

## Introduction

As demand for bandwidth increases in this information-based society, communications systems with advanced technologies are emerging to meet such demand. Embedding clocks into serial data streams is a popular technique in high-speed data communications systems applications. The embedded clock is recovered at the receiver by a Clock Data Recovery (CDR) circuit. Source-Synchronous mode provides another way of achieving a high speed data rate without embedding the clock. Low voltage differential signaling can be adopted to improve signal integrity while simultaneously reducing power and emissions. Common differential standards in use include LVDS, BLVDS and LVPECL. For more information on these standards please refer to Lattice technical note TN1000, *sysIO™ Usage Guidelines for Lattice Devices*.

Lattice provides sysHSI™ blocks on certain devices to support both embedded clock and source-synchronous clocking applications. This capability is combined with sysIO interfaces that provide support for a variety of standards including LVDS and BLVDS. The body of this document describes common aspects of the sysHSI block. The appendices detail device specific information.

## Modes of Operation

The sysHSI block can be operated in a number of modes. In Clock Data Recovery (CDR) mode, clock is encoded in the data stream and CDR recovers this clock from the incoming data. In Source-Synchronous mode, the clock is transmitted along with data via a separate channel.

Three fuse programmable modes and their related system specifications are summarized in Table 1.

**Table 1. Fuse Programmable Modes**

Mode	Data Code	Serial Data Rate (Mbps)	Pay Load Data Rate (Mbps)	Parallel Data/Clk (MHz)	Parallel Data Width	Serial/Parallel Ratio	Symbol Alignment Pattern	CDR Support
SERDES without Encoding/Decoding	8B/10B	400 to 850	320 to 680	40 to 85	10b Encoded	10	K28.5 +/-	CDR
SERDES with Encoding/Decoding	10B/12B	400 to 850	333 to 708	33.3 to 70.8	10b Raw Data	12	SymPat	CDR
Source-Synchronous (n channels)	N/A	400 to 800	n x (400 to 800)	50 to 100 67 to 133 100 to 200	n x 8b n x 6b n x 4b	8 6 4	SymPat <sup>1</sup> (De-skew)	De-skew (optional)

1. In Source-Synchronous mode, only De-skew mode requires symbol alignment. See Normal Mode section for detail.

## CDR Modes

As noted previously, some users embed clocks in serial data streams to achieve higher data transfer rates. This is achieved by encoding the transmitted data in such a way as to ensure a minimum number of clock transitions. From this minimum number of transitions a complete clock can be recovered at the receiver. This encoding scheme eliminates the need for a separate clock channel and assures that the clock and data are in phase. Thus, the CDR mode enables higher bandwidth at lower cost.

The sysHSI block supports two encoding options. In both options the sysHSI block recovers data using 16 times over-sampling. This leads to better performance than many other solutions that use lower over-sampling rates.

### SERDES without Encoding/Decoding

This mode supports serial links that use the common 8B/10B encoding scheme. With this scheme, eight bits of data are encoded into 10 bit symbols to ensure a minimum of 40% transition in every 10-bit code.

In 8B/10B mode, the sysHSI block does not encode or decode the data. The block receives encoded 8B/10B data as 10-bit wide parallel data and transmits it serially. It receives serial data and converts it to 10-bit wide 8B/10B encoded data. This data can be re-transmitted or decoded elsewhere, dependent on the application needs.

**SERDES with Encoding (10B/12B Encoding and Decoding is done by sysHSI Block)**

This mode supports serial links that use 10B/12B encoding. This high speed serial data format consists of 10 data bits plus two fixed insertion bits (01) to ensure a minimum of two transitions for every 12 bits in the serial data stream.

**Source-Synchronous (SS) Modes**

Some users implement source-synchronous clocking to achieve high speed data transfer. Here the clock is transmitted along with the data. This removes the propagation delay between the transmitter and receiver as a limit on clock speed and performance. Skew control and other factors limit the maximum performance that can be achieved using this method of data transfer.

Multiple sysHSI blocks can be combined to create source-synchronous interfaces of different widths. The maximum width supported is device dependent, see appendices for details. These interfaces can operate in two modes.

**Normal Mode (with Optional Phase Adjustment)**

*Normal Mode without Phase Adjustment:* In normal source-synchronous mode data for each channel is captured using a common phase shifted version of the incoming clock. This mode is useful in smaller devices where clock-tree skew is minimum.

*Normal Mode with Phase Adjustment:* Optional Phase Adjustment is provided in this mode. The known skew adjustment phase value can be programmed by user.

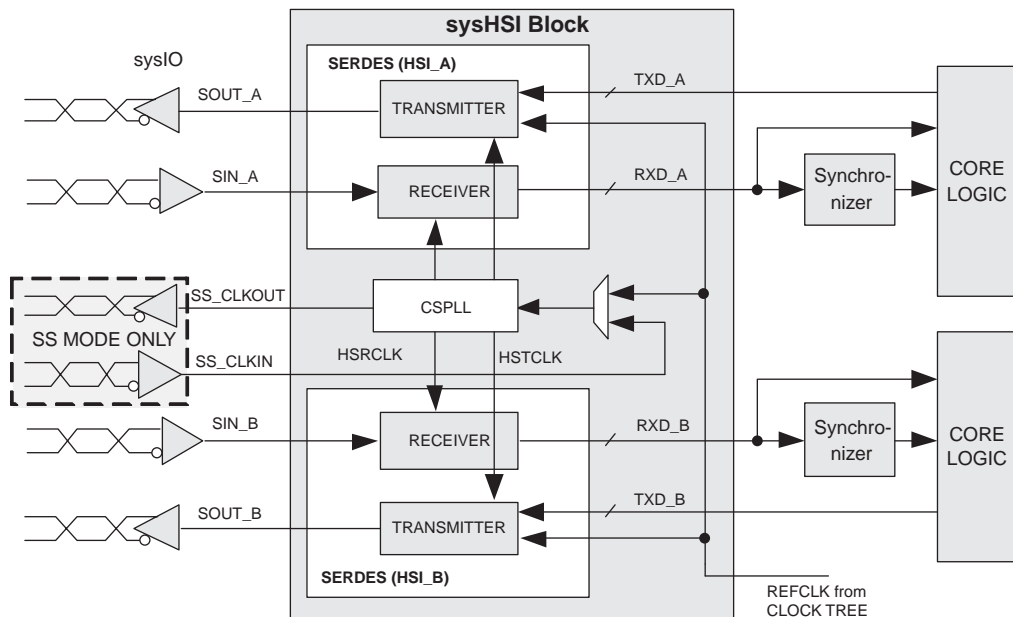
**De-Skew Mode**

In this mode, a calibration cycle allows the CDR circuitry to be used to select per channel different phases of the incoming clock with which to capture the incoming data. This allows the device to compensate for fixed system level skews. Thus allowing designers to achieve higher performance by conducting a calibration cycle at system start up.

**sysHSI Block**

Each sysHSI Block includes two SERDES units and one CSPLL. Each SERDES unit consists of one receiver and one transmitter circuit block. Each pair of receiver and transmitter can be used as a full duplex channel. For receiving, the SERDES receives high speed serial input data stream from the sysIO differential input buffer and provides low speed parallel data associated with recovered clock to synchronizer or core logic. For transmitting, the SERDES converts the parallel low speed data to high speed serial data stream and sends the data to the sysIO LVDS differential output buffers. Figure 1 shows high level representation of a sysHSI Block.

Figure 1. sysHSI Block Diagram

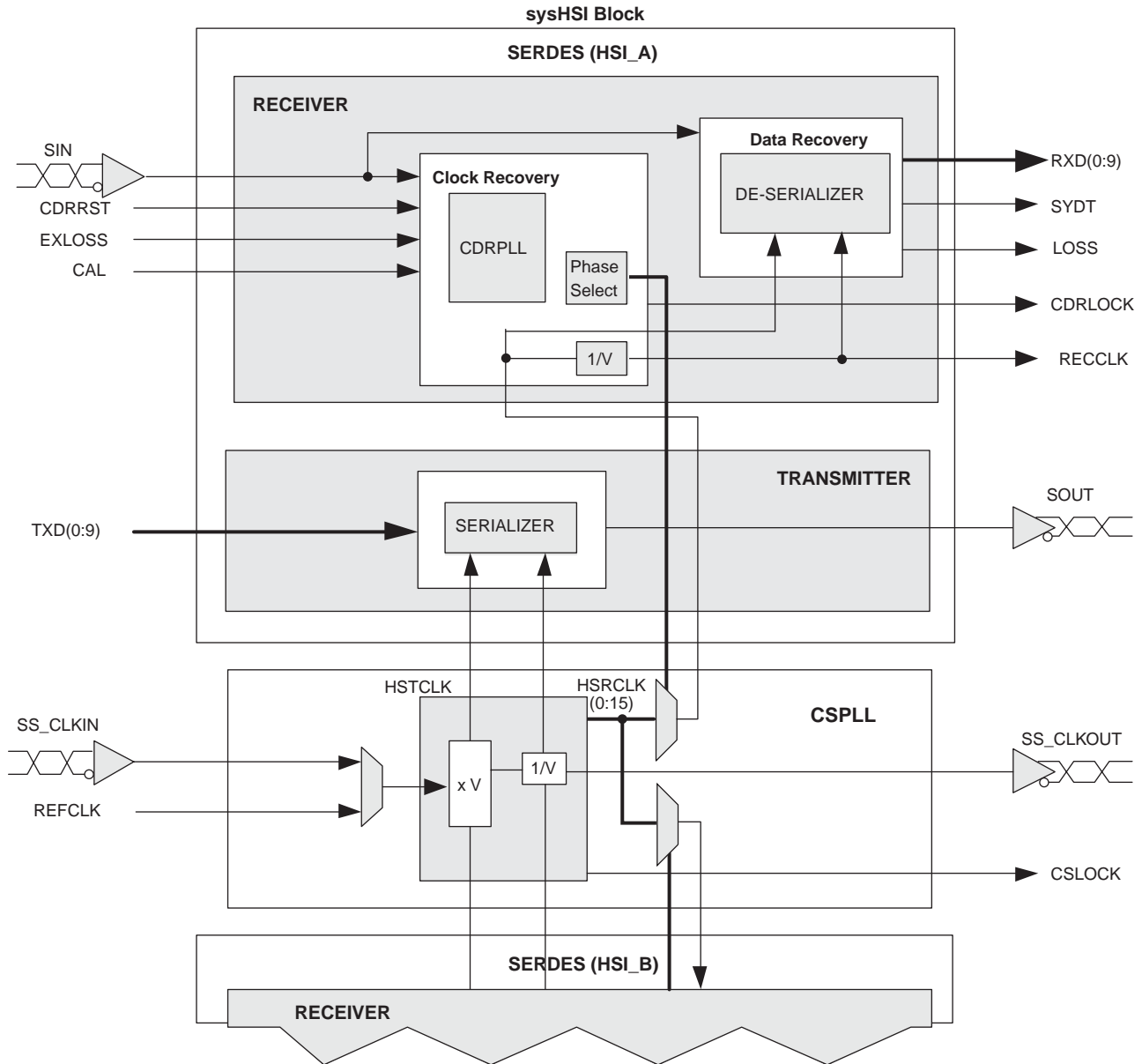


There is always a 10-bit wide data transmitted or received at the low speed side of the SERDES for both 10B/12B and 8B/10B modes. In 10B/12B encoding mode, the start bit(1) and stop bit(0) are added or removed within the sysHSI Block. In SERDES mode without encoding/decoding, (currently 8B/10B mode is supported), the encoding and decoding is done outside of the sysHSI Block where 10-bit wide data is expected at the low speed side of the SERDES. This is why the number of data bits at the parallel interface for 10B/12B and 8B/10B are same. In Source-Synchronous Mode, the low speed parallel data bits can be programmed to 4, 6 or 8.

The recovered clock is asynchronous to the on-chip reference clock. The maximum allowance of frequency deviation is 100 ppm. This interprets that at every 10,000 clock period one clock may be offset from the other by one full clock period. The solution to this problem is to use a synchronizer. In systems where frequency deviation is not a problem, this synchronizer can be bypassed. If desired, the recovered clock and data can be routed to the I/Os and the synchronization can be done off-chip.

The basic functional blocks in the sysHSI Block may be divided by three different functional blocks; Receiver, Transmitter and Clock Synthesizer Phase Locked-Loop. Figure 2 illustrates these functional blocks.

Figure 2. sysHSI Block Detail Diagram



Note: The transmitter and receiver pair above establishes one full duplex channel and is half of a sysHSI Block. The CSPLL is shared by both SERDES.

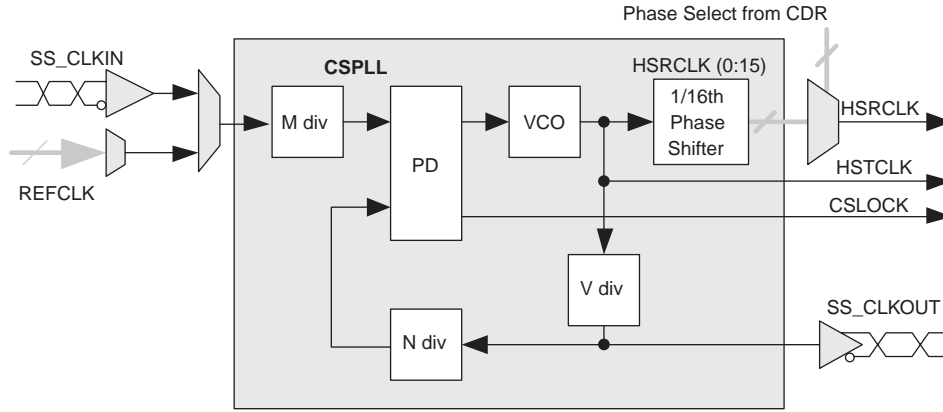
### CSPLL: Clock Synthesizer PLL

CSPLL (Clock Synthesizer PLL) multiplies low speed reference clock by the factor of  $v$  to achieve an high speed serial data rate clock. The low speed reference clock input can be either from a chip internal clock, REFCLK, or from an external differential clock input, SS\_CLKIN. Also, there are four choices for the internal clock to increase the flexibility. User can select one of four clocks available. See Appendices C and D for detail.

The multiplication factor,  $v$ , is the ratio of high speed vs. low speed. It can be 4, 6, or 8 for Source-Synchronous mode, 12 for 10B/12B and 10 for 8B/10B mode. CSPLL contains a fully monolithic analog PLL which does not require any external component. For a transmitter, the HSTCLK (High Speed Transmit Clock) is generated from REFCLK multiplied by a factor of  $v$ , and is used to clock the high speed Serial Data Output.

