XAPP289 (v1.1) August 15, 2001

Common Switch Interface CSIX-L1 Reference Design

Author: Markus Adhiwiyogo

Summary

This application note describes a Virtex™-II device implementation of a CSIX-L1 common switch interface between a network processor's traffic manager and the switching fabric for ATM, IP, MPLS, Ethernet, and similar data communications applications. This design uses a pipeline implementation to achieve a low clock period (approximately166 MHz), and uses the 32-bit interface CSIX scheme.

Introduction

CSIX-L1 is a Common Switch Interface specification managed by the Network Processing Forum. It defines a standard interface for transferring information between traffic managers (network processors) and the physical switching fabric. Figure 1 is an example of a 32-bit CSIX interface.

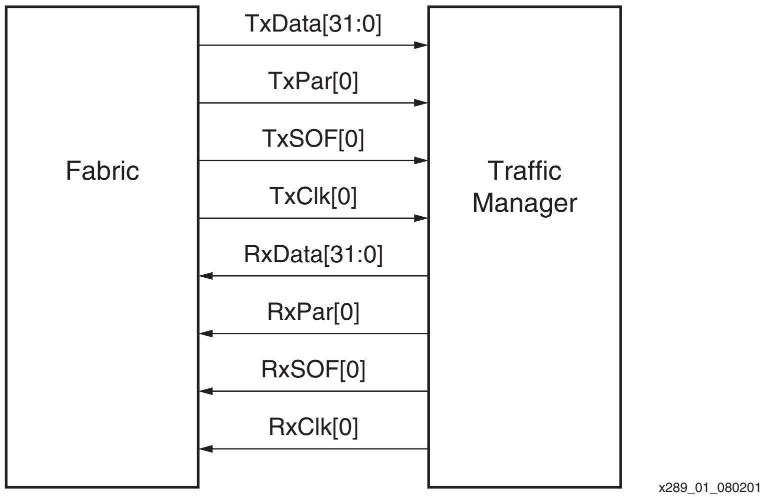

Figure 1: **32-bit CSIX Interface**

Table 1: **32-bit CSIX Interface Signal Functions**

Signal	Direction	Function
RxData[31:0]	TM to Fabric	Receive Data
RxPar[0]	TM to Fabric	Receive Data Odd Parity
RxSOF[0]	TM to Fabric	Receive Start of Frame
RxClk[0]	TM to Fabric	Receive Clock
TxData[31:0]	Fabric to TM	Transmit Data
TxPar[0]	Fabric to TM	Transmit Data Odd Parity
TxSOF[0]	Fabric to TM	Transmit Start of Frame
TxClk[0]	Fabric to TM	Transmit Clock

A CFrame is the base information unit transferred between the traffic managers and the CSIX fabric. A CFrame consists of a header, payload, and a vertical parity trailer. The frame format is found in the CSIX-L1 documentation at **http://www.csix.org/csix_drafts.htm**.

The CFrame header contains the information fields controlling the behavior of the traffic manager and CSIX fabric interface. The CFrame header contains a base header and an extension header. The CSIX-L1 standard determines the format and values of the CFrame header. The variable length payload is passed by the CSIX fabric from the transmitting to the receiving entity. Currently the format for the 32-bit interface is simply four bytes of data. The CFrame supports up to 256 bytes of data. Multiple CFrames are used when more than 256 bytes of data are needed. The vertical parity trailer is used for error detection at the CSIX-L1 layer. Combining the header, payload, and vertical parity trailer yields a maximum CFrame size of 264 bytes.

FPGA Implementation Summary

Design Features

- Pipeline data path designed to maximize speed (~166 MHz).
- 44-bit interface/data bus delivers an instruction in one clock cycle.
- 32-bit Tx/Rx CSIX buses used to minimize added pin requirements.
- Ability to transmit and receive at the same time.
- CSIX receive side clocked relative to the transmit side ensuring an appropriate data rate.
- In addition to horizontal parity, by including vertical parity allows the appropriate corrections during erroneous transmission without additional clock cycles.
- The reference design is optimized for Virtex-II devices. The architecture is not specific allowing code portability to other Virtex FPGA devices.
- Linearly expandable for 64-bit, 96-bit, and 128-bit bus width

The verilog reference design available on the Xilinx ftp site (**ftp://ftp.xilinx.com/pub/applications/xapp/xapp289.zip**) contains a 32-bit CSIX interface implementation including both receive (Rx) and transmit (Tx) modules. This implementation supports all operations defined by the Network Processor Forums CSIX group as of August 2001.

