

### Introduction

### Preliminary Information

High-speed serial data transmission allows designers to transmit high-bandwidth data using differential, low-voltage swing signaling. One serial channel can support the same bandwidth as multiple conventional single-ended I/O standards, reducing the number of board traces and I/O pins. However, board traces, connectors, and backplane connections still introduce skew between multiple serial channels and between clock and data channels. To guarantee successful data transfer, tight specifications exist for the channel-to-channel skew and clock-to-channel skew. Clock data recovery (CDR) eliminates the need for these skew specifications by encoding the clock into every serial data stream. Because the clock is encoded into the data streams, there is no need for a specified relationship between the clock and data channels. By eliminating the skew specifications, CDR also reduces topology restrictions, and enables multi-crystal operation.

A key feature of Mercury™ devices, CDR is implemented in the high-speed differential interface (HSDI), which is an embedded circuit dedicated to receiving and transmitting high speed serial data streams between the Mercury device and other devices on a circuit board or across a backplane. The HSDI circuitry has two modes: source-synchronous mode and CDR mode. This application note discusses the CDR mode of the HSDI block.



For a description of the Mercury device family, see the *Mercury Programmable Logic Device Family Data Sheet*.

Mercury device CDR circuitry includes the following features:

- Supports independent channels without a fixed phase relationship
- Supports data rates from 125 megabits per second (Mbps) to 1.25 gigabits per second (Gbps)
- Supports 3.3-V LVDS, LVPECL, LVTTTL, and PCML I/O standards
- Supports variable data rates and encoding schemes including: Gigabit Ethernet, SONET/SDH, Fibre Channel, and IEEE Std. 1394
- Supports two independent frequencies
- Supports multi-crystal operation
- Provides a run length detection circuit with 6-bit programmable run length

## CDR Architecture

The HSDI receiver and transmitter channels are located at the middle of the top I/O band. Table 1 shows the number of channels available in Mercury devices.

<b>Table 1. Number of HSDI Channels in Mercury Devices</b>	
<b>Device</b>	<b>Number of HSDI Channels</b>
EP1M120	8
EP1M350	18

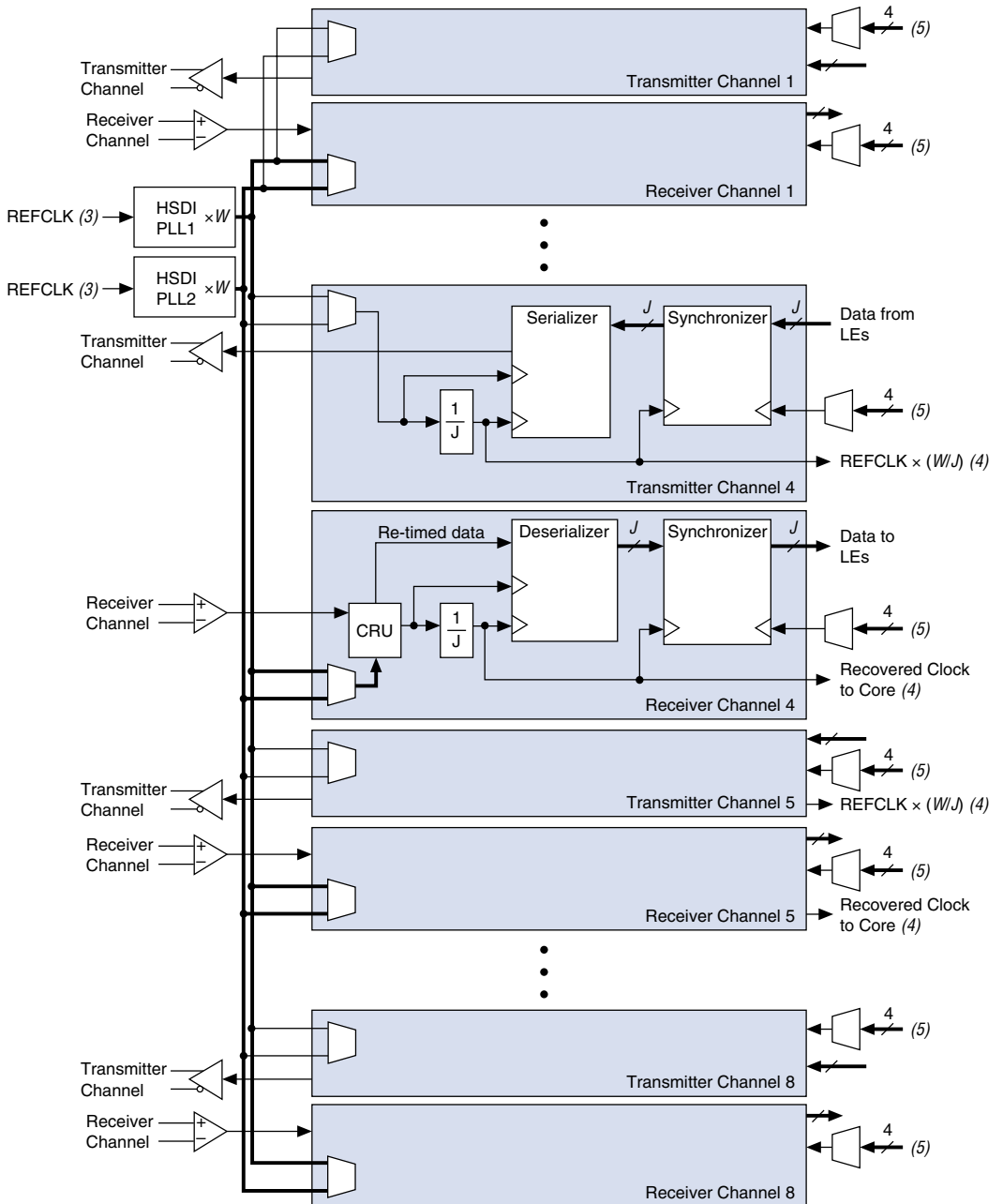
The HSDI CDR mode supports serial data rates of 125 Mbps to 1.25 Gbps. An external reference clock must be fed to one of the two dedicated HSDI PLLs. There are no phase-relationship requirements between the reference clock and the received data. The PLL multiplies the reference clock by a factor  $W$ . For example, if CDR is used to recover the clock from a 1.25 Gbps data stream, a 62.5-MHz signal can be provided as the reference clock. The multiplication factor  $W$  would then be set to 20 so that the resulting multiplied reference clock is 1.25 GHz.

