8b/10b Encoder V1.0

November 3, 2000 Product Specification

Xilinx Inc.
2100 Logic Drive
San Jose, CA 95124
Phone: +1 408-559-7778
Fax: +1 408-559-7114
URL: www.xilinx.com/ipcenter
Support: support.xilinx.com

Features

- Drop in module for Virtex™, Virtex™-E, Virtex™-II and Spartan®-II 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" insuring correct disparity of consecutive symbols.
- Optional KERR output indicates whether an invalid special character has been requested.
- Optional FORCE_CODE input generates a pre-selected 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 for selective enable/stall of Encoder.
- Optional ND (new data) output indicates DOUT has a new encoded symbol on its output.
- Block RAM implementation can implement secondary (B) Encoder with low additional resource overhead.

Notice

This byte oriented DC balanced 8b/10b partitioned block transmission code may contain material covered by patents owned by third parties including International Business Machines Corporation. By providing this core as one possible implementation of this standard, Xilinx is making no representation that the provided implementation of this standard is free from any claims of infringement by any third party. Xilinx expressly disclaims any warranty with respect to the adequacy of the implemenation, including but not limited to any warranty or representation that the implemenation is free from claims of any third party. Futhermore, Xilinx is providing this core as a courtesy to you and suggests that you contact all third parties including IBM to obtain the necessary rights to use this implementation.

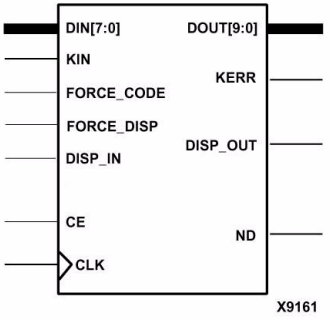

Figure 1: Encoder Schematic Symbol

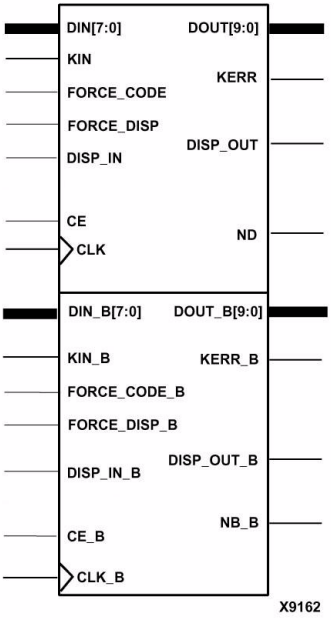

Figure 2: Dual Encoder Schematic Symbol

Functional Description

The ENCODE_8B10B_V1_0 core (Encoder) 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:

This datasheet uses a number of terms present in the original *IBM Journal* paper. Understanding this terminology is critical to a successful application of this 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, *e.g.* [+1 + (-2) = -1] or [-1 +(+2) = -1]. See Table 1 for at example of the two possible encoded symbol patterns for the D31.1 data symbol.

The code scheme actually partitions the input byte into a 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 three-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 lower case 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. Consult Table 2 for the appropriate state of the DIN input when generating special characters.

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	2	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

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

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

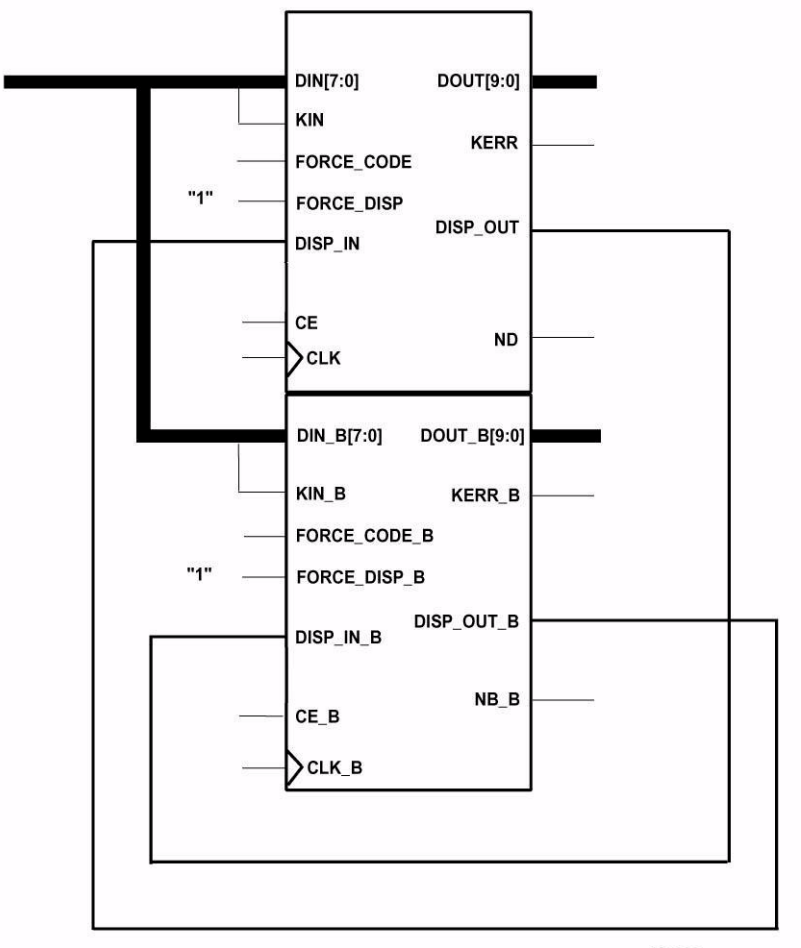

Figure 3: Example of Chaining two Encoders.

