INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 2, ISSUE 11, NOVEMBER 2013 ISSN 2277-8616

Serial Front Panel Data Port (SFPDP) Protocol Implementation In Xilinx Fpgas

M. Vanaja, P. Prasanth Babu.

Abstract: The Serial Front Panel Data Port (SFPDP) protocol for high speed data transfer presents in this paper. High-speed data transfer finds application in most modern day communication systems. This design has been mainly done for data transfer in radar systems but can be programmed and used for variety of applications involving high-speed data transfer. The design follows a systematic approach with design of SFPDP protocol and implementation on FPGA and explains all these stages of design in detail. The design can be programmed to work at different speeds as required by different systems and thus can be used in variety of systems involving high-speed data transfers. The efficient use of customized IP cores and resources of FPGA delivers high level of performance and area efficiency.

Index Terms: FPDP (Front Panel Data Port), SFPDP (Serial Front Panel Data Port), XAUI (Extended Attachment Unit Interface), FPGA (Field Programmable Gate Array), SOF (Start of Frame), SEOF (Status End Of Frame), FEOF (Frame End Of Frame), MEOF (Mark End Of Frame).

1 INTRODUCTION

Parallel Front Panel Data Port (FPDP) is a 32-bit parallel synchronous bus intended to provide high-speed data transfer between FPDP connections at speeds up to 160MB/s. Although widely used for high-speed communications within a single chassis or rack of equipment, one of the limitations of Parallel FPDP is that link connections are limited to relatively short distances – usually 1 to 3 meters. Serial FPDP, originally developed by Systran Corporation in its Simplex Link and Fiber Extreme products, is defined in the VITA 17.1 specification. It is a serial encapsulation of the Front Panel Data Port (FPDP) protocol (VITA 17). Serial Front Panel Data Port (SFPDP) is a high-speed, low-latency, data- streaming, serial communications protocol used for high-speed real-time data transfer applications. It is defined to operate at three distinct link speeds, 1.0625 Gbaud, 2.125 Gbaud, and 2.5Gbaud – with sustained data rates in excess of 105, 208, and 245 Mbytes/sec. respectively. Serial FPDP can also operate over long distances (up to 10 kilometers) using fiber optic cable.

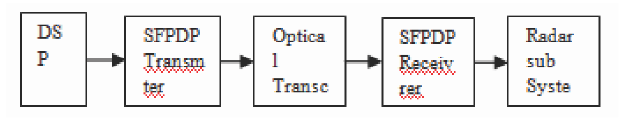


Figure 1 – Block diagram of typical Application of SFPDP Protocol

The typical application of SFPDP Protocol in the Radar Systems is as shown in figure1.Processed data from signal processor is acquired by the interface module and then transferred using this protocol as implemented in the FPGA. This module provides interface to Digital Signal Processors (DSP) for acquiring the processed data. Interface module receives processed data from DSP processors on parallel link and serializes the data of each channel and sends it over the XAUI following the SFPDP protocol. This can be used to send the data through SFPDP protocol to a distance extending up to 10KM for presentation on display terminal for radar systems. Field data can be recorded & replayed as and when required to further analyze the data in the control room to test the performance of the radar. The SFPDP data is sent through XAUI (extended Attachment Unit Interface), the serial interface. The data in the XAUI is looped back using fiber optic cable.

2. SERIAL FRONT PANEL DATA PORT (SFPDP)

Serial FPDP extends the maximum distance of FPDP connections by serializing the FPDP data stream and transmitting it over extended distance using fiber optic cable. Serial FPDP is basically a point-to-point, simplex protocol designed to transfer data from a sender to a receiver. The connection between a sender and a receiver is established and remains in effect for relatively lengthy periods of time. The proposed protocol contains Sync with no data, sync without data frame, and sync with data frame and normal data fiber frame. Our design uses Normal data frame and Sync with no data Frames Each Normal data frame contains Idle frame (IDLE), Start of frame (SOF), Data, Status End Of Frame (SEOF), Frame End Of Frame (FEOF) and Go/Stop Frame. Data frame varies from 0 to 512 words. Fig.2 shows the frame format. Sync with no data frame contains idle frame, SOF, Mark End of Frame (MEOF), SEOF and Go/stop frame shown In figure 3. For sending data & sync, protocol design uses t 2505 Sequence of format is as given in following section. Each Frame is recognized by a specified 32 bit pattern as given below in hex:

III. SOF	: BCB51717
IV. FEOF	: BC8A9595
V. SEOF	: BC957575
VI. Go/Stop	: BC85B5B5
VII. MEOF	: BC8AD5D5
VIII. IDLE	: BC95B5B5