On each HSDI receiver channel, the multiplied reference clock is used by a clock recovery unit (CRU) to generate a recovered clock in-phase with the received data. That recovered clock drives the deserializer and synchronizer. The synchronizer ensures proper register-to-register transfer from the recovered clock domain in the receiver to a global clock domain in the core.

## Receiver & Transmitter Channels

Figure 1 shows a block diagram of the HSDI receiver and transmitter channels. Each receiver channel has dedicated circuitry consisting of a CRU, deserializer, and synchronizer. The deserializer and synchronizer circuitry in the receiver channels are not part of the programmable core logic. The CRU generates the recovered clock used to deserialize and synchronize the data.

Figure 1. HSDI Circuitry Block Diagram Notes (1), (2)



**Notes:**

- (1) EP1M350 devices have 18 individual receiver and transmitter channels. EP1M120 devices have eight individual receiver and transmitter channels.
- (2)  $W = 1$  to 12, 14, 16, 18, or 20  
 $J = 3$  to 12, 14, 16, 18, or 20  
 $W$  does not have to equal  $J$ .
- (3) REFCLK sources are listed in Table 2.
- (4) Recovered clocks from receiver channels and  $\text{REFCLK} \times (W/J)$  from transmitter channels can be driven to the global clock lines from channels 4 and/or 5 on EP1M120 devices, and from channels 9 and/or 10 on EP1M350 devices.
- (5) This multiplexer is driven by one of four global clocks.

The transmitter channel has a dedicated synchronizer and serializer. Like the receiver channels, the transmitter channel circuitry is not part of the core logic. The reference clock is used to synchronize and serialize the data to be transmitted.

Mercury devices support up to two different serial data rates for the receiver and transmitter channels. Each channel can independently select either data rate. Received data must be non-return-to-zero (NRZ).