For CDR operation, the CSPLL combined with a phase interpolation circuit, generates 16-phase high speed Clocks, HSRCLK<0:15>(High Speed Receive Clock). At 1GHz baud rate, these 16 over-sampling phases achieve a resolution of 62.5ps.

Figure 3. CSPLL

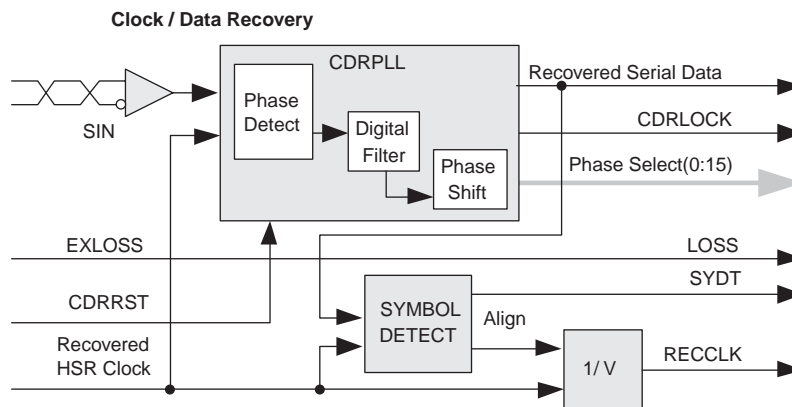


### Clock and Data Recovery

Each receiver channel has its own CDRPLL (Digital Phase-Locked Loop: DPLL) for Clock Data Recovery. The Clock Recovery module first extracts the embedded high speed clock from the Input Serial Data Stream by means of the CDRPLL. Then the Data Recovery Module uses the recovered clock to read the data from the high speed Input Serial Data Stream.

The recovered high speed clock is divided by the factor, v, and aligned to produce the low speed clock, RECCLK (RECovered CLock). CDR then de-serializes the recovered high speed Serial Data into low speed Parallel Data. This RECCLK and parallel data are sent to the synchronizer or core logic.

Figure 4. Clock and Data Recovery Block

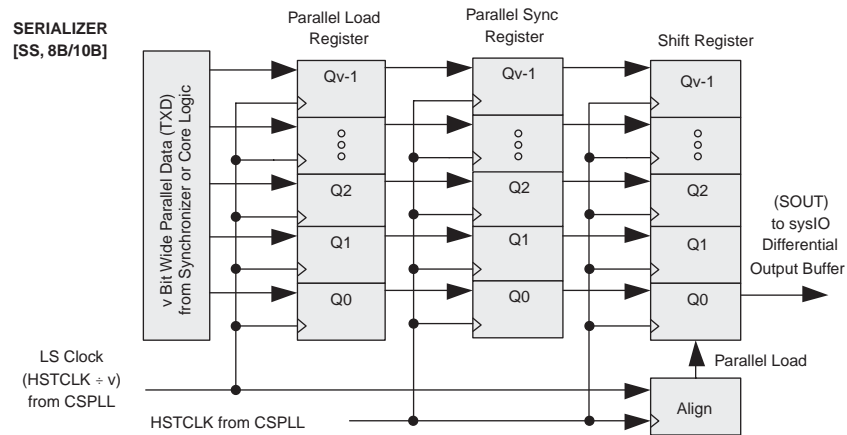


### Serializer/De-Serializer (SER/DES)

#### Serializer

The transmitter receives low speed parallel data, TXD, from the Synchronizer or Core Logic. TXD data is clocked by REFCLK from the clock tree (or SS\_CLKIN in SS mode). The CSPLL multiplies REFCLK by a factor of v to generate HSTCLK. The transmitter converts the low speed parallel Data, TXD, into a high speed Serial Data Stream, SOUT, that is running at HSTCLK.

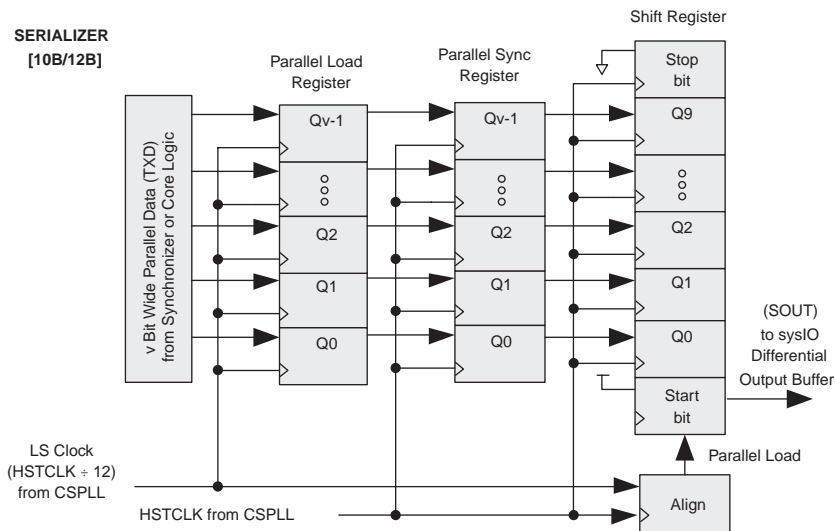
Figure 5. Serializer [SS, 8B/10B]



The alignment circuit synchronizes REFCLK and HSTCLK with the edge detect circuit to align SOUT with HSTCLK.

The 10B/12B Serializer has a built-in encoder circuit. The encoder adds Start bit(1) in front of parallel data and stop bit(0) at the end.

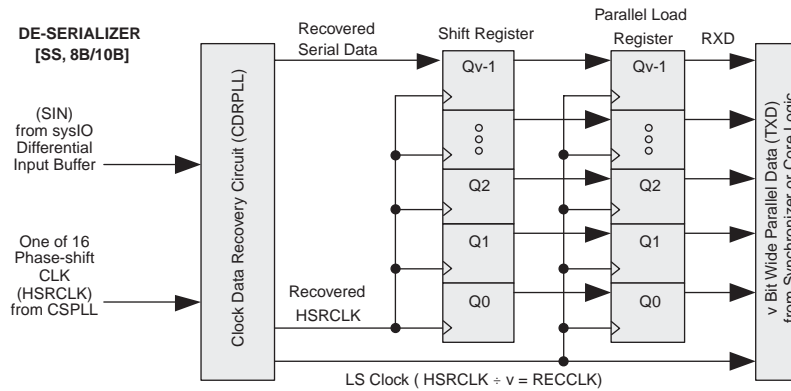
Figure 6. Serializer [10B/12B]



**De-Serializer**

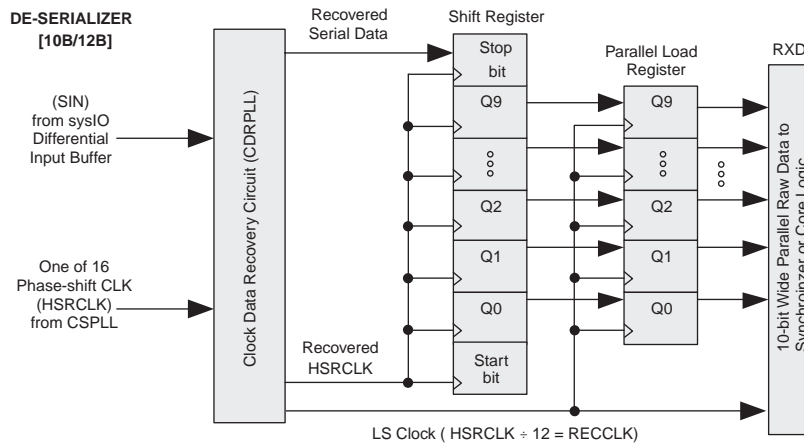
Receiver receives high speed serial data stream, SIN, from sysIO and De-Serializes into low speed Parallel Data, RXD, before it sends to Synchronizer or core logic.

Figure 7. De-Serializer [SS, 8B/10B]



10B/12B De-Serializer decodes incoming serial data, disregarding the first bit (start bit) and the last bit (stop bit) of a received symbol before serial-to-parallel conversion.

Figure 8. De-Serializer[10B/12B]

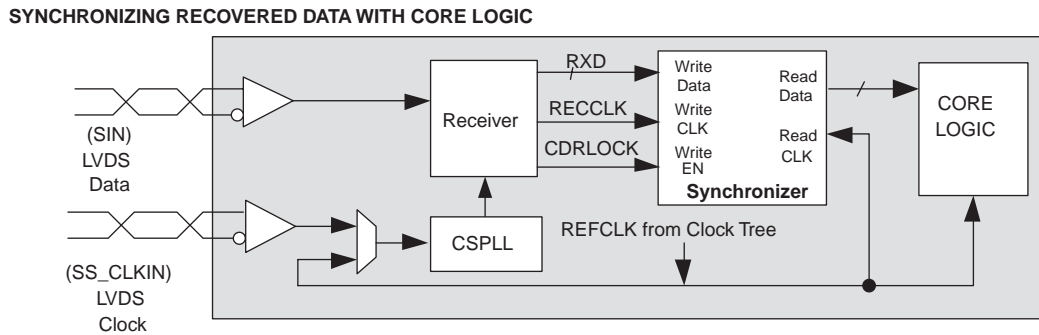


### Synchronizer

In the Receiver, the sysHSI Block writes with the Recovered Clock (RECCLK) and the Core Logic uses the system clock (usually REFCLK) to read. Depending on the device, FIFO or the Embedded Memory Block are used as a synchronizer. The use of a synchronizer is optional and may be bypassed if the user performs synchronization outside of the device.

Parallel Transmit Data enters the Transmitter of the sysHSI block from core logic clocked by REFCLK. The REFCLK at the same time is fed to CSPLL to generate the high speed Clock to transmit serialized data (HSTCLK). In the Transmitter, the REFCLK is re-aligned by the high speed clock to generate the parallel load clock to the Serializer shift register. If the skew between the REFCLK and the high speed Clock at Transmitter is larger than one high speed clock cycle then a synchronizer is required. Since the CSPLL drives only two transmitter and two receiver channels, the skew is manageable at 1 Gbps without a synchronizer. Figure 9 shows the synchronizer interface between core logic and receiver.

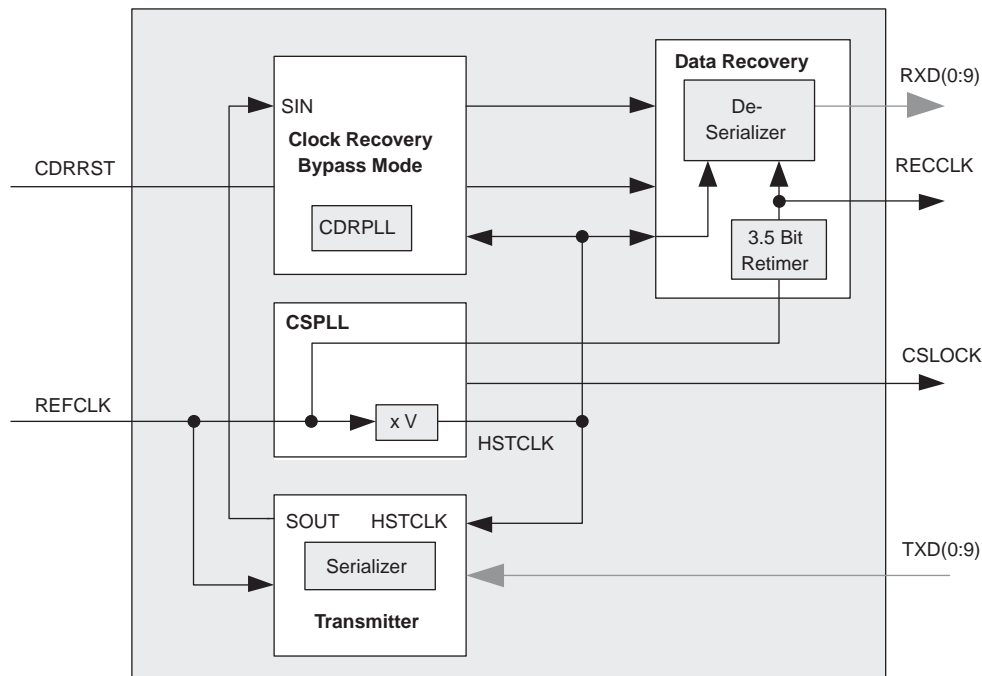
Figure 9. Synchronizer



### High Speed Loop Back Mode

In High Speed Loop Back (HSLB) mode, the transmitter output is looped back to the receiver input and the receiver uses the High Speed Transmitter Clock (HSTCLK) as its high speed clock. In this mode, Serializer, De-Serializer and CSPLL are running at full clock speed while the CDRPLL and differential I/Os are bypassed. This mode can be used to test a sysHSI Block in the ATE tester. The ATE only needs to provide parallel transmit data[TXD(0:9)] and analyze parallel receive data[RXD(0:9)]. Figure 10 shows the High Speed Loop Back mode diagram.

Figure 10. High Speed Loop Back Mode Diagram



### sysHSI Block and Source-Synchronous Mode with Multiple Data Channels

Each chip includes two groups of sysHSI Blocks. All sysHSI Blocks of the same group share the same LVDS clock input/output in Source-Synchronous mode.

In this mode, a whole group or a portion of a group can be used. The LVDS Clocks, SS\_CLKIN and SS\_CLKOUT, are connected to dedicated pins. Refer to Appendices C and D for detail.

Each group can be configured as either Receive mode or Transmit mode but not both. In Receive mode, the incoming LVDS Clock (SS\_CLKIN) is the input clock to CSPLL as a reference clock. In Transmit mode, the reference clock source is one of four clocks from the Clock Tree. The LVDS output Clock, SS\_CLKOUT is generated from the

CSPLL of the dedicated sysHSI Block in each group. Refer to Appendices C and D for the sysHSI Block Usage Map.

An example of Source-Synchronous Mode Block diagram is shown in Figures 11 and 12. Figure 11 illustrates how the HSI circuit is implemented in Source-Synchronous Receiver Mode. The example uses three sysHSI Blocks (numbers 1, 2 and 3 in group1) and consists of six data channels and one clock channel. The parallel data bit width is 8. Refer to Appendices C and D for sysHSI Block numbering and dedicated sysHSI Blocks.

