8b/10b Decoder v7.1

DS258 April 28, 2005 **Product Specification**

Features

- Drop-in module for Virtex™, Virtex-E, Virtex-II, Virtex-II Pro, Virtex-4, Spartan™-II, Spartan-IIE, and Spartan-3 FPGAs
- Decoding of 10-bit symbols into 8-bit bytes and an accompanying *K* bit
- Decoding of 268 unique transmitted characters: 256 byte values and 12 special (*K*) characters
- Fully synchronous operation
- Decoder tracks *running disparity* to verify that disparity sequence of the received symbols is valid
- Choice of an 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 RUN_DISP output tracks the running disparity of the decoder
- Optional DISP_ERR output indicates a violation of the running disparity rules
- Optional SINIT input forces the decoder into a user-defined known state
- Optional CE input can be used to selectively enable or stall the decoder
- Optional DISP_IN supports chaining multiple decoders together
- Optional CODE_ERR output indicates that the input symbol did not correspond to a valid member of the code set
- Optional SYM_DISP output provides disparity information on the current symbol being decoded
- Optional ND (new data) output indicates DOUT has a new decoded symbol on its output
- Block RAM implementation can implement secondary (*B*) decoder with almost no additional resource overhead
- To be used with v7.1i and later of the Xilinx CORE Generator™ system

