Implementing an 8b/10b Encoder/Decoder for Gigabit Ethernet in the Actel SX FPGA Family

Introduction

This application note describes how an Actel A54SX16 FPGA was used to implement an 8b/10b encoder/decoder function for a Gigabit Ethernet router.

The Gigabit Ethernet standards committee selected 8b/10b transmission coding for the physical coding sublayer. The definition for the 8b/10b transmission code specified in the IEEE 802.3z specification is identical to that of the ANSI X3.230-1994 (Fibre Channel FC-PH) specification. The 8b/10b transmission code was developed by Albert X. Widmer and Peter A. Franaszek of IBM Corporation in the early '80s. An excellent article on the development of this transmission code was published in the September 1983 *IBM Research and Development Journal* (Vol. 27 No. 5). The article describes the logic gates required to implement the encoding and decoding functions explicitly.

The 8b/10b transmission code converts a byte wide data stream of random 1s and 0s into a DC balanced stream of 1s and 0s with a maximum run length of 5. The code must also provide sufficient signal transitions to enable reliable clock recovery. A DC balanced data stream proves to be advantageous for fiber optic and electromagnetic wire connections. The average number of 1s and 0s in the serial stream must be maintained at equal or near equal levels. The 8b/10b transmission code constrains the disparity between the number of 1s and 0s to be –2, 0, or 2 across 6 and 4 bit block boundaries. Certain 10 bit codes in the 8b/10b transmission code have a nonzero disparity value of ±2. These codes require the encoder circuitry to retain the state of the current disparity and select the appropriate ±2 value encoding pair for transmission to maintain DC balance. The coding scheme also implements additional codes for signaling, called command codes.

Implementation Requirements

There are several pin-compatible 3.3V Gigabit Ethernet transceiver devices currently available on the market. These devices provide for the serialization, clock recovery, byte synchronization, and de-serialization of the Gigabit Ethernet data stream. However, these devices do not perform the 8b/10b encoding and decoding functions. The transmitter section of the transceiver requires a 125 MHz clock and a 10 bit encoded data value with a setup time of 1.5 ns and a hold time of 1.0 ns. Because of the 1.5 ns setup time, the Tco of the encoder must be 5.5 ns, allowing a 1.0 ns provision for PCB trace delay and device to device clock skew. The receiver section of the transceiver requires two orthogonal 62.5 MHz clocks and 10 bits of data with a Tco of 5.0 to 5.5 ns and a hold time of 1.5 to 2.0 ns. The Tsu and Th for the decoder circuit must be 1.5 ns and 0 ns respectively, again allowing a 1.0 ns provision for PCB trace delay and device to device clock skew.

The Programmable Logic Challenge

There are several challenges in implementing an 8b/10b encoder/decoder (ENDEC) in a programmable logic device. The device must have sufficient 3.3V internal speed to meet the 8 ns register to register requirement of the encoder, and 3.3V I/O performance to meet the 5.0 ns Tco requirement. The low input setup requirements for the decoder would normally require the use of specialized I/O cell registers, but the dual clocking requirement resulting from the orthogonal clocks prevents this. The two 62.5 MHz orthogonal clocks allow the design to employ two identical decoder circuits, but the overall decoder design must still check the running disparity between the two decoders for the incoming data stream. A single device solution must have a minimum of three high speed, low skew clock networks to meet the design performance requirements. The Actel SX family satisfies these requirements.

The design can be segmented into two distinct non-interacting blocks for the transmission (transmitter) and reception (receiver) of 10 bit codes conforming to the IEEE 802.3z specification. These blocks can then be broken down into smaller blocks. The only signal shared between the two blocks is the asynchronous reset input. The overall ENDEC (Encoder/Decoder) device block diagram is shown in Figure 1.

The deassertion of the asynchronous reset input to the device is synchronized in two stages for each of the three clock domains. Synchronization allows the full clock cycle in each domain to distribute the asynchronous reset signal to the asynchronous preset or reset of all flip-flops. If the synchronized reset is properly buffered and timing driven place and route[1] is used, the timing can be controlled to make sure the asynchronous recovery requirement is met.

1. Timing Driven Place and Route (TDPR) is a place and route feature in the Actel Designer Series Development System that allows a designer to enter timing constraints for critical paths in a design prior to place and route. During place and route, TDPR automatically makes architecture specific trade-offs to meet the performance requirements of the design.

The asynchronous recovery requirement is the timing relationship between the asynchronous preset or clear of a flip-flop to the clock input of the flip-flop to avoid metastability. This ensures a clean initialization of the ENDEC flip-flops. Figure 2 show the logic implementation for the RESET_SYNC circuit. Note that the circuit has 8 identical outputs for fanout management.

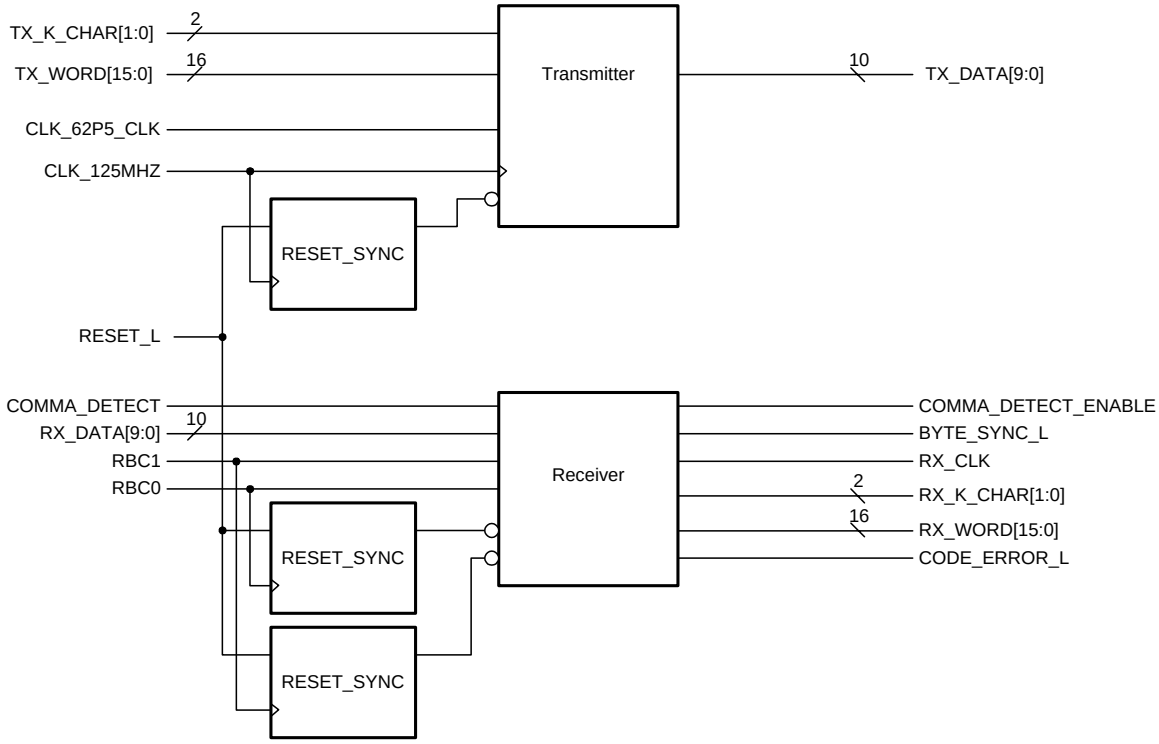

Figure 1 • *ENDEC Block Diagram*

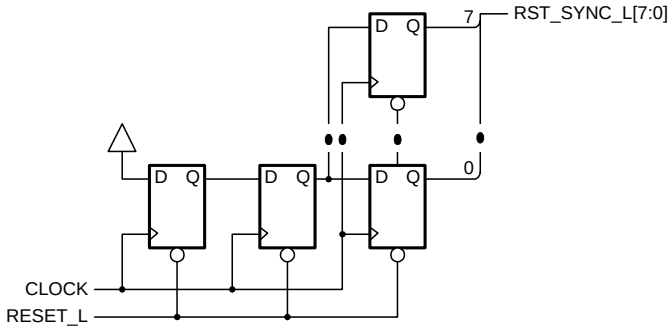

Figure 2 • *RESET_SYNC Circuit*

Terminology

The bits of the unencoded and encoded data correlate to alphabetic labels shown in Table 1.

Table 1 •

Unencoded Data Byte	Encoded Byte
	9=>j
7=>H	8=>h
6=>G	7=>g
5=>F	6=>f
	5=>i
4=>E	4=>e
3=>D	3=>d
2=>C	2=>c
1=>B	1=>b
0=>A	0=>a

Data values are referred to in the form of DX.Y or KX.Y where D indicates a data code and K indicates a command code. The X represents the integer value of the unencoded data bits EDCBA and Y represents the integer value of the unencoded data bits HGF.

