ML450 Bit Error Rate Tester (BERT) User Guide

UG089 (v1.1) December 2, 2005

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/13/05	1.0	Initial Xilinx release.
12/02/05	1.1	Added Chapter 4, “SPI-4.2 Interface,” and Chapter 5, “16-Channel DDR Interface (With SPI-4.2 Training Pattern).”

Table of Contents

About This Guide

This user guide describes the components and operation of the ML450 Bit Error Rate Tester (BERT) and its accompanying graphical user interface (GUI).

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Introduction," defines the components that are part of the ML450 reference design.
- Chapter 2, "SFI-4 Interface," describes the SFI-4 interface.
- Chapter 3, "16-Channel DDR Interface (Bit-Aligned)," describes the 16 Channel DDR bit alignment.
- Chapter 4, "SPI-4.2 Interface," discusses the interface between two ML450 boards.
- Chapter 5, "16-Channel DDR Interface (With SPI-4.2 Training Pattern)"describes a interface similar to the 16-channel DDR interface but uses the training pattern specified in the SPI 4.2 training protocol to perform both bit alignment and bus alignment.
- Chapter 6, "BERT and GUI," describes the operation of the BERT and the GUI.
- Chapter 7, "Getting Started with the Demonstration," lists the steps needed to begin using the ML450 reference design.

Additional Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at: http://www.xilinx.com/support.

For additional information on implementing an SFI-4 Interface, visit: Xilinx Application Note XAPP704: *Virtex-4 High-Speed Single Data Rate LVDS Transceiver* by Markus Adhiwiyogo.

For additional information on implementing a per-bit deskew algorithm in a source synchronous receiver, visit Xilinx Application Note XAPP705: *Virtex-4 High-Speed Dual Data Rate LVDS* by Markus Adhiwiyogo.

For additional information on implementing dynamic phase alignment (DPA) in a source synchronous receiver, visit XAPP700: *Dynamic Phase Alignment for Networking Applications* by Tze Yi Yeoh.

Typographical Conventions

This document uses the following typographical conventions. An example illustrates each convention.

Convention	Meaning or Use	Example
Italic font	References to other documents	See the Virtex-4 *Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page or document.	http://www.xilinx.com/virtex4

Chapter 1

Introduction

The ML450 reference design consists of three distinct components: the high-speed data interface, the bit error rate tester (BERT), and the graphical user interface (GUI). The BERT and the high-speed interface operate in asynchronous clock domains, independent of one another. The BERT can tolerate an interface running at any data rate within the specifications of the device.

The BERT can generate and detect data patterns. It is used to test a physical data interface by sending and receiving data through that interface. When the data is received, the BERT produces statistics about the accuracy of the received data, providing information about the integrity of the data interface. These statistics are reported to a GUI via a serial port on the ML450 board. The GUI is included as part of this reference design.

The BERT can attach to many interfaces. This document only discusses its attachment to an SFI-4 bus-aligned single data rate (SDR) interface and a 16-channel bit-aligned double data rate (DDR) interface.

Figure 1-1 shows the different clock domains of the BERT driving an SFI-4 interface.

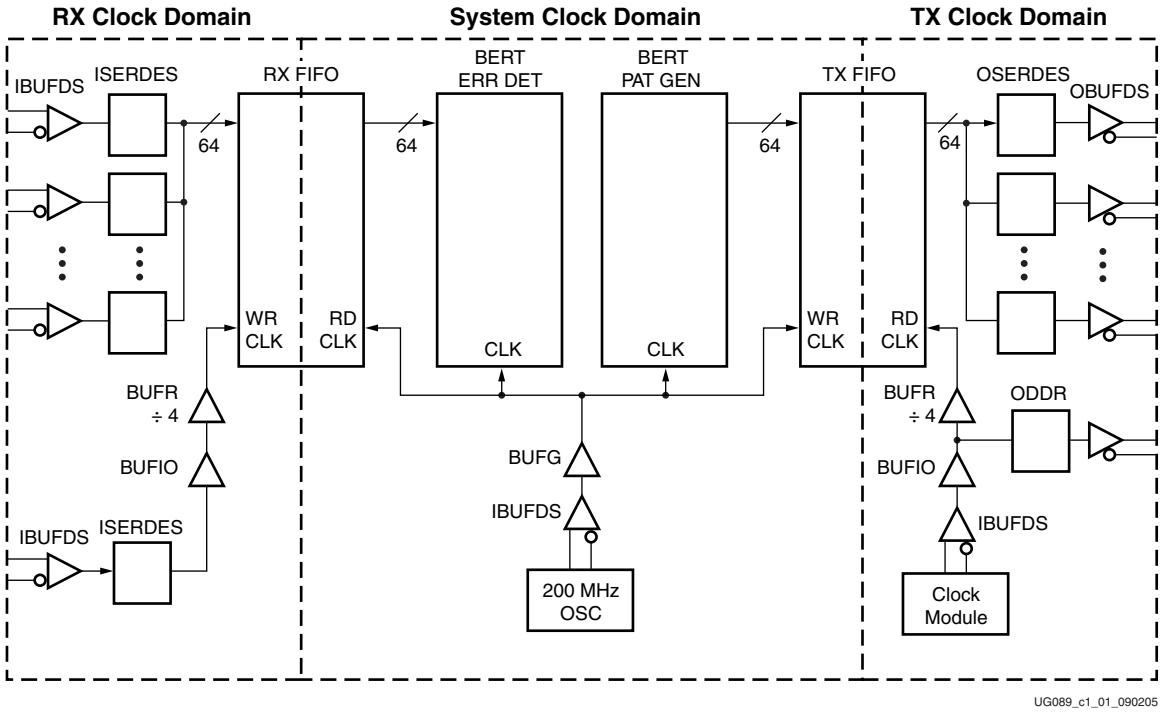


Figure 1-1: **BERT Clock Domains Driving an SFI-4 Interface**

Chapter 2

SFI-4 Interface

The SFI-4 interface consists of 16 LVDS data channels and one forwarded clock. Because the deserialization is 4:1, there are 64 bits in the parallel side of the interface. The method of aligning clock to data on the receive side is done using bus alignment, meaning all 16 data channels are treated as a single bus during the alignment process. This method requires the designer to assume that the 16 data channels have minimal skew between them, because any skew directly subtracts from the data valid window.

Bus-Alignment Overview

To align clock and data, the clock is sampled by an ISERDES as if it were a data channel. Assuming that there is little skew between the sampled clock and data (because they have the same path), the rising edges of the clock align with the transitions of the data, and the falling edges of the clock align with the center of the data eye. Given this relationship, an algorithm can delay the sampled clock and data until the falling edge is found, indicating that the sampling point is in the middle of the data eye.

Figure 2-1 shows the method of sampling the clock as a virtual data channel.

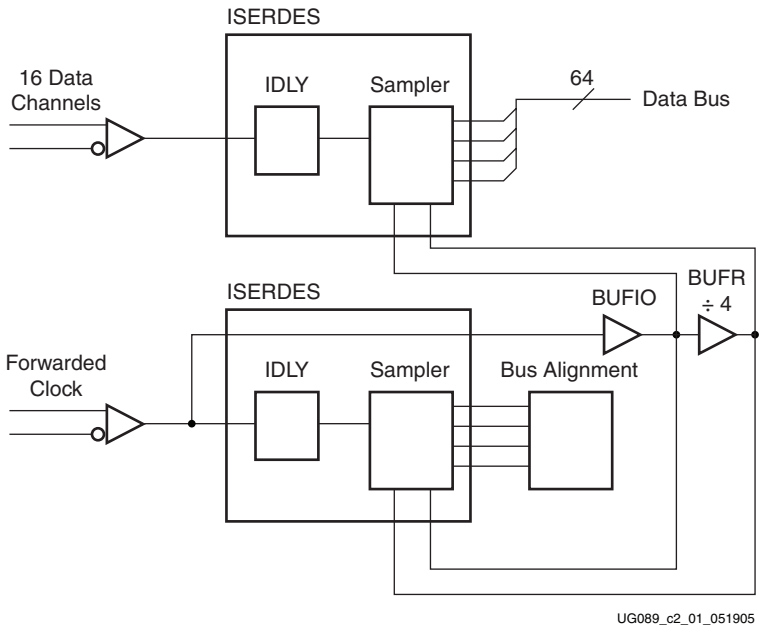


Figure 2-1: **Sampling the Clock as a Virtual Data Channel**

Figure 2-2 shows the timing relationship of the clock, sampled clock, and data before training has occurred. The actual clock arrives after the data because of the delay added by the BUFIO component (offset in part by the additional delay incurred on the data path due to IDELAY). The formula for determining the theoretical amount of delay required to align the rising edge of the clock to the center of the data eye is:

Total Delay Required = (DLY_{BUFIO}) – (DLY_{IDLY} at tap 0) + (½ Clock Period)

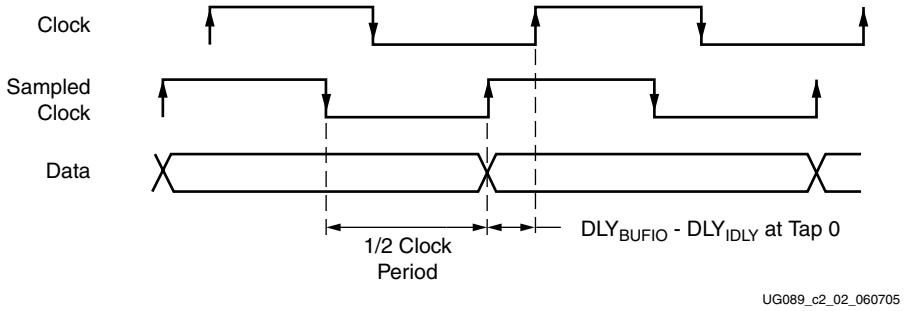


Figure 2-2: **Clock and Data Timing Relationships**

Bus-Alignment Implementation

Figure 2-3 shows the state machine that interrogates the sampled clock to find the falling edge. The sampled clock can have one of three possible values:

- a one
- a zero
- an ambiguous value

With the current design configuration, the sampled clock always starts out as a *one*, because it arrives before the actual clock. However, the training algorithm supports all cases.