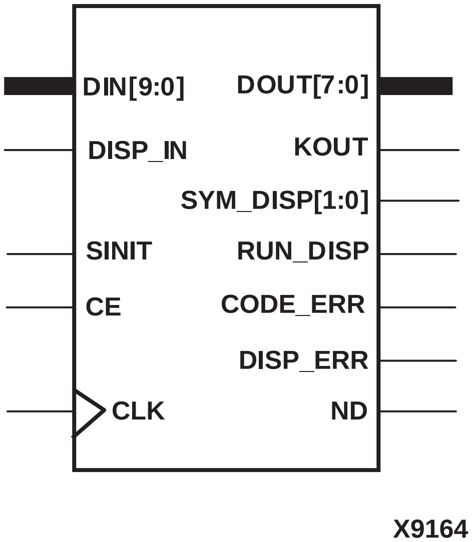

Figure 1: **Decoder Schematic Symbol**

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

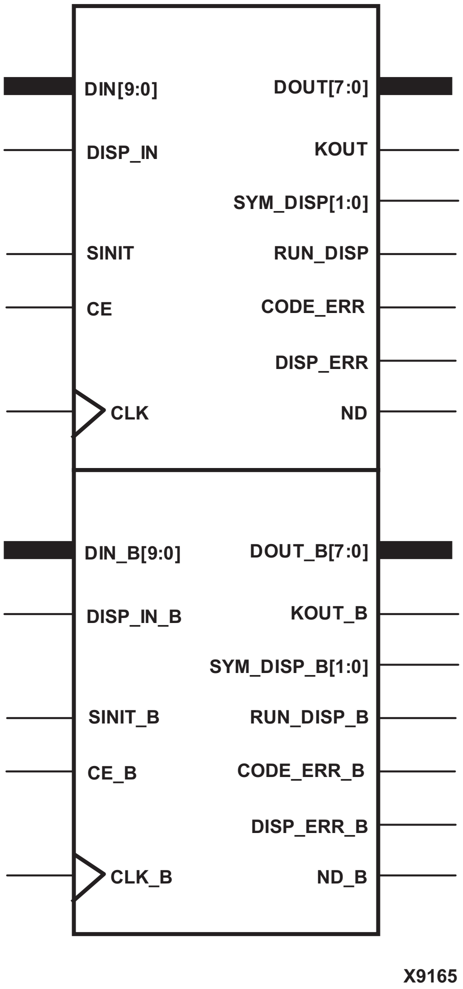

Figure 2: **Dual Decoder Schematic Symbol**

Functional Description

The 8b/10b Decoder 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. A number of 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 has a limited run length; no more than five 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 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 data sheet are taken from the original *IBM Journal* article, 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 values will have more than one potential symbol encoding with the encoded symbol pattern determined by the “Running Disparity.” Running disparity is simply a record of the 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 Decoder tracks the running disparity of the input data stream for error checking purposes. At the end of each 10-bit encoded symbol, the running disparity will be +1 or -1, which equates to a 1 or 0 respectively on the RUN_DISP output. Each symbol's disparity is combined with the current running disparity to produce the new running disparity. Symbols with a disparity of 0 would then not modify the running disparity [+1 + (0) = +1] or [-1 + (0) = -1]. Symbols with a disparity of +2 or -2 would swap the running disparity between +1 and -1; *e.g.*, [+1 + (-2) = -1] or [-1 + (+2) = +1].

If the symbol disparity is anything other than +2, 0, or -2, the disparity of the incoming data is invalid and a disparity error is flagged (DISP_ERR = 1). Such an error also occurs if the running disparity is +1 and the symbol disparity is +2, or when the running disparity is -1 and the symbol disparity is -2. Most invalid input symbols will also result in a disparity violation.

Running disparity validity is also enforced at the sub-block level. Since the 10-bit block is actually composed of a 6-bit block and a 4-bit block, the same rules for running disparity apply at those sub-block boundaries. The disparity of each sub-block must be +2, 0, or -2. The running disparity at the end of the sub-block must follow the rule that running disparity must be -1 or +1. Failure to meet this criteria will also be flagged as a disparity error by the Decoder.

The Decoder uses the scheme in Widmer and Franaszek’s paper to decode the 10-bit input symbol into an 8-bit output value. The paper’s nomenclature defines the bits of the 10-bit symbol as abcdei_fghj, where ‘a’ is the LSB and ‘j’ is the MSB. This 10-bit encoded symbol is partitioned as a 6-bit sub-block (abcdei) and a 4-bit sub-block (fghj). In the same way, the 8-bit output value is defined as the concatenation of a 5-bit and a 3-bit sub-block (ABCDE and FGH respectively). As defined in the original paper, these 8 bits are named ABCDE_FGH where A is the LSB, and H is the MSB.

When decoding the 10-bit input symbol, the decoder first decodes the six least significant bits abcdei of the input symbol into the five least significant bits of the output, ABCDE. Table 1 shows an example of this decoding for the symbol designated D31.1. The five output bits ABCDE equate to 31 in decimal (EDCBA=11111), thus providing the first part of the symbol name. Similarly, the four most significant bits fghj of the input symbol decode into the three most significant bits of the output FGH. The three output bits, FGH, represent a decimal 1, providing the second part of the symbol name (HGF=001).

Table 2 defines the decoding of the 12 special command symbols for both positive and negative disparity cases. These special characters are distinguished from the 256 standard characters by the KOUT output being asserted when the symbol is decoded. The DOUT value identifies which of the twelve special characters have been decoded.