The 8b/10b encoding scheme is the combination of two sub-block codes, a 5B/6B (ABCDE<=>abcdei) and a 3B/4B(FGH<=>fghj). The 10 encoded bits are serialized by the Gigabit Ethernet transceiver with bit 'a' transmitted first and bit 'j' last.

Transmitter

After converting the encoder logic functions provided in the Widmer/Franaszek article to HDL and synthesizing the description to gates, the transmitter was implemented in 10 levels of logic in the Actel SX family with a total register to register delay of 15.7 ns. However, because the encoder only needs to provide a 10 bit value on every clock cycle, the encoding process can be pipelined to meet the 8ns register to register requirement. Pipelining the encoding function was undertaken independent of the Widmer/Franaszek gates. The method chosen derives the encoded outputs from the 8b/10b found in Table 8 and Table 9. Pipelining requires that the implementation device have sufficient usable sequential elements and the architectural freedom of ample routing resources (No LAB/CLB/ input restrictions), which the Actel A54SX16 device has (528 Register-Cells).

To reduce the required clock rate for the device providing data for transmission, the transmitter was designed to work off a 16 bit input data bus with two additional inputs to select command or data codes for the high or low byte. This requires an additional 62.5 MHz clock input (TX_62P5_CLK) in phase with the 125 MHz encoder clock. All flip-flops in the transmitter design are clocked with the rising edge of the 125 MHz clock (CLK_125 MHz). The device sourcing data to the transmitter clocks the 16 bit data and command indicators with the rising edge of the TX_62P5_CLK clock signal, shown in Figure 3. The TX_62P5_CLK input is registered and distributed internally to gain control over the input setup and hold requirements for this signal. The inverter on the TX_62P5_CLK input provides additional delay on the input to reduce the hold time requirement in relation to the CLK_125 MHz input.

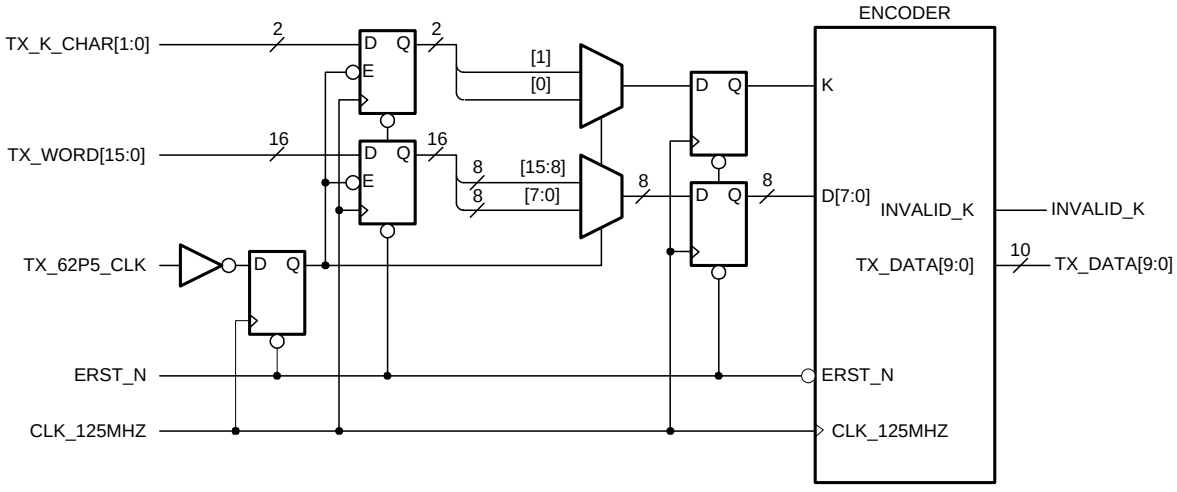

Figure 3 • *Transmitter Block Diagram*

Encoder

The first two stages of the encoder, described in the following section, are broken into three sub-modules. The outputs from each of these sub-modules are registered. Each sub-module does partial encoding of the 8 bit data input value and the command indicator and provides outputs to the third stage of the encoder. The block diagram of the encoder section can be seen in Figure 4. The third stage makes the following decisions and provides outputs to the fourth stage:

1. Determines whether the data encoding or the command encoding should be transmitted and provides the appropriate 6B and 4B codes.
2. Whether or not the 6B code provided requires inversion.
3. Whether or not the 4B code provided requires inversion.
4. Whether the transmitted encoding should flip the current value of the running disparity or not.
5. Pipelined output indicating an invalid command code was requested.

The fourth stage provides the device output registers for the final 8b/10b encoding of the data and a pipelined indicator for an invalid command code request. The output from stage 4 provides an 8b/10b 10-bit encoded value to the Gigabit Ethernet transmitter every 8 ns. The block diagram of the encoder section is shown in Figure 4.

ENC_K Block Description

This module has the following outputs:

- K_SEL[1:0]—Pipelined K character qualified by the fact that the 8 bit input code represents a valid command code (replicated twice for fanout management).
- K_ERR—Indicates that the K input was asserted, but the 8 bit input code does not represent a valid command code. This output is pipelined and indicated externally via the INVALID_K device output when the character is presented on the TX_DATA outputs.
- Kcode_6B—The 6-bit code encoding of the 5 bits (DATA[4:0]), assuming that the current running disparity is negative. If the running disparity is positive, stage 3 inverts all bits.
- Kcode_4B—The 4-bit code encoding of the 3 bits (DATA[7:5]), assuming that the current running disparity is negative. If the running disparity is positive stage 3 inverts all bits.
- KFLIP_RD—This output determines if the current running disparity should be flipped after transmission of the K code. This can be determined directly from the 8b/10b encoding Table 8.

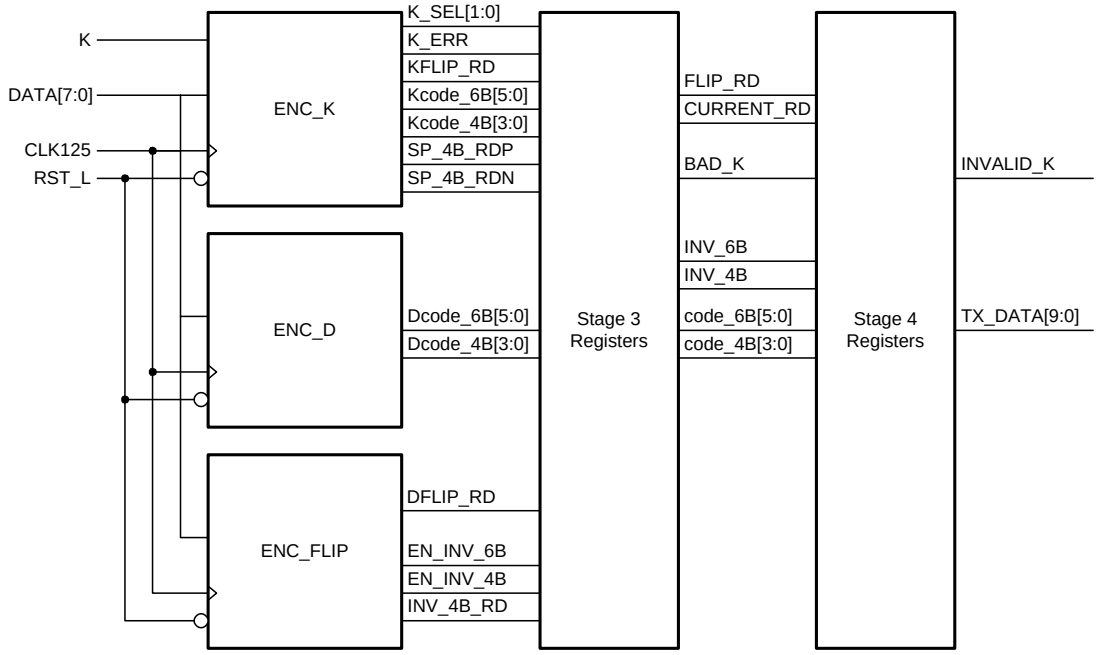

Figure 4 • *Encoder Block Diagram*

- SP_4B_RDP—Indicates one of the special 4 bit code cases when the Running Disparity (RD) is positive. If the RD is positive, the output stage forces the 4 bit code to a value of 1000 to avoid a run length 5 code (for D codes only, not K codes). The special data codes are D11.7, D13.7, or D14.7 when the current running disparity is positive. See the 8b/10b encoding Table 8.
- SP_4B_RDN—Indicates one of the special 4 bit code cases when the Running Disparity (RD) is negative. If the RD is negative, the output stage forces the 4 bit code to a value of 0111 to avoid a run length 5 code (for D codes only, not K codes). The special data codes are D17.7, D18.7, or D20.7 when the current running disparity is negative. See the 8b/10b encoding Table 8.