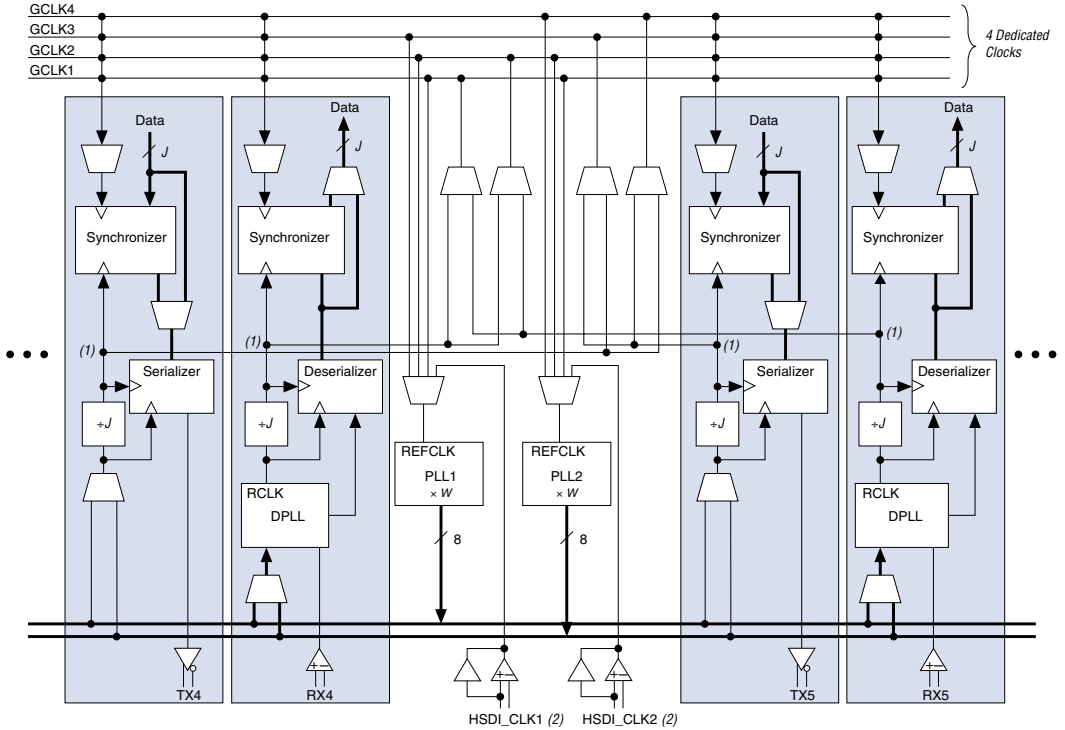
## Data & Clock Paths

The receiver channels handle clock data recovery, data deserialization, and synchronization. In order to maintain support for various encoding schemes, protocols, and standards, succeeding functions such as byte alignment, comma detection, frame detection, and byte decoding are implemented in core logic. Reference designs and IP blocks to address these needs are available from Altera.

Similarly, transmitter channels handle data synchronization and serialization. Core logic must perform preceding jobs such as byte encoding.

Figure 2 illustrates the data and clock paths in the HSDI circuitry, while Tables 2 and 3 list possible sources for the global and HSDI clocks.

Figure 2. HSDI Clock & Data Paths



Notes:

- (1) The two center channels adjacent to the HSDI PLLs (channels 4 and 5 for EP1M120 devices, channels 9 and 10 for EP1M350 devices) can drive the Mercury device's global clocks.
- (2) HSDI\_CLK1 and HSDI\_CLK2 pins can be either 3.3-V LVTTTL or differential. These clock pins drive HSDI PLLs only. They do not drive to the core.

Table 2. HSDI PLL REFCLK Sources

Sources	Destinations	
	HSDI PLL1	HSDI PLL2
GCLK1	✓	✓
GCLK2	✓	✓
GCLK3	✓	
GCLK4		✓
HSDI_CLK1	✓	
HSDI_CLK2		✓

**Table 3. HSDI Signals Drive Global Clocks**

Sources	Destinations			
	GCLK1	GCLK2	GCLK3	GCLK4
RXJCLK (1)	✓ (3)	✓ (3)		
TXJCLK (2)			✓ (4)	✓ (4)