IDLE	SOF	DATA	CRC	FEOF	SEOF	GO/STOP

Figure 2 – Normal Data Fiber Frame

IDLE	SOF	NODATA	CRC	FEOF	SEOF	GO/STOP

Figure 3 – Sync without Data Fiber Frame

The sequence of serialization of data from all data link is explained in this section. After sending the Sync Frame, the data transfer starts on the first channel. The data is read one word at a time, each word is sent with 6 frames. This design logic is followed to allow sufficient time for data to be buffered in the other link FIFOs in the mean time. If the FIFO is empty, IDLE frames are sent to MGT. After a specified time out waiting for data, the control goes to the next link. After all N word data on the first link is transferred and after sending Nth data, EOF frame data transfer starts on the next link. In this time period since sufficient data (greater than 512, the maximum data length in a frame) is buffered, data is read in blocks of 500 words and made into one normal data frame. This continues till the N words of data are transferred from particular link. Thus the overhead due to formation of frames is reduced. Same process is followed in the subsequent link FIFOs. After the data transfer is over for the final link, the control goes back to the first link & the process repeats as said above till the next CPI.

3. EXTENDED ATTACHMENT UNIT INTERFACE (XAUI)

The eXtended Attachment Unit Interface (XAUI) core is a High-performance; low pin count 10-Gbps interface intended to allow physical separation between data-link layer and physical layer devices in a 10-Gigabit Ethernet system. The Virtex-5, Virtex-4, and Vertex-II Pro FPGA families in combination with the XAUI core, enable the design of XAUI-based Interconnects whether they are chip-to-chip, over backplanes, or connected to 10 Gigabit optical modules. XAUI is a four-lane, 3.125 Gbps-per-lane serial interfaces. Each lane is a differential pair carrying current mode logic (CML) signaling, and the data on each lane is 8B/10B encoded before transmission. Special code groups are used to allow each lane to synchronize at a word boundary and to de-skew all four lanes into alignment at the receiving end.

4. FIELD PROGRAMMABLE GATE ARRAY (FPGA)

‘Field Programmable' means that the FPGA's function is defined by a user's program rather than by the manufacturer of the device. The FPGA is an integrated circuit that contains many (64 to over 10,000) identical logic cells that can be viewed as standard components. The individual cells are interconnected by a matrix of wires and programmable switches. The Interconnects between all the elements was designed to be user programmable. Field programmable Gate Array (FPGA) has multiple slices, which are programmable as digital units like gates. It is used for digital designs using millions of gates in specific ways for a specific application. Application Specific Integrated Circuit (ASIC) is very specific to the design of that specific application which makes it very difficult to modify the existing design after manufacturing. In this work, FPGA is used to make a complex digital design using serial FPDP protocol for high-speed data transfer.

5. DEVELOPMENT SPECIFICATIONS

Tools Used	Xilinx ISE 11.5 for development, ISE simulator for simulation, IP cores
Language Used	VHDL
Hardware Used	Xilinx Virtex-5

6. DESIGN OF SFPDP USING XAUI CORE

A. Generating the XAUI Core

To generate a XAUI (extended Attachment Unit Interface) core with default values using the CORE Generator, the steps are as follows:

1. Start the CORE Generator.
2. Choose File > New Project.
3. Type a directory name.
4. To set the project options:
   - From the Part tab, select a silicon family, part, speed grade, and package that support the XAUI core, for example, Vertex™-II Pro or Virtex-4 FPGAs.
   - From the Generation tab, select VHDL or Verilog; For Vendor, select other.
5. After creating the project, locate the core in the taxonomy tree at the left side of the CORE Generator window. The XAUI core appears under the following categories:
   - Communications & Networking/Ethernet
   - Communications & Networking/Networking
   - Communications & etworking/Telecommunications
6. Double-click the core to open it. The XAUI Customization screen appears.
7. In the Component Name field, enter a name for the core instance.
8. Accept the remaining default options and click Finish to generate the core.

B. Customizing the XAUI Core

The XAUI core can be customized to suit a wide variety of requirements using the CORE Generator software. The XAUI IP Customizer is presented when the user selects the XAUI core in the CORE Generator software. Figure 4 shows the customizer. The left side displays a representative block diagram of the XAUI core as currently configured. The right side consists of user-configurable parameters.

