Introduction to the sysHSI™ Block ispXPGA™ and ispGDX2™

April 2003 Technical Note

Introduction

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 and Data Recovery (CDR) circuit. Source Synchronous mode provides another way of achieving high speed data rate without embedding the clock.

Lattice provides sysHSI blocks on ispXPGA and ispGDX2 device families to support both embedded clock and source synchronous clocking applications. This document provides an introduction to the sysHSI Blocks. Refer to Technical Note TN1020 for detailed description.

Modes of Operation

The sysHSI block supports number of different modes. In Clock Data Recovery (CDR) mode, clock is encoded in the transmit data stream and CDR recovers this clock from the incoming data. In Source Synchronous Mode, 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[1] (De-skew)	De-skew (optional)

1. In Source-Synchronous mode, only De-skew mode requires symbol alignment.

CDR Mode

In CDR mode clock is encoded 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.

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 (8B/10B: Encoding and Decoding is not included)
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 2 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 are implementing 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. These interfaces can operate in two modes.

Normal Mode SS 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.

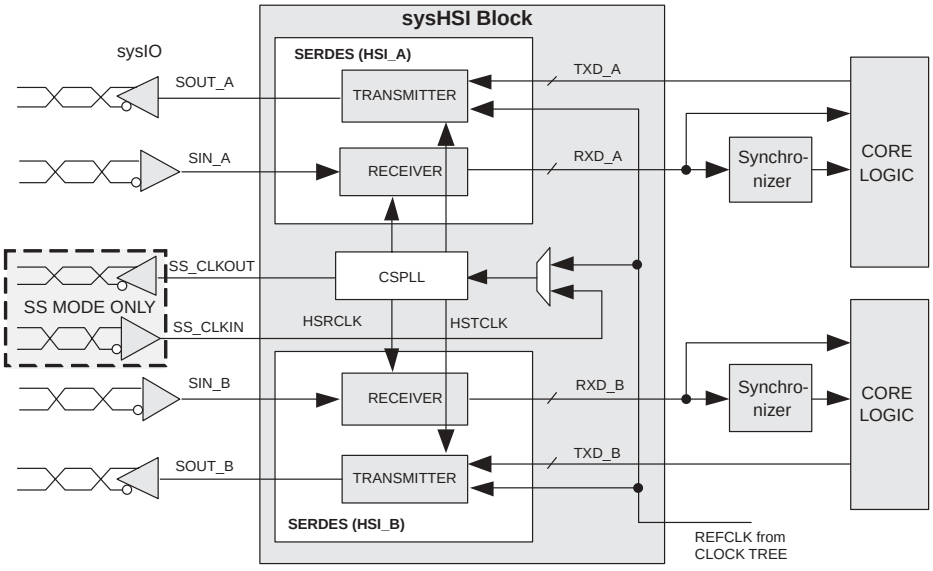
Normal Mode with Phase Adjustment:
Optional Phase Adjustment is provided in this mode. The known skew adjustment phase value can be programmed by user.

De-Skew SS 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 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 solution to this problem is to use a synchronizer. In systems where frequency deviation is not a problem this synchronizer can be bypassed.

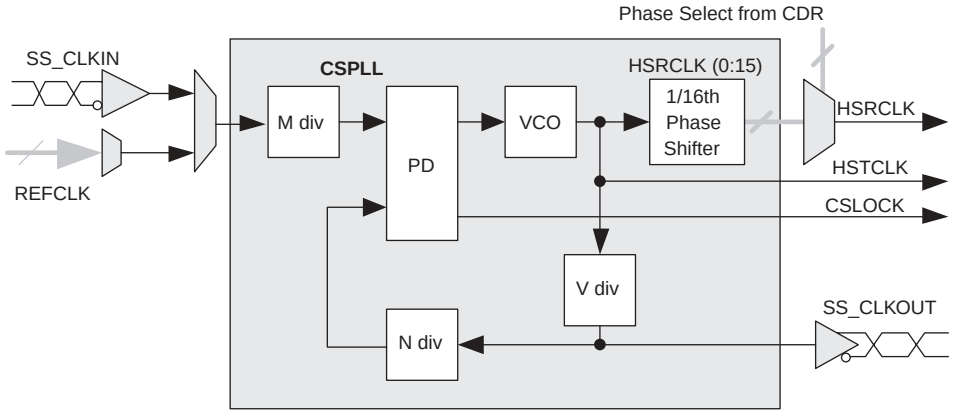
CSPLL: Clock Synthesizer PLL

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 LVDS clock input, SS_CLKIN. Also, there are 4 choices for the internal clock to increase the flexibility.