**Figure 11. Source-Synchronous Mode Example Diagram (CDRX\_SS\_8)**

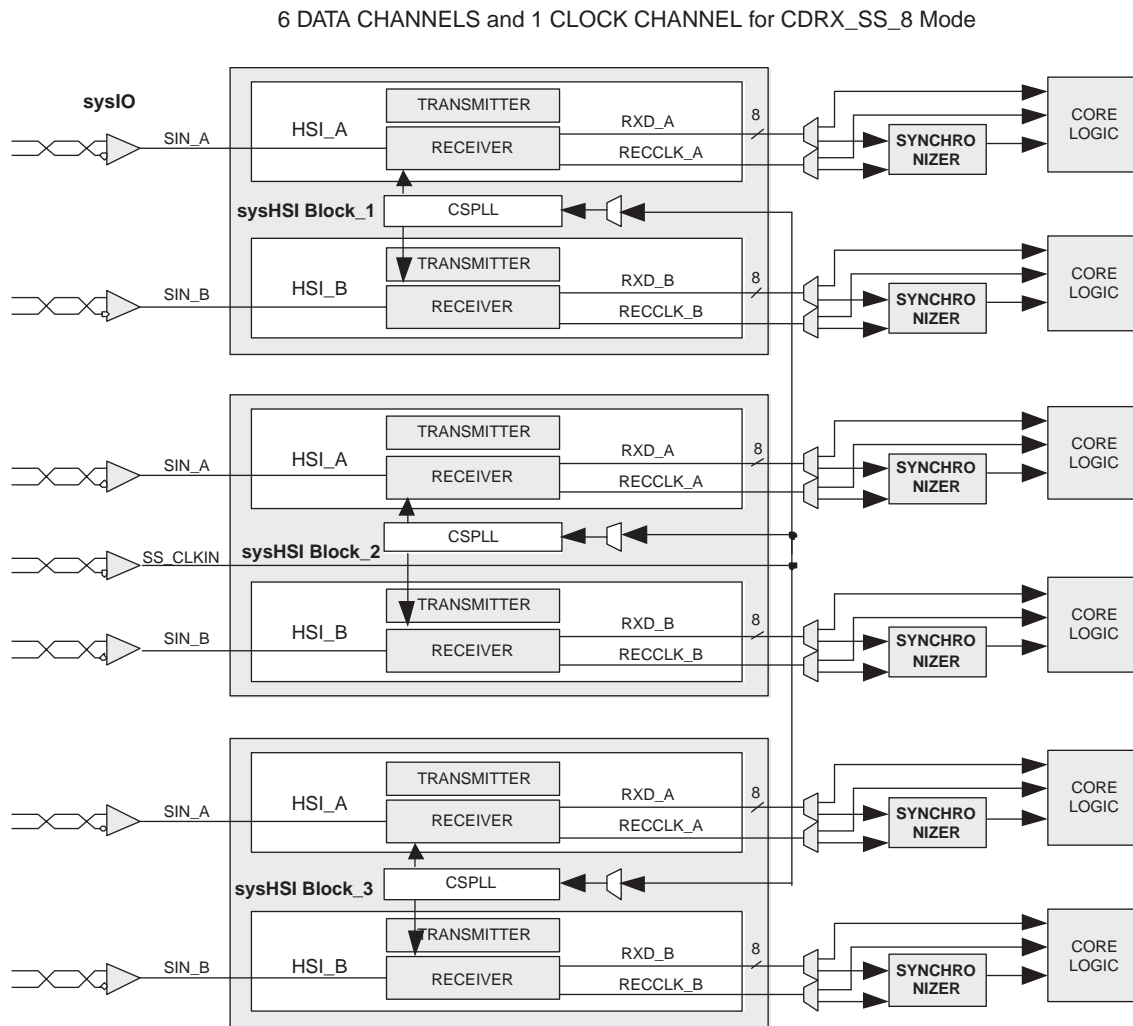
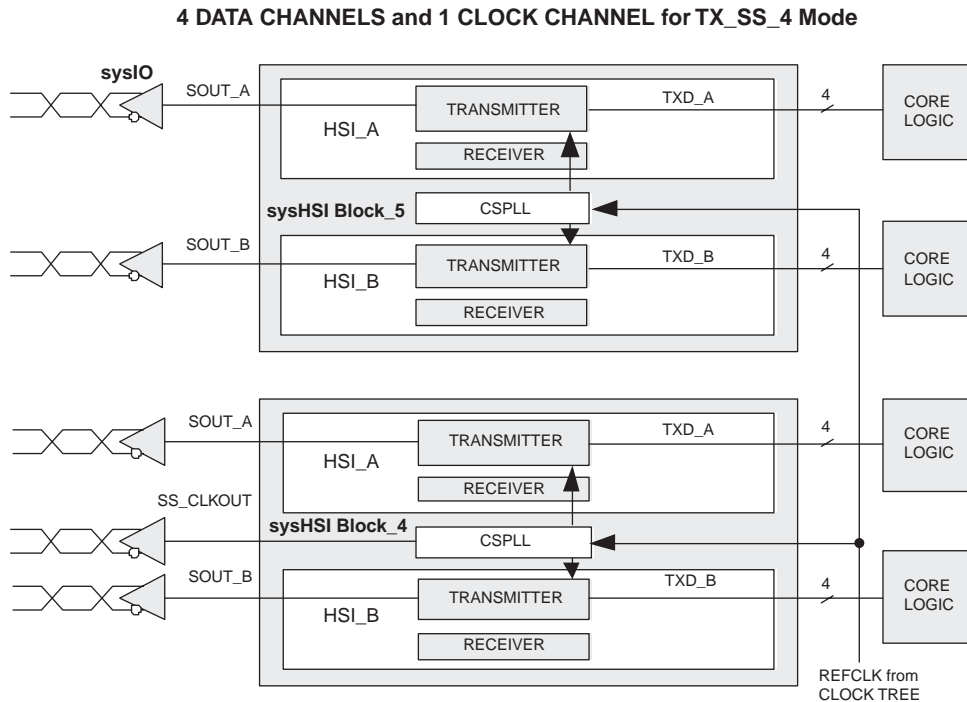


Figure 12 shows the Source-Synchronous transmitter mode. The example uses two sysHSI Blocks (numbers 4 and 5 in group 2) and consists of four data channels and one clock channel. The parallel data bit width is 4.

Figure 12. Source-Synchronous Mode Example Diagram (TX\_SS\_4)



## Using sysHSI Blocks in Design Tools

### Introduction to Macros

#### sysHSI Block Usage in CDR Mode

When both channels are used in a same sysHSI Block, they must share the same REFCLK, HSRCLK (High Speed Receiver Clock) and HSTCLK (High Speed Transmitter Clock) from the CSPLL. Multiple modes may be implemented using different sysHSI Blocks but the user must take phase jitter from different clock sources into consideration. This jitter increase may cause both receiver and transmitter performance to fall outside the guaranteed specifications.

The two SERDES Blocks, HSI\_A and HSI\_B, in the same sysHSI Block are independent from each other except sharing the same REFCLK and CSPLL.

#### sysHSI Block USAGE in Source-Synchronous Mode

When sysHSI Blocks are configured in the Source-Synchronous mode, the whole group is not available for other modes. But the sysIOs of unused sysHSI channels are available for other general I/O uses.

#### Macro Symbols

Thirteen functional macro modules are available representing seven different applications. These programmable modules are described in Table 2. Additionally, two macros are provided for high speed loop back testing and are supported for 8B/10B and 10B/12B modes.

Table 2. Macro Definitions

Mode	Symbol	Description
SS	RX_SS_x <sup>1</sup>	SS normal receive mode (no de-skew)
	CDRX_SS_x	SS De-skew receive mode
	TX_SS_x	SS transmit mode
10B12B	CDRX_10B12B	10B/12B CDR receive mode
	TX_10B12B	10B/12B transmit mode
8B10B	CDRX_8B10B	8B/10B CDR receive mode
	TX_8B10B	8B/10B transmit mode
8B10B	HSLB_8B10B	8B/10B High Speed Loop Back mode
10B12B	HSLB_10B12B	10B/12B High Speed Loop Back mode

1. x: Data width, 4, 6 or 8 resulting the total number of macros are 15.

## sysHSI Block and CSPLL I/O Description

### CSPLL I/O Description

**REFCLK\*** Reference Clock Input from Clock Tree. Used as Transmitter low speed Clock and Receiver Frequency Reference for Receiver. When this clock is used sysCLOCK PLL must be in Bypass Mode. User can select one of 4 REFCLKs available from the clock tree. Refer to Appendices C and D for available clocks at each sysHSI Block.

**CSLOCK\*** “1” indicates CSPLL is locked to SS\_CLKIN for Source-Synchronous mode, and for CDR mode, “1” indicates CSPLL is locked to REFCLK.

**SS\_CLKIN** Receiver Input Clock from LVDS Input Buffer to CSPLL as the Reference Clock in Source-Synchronous Mode. This clock input must be up and running at power up or system reset for at least 10 us to let CSPLL complete lock-in and active all the time to keep CSPLL stay lock. For low speed data communication this clock is not required. There are two SS\_CLKIN input pin pairs, one on each sysHSI Group.

**SS\_CLKOUT** Transmitter output Clock from CSPLL to LVDS Output Buffer. Only one clock is needed from designated sysHSI Block CSPLL on each sysHSI Group. See Appendices C and D for location of designated sysHSI Block. For low speed data communication this clock is not required.

\*Note: If both SERDES blocks are used in the same sysHSI Block, CSLOCK and REFCLK may be specified only once because CSPLL is shared. If both SERDES blocks specify these signals, they must refer to the same net names in the upper module.

### sysHSI Block I/O Description

**SIN** Serial input Data from LVDS Input Buffer.

**SOUT** Serial output Data to LVDS Output Buffer.

**CDRRST** “1” resets CDR to start “lock-in” process. At the chip level, CDRRST is user-controllable per the CDR channel. Minimum pulse width is three REFCLK cycles.

**RXD\*** Receiver Output low speed Parallel Data to Core logic.

**TXD\*** Transmitter Input low speed Parallel Data from Core logic.

**RECCLK** Receiver Output low speed recovered Clock.

**LOSS** “1” means Receiver Loss of Synchronization in 10B/12B mode. Or it is forced by EXLOSS=1 in 8B/10B mode.

<b>EXLOSS</b>	From optional external Optic LOSS detector. “1” indicates no signal and forces LOSS to 1.
<b>CDRLOCK</b>	<p>“1” means CDR finished Lock-In for CDR mode. This signal can be used as WE for asynchronous FIFO receiving RXD.</p> <p>For CDRX_SS, “1” means CDR finished Calibration.</p> <p>For 10B/12B, I/O block needs to send this signal to chip pin. On board, this signal can be connected to the SYNC input of the transmitter to tell if more SYNCPAT is needed.</p>
<b>SYDT</b>	Symbol Detected. When CDRLOCK=1, SYDT is set to 1 If RXD data matches the Sync Pattern. Falling edge of this signal can be used as beginning of real data.
<b>CAL</b>	From chip pin, “1” enables CDR calibration for CDRX_SS mode, rising edge resets the CDR to start lock-in process, falling edge latches the CDR calibration results by forcing its Push Pull to “0” to stay. Only one I/O pin is needed per group. In ispGDX2 devices, only one pin is provided for both sysHSI groups.

Note \*: The width of RXD and TXD; For SS: x = 4, 6, or 8 bit wide.

10B/12B: 10bit wide (Encoding and Decoding is done in HSI module).

8B/10B: 10bit wide (Encoding and Decoding is not done in HSI module).

Table 3 summarizes the sysHSI Block I/O usage for each macro

**Table 3. sysHSI IO Usage for Each Macro**

Port	I/O	SS			10B/12B		8B/10B	
		RX	CDRX	TX	CDRX	TX	CDRX	TX
REFCLK	I			X	X	X	X	X
SS_CLKIN	I	X	X					
SS_CLKOUT	O			X				
CSLOCK	O	X	X	X	X	X	X	X
SIN	I	X	X		X		X	
SOUT	O			X		X		X
CDRRST	I				X		X	
RXD	O	X	X		X		X	
TXD	I			X		X		X
RECCLK	O	X	X		X		X	
LOSS	O				X		X	
EXLOSS <sup>1</sup>	I						X	
CDRLOCK	O		X		X		X	
SYDT	O		X		X		X	
CAL <sup>1</sup>	I		X					

1. Signals EXLOSS, SYDTRST, and CAL are optional signals

### 4.3. User Parameters

**SYMPAT** Symbol Alignment Pattern. After CDR Locks in with Synchronization Pattern at high speed bits, the high speed serial data is converted to low speed parallel data and CDR performs Symbol Alignment. A 20 bits- or 12 bits-deep Comparator is used for Symbol Alignment Pattern. Each macro has its own default pattern (Table 4), but users can use their own patterns. In this case, users must rely on their own characterization of performance. The leading bit corresponds to the first bit received. Symbol alignment is not required in normal Source-Synchronous mode.

**Table 4. Default Symbol Alignment Patterns**

Function Mode	Symbol Alignment Pattern <sup>1</sup> (Byte Alignment)	Synchronization Pattern (Bit Alignment)
CDRX_8B10B	11000001010011111010	K28.5, D21.4, D21.5, D21.5 (Idle Pattern) <sup>2</sup>
CDRX_10B12B	111111000000	111111000000
CDRX_SS_4	110011001100	1100
CDRX_SS_6	111000111000	111000
CDRX_SS_8	000011110000	11110000

1. The left most bit of the Symbol Alignment Pattern is the LS Bit and transmitted or received first. These patterns are default patterns for each CDR macro and are transparent to users when using sysHSI macros provided by Lattice.

2. 0011111010, 1010101101, 1010101010, 1010101010

Synchronization patterns are not automatically output in 10B/12B and Source-Synchronous Transmit modes.

The logic that drives sysHSI Block in 10B/12B and Source-Synchronous Transmit mode must output at least 1024 synchronization patterns as preamble before the real data packet.

- **Synchronization (bit alignment):** This process is the clock and data recovery performed by the CDR PLL. The high speed clock generated by CSPLL is synchronized to incoming data bits. When this is done, the CDRLOCK signal goes high.
- **Symbol Alignment (byte alignment):** When the CDR PLL locks, the next step is Symbol Alignment or 'Byte Alignment'. This process finds the word boundary.

**PHASE\_ADJ** Phase Adjustment Value of CDRPLL for RX\_SS mode. 4 bits represent in 16 steps of 6.25% of a full clock period. This Phase Shift Value is an integer between -7 and +7 and the amount of phase shift achieved is given by the equation:  $\text{Phase Shift} = \text{PHASE\_ADJ}/16 * \text{HSRCLK Period}$ . The two CDRPLLs in the same sysHSI block can have different Phase Adjustment Values for different skew cancellation. This attribute is optional and is for Source-Synchronous normal receive mode only.