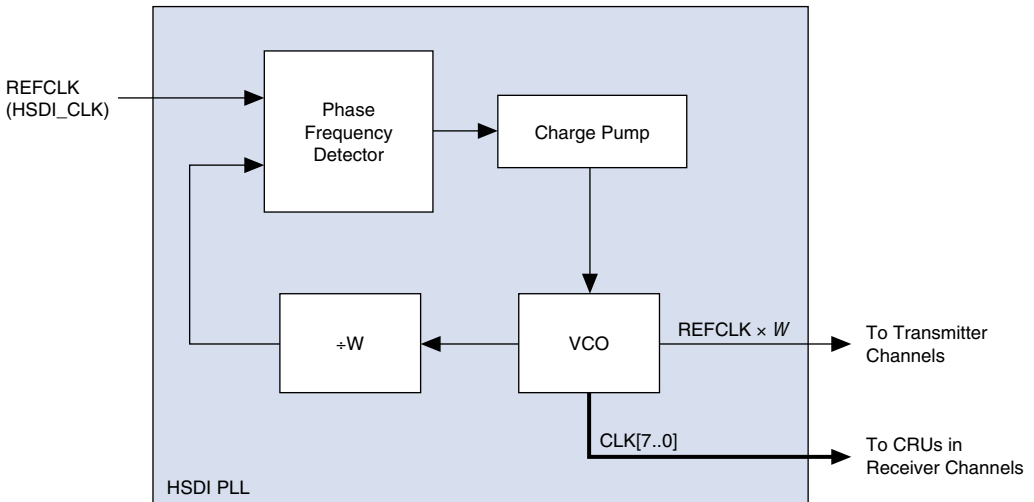
**Notes:**

- (1) RXJCLK is the output of the *J* divider in a receiver channel (RXJCLK = recovered clock/*J*).
- (2) TXJCLK is the output of the *J* divider in a transmitter channel (TXJCLK = REFCLK × *W*/*J*).
- (3) The RXJCLK signals of receiver channels 4 and 5 (EP1M120 devices) or receiver channels 9 and 10 (EP1M350 devices) can drive GCLK1 and GCLK2.
- (4) The TXJCLK signals of transmitter channels 4 and 5 (EP1M120 devices) or transmitter channels 9 and 10 (EP1M350 devices) can drive GCLK3 and GCLK4.

## HSDI PLLs

Two dedicated HSDI PLLs (separate from the general-purpose PLLs) are used for serializing and deserializing differential high-speed channels. Figure 3 shows a diagram of a HSDI PLL. Table 2 on page 5 lists the sources available to the REFCLK signal.

**Figure 3. HSDI PLL Block Diagram**



At each rising edge of the reference clock, the phase/frequency detector of the PLL determines the phase difference between the reference clock and the voltage controlled oscillator (VCO) clock (the VCO clock divided by  $W$ ). The high-frequency changes in the phase difference are filtered in the charge pump, and the resulting control voltage drives the VCO. By taking outputs from the VCO, the PLL generates eight clocks with the same frequency as the serial input data, which are output to the CRU. Each of these eight clocks has a  $1/8$  period phase shift from the next one.

Table 4 contains the specifications for the HSDI PLLs in CDR mode.

Parameter	Minimum	Maximum	Unit
Reference clock frequency (LVDS, LVPECL, or PCML)	25	650	MHz
Reference clock frequency (3.3-V LVTTTL)	25	200	MHz
VCO frequency	125	1,250	MHz
Input clock jitter		0.025	UI
Reference clock duty cycle	45	55	% period
Lock time		10	ms
Multiplication factor ( $W$ )	1 to 12, 14, 16, 18, 20		Integer

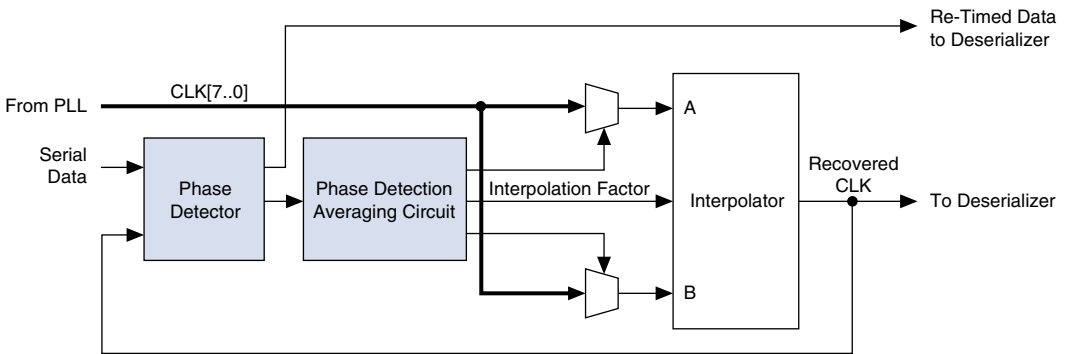
## CRUs

The 8-phase clocks from the PLL and the serial input data feed the CRU, which generates a recovered clock in-phase with the serial input data. Because phase information can only be extracted through a 1-to-0 or a 0-to-1 transition, there is a run length limitation for the serial input data. Run length refers to the number of consecutive ones or zeros in the serial data. During times of no transitions on the serial data (i.e., when the input data stays at 1 or 0), the CRU locks to the last recovered clock. This allows Mercury devices to meet the maximum run lengths required by the protocols listed in Table 6.