\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 transmitter, the HSTCLK (High Speed Transmit Clock) is generated from REFCLK multiplied by 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, generate 16- phase high speed Clocks, HSRCLK<0:15>(High Speed Receive Clock).

Figure 2. CSPLL

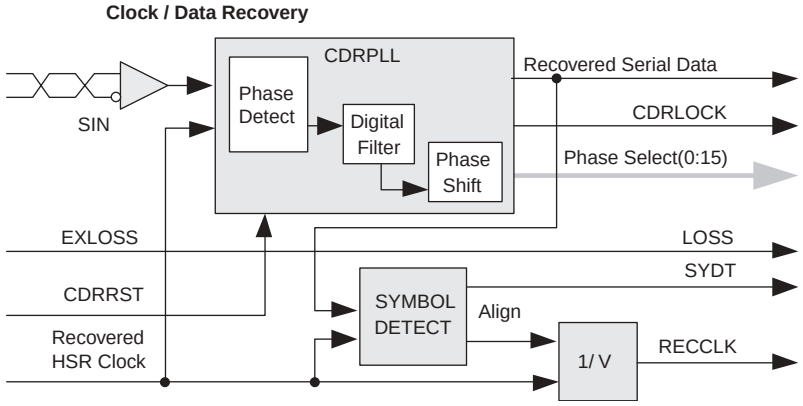


Clock and Data Recovery

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 synchronizer or core logic.

Figure 3. Clock and Data Recovery Block

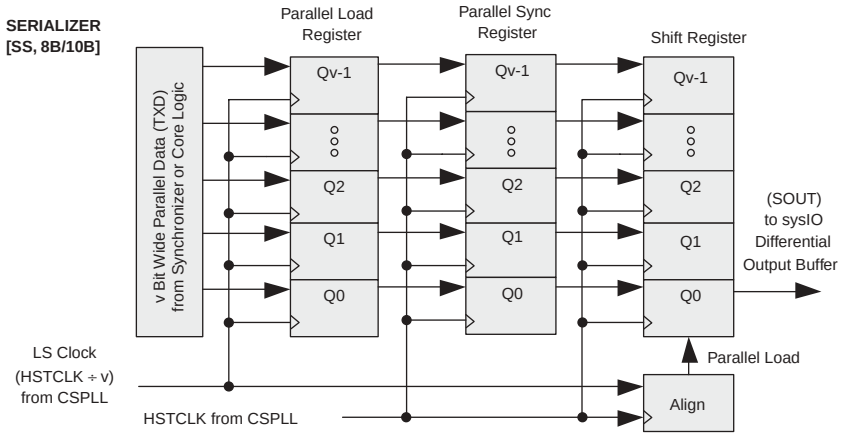


Serializer / De-Serializer(SER/DES)

Serializer

Transmitter receives low speed parallel Data, TXD, from the Synchronizer or Core Logic. TXD data is clocked by REFCLK from clock tree (or SS_CLKIN in SS mode). The CSPLL multiplies REFCLK by factor of v to generate HSTCLK. The Transmitter converts the low speed parallel Data, TXD, into high speed Serial Data Stream, SOUT, that is running at HSTCLK. The alignment circuit synchronizes REFCLK and HSTCLK with edge detect circuit to align SOUT with HSTCLK.