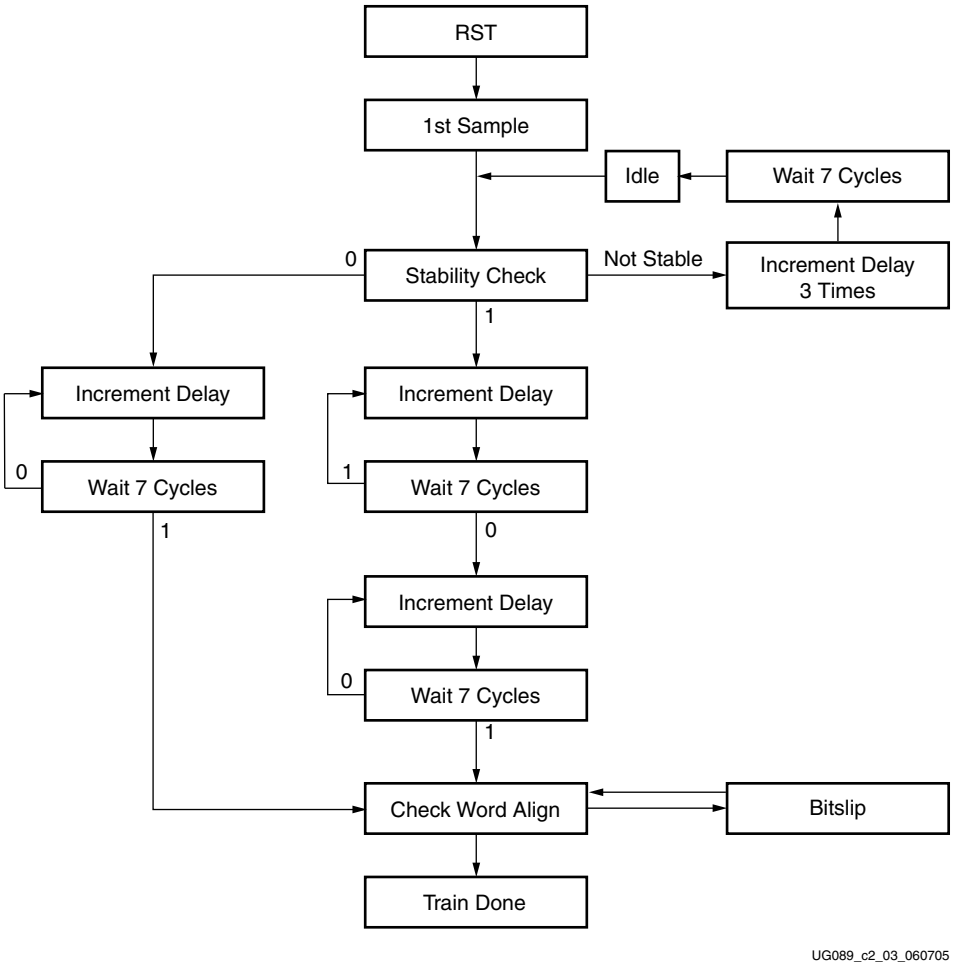


Figure 2-3: **Training Algorithm**

The process of word alignment begins after the bus alignment completes (shown in the last three states of Figure 2-3). Table 2-1 shows the performance of the bus-alignment algorithm compared to the theoretical calculation of the perfect delay setting for various frequencies. The results come from hardware testing on an ML450 board with an XC4VLX25-11.

Table 2-1: **Bus-Alignment Algorithm Performance**

Frequency (MHz)	Tap Delay Setting (From Formula)	Tap Delay Setting (Measured)	Amount of Error (Taps)
200	37.33	37	0
250	30.67	31	0
300	26.22	27	1
350	23.05	23	0
400	20.67	21	0
450	18.81	18	1

Table 2-1: **Bus-Alignment Algorithm Performance *(Continued)***

Frequency (MHz)	Tap Delay Setting (From Formula)	Tap Delay Setting (Measured)	Amount of Error (Taps)
500	17.33	17	1
550	16.12	16	0
600	15.11	15	0
620	14.75	14	1
640	14.42	14	0
700	13.52	13	1

Design Hierarchy

Figure 2-4 shows the hierarchy of HDL source files in the SFI-4 BERT design.

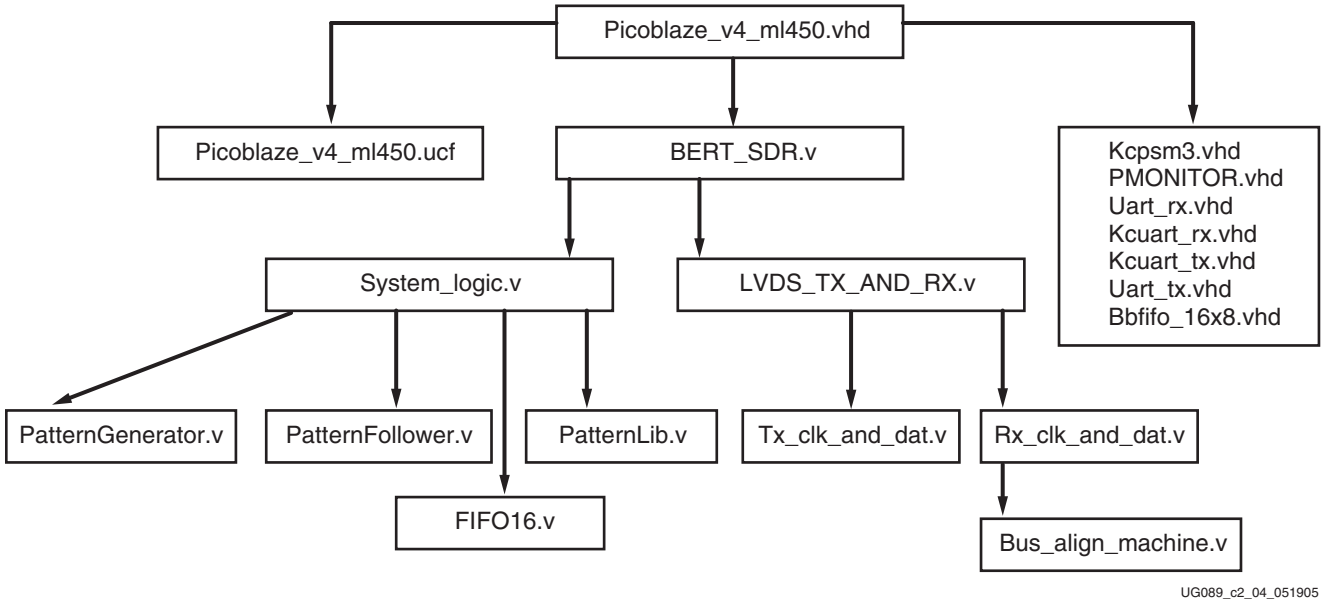


Figure 2-4: **HDL Source File Hierarchy**

Chapter 3

16-Channel DDR Interface (Bit-Aligned)

The 16-channel DDR interface (similar to SPI-4.2) consists of 16 LVDS data channels and one forwarded clock. The deserialization is 8:1, so there are 128 bits in the parallel side of the interface. The method of aligning clock to data on the receive side is done using bit alignment, meaning all 16 data channels are treated independently during the alignment process. This removes any error due to the skew between data channels. For a bus-aligned design, the skew directly subtracts from the data valid window.

Bit-Alignment Overview

To align clock and data, a training pattern is transmitted on all data lines immediately after a reset. The training pattern is a clock pattern of alternating ones and zeroes, where the bit-alignment logic is used to align the center of the data with the rising or falling edge of the clock. The data is delayed in the alignment process, while the clock remains constant.

Figure 3-1 shows the timing relationship of the clock and data before training has occurred. The actual clock arrives after the data because of the delay added by the BUFIO component (offset in part by the additional delay incurred on the data path due to IDELAY). The formula for determining the theoretical amount of delay required to align the rising edge of the clock to the center of the data eye is:

Total Delay Required = (DLY_{BUFIO}) - (DLY_{IDLY} at tap 0) + (1/4 Clock Period)

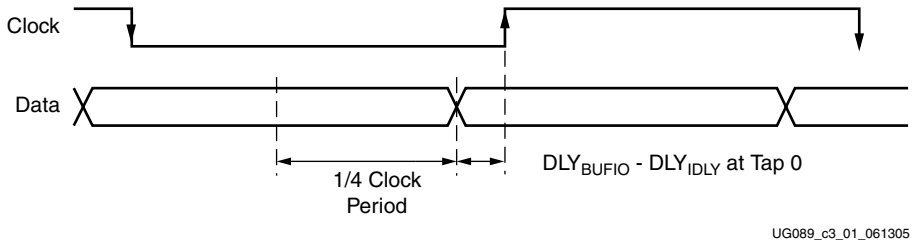


Figure 3-1: **Clock and Data Timing Relationships**

To determine the ¼ clock period position, the algorithm traverses the entire DDR data eye, counting the taps required for the traversal. At this point, the algorithm decrements the delay by half the number of taps it took to traverse the eye.