The design utilizes heavily pipelined data to achieve a clock cycle of approximately 166 MHz. With further relative location constraints, a 200 MHz implementation can be realized. The reference design uses 360 slices and 166 IOBs.

Simply by instantiating multiple CSIX modules, this reference design can generate 64-bit, 96-bit, and 128-bit bus width modules.

Figure 2 shows a block diagram and pin description of the top-level CSIX module. Table 2 describes the signals.

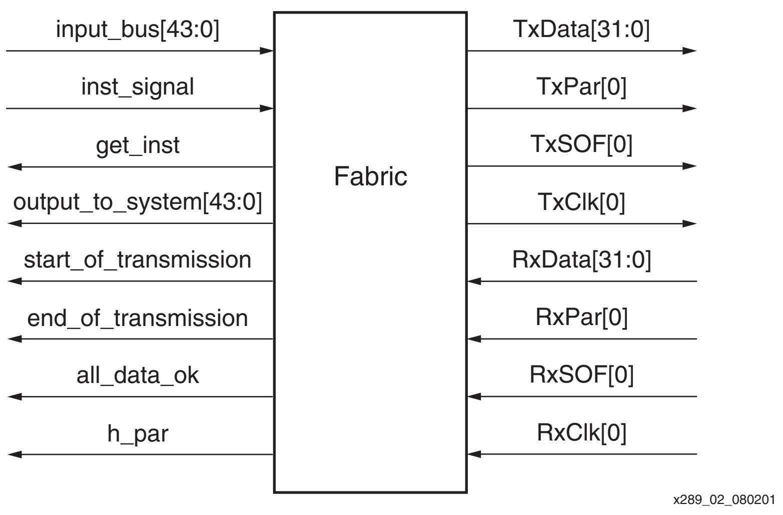

Figure 2: **Top-Level CSIX Module**

Table 2: **Top-Level CSIX Module Signal Description**

Signal	Bit Width	Description
Input_bus[43:0]	44	Unformatted CSIX data or an information bus for the CSIX to transmit.
Inst_signal	1	Determines whether the input_bus signals are data or information.
Get_inst	1	Tells the CSIX master when to put the instruction on the input_bus line.
Output_to_system[43:0]	44	Processed data or information bus received by CSIX.
Start_of_transmission	1	Determines if the current signal on output_to_system is the start or not.
End_of_transmission	1	Determines if all the data is received or not.
All_data_ok	1	Vertical Parity Check
H_par	1	Horizontal Parity Check
RxData[31:0]	32	Receive Data
RxPar[0]	1	Receive Data Odd Parity
RxSOF[0]	1	Receive Start of Frame
RxClk[0]	1	Receive Clock
TxData[31:0]	32	Transmit Data
TxPar[0]	1	Transmit Data Odd Parity
TxSOF[0]	1	Transmit Start of Frame
TxClk[0]	1	Transmit Clock

By implementing a network processor or other communications device and connecting it directly in the FPGA, the number of IOBs used for the CSIX alone can be reduced to 70 IOBs. The block diagram in Figure 3 illustrates an implementation using a network processor.