Table 1: **Example Decoding of d31.1 for Both Running Disparity (RD) Cases**

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

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

	DIN[9:0]										KOUT	DOUT[7:0]								DOUT Unsigned (7 as MSB)
	j	h	g	f	i	e	d	c	b	a	K	H	G	F	E	D	C	B	A	
K28.0	0	0	1	0	1	1	1	1	0	0	1	0	0	0	1	1	1	0	0	28
K28.0	1	1	0	1	0	0	0	0	1	1	1	0	0	0	1	1	1	0	0	28
K28.1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	1	1	1	0	0	60
K28.1	1	0	0	1	1	1	1	1	0	0	1	0	0	1	1	1	1	0	0	60
K28.2	1	0	1	0	1	1	1	1	0	0	1	0	1	0	1	1	1	0	0	92
K28.2	0	1	0	1	0	0	0	0	1	1	1	0	1	0	1	1	1	0	0	92
K28.3	1	1	0	0	1	1	1	1	0	0	1	0	1	1	1	1	1	0	0	124
K28.3	0	0	1	1	0	0	0	0	1	1	1	0	1	1	1	1	1	0	0	124
K28.4	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	1	0	0	156
K28.4	1	0	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	156
K28.5	0	1	0	1	1	1	1	1	0	0	1	1	0	1	1	1	1	0	0	188
K28.5	1	0	1	0	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	188
K28.6	0	1	1	0	1	1	1	1	0	0	1	1	1	0	1	1	1	0	0	220
K28.6	1	0	0	1	0	0	0	0	1	1	1	1	1	0	1	1	1	0	0	220
K28.7	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	252
K28.7	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	252
K23.7	0	0	0	1	0	1	0	1	1	1	1	1	1	1	1	0	1	1	1	247
K23.7	1	1	1	0	1	0	1	0	0	0	1	1	1	1	1	0	1	1	1	247
K27.7	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	251
K27.7	1	1	1	0	1	0	0	1	0	0	1	1	1	1	1	1	0	1	1	251
K29.7	0	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1	1	0	1	253

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

K29.7	1	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	1	0	1	253
K30.7	0	0	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	0	254
K30.7	1	1	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	0	254

Implementation Guidelines

The original paper by Widmer and Franaszek defines the 10-bit encoded symbol as a serial stream of bits named abcdei_fghj. This encoded symbol is transmitted serially from a to j. The 10-bit symbol contains sub-blocks of 6-bits and 4-bits. Expressed in this manner, the *a* bit is actually the LSB of the value, and the “j” bit is the MSB.

In the decoder, the 10-bit DIN bus receives the encoded symbol as DIN <9:0>, where DIN <9> is the MSB (the “j” bit) and DIN<0> is the LSB (the *a* bit).

The same rule applies to the DOUT decoded output byte. Represented as ABCDE_FGH in the original paper, with A as the LSB and H as the MSB, the DOUT <7:0> bus of the decoder stores this value with DOUT <7>, the MSB (the “H” bit) and DOUT <0> the LSB (the *a* bit).

The MSB/LSB ordering has no implications for data words, but has implications when decoding special characters (See Table 2).

Transmission is always LSB-first, therefore, when deserializing encoded symbols, DIN[0] (a) is received over a transmission line first, and DIN[9] (j) is received last.

The decoder is capable of decoding only 10-bit symbols into the appropriate characters. It contains no transmission protocol semantics and is not capable of directly manipulating serial data. When receiving serial transmitted data, logic external to the decoder is needed to convert the incoming a-to-j bitstream into the DIN <9:0> symbol that the decoder input bus requires. This external logic must also be responsible for partitioning the incoming bitstream at the proper symbol boundaries so that the decoder accurately receives the complete encoded symbol (jhgf_iedcba). Once the data is decoded, the decoder provides only the decoded byte, K value, and error flags. All error correction and data interpretation (such as interpreting a sequence of bytes as a data packet, and recognizing the K codes as proper commands/delimiters) must all be performed external to the decoder.

If more than one decoder is used to decode multiple symbols at a time into larger words, the running disparity inputs and outputs should be chained together. The RUN_DISP output of the first decoder drives the DISP_IN input of the second decoder. The final decoder's RUN_DISP output is then connected back to the DISP_IN of the first decoder.

Note that this example also requires that the input clocks to the two decoder stages be sequenced; *e.g.*, not the same clock or at least the same active clock edge.

Pinout

Signal names are shown in Figure 1 and Figure 2 and described in Table 3.

Clock - CLK (CLK_B)

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

CLK_B is the clock input for the optional secondary *B* decoder.

Clock Enable - CE (CE_B)

CE, an optional input, can be used to stall the decoder, preventing it from responding to changes on the inputs. If the decoder has a CE port and the CE input is inactive (logic 0), transitions on the clock port will have no effect. If CE is active (logic 1) or is not present, the inputs will be read and outputs updated on every rising edge of the CLK.