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. Refer to 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. See Figure 5 for a schematic of this case.

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 on the rising edge of the CLK input. CLK_B is the clock input for 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 affect and the Encoders 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 pre-defined 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 Encoders running disparity. If DISP_OUT logic 1 the next symbol will be encoded with a positive starting running disparity, if it is a logic 0 the next symbol is encoded with a negative staring running disparity. DISP_OUT_B performs the same function for the "B" Encoder.
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 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.

Pinout

Signal names are shown in Figures 1 and 2 and described in Table 3. The GUI interface

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. If the current inputs should be encoded with a positive running disparity, then 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, optional output, tracks the running disparity of the Encoder. If DISP_OUT is a logical one ("1"), the Encoder's running disparity (including the symbol currently on the DOUT<7:0> bus) is positive (+1). A logic 0 indicates that the current running disparity is negative (-1). 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.

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 since inputs of this type is a violation of 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 and hence indicates 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.

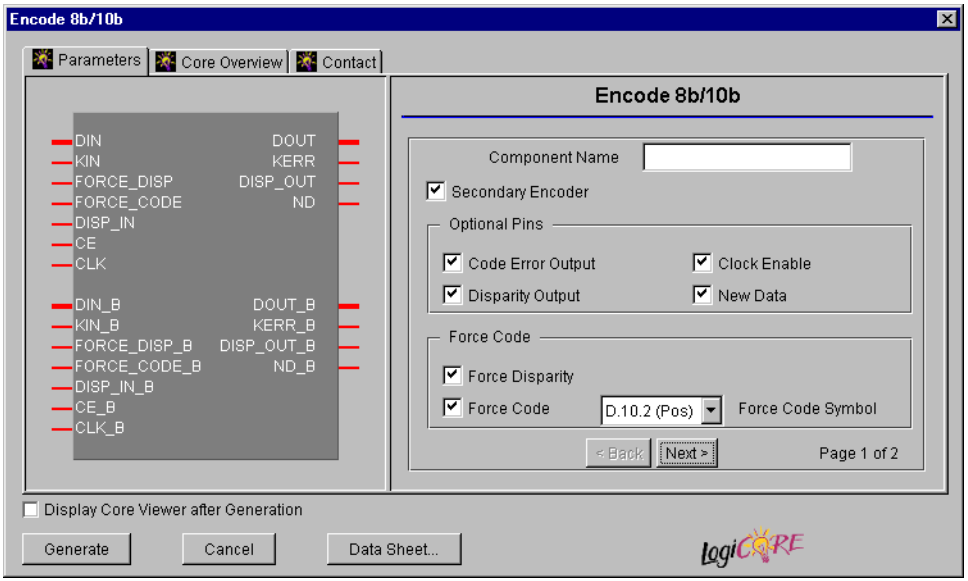

Figure 4: 8b/10b Encoder Main Parameterization Window

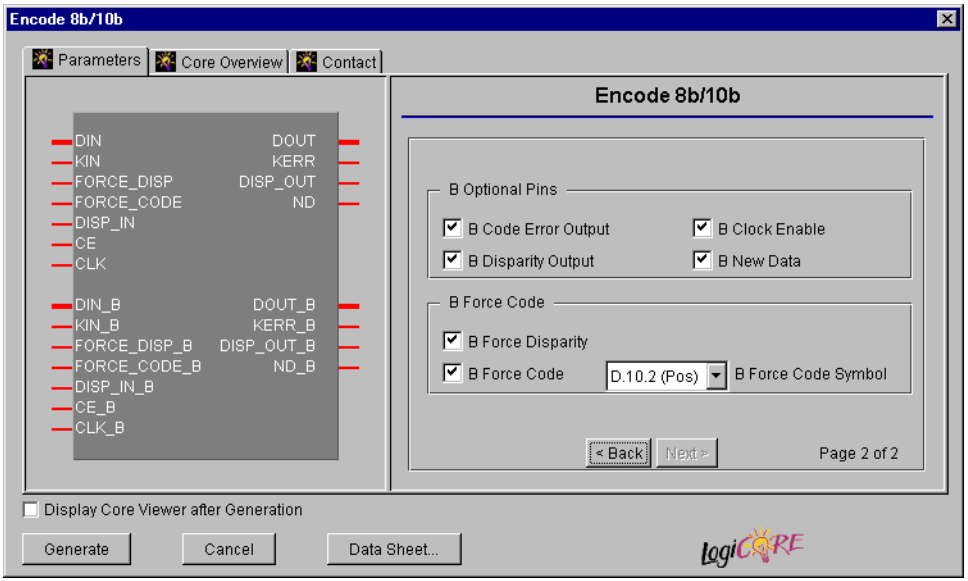

Figure 5: Secondary Encoder Parameterization Window

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
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 main Core Generator parameterization screen for this module is shown in Figure 4 and 5.

The XCO file parameters, values, and GUI defaults are shown below 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.

Ordering Information

This core can be downloaded, free of cost, from the Xilinx IP Center (http://www.xilinx.com/ipcenter) for use with the Xilinx CORE Generator™ System V3.1i and later. The CORE Generator System tool is bundled with all Xilinx Alliance and Foundation Series Software packages.

To order online, visit the Xilinx Silicon Expresso Cafe at http://toolbox.xilinx.com/cgi-bin/xilinx.storefront/241669816/catalog//1006).

Xilinx software can also be ordered through your local Xilinx sales office. Information on the sales office nearest you is available at http://www.xilinx.com/company/sales.htm.