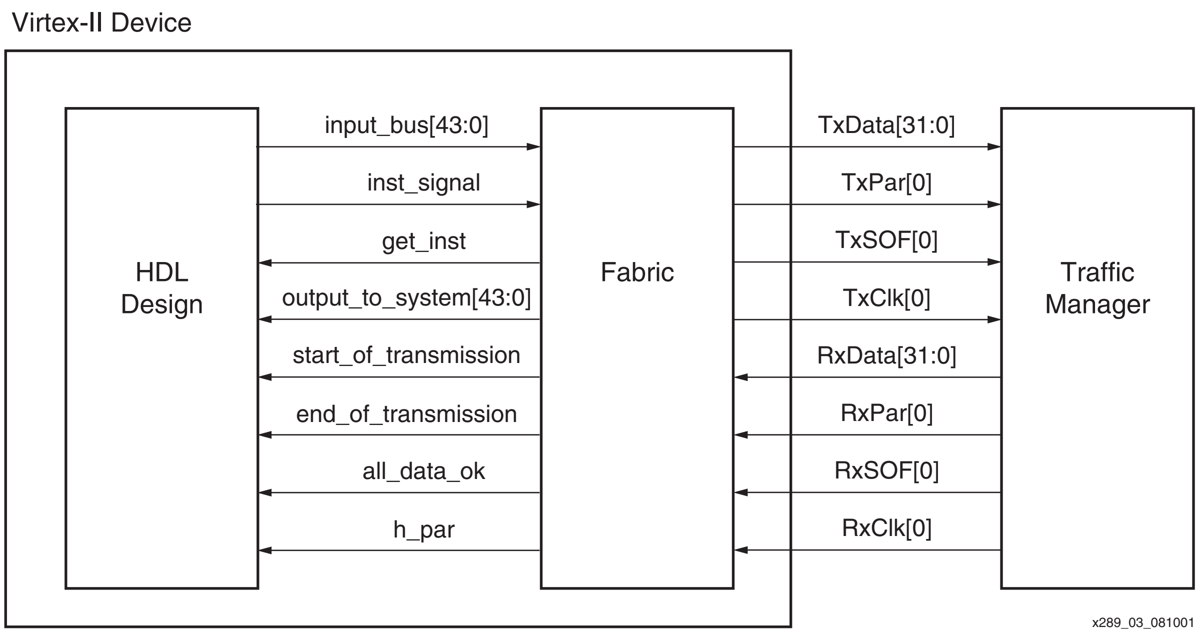

Figure 3: **Implementation of a CSIX with a Network Processor**

FPGA Implementation Information

The CSIX implementation in this reference design takes in data from the input_bus signal and formats the data into the CSIX format. The Table 3 shows all the possible input_bus bit configurations for the different operations. The content of each bit in the input_bus is also described in Table 3. The numbers given are the bit-length for each entry. The left-most entry signifies bit 44, while right-most entry signifies bit 0. For a detailed description of each instruction, refer to the CSIX standards documentation.

Table 3: **Functional Description**

Instruction	Type	Payload Length	Class	Flow Control	C	P	Speed	Destination Address	Bitmask Header	Bitmap	ID	Don't Cares	Data
Idle	4	8										32	
Unicast Operations	4	8	8					12				12	
Multicast M.	4	8	8						8	16			
Multicast ID	4	8	8								22	2	
Multicast Bin	4	8	8					Left:12 Right:12					
Broadcast	4	8	8									24	
Flow Control	4	8	8	2	1	1	4	12				4	
Data												12	32

On the receiving side, the CSIX implementation receives CFrames and converting them into the 44-bit format described in Table 3. This received 44-bit format is available on the output_to_system bus.

FPGA Method of Operation and Waveforms

Transmit Signals

Information on transmitted data (components of the header bits) is placed on the input_bus line shortly after the get_inst bus is asserted High. The inst_signal line has to be asserted High by the designer to indicate data on the input_bus line contains the information bits. The information bits are then followed by the number of data bytes required for transmission.

Four clock cycles after the information bits are placed on the input_bus line, the first CSIX frame appears as TxData. The beginning of the CSIX frame is accompanied by asserting TxSOF High to indicate the start of the CSIX frame transmission. TxSOF can also be High during an idle CSIX frame. The receive mechanism is able to distinguish between an idle and a regular CSIX frame.

Since TxPar is a parity check of the data being transmitted, it can either be High or Low.

TxClk is simply the clk (clock) signal being transmitted to the receive side of CSIX to indicate the data rate. The waveforms in Figure 4 illustrate the transmit signals.