After the center of the data eye is found, the transmitter begins sending the word alignment pattern, which is the following sequence: `10111111`. The receiver uses this sequence to word align each channel. After both bit alignment and word alignment are complete, the transmitter sends actual data.

Bit-Alignment Implementation

The training algorithm that interrogates each data channel to find the center of the data valid window is shown in the state machine in Figure 3-2. Three possible relationships between clock and data can exist at the beginning of the training process: data can be a one, a zero, or an ambiguous value. With the current design configuration, the data always starts out as a *1*, because it arrives before the actual clock. However, all cases are supported by the algorithm.

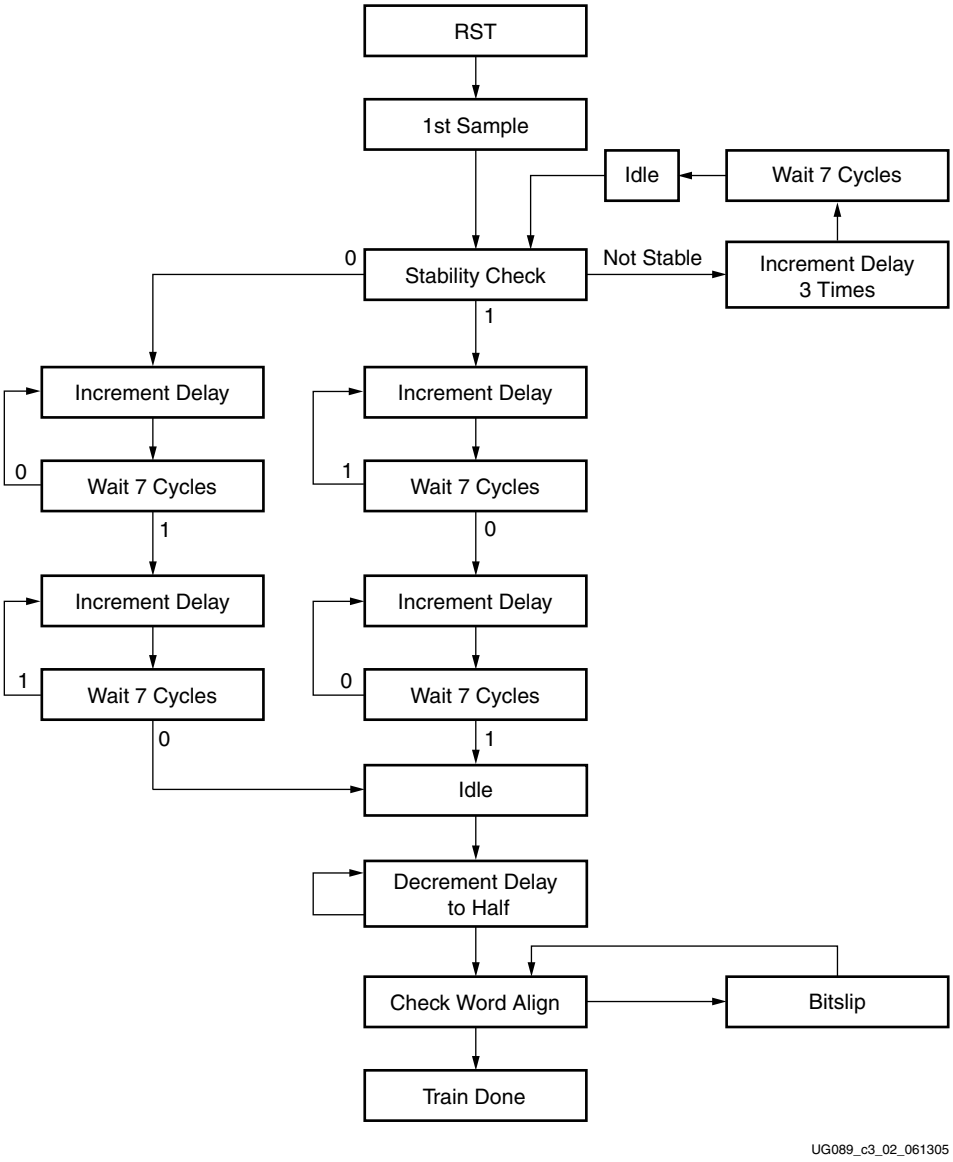


Figure 3-2: **Training Algorithm**

The performance of the bit-alignment algorithm can be evaluated using the GUI (described in detail in Chapter 6, "BERT and GUI". Table 3-1 shows a case tested on an ML450 board with an XC4VLX25-11 device. The frequency of the interface in the test case is 500 MHz, and the details of the alignment process for each of the 16 channels are shown.

Table 3-1: **Training Statistics of DDR Bit Alignment**

Frequency = 500 MHz Training Pattern = 1010					
Channel	**Taps to Transition**	**Eye Width Theoretical (taps)**	**Eye Width Measured (taps)**	**Decrements to Center of Eye (taps)**	**Final Setting (taps)**
0	6	13.3	14	8	12
1	2	13.3	14	8	8
2	4	13.3	14	8	10
3	6	13.3	13	7	12
4	4	13.3	14	8	10
5	3	13.3	13	7	9
6	5	13.3	14	8	11
7	2	13.3	14	8	8
8	4	13.3	13	7	10
9	5	13.3	14	8	11
10	6	13.3	14	8	12
11	3	13.3	13	7	9
12	4	13.3	13	7	10
13	2	13.3	15	8	9
14	6	13.3	14	8	12
15	3	13.3	14	8	9

The Taps to Transition column in Table 3-1 shows the skew between the 16 channels before the training process compensates for that skew. Channel 1 arrives only two taps before the clock, while channel 3 arrives six taps before the clock, indicating a maximum of four taps of skew between the channels. The final settings of the bit-alignment algorithm account for this skew, as shown in the last column of Table 3-1. The final settings range between 8 and 12 taps, a range of four taps, or 300 ps.

The actual data eye measured for all 16 channels has very little variance, and it matches the theoretical value to within one or two taps. By using the GUI to manually manipulate the data delay, the clock can be verified to be sampling in the center of the eye. However, when driving a pseudo-random pattern, it may not appear to be in the center because the bit-alignment algorithm uses a clock pattern for training. A pseudo-random pattern has much broader frequency content and causes the data eye to close, sometimes asymmetrically. To effectively measure the accuracy of the algorithm, the GUI is used to set the transmitted pattern to a clock pattern. After profiling the data eye using the procedure in section "Reset/Data Delay", the pattern is changed to a pseudo-random bit sequence and repeats the test. This provides a view of eye closure due to pattern jitter.

Design Hierarchy

Figure 3-3 shows the hierarchy of HDL source files in the 16-channel DDR BERT design.

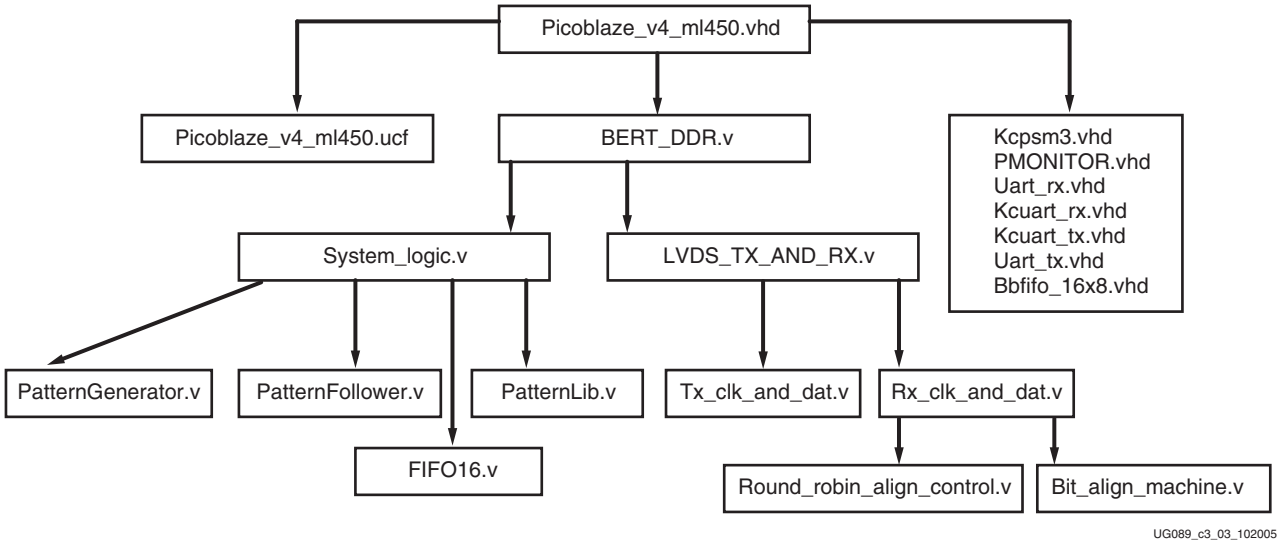


Figure 3-3: **HDL Source File Hierarchy in 16-Channel DDR BERT**

Chapter 4

SPI-4.2 Interface

The SPI-4.2 interface consists of 16 LVDS data channels and one forwarded clock. This version of the BERT uses the SPI-4.2 LogiCORE™ design. The deserialization is 4:1, and there are 64 bits in the parallel side of the interface. The method of aligning clock to data on the receive side is done using bit alignment, meaning all 16 data channels are treated independently during the alignment process. This removes any error due to skew between data channels. This alignment method is very similar to the implementation of the 16-channel DDR interface presented in Chapter 3. However, the training pattern used in the SPI-4.2 implementation is quite different and requires a different alignment algorithm.