The CE_B pin is the Clock Enable for the *B* Decoder.

Data-Input Bus - DIN[9:0] (DIN_B[9:0])

DIN is the input bus that provides the 10-bit encoded symbol to the decoder. In the case of serial data transmission, it is important to note that DIN(0) is the least-significant bit of the 10-bit encoded symbol. DIN must be aligned on symbol boundaries (D<0>=a, D<9>=j) for the decoder to produce correct results.

DIN_B is the data input bus for the *B* decoder.

Synchronous Initialization - SINIT (SINIT_B)

SINIT, an optional input, initializes the decoder to a defined state. If SINIT is active (logic 1) and CE is active, the outputs and internal state of the decoder will be set on the rising edge of the clock. In this case, DOUT will be set to a user-defined value, KERR will be set to zero, and the internal running disparity will be set to a user-defined value. It is recommended to assert SINIT at times when the decoder's state must be known, such as before receiving a packet, to insure that the running disparity is initialized correctly before receiving.

SINIT_B is the synchronous initialization input for the *B* decoder.

Data-Output Bus - DOUT (DOUT_B)

DOUT is the decoded output byte. Subsequent to every rising clock edge (provided that CE is not present and inactive), DOUT will hold the decoded data byte. The decoded data byte is the decoded value of the 10-bit symbol on the DIN bus at the clock edge.

DOUT_B is the decoded byte output for the B decoder.