ENC_D Block Description

This module outputs one of the two pair of encodings for the respective 3 and 5 bit input values. The ENC_FLIP module provides additional outputs to determine whether the output value needs to be inverted or not. This can be accomplished with two ROM lookup tables. The first ROM table is 4 bits wide and 8 values deep (4x8) and the second ROM table is 6x32. The 4x8 ROM table can use a sequential HDL case statement with the registered version of the D[7:5] inputs as the selector. The implementation of the 6x32 ROM table in the Actel SX family can be done with six 32-1 multiplexors. Constant values drive the data inputs to the 32-1 multiplexors. The select lines are controlled by the registered version of the D[4:0] inputs. The output from the multiplexor is then registered. Modeling of the larger ROM table in this fashion forces synthesis tools to implement the larger ROM table in a very specific manner, which should result in 3 levels of multiplexing in the SX family.

Choosing the 4B Values for the 4x8 ROM Table

For some 3 bit input data values (1, 2, 5, and 6), the 4 bit encoding has only a single value (1001, 0101, 1010, and 0110). In addition, each of the other 3 bit input data values (0, 3, 4, and 7) have 4 bit encoding code pairs. Table 2 lists the possible encoding values:

Table 2 •

0	∞	0100 or 1011	4	∞	0010 or 1101
1	∞	1001	5	∞	1010
2	∞	0101	6	∞	0110
3	∞	0011 or 1100	7	∞	0001 or 1110 *

** D11.7, D13.7, D14.7, D17.7, D18.7, or D20.7 depending on the running disparity are special cases as noted in the preceding ENC_K section.*

At this point a choice needs to be made about which of the two pairs to select for the 4x8 ROM table. As shown in Table 8, the transmitted 4 bit encoding depends on the running disparity and the value of D[4:0]. The choice was made to place into the 4x8 ROM table the 4 bit code for the given 3 bit data value when D[4:0] is 0000 and the running disparity is negative. With this selection it is possible to determine when inversion is needed. The ENC_FLIP module provides outputs used by stage 3 to determine if inversion of the 4B value is needed.

Choosing the 6B Values for the 6x32 ROM Table

For some 5 bit input data values (3, 5, 6, 9, 10, 11, 12, 13, 14, 17, 18, 19, 20, 21, 22, 25, 26, and 28), the 6 bit encoding has only a single value. The other 4 bit input values have a pair of encoding values dependent on the running disparity. These pairs are the inverse of one another. Table 3 lists the possible encoding values.

Table 3 •

0	∞	100111 or 011000	16	∞	011011 or 100100
1	∞	011101 or 100010	17	∞	100011
2	∞	101101 or 010010	18	∞	010011
3	∞	110001	19	∞	110010
4	∞	110101 or 001010	20	∞	001011
5	∞	101001	21	∞	101010
6	∞	011001	22	∞	011010
7	∞	111000 or 000111	23	∞	111010 or 000101
8	∞	111001or 000101	24	∞	110011 or 001100
9	∞	100101	25	∞	100110
10	∞	010101	26	∞	010110
11	∞	110100	27	∞	110110 or 001001
12	∞	001101	28	∞	001110
13	∞	101100	29	∞	101110 or 010001
14	∞	011100	30	∞	011110 or 100001
15	∞	010111 or 101000	31	∞	101011 or 010100

Again a choice needs to be made as to which values to select for the 6x32 ROM table. For this discussion, the choice was made to place into the 6x32 ROM table the 6 bit codes, assuming that the running disparity is negative. The ENC_FLIP module provides outputs used by stage 3 to determine if inversion of the 6B value is needed.

ENC_FLIP Block Description

This module has the following outputs:

- DFLIP_RD—This output determines if the current running disparity should flip after transmission of the D code. This can be determined directly from the 8b/10b encoding Table 8.
- EN_INV_6B—This output is asserted for the 6B code values that have two pairs as noted in the section discussing the ENC_D block. It does not assert for the 6B code values with a single value. The third stage combines this output with the current running disparity to determine if the 6B code provided by the ENC_D module requires inversion.

- EN_INV_4B—This output is asserted for the following values of DATA[7:5]: 0, 3, 4, and 7. These are the 4B code values with two pairs. When asserted, this output indicates one of the 4B data codes that have an alternate pair. The third stage uses this output to enable inversion for the 4B data code if INV_4B_RD and the current running disparity indicate that inversion of the 4B code provided by the ENC_D module is needed.
- INV_4B_RD—This output is asserted for the following values of DATA[4:0]: 0, 1, 2, 4, 8, 15, 16, 23, 24, 27, 29, 30, and 31. The third stage indicates that inversion of the 4B data encoding provided by the ENC_D module is required under the following conditions shown in Table 4.

Table 4 •

INV_4B_RD	RD	Stage 3 Inversion Required
0	+	NO
1	+	YES
0	–	YES
1	–	NO

This is the exclusive-nor of the INV_4B_RD signal and the running disparity.

Receiver

The receiver contains two identical decoders, a merge phase block, and a word alignment state machine, as shown in Figure 5. Each decoder circuit works independently off one of the two orthogonal clocks with a clock period of 16 ns. Each decodes the 10 bit input character into its corresponding eight bit value, a K value, eight disparity functions, and provides an indicator for illegal codes. These outputs are then merged into a single clock domain stream of 16 data bits, 2 K values, and a CODE_ERROR_L output. The CODE_ERROR_L can indicate one or both of the following error conditions:

1. A character was received with the incorrect disparity. Because detection of a disparity error could potentially be from an earlier received byte and not necessarily from the current byte, the prior three bytes will also be marked as in error with the CODE_ERROR_L output. There are three stages in the merge phase to enable backward indication of code errors when disparity errors occur.
2. One of the two bytes in the 16 bit output word was derived from a received code that did not contain a valid 10 bit encoding.

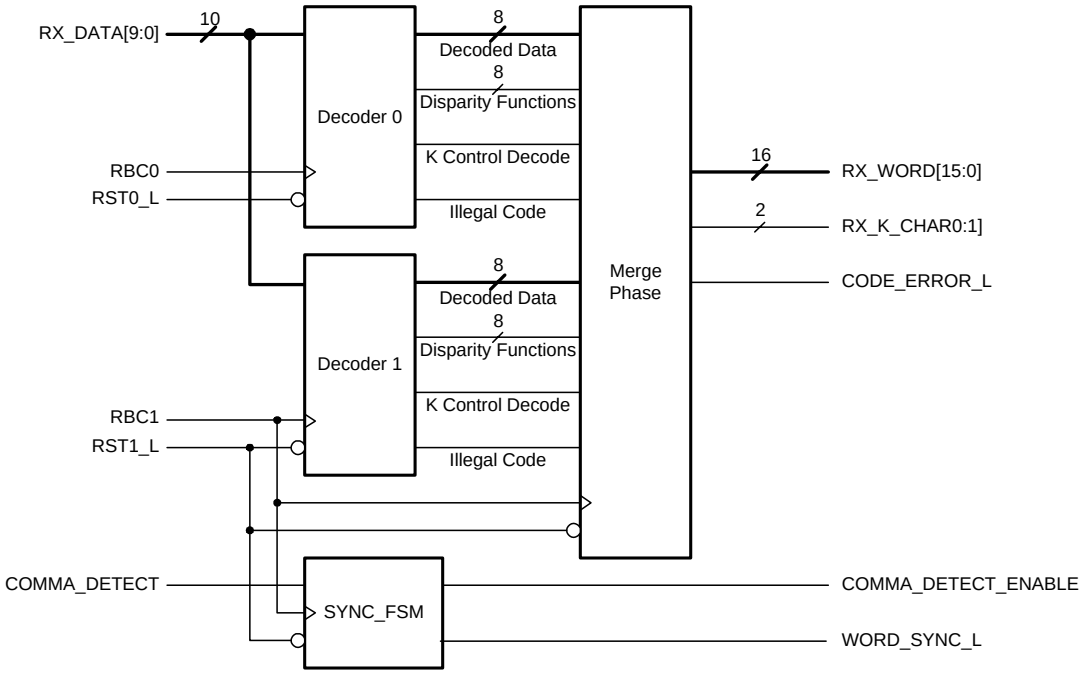

Figure 5 • *Receiver Block Diagram*

The receiver must validate that the received character had the correct disparity. One method of accomplishing this is to pass disparity information back and forth across the RBC0 and RBC1 clock domains. However, implementing the check in this fashion would require tighter timing constraints than the method chosen for this discussion. Checking the running disparity across the two decoders was solved without having to pass information between the two decoder clock domains. The 4 disparity function outputs from the decoder blocks are all captured with the rising edge of the RBC1 clock domain. The timing required from the registered outputs of the Decoder0 block to the registered inputs in the merge phase block is 8 ns, which is easily achieved and verified. Once registered in the merge phase block, the disparity functions are used to validate that the two characters received have the correct disparity. These inputs to the merge block are also used to update the current running disparity for the check of the next word of data.