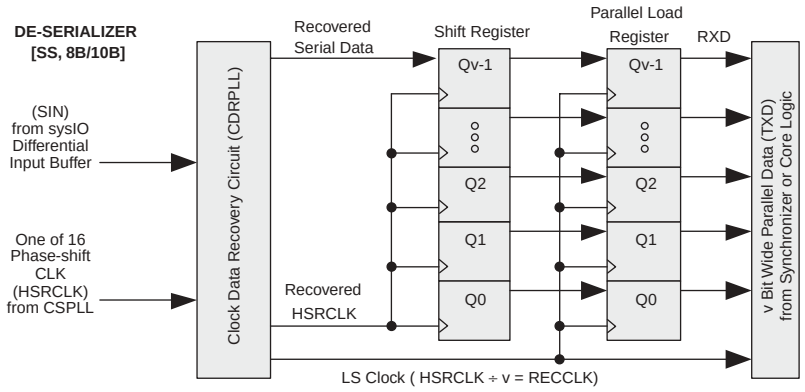
Figure 4. Serializer [SS, 8B/10B]



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

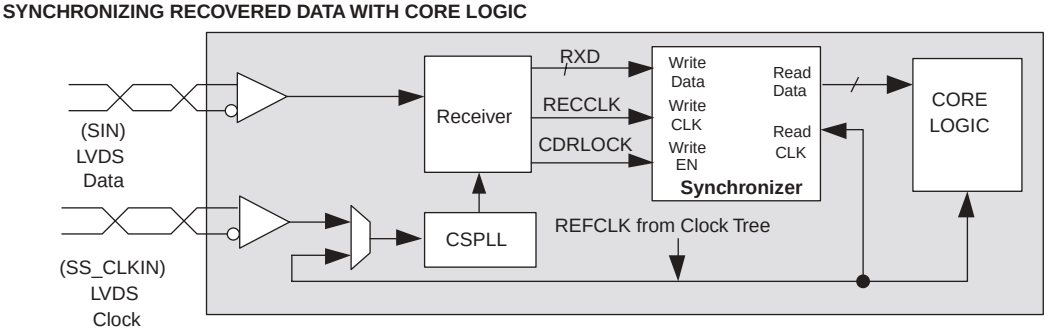


Synchronizer

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

Parallel Transmit Data enters Transmitter of sysHSI block from core logic clocked by REFCLK. The REFCLK at the same time is fed to CSPLL to generate high speed clock to transmit serialized data (HSTCLK). In the Transmitter, the REFCLK is re-aligned by high speed clock to generate parallel load clock to the Serializer shift register. If the skew between REFCLK and 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 without synchronizer. Figure 6 shows the synchronizer interface between core logic and receiver.