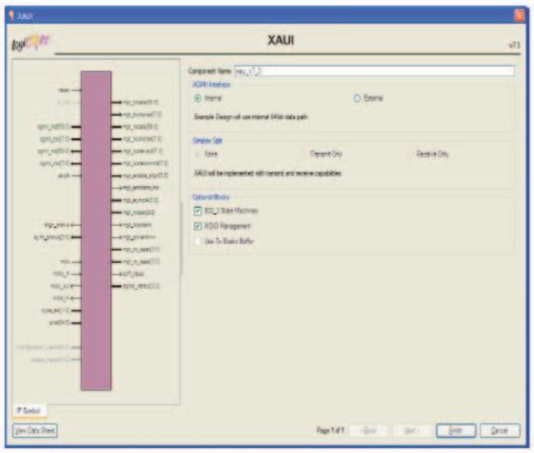


Figure: XAUI customizer

a. **Component Name**
The component name is used as the base name of the output files generated for the core. Names must begin with a letter and must be composed from the following characters: a through z, 0 through 9 and "_" (underscore).

b. XGMII Interface
This control selects between the internal 64-bit client-side interface and the external XGMII client-side interface. The default is the internal 64-bit interface.

C.802_3 State Machines
This control whether the receive synchronization and alignment state machines are implemented as full IEEE 802.3-2005 state machines in the fabric of the FPGA, or using the simplified state machines implemented inside the Rocketed transceivers. The default is to implement the IEEE 802.3-2005 state machines.

d. MDIO Management
Select this option to implement the MDIO interface for managing the core. Deselect the option to remove the MDIO interface and expose a simple bit vector to manage the core. The default is to implement the MDIO interface.

e. Use TX Elastic Buffer
Select this option to implement a clock-correcting elastic buffer in the transmit path of the core. Deselect the option to omit the buffer and have a single clock domain in the transmit path. The default is to omit the transmit elastic buffer.

f. Simplex Split
It is possible to generate cores that implement transmit-only and receive-only functions as well as the regular full-duplex implementation. This is controlled with the Simplex Split option. The default is to have full-duplex transmit and receive operation.

C. Implementation on FPGA
The Serial FPDP frame data is sent through the XAUI interface. The data in the XAUI is looped back using fiber optic cable. This data flow within the designed system is as shown in the figure 5. The hardware device used for implementation is Xilinx Virtex-5. The data transmitted through the transmitter is serialized, encoded, decoded and de- serialized within the core and the output is received at the receiver. The output at the receiver will be of the form as shown in the figure 6.

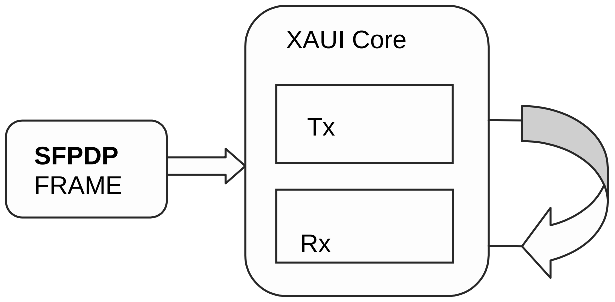


Figure 5. Data flow within the System

The figure 6 shows the format of the output received at the receiver. The received data contains:

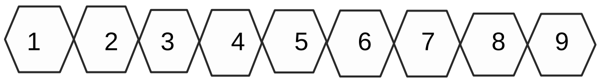


Figure 6. Data received at the receiver

I. XAUI IDLE (64 bit): 0707070707070707
II. XAUI START (64 bit): fbfbfbfbfbfbfbfb
III. SFPDP IDLE (32 bit): 00000000
IV. SFPDP START OF FRAME (SOF): 11223344
V. SFPDP DATA (0-512 words): ffffffff
VI. SFPDP FRAME END OF FRAME (FEOF): 55667788
VII. SFPDP STATUS END OF FRAME (SEOF): 99aabbcc
VIII. SFPDP GO/STOP: ddeeff00
IX. XAUI IDLE: 0707070707070707

7. RESULTS

A. Transmitted Data
SFPDP is developed & simulated in ISE simulator. The transmitted data is received at the receiver end with the speed of 10 Gbps which shows the effectiveness of the proposed (070707070707070,fbfbfbfbfbfbfbfb, 11223344ffffffff,55667 78899aabbcc, ddeeff0007070707)

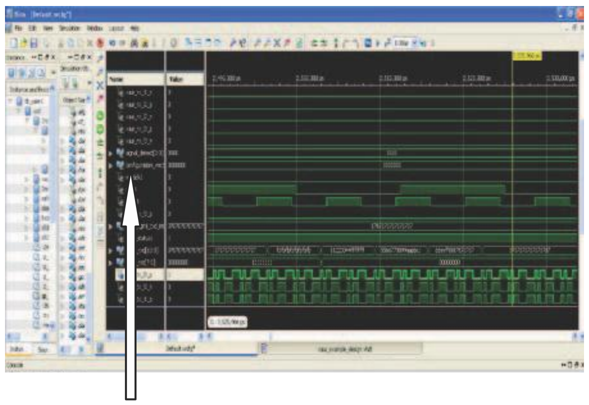


Figure 7. Simulation Results of Transmitted Data

B. Data Stored In the Buffer

The below figure 8 shows the data stored in the buffer. The 32 bit SFPDP frame data transmitted through XAUI core will be stored in the buffer of size 32 bit along with the idle and start bit of XAUI.