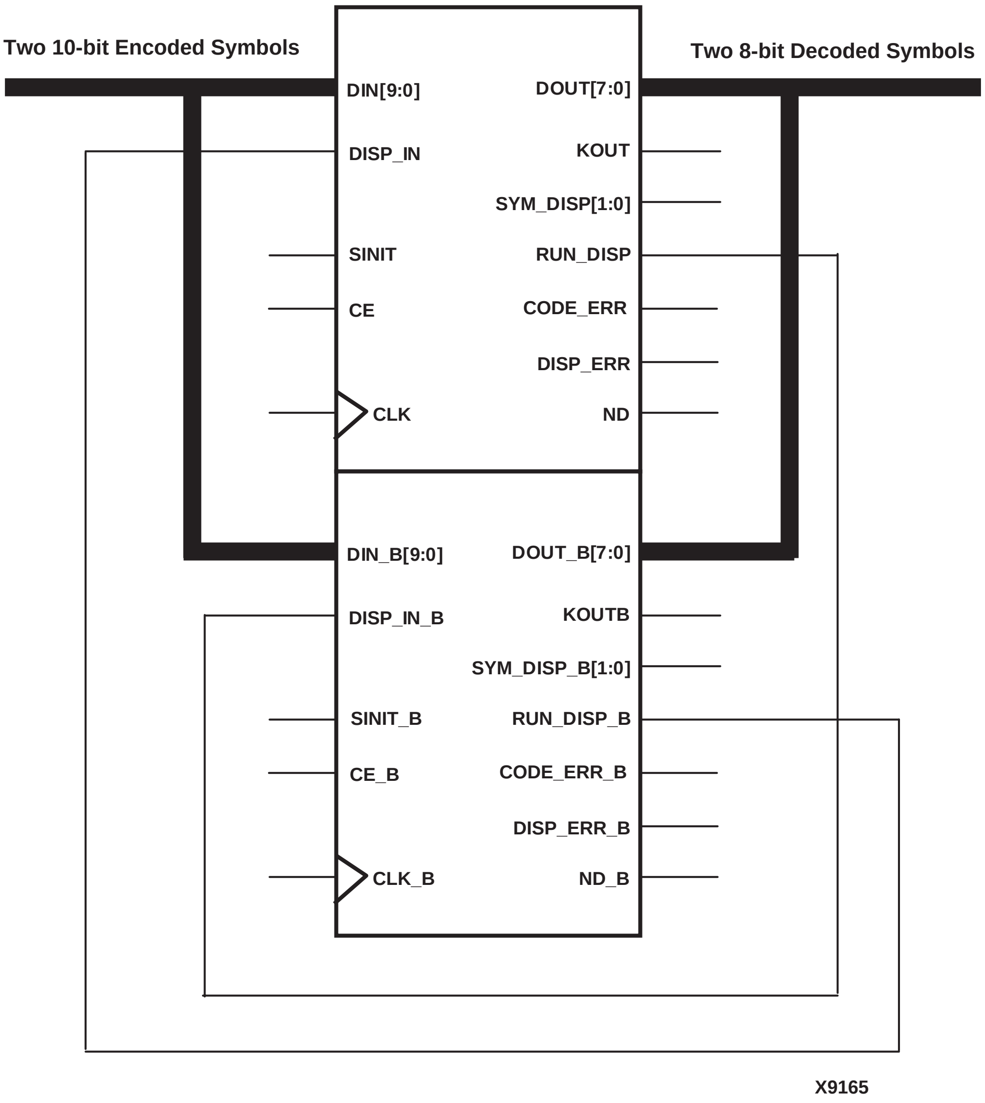

Figure 3: **Schematic Illustrating Connection of Multiple Decoders**

Command Symbol Output Flag - KOUT (KOUT_B)

KOUT indicates that the decoded 10-bit symbol was the encoding of a command symbol or special character. Like DOUT, KOUT is updated after the rising clock edge to reflect the decoded value of the input symbol on the DIN bus. When KOUT is inactive (logic 0), the 10-bit symbol being decoded is a representation of an ordinary 8-bit byte (presented on DOUT). When KOUT is active (logic 1), the 10-bit symbol is an encoding of one of the 12 special characters and the value on the DOUT bus indicates the identity of the special character.

KOUT_B is the command symbol output flag for the *B* decoder.

Disparity Input - DISP_IN (DISP_IN_B)

DISP_IN, an optional input, overrides the decoder's internal disparity. Normally, the running disparity (seen on the RUN_DISP output) is fed back, and on the rising edge of the clock added to the symbol disparity of the current symbol to produce the new running disparity. When DISP_IN is present, however, the new running disparity after the rising clock edge is defined as the symbol disparity combined with DISP_IN (instead of the old running disparity). A logic 0 asserted on the DISP_IN input indicates that an input disparity of -1 should be combined with the current symbol disparity, and a logic 1 asserted on the DISP_IN input indicates an input disparity of +1 should be used.

DISP_IN_B is the disparity input for the *B* decoder.

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

I/O Pin Name	Direction Port Status	Description
CLK (CLK_B)	Input	**Clock:** Clock input, all decoder inputs are sampled and all outputs are updated synchronously on the rising edge of the CLK input. CLK_B is the clock input for the optional secondary *B* decoder.
CE (CE_B) optional	Input	**Clock Enable:** The optional CE input port controls whether or not the decoder responds to a change in the CLK input. When present, the CE input must be Active (High) or the decoder will not change state in response to the CLK input. When not present, the decoder will always change state on the rising clock edge. CE_B is the clock enable input for the optional secondary *B* decoder.
SINIT (SINIT_B) optional	Input	**Synchronous Initialization:** When the SINIT input port is present and set Active (High), the decoder is initialized on the rising edge of the clock to a defined state. DOUT is set to a user-selectable symbol. KERR is set to 0. RUN_DISP is set to a user-defined disparity. SINIT_B is the synchronous initialization port for the *B* decoder.
DIN[9:0] (DIN_B[9:0])	Input	**Data Input:** Encoded 10-bit symbol input bus. DIN_B is the data input bus for the *B* decoder.
DISP_IN (DISP_IN_B) optional	Input	**Disparity Input:** If present, this port is a manual input for the running disparity to which the current symbol's disparity is added. If not present, the running disparity used is the decoder's internal running disparity. DISP_IN_B is the disparity input port for the *B* Decoder.
DOUT[7:0] (DOUT_B[7:0])	Output	**Data Output:** The decoded 8-bit byte resulting from decoding the input 10-bit encoded symbol. DOUT_B is the data output bus for the *B* decoder.
KOUT (KOUT_B)	Output	**K (Command) Output:** If the inputted 10-bit encoded symbol was an encoding of one of the 12 command codes, KOUT will be Active (High). KOUT_B is the command output flag for the *B* decoder.