**IN\_FREQ** The IN\_FREQ attribute specifies the frequency of REFCLK (SS\_CLKIN for Source-Synchronous mode) input to CSPLL in MHz. See Parallel Data/Clk column in Table 1. This attribute is a mandatory input.

Table 5 summarizes these attributes.

**Table 5. sysHSI Block Attributes**

Attribute	Description	Optional
IN_FREQ	Frequency of Reference Clock (REFCLK or SS_CLKIN).	No
SYMPAT	Symbol alignment pattern.	Yes <sup>1</sup>
PHASE_ADJ	CDRPLL phase adjustment value for normal Source-Synchronous mode.	Yes

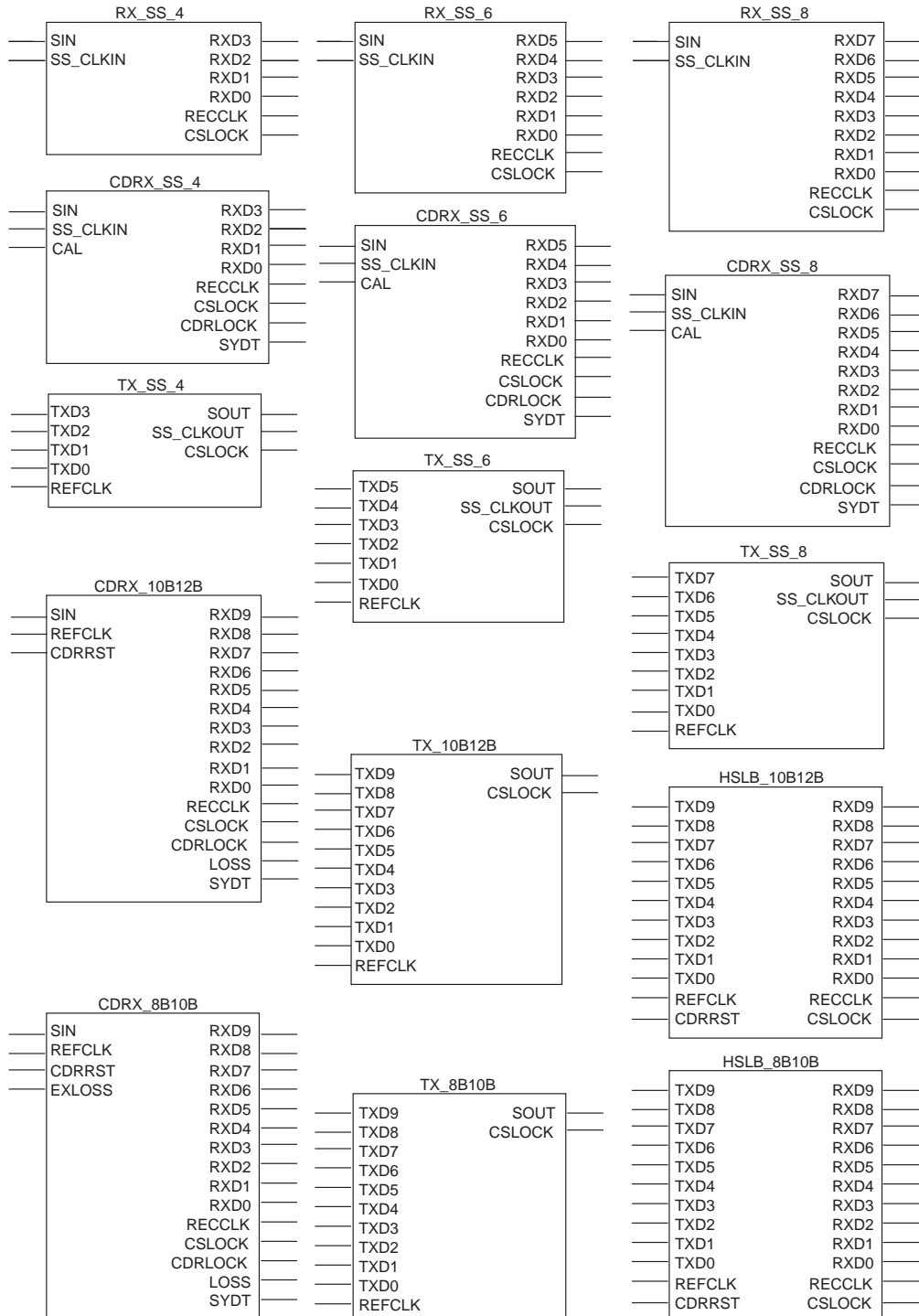
1. This attribute is optional only when users use their own pattern with Lattice-supported sysHSI macros. In this case, the AC and DC specifications of the data sheet do not apply.

**Macros Symbols of sysHSI Block**

Fifteen symbols are shown in Figure 13.

Note: A SERDES unit can be configured as either a transmitter or a receiver. In CDR modes, both Receiver and Transmitter can be configured in the same SERDES block, but in Source-Synchronous mode they cannot co-exist because the Receiver and Transmitter cannot share SS\_CLKIN as a Reference Clock.

**Figure 13. sysHSI Macro Symbols**



**HSLB\_10B12B and HSLB\_8B10B**

The HSLB\_10B12B and HSLB\_8B10B macros are used to configure the sysHSI block in Loop-Back mode. The High Speed Loop-Back (HSLB) mode provides offline testing capability of the sysHSI Block to ensure the integrity of the high speed channel. In High Speed Loop Back (HSLB) Mode, the transmitter output, SOUT, is looped back to SIN of the receiver and the receiver high speed clock utilizes the transmitter’s HSTCLK. In this mode, Serializer, Data Recovery, De-serializer, and CSPLL are running at full Serial Clock speed while the CDRPLL, SIN and SOUT are bypassed.

**sysHSI Usage with HDLs**

Refer to Appendix B.

**sysIO Usage with sysHSI Block**

The sysIO circuitry converts the differential LVDS signal to a single logic signal in the device, referred to as sysIO LVDS and BLVDS. For Point-to-Point and single terminated Multi-Drop applications, the 3.5 mA LVDS current driver is used. Multi-Point and double terminated Multi-Drop applications require higher driving capacity. The Lattice ispGDX2 family offers a 10.7 mA BLVDS driver for this type of application. The sysIOs of unused sysHSI channels are available for other general sysI/O uses.

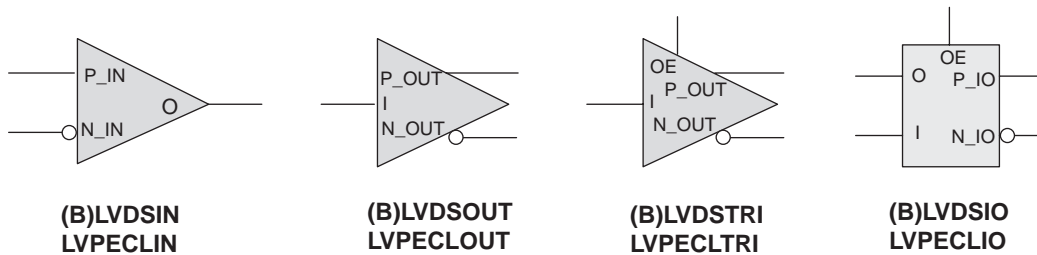
**sysIO LVDS/BLVDS/LVPECL Macro Definition**

The sysIO LVDS/BLVDS/LVPECL macros are simply input (LVDSIN/BLVDSIN/LVPECLIN), output (LVDSOUT/BLVDSOUT/LVPECLOUT), tri-state output (LVDSSTRI/BLVDSSTRI/LVPECLTRI), and bi-directional buffers (LVDSIO/BLVDSIO/LVPECLIO).

LVPECL macros are not available in sysHSI Block applications. In the ispXPGA device family, no BLVDS macros are available for sysHSI Block applications.

For more information on these differential buffers, refer to Lattice technical note TN1000, *sysIO Usage Guidelines for Lattice Devices*.

**Figure 14. sysIO LVDS/BLVDS/LVPECL Macro Symbols**



When macros LVDSIN and LVDSOUT are associated with SS\_CLKIN and SS\_CLKOUT respectively, the P\_IN, N\_IN, P\_OUT and N\_OUT must use dedicated pairs of pins in the sysHSI group.

Note: LVDSIO and BLVDSIO are not supported when these differential I/Os are used in conjunction with the sysHSI block.

**sysIO LVDS and BLVDS IO\_TYPES Usage with HDLs**

Refer to Appendix B.

## Appendix A. sysHSI Usage with HDLs

Synthesis tools such as Synplicity and Exemplar “black-box” the VHDL and Verilog instantiations and pass them through an EDIF netlist to the Lattice software. The Lattice software converts the “black-box” into the physical representation of the sysHSI within the device using the macros defined above. Verilog and VHDL pass the sysHSI attributes through parameters and generics, respectively.

Unlike other HDLs, ABEL requires special additions to support sysHSI functionality, the Lattice design tools provide direct support for ABEL and have been modified to support sysHSI functionality.

Below are VHDL, Verilog, and ABEL examples of instantiating these modules in the source code. Table 6 lists the acronyms that apply to the syntax.

**Table 6. sysHSI Acronyms**

Acronym	Definition	Optional
IN_FREQ	Reference Clock Frequency (REFCLK or SS_CLKIN)	No
PHASE_ADJ	CDRPLL Phase Adjustment Factor for Normal Source-Synchronous Mode	Yes <sup>1</sup>
SYMPAT	Symbol Alignment Pattern	Yes

1. This attribute is optional only when users use their own pattern with Lattice-supported sysHSI macros. In this case, the AC and DC specifications of the data sheet do not apply.

## Parameter Configuration Examples in Verilog

### CDRX\_SS\_4 Parameter Configuration

```
//exemplar attribute I1 in_freq 150.0000
```

### CDRX\_8B10B Parameter Configuration

```
//exemplar attribute I1 in_freq 70.0000
```

### TX\_SS\_8 Parameter Configuration

```
//exemplar attribute I1 in_freq 80.0000
```

## Instantiation Examples in Verilog

### RX\_SS\_4 Instantiation

```
RX_SS_4 I1 (.SIN(SIN), .SS_CLKIN(SS_CLKIN), .RXD0(RXD0), .RXD1(RXD1),
           .RXD2(RXD2), .RXD3(RXD3), .RECCLK(RECCLK), .CSLOCK(CSLOCK));
```

### CDRX\_SS\_4 Instantiation

```
CDRX_SS_4 I1 (.SIN(SIN), .SS_CLKIN(SS_CLKIN), .CAL(CAL), .RXD0(RXD0), .RXD1(RXD1),
           .RXD2(RXD2), .RXD3(RXD3), .RECCLK(RECCLK), .CSLOCK(CSLOCK),
           .CDRLOCK(CDRLOCK), .SYDT(SYDT));
```

### TX\_SS\_4 Instantiation

```
TX_SS_4 I1 (.TXD0(TXD0), .TXD1(TXD1), .TXD2(TXD2), .TXD3(TXD3), .REFCLK(REFCLK),
           .SOUT(SOUT), .SS_CLKOUT(SS_CLKOUT), .CSLOCK(CSLOCK));
```

### CDRX\_10B12B Instantiation (Refer to Appendix E for the CDRRST active level for ispGDx2)

```
CDRX_10B12B I1 (.SIN(SIN), .REFCLK(REFCLK), .CDRRST(CDRRST),
              .RXD0(RXD0), .RXD1(RXD1), .RXD2(RXD2), .RXD3(RXD3), .RXD4(RXD4),
              .RXD5(RXD5), .RXD6(RXD6), .RXD7(RXD7), .RXD8(RXD8), .RXD9(RXD9),
              .RECCLK(RECCLK), .CSLOCK(CSLOCK), .CDRLOCK(CDRLOCK), .LOSS(LOSS),
              .SYDT(SYDT));
```

**CDRX\_8B10B Instantiation (Refer to Appendix E for the CDRRST active level for ispGDX2)**

```
CDRX_8B10B I1(.SIN(SIN), .REFCLK(REFCLK), .CDRRST(CDRRST), .EXLOSS(EXLOSS),
  .RXD0(RXD0), .RXD1(RXD1), .RXD2(RXD2), .RXD3(RXD3), .RXD4(RXD4),
  .RXD5(RXD5), .RXD6(RXD6), .RXD7(RXD7), .RXD8(RXD8), .RXD9(RXD9),
  .RECCLK(RECCLK), .CSLOCK(CSLOCK), .CDRLOCK(CDRLOCK), .LOSS(LOSS),
  .SYDT(SYDT));
```

**TX\_8B10B Instantiation**

```
TX_8B10B I1(.TXD0(TXD0), .TXD1(TXD1), .TXD2(TXD2), .TXD3(TXD3), .TXD4(TXD4),
  .TXD5(TXD5), .TXD6(TXD6), .TXD7(TXD7), .TXD8(TXD8), .TXD9(TXD9),
  .REFCLK(REFCLK), .SOUT(SOUT), .CSLOCK(CSLOCK));
```

**TX\_10B12B Instantiation**

```
TX_10B12B I1(.TXD0(TXD0), .TXD1(TXD1), .TXD2(TXD2), .TXD3(TXD3), .TXD4(TXD4),
  .TXD5(TXD5), .TXD6(TXD6), .TXD7(TXD7), .TXD8(TXD8), .TXD9(TXD9),
  .REFCLK(REFCLK), .SOUT(SOUT), .CSLOCK(CSLOCK));
```