Figure 6. Synchronizer



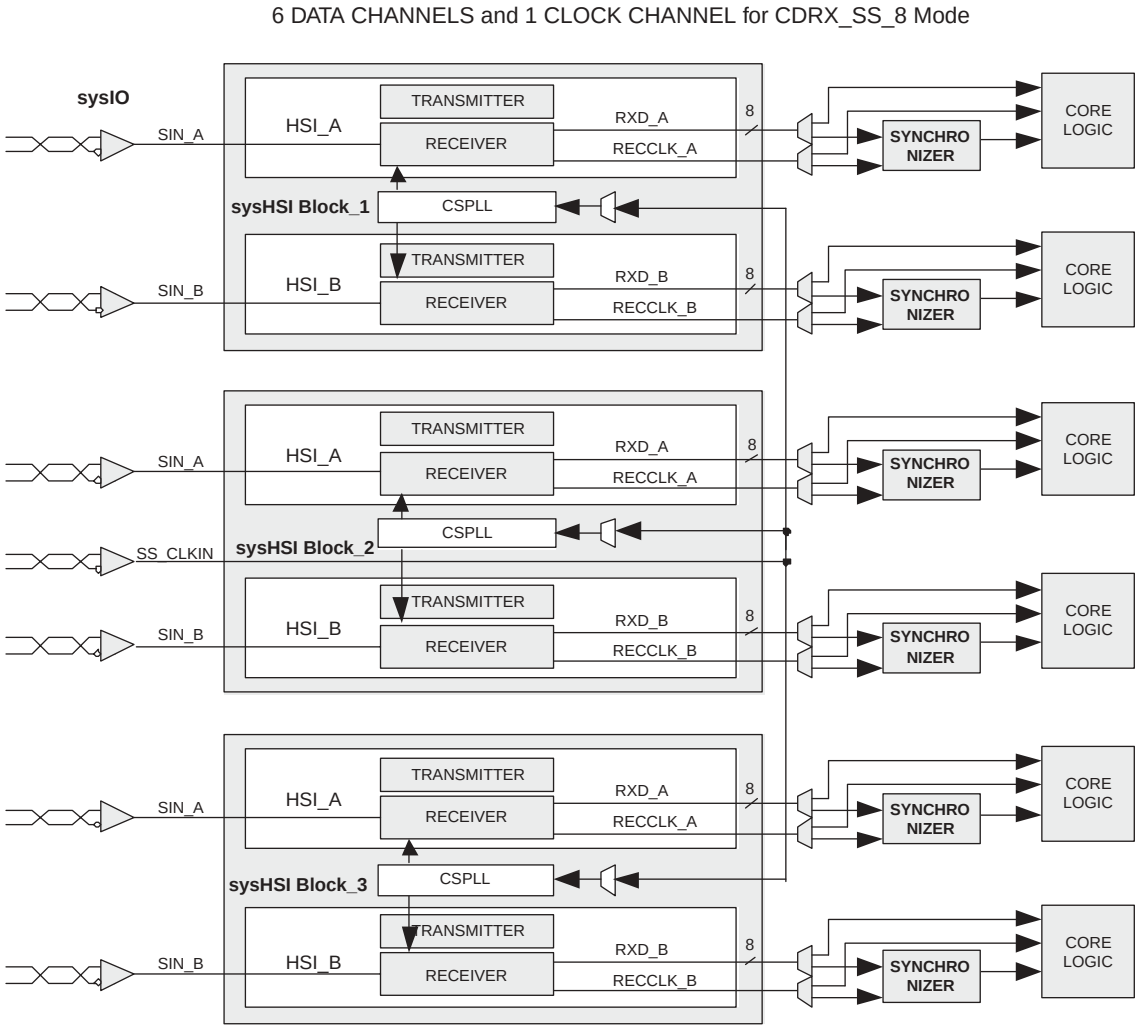
sysHSI Block and Source-Synchronous Mode with Multiple Data Channels

Each chip includes 2 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. 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.

An example of Source-Synchronous Mode Block diagram is shown in Figure 7. Figure 7 illustrates how the HSI circuit is implemented in Source-Synchronous Receiver Mode.

Figure 7. Source-Synchronous Mode Example Diagram (CDRX_SS_8)



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 user must take phase jitter from different clock sources into consideration. This jitter increase may 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 as 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.

Using sysHSI Blocks in Design Tools

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[1]	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 Usage with HDLs