Table 3: **Decoder Input and Output Ports** ***(Continued)***

I/O Pin Name	Direction Port Status	Description
CODE_ERR (CODE_ERR_B) optional	Output	**Code Error:** If present, CODE_ERR is Active (High) whenever the inputted 10-bit encoded symbol does not correspond to a valid member of the code set. CODE_ERR_B is the code error output flag for the *B* decoder. CODE_ERR flags errors only when the 10-bit code on the DISP_IN bus is inherently invalid. To catch all errors in an input sequence, this value must be combined with DISP_ERR.
SYM_DISP[1:0] (SYM_DISP_B[1:0]) optional	Output	**Symbol Disparity:** If present, SYM_DISP[1] designates a disparity error of the most recently decoded symbol, and SYM_DISP[0] designates the new running disparity.
RUN_DISP (RUN_DISP_B) optional	Output	**Running Disparity:** If present, RUN_DISP designates the running disparity, including the disparity of the data symbol that has been decoded. 1 = running disparity of +1 0 = running disparity of -1 RUN_DISP_B is the running disparity output for the *B* decoder.
DISP_ERR [DISP_ERR_B] optional	Output	**Disparity Error:** If present, DISP_ERR is Active (High) whenever the most recently decoded symbol violated running disparity rules. Causes for a disparity error include invalid symbols and running disparity violations at both the symbol block and sub-block levels. DISP_ERR_B is the disparity error output flag for the *B* decoder. It is possible that some codes may not produce a DISP_ERR, though the code is, by definition, invalid. For this reason, DISP_ERR should always be combined with CODE_ERR to detect all errors.
ND (ND_B) optional	Output	**New Data:** If present and CE is present, ND is Active (High) whenever the CE pin was High on the prior clock cycle, indicating that new data resides on the output pins. ND_B is the new data output flag for the *B* decoder.

Code Error Flag - CODE_ERR (CODE_ERR_B)

CODE_ERR, an optional output, indicates that the 10-bit symbol on the data-in bus during the rising edge of the clock was not a valid member of the code set. Active (logic 1) represents a CODE_ERR, and inactive (logic 0) indicates that the 10-bit code was a valid member of the code set (although other errors may still exist).

CODE_ERR_B is the code error flag for the B decoder.

Symbol Disparity - SYM_DISP[1:0] (SYM_DISP_B [1:0])

SYM_DISP, an optional output, where SYM_DISP[1] represents the disparity violation of the 10-bit symbol currently being decoded and where SYM_DISP[0] represents the new running disparity for the decoder.

The symbol disparity is combined with the current running disparity (+1 or -1) to generate the new running disparity for the decoder.

Most applications will not require symbol disparity, which is mostly useful as a debugging aid. To detect transmission faults, these implementations will monitor DISP_ERR.

SYM_DISP_B is the symbol disparity for the B decoder.

Running Disparity - RUN_DISP [RUN_DISP_B]

RUN_DISP, an optional output, exposes the current running disparity in the decoder to external logic. Running disparity, by definition, must be +1 or -1. The RUN_DISP output uses a logic 1 to represent a running disparity of +1, and a logic 0 to represent a running disparity of -1.

The running disparity, on each rising edge of the clock, is combined with the symbol disparity of the symbol being decoded to produce the new running disparity after the clock edge.