**Source Code Examples in VHDL**

```
library ieee;
use ieee.std_logic_1164.all;

entity RX8B10B is
  PORT(EXREFCLK, EXCDRRST, EXEXLOSS, RxP, RxN:IN STD_LOGIC;
    EXRXD0, EXRXD1, EXRXD2, EXRXD3, EXRXD4, EXRXD5, EXRXD6, EXRXD7, EXRXD8, EXRXD9,
    EXECCLK, EXCSLOCK, EXCDRLOCK, EXLOSS, EXSYDT:OUT STD_LOGIC);
end RX8B10B;

architecture behavioral of RX8B10B is

  COMPONENT LVDSIN
  PORT(P_IN, N_IN:IN STD_LOGIC;
    O :OUT STD_LOGIC);
  END COMPONENT;

  COMPONENT CDRX_8B10B
    generic (IN_FREQ : STRING);

    PORT(REFCLK, CDRRST, EXLOSS, SIN :IN STD_LOGIC;
      RXD0, RXD1, RXD2, RXD3, RXD4, RXD5, RXD6, RXD7, RXD8, RXD9,
      RECCLK, CSLOCK, CDRLOCK, LOSS, SYDT:OUT STD_LOGIC);
  END COMPONENT;

  attribute IN_FREQ :string;
  attribute IN_FREQ of I1: label is "80.000";

  SIGNAL LVDSO:STD_LOGIC;

begin

  I0:LVDSIN
    PORT MAP(P_IN =>RxP, N_IN=>RxN, O=>LVDSO);

  I1:CDRX_8B10B
```

```
generic map (IN_FREQ => "80.0000")

PORT MAP(
REFCLK=>EXREFCLK, CDRRST=>EXCDRRST, EXLOSS=>EXEXLOSS, SIN=>LVDSO,
RXD0=>EXRXD0, RXD1=>EXRXD1, RXD2=>EXRXD2, RXD3=>EXRXD3, RXD4=>EXRXD4,
RXD5=>EXRXD5, RXD6=>EXRXD6, RXD7=>EXRXD7, RXD8=>EXRXD8, RXD9=>EXRXD9,
RECCLK=>EXRECCLK, CSLOCK=>EXCSLOCK, CDRLOCK=>EXCDRLOCK,
LOSS=>EXLOSS, SYDT=>EXSYDT);

end behavioral;
```

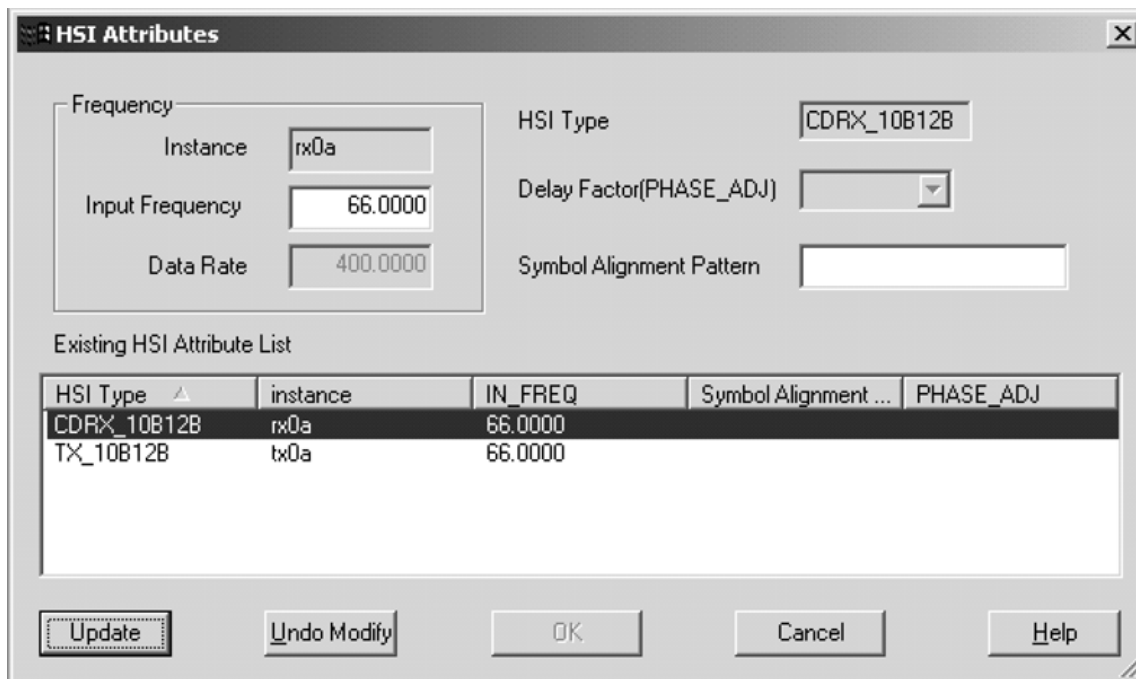
### sysHSI Usage in the ispLEVER Constraint Editor

The ispLEVER Constraint Editor includes a sysHSI Attributes Sheet. This sheet gives the user the ability to set the attributes of the sysHSI circuit. It does not allow changes in the connectivity of the sysHSI. Table 7 is an illustration of the sysHSI Attribute Sheet.

**Table 7. sysHSI Attributes Sheet**

Instance	HSI Type	IN_FREQ	Symbol Alignment Pattern	PHASE_ADJ
11	CDRX_8B10B	80.0000	—	—
12	RX_SS_4	170.0000	—	-3

**Figure 15. sysHSI Attributes Window**



### Coding Tips for sysHSI Usage in HDLs

1. **CAL**: This signal is used in Source Synchronous receiver macros CDRX\_SS\_4, CDRX\_SS\_6 and CDRX\_SS\_8. In ispXPGA devices, there are two CAL signal inputs, one for each group. In ispGDX2 devices, there is only one CAL for entire device. The instantiation of macros may share the same signal name for the CAL inputs, or only one macro may specify the CAL signal.

2. **IN\_FREQ**: This attribute is a mandatory input and must meet the specification.

sysHSI Macro	IN_FREQ Range (MHz)		V (HS/LS)	Serial Data Rate (Mbps)	
	Min.	Max.		Min.	Max.
CDRX_8B10B	40.00	85.00	10	400	850
CDRX_10B12B	33.34	70.83	12	400	850
CDRX_SS_4	100.00	200.00	4	400	800
CDRX_SS_6	66.67	133.33	6	400	800
CDRX_SS_8	50.00	100.00	8	400	800
RX_SS_4	100.00	187.50	4	400	750
RX_SS_6	66.67	125.00	6	400	750
RX_SS_8	50.00	93.75	8	400	750

3. **SYMPAT**: Symbol Alignment Pattern. This attribute is user-transparent when sysHSI macros are used as specified in the data sheet. This attribute is optional only when custom patterns are used. In this case, the user must be aware that the specification in the data sheet is no longer valid.
4. **Synchronization Pattern**: This pattern must be transmitted a minimum of 1,024 times before actual data for TX\_10B12B and all Source Synchronous transmitters. In TX\_8B10B, the idle pattern should be transmitted a minimum of 480 times.
5. **CSLOCK**: This signal is the CSPLL lock indicator. One sysHSI Block includes two full-duplex channel with two transmitters and two receivers. There a single CSPLL in each sysHSI Block. The four sysHSI macros in an sysHSI Block share this CSLOCK signal. All four macros may share the same signal name for CSLOCK or only one macro may specify the signal name for CSLOCK.
6. **REFCLK**: This reference clock is output from a four input MUX and can choose from four different global clocks. The sysHSI Block shares one REFCLK, as in the CSLOCK case above.

#### GDx2 Pin Assignments

7. **CSLOCK, LOSS, SYDT** (when used with FIFO) outputs are routed to dedicated output pins in the ispGDx2. When these signal outputs are not used, the pin is available for other output signals.
8. **RXD, RECCLK, CDRLOCK, SYDT** (when used without FIFO) outputs are routed to the input register and to the Global Routing Pool and may route to any pins available.
9. **CDRRST** pins are not available for other inputs.
10. **RXD(0:9)**: These recovered parallel data occupy the I/O cells as specified in the data sheet. These signals are routed to input registers, whether through FIFO or not, to the Global Routing Pool and to any pins available. The output pins of the I/O cells are available for other signals.
11. **TXD(0:9)**: These transmit parallel data to transmitter occupy the I/O cells as specified in the data sheet. These signals are muxed with FLAGs outputs [FIFO, PLLLOCK, CSLOCK, LOSS, SYDT (with FIFO mode)]. The I/O cells cannot share the TXD(0:9) and the FLAGs outputs. The I/O cells can share the TXD(0:9) and input signals.

## Appendix B. sysIO LVDS and BLVDS Usage with HDLs

The sysIO Differential Buffers are easily imported by instantiation.

The pin-lock of the sysIO Differential Buffer pins are optional. When instantiated without pin-lock, Lattice software automatically assigns the differential pair of pins.

### Attributes

Synplicity and Exemplar support sysIO LVDS and BLVDS using attribute passing and the IO\_TYPES attribute with VHDL and Verilog much the same as pin locations, pull-ups, and slew rates are already controlled. Similarly, ABEL HDL passes properties to the place and route tools as they are currently used.

Below is an example of how each HDL supports sysIO LVDS using the IO\_TYPES attribute. Only the P-side pin is required, the N-side is automatically assigned to the pair by the software.

### Verilog with Exemplar

```
// exemplar attribute [PinName] IO_TYPES LVDS pin,-,-,-
```

### Instantiation

Below is a Verilog example for instantiating these modules in the source code.

Table 8 lists acronyms that apply to the syntax.

**Table 8. sysIO LVDS/BLVDS Acronyms**

Acronym	Definition
P_IN	The p-side of the LVDS input
N_IN	The n-side of the LVDS input
O	The output of the LVDS input or bi-directional buffer
I	The input of the LVDS output or bi-directional buffer
P_OUT	The p-side of the LVDS output
N_OUT	The n-side of the LVDS output
OE	The output enable of the LVDS output or bi-directional buffer
P_IO	The p-side of the LVDS bi-directional signal
N_IO	The n-side of the LVDS bi-directional signal

### Instantiation Example in Verilog

```
(B)LVDSIN I1 (.P_IN(IN_P), .N_IN(IN_N), .O(NODE));
(B)LVDSOUT I2 (.I(NODE), .P_OUT(OUT_P), .N_OUT(OUT_N));
(B)LVDSTRI I3 (.I(NODE), .OE(OE), .P_OUT(OUT_P), .N_OUT(OUT_N));
(B)LVDSIO I4 (.I(NODE0), .OE(OE), .O(NODE1), .P_IO(IO_P), .N_IO(IO_N));
```

### Source Code Example in VHDL

```
library ieee;
use ieee.std_logic_1164.all;

entity LVDSIN_TOP is
  port(LVDS_IN_P:instd_logic;
        LVDS_IN_N:instd_logic;
        EXOUT:OUTstd_logic);
end LVDSIN_TOP;
```

```

architecture BEHAVE of LVDSIN_TOP is

component LVDSIN
  port(P_IN:in      STD_LOGIC;
       N_IN:in      STD_LOGIC;
       O   :out      STD_LOGIC);
end component;

begin
  I1: LVDSIN
    port map (P_IN => LVDS_IN_P,
             N_IN => LVDS_IN_N,
             O   => EXOUT);
end BEHAVE;

```

### Differential sysIO Support

The table below summarizes the differential I/O availability in ispXPGA and ispGDX2 devices when associated with sysHSI Blocks.

**Table 9. Differential sysIO Support (When Used for sysHSI Block I/O)**

sysIO Macro	ispXPGA	ispGDX2
LVDSIN	Yes	Yes
LVDSOUT		
LVDSTRI		
LVDSIO	No	No
BLVDSIN	No	Yes
BLVDSOUT		
BLVDSTRI		
BVLDSIO		No
LVPECLIN	No	No
LVPECLOUT		
LVPECLTRI		
LVPECLIO		

**Table 10. Differential sysIO Support (without sysHSI Block)**

sysIO Macro	ispXPGA	ispGDX2	ispXPLD
LVDSIN	Yes <sup>1</sup>	Yes	Yes
LVDSOUT			
LVDSTRI			
LVDSIO	No	Yes	No
BLVDSIN			
BLVDSOUT			
BLVDSTRI			
BVLDSIO	Yes <sup>1</sup>	Yes <sup>1</sup>	Yes <sup>1</sup>
LVPECLIN			
LVPECLOUT			
LVPECLTRI			
LVPECLIO			

1. For more information, see Lattice technical note TN1000, *sysIO Usage Guidelines for Lattice Devices*.

## Appendix C. sysHSI AC Specifications

### Lock-in Timing

Symbol	Description	Mode	Conditions	Min.	Max.	Units
$t_{CSLOCK}$	CSPLL Lock-Time	All	After Input is Stabilized		25	$\mu$ S
$t_{CDRLOCK}$	CDRPLL Lock-In Time	SS	With SS Mode Sync Pattern		1024	$t_{RCP}^1$
		10B12B	With 10B12B Sync Pattern		1024	$t_{RCP}$
		8B10B	With 80B10B Idle Pattern		480	$t_{RCP}$
$t_{SYNC}$	SyncPat Length	SS		1200		$t_{RCP}$
$t_{CAL}$	CAL Duration	SS		1100		$t_{RCP}$
$t_{SUSYNC}$	SyncPat Set-up Time to CAL	SS		50		$t_{RCP}$
$t_{HDSYNC}$	SyncPat Hold Time from CAL	SS		50		$t_{RCP}$

1. REFCLK clock period.

### REFCLK and SS\_CLKIN Timing

Symbol	Description	Mode	Conditions	Min.	Max.	Units
$f_{DREFCLK}$	Frequency Deviation Between TX REFCLK and CDRX REFCLK on One Link	8B10B, 10B12B		-100	100	ppm
$t_{JPPREFCLK}$	REFCLK, SS_CLKIN Peak-to-Peak Period Jitter	All	40-250 (MHz)	-0.005	0.005	UIPP
$t_{PWREFCLK}$	REFCLK, SS_CLKIN Pulse Width, (80%/20% to 80%/20%).	All		1		ns
$t_{RFREFCLK}$	REFCLK, SS_CLKIN Rise/Fall Time (20% to 80% or 80% to 20%)	All			2	ns

### Serializer Timing

Symbol	Description	Mode	Conditions	Min.	Max.	Units
$t_{JPPSOUT}$	SOUT Peak-to-Peak Output Data Jitter	All	Note 1		0.25	UIPP
$t_{JPP8B10B}$	SOUT Peak-to-Peak Random Jitter	8B10B	900 Mbps w/K28.7-		130	ps
	SOUT Peak-to-Peak Deterministic Jitter	8B10B	900 Mbps w/K28.5+		110	ps
$t_{RFSOUT}$	SOUT Output Data Rise/Fall Time, (20%, 80%)	LVDS			700	ps
		BLVDS			900	ps
$t_{COSOUT}$	REFCLK to SOUT Delay	SS/8B10B		$2Bt^1 + 2$	$2Bt + 10$	ns
		10B12B		$1Bt + 2$	$1Bt + 10$	ns
$t_{SKTX}$	Skew of SOUT with Respect to SS_CLKOUT	SS			200	ps
$t_{CKOSOUT}$	SS_CLKOUT to bit0 of SOUT	SS		$2Bt - t_{SKTX}$	$2Bt + t_{SKTX}$	ns
$t_{HSITXDDATAS}$	TXD Setup Time to REFCLK	All	Note 3	1.5		ns
$t_{HSITXDDATAH}$	TXD Hold Time from REFCLK	All	Note 3		1.0	ns

1. Bt: Bit Time Period. High-speed serial bit time.

2. The SIN and SOUT jitter specifications listed above are under the condition that the Clock Tree that drives the REFCLK to sysHSI Block is in sysCLOCK PLL BYPASS Mode.