Decoder

Each of the two Decoder blocks contain input registers to capture the 10 bits of received encoded data (RX_DATA[9:0] inputs) with their respective clock. Each decoder block performs four different functions; decode the 8 bit data value, calculate the 8 disparity functions, check for a command encoding, and check for illegal codes.

Data Values

The decoded value for bits 4 through 0 depend only on the encoded bits 0 to 5. Only 48 of the possible 64 values that can be received on bits 0 to 5 represent valid 8b/10b codes. Therefore, the decoding for bits 4 through 0 can be easily modeled in HDL code with a case statement. The decode values can be determined from the 8b/10b tables found in appendices A and B.

The decode of received data bits 7 through 5 are dependent on all ten of the received bits. The decoding of received bits 6 to 9 may change when bits 0 to 5 are equal to 110000, or a K28.X command encoding. Furthermore the above encoding only affects the decoding when the received bits 6 to 9 are equal to 0101, 0110, 1001, and 1010 (K28.2, K28.1, K28.6, and K28.2 respectively). Only 14 of the possible 16 values that can be received on bits 6 to 9 represent valid 8b/10b codes. Again this can be modeled in HDL code with a case statement. However, this time an “if” condition testing for the 110000 condition on received bits 0 to 5 must be used when the received bits 6 to 9 are equal to the values listed above. The decode values can be determined from the 8b/10b tables found in appendices A and B.

Disparity Functions

There are eight disparity function outputs from each decoder block. Four of these outputs are determined from the received bits 0 to 5, while the other four outputs are derived from received bits 6 to 9. The following description and understanding of these disparity functions requires naming them as follows: PD6BU, ND6BU, PD4BU, ND4BU, PD6BC, ND6BC, PD4BC, and ND4BC. The first four disparity functions determine only whether the received sub block code had positive or negative disparity. The last four outputs include the above determination plus the zero disparity cases of 000111, 111000, 0011, and 1100 respectively. The additional cases are DC balanced and have a running disparity of 0. Disparity is updated and checked at each sub block boundary. The first four disparity functions are used to update the resulting running disparity after reception of the first 6 bit sub block code and again after the reception of the 4 bit sub block code. The last four disparity functions are used to validate the received code based on the current running disparity and the disparity of the received code.

An example proves helpful to better understand the need for the additional cases in the last four disparity functions used to validate the received code. Assume that the current running disparity is positive and that the received 8b/10b code has the value 111000_1001 (D7.1). A disparity violation should be flagged. Note that the sub block code 111000 has a disparity value of zero and does not cause the current running disparity value to change. However, the proper code for transmission of D7.1 when the running disparity is positive is 000111_1001. The additional case of 000111 in the PD6BC function enables detection of this violation. The need for the additional zero disparity cases for ND6BC, PD4BC, and ND4BC can be explained a similar fashion.

Another example proves helpful in understanding why Widmer and Franaszek made this encoding choice. Assume that the current running disparity is negative and the consecutive codes D20.7 and D7.1 are received with and without a violation, as indicated in Table 5.

Table 5 •

	(RD-)	D20.7	(RD+)	D7.1	(RD+)
Violation	(RD-)	001011 0111	(RD+)	111000 1001	(RD+)
Correct	(RD-)	001011 0111	(RD+)	000111 1001	(RD+)

If the encoding of the 6 bit sub block for X=7 is 111000, as in the violation case above, a run length of six '1's results. This violates the 8b/10b encoding scheme's maximum run length limit of 5. Therefore, the encoding of X=7 when the current running disparity is positive was chosen to be 000111. Similarly, the correct value is 111000 when the current running disparity is negative.

To understand the 4 bit sub block encoding choice for Y=3, examine the encoding of the K28.3 character. The correct encoding is 001111_0011 when the running disparity is negative and 110000_1100 when positive. If the opposite choice of 1100 and 0011 had been made for Y=3, a run length of six '1's or six '0's would result.

Command Code Checking

Each decoder has a single output to indicate that the 10 bit received code was a command character. The following encodings are unique to command codes:

Each decoder has a single output to indicate that the 10 bit received code was a command character. The following encodings are unique to command codes:

- K28.X with positive running disparity indicated by bits 5 to 2 equal to 0000.
- K28.X with negative running disparity indicated by bits 5 to 2 equal to 1111.
- K23.7, K27.7 and K30.7 with positive running disparity indicated by bits 9 to 4 equal to 010111 and one single bit in 3 to 0 equal to '1' with all others equal to '0'.
- K23.7, K27.7 and K30.7 with negative running disparity indicated by bits 9 to 4 equal to 101000 and one single bit in 3 to 0 equal to '0' with all others equal to '1'.

Invalid Code Checking

The invalid code violation equations are taken directly from the Widmer/Franaszek article with two additions. The two 8b/10b codes 001111_0001 and 110000_1110 are not covered by the equations and they do not cause running disparity errors. Therefore, these two code checks have been added. The following conditions are violations of the coding rules and cause the output of Illegal_Code to be asserted:

- a = b = c = d
- P13 & !e & !i
- P31 & e & i
- f = g = h = j
- e = i = f = g = h
- i != e = g = h = j
- (e = i != g = h = j) & !(c = d = e)
- !P31 & e & !i & !g & !h & !j
- !P13 & !e & i & g & h & j
- !a & !b & c & d & e & i & !f & !g & !h & i

  = indicates that the referenced bits are the same

  & indicates a logical "AND" function.

  ! indicates a logical inversion function.

  P13 indicates that 3 or the 4 bits abcd are '0' and the other is "1', i.e., 0001.

  P31 indicates that 3 of the 4 bits avcd are '1' and the other is '0', i.e., 1110.

SYNC_FSM Block Description

The SYNC_FSM block controls the two ENDEC outputs COMMA_DETECT_ENABLE and WORD_SYNC_L. The state diagram for the finite state machine controlling these two outputs is shown in Figure 6. The Gigabit Ethernet transceiver devices provide 7-bit comma character recognition circuitry that enable the devices to word align the received data stream. The comma character pattern is 0011111XXX, where the leading zero corresponds to the first bit received and the Xs represent don't care bits. Comma characters only occur within the 10 bit command codes K28.1, K28.5, and K28.7. These codes are specifically defined to enable clock synchronization. The word alignment circuitry of the transceiver can be enabled and disabled using the COMMA_DETECT_ENABLE output from the ENDEC. When the ENDEC asserts the COMMA_DETECT_ENABLE output, the transceiver scans the incoming data stream for comma characters. When the transceiver finds a comma character it asserts the COMMA_DETECT output and aligns the 10 bit received data with the rising edge of the RBC1 clock.

The COMMA_DETECT_ENABLE output from the ENDEC design is asserted in either the SEARCH_1 or SEARCH_2 states. The WORD_SYNC_L output is asserted in any of the SYNC_0, SYNC_1, SYNC_2, or SYNC_3 states. The ENDEC captures the COMMA_DETECT input on the rising edge of RBC1 and feeds it into the SYNC_FSM block. When the finite state machine has detected two consecutive comma characters without code errors, it de-asserts the COMMA_DETECT_ENABLE output and asserts the WORD_SYNC_L output, disabling the transceiver from any additional stretching of the recovered RBC0 and RBC1 clocks. The finite state machine has hysteresis built in. The SYNC_FSM will return to search mode when 4 consecutive, or 5 out of 6 consecutive received codes are detected with code errors.