Bit-Alignment Overview

To align clock and data, a training pattern is transmitted on all data lines immediately after a reset. The SPI-4.2 training pattern is a 20-bit sequence of 10 ones followed by 10 zeroes. This sequence repeats until the training algorithm has aligned the center of each data channel with the rising (or falling) edge of the clock. The data is delayed in the alignment process, while the clock remains constant.

The SPI-4.2 training procedure requires that data be received on the master ISERDES and the slave ISERDES for each channel. The Virtex-4 architecture supports this parallel receiver implementation because it allows a designer to align clock and data using any type of training pattern or even no training pattern at all. This is achieved by setting the initial delay of the slave ISERDES to a value slightly greater than the master ISERDES. An algorithm then increases the delays on both slave and master ISERDES until the slave and master data do not match. This mismatch indicates that a transition in the data eye has been reached.

The timing relationship of the clock, master data, and slave data before training is shown in Figure 4-1. The slave data path is initialized to delay tap = 2, which places it 150 ps behind the master datapath. To determine the ¼ clock period position, the algorithm compares master and slave data to identify two edges of a data eye. The midpoint between those two edges is the center of the data eye. The algorithm then decrements the delay to place it at the midpoint.

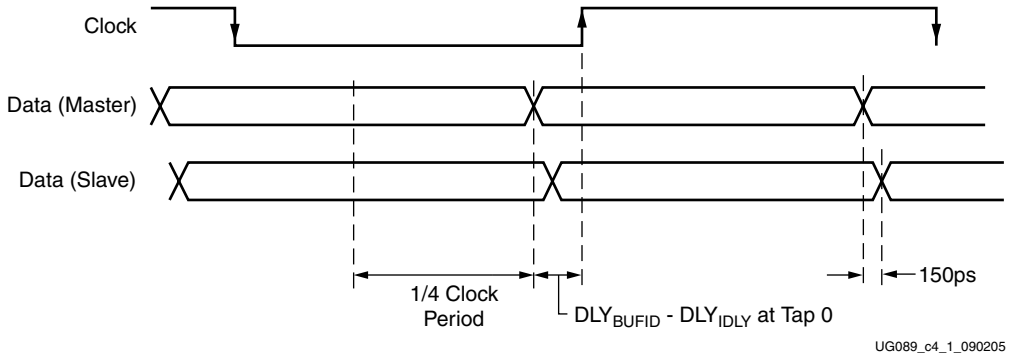


Figure 4-1: **Timing Relationship Before Training**

Word alignment is also performed on each channel after all channels have completed the alignment procedure. The training pattern for word alignment is the same as the alignment training pattern: `1111111110000000000`. By asserting the BITSLIP signal until detecting a *C* in the data stream, it is guaranteed that all 16 channels are vertically aligned with one another. See Figure 4-2. The *C* is present in only one of the four cases. The word alignment procedure could just as easily be done by searching for an *8*, *1*, *7*, or *E* because those are also exclusive values in Figure 4-2.

Each BITSLIP causes the pattern to “slip” this way.

7	3	1	0
0	0	0	0
0	0	8	C
E	F	F	F
F	F	F	F

UG089_04_090705

Figure 4-2: **20-Bit Training Pattern During Several Assertions of BITSLIP**

Bit-Alignment Implementation

The training algorithm that interrogates each data channel to find the center of the data valid window is shown in the state machine in Figure 4-3. After reset, master data is compared to slave data for 64 cycles. If they are identical during all 64 cycles, then the sampling point is somewhere within the data eye. The data delay is incremented until a transition is found, indicating the edge of the eye. After the first edge of the eye is found, a counter keeps track of the number of increment pulses it takes to reach the second transition. This number is the width of the data eye in terms of delay taps. By then decrementing to half of that number, the data channel is center-aligned, reaching the LOCK state.

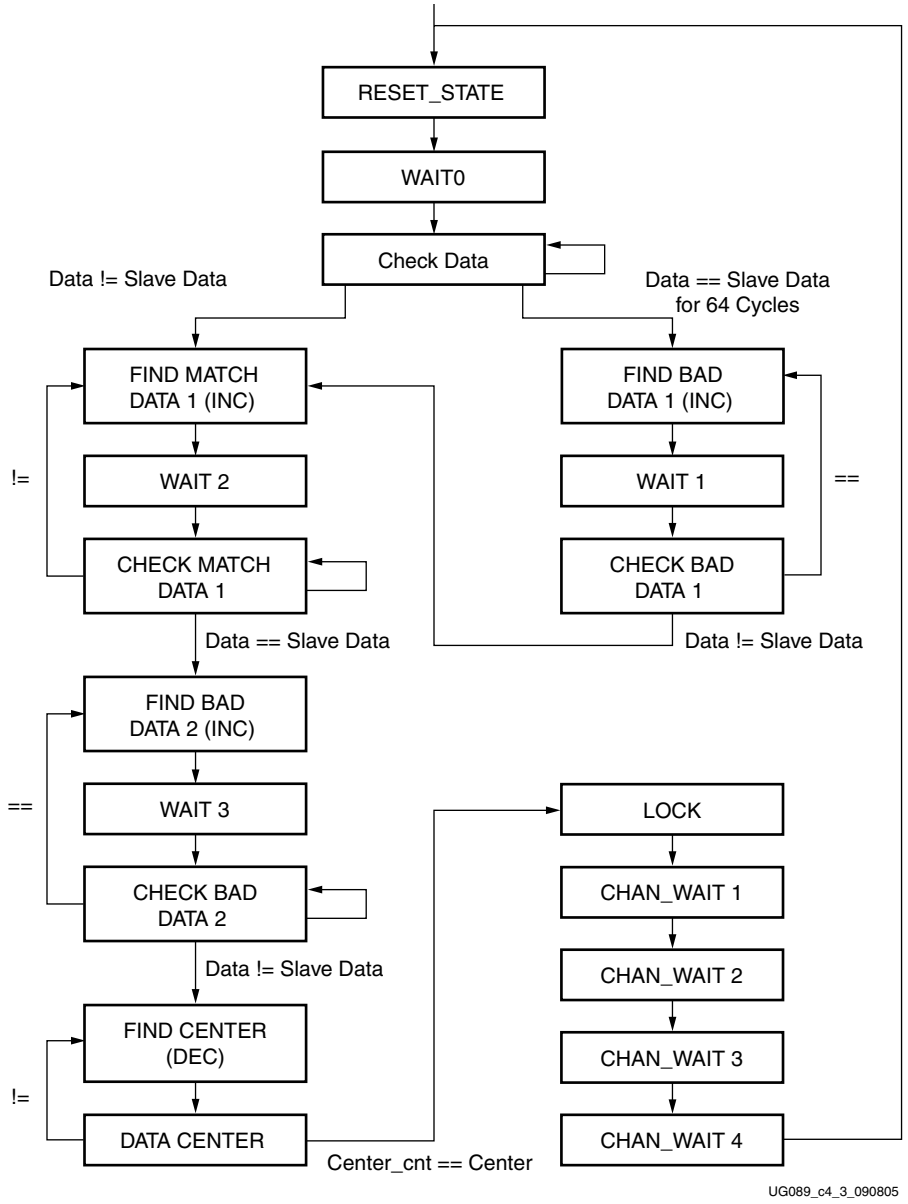


Figure 4-3: **Training Algorithm**

Design Hierarchy

Figure 4-4 shows the hierarchy of HDL source files in the SPI-4.2 interface BERT.

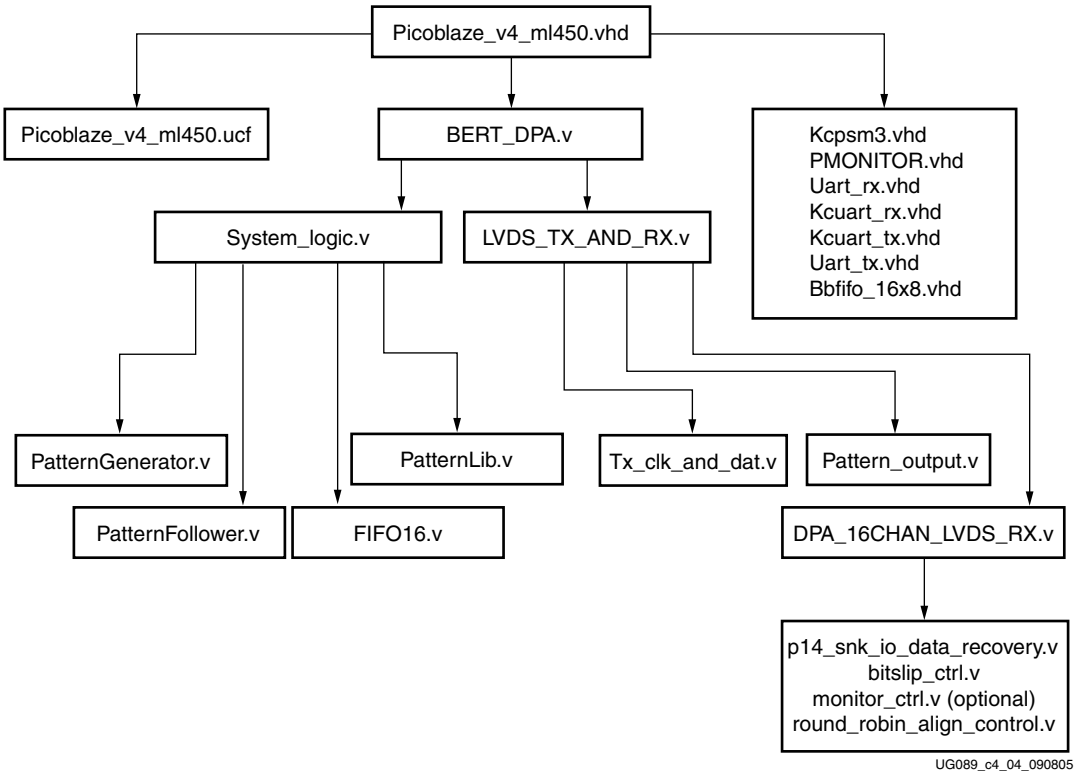


Figure 4-4: **SPI-4.2 Interface Hierarchy**

XILINX®

Chapter 5

16-Channel DDR Interface (With SPI-4.2 Training Pattern)