When the DISP_IN port is present, the running disparity will be based on the current 10-bit symbol and the DISP_IN pin. When the DISP_IN port is not present, the running disparity is based on the current 10-bit symbol and the decoder's internal running disparity.

The initial state of the RUN_DISP output is user configurable from the core’s GUI.

RUN_DISP_B is the running disparity of the B decoder.

Disparity Error Flag - DISP_ERR (DISP_ERR_B)

DISP_ERR, an optional output, indicates the presence of an error in the running disparity of the decoder. A disparity error (represented by a logic 1) can be caused by many things:

- If the symbol disparity of the 10-bit symbol currently being decoded does not abide by the rules for running disparity, it will produce a disparity error. It may be a valid 10-bit encoded symbol, but it must be used in the correct context of running disparity. In particular, if the current running disparity is +1, the symbol disparity must be 0 or -2. Or, if the current running disparity is -1, the symbol disparity must be 0 or +2. In general, the running disparity must always remain -1 or +1.
- The same rules for running disparity apply to the 6-bit and 4-bit symbol sub-blocks. Violating running disparity rules at this level will generate a disparity error.
- A disparity error will also occur for a majority of inputs that are not a member of the valid code set. These additional cases can also be detected because the CODE_ERR output will be set.

DISP_ERR_B is the Disparity Error output flag for the *B* Decoder.

New Data Output - ND (ND_B)

ND, an optional output, indicates that all decoder outputs have been updated on the most recent rising clock edge. This indicates to downstream logic that all outputs (DOUT, KOUT, all error flags, etc.) have been updated with the results of the newly decoded symbol at the last rising edge of the clock. The ND output is a delayed version of CE, taking on the value of CE at the rising edge of the clock on each clock cycle. If the CE input port is not present, ND is not available. ND is set to logic 0 on the clock cycle following an SINIT input of a logic 1.

ND_B is the new data output flag for the *B* decoder.

Table 4: **XCO File Parameters and Default Settings**

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_violation_out	One of the following keywords: true, false	false
b_code_violation_out	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
synchronous_reset	One of the following keywords: true, false	false
b_synchronous_reset	One of the following keywords: true, false	false
synchronous_reset_value	Virtex: D0.0 with negative running disparity. Virtex-II: D0.0 or D10.2 or D21.5 each with either a positive or negative running disparity. (Note: DOUT will be set to the synchronous reset value)	D0.0 neg
clock_enable	One of the following keywords: true, false	false
b_clock_enable	One of the following keywords: true, false	false
running_disparity_in	One of the following keywords: true, false	false
b_running_disparity_in	One of the following keywords: true, false	false
running_disparity_out	One of the following keywords: true, false	false
b_running_disparity_out	One of the following keywords: true, false	false
symbol_disparity_out	One of the following keywords: true, false	false
b_symbol_disparity_out	One of the following keywords: true, false	false
disparity_violation_out	One of the following keywords: true, false	false
b_disparity_violation_out	One of the following keywords: true, false	false
secondary_decoder	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 block RAM-based decoder requires three (4K-bit) Virtex block RAMs plus a single additional slice, as well as no more than four CLB's for the running disparity and new data logic. If a Virtex-II device is targeted, only one (16K-bit) block RAM is required, plus the same additional slice and extra logic.

For either architecture, a second decoder can be generated using the same block RAM resource, configured in a dual-port mode. The only additional logic needed to implement the second decoder is a second additional slice and no more than four more CLBs for the second set of running disparity and new data logic.

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 100 MHz.

References

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 v7.1i and later. The Xilinx CORE Generator system is bundled with all Foundation Series Development 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 on 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	Initial Xilinx release.
05/21/04	1.1	Support for v6.2i Xilinx CORE Generator system.
11/11/04	1.2	Updated definition of SYM_DISP signal in the *Core Pinout* section and software support for v6.2i of the Xilinx CORE Generator system.
4/28/05	1.3	Updated document to support 8b/10b Decoder core v7.1 and Xilinx software v7.1i.