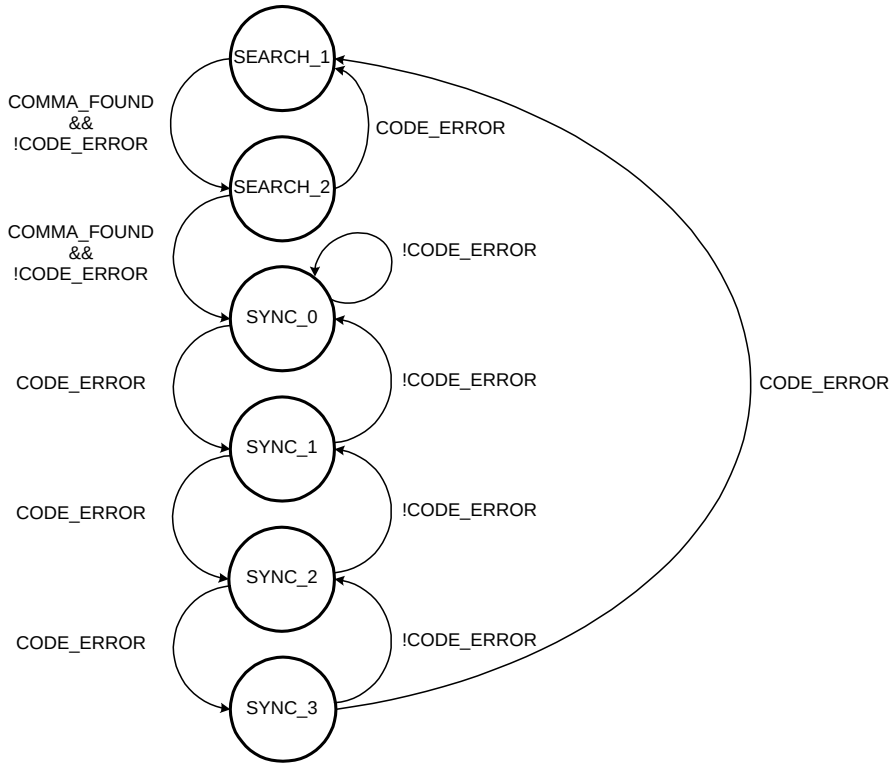

Figure 6 • *SYNC_FSM State Diagram*

Conclusion

The 8b/10b Encoder/Decoder design for Gigabit Ethernet was easily implemented in a single Actel A54SX16 –VQ100 FPGA. Table 6 and Table 7 below list device utilization and achieved timing numbers.

Table 6 • *Post-Combiner Device Utilization*

Sequential	**Used**	**311**	**Total:**	**528**	**(58.90%)**
COMB	Used	542	Total:	924	(58.66%)
LOGIC	Used	853	Total:	1452	(58.75%) (seq+comb)
I/O	Used	64	Total:	78	
CLOCK	Used	2	Total:	2	
HCLOCK	Used	1	Total:	1	

Table 7 • *A54SX16-2 Timing (WCCOM)*

Description	**Achieved**	**Required**
CLK_125MHz Reg-Reg performance	7 ns	8 ns
CLK_125MHz TX_DATA Tco performance	4.6 ns	5.5 ns
RBC0->RBC1 Reg-Reg performance	2.6 ns	8 ns
RBC0/RBC1 Reg-Reg performance	14.1 ns	16 ns
RX_DATA Tsu/Th	0.9 ns/–0.4 ns	1.5 ns/1 ns

Table 8 •

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.0	000	00000	100111_0100	011000_1011	same
D1.0	000	00001	011101_0100	100010_1011	same
D2.0	000	00010	101101_0100	010010_1011	same
D3.0	000	00011	110001_1011	110001_0100	flip
D4.0	000	00100	110101_0100	001010_1011	same
D5.0	000	00101	101001_1011	101001_0100	flip
D6.0	000	00110	011001_1011	011001_0100	flip
D7.0	000	00111	111000_1011	000111_0100	flip
D8.0	000	01000	111001_0100	000110_1011	same
D9.0	000	01001	100101_1011	100101_0100	flip
D10.0	000	01010	010101_1011	010101_0100	flip
D11.0	000	01011	110100_1011	110100_0100	flip
D12.0	000	01100	001101_1011	001101_0100	flip
D13.0	000	01101	101100_1011	101100_0100	flip
D14.0	000	01110	011100_1011	011100_0100	flip
D15.0	000	01111	010111_0100	101000_1011	same
D16.0	000	10000	011011_0100	100100_1011	same
D17.0	000	10001	100011_1011	100011_0100	flip
D18.0	000	10010	010011_1011	010011_0100	flip
D19.0	000	10011	110010_1011	110010_0100	flip
D20.0	000	10100	001011_1011	001011_0100	flip
D21.0	000	10101	101010_1011	101010_0100	flip
D22.0	000	10110	011010_1011	011010_0100	flip
D23.0	000	10111	111010_0100	000101_1011	same
D24.0	000	11000	110011_0100	001100_1011	same
D25.0	000	11001	100110_1011	100110_0100	flip
D26.0	000	11010	010110_1011	010110_0100	flip
D27.0	000	11011	110110_0100	001001_1011	same
D28.0	000	11100	001110_1011	001110_0100	flip
D29.0	000	11101	101110_0100	010001_1011	same
D30.0	000	11110	011110_0100	100001_1011	same
D31.0	000	11111	101011_0100	010100_1011	same

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.1	001	00000	100111_1001	011000_1001	flip
D1.1	001	00001	011101_1001	100010_1001	flip
D2.1	001	00010	101101_1001	010010_1001	flip
D3.1	001	00011	110001_1001	110001_1001	same
D4.1	001	00100	110101_1001	001010_1001	flip
D5.1	001	00101	101001_1001	101001_1001	same
D6.1	001	00110	011001_1001	011001_1001	same
D7.1	001	00111	111000_1001	000111_1001	same
D8.1	001	01000	111001_1001	000110_1001	flip
D9.1	001	01001	100101_1001	100101_1001	same
D10.1	001	01010	010101_1001	010101_1001	same
D11.1	001	01011	110100_1001	110100_1001	same
D12.1	001	01100	001101_1001	001101_1001	same
D13.1	001	01101	101100_1001	101100_1001	same
D14.1	001	01110	011100_1001	011100_1001	same
D15.1	001	01111	010111_1001	101000_1001	flip
D16.1	001	10000	011011_1001	100100_1001	flip
D17.1	001	10001	100011_1001	100011_1001	same
D18.1	001	10010	010011_1001	010011_1001	same
D19.1	001	10011	110010_1001	110010_1001	same
D20.1	001	10100	001011_1001	001011_1001	same
D21.1	001	10101	101010_1001	101010_1001	same
D22.1	001	10110	011010_1001	011010_1001	same
D23.1	001	10111	111010_1001	000101_1001	flip
D24.1	001	11000	110011_1001	001100_1001	flip
D25.1	001	11001	100110_1001	100110_1001	same
D26.1	001	11010	010110_1001	010110_1001	same
D27.1	001	11011	110110_1001	001001_1001	flip
D28.1	001	11100	001110_1001	001110_1001	same
D29.1	001	11101	101110_1001	010001_1001	flip
D30.1	001	11110	011110_1001	100001_1001	flip
D31.1	001	11111	101011_1001	010100_1001	flip

Table 8 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.2	010	00000	100111_0101	011000_0101	flip
D1.2	010	00001	011101_0101	100010_0101	flip
D2.2	010	00010	101101_0101	010010_0101	flip
D3.2	010	00011	110001_0101	110001_0101	same
D4.2	010	00100	110101_0101	001010_0101	flip
D5.2	010	00101	101001_0101	101001_0101	same
D6.2	010	00110	011001_0101	011001_0101	same
D7.2	010	00111	111000_0101	000111_0101	same
D8.2	010	01000	111001_0101	000110_0101	flip
D9.2	010	01001	100101_0101	100101_0101	same
D10.2	010	01010	010101_0101	010101_0101	same
D11.2	010	01011	110100_0101	110100_0101	same
D12.2	010	01100	001101_0101	001101_0101	same
D13.2	010	01101	101100_0101	101100_0101	same
D14.2	010	01110	011100_0101	011100_0101	same
D15.2	010	01111	010111_0101	101000_0101	flip
D16.2	010	10000	011011_0101	100100_0101	flip
D17.2	010	10001	100011_0101	100011_0101	same
D18.2	010	10010	010011_0101	010011_0101	same
D19.2	010	10011	110010_0101	110010_0101	
D20.2	010	10100	001011_0101	001011_0101	same
D21.2	010	10101	101010_0101	101010_0101	same
D22.2	010	10110	011010_0101	011010_0101	same
D23.2	010	10111	111010_0101	000101_0101	flip
D24.2	010	11000	110011_0101	001100_0101	flip
D25.2	010	11001	100110_0101	100110_0101	same
D26.2	010	11010	010110_0101	010110_0101	same
D27.2	010	11011	110110_0101	001001_0101	flip
D28.2	010	11100	001110_0101	001110_0101	same
D29.2	010	11101	101110_0101	010001_0101	flip
D30.2	010	11110	011110_0101	100001_0101	flip
D31.2	010	11111	101011_0101	010100_0101	flip