Chapter 3 described a 16-channel, source-synchronous DDR interface that performs bit alignment based on a training pattern consisting of a clock pattern of alternating ones and zeroes. This chapter describes a similar interface but uses the training pattern specified in the SPI-4.2 training protocol to perform both bit alignment and bus alignment. The SPI-4.2 protocol requires the receiving device to correct for up to ±1 bit time of data skew. The deserialization factor is 4:1, so there are 64 bits in the parallel side of the interface. The bit-alignment algorithm corrects for clock-to-data skew and channel-to-channel skew up to but not including one bit time, and it positions the data eye in such a way as to maximize the clock sampling margin. The bus-alignment algorithm corrects for one bit time of channel-to-channel skew. The bit alignment and bus alignment has been described in great detail in XAPP 700: *Dynamic Phase Alignment for Networking Applications* and is summarized here.

Training Overview

Data recovery and bus deskew are essential in many source-synchronous networking interfaces. Data may arrive at the FPGA with channel-to-channel skew due to layout constraints, resulting in trace length differences. To deskew the channels and properly align the bus in the proper word boundary, networking protocols such as SPI-4.2 require the transmitter to send a training pattern to the receiver during initialization. The SPI-4.2 training protocol is a 20-bit training pattern comprised of 10 zeros and 10 ones (`1111_1111_1100_0000_0000`) and is sent repeatedly by the transmitter on every channel as long as it is in training mode. The receiver executes a set of algorithms based on the training pattern it receives to align the data with respect to the sampling clock edge and align the data channels with respect to each other.

There are three components to the training algorithm:

- Bit alignment
- Word alignment
- Real-time window monitoring

The training function comprises the bit alignment and word alignment functions and occurs during the initialization phase. Bit alignment corrects for data skew of less than one bit period by positioning the clock edge at the center of the data eye. Word alignment corrects for data skew greater than one bit period by aligning the incoming pattern to the pre-specified training pattern. After the training phase is complete, the training module goes into monitor mode, continually scanning the incoming data stream to ensure the clock is always positioned at the center of the data eye. The training algorithm uses the Virtex-4 ChipSync™ features, including:

- Dedicated serial-to-parallel converter
- Bitslip sub-module
- Programmable 64-tap delay line

An overview block diagram of the training function is shown in Figure 5-1. Only one channel is shown. Some of the interconnections are simplified to avoid redundancy.

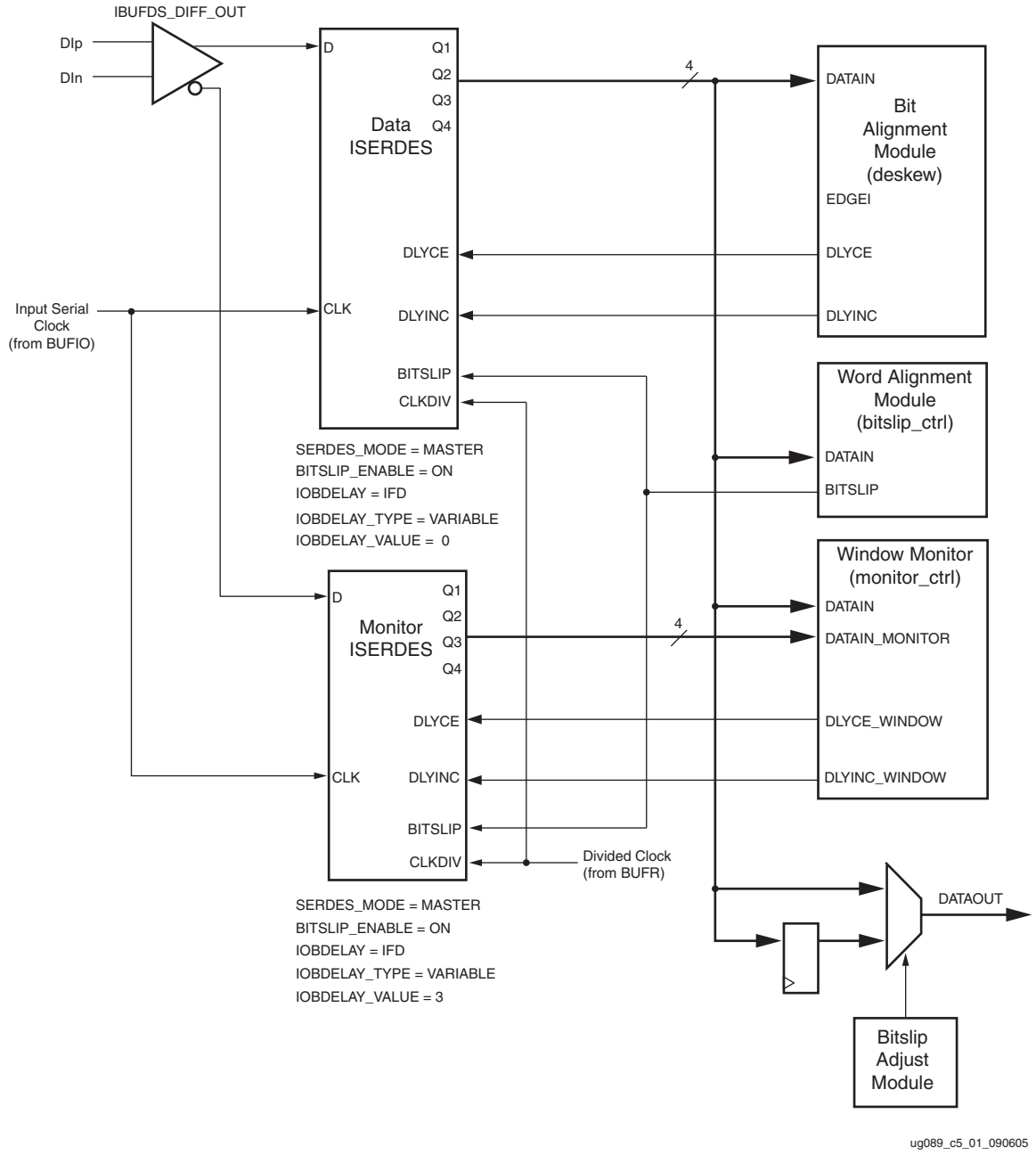


Figure 5-1: **Training Function Block Diagram**

Each ISERDES module can deserialize data up to six bits wide. In this design, data is deserialized to a 4-bit parallel word. Because LVDS is used as the signaling standard, two

IOBs and, therefore, two ISERDES modules are available per channel. One ISERDES is configured as the data ISERDES and the other as the monitor ISERDES. Both ISERDES modules are configured in master mode. The IBUFDS_DIFF_OUT primitive connects the input data stream to both ISERDES modules. The output of the IBUFDS_DIFF_OUT is differential. The negative output is connected to the input of the monitor ISERDES. The state machines in the fabric implement the bit alignment, word alignment, and window monitoring algorithm and are time-shared across all 16 channels to conserve FPGA resources.

Bit-Alignment Overview

The goal of the bit-alignment procedure is to position the captured clock edge in the center of the data eye to provide maximum margin. The bit-alignment procedure uses the tap delay line feature of the ISERDES. The algorithm delays the data channel with respect to the clock. At the start of the bit-alignment procedure, the control module increments the tap delay line until it finds the left edge of the data eye. When the left edge of the data eye is detected, it activates a counter and continues incrementing the tap delay line, keeping count of the number of taps incremented, until it detects the right edge. At this point, the control module decrements the tap delay line by half of the counter amount, positioning the clock edge at the center of the data eye. Figure 5-2 shows the state transition diagram for the bit-alignment algorithm.

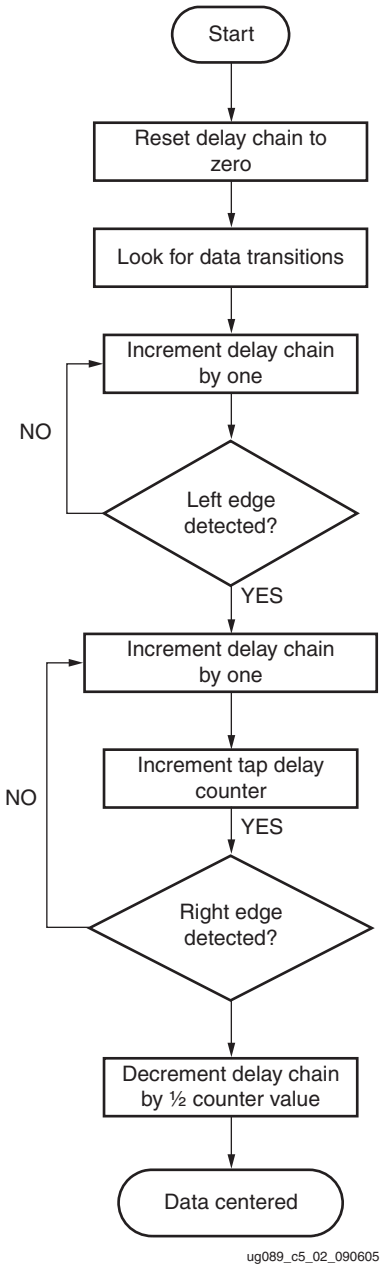


Figure 5-2: State Transition Diagram for the Bit-Alignment Algorithm

With only one bit-alignment state machine time-shared across all 16 data channels, the bit-alignment procedure is circular, starting with Channel 0 and ending with Channel 15. A 16-bit register, called chan_sel, is used as a scheduler to monitor the enabled channel.