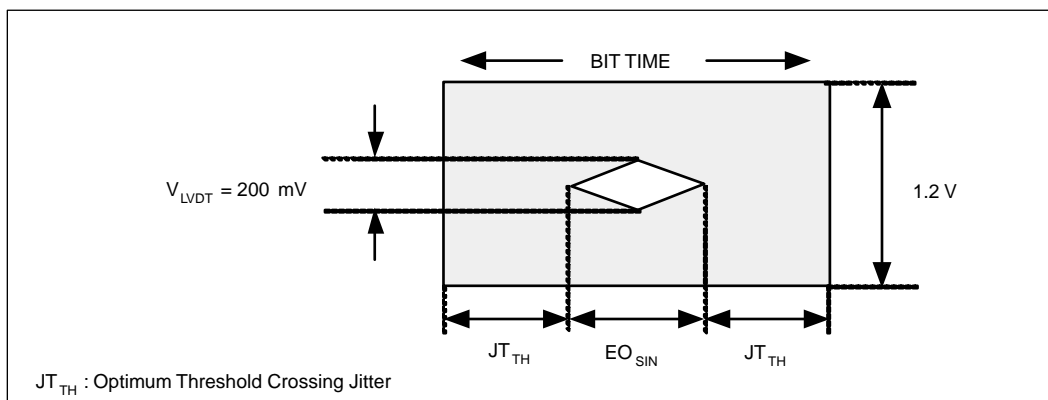
3. Internal timing for reference only.

**Deserializer Timing**

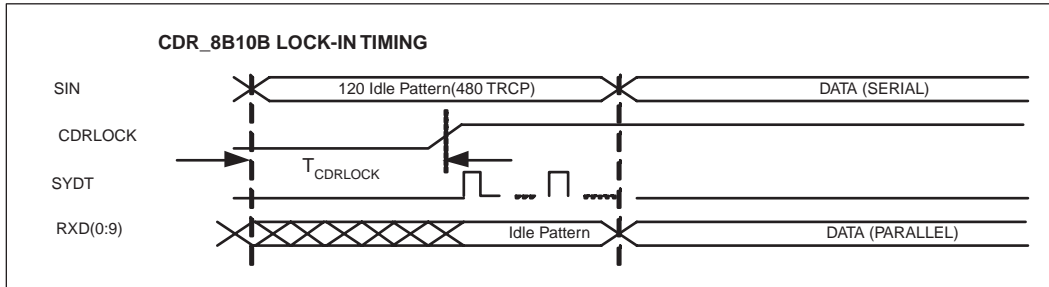
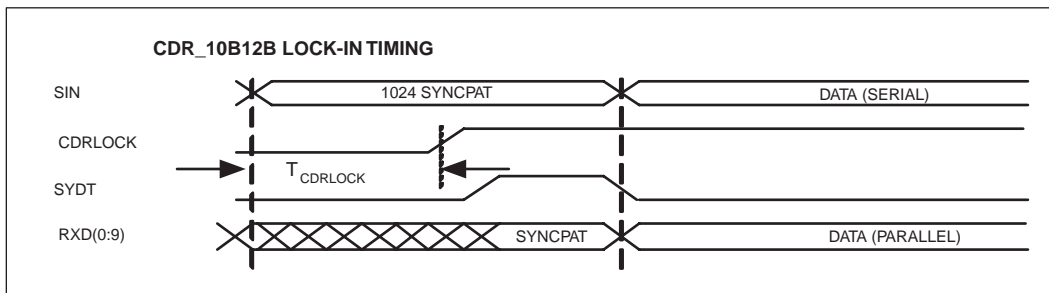
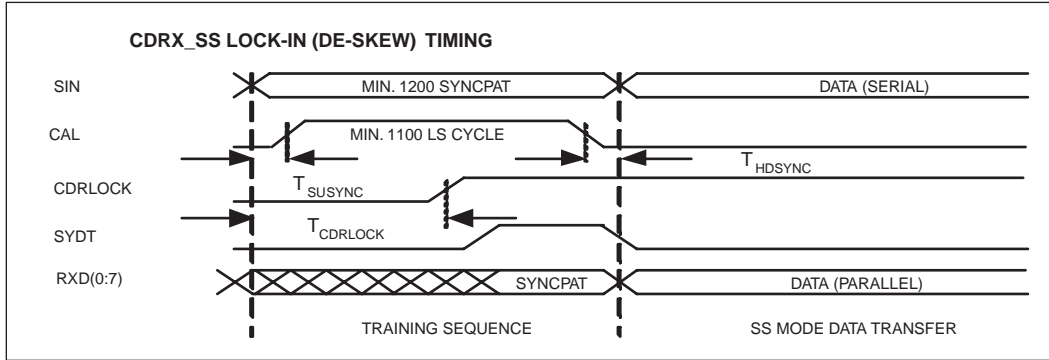
Symbol	Description	Mode	Conditions	Min.	Max.	Units
$f_{DSIN}$	SIN Frequency Deviation from REFCLK *V	8B10B/10B12B		-100	+100	ppm
$e_{OSIN}$	SIN Eye Opening Tolerance (see Eye Diagram)	All CDR		0.4		UIPP
		SS (Note 1)		0.65		UIPP
ber	Bit Error Rate	All			10-12	Bits
$t_{SKRX}$	Skew Margin Between SIN and SS_CLKIN	SS	Note 1		0.125	UIPP
$t_{CKISIN}$	SS_CLKIN to bit0 of SIN	SS	Note 1	$2Bt - t_{SKRX}$	$2Bt + t_{SKRX}$	ns
$t_{HSIOUTVALIDPRE}$	RXD, LOSS, CDRLOCK, SYDT Valid Time Before RECCLK Falling Edge	All	Note 3	$t_{RCP}/2 - 0.7$		ns
$t_{HSIOUTVALIDPOST}$	RXD, LOSS, CDRLOCK, SYDT Valid Time After RECCLK Falling Edge	All	Note 3	$t_{RCP}/2 - 0.7$		ns
$t_{DSIN}$	Bit 0 of SIN Delay to RXD Valid at RECCLK Falling Edge	All CDR		$1.5 t_{RCP} + 4.5Bt + 2$	$1.5 t_{RCP} + 4.5Bt + 10$	ns
		SS	Note 1	$1.5 t_{RCP} + 1.5Bt + 3$	$1.5 t_{RCP} + 1.5Bt + 15$	ns

1. SS normal receive mode (no de-skew option).
2. ispGDX2 only. RX\_SS mode only. This limit is increased if EO is increased.
3. Internal timing for reference only.

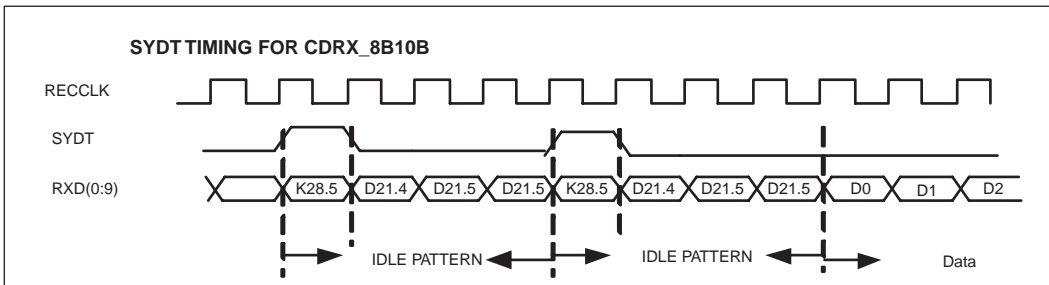
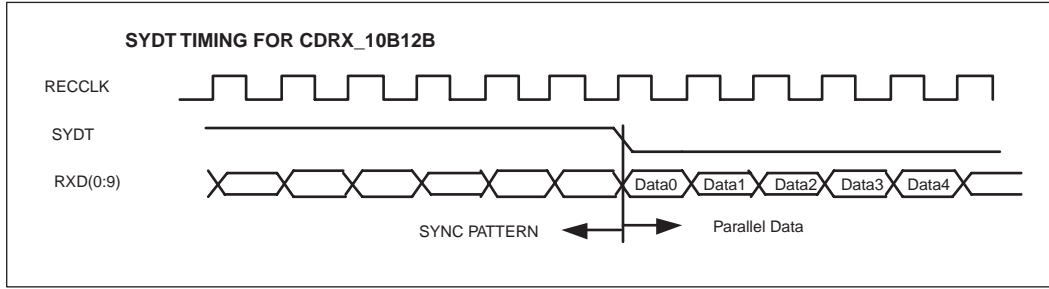
**Figure 16. Serial Input Data Eye Diagram Template (Differential)**



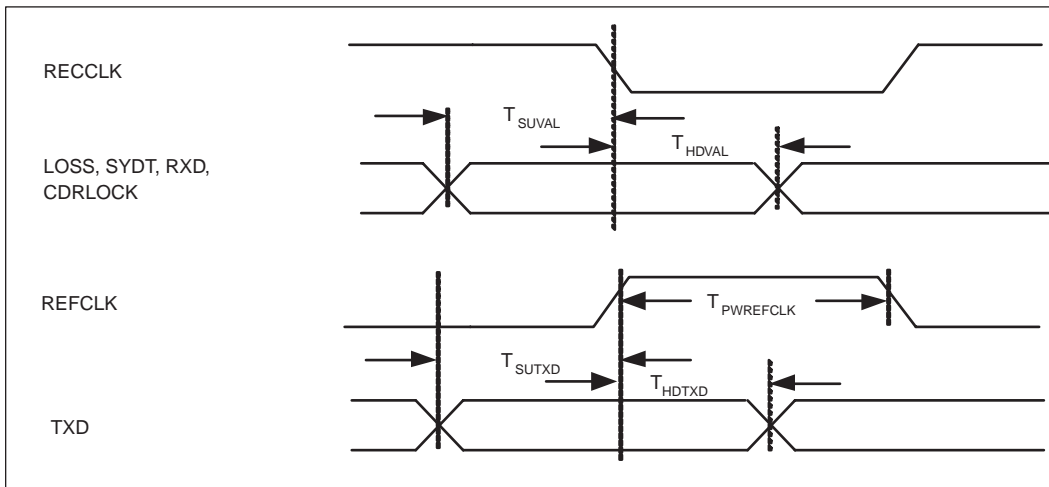
Lock-in Timing



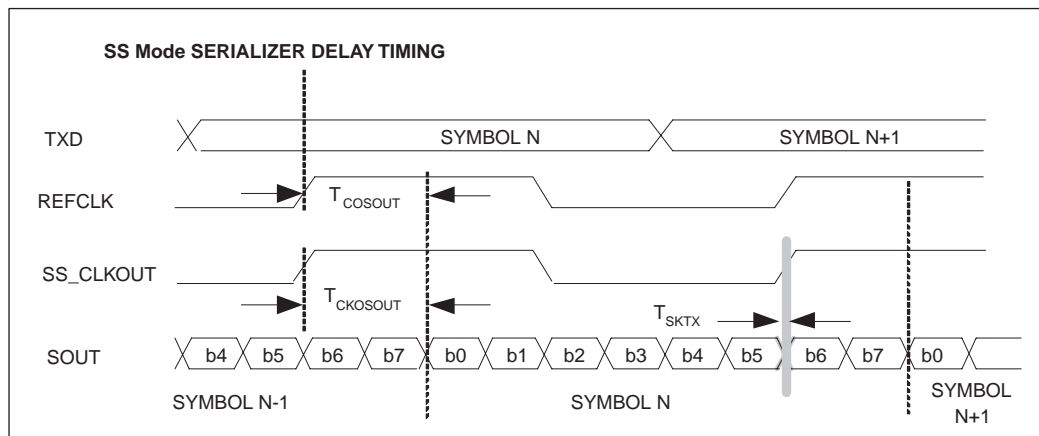
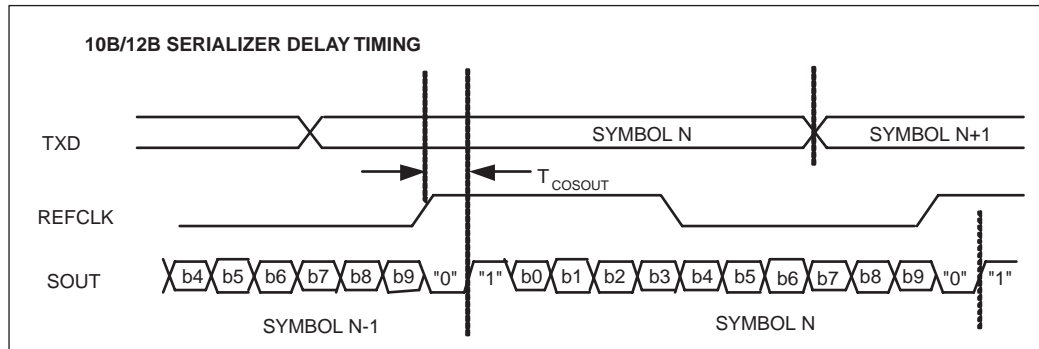
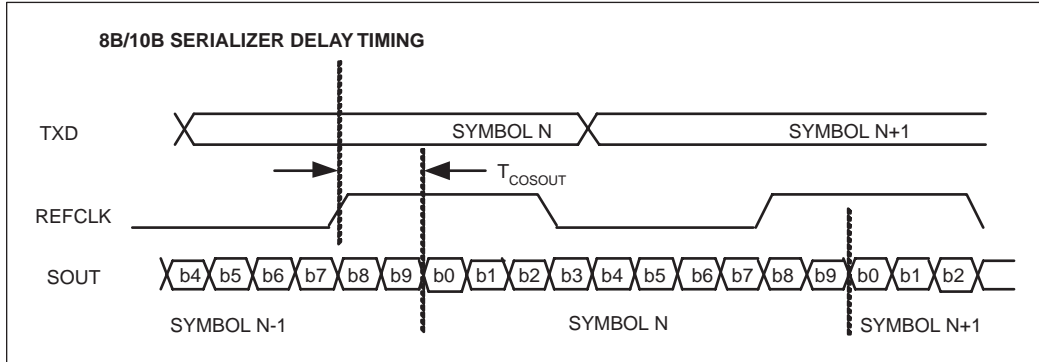
**SYDT Timing**