An optional run-length detector circuit compares the run length of the serial data against the 6-bit run length violation (RLV) value. The RLV value can be set to any integer from 1 to 62. A zero setting allows an infinite run length. The RLV flag is set when the run length exceeds the preset RLV value. The flag can be driven into core logic or driven out through a regular I/O pin.

Figure 4 shows the CRU block diagram. On each data transition, the phase detector decides if the current recovered clock is early or late. The phase detection averaging circuit samples and averages the phase detector decisions. The averaging circuit drives two multiplexers to select the two clock phases that are closest to the ideally recovered clock. The interpolator uses the interpolation factor from the averaging circuit to generate a clock that is fractionally between these two clocks. Each of the eight equally-spaced phase clocks is divided into seven fractions; therefore, the resulting best-case clock granularity is 1/56 of the clock period.

Figure 4. CRU Block Diagram



CRU specifications are shown in Table 5.

Parameter	Min	Max	Unit
Serial input frequency	125	1250	Mbps
Serial input data p-p jitter		0.73	UI
Data bits required to lock when receiving repeated 010101 pattern	320		Integer
Parallel factor ( <i>J</i> )	3 to 12, 14, 16, 18, 20		Integer

After the CRU initially locks on power up, an optional `rx_lock_lost` signal can be used to verify that the CRU is still locked to the data. The `rx_lock_lost` signal is asserted when a preset number of errors has been counted within 64 parallel clock cycles (64 cycles of recovered clock/*J*). This signal cannot be used to flag the initial lock of the CRU.

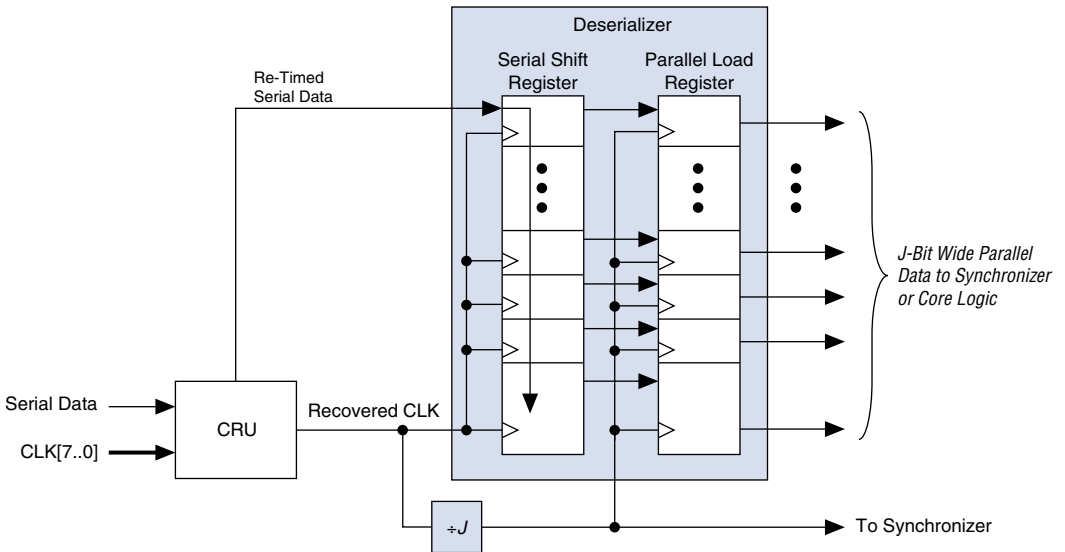
An error occurs when a data transition falls inside the sample error window. The sample error window width is chosen automatically based on the input data rate set by the user in the Quartus™ II software. The user sets the error count value.

The error counter is reset every 64 parallel clock cycles. Therefore, the `rx_lock_lost` signal is asserted if the preset error count is detected within 64 clock cycles. After the 64th clock cycle, the `rx_lock_lost` signal is deasserted and the running error count is reset to 0.