For a more detailed description of the bit-alignment process, please refer to *Bit Alignment* in XAPP 700.

Word Alignment Overview

The word-alignment procedure aligns the output pattern from the ISERDES to a specific training pattern. This procedure effectively removes word skew and aligns all channels to a specific word boundary. The word alignment unit primarily uses the Bitslip submodule of the ISERDES. The Bitslip submodule matches the output of the serial-to-parallel converter to a specific pattern. It accomplishes this by effectively shifting the output one bit at a time until the pattern is found. The Bitslip submodule is activated by asserting the BITSLIP port of the ISERDES for one CLKDIV cycle. In DDR mode, the result of a Bitslip operation is guaranteed to be valid only after two CLKDIV cycles. Therefore, the appropriate number of wait states needs to be accounted for in the word-alignment algorithm.

Figure 5-3 shows the state transition diagram for the word-alignment algorithm.

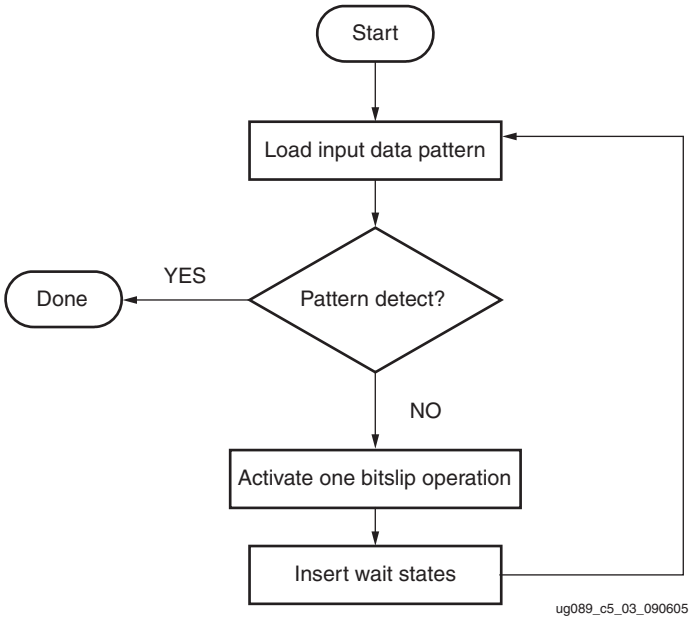


Figure 5-3: State Transition Diagram for the Word-Alignment Algorithm

With only one bit-alignment state machine, time-shared across all 16 data channels, the bit-alignment procedure is circular, starting with Channel 0 and ending with Channel 15. A 16-bit register, called chan_sel, is used as a scheduler to monitor the enabled channel.

For a more detailed description of the word alignment process, please refer to *Word Alignment* in XAPP 700.

Window Monitoring Overview

After the initial training procedure (bit alignment and word alignment) is completed, the channels are deskewed and the sampling clock edge is optimally positioned. However, due to operating conditions (voltage or temperature), the data valid window can shift. The window monitoring unit continuously monitors the data valid window during normal operation and adjusts the sampling point as necessary to provide maximum margin.

The algorithm uses the spare ISERDES module resulting from using differential I/O. The monitor ISERDES is configured in master mode, and the differential in/differential out input buffer (IBUFDS_DIFF_OUT) is used to feed the input serial stream to both ISERDES modules. Due to the differential nature of the buffer outputs, the input serial stream to the monitor ISERDES is inverted. In the window monitoring algorithm, the window monitoring control unit makes the appropriate adjustment when comparing the outputs of the monitor ISERDES with the data ISERDES.

The window monitoring procedure primarily uses the tap delay feature of the ISERDES. The algorithm is set up to delay the data channel with respect to the clock. Figure 5-4 shows the state transition diagram for the bit-alignment algorithm.

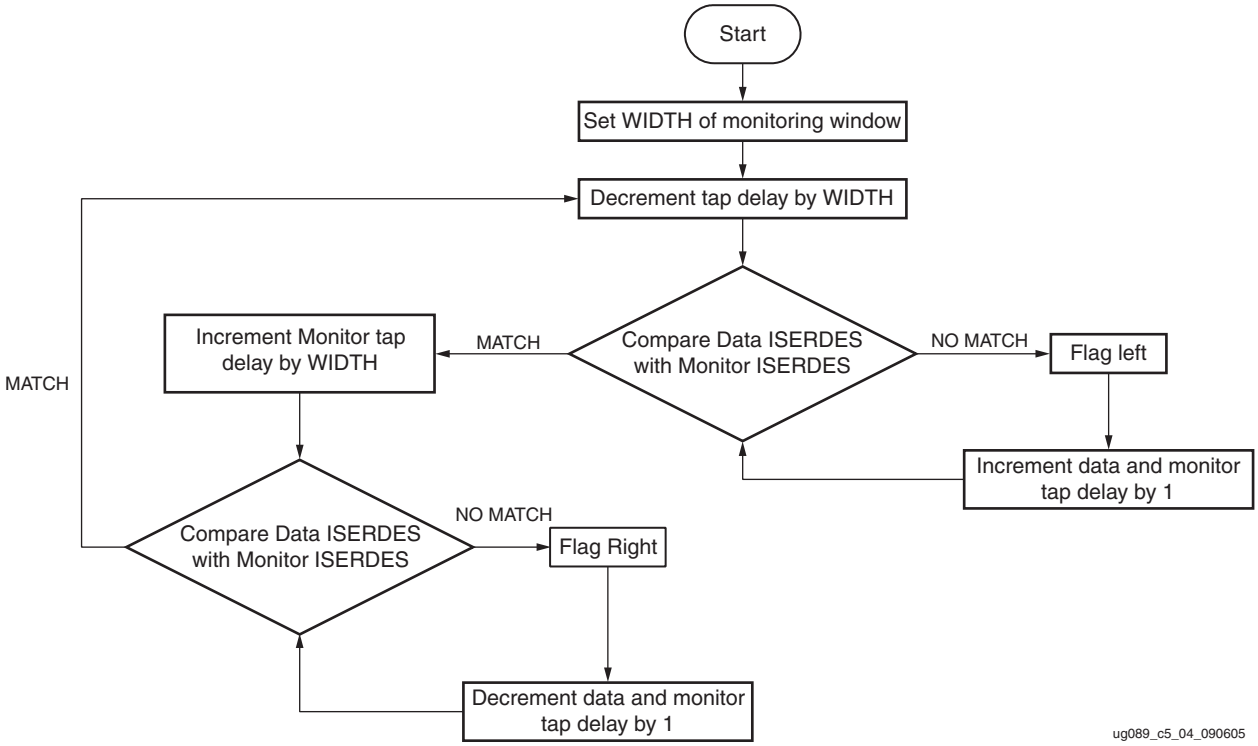


Figure 5-4: The State Transition Diagram for the Bit-Alignment Algorithm

With only one bit-alignment state machine, time-shared across all 16 data channels, the bit-alignment procedure is circular, starting with Channel 0 and ending with Channel 15. A 16-bit register, called chan_sel, is used as a scheduler to monitor the enabled channel.

For a more detailed description of the window monitoring process, please refer to *Window Monitoring* in XAPP 700.

Design Hierarchy

Figure 5-5 shows the hierarchy of HDL source files in the 16-channel DDR BERT design.

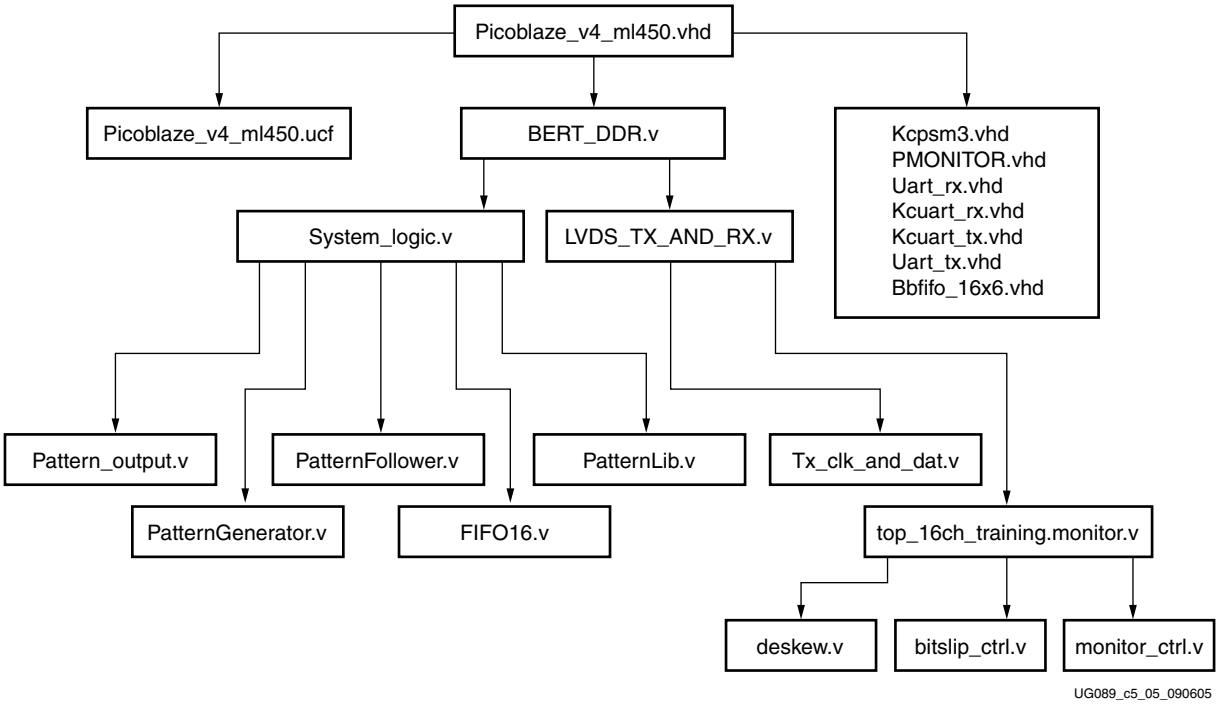


Figure 5-5: **HDL Source File Hierarchy in 16-Channel DDR BERT**

Chapter 6

BERT and GUI

This chapter covers the bit error rate tester (BERT) driving the interface with pseudo-random data, as well as the graphical user interface (GUI) allowing the user to change settings and view statistics. The BERT operates entirely in the system clock domain, which constantly runs at 200 MHz, regardless of the data rate of the interface.