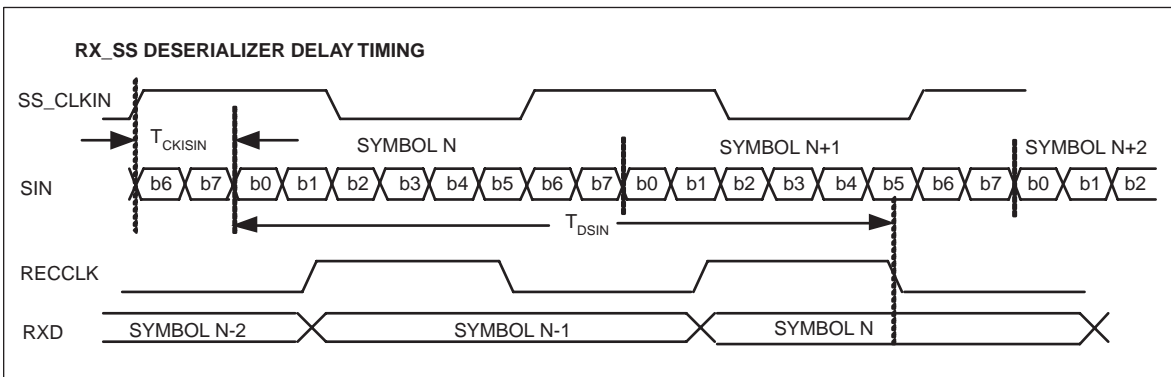
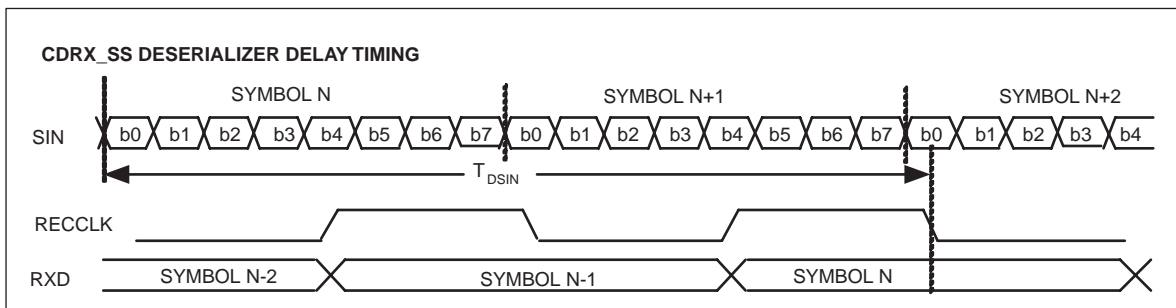
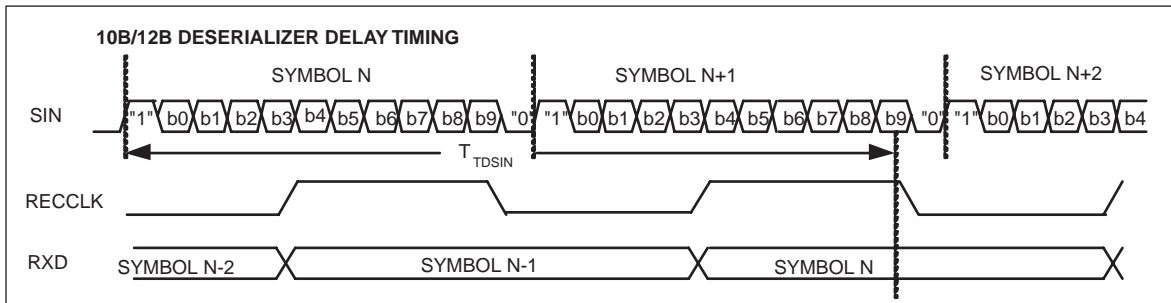
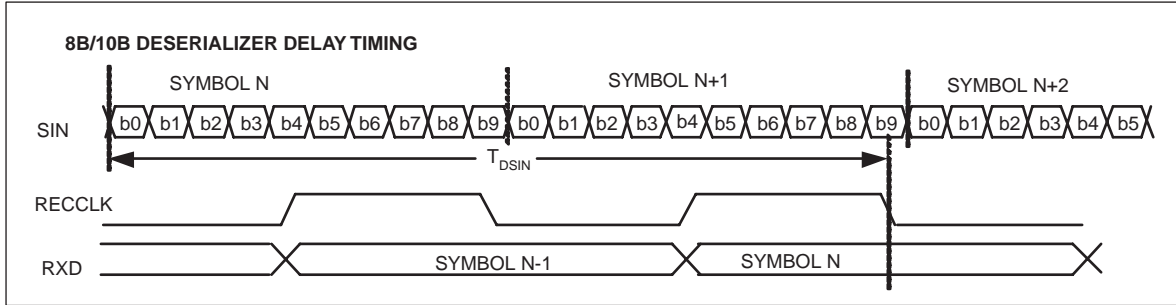
**AC Timing Diagram at sysHSI Block Boundary**



Serializer Timing



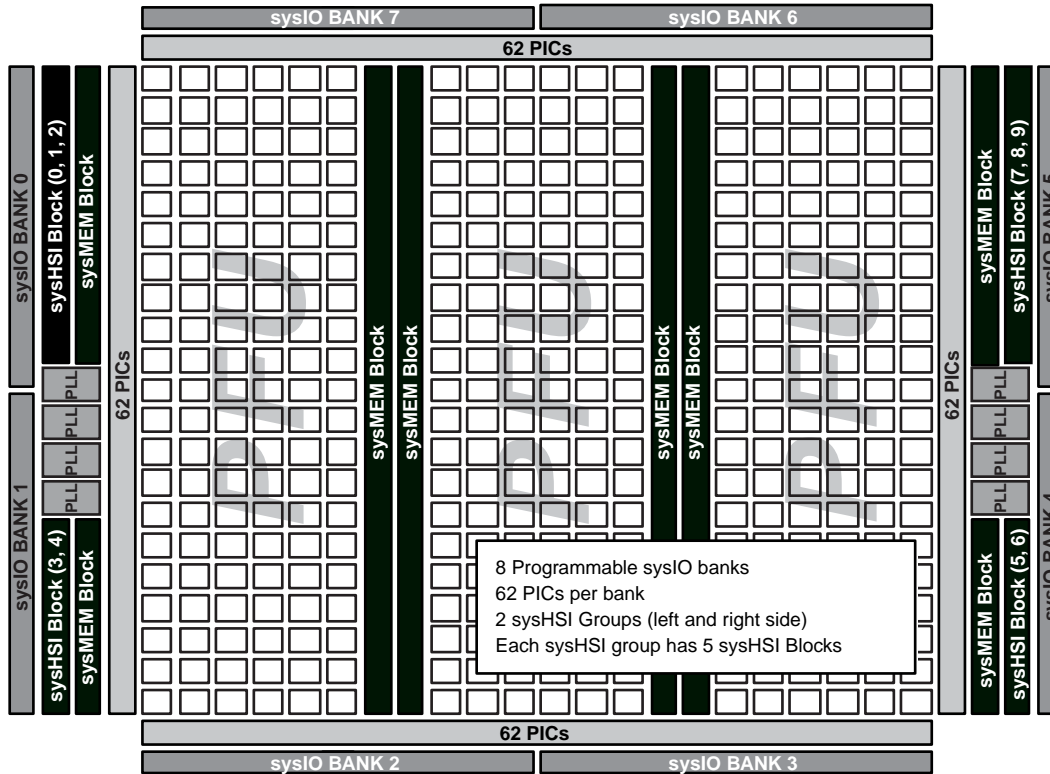
Deserializer Timing



## Appendix D. ispXPGA Family

The block diagram of ispXPGA-1200 is shown in Figure 17.

Figure 17. ispXPGA-1200 Block Diagram



## sysHSI Usage in SS Mode: ispXPGA Family 125 to 1200

Table 11. sysHSI Block Usage Guide Map

Number of Channels in Each Group					Package Type							
					1200		500		200		125	
9, 10	7, 8	5, 6	3, 4	1, 2	Group 1	Group 2	Group 1	Group 2	Group 1	Group 2	Group 1	Group 2
					0 <sup>1</sup>	9						
					1	8	0	5				
					2 <sup>2</sup>	7	1	4	0	3	0	1
					3	6	2	3	1	2		
					4	5						

1. sysHSI Block Number (see Figure 17)

2. Only the sysHSI blocks with italic numbers generate SS\_CLKOUT. They must be used in macros TX\_SS\_4,6,8. Other sysHSI blocks in the table are recommended to minimize channel-to-channel skew.

## ispXPGA-1200 Clock Tree

Table 12. ispXPGA-1200 Clock Tree

sysHSI BLOCK<0:9>	REFCLK0	REFCLK1	REFCLK2	REFCLK3
0	Clock0	Clock1	Clock2	Clock3
1	Clock0	Clock1	Clock2	Clock4
2	Clock0	Clock1	Clock2	Clock5
3	Clock0	Clock1	Clock3	Clock6
4	Clock0	Clock1	Clock3	Clock7
5	Clock0	Clock5	Clock7	Clock3
6	Clock0	Clock5	Clock7	Clock2
7	Clock0	Clock5	Clock6	Clock1
8	Clock0	Clock5	Clock6	Clock0
9	Clock0	Clock5	Clock6	Clock7

Note: This table applies to all other packages as well.

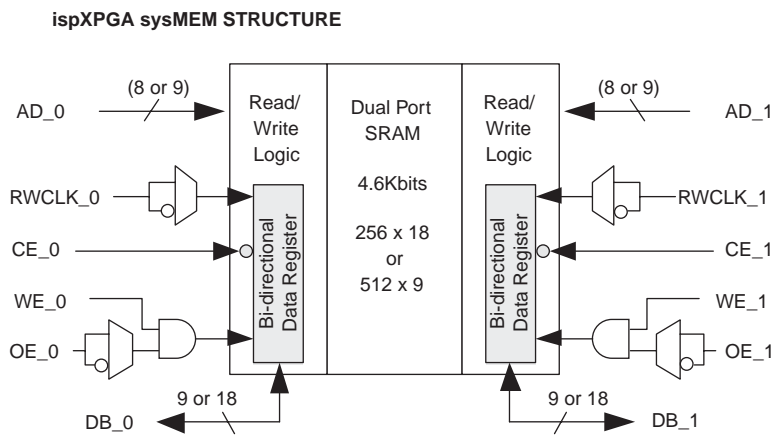
## sysMEM™ Embedded RAM (EBR) Usage as FIFO

The ispXPGA family includes sysMEM Embedded RAM Blocks that can be programmed as a FIFO for synchronization.

The embedded memory in the XPGA devices utilizes a bi-directional data bus. The RAM is configured as single-port RAM (dedicated input and dedicated output ports) because only the Receiver uses a synchronizer at up to 1GHz. The core generator will support all valid single-port configurations of the RAM up to 4096 x 18. The macros available include the basic single-port configurations, i.e. 256 x 8 and 512 x 9.

For information on EBR macros available, refer to Lattice technical note number TN1008, *ispXPGA sysMEM Memory Design and Usage Guidelines*.

Figure 18. sysMEM Block



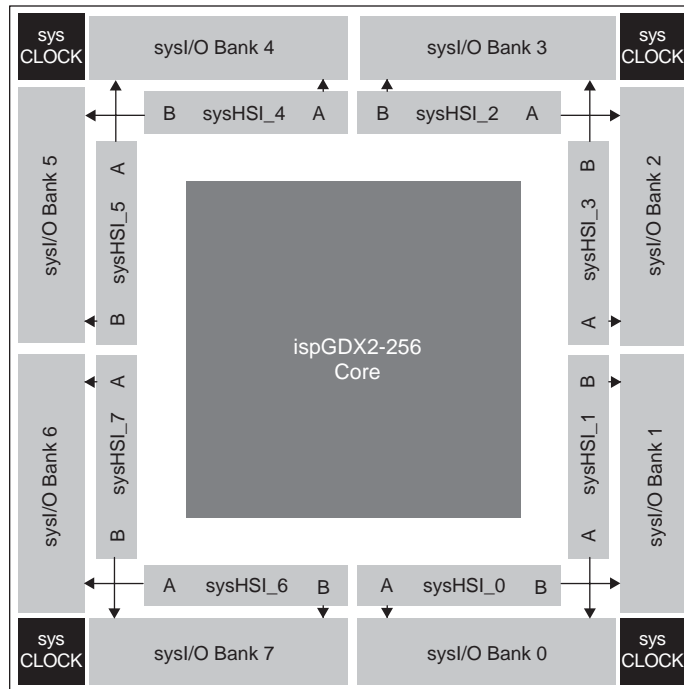
## Appendix E. ispGDX2 Family

### sysIO Banks and sysHSI Blocks

The ispGDX2 family devices are designed to minimize clock tree skew for high speed interface applications. The sysHSI sub-blocks, HSI\_A and HSI\_B are routed to nearest sysIO Bank. This is illustrated in Figure 19.

The ispGDX2-256 has eight sysHSI Blocks. Each sysHSI Block is divided to two SERDES blocks, HSI\_A and HSI\_B. Each SERDES Block occupies 16 I/O Cell Blocks in the sysIO Bank. Refer to the I/O connection table in the ispGDX2V/B/C Family data sheet for further information.