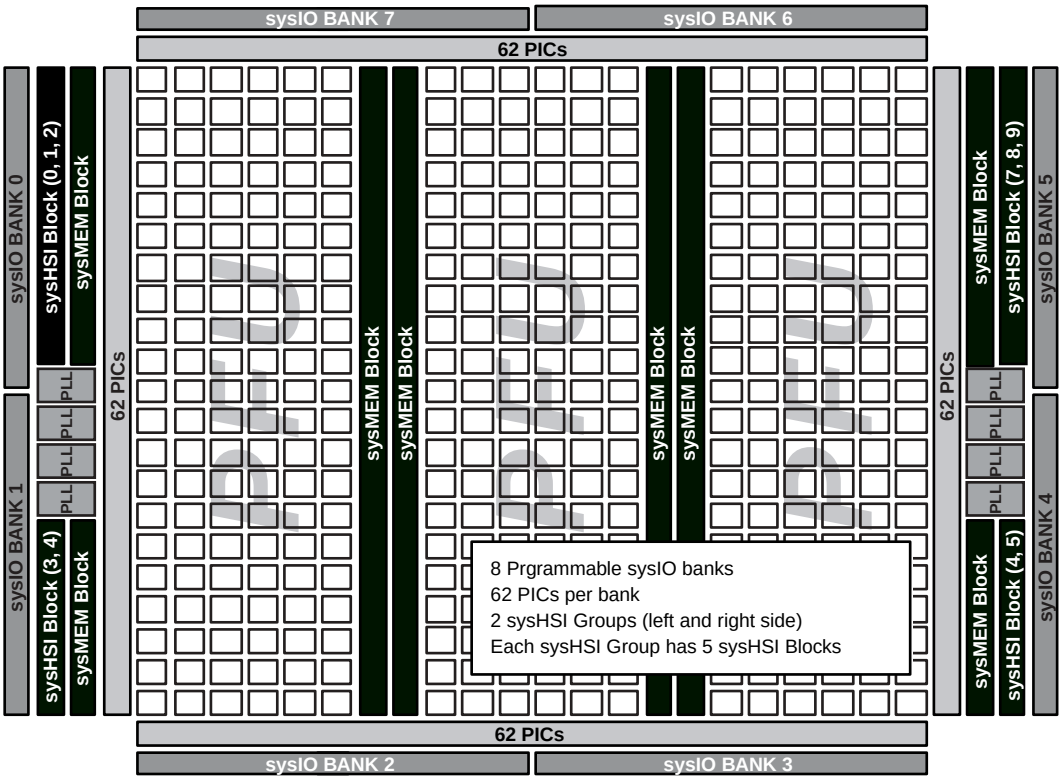
Synthesis tools such as Synplicity and Exemplar "black-box" the VHDL and Verilog instantiations and pass them through an EDIF netlist to the Lattice software. The Lattice software converts the "black-box" into the physical representation of the sysHSI within the device using the macros defined above.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

ispXPGA Family

The block diagram of LFX1200 is shown in Figure 8.

Figure 8. ispXPGA-1200 Block Diagram



sysMEM™ Embedded RAM (EBR) Usage as FIFO

spXPGA Family includes sysMEM Embedded Block RAM that can be programmed as a FIFO for synchronization. For macros available for the EBR, please refer to Lattice technical note number TN1028, ispXPGA sysMEM Memory Design and Usage Guidelines

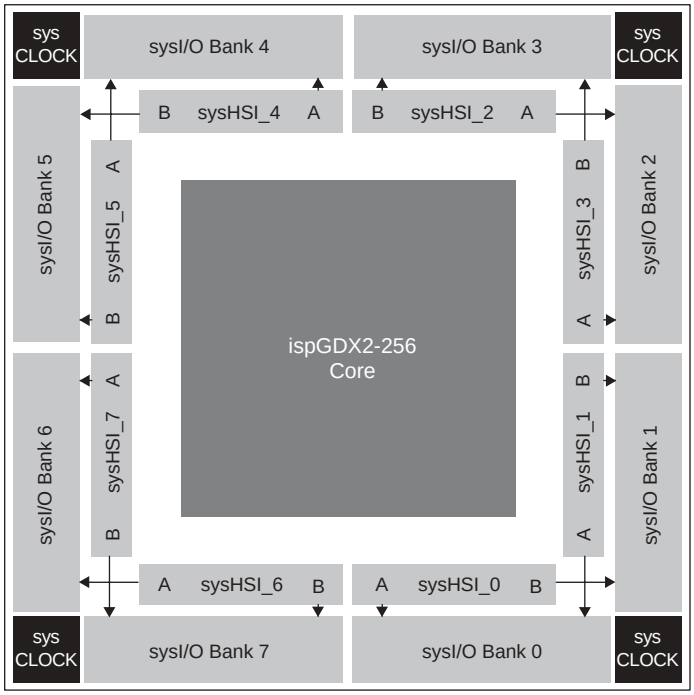
ispGDX2 Family

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.

The ispGDX2-256 has 8 sysHSI Blocks. Each sysHSI Block is divided to two SERDES blocks, HSI_A and HSI_B. Each SERDES block occupies 16 IO Cell Block in the sysIO Bank.

Figure 9. 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 10. sysHSI Block interface with FIFO in ispGDX2