BERT-to-Interface Communication

Communication between the system clock domain and the interface clock domains is achieved through the use of FIFO16 IP. Referring back to Figure 1-1, page 7, the system clock writes data into the Tx FIFO at 200 MHz, and the Tx clock reads data from the FIFO at the TXCLKDIV rate. TXCLKDIV is determined by the serialization ratio of the interface; for example, with a 4:1 serialization for a 700-MHz SDR interface, TXCLKDIV is 175 MHz.

Similarly, on the receive side, the system clock reads data from the Rx FIFO at 200 MHz, and the Rx clock writes data into the FIFO at the RXCLKDIV rate. RXCLKDIV is determined by the serialization ratio of the interface; for example, with a 4:1 serialization for a 700-MHz SDR interface, RXCLKDIV is 175 MHz.

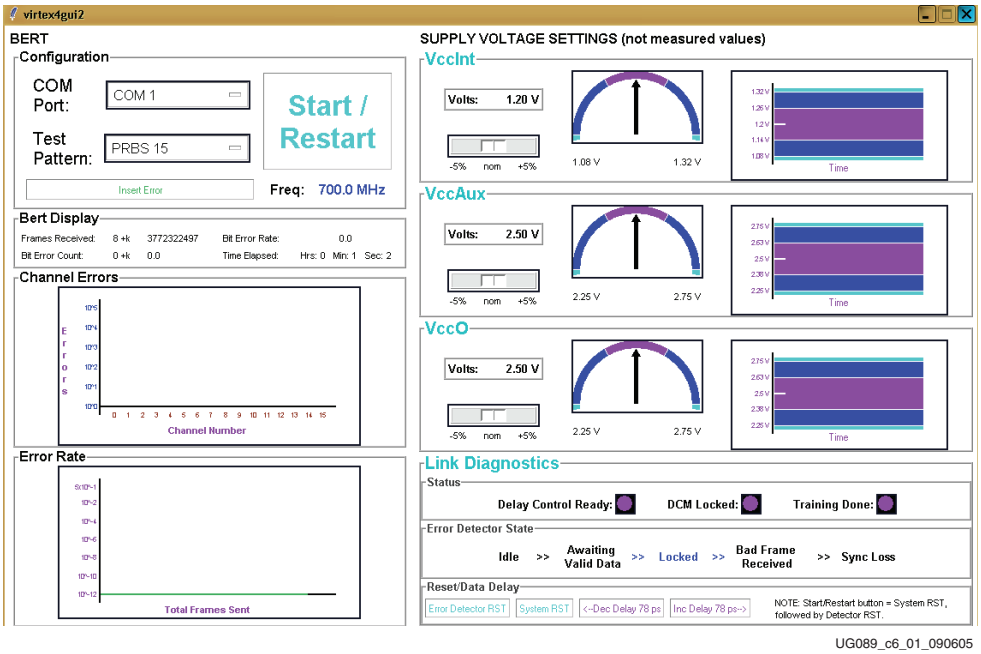
The system clock must be faster than the divided interface clock. If the divided clock of the interface runs at a rate faster than 200 MHz, the design suffers a hard failure. The interfaces transmit and receive data at a constant rate and do not have the flexibility to accommodate almost full or almost empty FIFOs. Instead, the system clock domain provides this flexibility, ceasing to read from the Rx FIFO if the FIFO raises an almost empty flag. Similarly on the transmit side, the system clock ceases to write to the Tx FIFO if the FIFO raises an almost full flag.

For the SPI-4.2 designs presented in Chapter 4 and Chapter 5 , the system clock is driven by a 250-MHz oscillator instead of a 200-MHz oscillator.

GUI Main Window

The PicoBlaze™ processor gathers information from the BERT and sends it to a PC via a null modem serial cable. The raw information is processed by an executable file written in C, and the processed information is displayed using an Active TCL/TK engine.

Figure 6-1 shows the user interface for the ML450 Bit Error Rate Tester.



UG089_c6_01_090605

Figure 6-1: **ML450 User Interface**

Configuration

In the Configuration section of the main GUI window, the user can start the test by pressing Start/Restart, which resets the entire system, including the error detectors. The error detector reset occurs automatically after bus or bit alignment and word alignment are complete.

A variety of patterns can be selected to simulate data across the link:

- Clock patterns to test the link with data of limited frequency content
- Pseudo-random patterns with a much more diverse spectral content

Pseudo-Random Bit Sequence 7 (PRBS) has strings of no greater than 7 ones or zeros in sequence, while PRBS15 has strings of no greater than 15 ones or zeros in sequence. PRBS15 is the more challenging pattern because it contains a larger variation of possible sequences.

The ability to insert errors is provided as a way to check the ability of the receiver to detect errors. When the Insert Error button is pressed, all data on all channels is inverted for one clock cycle, resulting in an entire frame of errors. These errors appear graphically and numerically in the sections of the GUI discussed in "BERT Display"and "Channel Errors." Errors might not be inserted every time the button is pressed. This is because the system clock domain runs faster than the interface domain clock and does not write data to the FIFO on every clock cycle. If *insert error* is asserted during a blank cycle, then no errors reach the interface.

There is a display for the frequency of the serial interface. This frequency is a measured value, reported by a circuit that continuously samples the internal clock. For SDR applications, this value also represents the data rate. For DDR applications, the frequency must be multiplied by two to obtain the data rate.

BERT Display

This section of the GUI shows the following information:

- Frames Received

  Number of data frames received while the receiver is locked to the incoming data
- Bit Error Count

  Sum of all bit errors received on all channels
- Bit Error Rate

  (Bit Error Count) / [(Frames Received) x (Number of Bits per Frame)]
- Time Elapsed

  Time since last Start/Restart

Channel Errors

The channel errors graph shows a histogram for the errors on each of the data channels, providing visibility into the performance of individual data channels. As errors are detected on a given channel, the bar graph grows in proportion to the errors. The Y axis is logarithmic, starting at 1 error and going up to 10,000. When a channel goes beyond 10,000 errors, the graph stops rising for that channel.

Figure 6-2 shows a channel errors graph, where errors are detected on channels 1 and 11.

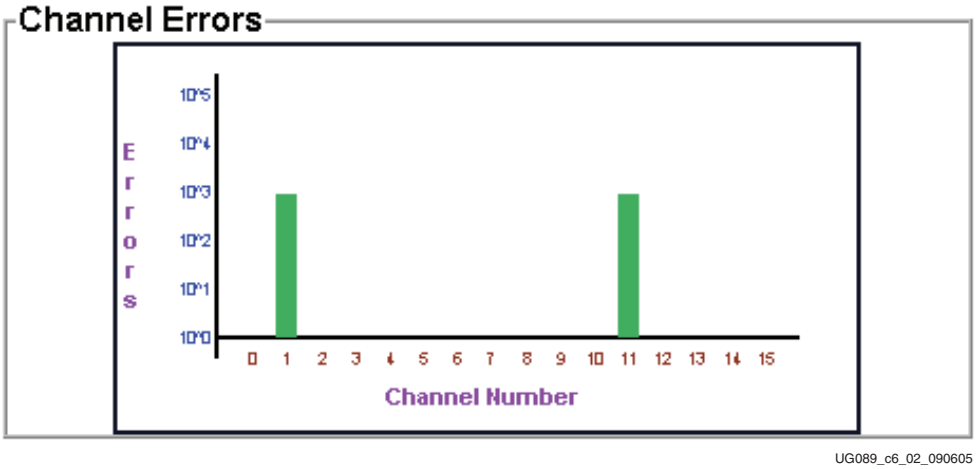


Figure 6-2: **Graph of Channel Errors**

Error Rate

When the interface suffers degradation, either intentionally inflicted by the user or for other reasons, the bit error rate (BER) changes. The ability to see this change as a function of time ensures that such degradation does not go unnoticed. Analog errors in the I/O typically create a constant or flat error rate over time, while errors due to clock failures or timing result in bursts of errors that appear as individual events on the graph.

Figure 6-3 shows an error-free link where the I/Os are deliberately marginalized to create a steady error rate.