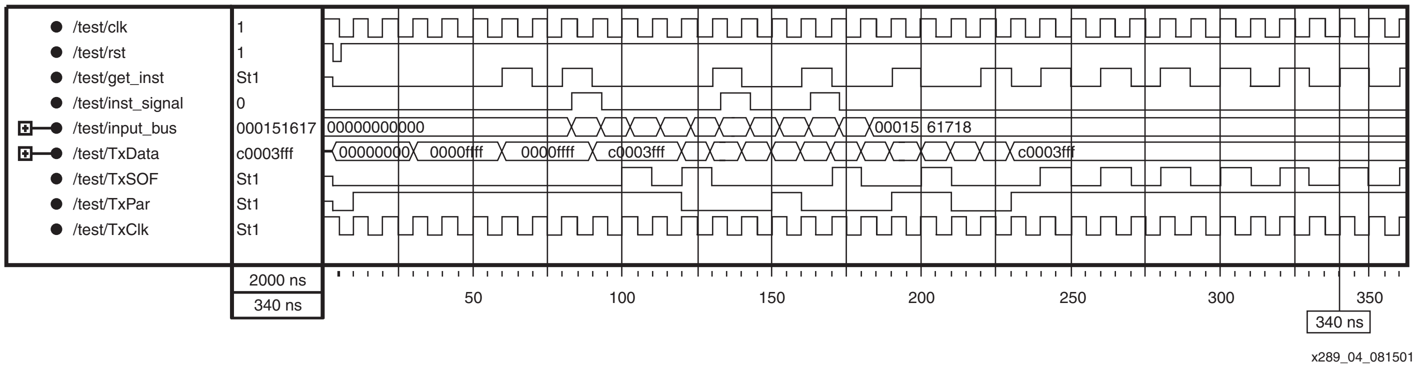

Figure 4: **Transmit Signals**

Receive Signals

The first CSIX frame is placed on the RxData by the transmitting CSIX fabric when RxSOF is asserted High. It is then followed by the required data received, with RxSOF asserted Low. RxSOF is also asserted High whenever idle CSIX frames are received. The receiving mechanism handles the determination between idle and data frames.

The first data going from the CSIX fabric to the receiving entity appears five clock cycles after the first CSIX frame is placed in RxData on the output_to_system bus. The first data is always information bits. The start_of_received bus is asserted High indicating that the information bits are the current data on the output_to_system bus.

To indicate reception of all the data, the end_of_transmission (eot) bus is asserted High one clock cycle after the last data is received. Data receive is completed whenever the start_of_received is asserted High for one clock cycle, followed by the end_of_transmission signal asserted High for one clock cycle. When asserting High the start_of_received signal for one clock cycle, then Low for one clock cycle, the High for one clock cycle does not translate to data receive, but idle.

The h_par signal is used to indicate whether there are any errors on the current data on output_to_system bus. Whenever h_par is asserted Low, there is an error on the current output_to_system bus. The h_par signal is generated by comparing RxPar and a horizontal parity of the CSIX frame received.

The all_data_ok (ado) signal is asserted High the same time the eot signal is asserted High. The ado signal is used as a vertical parity check looking for errors in any of the received frames.

The clock used is RxClk. RxCLK determines the data rate received from the transmitting CSIX fabric. The waveforms in Figure 4 illustrate the receive signals.

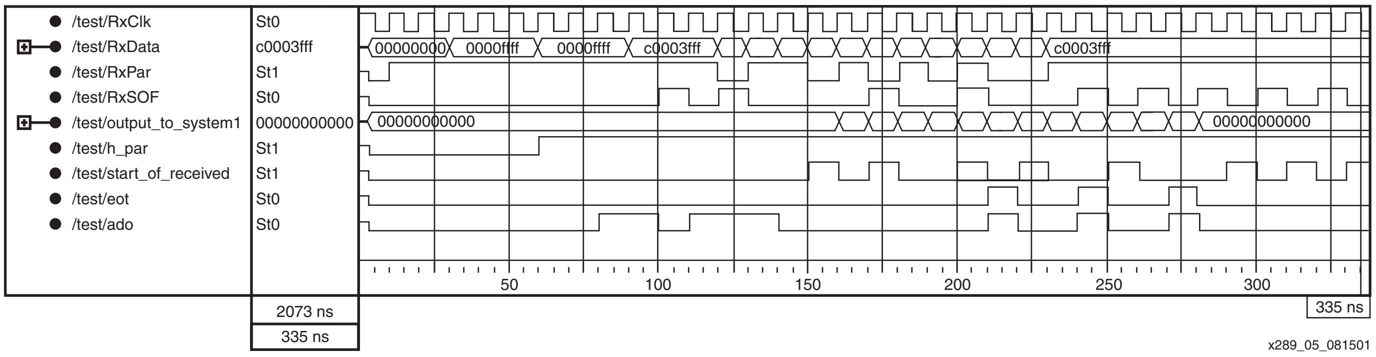

Figure 5: **Receive Signals**

Conclusion

This CSIX implementation is an ideal addition to a Network Processor or Traffic Manager implementation on a Virtex device, as the slice count is very small. A fast (~166MHz) CSIX interface implemented in an FPGA is a cost saving alternative to an ASIC solution.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
08/10/01	1.0	Initial Xilinx release.
08/15/01	1.1	Corrected ftp site link, updated Figure 4 and Figure 5.