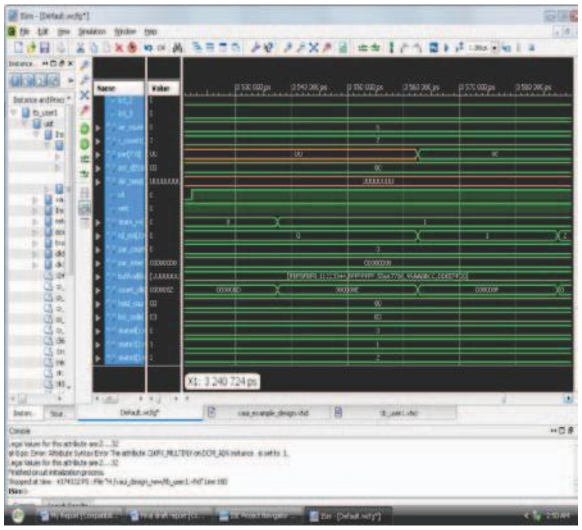
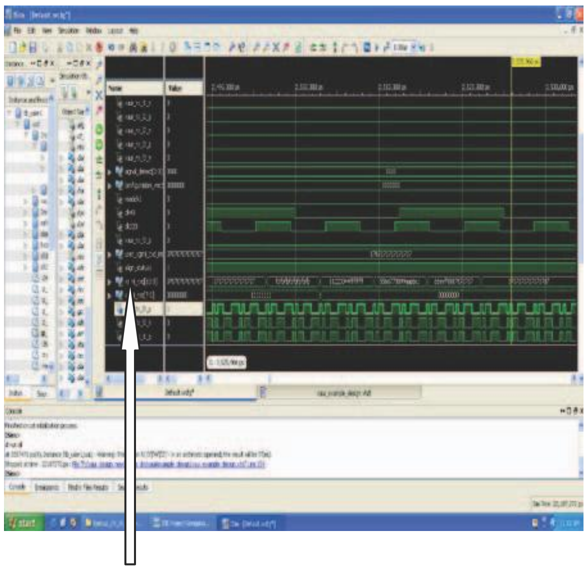


Figure 8. Simulation Results of Data Stored in the Buffer

C. Received Data

The main objective of this work is to transmit the data through SFPDP protocol. The transmitted data is stored in buffer & then received at the receiver. The output at the receiver is of the form as shown in figure 9 which meets the requirement of the designed model. The above results are excellent & justify the designed SFPDP protocol for high speed data transfer. The received data is as same as the transmitted data with the same values.



(070707070707070, fbfbfbfbfbfbfbfb, 11223344ffffffff, 5566778899aabbcc, ddeeff0 007070707)

Figure 9. Simulation Results of Received Data

8. CONCLUSION

Serial FPDP is a protocol which supports high speed data transfer. In this project work it has been realized with FPGA based hardware which does not require any additional interfaces for optical links compared to other hardware realization methods. In this work, XAUI (extended Attachment Unit Interface), the serial interface core is used. XAUI supports data rate of up to 10 Gbps, thus the SFPDP frame data is sent through the interface to achieve maximum data transfer rate. The data/SFPDP frame in the system is looped back using fiber optic cable. This module enables the long distance communication with high data rate through optical link. This work explains the implementation of reliable technique of high-speed data transfer through optical link by using SFPDP protocol implemented in FPGAs. The design can be programmed to work at different speeds as required by different systems and thus can be used in variety of systems involving high speed data transfers.

REFERENCES

[1] Volnei A. Pedroni, ‘Circuit Design with VHDL’, MIT Press, England. [2] Charless H. Roth, Jr (2005) ‘Digital Systems Design Using VHDL’, 3rd

[2] Edition, Thomson Asia private limited, Singapore.

[3] Bhaskar .J (2004) ‘A VHDL Primer’, 3rd edition, Pearson Education private limited, Singapore.

[4] Merrill I. Skolnik (2001) ‘Introduction to RADAR systems’, 3rd edition.

[5] Tata McGraw-Hill publishing company limited. Singapore. [5] Virtex-5 FPGA XAUI User Guide.

[6] ISE Design Suite 11.1 Release Notes and Installation Guide.

[7] SFPDP Draft standard vita 17.1-200x.

[8] Xilinx IP Cores User Guide.