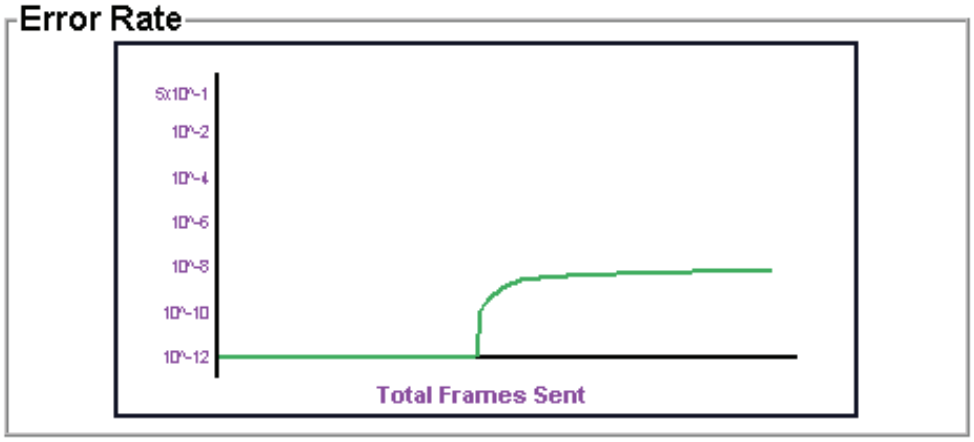


Figure 6-3: **Marginalized Link**

Supply Voltage Settings

The three supplies V_{CCINT}, V_{CCO}, and V_{CCAUX} to the FPGA can be marginalized to ±5% of their nominal value by manipulating the appropriate slider button on the GUI.

Figure 6-4 shows the supply controls for V_{CCINT}. Both V_{CCAUX} and V_{CCO} have the same controls.

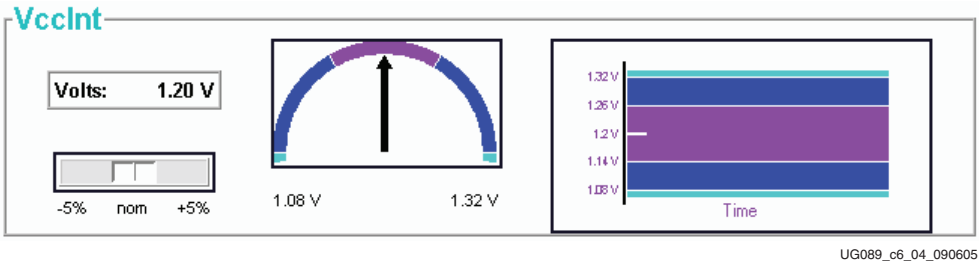


Figure 6-4: **V_{CCINT} Supply Controls**

Immediately after moving the button to ±5%, the actual voltage supplied to the FPGA changes to that voltage. No reset is required. The ability to change the supply voltages is accomplished by analog multiplexers at the outputs of the supply regulators. The multiplexer select lines are driven by controls from the FPGA to select one of the three voltages for each supply (±5% and nominal).

This feature is very useful for demonstrating margin. It is not enough to say that the link runs error-free at a given frequency. Designers want to understand the margin boundaries. Moving the supplies to the maximum and minimum limits in the data sheet is one way to demonstrate margin.

The graphs of each supply are *NOT* measured values. They are only graphical representations of the supply setting.

Link Diagnostics

There are three flags in the Link Diagnostics section, and their states are indicated by a red or a green LED. A green LED indicates the state required for proper operation of the link. After pressing Start/Restart, all LEDs should be green, indicating the following:

- Delay Control Ready
  - Reference clock to the IDLYCTRL module is present and stable.
  - IDLY tap values are nominal (75 ps).
  - Bus alignment in the receiver can begin.
- DCM Locked
  - System clock is present and stable.
  - Locked signal has initiated resets in the entire system.
- Training Done
  - Bus alignment is complete.
  - Word alignment is complete.
  - Data is being transmitted, and error detectors are processing the received data.

Error Detector State

The state of the error detectors is the best way to assess the status of the link. The state provides information about the quality of the received data.

The first state is IDLE, a reset state. Immediately following a reset, the error detectors enter the AWAITING VALID DATA state and begin looking for the same pattern that the transmitter is sending. The detector compares a single frame of data with each incoming data frame from the transmitter. When it sees the matching frame, the receiver pattern synchronizes to the incoming pattern making each subsequent frame also match. This state is known as the LOCKED state. When one or more errors is detected in a frame, the link temporarily enters the BAD FRAME RECEIVED state. If two frames in sequence contain errors, then the detector considers the link to have too many errors to remain locked to the incoming data. This state is known as SYNC LOSS, and the link does not recover from this state until the detector is reset.

Figure 6-5 is the state machine in the error detector.

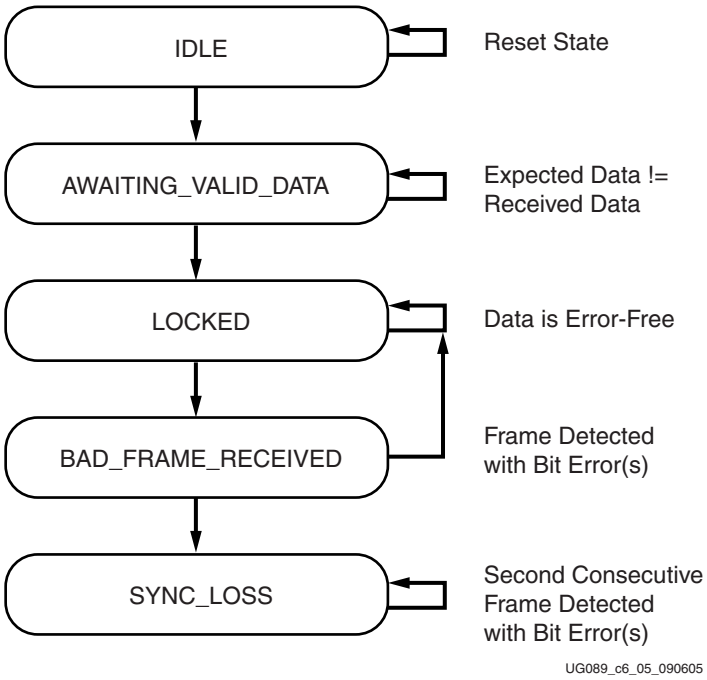


Figure 6-5: **Error Detector State Machine**

Reset/Data Delay

Manually changing the bus-alignment setting determined by the training algorithm is another way to determine the performance margin. By incrementing and decrementing the bus delay, the data eye width can be determined with a resolution of 75 ps (1 tap). Each time the increment or decrement button is pressed, it increments or decrements the delay setting of the data bus by one tap.

To effectively measure the data eye, the following steps must be performed:

1. Set the data pattern to CLK (`1010`).
2. Press the Start/Restart button to confirm that the link is running.
3. Increment the delay, counting each time the button is pressed, until the link begins to show errors. When errors or sync loss are encountered, press the *RST Detector* button. This resets the error detectors without resetting the delay. This verifies that there are errors at the current delay setting. This is the *setup edge* of the data eye.
4. Press the Start/Restart button to reset the delay to the bus or bit-aligned value.
5. Decrement the delay, counting each time the button is pressed, until the link begins to show errors. This is the *hold edge* of the data eye.
6. Add the number of increments and decrements to obtain the width of the eye. The width of the *collective eye* of all 16 channels includes errors due to skew between channels for bus-aligned designs.
7. Change the data pattern to PRBS and repeat steps 1 through 6. The eye becomes smaller and possibly asymmetric.

XILINX®

Chapter 7

Getting Started with the Demonstration

Getting Started in Nine Easy Steps

The following steps are performed to run the demo:

1. Install `ActiveTcl8.4.7.0-win32-ix86-108887.exe`. This file is required for the GUI to run on the PC.
2. Connect the cables from the PC to the ML450 board.
   - Parallel IV
   - Serial (female to female)
3. Connect two LVDS ribbon cables to the SAMTEC connectors.
   - From P49 to P3
   - From P46 to P6
4. Connect the 5V power supply to the board. This supply is included in the ML450 kit.
5. Program the ICS clock module and install it at location CM2. The formula for the switch settings is Frequency = 10 x M/N, where M is the frequency multiplier and N is the frequency divider (as shown in the Description column in Table 7-1). The jumper on the clock module must **NOT** be connected. After the switches for the desired frequency are set, press SW1. SW1 is also pressed after each power-up, even if the frequency is not changed.

   Table 7-1 shows examples of clock module settings for 700 MHz, 630 MHz, and 500 MHz.
6. Power up the ML450 board.
7. Configure the device with the SFI-4 BERT or the SPI 4.2 BERT. The GUI is compatible with both designs.
8. Launch the `GUI_rev3.0.exe` file.
9. Start the bit error rate test by pressing the Start/Restart button on the main window.

Table 7-1: **Clock Module Settings Based on Frequency**

	Description	Setting for 700 MHz	Setting for 630 MHz	Setting for 500 MHz
DIP 2-4	D/C	OFF	OFF	OFF
DIP 2-3	N (MSB)	OFF	OFF	OFF
DIP 2-2	N (LSB)	OFF	OFF	OFF
DIP 2-1	M (MSB)	OFF	OFF	OFF
DIP 1-8	M	OFF	OFF	OFF
DIP 1-7	M	ON	OFF	OFF
DIP 1-6	M	OFF	ON	ON
DIP 1-5	M	OFF	ON	ON
DIP 1-4	M	OFF	ON	OFF
DIP 1-3	M	ON	ON	OFF
DIP 1-2	M	ON	ON	ON
DIP 1-1	M (LSB)	OFF	ON	OFF

Table 7-2 shows the divisor values for the four settings of N.

Table 7-2: **Divisor Values for N**

N (Binary)	Divisor
00	1
01	2
10	4
11	8