## Deserializer & Serializer

The deserializer generates  $J$ -bit wide parallel data from the serial data, as shown in Figure 5. The value of the parallel bus width ( $J$ ) can be programmed as 3 to 12, 14, 16, 18, or 20;  $J$  and  $W$  values (the reference clock multiplication factor) can be set independently. See Table 4 for  $W$  values.

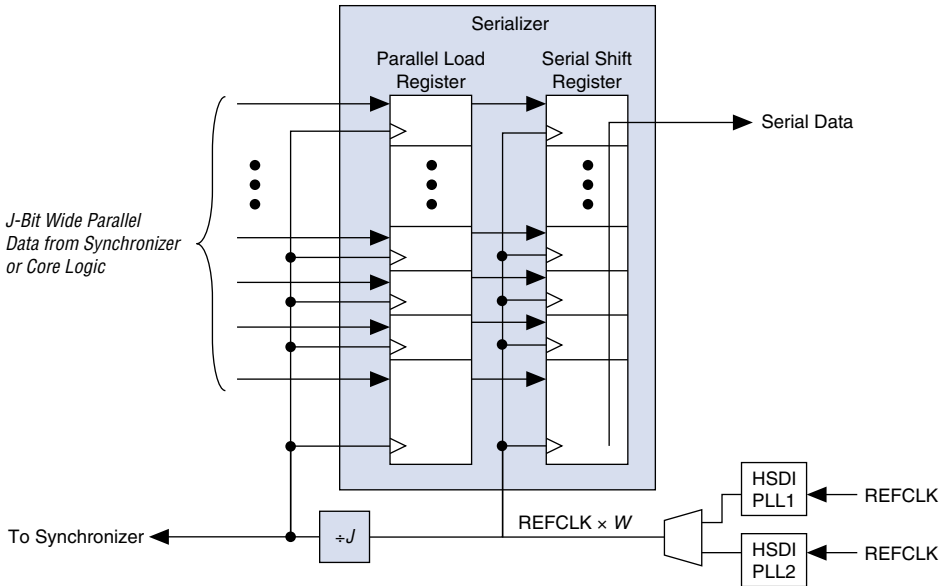
Figure 5. Deserializer Block Diagram



The deserializer supports operations up to 20 bits wide. One advantage to this 20-bit width is that two words (each up to 10 bits wide) can be brought in at the same time. This allows the core frequency to run half as fast as if only one word was deserialized. The parallel load registers' outputs can drive to the synchronizer or directly to the core.

The serializer (shown in Figure 6) supports serializing parallel data of up to 20 bits wide. The parallel bus width ( $J$ ) can be set to 3 to 12, 14, 16, 18, or 20. The parallel data is written to the serializer on every transmit parallel clock cycle (reference clock  $\times W/J$ ).

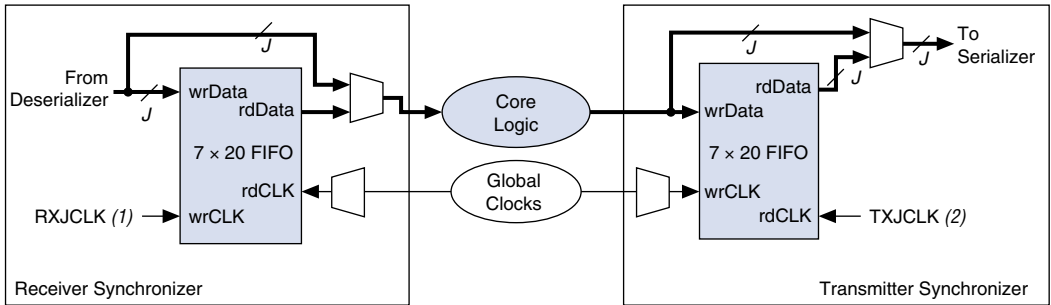
**Figure 6. Serializer Block Diagram**



## Synchronizer

Each receiver has its own synchronizer, as shown in Figure 1 on page 3. Together, the receiver and synchronizer match the frequencies of the parallel data from all the receivers to one global clock, even though each receiver has a recovered clock with different phase. The synchronizer is a 7-word  $\times$  20-bit first-in first-out (FIFO), which can be bypassed to drive data directly to the core, as shown in Figure 7.

