Table 2). Each pin of the bus should be driven to a valid logic level, *e.g.,* if data words of less than 8 bits are to be encoded, the unused DIN bits must still be driven to a valid logic level. DIN_B has the same functionality for the B Encoder.

Command-Input - KIN [KIN_B]

KIN, command input, is used to differentiate between the encoding of data bytes and special characters. If KIN is inactive (logic 0), the DIN bus inputs will be encoded into a data symbol representing one of the 256 possible permutations of the data byte (DIN). When KIN is Active (logic 1), a special character, determined by the DIN input, will be encoded. Note that only 12 special characters are defined and hence many combinations of KIN=1 and DIN are not valid. While the behavior of the Encoder with these invalid inputs will be deterministic, the only thing guaranteed is that the KERR output will become active when an undefined special character is requested. KIN_B has the same functionality for the B Encoder.

Data-Output Bus - DOUT<9:0> [DOUT_B<9:0>]

DOUT is the encoded output symbol. Subsequent to every rising clock edge (and CE=1, if present) DOUT will hold the encoded data symbol of the DIN and KIN inputs (prior to the clock edge). As mentioned earlier, the encoded symbol is also dependent on the running disparity as maintained by the Encoder or directed by the DISP_IN port (FORCE_DISP active). DOUT_B is the encoded symbol output for the B Encoder.

Force Disparity - FORCE_DISP [FORCE_DISP_B]

FORCE_DISP, optional input. If present, the FORCE_DISP port can be used to override the Encoder's internal running disparity. When Active (logic 1), DIN/KIN input will be encoded based on a running disparity set by the DISP_IN port. This input can be used to force data packets to start with a given disparity or, alternatively, can be tied to logic 1 when chaining Encoders together. FORCE_DISP_B provides the same functionality for the B Encoder.

Disparity Input - DISP_IN [DISP_IN_B]

DISP_IN, optional input, is required when FORCE_DISP is present. The DISP_IN can be used to control the running disparity against which the current input data byte (or special character) is encoded. In the event the current inputs should be encoded with a positive running disparity, DISP_IN should be driven High (logic 1). For a negative running disparity, drive DISP_IN Low (logic 0). The DISP_IN input will have no effect on the symbol encoding when FORCE_DISP is inactive (logic 0). FORCE_DISP_B provides the same functionality for the B Encoder.

Force Code - FORCE_CODE [FORCE_CODE_B]

FORCE_CODE, optional input, when present, can be used to force the Encoder to output a preselected data symbol. If Active (logic 1), the Encoder will output an encoded symbol representing the data symbol and running disparity selected using the GUI. Users should note that if active for more than one clock period, the consecutive symbols will be generated with the same running disparity and hence have the potential (if the disparity of the generated symbol is not zero) to violate the coding schemes disparity rules. FORCE_CODE_B provides the same functionality for the B Encoder. FORCE_CODE has limited functionality for the Virtex device family.

Disparity Output - DISP_OUT [DISP_OUT_B]

DISP_OUT is the internal running disparity of the 8b10b Encoder Core. This output allows the user to see the internal state of the running disparity within the core. If the DISP_OUT output is logic 0, the new internal running disparity is negative. This means that the next value encoded must be encoded with either a neutral or positive code disparity in order to comply with the disparity rules. Similarly, if DISP_OUT is a logic 1, the new internal running disparity is positive, which will force the next encoded symbol to be encoded with a neutral or negative code disparity. If multiple Encoders are chained together, the DISP_OUT signal should be connected to the DISP_IN pin of the next Encoder (the Encoder that produces the next symbol to be serialized). DISP_OUT_B has the same functionality for the B Encoder.