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.3	011	00000	100111_0011	011000_1100	flip
D1.3	011	00001	011101_0011	100010_1100	flip
D2.3	011	00010	101101_0011	010010_1100	flip
D3.3	011	00011	110001_1100	110001_0011	
D4.3	011	00100	110101_0011	001010_1100	flip
D5.3	011	00101	101001_1100	101001_0011	same
D6.3	011	00110	011001_1100	011001_0011	same
D7.3	011	00111	111000_1100	000111_0011	same
D8.3	011	01000	111001_0011	000110_1100	flip
D9.3	011	01001	100101_1100	100101_0011	same
D10.3	011	01010	010101_1100	010101_0011	same
D11.3	011	01011	110100_1100	110100_0011	same
D12.3	011	01100	001101_1100	001101_0011	same
D13.3	011	01101	101100_1100	101100_0011	same
D14.3	011	01110	011100_1100	011100_0011	same
D15.3	011	01111	010111_0011	101000_1100	flip
D16.3	011	10000	011011_0011	100100_1100	flip
D17.3	011	10001	100011_1100	100011_0011	same
D18.3	011	10010	010011_1100	010011_0011	same
D19.3	011	10011	110010_1100	110010_0011	same
D20.3	011	10100	001011_1100	001011_0011	same
D21.3	011	10101	101010_1100	101010_0011	same
D22.3	011	10110	011010_1100	011010_0011	same
D23.3	011	10111	111010_0011	000101_1100	flip
D24.3	011	11000	110011_0011	001100_1100	flip
D25.3	011	11001	100110_1100	100110_0011	same
D26.3	011	11010	010110_1100	010110_0011	same
D27.3	011	11011	110110_0011	001001_1100	flip
D28.3	011	11100	001110_1100	001110_0011	same
D29.3	011	11101	101110_0011	010001_1100	flip
D30.3	011	11110	011110_0011	100001_1100	flip
D31.3	011	11111	101011_0011	010100_1100	flip

Table 8 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.4	100	00000	100111_0010	011000_1101	same
D1.4	100	00001	011101_0010	100010_1101	same
D2.4	100	00010	101101_0010	010010_1101	same
D3.4	100	00011	110001_1101	110001_0010	flip
D4.4	100	00100	110101_0010	001010_1101	same
D5.4	100	00101	101001_1101	101001_0010	flip
D6.4	100	00110	011001_1101	011001_0010	flip
D7.4	100	00111	111000_1101	000111_0010	flip
D8.4	100	01000	111001_0010	000110_1101	same
D9.4	100	01001	100101_1101	100101_0010	flip
D10.4	100	01010	010101_1101	010101_0010	flip
D11.4	100	01011	110100_1101	110100_0010	flip
D12.4	100	01100	001101_1101	001101_0010	flip
D13.4	100	01101	101100_1101	101100_0010	flip
D14.4	100	01110	011100_1101	011100_0010	flip
D15.4	100	01111	010111_0010	101000_1101	same
D16.4	100	10000	011011_0010	100100_1101	same
D17.4	100	10001	100011_1101	100011_0010	flip
D18.4	100	10010	010011_1101	010011_0010	flip
D19.4	100	10011	110010_1101	110010_0010	flip
D20.4	100	10100	001011_1101	001011_0010	flip
D21.4	100	10101	101010_1101	101010_0010	flip
D22.4	100	10110	011010_1101	011010_0010	flip
D23.4	100	10111	111010_0010	000101_1101	same
D24.4	100	11000	110011_0010	001100_1101	
D25.4	100	11001	100110_1101	100110_0010	flip
D26.4	100	11010	010110_1101	010110_0010	flip
D27.4	100	11011	110110_0010	001001_1101	same
D28.4	100	11100	001110_1101	001110_0010	flip
D29.4	100	11101	101110_0010	010001_1101	same
D30.4	100	11110	011110_0010	100001_1101	same
D31.4	100	11111	101011_0010	010100_1101	same

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.5	101	00000	100111_1010	011000_1010	flip
D1.5	101	00001	011101_1010	100010_1010	flip
D2.5	101	00010	101101_1010	010010_1010	flip
D3.5	101	00011	110001_1010	110001_1010	same
D4.5	101	00100	110101_1010	001010_1010	flip
D5.5	101	00101	101001_1010	101001_1010	same
D6.5	101	00110	011001_1010	011001_1010	same
D7.5	101	00111	111000_1010	000111_1010	same
D8.5	101	01000	111001_1010	000110_1010	flip
D9.5	101	01001	100101_1010	100101_1010	same
D10.5	101	01010	010101_1010	010101_1010	same
D11.5	101	01011	110100_1010	110100_1010	same
D12.5	101	01100	001101_1010	001101_1010	same
D13.5	101	01101	101100_1010	101100_1010	same
D14.5	101	01110	011100_1010	011100_1010	same
D15.5	101	01111	010111_1010	101000_1010	flip
D16.5	101	10000	011011_1010	100100_1010	flip
D17.5	101	10001	100011_1010	100011_1010	same
D18.5	101	10010	010011_1010	010011_1010	same
D19.5	101	10011	110010_1010	110010_1010	same
D20.5	101	10100	001011_1010	001011_1010	same
D21.5	101	10101	101010_1010	101010_1010	same
D22.5	101	10110	011010_1010	011010_1010	same
D23.5	101	10111	111010_1010	000101_1010	flip
D24.5	101	11000	110011_1010	001100_1010	flip
D25.5	101	11001	100110_1010	100110_1010	same
D26.5	101	11010	010110_1010	010110_1010	same
D27.5	101	11011	110110_1010	001001_1010	flip
D28.5	101	11100	001110_1010	001110_1010	same
D29.5	101	11101	101110_1010	010001_1010	flip
D30.5	101	11110	011110_1010	100001_1010	flip
D31.5	101	11111	101011_1010	010100_1010	flip

Table 8 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.6	110	00000	100111_0110	011000_0110	flip
D1.6	110	00001	011101_0110	100010_0110	flip
D2.6	110	00010	101101_0110	010010_0110	flip
D3.6	110	00011	110001_0110	110001_0110	same
D4.6	110	00100	110101_0110	001010_0110	flip
D5.6	110	00101	101001_0110	101001_0110	same
D6.6	110	00110	011001_0110	011001_0110	same
D7.6	110	00111	111000_0110	000111_0110	same
D8.6	110	01000	111001_0110	000110_0110	flip
D9.6	110	01001	100101_0110	100101_0110	same
D10.6	110	01010	010101_0110	010101_0110	same
D11.6	110	01011	110100_0110	110100_0110	
D12.6	110	01100	001101_0110	001101_0110	same
D13.6	110	01101	101100_0110	101100_0110	same
D14.6	110	01110	011100_0110	011100_0110	same
D15.6	110	01111	010111_0110	101000_0110	flip
D16.6	110	10000	011011_0110	100100_0110	flip
D17.6	110	10001	100011_0110	100011_0110	same
D18.6	110	10010	010011_0110	010011_0110	same
D19.6	110	10011	110010_0110	110010_0110	same
D20.6	110	10100	001011_0110	001011_0110	same
D21.6	110	10101	101010_0110	101010_0110	same
D22.6	110	10110	011010_0110	011010_0110	same
D23.6	110	10111	111010_0110	000101_0110	flip
D24.6	110	11000	110011_0110	001100_0110	same
D25.6	110	11001	100110_0110	100110_0110	same
D26.6	110	11010	010110_0110	010110_0110	same
D27.6	110	11011	110110_0110	001001_0110	flip
D28.6	110	11100	001110_0110	001110_0110	same
D29.6	110	11101	101110_0110	010001_0110	flip
D30.6	110	11110	011110_0110	100001_0110	flip
D31.6	110	11111	101011_0110	010100_0110	flip

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
D0.7	111	00000	100111_0001	011000_1110	same
D1.7	111	00001	011101_0001	100010_1110	same
D2.7	111	00010	101101_0001	010010_1110	same
D3.7	111	00011	110001_1110	110001_0001	flip
D4.7	111	00100	110101_0001	001010_1110	same
D5.7	111	00101	101001_1110	101001_0001	flip
D6.7	111	00110	011001_1110	011001_0001	flip
D7.7	111	00111	111000_1110	000111_0001	flip
D8.7	111	01000	111001_0001	000110_1110	same
D9.7	111	01001	100101_1110	100101_0001	flip
D10.7	111	01010	010101_1110	010101_0001	flip
D11.7	111	01011	110100_1110	110100_1000	flip
D12.7	111	01100	001101_1110	001101_0001	flip
D13.7	111	01101	101100_1110	101100_1000	flip
D14.7	111	01110	011100_1110	011100_1000	flip
D15.7	111	01111	010111_0001	101000_1110	same
D16.7	111	10000	011011_0001	100100_1110	same
D17.7	111	10001	100011_0111	100011_0001	flip
D18.7	111	10010	010011_0111	010011_0001	flip
D19.7	111	10011	110010_1110	110010_0001	flip
D20.7	111	10100	001011_0111	001011_0001	flip
D21.7	111	10101	101010_1110	101010_0001	flip
D22.7	111	10110	011010_1110	011010_0001	flip
D23.7	111	10111	111010_0001	000101_1110	same
D24.7	111	11000	110011_0001	001100_1110	same
D25.7	111	11001	100110_1110	100110_0001	flip
D26.7	111	11010	010110_1110	010110_0001	flip
D27.7	111	11011	110110_0001	001001_1110	same
D28.7	111	11100	001110_1110	001110_0001	flip
D29.7	111	11101	101110_0001	010001_1110	same
D30.7	111	11110	011110_0001	100001_1110	same
D31.7	111	11111	101011_0001	010100_1110	same