Figure 7. Synchronizers Interface Between the Core & the HSDI Channels



Notes:

- (1) RXJCLK is the parallel receive clock (the recovered clock divided by  $J$ ).
- (2) TXJCLK is the parallel transmit clock ( $REFCLK \times (W/J)$ )

When bypassing the synchronizer, the designer must use internal logic to perform core-based synchronization. To ensure proper register-to-register transfer from the recovered clock domain in the receiver to a global clock domain in the core, Altera recommends using the dedicated FIFO buffers for synchronization.

Each transmitter also has its own synchronizer, as shown in Figure 1 on page 3. The synchronizer guarantees successful parallel data transfer from the core into the HSDI transmitter, independent of the phase difference between the global clock and the parallel transmit clock ( $REFCLK \times W/J$ ). The transmit synchronizer can also be bypassed so that the parallel data from the core is directly written to the serializer.

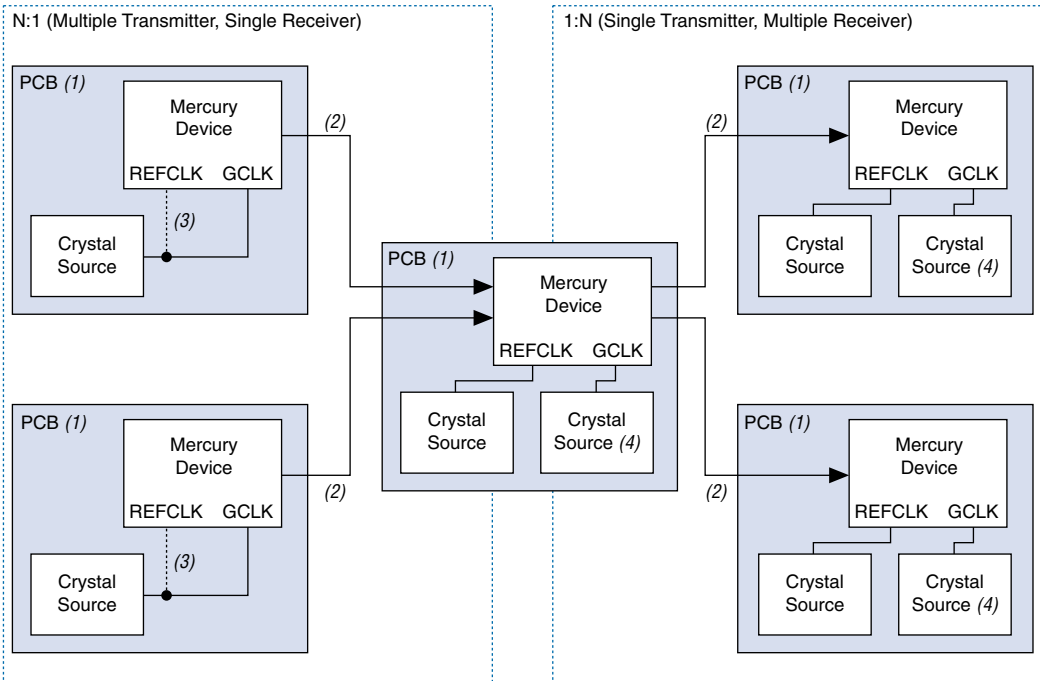
The synchronizers in the receiver channels each have an empty flag and a full flag that is sent to the core. When the synchronizer asserts the empty flag, the FIFO buffer is empty, and no data is available. When the synchronizer asserts the full flag, the buffer is full. If data continues to be transmitted after the buffer is full, some incoming data bits may be lost due to FIFO overflow, and some data packets may be corrupted.

The synchronizers in the transmitter channels also send empty and full flags to the core. When the synchronizer asserts a full flag, the FIFO buffer is full and cannot accept any more data. When the empty flag is asserted, the transmitter is shifting out unnecessary data bits due to underflow.

## Frequency/ Crystal Combinations

Mercury devices support CDR in multi-crystal and single-crystal operation. The data rate of all serial input channels must be within  $\pm 100$  parts per million (PPM) of the frequency of the multiplied reference clock. In multi-crystal operation, the HSDI PLL reference clock, global clock and serial input data are not all driven off the same crystal source. Mercury devices can be used in N:1 or 1:N configurations, where the N points do not require a common crystal source. A combination of an N:1 and a 1:N configuration is shown in Figure 8.