The rules of disparity are described below:

Current Running Disparity	Code Disparity	New Running Disparity
–	0	–
–	+	+
+	0	+
+	–	–
–	–	error/invalid
+	+	error/invalid

Command Error - KERR [KERR_B]

KERR, optional output, indicates that KIN was active on the immediately prior clock cycle but the inputs on the DIN<7:0.> bus did not correspond to the a valid special character. This port is primarily provided as a debugging aid, as inputs of this type violate the coding scheme and therefore should not occur in normal-use scenarios. KERR_B has the same functionality for the B Encoder.

New Data Output - ND [ND_B]

ND, optional output, indicates that all Encoder outputs have to be updated on the most recent clock, hence indicating to downstream logic that a new encoded symbol is ready for transition. ND requires that the Encoder also have a CE input, and is simply a registered version of CE. ND will not be active on any clock cycle following an active FORCE_CODE input. ND_B has the same functionality and restrictions for the B Encoder.

Table 4: **Default Values and XCO File Values**

Parameter	XCO File Values	Default GUI Setting
component_name	ASCII text starting with a letter and based upon the following character set: a-z, 0-9, and _	blank
implementation	One of the following keywords: LUT-based, Block_Ram_Based	LUT_Based
code_error_output	One of the following keywords: true, false	false
b_code_error_output	One of the following keywords: true, false	false
new_data	One of the following keywords: true, false	false
b_new_data	One of the following keywords: true, false	false
force_code_symbol	One of the following keywords [Virtex]: Reset_neg	Reset_neg
b_force_code_symbol	One of the following keywords [Virtex]: B_Reset_neg	B_Reset_neg
force_code_symbol	One of the following keywords [Virtex-II]: D_10_2_pos, D_10_2_neg, D_21_5_pos, D_21_5_neg, K_28_5_neg	D_10_2_neg
b_force_code_symbol	One of the following keywords [Virtex-II]: B_D_10_2_pos, B_D_10_2_neg, B_D_21_5_pos, B_D_21_5_neg, B_K_28_5_neg	B_D_10_2_neg
clock_enable	One of the following keywords: true, false	false
b_clock_enable	One of the following keywords: true, false	false
disparity_output	One of the following keywords: true, false	false
b_disparity_output	One of the following keywords: true, false	false
secondary_encoder	One of the following keywords: true, false	false
force_code	One of the following keywords: true, false	false
b_force_code	One of the following keywords: true, false	false
force_disparity	One of the following keywords: true, false	false
b_force_disparity	One of the following keywords: true, false	false

CORE Generator Parameters

The XCO file parameters, values, and GUI defaults are shown in Table 4. Names of XCO file parameters and their parameter values are identical to the names and values shown in the GUI, except that underscore characters (_) are used instead of spaces. The text in an XCO file is case insensitive.

Core Resource Utilization & Performance

The BlockRAM-based encoder requires three (4K-bit) Virtex BlockRAMs plus a single additional slice.

If a Virtex-II device is targeted, only one (16K-bit) BlockRAM is required, plus the same additional slice.

For either architecture, a second encoder can be generated using the same BlockRAM resource, configured in a dual-port mode. The only additional logic requirement is a second additional slice, to support the second Encoder.

The core performance is gated entirely by the clock to out and setup times of the targeted device. The worst-case performance for all Virtex families will be 100MHz.

Reference

1. A.X. Widmer and P.A. Franaszek, A DC-BALANCED, PARTITIONED-BLOCK, 8B/10B TRANSMISSION CODE, *IBM Journal of Research and Development*, Volume 27, Number 5, September 1983.

Ordering Information

This core may be downloaded from the Xilinx IP Center for use with the Xilinx CORE Generator System v6.2i and later. The Xilinx CORE Generator System tool is bundled with all Alliance Series Software packages, at no additional charge.

To order Xilinx software, please visit the Xilinx Silicon Xpresso Cafe or contact your local Xilinx sales representative.

Information about additional Xilinx LogiCORE modules is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/28/03	1.0	Revision history added to document.
05/21/04	1.1	Xilinx CORE Generator v6.2i support.