Table 8 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei_fghj	CURRENT RD+ abcdei_fghj	RD
K28.0	000	000	001111_0100	110000_1011	same
K28.1	001	001	001111_1001	110000_0110	flip
K28.2	010	010	001111_0101	110000_1010	flip
K28.3	011	011	001111_0011	110000_1100	flip
K28.4	100	100	001111_0010	110000_1101	same
K28.5	101	101	001111_1010	110000_0101	flip
K28.6	110	110	001111_0110	110000_1001	flip
K28.7	111	111	001111_1000	110000_0111	same
K23.7	111	111	111010_1000	000101_0111	same
K27.7	111	111	110110_1000	001001_0111	same
K29.7	111	111	101110_1000	010001_0111	same
K30.7	111	111	011110_1000	100001_0111	same

Table 9 •

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei fghj	CURRENT RD+ abcdei fghj	RD
D0.0	000	00000	100111 0100	011000 1011	same
D0.1	001	00000	100111 1001	011000 1001	flip
D0.2	010	00000	100111 0101	011000 0101	flip
D0.3	011	00000	100111 0011	011000 1100	flip
D0.4	100	00000	100111 0010	011000 1101	same
D0.5	101	00000	100111 1010	011000 1010	flip
D0.6	110	00000	100111 0110	011000 0110	flip
D0.7	111	00000	100111 0001	011000 1110	same
D1.0	000	00001	011101 0100	100010 1011	same
D1.1	001	00001	011101 1001	100010 1001	flip
D1.2	010	00001	011101 0101	100010 0101	flip
D1.3	011	00001	011101 0011	100010 1100	flip
D1.4	100	00001	011101 0010	100010 1101	same
D1.5	101	00001	011101 1010	100010 1010	flip
D1.6	110	00001	011101 0110	100010 0110	flip
D1.7	111	00001	011101 0001	100010 1110	same
D2.0	000	00010	101101 0100	010010 1011	same
D2.1	001	00010	101101 1001	010010 1001	flip
D2.2	010	00010	101101 0101	010010 0101	flip
D2.3	011	00010	101101 0011	010010 1100	flip
D2.4	100	00010	101101 0010	010010 1101	same
D2.5	101	00010	101101 1010	010010 1010	flip
D2.6	110	00010	101101 0110	010010 0110	flip
D2.7	111	00010	101101 0001	010010 1110	same
D3.0	000	00011	110001 1011	110001 0100	flip
D3.1	001	00011	110001 1001	110001 1001	same
D3.2	010	00011	110001 0101	110001 0101	same
D3.3	011	00011	110001 1100	110001 0011	same
D3.4	100	00011	110001 1101	110001 0010	flip
D3.5	101	00011	110001 1010	110001 1010	same
D3.6	110	00011	110001 0110	110001 0110	same
D3.7	111	00011	110001 1110	110001 0001	flip
D4.0	000	00100	110101 0100	001010 1011	same
D4.1	001	00100	110101 1001	001010 1001	flip
D4.2	010	00100	110101 0101	001010 0101	flip
D4.3	011	00100	110101 0011	001010 1100	flip
D4.4	100	00100	110101 0010	001010 1101	same
D4.5	101	00100	110101 1010	001010 1010	flip
D4.6	110	00100	110101 0110	001010 0110	flip
D4.7	111	00100	110101 0001	001010 1110	same
D5.0	000	00101	101001 1011	101001 0100	flip
D5.1	001	00101	101001 1001	101001 1001	same
D5.2	010	00101	101001 0101	101001 0101	same
D5.3	011	00101	101001 1100	101001 0011	same
D5.4	100	00101	101001 1101	101001 0010	flip
D5.5	101	00101	101001 1010	101001 1010	same
D5.6	110	00101	101001 0110	101001 0110	same
D5.7	111	00101	101001 1110	101001 0001	flip
D6.0	000	00110	011001 1011	011001 0100	flip
D6.1	001	00110	011001 1001	011001 1001	same
D6.2	010	00110	011001 0101	011001 0101	same
D6.3	011	00110	011001 1100	011001 0011	same
D6.4	100	00110	011001 1101	011001 0010	flip
D6.5	101	00110	011001 1010	011001 1010	same
D6.6	110	00110	011001 0110	011001 0110	same
D6.7	111	00110	011001 1110	011001 0001	flip
D7.0	000	00111	111000 1011	000111 0100	flip
D7.1	001	00111	111000 1001	000111 1001	same
D7.2	010	00111	111000 0101	000111 0101	same
D7.3	011	00111	111000 1100	000111 0011	same
D7.4	100	00111	111000 1101	000111 0010	flip
D7.5	101	00111	111000 1010	000111 1010	same
D7.6	110	00111	111000 0110	000111 0110	same
D7.7	111	00111	111000 1110	000111 0001	flip

Table 9 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei fghj	CURRENT RD+ abcdei fghj	RD
D8.0	000	01000	111001 0100	000110 1011	same
D8.1	001	01000	111001 1001	000110 1001	flip
D8.2	010	01000	111001 0101	000110 0101	flip
D8.3	011	01000	111001 0011	000110 1100	flip
D8.4	100	01000	111001 0010	000110 1101	same
D8.5	101	01000	111001 1010	000110 1010	flip
D8.6	110	01000	111001 0110	000110 0110	flip
D8.7	111	01000	111001 0001	000110 1110	same
D9.0	000	01001	100101 1011	100101 0100	flip
D9.1	001	01001	100101 1001	100101 1001	same
D9.2	010	01001	100101 0101	100101 0101	same
D9.3	011	01001	100101 1100	100101 0011	same
D9.4	100	01001	100101 1101	100101 0010	flip
D9.5	101	01001	100101 1010	100101 1010	same
D9.6	110	01001	100101 0110	100101 0110	same
D9.7	111	01001	100101 1110	100101 0001	flip
D10.0	000	01010	010101 1011	010101 0100	flip
D10.1	001	01010	010101 1001	010101 1001	same
D10.2	010	01010	010101 0101	010101 0101	same
D10.3	011	01010	010101 1100	010101 0011	same
D10.4	100	01010	010101 1101	010101 0010	flip
D10.5	101	01010	010101 1010	010101 1010	same
D10.6	110	01010	010101 0110	010101 0110	same
D10.7	111	01010	010101 1110	010101 0001	flip
D11.0	000	01011	110100 1011	110100 0100	flip
D11.1	001	01011	110100 1001	110100 1001	same
D11.2	010	01011	110100 0101	110100 0101	same
D11.3	011	01011	110100 1100	110100 0011	same
D11.4	100	01011	110100 1101	110100 0010	flip
D11.5	101	01011	110100 1010	110100 1010	same
D11.6	110	01011	110100 0110	110100 0110	same
D11.7	111	01011	110100 1110	110100 1000	flip
D12.0	000	01100	001101 1011	001101 0100	flip
D12.1	001	01100	001101 1001	001101 1001	same
D12.2	010	01100	001101 0101	001101 0101	same
D12.3	011	01100	001101 1100	001101 0011	same
D12.4	100	01100	001101 1101	001101 0010	flip
D12.5	101	01100	001101 1010	001101 1010	same
D12.6	110	01100	001101 0110	001101 0110	same
D12.7	111	01100	001101 1110	001101 0001	flip
D13.0	000	01101	101100 1011	101100 0100	flip
D13.1	001	01101	101100 1001	101100 1001	same
D13.2	010	01101	101100 0101	101100 0101	same
D13.3	011	01101	101100 1100	101100 0011	same
D13.4	100	01101	101100 1101	101100 0010	flip
D13.5	101	01101	101100 1010	101100 1010	same
D13.6	110	01101	101100 0110	101100 0110	same
D13.7	111	01101	101100 1110	101100 1000	flip
D14.0	000	01110	011100 1011	011100 0100	flip
D14.1	001	01110	011100 1001	011100 1001	same
D14.2	010	01110	011100 0101	011100 0101	same
D14.3	011	01110	011100 1100	011100 0011	same
D14.4	100	01110	011100 1101	011100 0010	flip
D14.5	101	01110	011100 1010	011100 1010	same
D14.6	110	01110	011100 0110	011100 0110	same
D14.7	111	01110	011100 1110	011100 1000	flip
D15.0	000	01111	010111 0100	101000 1011	same
D15.1	001	01111	010111 1001	101000 1001	flip
D15.2	010	01111	010111 0101	101000 0101	flip
D15.3	011	01111	010111 0011	101000 1100	flip
D15.4	100	01111	010111 0010	101000 1101	same
D15.5	101	01111	010111 1010	101000 1010	flip
D15.6	110	01111	010111 0110	101000 0110	flip
D15.7	111	01111	010111 0001	101000 1110	same