**Figure 19. ispGDX2 sysIO Bank and sysHSI Block**



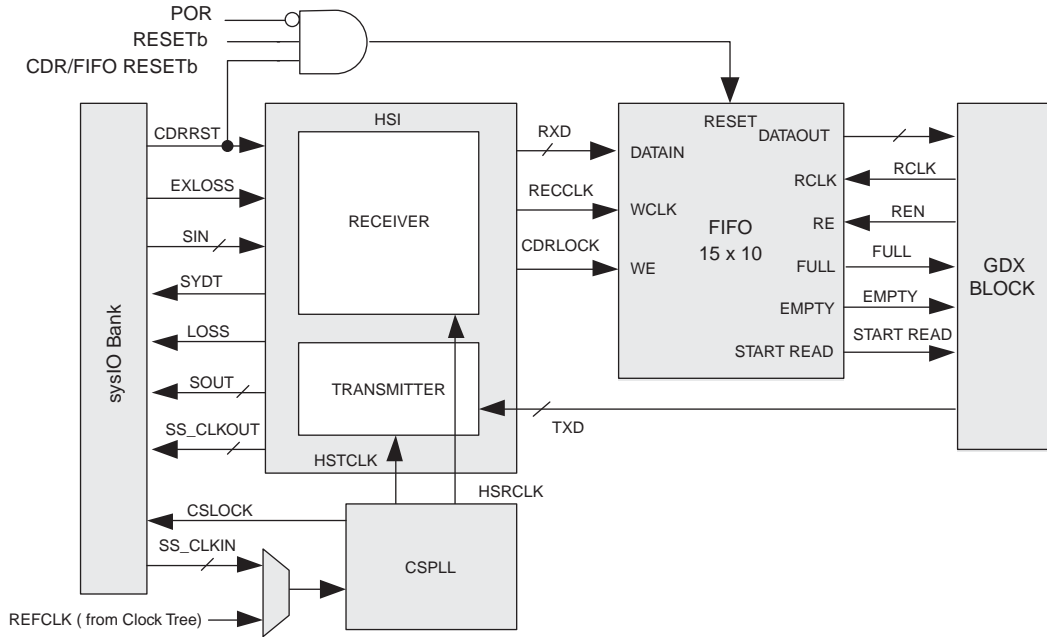
## FIFO

### sysHSI Block Interface with FIFO

The ispGDX2 Family includes dedicated FIFO for synchronization of recovered data.

The FIFO is 15 x 10 and is intended to support CDR. The usage of FIFO is optional.

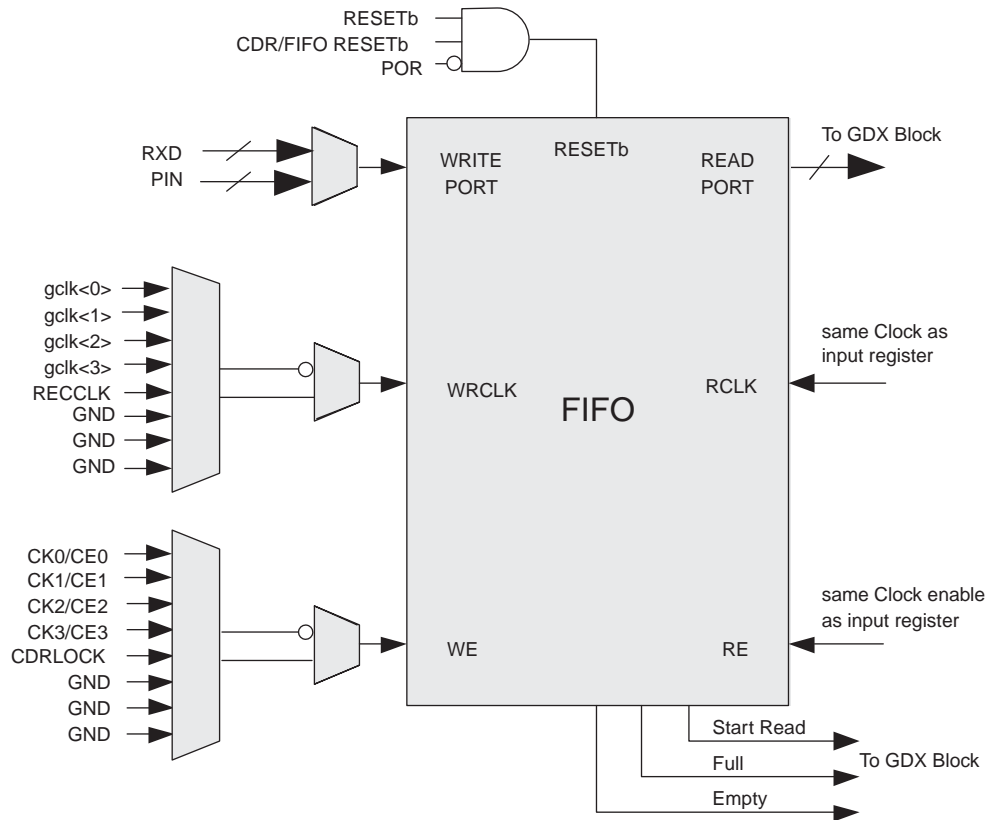
Figure 20. sysHSI Block interface with FIFO in ispGDX2



**FIFO I/O**

Figure 21 shows the I/O of FIFO.

Figure 21. FIFO I/O



The Input Register of the I/O cells are used as FIFO output registers. FIFO used the input Register clock as the Read Clock and the Input Register Clock Enable as the Read Enable.

## FIFO Macro Definitions

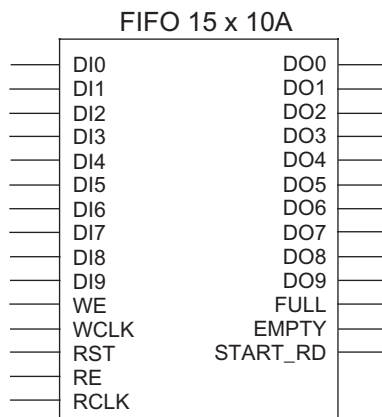
### FIFO I/O Definitions

<b>DI</b>	Parallel Data Write Port (Data width is 4, 6, 8, 10 depend on sysHSI Macro)
<b>DO</b>	Parallel Data Read Port (Data width is 4, 6, 8, 10 depend on sysHSI Macro)
<b>WRCLK</b>	Write Clock. In SERDES with FIFO Mode, RECCLK is used for Write Clock
<b>WE</b>	Write Enable, In SERDES with FIFO Mode, CDRLOCK is used for Write Enable
<b>RCLK</b>	Read Clock. Same as Input Register Clock
<b>RE</b>	Read Enable. Same as Input Register Clock Enable
<b>START_RD</b>	Start Read Flag. Active Low. This signal is not used in FIFO Only Mode.
<b>FULL</b>	FIFO FULL Flag
<b>EMPTY</b>	FIFO EMPTY Flag
<b>RST</b>	Reset. In SERDES with FIFO Mode, CDRRST is used for RST. Active High.

The RST inputs of the synthesis macro FIFO15X10A and its simulation models are ACTIVE HIGH, while the silicon is ACTIVE LOW when used with sysHSI Block. The RST input at the pin must be inverted as shown in the example below.

## Macro Symbol Drawing

Figure 22. FIFO Macro Symbol



## FIFO Instantiation in Verilog

```
FIFO15X10A I1 (.DI0(DI0), .DI1(DI1), .DI2(DI2), .DI3(DI3), .DI4(DI4),
               .DI5(DI5), .DI6(DI6), .DI7(DI7), .DI8(DI8), .DI9(DI9),
               .WE(WE), .WCLK(WCLK), .RST(!RST), .RE(RE), .RCLK(RCLK),
               .DO0(DO0), .DO1(DO1), .DO2(DO2), .DO3(DO3), .DO4(DO4),
               .DO5(DO5), .DO6(DO6), .DO7(DO7), .DO8(DO8), .DO9(DO9),
               .FULL(FULL), .EMPTY(EMPTY), .START_RD(START_RD));
```

## sysHSI Block Usage in Source-Synchronous Mode: ispGDX2 Family 64 to 256

Table 13. sysHSI Block Usage Guide Map

Number of Transmitter Channels in Each Group				Package Type					
				256		128		64	
				Group 1	Group 2	Group 1	Group 2	Group 1	Group 2
7, 8	5, 6	3, 4	1, 2	<i>2<sup>2</sup></i>	<i>4</i>	<i>0</i>	<i>2</i>	<i>0</i>	<i>1</i>
				<i>3<sup>1</sup></i>	<i>5</i>	<i>1</i>	<i>3</i>		
				<i>1</i>	<i>7</i>				
				<i>0</i>	<i>6</i>				

1. sysHSI Block Number (see Figure 19).

2. Only the sysHSI blocks in *italic* generate SS\_CLKOUT. They must be used in macros TX\_SS\_4, 6, 8. Other sysHSI blocks in the table are recommended to minimize channel-to-channel skew.

## ispGDX2-256 Clock Tree

Table 14. GD<sub>X</sub>256 Clock Tree

sysHSI BLOCK<0:7>	REFCLK0	REFCLK1	REFCLK2	REFCLK3
sysHSI Block<0:7>	GCLK0	GCLK1	GCLK2	GCLK3

## sysHSI I/O Cell Data Path

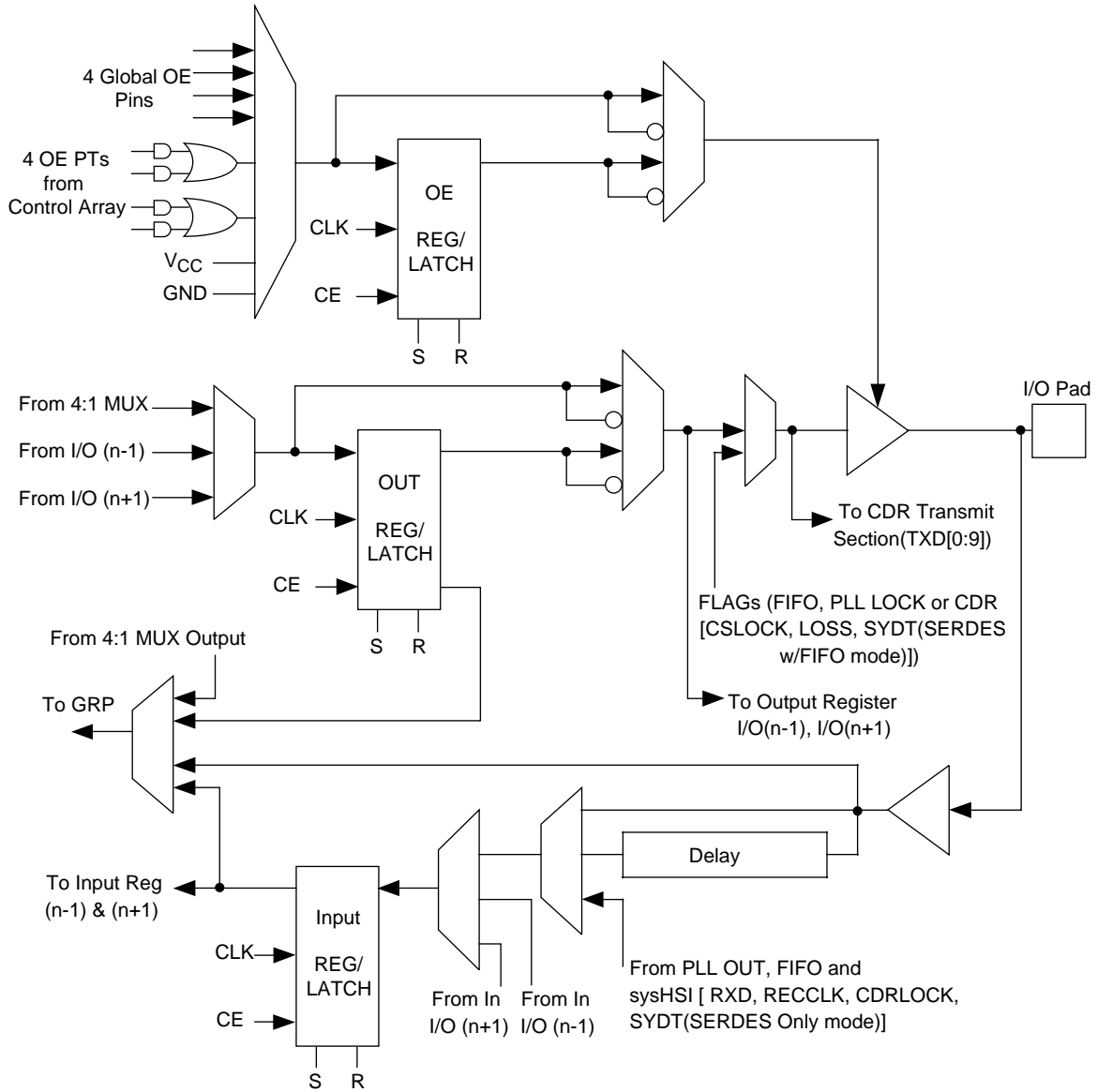
Figure 23 shows the path for three registers associated with the I/O cell. The signals, PLL Clock output, FIFO output, sysHSI Flags (CDRLOCK, SYDT), sysHSI-received deserialized data output (RXD0..RXD9), recovered clock (RECCLK) can be routed to GRP via the input register of the I/O cell.

The PLL Flag (LOCK, FIFO Flags (FULL, EMPTY), or sysHSI Flags [CSLOCK, SYDT(SERDES with FIFO), LOSS] are routed to the I/O pin via the output MUX associated with the output register of the I/O cell.

The Symbol Detect (SYDT) signal is routed to the input register when the sysHSI Block is used as “SERDES with FIFO Mode.” Otherwise it is routed to the I/O pin.

If any SYDT, LOSS or CSLOCK signals are unused in the design, the pad is available for other signal outputs.

Figure 23. ispGDX2 I/O Cell Diagram



The CDRRST inputs of the synthesis macro CDRX\_10B12B, CDRX\_8B10B and their simulation models are ACTIVE HIGH, while the silicon is ACTIVE LOW. The CDRRST signal at the pin must be inverted as shown in the examples below.

### CDRX\_10B12B Instantiation in Verilog

```

CDRX_10B12B I1 (.SIN(SIN), .REFCLK(REFCLK), .CDRRST(!CDRRST), .RXD0(RXD0),
    .RXD1(RXD1), .RXD2(RXD2), .RXD3(RXD3), .RXD4(RXD4), .RXD5(RXD5),
    .RXD6(RXD6), .RXD7(RXD7), .RXD8(RXD8), .RXD9(RXD9), .RECCLK(RECCLK),
    .CSLOCK(CSLOCK), .CDRLOCK(CDRLOCK), .LOSS(LOSS), .SYDT(SYDT));
    
```

## CDRX\_8B10B Instantiation in Verilog

```
CDRX_8B10B I1 (.SIN(SIN), .REFCLK(RECLK), .CDRRST(!CDRRST), .EXLOSS(EXLOSS), .RXD0
(RXD0), .RXD1(RXD1), .RXD2(RXD2), .RXD3(RXD3), .RXD4(RXD4), .RXD5(RXD5),
.RXD6(RXD6), .RXD7(RXD7), .RXD8(RXD8), .RXD9(RXD9), .RECCLK(RECCLK),
.CSLOCK(CSLOCK), .CDRLOCK(CDRLOCK), .LOSS(LOSS), .SYDT(SYDT));
```

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-408-826-6002 (Outside North America)

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)