**Figure 8. Single-Frequency, Multi-Crystal Configurations**



### Notes:

- (1) PCB: printed circuit board.
- (2) Serial data channels do not need to be in phase, but the  $\pm 100$ -PPM specification must be met between the serial input data rate and the frequency of the multiplied reference clock at every device.
- (3) The REFCLK signal may be internally driven from the GCLK signal.
- (4) The global clock frequency must be higher than the receiver and transmitter's parallel clock frequencies ( $\text{REFCLK} \times W/J$ ). GCLK frequency  $>$   $\text{REFCLK} \times (W/J)$ . The global clock can also be generated by feeding the REFCLK source to a general PLL.

CDR does not eliminate all frequency restrictions. In the case of multi-crystal operation, a  $\pm 100$ -PPM specification must be met between the serial input data rate and the frequency of the multiplied reference clock. Also, the global clock frequency must be higher than the receiver and transmitter parallel clock ( $REFCLK \times W/J$ ) frequencies.



In order to meet the  $\pm 100$ -PPM specification between the serial input data rate and the frequency of the multiplied reference clock, Altera recommends that all  $REFCLK$  crystal sources meet a specification of  $\pm 50$  PPM or better.

In the case of single crystal operation, the frequency of the global clock source must be equal to or higher than the receiver parallel clock frequencies.

## Applications

Mercury devices support variable data rates and encoding schemes through the HSDI circuitry. These applications are listed in Table 6.

**Table 6. Common Applications of CDR in Mercury Devices**

Application	Data Rate (Mbps)	Multiplication Factor ( $W$ )		Reference Clock Frequency (MHz)		Parallel Bus Width ( $J$ )		Global Clock Frequency (MHz)
		Min	Max	Min	Max	Min	Max	Min (1)
Mercury specifications	125 to 1,250	1 to 12, 14, 16, 18, 20		25	650	3 to 12, 14, 16, 18, 20		Data rate/ $J$
Gigabit Ethernet	1,250	2	20	62.5	625	10	20	1,250/ $J$
SONET OC-12	622.08	1	20	31.104	622.08	4	20	622.08/ $J$
SONET OC-3	155.52	1	6	25.92	155.52	4	20	155.52/ $J$
IEEE Std. 1394	1,200	2	20	60	600	8	20	1,200/ $J$
	800	2	20	40	400	5	20	800/ $J$
Fibre Channel	1,062	2	20	53.1	531	7	20	1,062/ $J$
HDTV	742.5	2	20	37.125	371.25	5	20	742.5/ $J$
SDTV	360	1	14	25.71	360	3	20	360/ $J$
	270	1	10	27	270	3	20	270/ $J$
Fast Ethernet	125	1	5	25	125	3	20	125/ $J$

**Note:**

(1) For multi-crystal applications, the global clock frequency must be greater than the value shown.

Mercury devices provide support for 3.3-V LVDS, LVPECL, 3.3-V PCML, and single-ended I/O standards.

In addition to the protocols listed in Table 6, the Mercury device HSDI circuitry may be used on boards designed for packet over SONET (POS-PHY), or RapidIO architecture. For these architectures, the design uses the source-synchronous HSDI mode.

To improve time-to-market, Altera offers intellectual property (IP) megafunctions for operations such as encoding/decoding, comma detect, and byte-alignment. For more information, refer to the IP MegaStore™ web site at <http://www.altera.com>.

## Summary

CDR eliminates the restriction of phase-matching the serial data and the reference clock at the receiver channels. As a result, CDR eliminates tight board routing and topology restrictions. CDR is also required by many applications. By using a Mercury device, a designer can integrate discrete CDR devices into the programmable logic device (PLD), saving board space and improving performance.

## References

- SONET Standard GR-253 (Bellcore), G.958 (ITU), G.825 (ITU)
- Gigabit Ethernet Standard 802.3Z



*Notes:*



101 Innovation Drive  
San Jose, CA 95134  
(408) 544-7000  
<http://www.altera.com>  
Applications Hotline:  
(800) 800-EPLD  
Customer Marketing:  
(408) 544-7104  
Literature Services:  
[lit\\_req@altera.com](mailto:lit_req@altera.com)

Altera, MegaStore, Mercury, Quartus, Quartus II, and specific device designations are trademarks and/or service marks of Altera Corporation in the United States and other countries. Altera acknowledges the trademarks of other organizations for their respective products or services mentioned in this document. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

Copyright © 2001 Altera Corporation. All rights reserved.



I.S. EN ISO 9001