Table 9 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei fghj	CURRENT RD+ abcdei fghj	RD
D16.0	000	10000	011011 0100	100100 1011	same
D16.1	001	10000	011011 1001	100100 1001	flip
D16.2	010	10000	011011 0101	100100 0101	flip
D16.3	011	10000	011011 0011	100100 1100	flip
D16.4	100	10000	011011 0010	100100 1101	same
D16.5	101	10000	011011 1010	100100 1010	flip
D16.6	110	10000	011011 0110	100100 0110	flip
D16.7	111	10000	011011 0001	100100 1110	same
D17.0	000	10001	100011 1011	100011 0100	flip
D17.1	001	10001	100011 1001	100011 1001	same
D17.2	010	10001	100011 0101	100011 0101	same
D17.3	011	10001	100011 1100	100011 0011	same
D17.4	100	10001	100011 1101	100011 0010	flip
D17.5	101	10001	100011 1010	100011 1010	same
D17.6	110	10001	100011 0110	100011 0110	same
D17.7	111	10001	100011 0111	100011 0001	flip
D18.0	000	10010	010011 1011	010011 0100	flip
D18.1	001	10010	010011 1001	010011 1001	same
D18.2	010	10010	010011 0101	010011 0101	same
D18.3	011	10010	010011 1100	010011 0011	same
D18.4	100	10010	010011 1101	010011 0010	flip
D18.5	101	10010	010011 1010	010011 1010	same
D18.6	110	10010	010011 0110	010011 0110	same
D18.7	111	10010	010011 0111	010011 0001	flip
D19.0	000	10011	110010 1011	110010 0100	flip
D19.1	001	10011	110010 1001	110010 1001	same
D19.2	010	10011	110010 0101	110010 0101	same
D19.3	011	10011	110010 1100	110010 0011	same
D19.4	100	10011	110010 1101	110010 0010	flip
D19.5	101	10011	110010 1010	110010 1010	same
D19.6	110	10011	110010 0110	110010 0110	same
D19.7	111	10011	110010 1110	110010 0001	flip
D20.0	000	10100	001011 1011	001011 0100	flip
D20.1	001	10100	001011 1001	001011 1001	same
D20.2	010	10100	001011 0101	001011 0101	same
D20.3	011	10100	001011 1100	001011 0011	same
D20.4	100	10100	001011 1101	001011 0010	flip
D20.5	101	10100	001011 1010	001011 1010	same
D20.6	110	10100	001011 0110	001011 0110	same
D20.7	111	10100	001011 0111	001011 0001	flip
D21.0	000	10101	101010 1011	101010 0100	flip
D21.1	001	10101	101010 1001	101010 1001	same
D21.2	010	10101	101010 0101	101010 0101	same
D21.3	011	10101	101010 1100	101010 0011	same
D21.4	100	10101	101010 1101	101010 0010	flip
D21.5	101	10101	101010 1010	101010 1010	same
D21.6	110	10101	101010 0110	101010 0110	same
D21.7	111	10101	101010 1110	101010 0001	flip
D22.0	000	10110	011010 1011	011010 0100	flip
D22.1	001	10110	011010 1001	011010 1001	same
D22.2	010	10110	011010 0101	011010 0101	same
D22.3	011	10110	011010 1100	011010 0011	same
D22.4	100	10110	011010 1101	011010 0010	flip
D22.5	101	10110	011010 1010	011010 1010	same
D22.6	110	10110	011010 0110	011010 0110	same
D22.7	111	10110	011010 1110	011010 0001	flip
D23.0	000	10111	111010 0100	000101 1011	same
D23.1	001	10111	111010 1001	000101 1001	flip
D23.2	010	10111	111010 0101	000101 0101	flip
D23.3	011	10111	111010 0011	000101 1100	flip
D23.4	100	10111	111010 0010	000101 1101	same
D23.5	101	10111	111010 1010	000101 1010	flip
D23.6	110	10111	111010 0110	000101 0110	flip
D23.7	111	10111	111010 0001	000101 1110	same

Table 9 • *(Continued)*

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei fghj	CURRENT RD+ abcdei fghj	RD
D24.0	000	11000	110011 0100	001100 1011	same
D24.1	001	11000	110011 1001	001100 1001	flip
D24.2	010	11000	110011 0101	001100 0101	flip
D24.3	011	11000	110011 0011	001100 1100	flip
D24.4	100	11000	110011 0010	001100 1101	same
D24.5	101	11000	110011 1010	001100 1010	flip
D24.6	110	11000	110011 0110	001100 0110	same
D24.7	111	11000	110011 0001	001100 1110	same
D25.0	000	11001	100110 1011	100110 0100	flip
D25.1	001	11001	100110 1001	100110 1001	same
D25.2	010	11001	100110 0101	100110 0101	same
D25.3	011	11001	100110 1100	100110 0011	same
D25.4	100	11001	100110 1101	100110 0010	flip
D25.5	101	11001	100110 1010	100110 1010	same
D25.6	110	11001	100110 0110	100110 0110	same
D25.7	111	11001	100110 1110	100110 0001	flip
D26.0	000	11010	010110 1011	010110 0100	flip
D26.1	001	11010	010110 1001	010110 1001	same
D26.2	010	11010	010110 0101	010110 0101	same
D26.3	011	11010	010110 1100	010110 0011	same
D26.4	100	11010	010110 1101	010110 0010	flip
D26.5	101	11010	010110 1010	010110 1010	same
D26.6	110	11010	010110 0110	010110 0110	same
D26.7	111	11010	010110 1110	010110 0001	flip
D27.0	000	11011	110110 0100	001001 1011	same
D27.1	001	11011	110110 1001	001001 1001	flip
D27.2	010	11011	110110 0101	001001 0101	flip
D27.3	011	11011	110110 0011	001001 1100	flip
D27.4	100	11011	110110 0010	001001 1101	same
D27.5	101	11011	110110 1010	001001 1010	flip
D27.6	110	11011	110110 0110	001001 0110	flip
D27.7	111	11011	110110 0001	001001 1110	same

DX.Y	Y HGF	X EDCBA	CURRENT RD- abcdei fghj	CURRENT RD+ abcdei fghj	RD
D28.0	000	11100	001110 1011	001110 0100	flip
D28.1	001	11100	001110 1001	001110 1001	same
D28.2	010	11100	001110 0101	001110 0101	same
D28.3	011	11100	001110 1100	001110 0011	same
D28.4	100	11100	001110 1101	001110 0010	flip
D28.5	101	11100	001110 1010	001110 1010	same
D28.6	110	11100	001110 0110	001110 0110	same
D28.7	111	11100	001110 1110	001110 0001	flip
D29.0	000	11101	101110 0100	010001 1011	same
D29.1	001	11101	101110 1001	010001 1001	flip
D29.2	010	11101	101110 0101	010001 0101	flip
D29.3	011	11101	101110 0011	010001 1100	flip
D29.4	100	11101	101110 0010	010001 1101	same
D29.5	101	11101	101110 1010	010001 1010	flip
D29.6	110	11101	101110 0110	010001 0110	flip
D29.7	111	11101	101110 0001	010001 1110	same
D30.0	000	11110	011110 0100	100001 1011	same
D30.1	001	11110	011110 1001	100001 1001	flip
D30.2	010	11110	011110 0101	100001 0101	flip
D30.3	011	11110	011110 0011	100001 1100	flip
D30.4	100	11110	011110 0010	100001 1101	same
D30.5	101	11110	011110 1010	100001 1010	flip
D30.6	110	11110	011110 0110	100001 0110	flip
D30.7	111	11110	011110 0001	100001 1110	same
D31.0	000	11111	101011 0100	010100 1011	same
D31.1	001	11111	101011 1001	010100 1001	flip
D31.2	010	11111	101011 0101	010100 0101	flip
D31.3	011	11111	101011 0011	010100 1100	flip
D31.4	100	11111	101011 0010	010100 1101	same
D31.5	101	11111	101011 1010	010100 1010	flip
D31.6	110	11111	101011 0110	010100 0110	flip
D31.7	111	11111	101011 0001	010100 1110	same

Take it to a higher level.

http://www.actel.com

Actel Europe Ltd.
Daneshill House, Lutyens Close
Basingstoke, Hampshire RG24 8AG
United Kingdom
Tel: +44.(0)1256.305600
Fax: +44.(0)1256.355420

Actel Corporation
955 East Arques Avenue
Sunnyvale, California 94086
USA
Tel: 408.739.1010
Fax: 408.739.1540

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ka
Tokyo 150 Japan
Tel: +81.(0)3.3445.7671
Fax: +81.(0)3.3445.7668

5192650-0/10.98